Project Planning Algorithms:
Lowering Cost and Improving Delivery Time in Capital Projects

by

Arman Jabbari

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Industrial Engineering & Operations Research

in the

Graduate Division

of the

University of California, Berkeley

Spring 2020

Project Planning Algorithms:
Lowering Cost and Improving Delivery Time in Capital Projects

Abstract

Project Planning Algorithms:
Lowering Cost and Improving Delivery Time in Capital Projects

by

Arman Jabbari

Doctor of Philosophy in Engineering - Industrial Engineering & Operations Research

University of California, Berkeley

Professor Philip Kaminsky, Chair

With the goal of developing models and approaches leading to better operation of large-scale project delivery supply chains, we interviewed a variety of consultants and project and supply chain managers (with a particular emphasis on oil and gas major capital project delivery), and asked them a set of questions so that we could better understand current capital project delivery views of supply chain management, inventory, risk management tools, and related topics (Appendix A). Our interviewees expressed surprisingly diverse opinions, particularly regarding the future of mega-project delivery and the need for more closely coordinating supplier deliveries with onsite needs.

The work in the dissertation is particularly motivated by mega-projects in the oil and gas industry, and our goal is to build models that lead to better operation of capital project delivery supply chains. The characteristics of this industry, and of these projects, place specific requirements on project scheduling models, and many of the existing models in the literature do not meet these requirements. Our focus in this dissertation is to formulate models and develop approaches that are particularly useful for mega-projects in the oil and gas industry, and that enable the concurrent determination of the project schedule and inventory delivery times in order to efficiently manage the project supply chain, and to effectively control project delivery time and cost.

We consider the Stochastic Resource-Constrained Project Scheduling Problem with inventory, where the objective is to minimize a weighted combination of the expected project makespan and the expected inventory holding costs. Motivated by the requirements of major oil and gas industry projects, we introduce a class of proactive policies for the problem. We develop several effective heuristics for this problem, as well as deterministic and probabilistic lower bounds on the optimal solution. In computational testing, we demonstrate the effectiveness of these heuristics and develop insights into the value of explicitly considering inventory in this setting.

A related problem that arises is the scheduling of oil field drilling operations, where the goal is to maximize the expected revenue generated by oil extraction. For this problem, we build a model and propose a heuristic approach. We confirm the effectiveness of our heuristic approach by analyzing its performance compared to the current practice in a real-world case study. Our results demonstrate the potential to increase the efficiency and productivity of drilling operations significantly and to boost profitability by decreasing the time until wells start the extraction.

Through our interviews, and through analysis in subsequent models, it is clear that suppliers, and the timely delivery of supplies, plays a critical role in the successful implementation of large-scale projects. In the context of oil and gas projects, our focus is on the suppliers that provide customized materials. While the bulk of this dissertation focuses on projects from the project planner's point of view, we were also motivated to consider these problems from the perspective of the supplier, and hence, to consider scheduling models with due dates. We present a stylized model, where we consider sequencing decisions on a single processor, here representing a supplier, in an online setting where no data about the future incoming opportunities is available. With the goal of minimizing total weighted (modified) earliness and tardiness cost, we introduce a new scheduling policy, which we refer to as the list-based delayed shortest processing time policy, and develop lower and upper bounds on the performance of this policy.

Finally, we consider an alternative view of managing construction in projects, a location-based method known as the Work Density Method for takt planning. Given a work space and the number of zones in which to divide that space, the so-called WoLZo problem is to identify the shape and dimensions of each zone while minimizing the peak in the trades' workloads per zone. We model this problem and develop an optimization algorithm to divide a work space into zones while leveling work densities across trades in a process.

The tools presented in this dissertation are useful for managing different elements of mega-projects and significantly advance the state-of-the-art in those areas. We confirm the effectiveness of these tools by analyzing their performance compared to current practice in real-world case studies as well as their performance over the benchmark test problems that are available in the literature.

To my beloved family...

I am dedicating this dissertation to my loving parents, Hadi and Shahin, who helped me through all things great and small, and my sister Aylar who has never left my side.

Table of Contents

Table of Contents **ii**

List of Figures **iv**

List of Tables **vii**

1 Introduction **1**

2 Proactive Policies for Mega-Project Planning **6**
2.1 Introduction 6
2.2 Literature Review 9
2.3 The Models 13
2.4 Solution Approach 23
2.5 Computational Experiments 41

3 Scheduling of Oil Field Drilling Operations **62**
3.1 Problem Description 62
3.2 Mathematical Model Formulation 64
3.3 Solution Approach 67
3.4 Case Study 77

4 Online Scheduling Policies **83**
4.1 Introduction and Problem Description 83
4.2 Literature Review 84
4.3 DSPT, DWSPT, and LDWSPT Policies 86
4.4 Total Weighted (Modified) Earliness and Tardiness 87
4.5 Total Weighted (Modified) Unequal Earliness and Tardiness Cost 104
4.6 Future Research 108

5 Takt Planning - WoLZo Method **109**
5.1 Introduction 109
5.2 Work Density Method for Takt Planning 109
5.3 Literature Review 116

5.4 The Models . . . 117
5.5 Solution Method . . . 123
5.6 Application Example . . . 125
5.7 Discussion . . . 134

6 Conclusion **138**

References **140**

A Interviews **148**
A.1 Balancing Efficiency and Reliability . . . 148
A.2 Impact on Suppliers . . . 151
A.3 Decision-Making . . . 152

B Details of Running the WoLZo Algorithm **158**

C Optimal Partitions Suggested by WoLZo Algorithm **162**
C.1 Scenario One: $\mathbf{1 \times 1}$ matrix (1 cell) . . . 162
C.2 Scenario Two: $\mathbf{3 \times 3}$ matrix (9 cells) . . . 163
C.3 Scenario Three: $\mathbf{6 \times 9}$ matrix (54 cells) . . . 166
C.4 Scenario Four: $\mathbf{6 \times 18}$ matrix (108 cells) . . . 170

List of Figures

2.1 Summary of SAAD+. 28
2.2 Summary of NDH, EACH and PAH. 30
2.3 Illustration of the project that is introduced in Example 2. 31
2.4 Two optimal resource flows for the FCT. Different colors illustrate the flow of different resources. 32
2.5 Reducing a 3-partition problem to finding the resource flow that imposes the minimum number of effective arcs, where activity start times are fixed. 34
2.6 (a) Illustration of the project that is introduced in Example 3, and (b) the optimal solution of the deterministic version, where activity durations are assumed to be equal to their mean. 35
2.7 Two optimal resource flows for the FCT. Different colors illustrate the flow of different resources. 36
2.8 Summary of iNDH, iEACH and iPAH. 39
2.9 Summary of iSAAD. 40
2.10 Summary of iSAAD+. 40
2.11 Characteristics of the selected 282 instances. 43
2.12 Impact of NC, RF_r and RS_r on the best optimality gap for SFCT for instances with (a): 30 activities and (b): 60 activities. 46
2.13 Sensitivity analysis of the best optimality gap over NC, RF_r and RS_r for instances with 30 and 60 activities. 47
2.14 Illustration of the lower bound, the probabilistic lower bound, SAA solutions, SAAD+ solution, and NDH solution (indicator of current practice in real world). 50
2.15 SAAD+ solution, solutions of 30 replication of SAA and the best solution provided by the our deterministic heuristics. 52
2.16 Average optimality gap of SAAD+ for different number of replications. 53
2.17 Efficient frontier of expected makespan vs. expected inventory holding cost. . . 56
2.18 Efficient frontier of expected makespan vs. expected inventory holding cost. . . 60
2.19 Average expected inventory holding cost of iSAAD+ solution for different given expected makespan and different number of replications. 61

3.1 Details of the project that is introduced in Example 4. 64
3.2 Two alternative schedules for the project. 64
3.3 Decomposed Approach – decomposing the integrated model into three modules. 67

3.4 Illustration of the counterexample for Lemma 12 68
3.5 Illustration of a counterexample for Lemma 13 69
3.6 Partially Decomposed Approach – decomposing the integrated model into two modules. 75
3.7 Minimum total transportation time with respect to the forced upper bound on the expected average weighted finish times 78
3.8 Minimum expected inventory holding cost with respect to the forced upper bound on the expected average weighted finish times 79
3.9 Range of feasible upper bounds on the total transportation times and the maximum transportation time of the resources 79
3.10 Efficient frontier of the upper bound on the total transportation time and the expected average weighted finish times 80
3.11 Efficient frontier of the upper bound on the total transportation time vs. the expected average weighted finish times. Distinct colors indicate different values of upper bound on the maximum length of trips. 81
3.12 Efficient frontier of the upper bound on the total transportation time vs. the expected average weighted finish times. Distinct colors indicate different values of upper bound on the maximum transportation time of resources. 81
3.13 Percentage of decrease in the expected average weighted finish times per additional resource 82

4.1 Illustration of an instance for which the LDWSPT schedule consist of k blocks of jobs. 89
4.2 Illustration of an instance of three jobs with equal ratios of p_j/w_j, and its split version. 93
4.3 General formation of the LDWSPT sequence that contains l sub-blocks of jobs. 99

5.1 Process with 4 steps, scheduled when work space is divided respectively in 1, 2, or 3 zones 112
5.2 Process with 4 steps that repeats over 3 floors, scheduled when work space is divided in 4 zones 112
5.3 Work density for four trades and one zone 113
5.4 Work density for four trades and work space divided into three zones 115
5.5 Shapes of the zones: (a) irregular complex shapes and (b) simple shapes 118
5.6 Shapes with at most six facets, where all facets are perpendicular 118
5.7 Notation for rectangular and L-shaped zones 119
5.8 Notation tying work density to each cell 120
5.9 Solution procedure flowchart 124
5.10 Electrical work density based on a (1) 1×1 matrix, (2) 3×3 matrix, (3) 6×9 matrix, and (4) 6×18 matrix 126
5.11 Optimal partition suggested by the WoLZo algorithm based on (a) Model R and (b) Model L (NOTE: each zone has its own color) 127

5.12 Histogram of workloads by zone for each step in the process based on (a) Model R and (b) Model L (NOTE: the background color matches the color depicting the zone in Figure 5.11) . . . 127
5.13 Number of zones Z vs. takt $T(Z)$ in (a) Model R and (b) Model L . . . 128
5.14 Phase duration vs. number of zones for 1, 2, 5, and 10 floors in Model R . . . 131
5.15 Phase duration vs. number of zones for 1, 2, 5, and 10 floors in Model L . . . 132
5.16 Transforming (a) any work space to (b) a rectangular work space (hashed area has work density of 0 for all trades) . . . 135
5.17 Definition of size, height, and width of a zone . . . 136

List of Tables

2.1 Parameters that generate 48 different categories of instances in PSPLIB. 41
2.2 Optimality gap and total solution time of deterministic heuristics. 43
2.3 Average optimality gap and average solution time of deterministic heuristics for each category of instances with 30 activities. 44
2.4 Average optimality gap and average solution time of deterministic heuristics for each category of instances with 60 activities. 45
2.5 Optimality gap and total solution time of all proposed heuristics. 48
2.6 Average solution time of all proposed heuristics for each category of instances. . 49
2.7 Average optimality gap and median probabilistic optimality gap of deterministic heuristics for each category of instances. 51
2.8 Expected inventory holding cost of deterministic heuristics for instances with 30 activities. 54
2.9 Expected inventory holding cost of deterministic heuristics for instances with 60 activities. 54
2.10 Expected inventory holding cost of iEACH and iPAH for instances with 30 activities. 55
2.11 Expected inventory holding cost of iEACH and iPAH for instances with 60 activities. 55
2.12 Percentage of decrease in expected inventory holding cost when, given the fixed expected makespan, one heuristic replaces another one. 57
2.13 Impact of solving iSFCT($\mathcal{S}'$) rather than SFCT($\mathcal{S}'$) in the first stage of iSAAD. 58
2.14 Expected inventory holding cost and solution times of all proposed heuristics. . 59
2.15 Percentage of decrease in expected inventory holding cost when, given the fixed expected makespan, one heuristic replaces another one. 60

4.1 State of the art bounds on the optimal competitive ratio. 85
4.2 Values of v_j^1 and v_j^2 for each job. 101
4.3 Values of v_j^1 and v_j^2 for each job in the extended problem. 106

5.1 Number of zones Z vs. takt $T(Z)$ computed with WoLZo algorithm for Model R 129
5.2 Number of zones Z vs. takt $T(Z)$ computed with WoLZo algorithm for Model L 129
5.3 Average run-time of the algorithm in different scenarios when acceptable shapes are either only rectangles or rectangles as well as L-shapes, and different choices of *alpha* 133

5.4 Average number of iterations of the algorithm in different scenarios when acceptable shapes are either only rectangles or rectangles as well as L-shapes, and different choices of *alpha* 133

B.1 Results of WoLZo Computation for Model R, Scenario 1 158
B.2 Results of WoLZo Computation for Model L, Scenario 1 158
B.3 Results of WoLZo Computation for Model R, Scenario 2 159
B.4 Results of WoLZo Computation for Model L, Scenario 2 159
B.5 Results of WoLZo Computation for Model R, Scenario 3 160
B.6 Results of WoLZo Computation for Model L, Scenario 3 160
B.7 Results of WoLZo Computation for Model R, Scenario 4 161
B.8 Results of WoLZo Computation for Model L, Scenario 4 161

Acknowledgments

I am heartily thankful to my supervisor, Professor Philip Kaminsky, whose unique skill for identifying important and challenging research problems, emphasis on rigor and precision, attention to detail, sharp insights and deep knowledge have greatly helped me through my research. With his efforts, I was able to witness my transformation from an immature first-year graduate student to an independent researcher.

Besides my advisor, I would like to thank the rest of my qualifying exam and dissertation workshop committee members (Professor Rhonda Righter, Professor Zuo-Jun Max Shen, and Professor Iris D. Tommelein) for their great support and invaluable advice.

I am particularly grateful for the assistance given by Professor Iris D. Tommelein, who has helped me with her inspiring guidance, valuable discussions, and encouragement during our collaboration. I also want to sincerely thank Professor Candace A. Yano for her precious guidance on academic writing.

I am grateful to the Project Production Institute (PPI) for their financial support as this dissertation would not have been possible without their generous funding. The development of the WoLZo algorithm was supported in part by members of the Project Production Systems Laboratory (P2SL) at UC Berkeley and in part by the National Science Foundation (NSF) under grant number CMMI-1563511. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of members of PPI, P2SL or NSF.

Last, but not least, I wish to acknowledge the support and great love of my parents, Hadi and Shahin, and my sister, Aylar. They kept me going on and their love is always with me regardless of our distance.

Chapter 1

Introduction

Chen and Lee [13] suggest that the concern of most existing "operations management" literature is managing the flow of products for which there exist frequent demands. They referred such supply chains as "product supply chain management." The focus in product supply chain management is mostly on the end item production/demand, the level of inventory, and integration of supply chain stages. The primary trade-off in this area of research is inventory cost versus shortage cost.

In contrast, supply chain management is also essential for customized projects. Chen and Lee [13] call it "project supply chain management." In project supply chain management the focus is primarily on the schedule of activities and the release dates of materials, and the primary trade-off is the cost of earliness versus the cost of delay at the primary project site. Project supply chain management appears to be less-explored in the literature.

The primary reasons that it may be valuable to separately consider product supply chain management and project supply chain management have to do with different uncertainties and different model structures in these two categories. Empirical evidence regarding the performance of mega-projects suggests that investigation of project supply chains is important; however, the bulk of research in this area seems to be qualitative and descriptive, rather than prescriptive.

According to Flyvbjerg [26], *mega-projects* are "large-scale, complex ventures that typically cost $1 billion or more, take many years to develop and build, involve multiple public and private stakeholders, are transformational, and impact millions of people".

Merrow [55] investigates data from more than 300 global mega-projects and shows 65% of the mega-projects fails to meet their objective. He defines a mega-project as successful if it is delivered without exceeding cost and time estimates by more than 25%. He observes that cost overruns are frequently so significant that they often make the total project net present value (NPV) negative, and thus projects in this industry frequently need unexpected increases in demand or price in order to be profitable. In addition, he argues these results are not inherent to the nature of mega-projects; instead, they are caused by poor decision-making.

A study of 365 projects with a proposed capital investment of more than $1 billion (which thus count as mega-projects) was conducted by EY Research and Analysis [25] in the following industries: upstream (or exploration and production – E&P), LNG, pipelines, and refining. Their research shows that 64% of these projects faced cost overruns and 73% of the projects were delivered late. Also, these projects spent on average 59% more than their initial budget. They explain that these cost overruns traditionally were covered by the increase in oil and gas prices. However, this is no longer the case, and predictability is increasingly valued as companies need to be certain about the performance of their mega-projects.

Based on the research conducted by Changali, Mohammad, and Nieuwland [12] on mega-projects (in mining, oil and gas, and infrastructure industries), 98% of mega-projects incur cost overruns (of more than 30%) or delays. Moreover, the average cost increase is 80% of the original estimate, and 77% of mega-projects are at least 40% late. The authors observe that construction productivity has been flat for decades while in manufacturing it has nearly doubled in the same period. Their reasons for poor mega-project performance include: poor decision-making, inadequate communication, and insufficient risk management.

According to Flyvbjerg, Bruzelius, and Rothengatter [27], many mega-projects perform strikingly poorly. They described many cases of infrastructure mega-projects that experienced major cost overruns. The authors question whether or not the front-end analysis techniques such as cost-benefit analysis, financial analysis, etc. that are routinely used in mega-projects preparation are effective. They observe that "mega-project development is currently a field where little can be trusted at the time of determining the project initial viability, even – some would say especially not – numbers produced by analysts." In their opinion, the main causes of the cost and time overruns are insufficient risk analysis and the lack of accountability in the decision-making process.

The overall goal of this dissertation is to improve the operation of mega-project supply chains, and in the early stages of this project, to better understand the current state of these supply chains, we interviewed a variety of consultants and project and supply chain managers (with a particular emphasis on oil and gas mega-project delivery). We asked them a set of questions so that we could better understand current mega-projects delivery views of supply chain management, inventory management, risk management tools, and related topics. The details of our interviews with experts appear in Appendix A.

One common theme in these discussions is that there is often too much inventory that arrives too soon at the projects. This overwhelming amount of inventory makes it challenging for associates to retrieve what they want when they need it, even though the materials have been sitting in inventory on-site for months, or even years. In addition, the storage and handling are often haphazard, often leading to excessive movement that increase the likelihood of damaged, defective, or unsuitable materials. Therefore, massive amounts of inventory in mega-projects impacts financial metrics not only by tying up cash, but also by adding costs for the staging and preservation of materials, and in many cases, potential repair costs for damaged materials (Jabbari and Kaminsky [37]).

The presence of large amounts of inventory far before it is needed on mega-project sites is in dramatic contrast to many other industries, such as automobile manufacturing, retail, and high-tech manufacturing, where the timing of orders and deliveries is more closely coordinated with actual needs in order to better optimize overall system performance. We hypothesize that this is a symptom of a sub-optimal supply chain operation, a view that is further supported by McKinsey & Company [12], who finds that 98% of mega-projects are facing cost overruns of more than 30%, and 77% are at least 40% late.

Our interviews suggested that it is often the case that owner-operators look to minimize the risk of schedule delay due to late material and part delivery by mandating that materials be delivered far in advance of when they are needed, and that the delivery times of materials are not determined scientifically; instead, delivery times are set in an ad-hoc way, using industry rules of thumb.

Effective supply chain management strategy focuses on careful integration of the entire supply chain network. In contrast, the majority of mega-projects suffer from a lack of integration between project scheduling and other related decisions (specifically the delivery times of the orders). This is not to say that the challenge is unrecognized; interviewees highlighted the use of regular multi-disciplinary meetings with stakeholders from the project and supply chain functions aimed at improving coordination. However, even these regular meetings appear to be insufficient. Firms appear to lack a comprehensive model to optimize supply chain activities, and developing such a model is the goal of this dissertation research.

The work in this dissertation is particularly motivated by mega-projects in the oil and gas industry, and our goal is to build models that address these challenges. The characteristics of this industry, and of these projects, place specific requirements on project scheduling models, and many of the existing models in the literature do not meet these requirements.

In Chapter 2, our focus is on formulating models and developing approaches that are particularly useful for mega-projects in the oil and gas industry, and that enable the concurrent determination of project schedule and inventory delivery times in order to efficiently manage the project supply chain, and to effectively control project delivery time and cost. Specifically, in this chapter, we consider the Stochastic Resource-Constrained Project Scheduling Problem with inventory, where the objective is to minimize a weighted combination of expected project duration and expected inventory holding costs.

Motivated by the requirements of major oil and gas industry projects, we introduce the class of proactive policies. In order to provide reasonable solutions, we develop multiple deterministic and sample average approximation-based heuristic approaches. We introduce a lower bound to evaluate the performance of the proposed heuristics; we also propose a concept that we refer to as the probabilistic lower bound, which we define as an estimated lower bound along with the probability that it is in fact an actual lower bound, to develop a better understanding of the optimality gap.

Our results suggest that the advantages of our comprehensive model are two-fold. First, optimizing the stochastic version of the problem – instead of the simplified deterministic version – both decreases the expected makespan of the project, and also significantly lowers the expected inventory holding cost, even when the variability of activity durations is small. Second, determining the schedule and the suppliers' delivery times concurrently – instead of sequentially – significantly reduces the project cost. To demonstrate the inventory cost versus timeliness trade-off in the project deliveries, we also provide an efficient frontier as a managerial decision-making tool that sheds light on the relation between the cost and timeliness of the projects.

In Chapter 3, our focus is on related issue to the one explored in Chapter 2, but in the context of scheduling of oil field drilling operations, where the goal is to maximize the expected revenue generated by oil extraction. For this problem, we build a model and propose a heuristic approach for solving this model. We confirm the effectiveness of our heuristic approach by analyzing its performance compared to the current practice in a real-world case study. Finally, we explore the impact of changing the quantity of resources, and of constraining resource transportation time, on the profitability of projects. Our results demonstrate that advanced approaches to the scheduling of oil field drilling operations have the potential to increase the efficiency and productivity of drilling operations significantly and to boost profitability by decreasing the time until wells start the extraction.

Without any doubt, suppliers play a critical role in the success of large-scale projects. In the context of oil and gas projects, our focus is on suppliers that provide customized materials. While the bulk of this dissertation focuses on projects from the project planner's point of view, we were also motivated to consider these problems from the perspective of the supplier, and hence, to consider scheduling models with due dates. In Chapter 4, we present a stylized model, where we consider sequencing decisions on a single processor, here representing a supplier, in an online setting where no data about the future incoming opportunities is available. We formulate a single machine online scheduling problem where jobs with distinct processing times, weights, and due dates arrive over time and must be processed one at a time without preemption in order to minimize the total weighted earliness and tardiness cost. We introduce a new scheduling policy, the list-based delayed shortest processing time (LDWSPT) policy, which is amenable to theoretical analysis. We develop lower and upper bounds on the performance of the LDWSPT policy for the minimization of total weighted (modified) earliness and tardiness cost for the case of equal earliness and tardiness costs, and then extend our results for the case when these costs are not equal. Finally, we close the optimality gap that currently exists in the literature for several variants of single machine online scheduling problems in the presence of earliness and tardiness by proving that our proposed policy is an optimal online algorithm for these variants.

Construction management plays a key role in capital projects, and a variety of methods for production planning of construction work are used in practice. Many are location-based planning methods, meaning that they explicitly take into consideration where work is to be done by modeling space in 1, 2, or 3 dimensions. Such methods are known by a variety of names. Early examples are line of balance (LOB) (e.g., [83]), resource scheduling method (RSM) (e.g., [33]), Location Based Management System (LBMS) [41], Short Interval Production Scheduling (SIPS) [10], even-flow production (e.g., [6]), and space scheduling (e.g, [84], [72], [73]). More recent examples describe week-beat scheduling ([17]) and takt planning (e.g., [29], [28]). Some of these methods are more formally described in the literature than others. They tend to differ in the formulation of the objectives they aim to achieve. Objectives include aiming to complete the project within a set duration and achieving a steady flow of work for each of the specialty contractors (trades) involved. Some methods also aim to level workloads of trades that follow each other in a "Parade of Trades" [78].

Despite the use of such production planning methods in practice, the underlying mathematical algorithms are not necessarily well-articulated. This may be due to lack of definition of the construction management principles to be supported. It may also be due to the heuristic, opportunistic, and intuitive nature of the approach taken and the presumption that formalization would take a significant amount of effort.

Chapter 5 focuses on a location-based method used in construction, namely the Work Density Method (WDM) for takt planning. We present a novel problem formulation and an optimization algorithm to divide a work space into zones while leveling work densities across trades in a process. Given a work space and the number of zones in which to divide that space, the so-called WoLZo problem is to identify the shape and dimensions of each zone while minimizing the peak in the trades' workloads per zone. Increasing the number of zones tends to result in lowering the peak, but application of the WoLZo algorithm shows that this trend levels out due to the spatial distribution of work densities. Application to work density maps of different granularity sheds light on zoning work spaces and sharpens one's intuition on how many zones to select. These findings inform data collection pertaining to work density as needed when planning construction work on the basis of takt.

Finally, in Chapter 6, we conclude this dissertation and discuss the limitations of this work as well as the future research directions.

Chapter 2

Effective Proactive Policies for the Stochastic Resource-Constrained Project Scheduling Problem with Inventory Considerations

2.1 Introduction

We consider projects that consist of a set of individual tasks that must be completed before the project can be considered complete. In the project scheduling literature, these individual tasks are referred to as *activities*. Activities in a project are not independent from each other – they may have precedence constraints, which capture that notion that for technical reasons, some activities cannot be processed before the completion of predecessor activities.

Each activity require different quantity of different types of valuable primary renewable resources (e.g., a drilling rig), which we will subsequently call "resources." As these resources are typically costly, it is often the case that only a limited number of each type are available for a particular project. Therefore, allocating resources to activities is a key decision that impacts project completion times. Once an activity is completed, each of its resources must be transferred to the location of the next activity to which that resource is assigned. In addition, in many cases, these transportation times can be ignored as they are minimal compared to activity durations, but in some cases they must be explicitly considered. Particularly in oil and gas mega-projects, resources are bulky and need specific types of transportation, which may require building or improving roads. As building or improving roads take time, the sequence of activities visited by each resource must be determined at the beginning of the project.

Many research papers in the literature assumed the duration of activities is deterministic. Scheduling a project with deterministic activity durations typically involves determining the start and end time of each activity in order to minimize the makespan (project completion time) under resource and precedence constraints, and this problem is referred to as the *Deterministic Resource-constrained Project Scheduling Problem (RCPSP)* in the literature.

In reality, although there are engineering estimates regarding the duration of each activity, these estimates are not necessarily precise. Uncertainties can be a consequence of inherent uncertainty in the process, as well as as human error, weather conditions, and technical complications, etc. Some sources of uncertainty, e.g., weather conditions, impact multiple activities at a time. Therefore, assuming that activity durations are independent is not realistic. The problem of scheduling a project with stochastic activity durations to minimize the expected makespan under resource and precedence constraints is referred to as the *Stochastic Resource-constrained Project Scheduling Problem (SRCPSP)* in the literature. Many researchers (e.g., Ballestıén [4]) have observed that when the variability of activity durations is large, it is worthwhile to explicitly account for this uncertainty in modeling.

In a stochastic setting, it is not possible to specify start and end times for activities, as activity durations vary in each possible scenario. Indeed, as Hall [31] observes, the effect of these changes on the project completion time is not symmetrical. Activities which run longer than expected typically delay the project, whereas those that are shorter than expected fail to reduce the project completion time. Many experts believe that there are two reasons for this phenomena, namely: Parkinson's Law and Merge Bias. Parkinson's Law is a well-known principle states that "Work expands so as to fill the time available for its completion" (Parkinson and Osborn [62]). Merge Bias, as it is explained in Elfving and Tommelein [24], is about the impact of parallel activities that are precedent to another activity. If only one of the parallel activities completes early, the start time of the subsequent activity does not decrease; while if only one of them completes late, the start time of the subsequent activity is delayed as well.

One approach to the SRCPSP, given that the start and end time of activities cannot be determined at the beginning of a project, is to characterize a static policy, which is a set of rules that are applied dynamically as the project is executed. Since in the oil and gas mega-projects, the sequence of activities visited by each resource needs to be determined at the start of the horizon, our focus here is on static policies that proactively set the sequence of activities visited by each resource at the beginning of the project. In contrast, the start times of activities are determined dynamically during the execution of the project.

In addition to these resources, activities may require specific make-to-order materials (e.g., special pipe spools). For each activity, the engineering team decides the specifics of the required materials. It is assumed that all materials that are required for an activity are delivered together, that processing on an activity cannot start before this delivery, and that deliveries are made on-time. However, as the make-to-order materials have long lead-times, their delivery times also need to be determined at the start of the project. The inherent trade-off in this decision is that early delivery leads to inventory holding costs (as we have detailed above), and late delivery might delay the completion of the project.

We define the *Inventory Integrated Stochastic Resource-constrained Project Scheduling Problem (iSRCPSP)* as the problem of determining a schedule for activities, sequences of activities visited by each resource, and delivery times of materials, in order to minimize the expected total cost of the project under resource, precedence and materials constraints; where both the expected makespan and the expected inventory holding cost contribute to the cost of a project.

As with the SRCPSP, our goal here is to find a static policy that proactively specifies the sequence of activities visited by each resource and the delivery times of materials at the beginning of the project. In contrast, the start times of activities are determined dynamically during the execution of the project.

The rest of this chapter is organised as follows. In Section 2.2, we review literature related to project supply chain management in three categories: empirical evidence of the performance of mega-projects, mathematical formulations of the deterministic resource-constrained project scheduling, and stochastic resource-constrained project scheduling problem.

In Section 2.3, we develop the relevant mathematical optimization formulations. First, in Subsection 2.3.1, for the sake of completeness, we present the Flow-based Continuous Time (FCT) model, which is a mixed-integer linear programming (MILP) formulation for RCPSP. In Subsection 2.3.2, we extend the FCT formulation by extending the deterministic activity durations to the stochastic parameters. Then, we develop a formulation for the SRCPSP, which we refer to as the Stochastic Flow-based Continuous Time (SFCT) model, by introducing the scenario-based equivalent version of its stochastic formulation. Finally, in Subsection 2.3.3, we extend the SFCT formulation by integrating inventory related decisions and present a formulation for the iSRCPSP, which we refer to as the Inventory Integrated Stochastic Flow-based Continuous Time (iSFCT) model.

In Section 2.4, we discuss our solution approaches. In Subsection 2.4.1, we present an approach for finding a lower bound for both SRCPSP and iSRCPSP. In addition to the lower bound, we propose a "probabilistic lower bound", which we define as an estimated lower bound along with the probability that it is in fact an actual lower bound. The probabilistic lower bounds enhances our intuition about the tightness of both our lower and upper bounds. We also propose a variety of heuristics for these models, including sample average approximation-based approaches and deterministic approaches for SRCPSP and iSRCPSP, which we present in Subsections 2.4.2 and 2.4.3, respectively. Although all of these approaches result in upper bounds, they differ in terms of computation time, difficulty of implementation, and optimality gap. To select a particular approach for a project, one must trade off level of sophistication and available computing power.

Finally, in Section 2.5, we compare the computational performance of our heuristics on both SRCPSP and iSRCPSP. We show that even when the variability of activity durations is small, explicitly accounting for the uncertainty in the model results in a moderate decrease in the expected makespan and a significant decrease in the expected inventory holding cost. Lastly, we confirm that determining the schedule and the delivery times of orders concurrently helps to reduce the total cost of projects.

2.2 Literature Review

In this chapter, we present a novel problem, in which the inventory holding cost is considered along with the project schedule, that we refer to as the iSRCPSP. To the best of our knowledge, this problem does not exist in the literature. This problem is in some respects similar to the category of problems with the goal of minimizing the total deviation from the due date or baseline schedule of activities (e.g., Khalilzadeh, Kianfar, and Ranjbar [42] and Brčić and Mlinarić [8]). However, they are different as the iSRCPSP minimizes the expected makespan of the project along with the inventory holding cost, and activities do not have due dates or baseline schedule.

Here, our primary focus is on mega-projects in the oil and gas industry. Mega-projects are, as the name implies, very large projects. More specifically, a mega-project is defined by Flyvbjerg [26] as any project with a total capital cost of more than $1 billion. In Chapter 1, we explore some empirical research on mega-projects. Although the focus of the literature is on scheduling, and these models typically do not consider other operational decisions, they can often be viewed as a key components for creating an inventory integrated stochastic optimization model that capture additional relevant details. Lastly, in Subsection 2.2.2, we discuss different classes of policies for stochastic project scheduling problems.

2.2.1 Deterministic Resource-Constrained Project Scheduling Problem

Garey and Johnson [30] proved that the decision version of RCPSP is NP-complete in the strong sense by reduction from the 3-partition problem. Thus, RCPSP is NP-hard in the strong sense. There are several different Mixed Integer Linear Programming (MILP) formulations in the literature for the RCPSP. Two important types of models are time-indexed and compact models.

Time-indexed models divide the time horizon into discrete equal time intervals and assign the start time of each activity to one of those time intervals. In these models, the resource constraint is applied by enforcing the total resource consumption at each time interval to be less than or equal to the total available resources in the project. The size of these models increases based on the number of time intervals in the time horizon. The most common time-indexed model is called the discrete-time (DT) model and is presented by Pritsker and Watters [64]. Christofides, Alvarez-Valdes, and Tamarit [16] presented an extension of discrete-time model called the disaggregated discrete-time (DDT) model and proved that DDT model leads to a stronger linear relaxation than the DT model.

Artigues, Michelon, and Reusser [2] suggested a compact model known as the Flow-based Continuous Time (FCT) model in which the start time of each activity is a continuous variable and the resources are assumed to move from one activity to another (similar to flow problems). In this model, the resource constraint is applied by ensuring that resources have a feasible flow among the activities. The FCT formulation yields a poor relaxation.

However, Koné et al. [49] argued that FCT can be preferable to DT and DDT for solving large time-horizon problem instances.

Koné et al. [49] introduced a compact model which is called the On/Off Event-based (OOE) model. The OOE model partitions the time horizon into a set of disjoint unequal intervals, where the start time of each interval is a continuous variable. In each interval, each of the activities that are being processed are called ‘active’. Each activity must be active in one or multiple consecutive intervals, and the start time of each activity is defined to be the start time of the first interval in which it is active. In this model, the resource constraint is applied by ensuring that in each interval, the total resource consumption from all active activities is less than or equal to the total available resources in the project. Tesch [71] replaced some of the constraints in OOE model with new constraints that dominate the former ones, and showed that the LP-relaxation of the OOE is very weak.

Koné et al. [49] proposed another compact model that is similar to OOE except that instead of determining all intervals in which an activity is active, just the first and the last intervals in which an activity is active are determined. This model is called the Start/End Event-based (SEE) model. Tesch [71] replaced some of the constraints in the SEE model with new constraints that dominate the former ones, and showed that although the LP-relaxation of the SEE is stronger than the LP-relaxation of the OOE, it is still very weak. Tesch [71] offers more details on MILP formulations of the RCPSP.

The RCPSP has been analyzed for over 50 years and numerous computationally effective solution approaches have been developed.

The exact methods that are applied to the RCPSP include: zero-one programming (Icmeli and Rom [35]), dynamic programming (Petrović [63]), branch-and-bound (Brucker et al. [9], Morillo-Torres, Moreno-Velásquez, and Díaz-Serna [58], and Dorndorf, Pesch, and Phan-Huy [22]), and and branch and cut (Chakrabortty, Sarker, and Essam [11]). Liess and Michelon [52] transformed the RCPSP into a satisfiability problem, and showed promising results using a (SAT) solver.

Many authors, including Kolisch and Hartmann [47], Laborie [50], Demassey, Artigues, and Michelon [20] and Baptiste and Demassey [5] proposed constraint programming approaches. Laborie [50] recently was able to find the optimal solution of some of the unsolved instances of the PSPLIB benchmark. He also reported this approach optimally solves 100.00%, 88.33% and 46.00% of the projects in the PSPLIB benchmark with 30, 60 and 120 activities, respectively.

Other heuristic procedures have also received extensive attention in the literature. Priority rules (Valls, Ballestıén, and Quintanilla [80] and Klein [44]), insertion techniques (Artigues, Michelon, and Reusser [2]), and Lagrangean heuristics (Möhring et al. [57]) have all been considered. Kolisch and Hartmann [47] offers a survey of these heuristic approaches. Genetic algorithms (GA), tabu search (TS), simulated annealing (SA), particle swarm optimization (PSO) and ant colony optimization (ACO) are the most common metaheuristics used to solve RCPSP. *Handbook on Project Management and Scheduling Vol.1* [32] reported that applying state of the art metaheuristics resulting in an average optimality gap of 0.10%, 11.10% and 34.15%, respectively, on projects in the PSPLIB benchmark with 30, 60 and 120 activities.

See Demeulemeester and Herroelen [21], Neumann, Schwindt, and Zimmermann [59] and Kolisch and Padman [46] for surveys of solution methods for RCPSP.

2.2.2 Stochastic Resource-Constrained Project Scheduling Problem

In the literature, two approaches stand out as the primary focus of research on dealing with uncertainty of activity durations in the SRCPSP: proactive and reactive scheduling, and scheduling policies.

Proactive and reactive scheduling is a two-stage process. In the first stage, a baseline schedule is constructed that is designed to be as robust as possible, and then, in the second stage, if necessary, the schedule is revised or re-optimized after the realization of the duration of each activity. For more information about proactive and reactive scheduling, see Van de Vonder et al. [81] and Davari and Demeulemeester [19].

Scheduling policies do not involve a baseline schedule. Instead, the schedule is determined dynamically as time progresses and activity durations are realized. Our focus in this chapter is on a class of policies in which decisions are only made at the start of the time horizon and at the completion of each activity (decision points). In the literature, this class of policies is referred to as the elementary policies. Rostami [67] shows that the set of all elementary policies does not necessarily include the optimal policy. However, as non-elementary policies are computationally very challenging, they are rarely considered in the literature.

To the best of our knowledge, Creemers [18] is the only work that proposed an exact algorithm for finding an optimal elementary policy. He modeled the SRCPSP as a Markov decision process and assumed that activity durations are independent from each other and a have phase-type distributions. He concluded that this approach is successful for small- to medium-sized problems in which the coefficient of variation of activity durations is at least 0.7. Most authors focus on finding the optimal policy within a predefined subclass of elementary policies. In the following, we review the most common subclasses of elementary policies:

1. Resource-based policies (RB-policies):
   RB-policies (Ashtiani, Leus, and Aryanezhad [3]) take a list of all activities as an input. At each decision point, RB-policies go through all activities one at a time in the order of the list, and for each activity, if processing it does not create a resource conflict, its processing starts and it is removed from the list.

2. Activity-based policies (AB-policies):
   AB-policies (Ballestıén [4]) take a list of all activities as an input and at each decision point, remove the top k activities from the list and start processing them, where k, at each decision point, is the largest number of activities that does not result in a resource conflict.

3. Earliest-start policies (ES-policies):
   ES-policies (Radermacher [66]) find all sets of activities that cannot be processed simultaneously due to resource constraints and add a precedence constraint between at least a pair of activities in each set in order to remove all possible resource conflicts. Then, every activity start as soon as possible given the set of precedence constraints.

4. Preselective policies (PS-policies):
   PS-policies (Igelmund and Radermacher [36]) find all sets of activities that cannot be processed simultaneously due to resource constraints. Then, one activity in each set is selected and its start time is forced to be after at least one of the other activities in that set is completed. Every activity starts as soon as possible.

5. Preprocessor policies (PP-policies):
   PP-policies (Ashtiani, Leus, and Aryanezhad [3]) take a set of pairs of activities, E', and a list of all activities, L, as input. Then, a precedence constraint is added between activities of any pair in E', and finally, the altered project is scheduled by an RB-policy over the list L. Note that the precedence constraints that are added to the project do not necessarily remove all possible resource conflicts.

6. Generalized preprocessor policies (GP-policies):
   GP-policies (Ashtiani, Leus, and Aryanezhad [3]) are generalization of PP-policies. They take two sets of pairs of activities, E^{FS} and E^{SS}, and a list of all activities, L, as an input. Then, a precedence constraint is added between activities of any pair in E^{FS}. Note that a precedence constraint from activity i to activity j prevents activity j from starting before the completion of activity i. SS-constraints are a similar concept to precedence constraints, except that when there is an SS-constraint from activity i to activity j, it prevents activity j from starting before the start of activity i. GP-policies add an SS-constraint between activities of any pair in E^{SS}. Finally, the altered project is scheduled by an RB-policy over the list L. Note that the precedence constraints and the SS-constraints that are added to the project do not necessarily remove all possible resource conflicts.

For more information on the different classes of scheduling policies, see Chen et al. [15].

Motivated by the requirement for scheduling mega-projects in oil and gas industry, our goal in this chapter is develop an effective approach to resource constrained project scheduling that determines, prior to the start of the project, the sequence of activities visited by each resource as well as the delivery times for all materials. Therefore, our focus in this chapter is on a subclass of elementary policies that makes these decisions before the execution of the project, and then starts as soon as possible given these decisions. We refer to this class of policies as "proactive policies".

For SRCPSP, the set of all proactive policies is equivalent to the set of all ES-policies, as in any ES-policy, without impacting the expected makespan of the project, we can fix the sequence of activities visited by each resource at the beginning of the horizon. Note that ES-policies are specific to SRCPSP, while proactive policies can be applied to many variations of resource-constrained project scheduling problems, including iSRCPSP.

Our goal in this chapter is to develop heuristics to find effective policies within the class of proactive policies for both SRCPSP and iSRCPSP. In what follows, the *optimal solution* of the SRCPSP refers to the policy within this class that results in the minimum expected makespan, and the *optimal solution* of the iSRCPSP refers to the policy within this class that results in the minimum expected cost. We are interested in heuristic approaches that allow for dependencies between activity durations, and apply to instances of the size we observed when talking to industry experts, projects with 30-60 activities. As far as we know, none of the approaches in the literature meet these requirements.

2.3 The Models

In this section, we develop the relevant mathematical models. In all of our models, a project and its precedence constraints are represented by a directed acyclic graph $G(\mathcal{A}, \mathcal{E})$, where $\mathcal{A}$ is the set of nodes and $\mathcal{E}$ is the set of arcs that represent the precedence constraints. Each node in $\mathcal{A} = \{0, 1, ..., n+1\}$ represents an activity. Activities 0 and $n+1$ are dummy zero-duration activities that indicate the start and end of the project. All activities must be executed without preemption. $\mathcal{R}$ is the set of resource types and D_r units of resource type r for $r \in \mathcal{R}$ are available to be utilized by the project. The processing on activity i requires $d_{i,r}$ units of resource type r for $i \in \mathcal{A}$ and $r \in \mathcal{R}$. For dummy nodes, $d_{0,r} = d_{n+1,r} = D_r$ for $r \in \mathcal{R}$.

In Subsection 2.3.1, we begin by presenting a relevant formulation of the deterministic RCPSP, the mixed-integer linear programming (MILP) Flow-based Continuous Time (FCT) formulation. In the RCPSP, activity durations are deterministic, and we denote these deterministic activity durations by p_i for $i \in \mathcal{A}$.

In Subsection 2.3.2, we develop a MILP formulation to find the optimal solution of the SRCPSP within the class of proactive policies. In our formulations, we assumed activity durations are stochastic with known distributions, and we represent these stochastic durations by random variables P_i for $i \in \mathcal{A}$. The dummy activities have deterministic activity durations of zero; all other durations have no restrictions on distributions or dependencies.

Finally, in Subsection 2.3.3, we develop a MILP formulation to find the optimal solution of the iSRCPSP within the class of proactive policies.

2.3.1 Flow-based Continuous Time (FCT) Formulation of the RCPSP

The FCT formulation (Artigues, Michelon, and Reusser [2]) is a MILP formulation that focuses on the flow of resources rather than the start time of activities. In this formulation, the processing of an activity starts after all required resources have moved to that activity. Also, each resource that moves to an activity stays there until the completion of processing of that activity. The flow of resources is represented by a continuous variable, $f_{i,j,r}$ for $i, j \in \mathcal{A}$ and $r \in \mathcal{R}$, which represents the quantity of type r resource that moves to activity j from activity i (after i has completed processing).

In Lemma 1, we prove that when the flow of resources, $f_{i,j,r}$ for $i, j \in \mathcal{A}$ and $r \in \mathcal{R}$, has only integer values, we can construct the sequence of activities visited by each resource in pseudo-polynomial time. Thus, because the FCT formulation determines the flow of resources, it is well-suited for our setting in which we must determine the sequence of activities visited by each resource at the beginning of the horizon. Therefore, we focus on FCT as a starting point for developing our formulations.

Lemma 1. *When the flow of resources ($f_{i,j,r}$ for $i, j \in \mathcal{A}$ and $r \in \mathcal{R}$) takes on integer values, the sequence of activities visited by each resource can be constructed in pseudo-polynomial time.*

Proof. Let Υ represent the set of all resources, and $[\tau]$ represent the type of resource $\tau \in \Upsilon$. Consider any integer feasible flow of resources, $f_{i,j,r}$ for $i, j \in \mathcal{A}$ and $r \in \mathcal{R}$. The sequences of activities visited by each resources can be constructed by the following procedure:

```
Create an empty list, $L_\tau$ for each resource $\tau \in \Upsilon$.
 Step 1:
  Select a resource (e.g., resource $\tau$) from $\Upsilon$ and remove it from $\Upsilon$.
  Add activity 0 to the $L_\tau$.
 Step 2:
  Find $j : f_{L_\tau[-1],j,[\tau]} > 0$, where $L_\tau[-1]$ is the last activity that is added to $L_\tau$.
  Add activity $j$ to $L_\tau$.
  Update $f_{L_\tau[-1],j,[\tau]} := f_{L_\tau[-1],j,[\tau]} - 1$.
 Step 3:
  If $j \neq n+1$, go to *step 2*; otherwise, if $\Upsilon$ is not empty, go to *step 1*.
```

The running time of this procedure is $O(|\Upsilon| \times |\mathcal{A}|)$, which is a pseudo-polynomial function of the problem size. Note that in practice, as the total number of resources is bounded, the run-time of this procedure is a polynomial function of the problem size ($O(|\mathcal{A}|)$).

□

The FCT formulation determines the flow of resources in order to minimize the makespan of the project. It captures precedence and resource constraints, but not uncertainty in durations, delivery times, or inventory.

We present the FCT formulation below. In this formulation, the binary variable $x_{i,j}$ for $i, j \in \mathcal{A}$ equals zero if activity i does not complete before activity j starts, it equals one if any resource moves from activity i to activity j, and it can equal either zero or one otherwise. The variable z_i for $i \in \mathcal{A}$ equals the start time of activity i, and lastly, constants M and N are very large numbers (a.k.a. big-M).

$$\text{(FCT)} \quad \mathit{Minimize} \quad z_{n+1} \tag{2.1}$$

$$x_{i,j} = 1 \qquad \forall\, (i,j) \in \mathcal{E} \tag{2.2}$$

$$f_{i,j,r} - \min(d_{i,r}, d_{j,r})\, x_{i,j} \leq 0 \qquad \forall\, i,j \in \mathcal{A}, r \in \mathcal{R} \tag{2.3}$$

$$z_j - z_i \geq p_i - M(1 - x_{i,j}) \qquad \forall\, i,j \in \mathcal{A} \tag{2.4}$$

$$\sum_{j \in \mathcal{A}} f_{i,j,r} = d_{i,r} \qquad \forall\, i \in \mathcal{A} - \{n+1\}, r \in \mathcal{R} \tag{2.5}$$

$$\sum_{i \in \mathcal{A}} f_{i,j,r} = d_{j,r} \qquad \forall\, j \in \mathcal{A} - \{0\}, r \in \mathcal{R} \tag{2.6}$$

$$f_{i,j,r} \geq 0 \qquad \forall\, i,j \in \mathcal{A}, r \in \mathcal{R} \tag{2.7}$$

$$z_i \geq 0 \qquad \forall\, i \in \mathcal{A} \tag{2.8}$$

$$x_{i,j} \in \{0,1\} \qquad \forall\, i,j \in \mathcal{A} \tag{2.9}$$

The objective function (Equation 2.1) minimizes the makespan, which is equal to the start time of the dummy activity that represents the completion of the project. Constraints 2.2 and 2.3 enforce binary variables $x_{i,j}$ for $i,j \in \mathcal{A}$ to equal one if the processing of activity i must be completed before the processing of activity j starts, or if any resource moves from activity i to activity j. Constraint 2.4 ensures that activity $j \in \mathcal{A}$ starts after the completion of activity $i \in \mathcal{A}$ when variable $x_{i,j}$ is one. Constraints 2.5 and 2.6 ensure that for each type of resource, the quantity of resources that move to each activity equals the quantity of resources that move from that activity and equals the quantity of resources that the activity requires in order to be processed.

In most cases, the resources are discrete and indivisible, and this is assumed for the rest of this chapter. In this case, we can substitute Constraint 2.10 for Constraint 2.7.

$$f_{i,j,r} \in \{0,1,2,...\} \qquad \forall\, i,j \in \mathcal{A}, r \in \mathcal{R} \tag{2.10}$$

In Lemma 2, we prove that when the resources are discrete and indivisible, for any given feasible solution for the FCT formulation in which the flow of resources has non-integer values, in polynomial time, we can construct another feasible solution with the same objective function value in which the flow of resources has only integer values. Thus, we conclude that substituting Constraint 2.10 for Constraint 2.7 does not change the objective function value.

Lemma 2. *If D_r and $d_{i,r}$ for all $r \in \mathcal{R}$ and $i \in \mathcal{A}$ take integer values, there exists an optimal solution in which $f_{i,j,r}$ for $i, j \in \mathcal{A}$ and $r \in \mathcal{R}$ has only integer values.*

Proof. Consider any feasible solution for the FCT formulation and denote its flow of resources $f'_{i,j,r}$ for $i, j \in \mathcal{A}$ and $r \in \mathcal{R}$. For resource type $r' \in \mathcal{R}$, create graph $G'_{r'}(\mathcal{A}', \mathcal{E}')$ as follows:

> **Nodes ($\boldsymbol{A'}$):** for each activity $i \in \mathcal{A}$, create two nodes and let $a(i)$ and $b(i)$ represent them.
>
> **Arcs ($\boldsymbol{E'}$):** for each $i \in \mathcal{A}$, add an arc from $a(i)$ to $b(i)$ with lower bound of $d_{i,r'}$ and upper bound of $d_{i,r'}$. Note that $d_{0,r'} = d_{n+1,r'} = D_{r'}$. Also, for each $i, j \in \mathcal{A}$, if $x_{i,j} = 1$, add an arc between $b(i)$ and $a(j)$ with lower bound of zero and upper bound of infinity, otherwise, add an arc between $b(i)$ and $a(j)$ with lower bound of zero and upper bound of zero. Finally, add an arc from node $b(n+1)$ to node $a(0)$ with the lower bound of zero and upper bound of infinity.

We can construct a feasible network flow ($\hat{f}_{i,j}$ for $(i, j) \in \mathcal{A}'$) for $G'_{r'}(\mathcal{A}', \mathcal{E}')$ as follows:

$$\hat{f}_{a(i),b(i)} = d_{i,r'} \qquad \forall\, i \in \mathcal{A}$$

$$\hat{f}_{b(i),a(j)} = f'_{i,j,r'} \qquad \forall\, i, j \in \mathcal{A}, \text{ if } i \neq j, (i, j) \neq (n+1, 0)$$

$$\hat{f}_{b(n+1),a(0)} = D_{r'}$$

As the formulation of the network flow problem is totally unimodular, and there exists a feasible network flow for $G'_{r'}(\mathcal{A}', \mathcal{E}')$, there also exist an integer feasible network flow and it can be found in polynomial time. Represent that integer feasible network flow by $\hat{f}'_{i,j}$ for $(i, j) \in \mathcal{A}'$. We create a new feasible solution for the FCT formulation by substituting $f'_{i,j,r}$ with $f''_{i,j,r}$ for $i, j \in \mathcal{A}$ and $r \in \mathcal{R}$, where:

$$f''_{i,j,r} = f'_{i,j,r} \qquad \forall\, i, j \in \mathcal{A}, r \in \mathcal{R} - \{r'\}$$

$$f''_{i,j,r'} = \hat{f}'_{b(i),a(j)} \qquad \forall\, i, j \in \mathcal{A}, \text{ if } i \neq j, (i, j) \neq (n+1, 0)$$

$$f''_{i,i,r'} = 0 \qquad \forall\, i \in \mathcal{A}$$

$$f''_{n+1,0,r'} = 0$$

As activity start times in the new feasible solution are the same as activity start times in the former feasible solution, the objective function value of both feasible solutions are the same. Also, $f''_{i,j,r'}$ for $i, j \in \mathcal{A}$ takes on only integer values.

By repeating this procedure for all resource types, we can construct a feasible solution with the same objective function value in which the flow of resources has only integer values.

Thus, we conclude that there exists an optimal solution in which $f_{i,j,r}$ for $i, j \in \mathcal{A}$ and $r \in \mathcal{R}$ has only integer values. $\square$

2.3.2 Stochastic Flow-based Continuous Time (SFCT) Model

In the RCPSP, we assume that activity durations are deterministic. However, in most projects, activity durations are highly uncertain. Hence, we modify the RCPSP to allow random processing times with known distributions, resulting in the SRCPSP. The dummy activities have deterministic activity durations of zero; all other durations have no restrictions on distributions or dependencies. We model the problem of finding the optimal solution of SRCPSP within the class of proactive policies as an optimization problem, and we refer to it as the Stochastic Flow-based Continuous Time (SFCT) model:

$$\text{SFCT}: \quad \min_{\vec{f}} E_{\vec{P}}\Big[F(\vec{P}, \vec{f})\Big], \tag{2.11}$$

where

$\vec{f}$: Vector of flow of resources ($f_{i,j,r}$ for $i, j \in \mathcal{A}$ and $r \in \mathcal{R}$);
$\vec{P}$: Vector of activity durations, where each element has an associated probability distribution (P_i for $i \in \mathcal{A}$);
$F(.)$: A function that provides the stochastic makespan of the project.

Note that this model identifies the flow of resources, from which, by Lemma 1, we can identify the sequence of activities visited by each resource.

The expected value component in Model 2.11 is a high-dimensional integral. Define a "scenario" as a realization of the stochastic activity durations. Intuitively, a scenario can be interpreted as a plausible realizations of the stochastic durations. By using the Sample Average Approximation (SAA) technique (as in Kleywegt, Shapiro, and Homem-de-Mello [45]), the SFCT (Model 2.11) can be approximated by SFCT($\mathcal{S}$):

$$\text{SFCT}(\mathcal{S}): \quad \min_{\vec{f}} \sum_{s \in \mathcal{S}} \hat{F}(\vec{p}_s, \vec{f})/|\mathcal{S}|, \tag{2.12}$$

where

$\mathcal{S}$: Set of scenarios;
$|.|$: cardinality of the set;
$\vec{p}_s$: Vector of activity durations in scenario $s \in \mathcal{S}$;
$\hat{F}(.)$: A function that provides the makespan of the project given the flow of resources, $\vec{f}$, and the realized durations, $\vec{p}_s$.

Let Z^*_{SFCT} and $Z^{\mathcal{S}}_{SFCT}$, respectively, denote the optimal objective function values of SFCT and SFCT($\mathcal{S}$). Shapiro and Philpott [68] show that when i.i.d. scenarios are randomly selected (e.g., Monte Carlo simulation), the sample average converges to the corresponding expectation at a rate of $O(|\mathcal{S}|^{-0.5})$. In other words, the *error*, denoted $|Z^*_{SFCT} - Z^{\mathcal{S}}_{SFCT}|$, is bounded by $O(|\mathcal{S}|^{-0.5})$. Therefore, the number of scenarios selected in practice should be sufficiently large to reduce the *error* to an acceptable threshold. In Lemma 3, we show that the optimal objective function value of SFCT($\mathcal{S}$) converges to the optimal objective function value of SFCT as $|\mathcal{S}|$ goes to infinity.

Lemma 3. *When $|\mathcal{S}| \to \infty$, the optimal objective function value of SFCT($\mathcal{S}$) converges to the optimal objective function value of SFCT with probability one.*

Proof. Let $\mathcal{F}$ denote the set of all feasible flow of resources. For any $\vec{f} \in \mathcal{F}$, based on the law of large numbers:

$$\sum_{s \in \mathcal{S}} \hat{F}(\vec{p}_s, \vec{f})/|\mathcal{S}| \to E_{\vec{P}}[F(\vec{P}, \vec{f})],\ w.p.1 \text{ as } |\mathcal{S}| \to \infty,$$

and since $|\mathcal{F}|$ is finite, we conclude that:

$$Z^{\mathcal{S}}_{SFCT} \to Z^{*}_{SFCT},\ w.p.1 \text{ as } |\mathcal{S}| \to \infty.$$

□

In Lemma 4, we show that when activity durations are mutually independent and have light-tailed distributions (any distribution where the tail is lighter than an exponential function), the probability of converging to an ϵ-optimal solution approaches one at an exponential rate, where an ϵ-optimal solution is a solution in which the objective function value is at most ϵ away from the optimal objective function value.

Lemma 4. *Under mild regularity conditions, given activity durations are mutually independent and have light tailed distributions, the probability of converging to an ϵ-optimal solution approaches one exponentially fast by increasing the sample size.*

Proof. Kleywegt, Shapiro, and Homem-de-Mello [45] prove in SAA, under mild regularity conditions, the probability of converging to an ϵ-optimal solution approaches one exponentially fast by increasing the sample size when the following assumption holds:

> *For every feasible solution, $\vec{f}$, the moment-generating function of the random variable $H(\vec{P}, \vec{f})$ is finite valued in a neighborhood of 0,*

where

$$H(\vec{p}_s, \vec{f}) := \hat{F}(\vec{p}_s, u(\vec{f})) - \hat{F}(\vec{p}_s, \vec{f}) \quad \forall\, s \in \mathcal{S},$$

and the mapping $u(\vec{f})$ returns $\vec{f}$ when $\vec{f}$ is ϵ-optimal, and as follows when $\vec{f}$ is not ϵ-optimal:

$$E[F(\vec{P}, u(\vec{f}))] \leq E[F(\vec{P}, \vec{f})] - \epsilon.$$

Let $\vec{f'}$ represent a feasible resource flow that enforces activities to be processed sequentially (one at a time and without any idle time). For any scenario $s \in \mathcal{S}$:

$$|H(\vec{f},\vec{p}_s)| := |\hat{F}(\vec{p}_s,u(\vec{f})) - \hat{F}(\vec{p}_s,\vec{f})| \leq \max\left(\hat{F}(\vec{p}_s,u(\vec{f})),\hat{F}(\vec{p}_s,\vec{f})\right) \leq \hat{F}(\vec{p}_s,\vec{f}') = \sum_{j\in\mathcal{A}} p_j^s,$$

and therefore:

$$\begin{aligned} M(t) = E\Big[e^{t.H(\vec{f},\vec{P})}\Big] &\leq E\Big[e^{t.|H(\vec{f},\vec{P})|}\Big] \\ &\leq E\Big[e^{t.\sum_{j\in\mathcal{A}} P_j}\Big] = \int \cdots \int_{P_j} e^{t.\sum_{j\in\mathcal{A}} P_j} f(P_1,\ldots,P_n)\, dP_1 \ldots dP_n, \end{aligned}$$

where n is the number of activities ($|\mathcal{A}|$) and $f(P_1,\ldots,P_n)$ is the joint distribution of activity durations. Finally, as activity durations are mutually independent:

$$M(t) \leq \int_{P_1} e^{t.P_1} f(P_1)\, dP_1 \times \ldots \times \int_{P_n} e^{t.P_n} f(P_n)\, dP_n,$$

where $f(P_i)$ denote the probability distribution of random variable P_i. For any distribution with a tail lighter than the tail of an exponential distribution, the value of $\int_{P_i} e^{t.P_i} f(P_i)\, dP_i$ is finite when t is in the neighborhood of zero. As multiplying a finite number of finite values remains finite, we conclude that $M(t)$ is finite in the neighborhood of zero. □

Next, we present a MILP formulation of SFCT($\mathcal{S}$) that determines the flow of resources in order to minimize the expected makespan over the scenarios in $\mathcal{S}$, while capturing resource and precedence constraints. The parameter p_i^s for $i\in\mathcal{A}$ and $s\in\mathcal{S}$ represents the duration of activity i in scenario s, variable z_i^s for $i\in\mathcal{A}$ and $s\in\mathcal{S}$ represents the start time of activity i in scenario s, and the remaining notation is the same as the FCT formulation presented in Subsection 2.3.1.

$$\Big(\text{SFCT}(\mathcal{S})\Big) \quad \textit{Minimize} \quad E(z_{n+1}^s) = \sum_{s\in\mathcal{S}} z_{n+1}^s/|\mathcal{S}| \tag{2.13}$$

$$x_{i,j} = 1 \qquad \forall\,(i,j)\in\mathcal{E} \tag{2.14}$$

$$f_{i,j,r} - \min(d_{i,r},d_{j,r})x_{i,j} \leq 0 \qquad \forall\, i,j\in\mathcal{A}, r\in\mathcal{R} \tag{2.15}$$

$$z_j^s - z_i^s \geq p_i^s - M(1-x_{i,j}) \qquad \forall\, i,j\in\mathcal{A}, s\in\mathcal{S} \tag{2.16}$$

$$\sum_{j\in\mathcal{A}} f_{i,j,r} = d_{i,r} \qquad \forall\, i\in\mathcal{A}-\{n+1\}, r\in\mathcal{R} \tag{2.17}$$

$$\sum_{i\in\mathcal{A}} f_{i,j,r} = d_{j,r} \qquad \forall\, j\in\mathcal{A}-\{0\}, r\in\mathcal{R} \tag{2.18}$$

$$f_{i,j,r} \geq 0 \qquad \forall\, i,j \in \mathcal{A}, r \in \mathcal{R} \tag{2.19}$$

$$z_i^s \geq 0 \qquad \forall\, i \in \mathcal{A}, s \in \mathcal{S} \tag{2.20}$$

$$x_{i,j} \in \{0,1\} \qquad \forall\, i,j \in \mathcal{A} \tag{2.21}$$

The objective function (Equation 2.13) minimizes the expected makespan of the project. Constraints 2.14 and 2.15 force binary variable $x_{i,j}$ for $i,j \in \mathcal{A}$ to equal one if activity i must be completed before activity j can start, or if any resource moves from activity i to activity j. Constraint 2.16 ensures that in all scenarios, activity $j \in \mathcal{A}$ starts after the completion of activity $i \in \mathcal{A}$ when variable $x_{i,j}$ is one. Finally, Constraints 2.17 and 2.18 ensure that for each type of resources, the quantity of resources that move to each activity equals the quantity of resources that move from that activity and equals the quantity of resources that activity requires to be processed.

2.3.3 Inventory Integrated Stochastic Flow-based Continuous Time (iSFCT) Model

In most projects, in addition to the resources, processing each activity requires specific make-to-order materials (e.g., special pipe spools), and processing of an activity cannot start before the delivery of required materials. For each activity, the engineering team decides the specifics of the required materials. In our formulations, we assume that all materials that are required for an activity are delivered together and that deliveries are made on time. Similar to the SRCPSP, activity durations are random variables with known distributions, and durations have no restrictions on distributions or dependencies.

We model the problem of finding the optimal solution of iSRCPSP within the class of proactive policies as an optimization problem, and we refer to it as the Inventory Integrated Stochastic Flow-based Continuous Time (iSFCT) model:

$$\text{iSFCT}: \quad \min_{\vec{f}} \min_{\vec{o}} E_{\vec{P}}\Big[G(\vec{P}, \vec{f}, \vec{o})\Big], \tag{2.22}$$

where

$\vec{f}$: Vector of flow of resources ($f_{i,j,r}$ for $i,j \in \mathcal{A}$ and $r \in \mathcal{R}$);
$\vec{o}$: Vector of materials delivery times (o_i for $i \in \mathcal{A}$);
$\vec{P}$: Vector of activity durations, where each element has an associated probability distribution (P_i for $i \in \mathcal{A}$);
$G(.)$: A function that provides the stochastic cost of the project, given the resource flow, $\vec{f}$, and the delivery times of materials, $\vec{o}$.

Note that our goal here is to minimize the expected cost of a project. In iSFCT, both expected makespan and expected inventory holding cost contribute to the cost of a project.

The inventory holding cost is a consequence of early deliveries. At each activity, when materials is delivered early, an inventory holding cost proportional to the amount of time that it is delivered before the start of the process is accrued. iSFCT identifies delivery time of materials as well as the flow of resources, from which, by Lemma 1, we can determine the sequence of activities visited by each resource.

By using SAA technique, the iSFCT (Model 2.22) can be approximated by iSFCT($\mathcal{S}$):

$$\text{iSFCT}(\mathcal{S}): \quad \min_{\vec{f}} \min_{\vec{o}} \sum_{s\in\mathcal{S}} \hat{G}(\vec{p}_s, \vec{f}, \vec{o})/|\mathcal{S}|, \tag{2.23}$$

where

$\mathcal{S}$: Set of scenarios;
$|.|$: cardinality of the set;
$\vec{p}_s$: Vector of activity durations in scenario $s \in \mathcal{S}$;
$\hat{G}(.)$: A function that provides the cost of the project given the flow of resources, $\vec{f}$, the delivery times of materials, $\vec{o}$, and the realized durations, $\vec{p}_s$.

Note that in iSFCT($\mathcal{S}$), given the flow of resources $\vec{f}$, $\min_{\vec{o}} \sum_{s\in\mathcal{S}} \hat{G}(\vec{p}_s, \vec{f}, \vec{o})/|\mathcal{S}|$ is a linear programming (LP) formulation that determines materials delivery times to minimize the expected cost of the project over the set of scenarios $\mathcal{S}$.

As to Subsection 2.3.2, the number of scenarios selected in practice should be sufficiently large to reduce the *error* to an acceptable threshold. In Lemma 3, we show that the optimal objective function value of iSFCT($\mathcal{S}$) converges to the optimal objective function value of iSFCT as $|\mathcal{S}|$ goes to infinity.

Lemma 5. *When $|\mathcal{S}|\rightarrow \infty$, the optimal objective function value of iSFCT($\mathcal{S}$) converges to the optimal objective function value of iSFCT with probability of one.*

Proof. Proof is similar to the proof of Lemma 3. □

Next, we propose the MILP formulation for iSFCT($\mathcal{S}$) that is an extension to the MILP formulation of SFCT($\mathcal{S}$). iSFCT($\mathcal{S}$) determines the flow of resources and the delivery times of the materials simultaneously, in order to minimize the expected total cost of the project over the scenarios in $\mathcal{S}$, while capturing precedence, resource and inventory-related constraints.

We define the objective function to be the minimization of the expected makespan plus γ times the expected inventory holding cost, where the parameter $1/\gamma$ can be interpreted as the monetary cost of extending the expected makespan of the project by one time unit. The value of parameter γ must be estimated in practice – this estimation is beyond the scope of this research.

In this formulation, the parameter ω_i for $i \in \mathcal{A}$ represents the rate of inventory holding cost for materials of activity i and the variable o_i for $i \in \mathcal{A}$ represents the delivery time of materials of activity i. The remaining notation is the same as that of the MILP formulation of SFCT($\mathcal{S}$) presented in Subsection 2.3.2.

$$\Big(\text{iSFCT}(\mathcal{S})\Big) \quad \mathit{Minimize} \quad \sum_{s\in\mathcal{S}} z^s_{n+1}/|\mathcal{S}| + \gamma \sum_{i\in\mathcal{A}} \omega_i \times \sum_{s\in\mathcal{S}} (z^s_i - o_i)/|\mathcal{S}| \tag{2.24}$$

$$x_{i,j} = 1 \qquad \forall\,(i,j) \in \mathcal{E} \tag{2.25}$$

$$f_{i,j,r} - \min(d_{i,r}, d_{j,r})x_{i,j} \le 0 \qquad \forall\, i,j \in \mathcal{A}, r \in \mathcal{R} \tag{2.26}$$

$$z^s_j - z^s_i \ge p^s_i - M(1 - x_{i,j}) \qquad \forall\, i,j \in \mathcal{A}, s \in \mathcal{S} \tag{2.27}$$

$$z^s_i \ge o_i \qquad \forall\, i \in \mathcal{A}, s \in \mathcal{S} \tag{2.28}$$

$$\sum_{j\in\mathcal{A}} f_{i,j,r} = d_{i,r} \qquad \forall\, i \in \mathcal{A} - \{n+1\}, r \in \mathcal{R} \tag{2.29}$$

$$\sum_{i\in\mathcal{A}} f_{i,j,r} = d_{j,r} \qquad \forall\, j \in \mathcal{A} - \{0\}, r \in \mathcal{R} \tag{2.30}$$

$$f_{i,j,r} \ge 0 \qquad \forall\, i,j \in \mathcal{A}, r \in \mathcal{R} \tag{2.31}$$

$$z^s_i \ge 0 \qquad \forall\, i \in \mathcal{A}, s \in \mathcal{S} \tag{2.32}$$

$$x_{i,j} \in \{0,1\} \qquad \forall\, i,j \in \mathcal{A} \tag{2.33}$$

The objective function (Equation 2.24) minimizes the expected cost of the project. Constraints 2.25 and 2.26 enforce the value of the binary variable $x_{i,j}$ for $i,j \in \mathcal{A}$ to be equal to one if the completion of activity i must precede to the start of processing of activity j, or if any resource moves from activity i to activity j. Constraint 2.27 ensures that in all scenarios, activity $j \in \mathcal{A}$ starts after the completion of activity $i \in \mathcal{A}$ when variable $x_{i,j}$ is one. Constraint 2.28 ensures that processing of an activity does not start before the delivery of its required materials. Finally, Constraints 2.29 and 2.30 ensure that for each type of resources, the quantity of resources that moves to each activity equals the quantity of resources that moves from that activity and equals the quantity of resources that activity requires to be processed.

2.4 Solution Approach

Consider any instance of deterministic RCPSP. The optimal solution of the RCPSP can be found by finding the optimal policy within the class of proactive policies for SRCPSP or iSRCPSP when activity durations are deterministic. Thus, we conclude that finding the optimal policies within the class of proactive policies for SRCPSP and iSRCPSP are at least as hard as solving RCPSP, so they are NP-hard in the strong sense.

The MILP SFCT($\mathcal{S}$) and iSFCT($\mathcal{S}$) formulations have $O(|\mathcal{A}|^2)$ binary variables, $O(|\mathcal{A}||\mathcal{S}| +|\mathcal{A}|^2|\mathcal{R}|)$ continuous variables and $O(|\mathcal{A}|^2(|\mathcal{R}|+|\mathcal{S}|))$ constraints, where $|\mathcal{A}|$ is the number of activities, $|\mathcal{S}|$ is the number of scenarios, and $|\mathcal{R}|$ is the number of resource types. As we discussed in Subsections 2.3.2 and 2.3.3, a sufficiently large number of scenarios are required to impose an acceptable bound on the error. For large $|\mathcal{S}|$, e.g., $|\mathcal{S}|= 2000$, we observed Gurobi (version 9.0.0) cannot provide even a feasible solution within the time limit of 3600 seconds for instances of PSPLIB benchmark with 30 activities and $|\mathcal{R}|= 4$. Although by increasing $|\mathcal{S}|$, the number of binary variables remains the same, we conjecture that the computational challenges increases due to the increase in the number of constraints, which increases the computation time at each node.

We are not aware of any approach that can find the optimal solution of SFCT($\mathcal{S}$) and iSFCT($\mathcal{S}$), given $A = 30, 60$ and $|\mathcal{S}|$ is large. Therefore, in this chapter, we focus on developing heuristics to find feasible solutions. In order to compare the performance of heuristics, in Subsection 2.4.1, we present an approach for finding a lower bound for both SFCT and iSFCT. In addition to the lower bound, in Subsection 2.4.1, we propose a probabilistic lower bound (that is, an approximate lower bound along with the probability that it is in fact an actual lower bound). The probabilistic lower bounds enhances our intuition about the tightness of both our lower and upper bounds.

We suggest two alternative strategies for developing heuristics. One involves restricting the number of scenarios for SAA approach, and the other involves leveraging successful computational methods for the deterministic version of the problem. In Subsection 2.4.2, we propose multiple deterministic and SAA-based heuristics for finding feasible solutions for SFCT. Finally, in Subsection 2.4.3, we extend those heuristics to provide feasible solutions for iSFCT. Although all proposed heuristics result in feasible solutions, they differ in terms of computation time, difficulty of implementation, and optimality gap. To select a particular approach for a project, one must trade off level of sophistication and available computing power.

2.4.1 Lower Bound and Probabilistic Lower Bounds

Lower Bound

In Lemma 6, we prove that regardless of the distribution of activity durations, the optimal objective function value of the RCPSP when activity durations are assumed to be equal to their means is a lower bound on both SFCT and iSFCT.

Lemma 6. *The optimal objective function value of the RCPSP in which activity durations are assumed to be equal to the mean of their distributions is a lower bound on the optimal objective function value of both SFCT and iSFCT.*

Proof. Define $\hat{F}(\vec{p}, \vec{f})$ to be a function that given the flow of resources, $\vec{f}$, and vector of deterministic durations, $\vec{p}$, provides the optimal objective function value of the FCT formulation. For a fixed flow of resources, $\vec{f}$, there are a finite number of paths between the source and the tail of a project. Let $\Omega_{\vec{f}}$ denote the set of such paths, $\vec{p_1}$ and $\vec{p_2}$ denote two vectors of activity durations, and $p_{1,i}$ and $p_{2,i}$ denote the duration of activity i in $\vec{p_1}$ and $\vec{p_2}$, respectively. It follows that:

$$\hat{F}(\vec{p_1}, \vec{f}) = \max_{\omega \in \Omega_{\vec{f}}} \Big(\sum_{i \in \omega} p_{1,i} \Big)$$

$$\hat{F}(\vec{p_2}, \vec{f}) = \max_{\omega \in \Omega_{\vec{f}}} \Big(\sum_{i \in \omega} p_{2,i} \Big)$$

$$\hat{F}(\vec{p_1} + \vec{p_2}, \vec{f}) = \max_{\omega \in \Omega_{\vec{f}}} \Big(\sum_{i \in \omega} (p_{1,i} + p_{2,i}) \Big)$$

For a given flow of resources, $\vec{f}$:

$$\max_{\omega \in \Omega_{\vec{f}}} \Big(\sum_{i \in \omega} (p_{1,i} + p_{2,i}) \Big) \leq \max_{\omega \in \Omega_{\vec{f}}} \Big(\sum_{i \in \omega} p_{1,i} \Big) + \max_{\omega \in \Omega_{\vec{f}}} \Big(\sum_{i \in \omega} p_{2,i} \Big),$$

thus, we conclude that for a given flow of resources, $\vec{f}$, function $\hat{F}(., \vec{f})$ is convex. So, for any two vectors of activity durations, $\vec{p_1}$ and $\vec{p_2}$, and a flow of resources $\vec{f}$:

$$\hat{F}(\vec{p_1} + \vec{p_2}, \vec{f}) \leq \hat{F}(\vec{p_1}, \vec{f}) + \hat{F}(\vec{p_2}, \vec{f}). \tag{2.34}$$

Let $\vec{p}_{mean}$ represent the vector of activity durations that contains $E[P_i]$ for each activity $i \in \mathcal{A}$. With this notation, the optimal objective function value of the FCT formulation when activity durations are equal to the mean of their distribution is $\hat{F}(\vec{p}_{mean}, \vec{f^*}_{mean})$, where $\vec{f^*}_{mean}$ is the optimal resource flow with regards to $\vec{p}_{mean}$. Also let $\vec{p}_{sum}$ represent the vector of activity durations that contains $\sum_{s \in \mathcal{S}} p_i^s$ for $i \in \mathcal{A}$, where $|\mathcal{S}| \rightarrow \infty$. By changing the scale of time, it is straightforward to see that for any resource flow $\vec{f}$:

$$|\mathcal{S}| \times \hat{F}(\vec{p}_{mean}, \vec{f}) = \hat{F}(|\mathcal{S}| \times \vec{p}_{mean}, \vec{f}) = \hat{F}(\vec{p}_{sum}, \vec{f}) \tag{2.35}$$

Let $\vec{p}_s$ for $s \in \mathcal{S}$ represents the vector of activity durations that contain p_i^s for $i \in \mathcal{A}$, and $\vec{f}^*$ represents the optimal flow of resources for the SFCT($\mathcal{S}$). Based on Equations 2.34 and 2.35:

$$\hat{F}(\vec{p}_{mean}, \vec{f}^*_{mean}) \leq \hat{F}(\vec{p}_{mean}, \vec{f}^*) = \lim_{|\mathcal{S}| \rightarrow \infty} \hat{F}(\vec{p}_{sum}, \vec{f}^*)/|\mathcal{S}| \leq \lim_{|\mathcal{S}| \rightarrow \infty} \sum_{s \in S} \hat{F}(\vec{p}_s, \vec{f}^*)/|\mathcal{S}| \tag{2.36}$$

Let $Z^{\mathcal{S}}_{SFCT}$ and Z^{*}_{SFCT} represent the optimal objective function value of SFCT($\mathcal{S}$) and SFCT, respectively. Based on Equation 2.36 and Lemma 3:

$$\hat{F}(\vec{p}_{mean}, \vec{f}^{*}_{mean}) \leq \lim_{|\mathcal{S}|\to\infty} \sum_{s\in S} \hat{F}(\vec{p}_s, \vec{f}^{*})/|\mathcal{S}| = \lim_{|\mathcal{S}|\to\infty} Z^{\mathcal{S}}_{SFCT} = Z^{*}_{SFCT} \tag{2.37}$$

Thus, $\hat{F}(\vec{p}_{mean}, \vec{f}^{*}_{mean})$ is a lower bound for the SFCT. Also, the optimal objective function value of the iSFCT is always grater than or equal to the optimal objective function value of the SFCT, therefore, $\hat{F}(\vec{p}_{mean}, \vec{f}^{*}_{mean})$ is also a lower bound for the iSFCT. □

Probabilistic Lower Bound

Consider a randomly selected sample subset of scenarios with finite cardinality, $\mathcal{S}'$, and let $Z^{\mathcal{S}'}_{SFCT}$ represents the optimal objective function value of SFCT($\mathcal{S}'$). As $\mathcal{S}'$ is selected randomly, $Z^{\mathcal{S}'}_{SFCT}$ is neither an upper bound nor a lower bound for the optimal objective function value of the SFCT. However, by Lemma 7, its expected value ($E_{\mathcal{S}'}[Z^{\mathcal{S}'}_{SFCT}]$) is a lower bound for the optimal objective function value of the SFCT.

Lemma 7. *$E_{\mathcal{S}'}[Z^{\mathcal{S}'}_{SFCT}]$, where $Z^{\mathcal{S}'}_{SFCT}$ is the optimal objective function value of the SFCT($\mathcal{S}'$), is a lower bound for SFCT.*

Proof. Consider the following notation:

$\hat{F}(.)$: A function that provides the makespan of the project given the flow of resources, $\vec{f}$, and the realized durations, $\vec{p}_s$;

$\mathcal{S}'_1, \mathcal{S}'_2, ..., \mathcal{S}'_m$: m i.i.d randomly selected subsets of scenarios with the same finite cardinality;

$\vec{f}_{s'_k}$ for $k \in \{1, 2, ..., m\}$: Vector of the optimal resource flow of $SFCT(\mathcal{S}'_k)$;

$\vec{f}^{*}$: Vector of the optimal resource flow of SFCT;

$\vec{p}_s$: Vector of activity durations in scenario s.

Based on the strong law of large numbers and Lemma 3:

$$E_{\mathcal{S}'}[Z^{\mathcal{S}'}_{SFCT}] = \lim_{m\to\infty} \frac{\sum_{k=1}^{m}\sum_{s\in\mathcal{S}'_k} \hat{F}(\vec{p}_s, \vec{f}_{s'_k})}{\sum_{k=1}^{m}|\mathcal{S}'_k|} \leq \lim_{m\to\infty} \frac{\sum_{k=1}^{m}\sum_{s\in\mathcal{S}'_k} \hat{F}(\vec{p}_s, \vec{f}^{*})}{\sum_{k=1}^{m}|\mathcal{S}'_k|} = Z^{*}_{SFCT} \tag{2.38}$$

Therefore, $E_{\mathcal{S}'}[Z^{\mathcal{S}'}_{SFCT}]$ is a lower bound for the optimal objective function value of the SFCT. □

By Lemma 7, $E_{\mathcal{S}'}[Z^{\mathcal{S}'}_{SFCT}]$ is a lower bound for the SFCT. Nonetheless, the distribution of $Z^{\mathcal{S}'}_{SFCT}$ and its expected value ($E_{\mathcal{S}'}[Z^{\mathcal{S}'}_{SFCT}]$) are unknown. To calculate $E_{\mathcal{S}'}[Z^{\mathcal{S}'}_{SFCT}]$, one approach is to sample infinite number of subsets of scenarios ($\mathcal{S}'_1, \mathcal{S}'_2, ...$), solve the $SFCT(\mathcal{S}'_k)$

for $k \in \{1, 2, ...\}$, and averaging the results. However, as almost always the computational power is limited, this approach is not practical.

Instead, we can sample k' number of subsets of scenarios $(\mathcal{S}'_1, \mathcal{S}'_2, ..., \mathcal{S}'_{k'})$ and solve $SFCT(\mathcal{S}'_k)$ for $k \in \{1, 2, ..., k'\}$. This case is equivalent to having k' observations from an unknown distribution. When k' is sufficiently large, $\sum_{k=1}^{k'} Z_{SFCT}^{\mathcal{S}'_k}/k'$ has a t-distribution. The $1-\alpha$ percentile confidence interval represents an interval that contains $E_{\mathcal{S}'}[Z_{SFCT}^{\mathcal{S}'}]$ with probability of $1-\alpha$ for any α strictly between zero and one. We define the "probabilistic lower bound" as the lower wing of the $1-\alpha$ percentile confidence interval for $E_{\mathcal{S}'}[Z_{SFCT}^{\mathcal{S}'}]$. Thus, the probabilistic lower bound is lower than $E_{\mathcal{S}'}[Z_{SFCT}^{\mathcal{S}'}]$ and consequently lower than the optimal objective function value of the SFCT with the probability of at least $1-\alpha/2$.

For a fixed sample size ($|\mathcal{S}'|$) and a fixed maximum number of replications (k'), represent the probabilistic lower bound with $LB(\beta)$. Note that $LB(\beta)$ is the largest number that we can verify it is in fact an actual lower bound with the probability of at least β. For any given β, $LB(\beta)$ can be calculated with Equation 2.39.

$$LB(\beta) = \hat{\mu} - t_{1-\beta,k'-1} \times \hat{\sigma}/\sqrt{k'}, \tag{2.39}$$

where $t_{1-\beta,k'-1}$ is the β percentile of t-distribution with $k'-1$ degrees of freedom,

$$\hat{\mu} = \sum_{k=1}^{k'} Z_{SFCT}^{\mathcal{S}'_k}/k'$$

and

$$\hat{\sigma}^2 = \frac{\sum_{k=1}^{k'} (Z_{SFCT}^{\mathcal{S}'_k} - \hat{\mu})^2}{k'-1}.$$

The probabilistic lower bound can also be used to provide the probability that any estimated lower bound is an actual lower bound. For a fixed sample size and a fixed maximum number of replications, given the observations, the maximum probability (β) that any estimated lower bound, $\hat{LB}$, is an actual lower bound can be calculated with Equation 2.40 if all observations are greater than $\hat{LB}$.

$$\beta = LB^{-1}(\hat{LB}), \tag{2.40}$$

where $LB^{-1}(.)$ is the inverse of the probabilistic lower bound function that is presented in Equation 2.39.

Note that we can also introduce the probabilistic lower bound for iSFCT with similar approach and reasoning (substitute Lemma 8 for Lemma 7).

Lemma 8. *$E_{\mathcal{S}'}[Z_{iSFCT}^{\mathcal{S}'}]$, where $Z_{iSFCT}^{\mathcal{S}'}$ is the optimal objective function value of the iSFCT($\mathcal{S}'$), is a lower bound for iSFCT.*

Proof. Consider the following notation:

$\hat{G}(.)$: A function that provides the cost of the project given the flow of resources, $\vec{f}$, the delivery times of materials, $\vec{o}$, and the realized durations, $\vec{p_s}$;

$\mathcal{S'}_1, \mathcal{S'}_2, ..., \mathcal{S'}_m$: m i.i.d randomly selected subsets of scenarios with the same finite cardinality;
$\vec{f}_{s'_k}$ for $k \in \{1, 2, ..., m\}$: Vector of the optimal resource flow of $iSFCT(\mathcal{S'}_k)$;
$\vec{f}^*$: Vector of the optimal resource flow of iSFCT;
$\vec{o}$: Vector of materials delivery times;
$\vec{p}_s$: Vector of activity durations in scenario s.

Based on the strong law of large numbers and Lemma 5:

$$\begin{aligned} E_{\mathcal{S'}}[Z_{iSFCT}^{\mathcal{S'}}] = \lim_{m\to\infty} \frac{\sum_{k=1}^{m} \min_{\vec{o}} \sum_{s\in\mathcal{S'}_k} \hat{G}(\vec{p}_s, \vec{f}_{s'_k}, \vec{o})}{\sum_{k=1}^{m} |\mathcal{S'}_k|} \leq \\ \lim_{m\to\infty} \frac{\sum_{k=1}^{m} \min_{\vec{o}} \sum_{s\in\mathcal{S'}_k} \hat{G}(\vec{p}_s, \vec{f}^*, \vec{o})}{\sum_{k=1}^{m} |\mathcal{S'}_k|} \leq \\ \lim_{m\to\infty} \min_{\vec{o}} \frac{\sum_{k=1}^{m} \sum_{s\in\mathcal{S'}_k} \hat{G}(\vec{p}_s, \vec{f}^*, \vec{o})}{\sum_{k=1}^{m} |\mathcal{S'}_k|} = Z_{iSFCT}^* \end{aligned} \quad (2.41)$$

Therefore, $E_{\mathcal{S'}}[Z_{iSFCT}^{\mathcal{S'}}]$ is a lower bound for the optimal objective function value of the iSFCT. □

2.4.2 Heuristics for SFCT

This subsection is divided into two sections, as we consider two alternative strategies for heuristics. The first involves restricting the number of scenarios for SAA approach, and we discuss this in the first part. The other involves leveraging successful computational methods for the deterministic version of the problem, and we discuss this in the second part.

Although all of these approaches result in feasible solutions, they differ in terms of computation time, difficulty of implementation, and optimality gap. To select a particular approach for a project, one must trade off level of sophistication and available computing power.

Sample Average Approximation-based Heuristics

As we discussed above, solving SFCT($\mathcal{S}$) is not practical when $\mathcal{S}$ is large. SAA technique approximates stochastic problems by solving them over a small sample of scenarios. Define $\mathcal{S'}$ as a set of scenarios with small cardinality ($|\mathcal{S'}| \ll |\mathcal{S}|$). In SAA, instead of solving SFCT($\mathcal{S}$), we solve SFCT($\mathcal{S'}$), and define $\vec{f}_{s'}$ as its optimal solution. As $\vec{f}_{s'}$ satisfies all constraints in SFCT($\mathcal{S}$) and SFCT, it is a feasible solution for them.

To measure the quality of solutions, given a feasible solution $\vec{f}_{s'}$, we calculate $\hat{F}(\vec{p}_s, \vec{f}_{s'})$ for each scenario $s \in \mathcal{S}$, and define $Z_{SFCT}^{\mathcal{S}}(\vec{f}_{s'})$, as in Equation 2.42, as their average. Note that calculating $\hat{F}(\vec{p}_s, \vec{f}_{s'})$ is equivalent to finding the longest path in a graph, and can be done in polynomial time. Then, $Z_{SFCT}^{\mathcal{S}}(\vec{f}_{s'})$ denote the objective function value of this solution in the SFCT($\mathcal{S}$), and by Lemma 3, in the SFCT when $|\mathcal{S}|$ is large. Using this objective function value and the lower bound provided in Subsection 2.4.1, we can analyse the optimality gap.

$$Z^{\mathcal{S}}_{SFCT}(\vec{f}_{s'}) = \sum_{s \in \mathcal{S}} \hat{F}(\vec{p}_s, \vec{f}_{s'})/|\mathcal{S}| \tag{2.42}$$

One way to decrease the optimality gap of SAA solution is to increase the number of scenarios in $\mathcal{S}'$. However, when standard solvers are utilized to optimize SFCT($\mathcal{S}'$), we observed that they are unable to find even a feasible solution within a reasonable time limit when $|\mathcal{S}'|$ is large. As a result, we can only increase the sample size up to some extent.

SAA solution is a function of the selected set of scenarios $\mathcal{S}'$, and if it changes, the solution might change as well. Hence, independently applying SAA multiple times results in multiple solutions. We use this fact and propose the Reinforced Sample Average Approximation Decomposition (SAAD+) approach in order to find feasible solutions with lower optimality gap.

In SAAD+, we replicate SAA for k times to find k feasible solutions. Then, we select the solution with the lowest objective function value in the SFCT($\mathcal{S}$). The summary of SAAD+ is depicted in Figure 2.1.

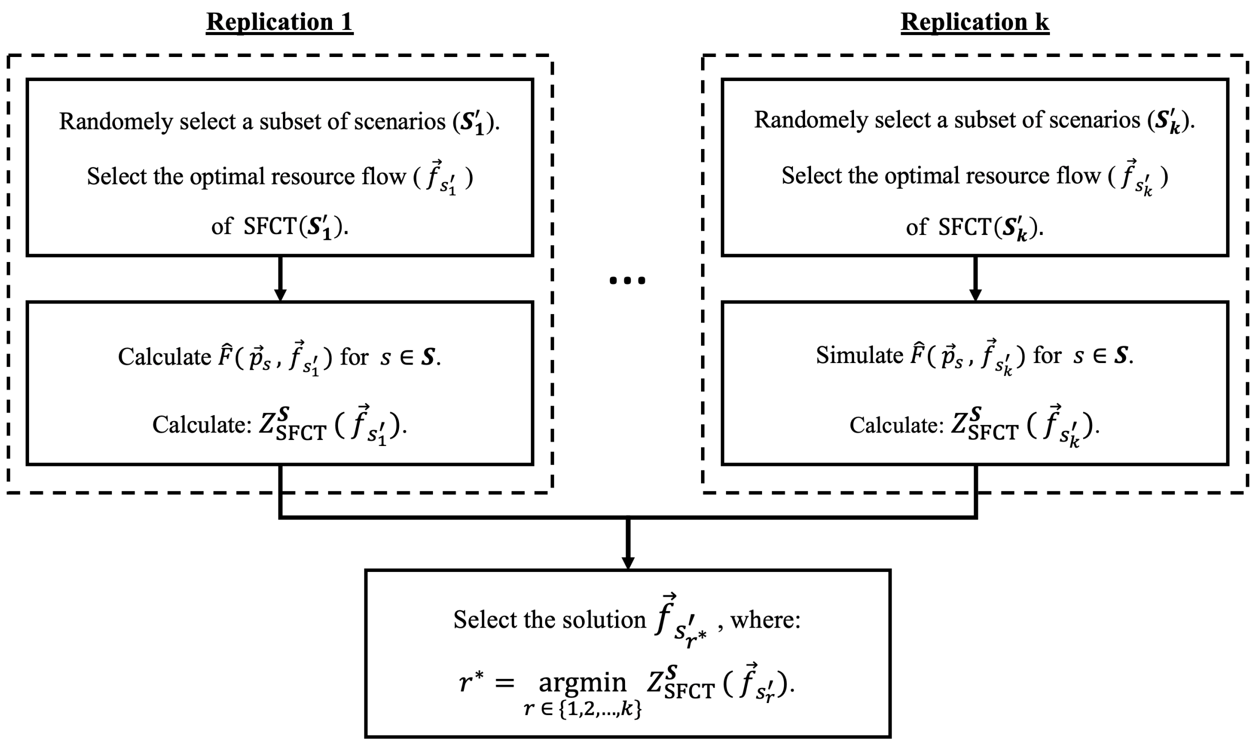

Replication 1
Randomely select a subset of scenarios (S'_1).
Select the optimal resource flow ($\vec{f}_{s'_1}$)
of SFCT(S'_1).
Calculate $\hat{F}(\vec{p}_s, \vec{f}_{s'_1})$ for $s \in S$.
Calculate: $Z^{S}_{\text{SFCT}}(\vec{f}_{s'_1})$.
Replication k
Randomely select a subset of scenarios (S'_k).
Select the optimal resource flow ($\vec{f}_{s'_k}$)
of SFCT(S'_k).
Simulate $\hat{F}(\vec{p}_s, \vec{f}_{s'_k})$ for $s \in S$.
Calculate: $Z^{S}_{\text{SFCT}}(\vec{f}_{s'_k})$.
Select the solution $\vec{f}_{s'_{r^*}}$, where:
$r^* = \underset{r \in \{1,2,\ldots,k\}}{\text{argmin}}\ Z^{S}_{\text{SFCT}}(\vec{f}_{s'_r})$.

Figure 2.1: Summary of SAAD+.

The complexity of SAAD+ approach increases linearly as the number of replications increases. Therefore, we can have many replications while maintaining the same order of magnitude of solution time. Moreover, as different replications are independent from each other, this process can be parallelized. Note that it is not worthwhile to have an extremely large number of replications, as the incremental expected decrease in the objective function value in the SFCT($\mathcal{S}$) due to this approach, and by Lemma 3, in the SFCT when $|\mathcal{S}|$ is large, decreases as the number of replications increases (Lemma 9).

Lemma 9. *Increasing k decreases* $E\Big[\min_{r=1}^{k} Z^{\mathcal{S}}_{SFCT}(\vec{f_{s'_r}}) - \min_{r=1}^{k+1} Z^{\mathcal{S}}_{SFCT}(\vec{f_{s'_r}})\Big]$.

Proof. Let $\theta(k)$ denotes $E\Big[\min_{r=1}^{k} Z^{\mathcal{S}}_{SFCT}(\vec{f_{s'_r}}) - \min_{r=1}^{k+1} Z^{\mathcal{S}}_{SFCT}(\vec{f_{s'_r}})\Big]$. Consider any k_1 and k_2 where $k_1 < k_2$. Then, for any constant y:

$$Pr\Big(\min_{r=1}^{k_1} Z^{\mathcal{S}}_{SFCT}(\vec{f_{s'_r}}) > y\Big) \geq Pr\Big(\min_{r=1}^{k_2} Z^{\mathcal{S}}_{SFCT}(\vec{f_{s'_r}}) > y\Big),$$

and since all replications are i.i.d.:

$$Pr\Big(\max\Big(\min_{r=1}^{k_1} Z^{\mathcal{S}}_{SFCT}(\vec{f_{s'_r}}) - Z^{\mathcal{S}}_{SFCT}(\vec{f_{s'_{k_1+1}}}), 0\Big) > y\Big) \geq$$
$$Pr\Big(\max\Big(\min_{r=1}^{k_2} Z^{\mathcal{S}}_{SFCT}(\vec{f_{s'_r}}) - Z^{\mathcal{S}}_{SFCT}(\vec{f_{s'_{k_2+1}}}), 0\Big) > y\Big). \quad (2.43)$$

From 2.43, we conclude that:

$$\theta(k_1) - \theta(k_2) = E\Big[\max\Big(\min_{r=1}^{k_1} Z^{\mathcal{S}}_{SFCT}(\vec{f_{s'_r}}) - Z^{\mathcal{S}}_{SFCT}(\vec{f_{s'_{k_1+1}}}), 0\Big)\Big] -$$
$$E\Big[\max\Big(\min_{r=1}^{k_2} Z^{\mathcal{S}}_{SFCT}(\vec{f_{s'_r}}) - Z^{\mathcal{S}}_{SFCT}(\vec{f_{s'_{k_2+1}}}), 0\Big)\Big] \geq 0$$

□

Deterministic Heuristics

Our interviews with a variety of consultants and project and supply chain managers (see Jabbari and Kaminsky [37] and Appendix A) reveal that in practice, the mean of activity durations are often used to obtain a project plan in a deterministic manner. This is a natural approach to approximating stochastic problems, and in fact, as Ballestıén [4] discussed, an effective way to schedule SRCPSP when the variability in activity durations is moderate.

In this subsection, we also assume activity durations are deterministic and equal to the mean of their distributions, and select an optimal resource flow of the FCT formulation of the deterministic problem. An important consideration is that, as it is highlighted in Example 1, there are many alternative optimal solutions for the FCT formulation and we must decide which one to select. Note that by utilizing different approaches for selecting an optimal solution among the large number of alternatives, different variants of our deterministic heuristics can be created.

Example 1. *Consider a project with $2N$ activities with the deterministic duration of one, and no precedence constraint. All resources are similar, N resources are available at the project and each activity requires one resource to be processed.*

The optimal objective function value for this problem is 2. However, there are $(2N)!/N!$ optimal flow of resources for this problem. Even for a given optimal activity start times, there are $N!$ optimal flow of resources.

SAA-based heuristics involve solving SFCT($\mathcal{S}'$), which is NP-hard in the strong sense, requires extensive computational power and have been rarely considered in the literature. All deterministic heuristics are also NP-hard in the strong sense, as they required the solution of the deterministic RCPSP. However, as we discussed in Subsection 2.2.1, a large amount of technology has been developed to solve the deterministic problem effectively. The goal of these heuristics is to leverage that technology for the stochastic problem. Note that in deterministic heuristic, it is not necessary to solve the FCT formulation directly. Instead, we can use any formulation, and then, given the optimal activity start times, construct an FCT optimal solution. In Section 2.5, we show that deterministic heuristics in general requires much less computational power that SAA-based heuristics and can be applied to projects with more activities.

In recent years, Laborie [50] developed a package specifically for solving RCPSP that performs well on all benchmark test problems and extends to project with many activities. In this chapter, for our deterministic heuristics, we leverage this black-box package through the Constraint Programming solver of IBM ILOG CP Optimizer (version 2.12.182) to solve the deterministic RCPSP, as it is explained in Laborie [50]. The output of this solver is the optimal activity start times, and if an instance is not solved optimally within the time limit, its output is the activity start times in the best solution it found. Let $\hat{z}_i$ for $i \in \mathcal{A}$ represents the start time of activity i that is provided by this CP Optimizer.

In the following, we propose three variants of our deterministic heuristics, namely: Naive Deterministic Heuristic (NDH), Effective Arc Cardinality Minimization Heuristic (EACH), and Penalized Arc Minimization Heuristic (PAH). The difference among these deterministic heuristics is that they utilize different approaches in selecting an optimal solution among the large number of alternatives. Figure 2.2 depict a summary of these heuristics.

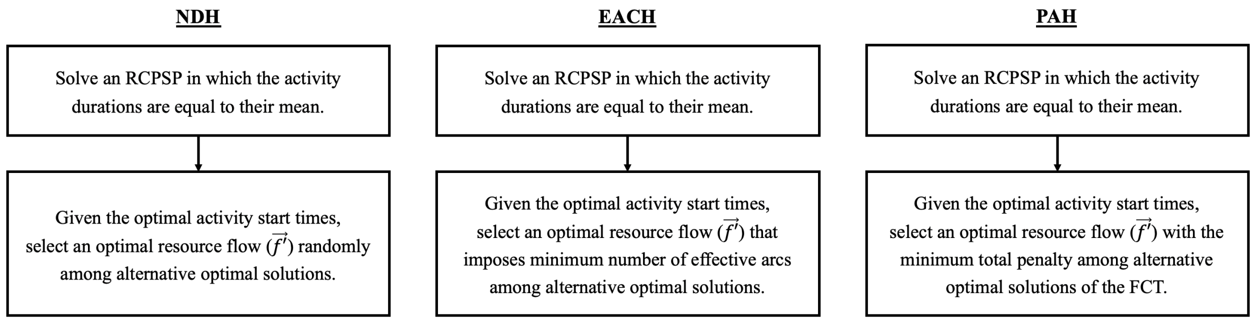

NDH
Solve an RCPSP in which the activity durations are equal to their mean.
Given the optimal activity start times, select an optimal resource flow ($\vec{f}'$) randomly among alternative optimal solutions.
EACH
Solve an RCPSP in which the activity durations are equal to their mean.
Given the optimal activity start times, select an optimal resource flow ($\vec{f}'$) that imposes minimum number of effective arcs among alternative optimal solutions.
PAH
Solve an RCPSP in which the activity durations are equal to their mean.
Given the optimal activity start times, select an optimal resource flow ($\vec{f}'$) with the minimum total penalty among alternative optimal solutions of the FCT.

Figure 2.2: Summary of NDH, EACH and PAH.

NDH:

NDH randomly selects one of the optimal resource flows. This approach represents the current practice, as practitioners completely neglect the uncertainty in activity durations in the initial planning phase of projects (see Jabbari and Kaminsky [37]). Given the optimal activity start times from the CP optimizer, to construct an optimal solution for the FCT, we find a resource flow that satisfies set of Equations 2.44-2.47. As all of these equations are linear, a feasible solution can be found in polynomial time.

$$f_{i,j,r} = 0 \qquad \forall\, i, j \in \mathcal{A}, r \in \mathcal{R} \;\; if \;\; \hat{z}_i + p_i > \hat{z}_j \tag{2.44}$$

$$\sum_{j \in \mathcal{A}} f_{i,j,r} = d_{i,r} \qquad \forall\, i \in \mathcal{A} - \{n+1\}, r \in \mathcal{R} \tag{2.45}$$

$$\sum_{i \in \mathcal{A}} f_{i,j,r} = d_{j,r} \qquad \forall\, j \in \mathcal{A} - \{0\}, r \in \mathcal{R} \tag{2.46}$$

$$f_{i,j,r} \geq 0 \qquad \forall\, i, j \in \mathcal{A}, r \in \mathcal{R} \tag{2.47}$$

Equation 2.44 ensures that resources cannot move from activity $i \in \mathcal{A}$ to activity $j \in \mathcal{A}$ it process on activity j starts before the completion of activity j. Equations 2.45 and 2.46 ensure that for each type of resources, the quantity of resources that move to each activity equals the quantity of resources that move from that activity and equals the quantity of resources that activity requires to be processed.

The limitation of NDH is that it does not distinguish between the alternative optimal solutions, which is why we refer to it as *naive*. In Example 2, we highlight this limitation.

Example 2. *A project with three activities is presented in Figure 2.3. In this project, the duration of activities 1 and 2 are uniform distribution between zero and one, and activity 3 has deterministic activity duration of one. Two similar resources are available at the project, each activity requires one resource to be processed, and activity 1 must precede activity 1.*

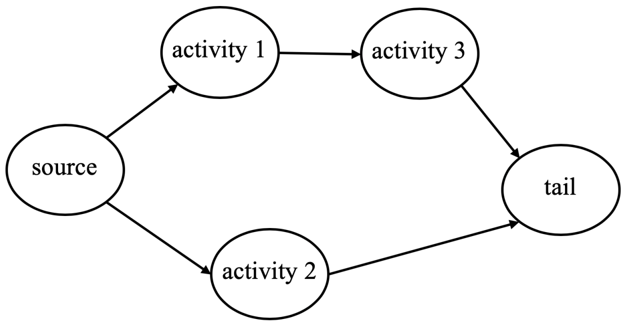


Figure 2.3: Illustration of the project that is introduced in Example 2.

The deterministic activity durations, which are defined as the mean of their distribution, are: $p_1 = p_2 = 0.5$ *and* $p_3 = 1$*. NDH, can select both resource flows that are presented in Figure 2.4, as they are both optimal for the FCT formulation. However, the true expected makespan of the project given the flow of resources that are presented in Figures 2.4a and 2.4b are* $3/2$ *and* $5/3$*, respectively. The reason is that the flow of resources presented in Figure 2.4b prevents activity 3 from starting before the completion of activity 2 while it is not necessary to add such constraint.*

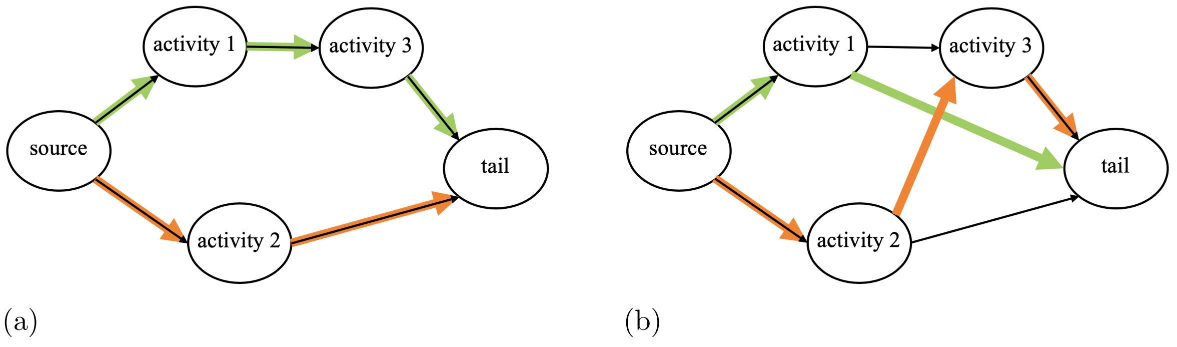


Figure 2.4: Two optimal resource flows for the FCT. Different colors illustrate the flow of different resources.

EACH:

Example 2 inspired a new heuristic, EACH, which never selects the resource flow presented in Figure 2.4b in that project. In any resource flow, when the quantity of resources that move from activity $i \in \mathcal{A}$ to activity $j \in \mathcal{A}$ is greater than zero, the processing of activity j must start after the completion of activity i. In other words, by selecting that resource flow we are imposing a precedence constraint between activities i and j. In what follows, we refer to the precedence constraints that are imposed by the resource flow and that add restrictions to the project as "effective arcs". Note that some of the imposed precedence constraints are not effective. For instance, when there is an arc from activity i to activity j, and there is another arc from activity j to activity k, it is already enforced that activity k cannot start before the completion of activity i, and hence, imposing a precedence constraint between activity i and k does not add any restrictions.

In EACH, the goal is to select a resource flow that imposes the minimum number of effective arcs among the alternative optimal solutions with the given activity start times. To find such resource flow, EACH solves the MILP formulation (2.48-2.57). In this model, the notation is the same as in the FCT formulation presented in Subsection 2.3.1. Note that although EACH does not consider alternative optimal solutions with different start times, and the search is limited to the alternative optimal solutions with the same activity start times that is given by the CP optimizer, as we show in Lemma 10, this MILP formulation is NP-complete in the strong sense. However, as we discuss in Section 2.5, Gurobi (version

9.0.0) with default settings efficiently optimizes this MILP formulation for projects with our desired sizes (30 and 60 activities.)

$$Minimize \quad \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{A}} x_{i,j} \tag{2.48}$$

$$x_{i,j} = 1 \qquad \forall\,(i,j) \in \mathcal{E} \tag{2.49}$$

$$f_{i,j,r} - Nx_{i,j} \leq 0 \qquad \forall\, i,j \in \mathcal{A}, r \in \mathcal{R} \tag{2.50}$$

$$x_{i,j} = 0 \qquad \forall\, i,j \in \mathcal{A} \;\; if \;\; \hat{z}_i + p_i > \hat{z}_j \tag{2.51}$$

$$\sum_{j\in\mathcal{A}} f_{i,j,r} \geq d_{i,r} \qquad \forall\, i \in \mathcal{A} - \{n+1\}, r \in \mathcal{R} \tag{2.52}$$

$$\sum_{i\in\mathcal{A}} f_{i,j,r} \geq d_{j,r} \qquad \forall\, j \in \mathcal{A} - \{0\}, r \in \mathcal{R} \tag{2.53}$$

$$\sum_{j\in\mathcal{A}} f_{k,j,r} = \sum_{i\in\mathcal{A}} f_{ikr} \qquad \forall\, k \in \mathcal{A} - \{0, n+1\}, r \in \mathcal{R} \tag{2.54}$$

$$\sum_{j\in\mathcal{A}} f_{0,j,r} = \sum_{i\in\mathcal{A}} f_{i,n+1,r} = D_r \qquad \forall\, r \in \mathcal{R} \tag{2.55}$$

$$f_{i,j,r} \geq 0 \qquad \forall\, i,j \in \mathcal{A}, r \in \mathcal{R} \tag{2.56}$$

$$x_{i,j} \in \{0,1\} \qquad \forall\, i,j \in \mathcal{A} \tag{2.57}$$

The objective function (Equation 2.48) minimizes the total number of effective arcs. Constraints 2.49 and 2.50 ensure that binary variable $x_{i,j}$ for $i,j \in \mathcal{A}$ equals one if the completion of activity i must precede to start the process of activity j, or if any resource moves from activity i to activity j. Constraint 2.51 enforces binary variable $x_{i,j}$ for $i,j \in \mathcal{A}$ equals zero if the completion of activity i is after the process starts on activity j. Constraints 2.52-2.54 ensure that for each type of resources, the quantity of resources that move to each activity at least equals the quantity of resources that activity requires to be processed, and equals the quantity of resources that move from that activity. Finally, Constraint 2.55 defines the total quantity of resources of each type that are available at the project.

Lemma 10. *Given fixed activity start times of a project, finding the resource flow that imposes the minimum number of effective arcs is NP-complete in the strong sense.*

Proof. Consider any instance of a 3-partition problem with a set $L = \{l_1, l_2, ..., l_{3m}\}$ that contains $3m$ positive integer numbers that are strictly between $B/2$ and $B/4$ (where B is a constant), and:

$$\sum_{j=1}^{3m} l_j = m \times B$$

The goal is to find whether it is possible to partition set L into m disjoint sub-sets in a way that the summation of numbers in each sub-set be equal to B. We reduce this problem to finding the flow of resources that imposes the minimum number of effective arcs in a project.

To do so, consider the project that is depicted in Figure 2.5. In this project, the start times of activities in Group 1 are zero and their completion times are one, and the start times of activities in Group 2 are one and their completion times are two. Only one resource type exists at the project, and $m \times B$ units of that resource is available. Activities $1, 2, ..., 3m$ require $l_1, l_2, ..., l_{3m}$ resources, respectively, and each of the activities $1', 2', ..., m'$ requires B resources. Note that arcs in this figure represent the precedence constraints.

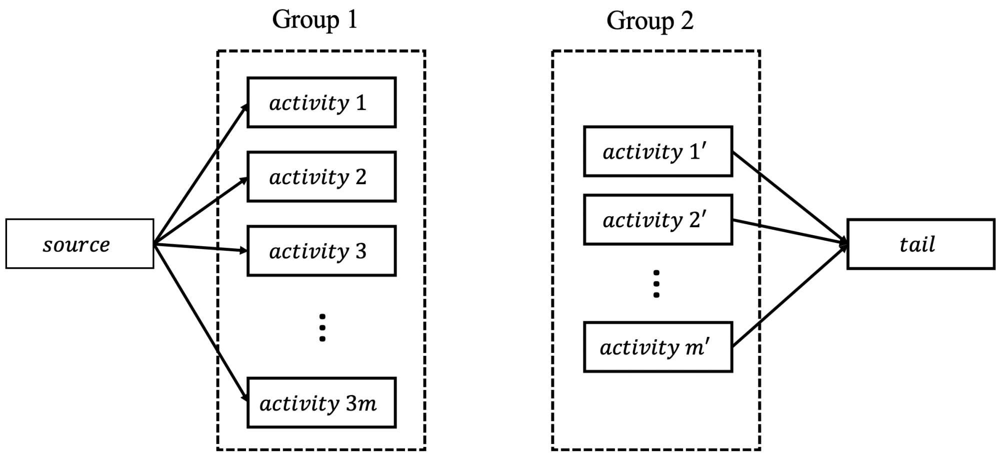


Figure 2.5: Reducing a 3-partition problem to finding the resource flow that imposes the minimum number of effective arcs, where activity start times are fixed.

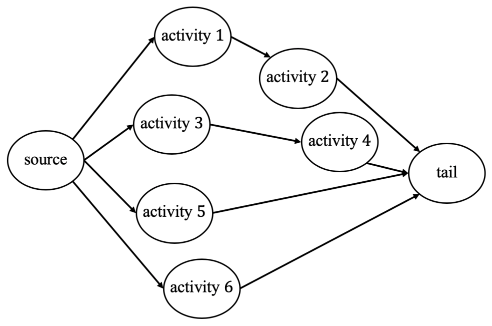
As each activity in Group 2 requires B resources, and the number of available resources at each activity in Group 1 is strictly between $B/4$ and $B/2$, resources from at least three activities in Group 1 needs to move to each activity in Group 2. So, any feasible solution imposes at least $3m$ arcs between two groups. If the set L can be partitioned into triples that all have the same sum (equal to B), then, there exists a resource flow that only imposes exactly $3m$ arcs between two groups as all resources in each activity in Group one only move to one activity in Group 2.

If no such partition exists, then it is impossible to find a feasible flow that imposes only $3m$ arcs between two groups, as at least the resources from one of the activities in Group 1 needs to move to more than one activity in Group 2, which makes the number of imposed arcs between two groups at least $3m + 1$.

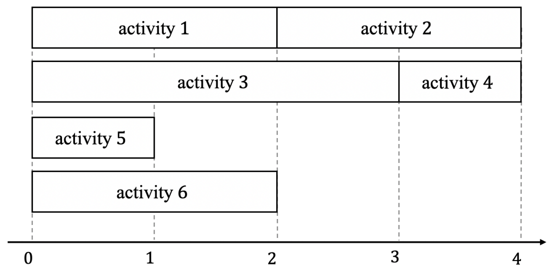
As any 3-partition problem with the positive integer numbers that are strictly between $B/4$ and $B/2$ can be solved by finding the flow of resources that imposes the minimum number of effective arcs in the project that is depicted in Figure 2.5, finding such resource flow is at least as hard as 3-partition problem, or in other words, it is NP-complete in the strong sense. □

The limitation of EACH is that it does not consider the tightness of different imposed constraint. In Example 3, we highlight how imposing different constraints impacts the project differently.

Example 3. *Consider the project that is depicted in Figure 2.6a. In this figure, arcs represent the precedence constraints. There are only two similar units of resources, and activities 2, 4, 5 and 6 require one unit of resources to be processed. In this project,* $P_1 = 2$, $P_2 = 2$, $P_3 = 3$, $P_4 = 1$, $P_5 = unif(0, 2)$, *and* $P_6 = unif(1, 3)$. *When activity durations are assumed to be equal to their mean, as it shown in Figure 2.6b, the optimal makespan is 4 and an optimal activity start times are:* $\hat{z}_1 = 0$, $\hat{z}_2 = 2$, $\hat{z}_3 = 0$, $\hat{z}_4 = 3$, $\hat{z}_5 = 0$, *and* $\hat{z}_6 = 0$.



(a)



(b)

Figure 2.6: (a) Illustration of the project that is introduced in Example 3, and (b) the optimal solution of the deterministic version, where activity durations are assumed to be equal to their mean.

Two optimal resource flows for the FCT are presented in Figure 2.7. As they are both optimal for the FCT, NDH can select both, and as they both impose two effective arcs, EACH can also select both. However, the true expected makespan of the project given the resource flows that are presented in Figures 2.6a and 2.6b are 4.00 *and* 4.25, *respectively. This happens because the completion time of activity 6 (when activity durations are assumed to be equal* *to their mean) is so close to the start time of activity 2. This suggests that the imposed precedence constraint between activity 6 and activity 2 is very likely to delay to the project.*

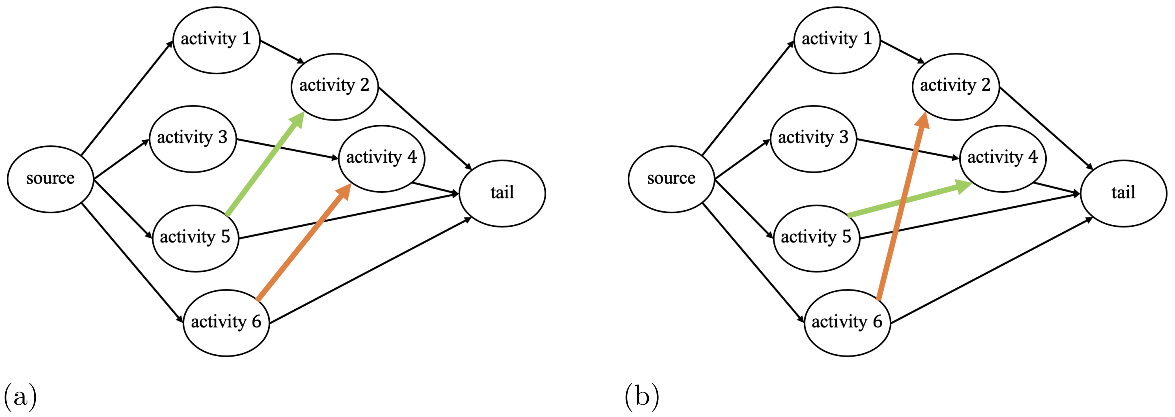


Figure 2.7: Two optimal resource flows for the FCT. Different colors illustrate the flow of different resources.

PAH:

Example 3 inspired us to propose a heuristic, PAH, that accounts for the different tightness of imposed constraints, never selects the resource flow presented in Figure 2.4b for the project that is introduced in Example 2, and never selects the resource flow presented in 2.7b for the project that is introduced in Example 3.

To account for the tightness of different imposed constraints, for every $i, j \in \mathcal{A}$, we first define a penalty function(Equation 2.58) for imposing a precedence constraint from activity i to activity j.

$$\text{Penalty}(i, j) = \frac{\sigma_i}{\hat{z}_j - \hat{z}_i - p_i + 0.5} \ , \tag{2.58}$$

where σ_i for $i \in \mathcal{A}$ is the standard deviation of duration of activity i. Note that the constant 0.5 can be changed to any strictly positive number and is thus a hyper-parameter for this heuristic approach.

This penalty function is designed to assign a higher penalty when the completion time of activity i is close to the start time of activity j. It also assigns a higher penalty if the duration of activity i has higher uncertainty. PAH selects a resource flow with the minimum total penalty among the alternative optimal solutions with the given activity start times by solving the MILP formulation (2.59-2.66). although PAH does not consider alternative optimal solutions with different start times, and the search is limited to the alternative optimal solutions with the same activity start times that is given by the CP optimizer, as we show in Lemma 11, this MILP formulation is NP-complete in the strong sense. However, as we discuss in Section 2.5, Gurobi (version 9.0.0) with default settings efficiently optimizes this MILP formulation for projects with our desired sizes (30 and 60 activities).

$$Minimize \quad \sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{A}} \frac{\sigma_i \times x_{i,j}}{\hat{z}_j - \hat{z}_i - p_i + 0.5} \tag{2.59}$$

$$x_{i,j} = 1 \qquad \forall\,(i,j) \in \mathcal{E} \tag{2.60}$$

$$f_{i,j,r} - \min(d_{i,r}, d_{j,r})x_{i,j} \leq 0 \qquad \forall\, i,j \in \mathcal{A}, r \in \mathcal{R} \tag{2.61}$$

$$x_{i,j} = 0 \qquad \forall\, i,j \in \mathcal{A} \;\; if \;\; \hat{z}_i + p_i > \hat{z}_j \tag{2.62}$$

$$\sum_{j\in\mathcal{A}} f_{i,j,r} = d_{i,r} \qquad \forall\, i \in \mathcal{A} - \{n+1\}, r \in \mathcal{R} \tag{2.63}$$

$$\sum_{i\in\mathcal{A}} f_{i,j,r} = d_{j,r} \qquad \forall\, j \in \mathcal{A} - \{0\}, r \in \mathcal{R} \tag{2.64}$$

$$f_{i,j,r} \geq 0 \qquad \forall\, i,j \in \mathcal{A}, r \in \mathcal{R} \tag{2.65}$$

$$x_{i,j} \in \{0,1\} \qquad \forall\, i,j \in \mathcal{A} \tag{2.66}$$

The objective function (Equation 2.59) minimizes the total penalty. Constraints 2.60 and 2.61 enforce binary variable $x_{i,j}$ for $i,j \in \mathcal{A}$ equals one if the completion of activity i must precede to start the process of activity j, or if any resource moves from activity i to activity j. Constraint 2.62 enforces binary variable $x_{i,j}$ for $i,j \in \mathcal{A}$ equals zero if the completion of activity i is after the process starts on activity j. Constraints 2.63 and 2.64 ensure that for each type of resources, the quantity of resources that move to each activity equals the quantity of resources that move from that activity and equals the quantity of resources that activity requires to be processed.

Lemma 11. *Given fixed activity start times of a project, finding the flow of resources with the minimum total penalty is NP-complete in the strong sense, where the penalty is defined as in Equation 2.58.*

Proof. We reduce any 3-partition problem in which the integer values are strictly between $B/4$ and $B/2$ to finding the resource flow with the minimum total penalty. To do so, consider the project that is presented in Figure 2.5 and assign the same standard deviation to the duration of all activities. Then, the penalty of any imposed precedence constraint between an activity in Group 1 to an activity in Group 2 is equal. The rest of the proof is similar to the proof of Lemma 10. □

Similar to SAA, to measure the quality of solutions, given a feasible solution $\vec{f'}$, we calculate $\hat{F}(\vec{p}_s, \vec{f'})$ for each scenario $s \in \mathcal{S}$. Then we define $Z^{\mathcal{S}}_{SFCT}(\vec{f'})$, as in Equation 2.42, as their average, which denote the objective function value of this solution in the SFCT($\mathcal{S}$), and by Lemma 3, in the SFCT when $|\mathcal{S}|$ is large. Using this objective function value and the lower bound provided in Subsection 2.4.1, we can analyze the optimality gap.

2.4.3 Heuristics for iSRCPSP

In Subsection 2.4.2, we present SAA and proposed SAAD+. We also introduced three variations of deterministic heuristics, namely: NDH, EACH and PAH. All of these heuristics find feasible solutions for the SFCT. In this subsection, we extend these heuristics to find feasible solutions for the iSFCT. In the first part, we focus on the deterministic heuristics and in the second part, we focus on SAA-based heuristics.

Deterministic Heuristics

A feasible solution for iSFCT consist of a resource flow ($\vec{f'}$) and the delivery times of materials ($\vec{o'}$). We find the feasible solutions for iSFCT by extending the deterministic approaches into two stage heuristics. In our two stage heuristics, some of the decisions (the flow of resources) are determined in the first stage, and given the flow of resources, the rest of the decisions (the material delivery times) are determined in the second stage.

We refer to the extended NDH, EACH and PAH, respectively, iNDH, iEACH and iPAH. Their first stages are identical to their original version, in which first a deterministic RCPSP is solved, and given the optimal activity start times, an optimal resource flow, $\vec{f'}$, is selected. In the second stage, given the selected resource flow $\vec{f'}$, the delivery times of material must be determined. We model the determination of the delivery times of materials as follows, and we refer to it as the ZāL($\mathcal{S}$):

$$\text{ZāL}(\mathcal{S}): \quad \min_{\vec{o}} \sum_{s \in \mathcal{S}} \hat{G}(\vec{p}_s, \vec{f'}, \vec{o})/|\mathcal{S}| \tag{2.67}$$

where the notation is identical to that of Subsection 2.3.3. The linear programming (LP) formulation of the ZāL($\mathcal{S}$) is as follows:

$$\Big(\text{ZāL}(\mathcal{S})\Big) \quad \mathit{Minimize} \quad \sum_{s \in \mathcal{S}} z^s_{n+1}/|\mathcal{S}| + \gamma \sum_{i \in \mathcal{A}} \omega_i \times \sum_{s \in \mathcal{S}} (z^s_i - o_i)/|\mathcal{S}| \tag{2.68}$$

$$z^s_j - z^s_i \geq p^s_i \qquad \forall\, i, j \in \mathcal{A}, s \in \mathcal{S} \;\; \text{if } (i,j) \in \mathcal{E} \;\; \text{or} \sum_{r \in \mathcal{R}} f'_{i,j,r} > 0 \tag{2.69}$$

$$z^s_i \geq o_i \qquad \forall\, i \in \mathcal{A}, s \in \mathcal{S} \tag{2.70}$$

$$o_i \geq 0 \qquad \forall\, i \in \mathcal{A} \tag{2.71}$$

The objective function (Equation 2.68) minimizes the expected cost of the project. Constraint 2.69 ensures that in all scenarios, activity $j \in \mathcal{A}$ starts after the completion of activity $i \in \mathcal{A}$ if the completion of activity i must precede to the start of processing of activity j, or if any resource moves from activity i to activity j. Finally, Constraint 2.70 ensures that processing of an activity does not start before the delivery of its required materials.

We represent the optimal objective function value of ZāL($\mathcal{S}$) by $Z^{\mathcal{S}}_{iSFCT}(\vec{f'}, \vec{o'})$, which also denotes the objective function value of this solution in the iSFCT($\mathcal{S}$), and by Lemma 5, in the iSFCT when $|\mathcal{S}|$ is large. Using this objective function value and the lower bound provided in Subsection 2.4.1, we can analyse the optimality gap. The extended deterministic heuristics are summarized in Figure 2.8.

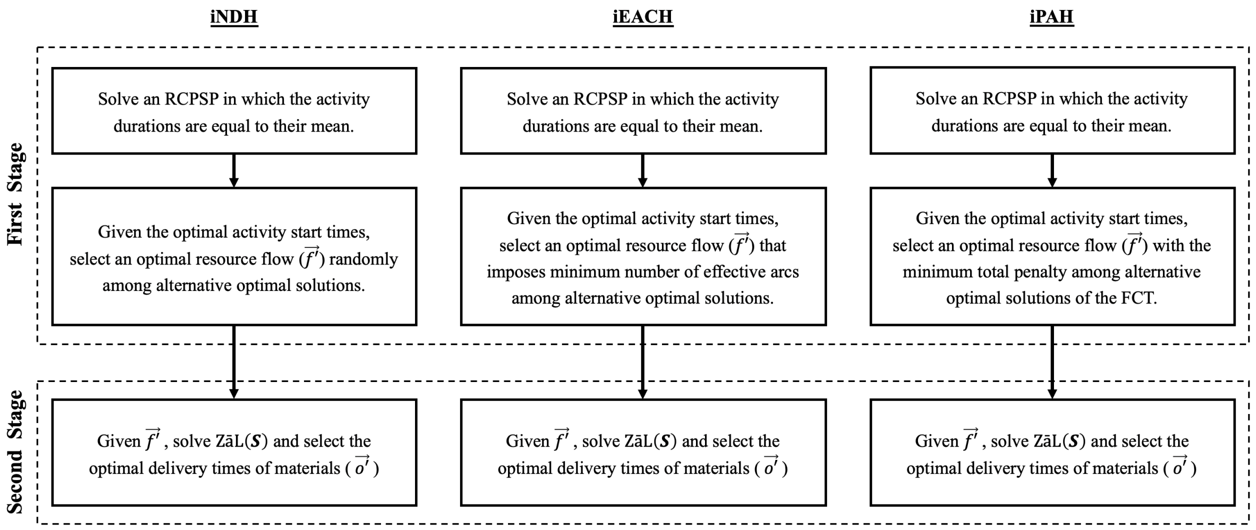

iNDH
iEACH
iPAH
First Stage
Second Stage
Solve an RCPSP in which the activity durations are equal to their mean.
Given the optimal activity start times, select an optimal resource flow ($\vec{f'}$) randomly among alternative optimal solutions.
Given the optimal activity start times, select an optimal resource flow ($\vec{f'}$) that imposes minimum number of effective arcs among alternative optimal solutions.
Given the optimal activity start times, select an optimal resource flow ($\vec{f'}$) with the minimum total penalty among alternative optimal solutions of the FCT.
Given $\vec{f'}$, solve ZāL($\mathcal{S}$) and select the optimal delivery times of materials ($\vec{o'}$)

Figure 2.8: Summary of iNDH, iEACH and iPAH.

Sample Average Approximation-based Heuristics

Similar to the extended deterministic heuristics, we extend SAA by transforming it into a two stage heuristic. We refer to the extended version Sample Average Approximation-based Decomposition Heuristic (iSAAD). Similar to SAA, define $\mathcal{S}'$ as a set of scenarios with small cardinality ($|\mathcal{S}'| \ll |\mathcal{S}|$). In the first stage of iSAAD, we solve iSFCT($\mathcal{S}'$), and define $\vec{f}_{s'}$ as its optimal solution. The second stage of iSAAD is identical to the second stage of the extended deterministic heuristic, which is given the resource flow $\vec{f}_{s'}$, finding the optimal delivery times of materials in ZāL($\mathcal{S}$). The summary of iSAAD is depicted in Figure 2.9.

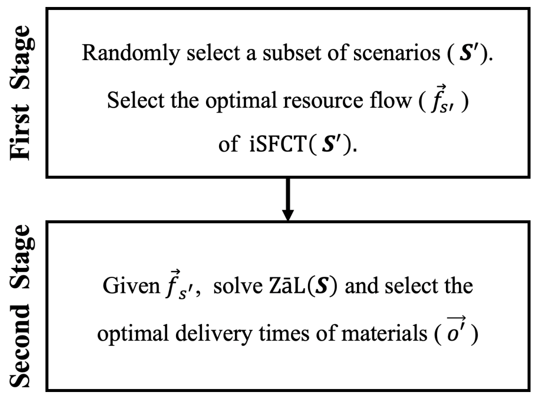

First Stage
Randomly select a subset of scenarios (S').
Select the optimal resource flow ($\vec{f}_{S'}$)
of iSFCT(S').
Second Stage
Given $\vec{f}_{S'}$, solve ZāL(S) and select the
optimal delivery times of materials ($\vec{o'}$)

Figure 2.9: Summary of iSAAD.

Note that iSAAD distinguishes activities with different rate of inventory holding cost while determining the resource flow, as it solves iSFCT($\mathcal{S}'$) in the first stage and concurrently determines the optimal resource flow and delivery times over $\mathcal{S}'$. The second stage of iSAAD, given the flow of resources, refines the delivery times by determining the optimal delivery times over $\mathcal{S}$.

Finally, we extend SAAD+ to iSAAD+ by replicating iSAAD rather than SAA for k times. The summary of iSAAD+ is presented in Figure 2.10.

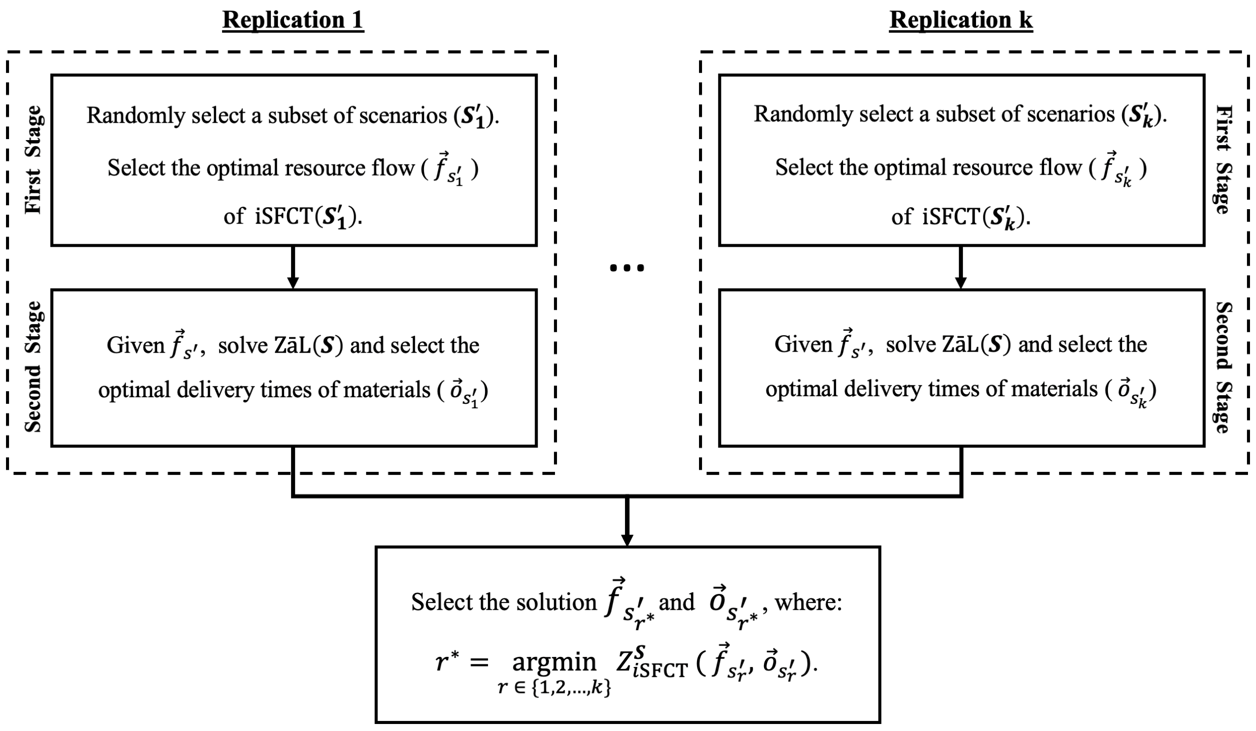

Replication 1
First Stage
Randomly select a subset of scenarios (S'_1).
Select the optimal resource flow ($\vec{f}_{S'_1}$)
of iSFCT(S'_1).
Second Stage
Given $\vec{f}_{S'}$, solve ZāL(S) and select the
optimal delivery times of materials ($\vec{o}_{S'_1}$)
...
Replication k
Randomly select a subset of scenarios (S'_k).
Select the optimal resource flow ($\vec{f}_{S'_k}$)
of iSFCT(S'_k).
First Stage
Given $\vec{f}_{S'}$, solve ZāL(S) and select the
optimal delivery times of materials ($\vec{o}_{S'_k}$)
Second Stage
Select the solution $\vec{f}_{S'_{r^*}}$ and $\vec{o}_{S'_{r^*}}$, where:
$r^* = \underset{r \in \{1,2,\ldots,k\}}{\text{argmin}} \; Z^{S}_{\text{iSFCT}}(\vec{f}_{S'_r}, \vec{o}_{S'_r})$.

Figure 2.10: Summary of iSAAD+.

2.5 Computational Experiments

We use the benchmark library PSPLIB instances with 30 and 60 activities (32 and 62 activities when dummy activities are included) to analyze our proposed solution approaches. In PSPLIB, there are 48 categories of instances for the RCPSP and 10 instances of each category for projects with 30 and 60 activities, which results in 960 instances in total (480 instances with 30 activities and 480 instances with 60 activities). These categories are differentiated based on three characteristics of the problem – network complexity (NC), resource factor (RF_r) and resource strength (RS_r).

NC describes the complexity of networks and it is defined as the average number of non-redundant arcs (precedence constraints) per node (activity, including dummy activities). RF_r provides information on the average variety of required resources per activity, and finally, RS_R represents the average number of units of resources per activity. The 48 categories of instances in benchmark library PSPLIB are constructed by taking any combinations of NC, RF_r and RS_r that are presented in the Table 2.1.

Table 2.1: Parameters that generate 48 different categories of instances in PSPLIB.

parameters	levels			
NC	1.50	1.80	2.10	
RF_r	0.25	0.50	0.75	1.00
RS_r	0.20	0.50	0.70	1.00

In PSPLIB, activity durations are deterministic. After our discussions with consultants and project and supply chain managers (see Jabbari and Kaminsky [37]), we defined the stochastic activity durations as having Weibull distributions with coefficient of variation of 0.22. The mean of the distributions are assumed to be equal to the deterministic activity durations provided by PSPLIB.

As PSPLIB instances are designed for the RCPSP, they do not offer any data regarding the inventory holding cost. Therefore, we extent them by randomly generating the per unit time inventory holding cost (w_j for $j \in \mathcal{A}$) with a distribution that is zero with a probability of 0.5, as the per unit time inventory holding cost of materials of some activities are negligible, and uniformly distributed between zero and one with a probability of 0.5.

In this section, the cardinality of set $\mathcal{S}$ in SFCT($\mathcal{S}$) and iSFCT($\mathcal{S}$) presented in Subsections 2.3.2 and 2.3.3, respectively, is set to 2000. We assumed 2000 is sufficiently large to make the error negligible. To generate the scenarios, for each activity, we sampled a set of 2000 observations from the Weibull distribution, adjust the mean to equal the deterministic duration of that activity, and setting the coefficient of variation equal to 0.22. The cardinality of subset $\mathcal{S}'$ in SFCT($\mathcal{S}'$) and iSFCT($\mathcal{S}'$) is set to be 20. Also, the number of replications in SAAD+ and iSAAD+, k', is set to be 30.

As we discussed in Section 2.4, due to the complexity and the size of the SFCT($\mathcal{S}$) when $|\mathcal{S}|$ is large (2000 in this section), we are unable to solve it directly. Therefore, we proposed various deterministic and SAA-based heuristics that provides feasible solutions for SFCT($\mathcal{S}$) and iSFCT($\mathcal{S}$). Let UB represent the objective function value of a feasible solution in SFCT or in iSFCT. In Subsection 2.4.1, we showed that the optimal objective function value of the RCPSP in which activity durations are assumed to be equal to their means is a lower bound for the SRCPSP. Let LB represents this lower bound. Then, for each instance, we can calculate the optimality gap which is the maximum possible percentage distance that the upper bound has from the optimal solution. The optimality gap is calculated as: $(UB - LB)/LB$.

To to find the LB (the optimal objective function value of RCPSP), we use IBM ILOG CP Optimizer (version 2.12.182) to solve the CP formulation of RCPSP as is explained in Laborie [50]. It optimally solved 480 (out of 480) instances with 30 activities within the time limit of 180 seconds, and 450 (out of 480) instances with 60 activities within the time limit of 600 seconds. For each of those 30 instances with 60 activities that are not solved optimally within the time limit, we set the LB to be the best lower bound that has been reported in the literature (available here).

All computations in this section are performed using a 2.6 GHz 6-Core Intel Core i7, and all algorithms are implemented in Python 3.7.4.

In Subsection 2.5.1, we analyze the performances of proposed heuristics on the SFCT, and in Subsection 2.5.2, we analyze the performances of the extended heuristics on the iSFCT.

2.5.1 Computational Analysis of SFCT

This subsection is divided into two sections. In the first part, we consider all 960 instances, and compare the performance of the deterministic heuristic on them. However, as SAA-based heuristics are not able to find feasible solution for all 960 instances, we do not consider them here.

We select a set of 282 instances with 30 activities for which Gurobi (version 9.0.0) with default settings can solve SFCT($\mathcal{S}'$), where $|\mathcal{S}'|= 20$, to optimality within the time limit of 1800 seconds. Figure 2.11 represents characteristics (NC, RF_r and RS_r) of these 282 instances. This figure shows that instances with larger NC, larger RS_r and smaller RF_r are easier to solve with SAA-based heuristics, and thus, we conclude that SAA and SAAD+ are more suitable for the instances with these characteristics. In the second part, we compare the performance of all heuristics on these 282 instances.

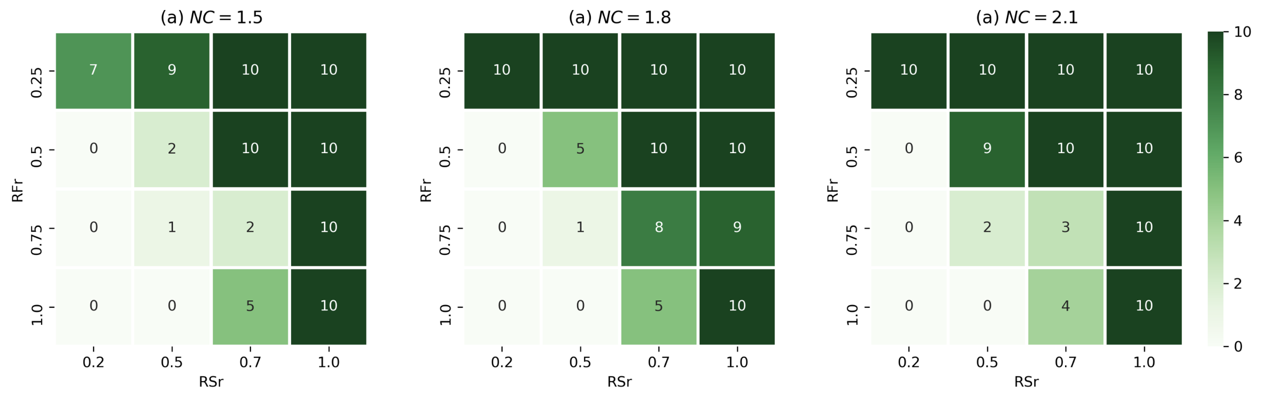


Figure 2.11: Characteristics of the selected 282 instances.

Deterministic Heuristics: NDH, EACH and PAH

Table 2.2 provides the average, median, minimum and maximum optimality gap and the total solution times when different deterministic heuristics are used to find feasible solutions for SFCT. This table also provides the number of instances out of 480 for which each heuristic provides the best solution among all three heuristics. Note that for some instances, both EACH and PAH provide identical solutions, and this is why the summation of the numbers in the last row of the table for projects with 30 and 60 activities is more than 480.

Table 2.2: Optimality gap and total solution time of deterministic heuristics.

		30 Activities			60 Activities		
	Approach	NDH	EACH	PAH	NDH	EACH	PAH
Optimality gap	Average	0.0915	0.0558	0.0492	0.1418	0.0817	0.0706
	Median	0.0910	0.0532	0.0473	0.1425	0.0808	0.0676
	Minimum	0.0121	0.0000	0.0012	0.0796	0.0104	0.0025
	Maximum	0.1633	0.1236	0.1193	0.2202	0.1506	0.1462
Total solution time (sec)	Average	2.02	2.19	2.26	70.28	90.56	147.66
	Median	0.05	0.24	0.30	0.09	11.47	11.94
	Minimum	0.03	0.06	0.04	0.06	0.63	0.08
	Maximum	177.40	177.51	177.55	600.09	607.06	6387.81
Best solution (out of 480)*		0	81	415	0	27	453

* For some instances more than one approach finds the best solution.

Tables 2.3 and 2.4, respectively, provide extensive details on the average solution time for instances of each of the 48 categories of projects with 30 and 60 activities. The solution times in these tables are split into two parts. In each deterministic heuristic, first, a constraint programming model is solved. The solution time for the CP model is presented in column 'Avg CP time'. Then, a flow of resource is selected among alternative optimal solution. The solution time for selecting a resource flow is presented in column 'Avg FL time'. Finally, the optimality gap of each of the heuristics is presented in column 'Avg OptGap'.

Table 2.3: Average optimality gap and average solution time of deterministic heuristics for each category of instances with 30 activities.

					NDH		EACH		PAH	
# [1]	NC	RFr	RSr	Avg CP time [2]	Avg FL time [3]	Avg OptGap [4]	Avg FL time	Avg OptGap	Avg FL time	Avg OptGap
all [5]	-	-	-	2.01	0.01	0.0915	0.18	0.0558	0.25	0.0492
1	1.5	0.25	0.2	0.14	0.01	0.0970	0.08	0.0553	0.01	0.0498
2			0.5	0.04	0.01	0.1078	0.09	0.0482	0.02	0.0400
3			0.7	0.03	0.01	0.0773	0.08	0.0295	0.01	0.0237
4			1.0	0.03	0.01	0.0859	0.09	0.0410	0.01	0.0258
5		0.5	0.2	0.99	0.01	0.0832	0.13	0.0632	0.06	0.0587
6			0.5	0.26	0.01	0.1125	0.20	0.0779	0.10	0.0710
7			0.7	0.05	0.01	0.1076	0.21	0.0634	0.13	0.0574
8			1.0	0.04	0.01	0.0948	0.19	0.0450	0.11	0.0288
9		0.75	0.2	5.56	0.01	0.0677	0.17	0.0590	0.26	0.0520
10			0.5	0.32	0.01	0.1289	0.30	0.0995	0.30	0.0931
11			0.7	0.05	0.01	0.1048	0.29	0.0787	0.45	0.0677
12			1.0	0.04	0.01	0.1048	0.36	0.0484	0.50	0.0371
13		1.0	0.2	21.58	0.01	0.0835	0.18	0.0699	0.31	0.0688
14			0.5	0.79	0.01	0.1173	0.32	0.0891	0.51	0.0863
15			0.7	0.19	0.01	0.0994	0.38	0.0566	0.98	0.0503
16			1.0	0.04	0.01	0.1027	0.27	0.0386	0.78	0.0275
17	1.8	0.25	0.2	0.14	0.01	0.0873	0.08	0.0485	0.01	0.0432
18			0.5	0.03	0.01	0.0919	0.08	0.0385	0.01	0.0319
19			0.7	0.03	0.01	0.1030	0.10	0.0508	0.02	0.0448
20			1.0	0.03	0.01	0.1037	0.06	0.0462	0.02	0.0344
21		0.5	0.2	0.89	0.01	0.0725	0.10	0.0550	0.06	0.0507
22			0.5	0.20	0.01	0.1041	0.22	0.0718	0.12	0.0675
23			0.7	0.04	0.01	0.0779	0.19	0.0462	0.12	0.0395
24			1.0	0.04	0.01	0.0995	0.16	0.0537	0.11	0.0357
25		0.75	0.2	2.67	0.01	0.0720	0.14	0.0620	0.19	0.0577
26			0.5	0.05	0.01	0.1017	0.22	0.0709	0.33	0.0632
27			0.7	0.04	0.01	0.0977	0.27	0.0577	0.64	0.0486
28			1.0	0.04	0.01	0.0858	0.32	0.0360	0.38	0.0314
29		1.0	0.2	25.66	0.01	0.0711	0.15	0.0601	0.27	0.0594
30			0.5	0.75	0.01	0.1214	0.20	0.0950	0.39	0.0911
31			0.7	0.17	0.01	0.0995	0.38	0.0622	1.02	0.0556
32			1.0	0.04	0.01	0.0969	0.27	0.0413	0.72	0.0373
33	2.1	0.25	0.2	0.06	0.01	0.0841	0.06	0.0448	0.01	0.0424
34			0.5	0.03	0.01	0.0883	0.06	0.0452	0.01	0.0341
35			0.7	0.03	0.01	0.0789	0.07	0.0348	0.01	0.0305
36			1.0	0.03	0.01	0.0730	0.06	0.0319	0.01	0.0254
37		0.5	0.2	0.85	0.01	0.0662	0.11	0.0488	0.06	0.0447
38			0.5	0.04	0.01	0.0897	0.12	0.0591	0.10	0.0532
39			0.7	0.04	0.01	0.0898	0.16	0.0509	0.14	0.0398
40			1.0	0.04	0.01	0.0920	0.11	0.0473	0.10	0.0354
41		0.75	0.2	2.71	0.01	0.0475	0.12	0.0367	0.13	0.0364
42			0.5	0.11	0.01	0.0874	0.25	0.0644	0.22	0.0591
43			0.7	0.04	0.01	0.0910	0.17	0.0672	0.24	0.0631
44			1.0	0.04	0.01	0.0966	0.14	0.0452	0.35	0.0410
45		1.0	0.2	30.89	0.01	0.0422	0.13	0.0367	0.22	0.0352
46			0.5	0.64	0.01	0.1073	0.17	0.0895	0.28	0.0874
47			0.7	0.10	0.01	0.1127	0.24	0.0727	0.38	0.0718
48			1.0	0.04	0.01	0.0829	0.17	0.0422	0.58	0.0323

[1] the category number
[2] the resource flow selection solution time
[3] the solution time of CP model
[4] the optimality gap
[5] the average over all instances

Table 2.4: Average optimality gap and average solution time of deterministic heuristics for each category of instances with 60 activities.

					NDH		EACH		PAH	
# [1]	NC	RFr	RSr	Avg CP time [2]	Avg FL time [3]	Avg OptGap [4]	Avg FL time	Avg OptGap	Avg FL time	Avg OptGap
all [5]	-	-	-	70.24	0.04	0.1418	20.32	0.0817	77.42	0.0706
1	1.5	0.25	0.2	0.46	0.03	0.1420	1.38	0.0745	0.08	0.0653
2			0.5	0.04	0.03	0.1533	11.00	0.0643	0.12	0.0461
3			0.7	0.14	0.03	0.1400	6.92	0.0642	0.12	0.0444
4			1.0	0.04	0.03	0.1336	7.56	0.0448	0.11	0.0231
5		0.5	0.2	29.73	0.04	0.1503	1.87	0.1172	0.45	0.1095
6			0.5	0.19	0.04	0.1637	19.21	0.1119	2.30	0.0972
7			0.7	0.04	0.03	0.1473	23.49	0.0706	13.60	0.0608
8			1.0	0.04	0.03	0.1474	22.90	0.0587	10.60	0.0464
9		0.75	0.2	513.89	0.05	0.1460	2.00	0.1222	1.95	0.1190
10			0.5	0.05	0.04	0.1613	64.66	0.1039	37.01	0.0955
11			0.7	0.04	0.04	0.1561	91.53	0.0868	158.26	0.0748
12			1.0	0.04	0.04	0.1447	91.89	0.0483	453.68	0.036
13		1.0	0.2	600.02	0.06	0.1498	3.05	0.1237	5.87	0.1204
14			0.5	69.37	0.05	0.1703	21.23	0.1272	133.45	0.1195
15			0.7	0.05	0.05	0.1458	46.02	0.0645	373.22	0.0534
16			1.0	0.05	0.04	0.1516	44.44	0.0648	470.74	0.0498
17	1.8	0.25	0.2	0.23	0.03	0.1427	1.13	0.0818	0.07	0.0719
18			0.5	0.04	0.03	0.1369	3.74	0.0624	0.14	0.0463
19			0.7	0.03	0.03	0.1382	8.56	0.0640	0.11	0.0397
20			1.0	0.03	0.03	0.1309	2.52	0.0476	0.14	0.0316
21		0.5	0.2	5.96	0.04	0.1276	1.43	0.0990	0.49	0.0908
22			0.5	0.15	0.03	0.1543	8.15	0.1055	4.47	0.0920
23			0.7	0.04	0.03	0.1475	29.07	0.0760	10.65	0.0576
24			1.0	0.04	0.03	0.1391	16.10	0.0631	5.41	0.0473
25		0.75	0.2	511.48	0.05	0.1279	1.88	0.1074	1.07	0.1028
26			0.5	1.16	0.04	0.1573	11.40	0.1095	27.51	0.1012
27			0.7	0.04	0.04	0.1493	96.51	0.0789	220.21	0.0698
28			1.0	0.04	0.04	0.1314	44.82	0.0512	238.14	0.0400
29		1.0	0.2	600.02	0.06	0.1301	2.20	0.1101	3.47	0.1059
30			0.5	102.31	0.05	0.1587	31.53	0.1127	60.99	0.1063
31			0.7	0.05	0.05	0.1558	38.44	0.0848	203.38	0.0712
32			1.0	0.05	0.05	0.1287	63.13	0.0420	863.34	0.0340
33	2.1	0.25	0.2	0.47	0.03	0.1196	1.25	0.0589	0.07	0.0480
34			0.5	0.05	0.03	0.1377	2.14	0.0714	0.08	0.0567
35			0.7	0.04	0.03	0.1347	5.23	0.0618	0.13	0.0476
36			1.0	0.03	0.03	0.1373	2.32	0.0570	0.12	0.0365
37		0.5	0.2	8.54	0.04	0.1165	1.13	0.0920	0.47	0.0866
38			0.5	0.39	0.03	0.1540	7.55	0.1082	2.68	0.0987
39			0.7	0.04	0.03	0.1361	11.74	0.0788	5.54	0.0663
40			1.0	0.04	0.03	0.1294	7.38	0.0509	7.41	0.0341
41		0.75	0.2	224.37	0.05	0.1124	1.73	0.0947	1.35	0.0918
42			0.5	1.45	0.04	0.1477	9.35	0.1105	20.33	0.1021
43			0.7	0.05	0.04	0.1325	20.74	0.0753	61.22	0.0670
44			1.0	0.04	0.04	0.1291	13.03	0.0530	73.06	0.0400
45		1.0	0.2	600.02	0.06	0.1196	1.83	0.1023	2.75	0.1002
46			0.5	100.08	0.05	0.1602	9.67	0.1235	16.31	0.1190
47			0.7	0.05	0.05	0.1534	32.06	0.0944	111.96	0.0864
48			1.0	0.05	0.04	0.1282	28.38	0.0457	111.58	0.0379

[1] the category number
[2] the resource flow selection solution time
[3] the solution time of CP model
[4] the optimality gap
[5] the average over all instances

For each instance, define the "best optimality gap" as the optimality gap of the best solution among the solutions provided by our deterministic heuristics (NDH, EACH and PAH) for that instance. Figure 2.12 represents the average best optimality gap among 10 instances of each category of instances. This figure suggests that the optimality gap is very sensitive to the instances' characteristic.

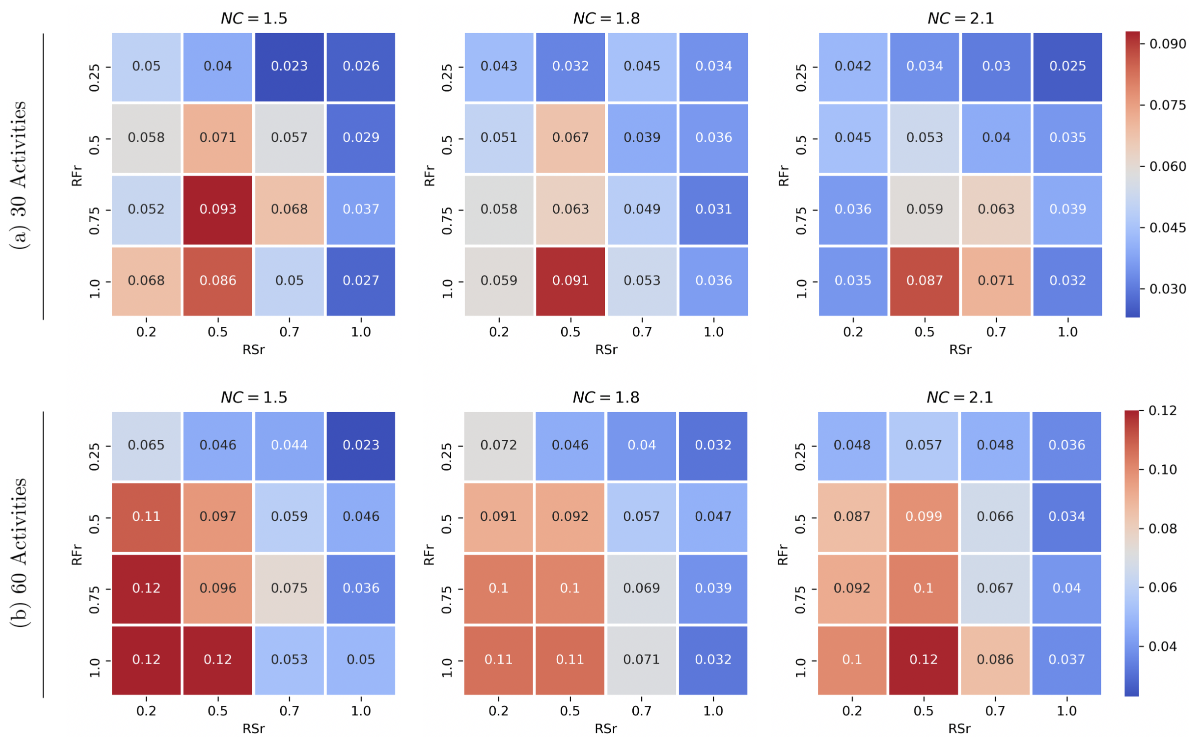


Figure 2.12: Impact of NC, RF_r and RS_r on the best optimality gap for SFCT for instances with (a): 30 activities and (b): 60 activities.

Figure 2.13 illustrates the sensitivity analyses over the characteristics of the instances (NC, RF_r and RS_r) on the best optimality gap. This figure suggests that the optimality gap is not very sensitive to the changes in the NC. It also suggests the optimality gap increases as RF_r increases and as RS_r decreases. This figure also suggests that the impact of RF_r and RS_r escalates as the number of activities in the project increases.

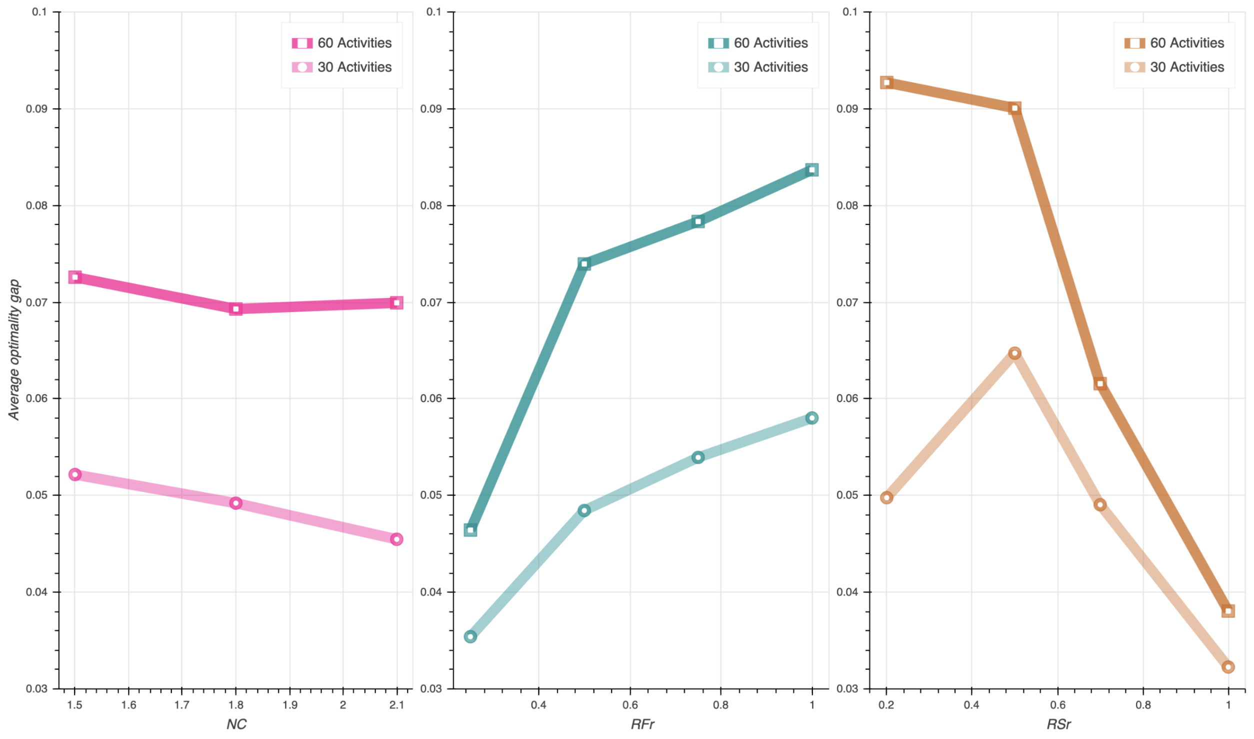


Figure 2.13: Sensitivity analysis of the best optimality gap over NC, RF_r and RS_r for instances with 30 and 60 activities.

Table 2.2 shows that for most instances, PAH finds the best solutions, and for some instances, EACH finds the best solution. When the choices of heuristics is limited to the deterministic ones, as Tables 2.2-2.4 show, it is best for practitioners to use both EACH and PAH and select the best solution. However, if the computation power is limited, then PAH is the best choice as on average, it provides better solutions. Note that one might prefer EACH over PAH as the maximum solution time of EACH is significantly less than the maximum solution time of PAH, while their average optimality gaps are comparable.

All Heuristics: NDH, EACH, PAH, SAA and SAAD+

We define the probabilistic optimality gap as $(UB - LB(\beta))/LB(\beta)$. In Table 2.5, β is set to be 97.5%, which means that the probabilistic lower bound is in fact an actual lower bound with the probability of at least 97.5%. This table provides the average, median, minimum and maximum optimality gap and total solution times, and the median and maximum probabilistic optimality gap of 282 instances with 30 activities when different heuristics are used to find feasible solutions for the SFCT. This table also demonstrates the number of instances (out of 282) that each approach finds the best solution.

Table 2.5: Optimality gap and total solution time of all proposed heuristics.

	Approach	NDH	EACH	PAH	SAA	SAAD+
Optimality Gap	Average	0.0921	0.0457	0.0376	0.0297	0.0269
	Median	0.0911	0.0433	0.0341	0.0261	0.0236
	Minimum	0.0456	0.0000	0.0012	0.0003	0.0000
	Maximum	0.1594	0.1039	0.0989	0.0924	0.0893
Probabilistic Optimality Gap [1]	Median	0.0712	0.0247	0.0158	0.0100	0.0080
	Maximum	0.1324	0.0743	0.0597	0.0347	0.0194
Total Solution Time (sec)	Average	0.05	0.21	0.27	29.55	1901.70
	Median	0.05	0.16	0.14	2.61	418.53
	Minimum	0.03	0.06	0.04	0.08	38.30
	Maximum	0.99	1.13	2.39	1049.07	15567.77
Best Solution (out of 282) [2]		0	3	23	22	263

[1] $\beta = 0.975$

[2] For some instances more than one approach finds the best solution.

Table 2.6 provides extensive details on the average solution time of all proposed heuristics on each category of the selected 282 instances. The solution times in this table are split into two parts.

Figure 2.14, for each instance, depicts the lower bound (pink line), NDH solution which represents the current practice (purple line), the objective function value of SFCT($\mathcal{S}'_r$) for $r \in \{1, 2, ..., 30\}$ (grey lines), SAAD+ solution (green line), and the probabilistic lower bound that is an actual lower bound with the probability of 97.5% (orange line). In this figure, the light blue area represents the histogram of the the objective function value of SFCT($\mathcal{S}'_r$) for $r \in \{1, 2, ..., 30\}$, and the red t-distribution PDF shows the estimated distribution of the $E_{\mathcal{S}'}[Z^{\mathcal{S}'}_{SFCT}]$, and the yellow area represents the 95% confidence interval of the estimated distribution. This figure illustrates instances $(1-2)$, $(17-4)$ and $(47-8)$. The same figures for all 282 instances (in PNG format) are available here.

Table 2.6: Average solution time of all proposed heuristics for each category of instances.

				Average Solution Time				
#*	NC	RFr	RSr	NDH	EACH	PAH	SAA	SAAD+
1	1.5	0.25	0.20	0.09	0.17	0.10	31.79	2291.32
2			0.50	0.04	0.12	0.05	2.71	348.36
3			0.70	0.04	0.11	0.05	1.18	174.53
4			1.00	0.04	0.12	0.05	0.39	97.89
6		0.5	0.50	0.47	0.59	0.57	305.93	10824.87
7			0.70	0.06	0.26	0.18	108.74	6651.02
8			1.00	0.04	0.22	0.14	2.22	400.55
10		0.75	0.50	0.05	0.58	0.40	158.28	6575.84
11			0.70	0.05	0.23	0.66	9.27	1448.46
12			1.00	0.05	0.40	0.54	7.73	1168.65
15		1.0	0.70	0.06	0.54	1.16	80.97	6360.34
16			1.00	0.05	0.31	0.82	5.03	789.67
17	1.8	0.25	0.20	0.14	0.21	0.15	11.00	894.20
18			0.50	0.04	0.12	0.05	1.45	204.53
19			0.70	0.04	0.13	0.05	0.80	142.49
20			1.00	0.04	0.09	0.05	0.20	64.83
22		0.5	0.50	0.05	0.29	0.12	73.77	5030.81
23			0.70	0.04	0.23	0.16	182.79	4760.39
24			1.00	0.04	0.20	0.14	12.11	1332.87
26		0.75	0.50	0.05	0.19	0.30	27.90	4041.65
27			0.70	0.05	0.30	0.74	60.36	4462.54
28			1.00	0.05	0.38	0.44	5.56	1102.73
31		1.0	0.70	0.06	0.44	0.84	24.99	3161.39
32			1.00	0.06	0.31	0.76	10.61	1846.55
33	2.1	0.25	0.20	0.06	0.12	0.07	3.33	551.32
34			0.50	0.04	0.10	0.04	0.74	161.94
35			0.70	0.04	0.10	0.05	0.62	113.34
36			1.00	0.04	0.09	0.04	0.15	52.22
38		0.5	0.50	0.05	0.17	0.15	71.74	5133.26
39			0.70	0.05	0.20	0.19	11.33	1652.43
40			1.00	0.05	0.15	0.15	2.06	362.76
42		0.75	0.50	0.05	0.27	0.20	160.21	11730.20
43			0.70	0.05	0.18	0.21	212.47	9456.76
44			1.00	0.05	0.18	0.39	3.43	615.68
47		1.0	0.70	0.06	0.25	0.41	133.64	9694.60
48			1.00	0.06	0.21	0.62	4.05	636.50

* the category number

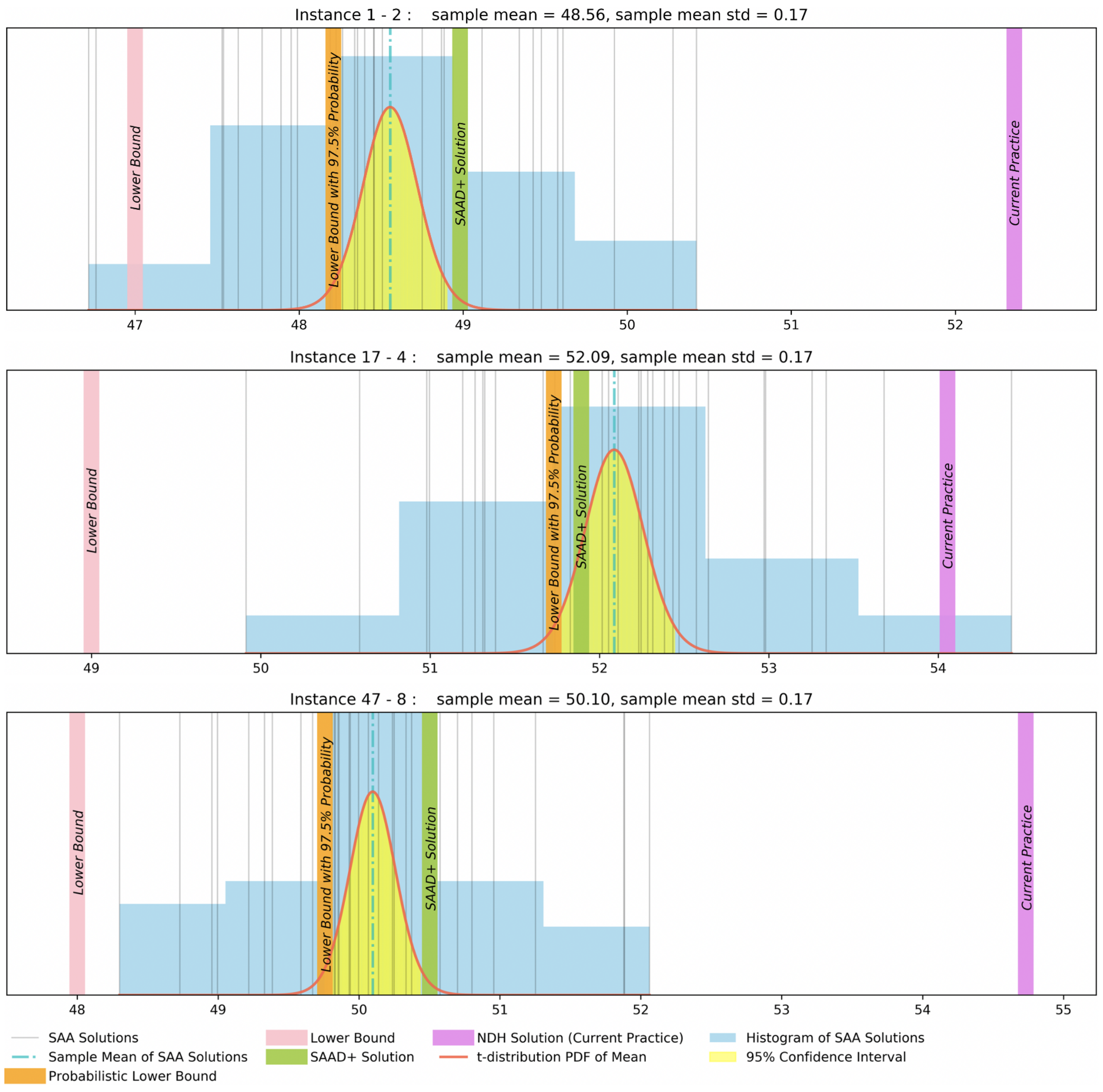


Figure 2.14: Illustration of the lower bound, the probabilistic lower bound, SAA solutions, SAAD+ solution, and NDH solution (indicator of current practice in real world).

We cannot be sure if the probabilistic lower bounds in Figure 2.14 are in fact actual lower bounds (there is at most 2.5% chance that each one is not an actual lower bound). Nevertheless, the probabilistic lower bounds reveal that the majority of the optimality gaps that we report are due to the loose lower bounds and the actual optimality gaps are most likely much smaller than what we reported in Table 2.5.

Table 2.7 provides extensive details on the average optimality gap and the median of probabilistic optimality gap, where the probability that the probabilistic lower bound is in fact an actual lower bound is set to be 97.5% ($\beta = 97.5\%$), for all proposed heuristics on each category of the selected 282 instances.

Table 2.7: Average optimality gap and median probabilistic optimality gap of deterministic heuristics for each category of instances.

				Average Optimality Gap					Median Probabilistic Optimality Gap [1]				
# [2]	NC	RFr	RSr	NDH	EACH	PAH	SAA	SAAD+	NDH	EACH	PAH	SAA	SAAD+
1	1.5	0.25	0.2	0.0950	0.0500	0.0450	0.0360	0.0340	0.0660	0.0230	0.0200	0.0110	0.0090
2			0.5	0.1090	0.0460	0.0370	0.0320	0.0300	0.0810	0.0180	0.0150	0.0110	0.0060
3			0.7	0.0770	0.0300	0.0240	0.0150	0.0140	0.0670	0.0240	0.0160	0.0080	0.0070
4			1	0.0860	0.0410	0.0260	0.0180	0.0160	0.0790	0.0290	0.0140	0.0080	0.0050
6		0.5	0.5	0.0920	0.0520	0.0510	0.0520	0.0470	0.0540	0.0160	0.0140	0.0150	0.0100
7			0.7	0.1080	0.0630	0.0570	0.0450	0.0400	0.0730	0.0350	0.0290	0.0150	0.0120
8			1	0.0950	0.0450	0.0290	0.0170	0.0150	0.0840	0.0360	0.0140	0.0090	0.0070
10		0.75	0.5	0.1140	0.0570	0.0480	0.0060	0.0020	0.1260	0.0680	0.0600	0.0160	0.0130
11			0.7	0.0590	0.0330	0.0170	0.0010	0.0000	0.0680	0.0420	0.0260	0.0100	0.0100
12			1	0.1050	0.0480	0.0370	0.0320	0.0240	0.0910	0.0310	0.0180	0.0160	0.0080
15		1.0	0.7	0.0940	0.0470	0.0410	0.0170	0.0100	0.0900	0.0380	0.0360	0.0120	0.0090
16			1	0.1030	0.0390	0.0280	0.0180	0.0160	0.0950	0.0330	0.0180	0.0110	0.0090
17	1.8	0.25	0.2	0.0870	0.0490	0.0430	0.0380	0.0360	0.0560	0.0190	0.0130	0.0090	0.0090
18			0.5	0.0920	0.0390	0.0320	0.0260	0.0260	0.0780	0.0180	0.0100	0.0070	0.0070
19			0.7	0.1030	0.0510	0.0450	0.0380	0.0350	0.0730	0.0220	0.0180	0.0110	0.0090
20			1	0.1040	0.0460	0.0340	0.0330	0.0320	0.0820	0.0180	0.0090	0.0080	0.0070
22		0.5	0.5	0.1020	0.0680	0.0640	0.0530	0.0500	0.0580	0.0230	0.0210	0.0140	0.0120
23			0.7	0.0780	0.0460	0.0400	0.0340	0.0300	0.0540	0.0180	0.0160	0.0130	0.0090
24			1	0.1000	0.0540	0.0360	0.0270	0.0230	0.0890	0.0380	0.0240	0.0120	0.0100
26		0.75	0.5	0.1250	0.0530	0.0440	0.0110	0.0080	0.1290	0.0570	0.0480	0.0150	0.0120
27			0.7	0.0910	0.0490	0.0380	0.0260	0.0210	0.0760	0.0350	0.0270	0.0090	0.0080
28			1	0.0810	0.0310	0.0260	0.0170	0.0140	0.0760	0.0250	0.0180	0.0090	0.0060
31		1.0	0.7	0.0880	0.0380	0.0350	0.0150	0.0110	0.0850	0.0190	0.0330	0.0150	0.0070
32			1	0.0970	0.0410	0.0370	0.0270	0.0250	0.0790	0.0230	0.0170	0.0110	0.0080
33	2.1	0.25	0.2	0.0840	0.0450	0.0420	0.0390	0.0370	0.0550	0.0140	0.0090	0.0080	0.0070
34			0.5	0.0880	0.0450	0.0340	0.0290	0.0280	0.0650	0.0150	0.0080	0.0060	0.0060
35			0.7	0.0790	0.0350	0.0310	0.0240	0.0230	0.0650	0.0140	0.0090	0.0070	0.0070
36			1	0.0730	0.0320	0.0250	0.0220	0.0210	0.0620	0.0160	0.0110	0.0090	0.0090
38		0.5	0.5	0.0900	0.0590	0.0540	0.0490	0.0440	0.0530	0.0250	0.0200	0.0110	0.0070
39			0.7	0.0900	0.0510	0.0400	0.0350	0.0320	0.0630	0.0280	0.0140	0.0110	0.0090
40			1	0.0920	0.0470	0.0350	0.0290	0.0270	0.0730	0.0270	0.0130	0.0100	0.0080
42		0.75	0.5	0.0680	0.0390	0.0350	0.0250	0.0190	0.0580	0.0300	0.0260	0.0160	0.0100
43			0.7	0.1070	0.0750	0.0690	0.0620	0.0590	0.0650	0.0260	0.0240	0.0130	0.0130
44			1	0.0970	0.0450	0.0410	0.0370	0.0330	0.0690	0.0190	0.0140	0.0120	0.0070
47		1.0	0.7	0.1130	0.0620	0.0620	0.0460	0.0440	0.0770	0.0310	0.0330	0.0160	0.0120
48			1	0.0830	0.0420	0.0320	0.0230	0.0210	0.0720	0.0240	0.0190	0.0070	0.0060

[1] $\beta = 0.975$

[2] the category number

Figure 2.15, for each instance, depicts the best solution among the solutions of the deterministic heuristics (red line), the solutions provided by 30 replication of SAA (gold lines), and the solution provided by SAAD+ (green line). Note that the solution provided by SAAD+ is the best solution among the 30 replications of SAA. This figure illustrates only nine instances. The same Figures for all 282 instances (in PNG format) are available here.

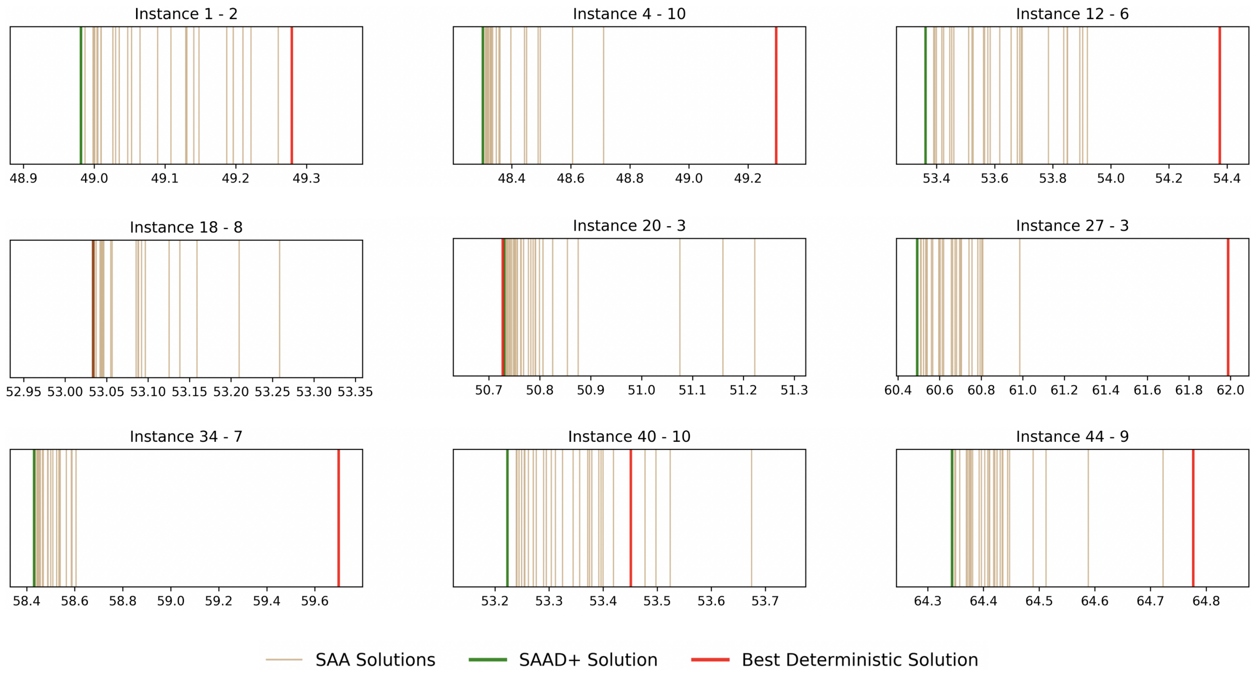


Figure 2.15: SAAD+ solution, solutions of 30 replication of SAA and the best solution provided by the our deterministic heuristics.

Out of 282 instances, in 125 instances, every replication of SAA results in a better solution than the best deterministic solution (e.g., instance $(4 - 10)$), in 131 instances (e.g., instance $(40 - 10)$) the best deterministic solution is better than some but not all SAA solutions, in 7 instances (e.g., instance $(18 - 8)$) the best deterministic solution is equal to SAAD+ solution, and finally, in 19 instances (e.g., instance $(20 - 3)$) the best deterministic solution is better than SAAD+ solution.

Table 2.5 and Figure 2.15 show that, when sufficient computational power is available, it is best for a practitioner to use SAAD+, as its performance is significantly superior to the performance of other proposed heuristics. The limitation of SAAD+ is that it cannot be applied to projects with many activities. Among projects with 30 activities, SAAD+ is most suitable for projects with large RS_r and small RF_r, as it is easier to optimally solve SFCT$(\mathcal{S}')$ for those projects.

The complexity of SAAD+ increases linearly as the number of replications increases, and thus, we can choose large numbers of replications without significantly increasing the required computational effort. Figure 2.16 shows the impact of increasing the number of replications (from 1 to 30) on the average optimality gap of SAAD+. In Lemma 9, we proved the incremental benefit of increasing the number of replications is decreasing, which as well can be seen in this figure.

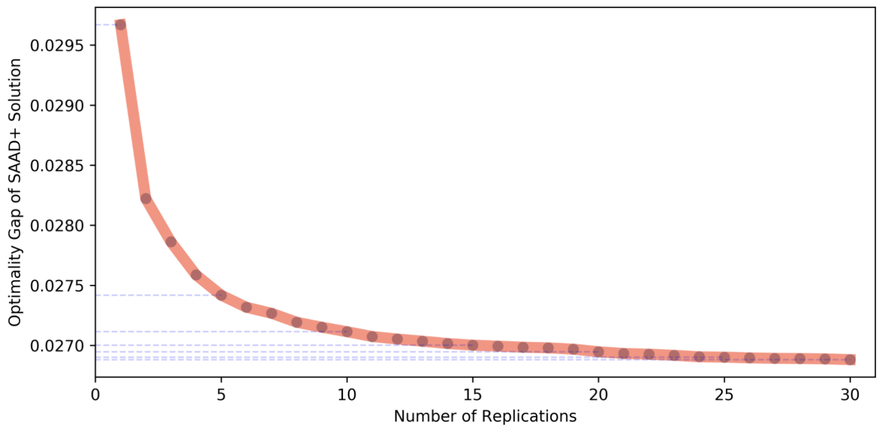


Figure 2.16: Average optimality gap of SAAD+ for different number of replications.

2.5.2 Computational Analysis of iSFCT

As in the previous subsection, we divide this subsection into two sections. In the first part, we analyze the performance of deterministic heuristics over all 960 instances. Then, in the second part, we compare the performance of all heuristics over the same selected 282 instances as in the previous subsection.

To illustrate the performance of the alternative heuristics on iSFCT($\mathcal{S}$), we add constraint 2.72 to the LP formulation of the ZāL($\mathcal{S}$). This constraint ensures the expected makespan of the project is equal to $\tilde{UB}$, where $\tilde{UB}$ is a predefined parameter. Also, we modify the objective function to only minimize the expected inventory holding cost (2.73).

$$\sum_{s\in\mathcal{S}} z^s_{n+1}/|\mathcal{S}| = \tilde{UB} \tag{2.72}$$

$$Minimize \quad \sum_{i\in\mathcal{A}} \omega_i \times \sum_{s\in\mathcal{S}} (z^s_i - o_i)/|\mathcal{S}| \tag{2.73}$$

We refer to this adjusted LP formulation of ZāL($\mathcal{S}$) as the AZāL($\mathcal{S}$). Then, for a given expected makespan, $\tilde{UB}$, and given the flow of resources that is selected in the first stage of the extended heuristics, we find the objective function value of the AZāL($\mathcal{S}$), which is the expected inventory holding cost. Note that the expected inventory holding cost is a function of the selected resource flow and the given expected makespan.

Deterministic Heuristics: iNDH, iEACH and iPAH

Let $\vec{f}_{\text{NDH}}$, $\vec{f}_{\text{EACH}}$ and $\vec{f}_{\text{PAH}}$ denote the flow of resources that NDH, EACH and PAH, respectively, select in their first stages. To compare the performance of all three deterministic heuristics, we select $\tilde{UB}$ so that given any of the $\vec{f}_{\text{NDH}}$, $\vec{f}_{\text{EACH}}$ and $\vec{f}_{\text{PAH}}$, AZāL($\mathcal{S}$) has a feasible solution. To ensure this, for each instance, we define U_3 as:

$$U_3 = \max\Big(Z^{\mathcal{S}}_{SFCT}(\vec{f}_{\text{NDH}}), Z^{\mathcal{S}}_{SFCT}(\vec{f}_{\text{EACH}}), Z^{\mathcal{S}}_{SFCT}(\vec{f}_{\text{PAH}})\Big). \tag{2.74}$$

Tables 2.8 and 2.9 represent the expected inventory holding cost and the total solution times of our deterministic heuristics when we set the expected makespan to be $1.005 \times U_3$ and $1.02 \times U_3$.

Table 2.8: Expected inventory holding cost of deterministic heuristics for instances with 30 activities.

	$\tilde{UB}$	$1.005 \times U_3$			$1.02 \times U_3$		
	Approach	iNDH	iEACH	iPAH	iNDH	iEACH	iPAH
Expected Total Inventory Holding Cost	Average	18.8214	7.2699	6.3993	10.3618	4.9976	4.4727
	Median	17.6826	5.1875	4.4213	9.7300	3.8367	3.3061
	Minimum	3.5011	0.1448	0.0831	1.7318	0.0986	0.0545
	Maximum	56.0612	43.6031	40.6248	32.8500	27.5585	26.7212
Total Solution Time (sec)	Average	24.54	8.63	11.12	22.87	9.01	11.44
	Median	22.15	6.77	9.33	20.13	7.10	9.77
	Minimum	11.65	1.72	2.36	12.34	1.84	2.47
	Maximum	203.39	185.00	186.09	198.53	184.42	188.94
Best Solution (out of 480)*		0	83	405	0	83	406

* For some instances more than one approach finds the best solution.

Table 2.9: Expected inventory holding cost of deterministic heuristics for instances with 60 activities.

	$\tilde{UB}$	$1.005 \times U_3$			$1.02 \times U_3$		
	Approach	iNDH	iEACH	iPAH	iNDH	iEACH	iPAH
Expected Total Inventory Holding Cost	Average	42.2537	9.2837	7.8018	21.9206	6.5326	5.5566
	Median	39.6311	5.8385	4.6427	20.2670	4.4313	3.4933
	Minimum	12.2353	0.1841	0.1551	5.5562	0.1332	0.1131
	Maximum	111.7436	67.1258	65.7836	59.9718	42.5978	41.7576
Total Solution Time (sec)	Average	205.24	129.80	194.71	196.38	130.14	195.83
	Median	134.17	55.97	67.89	126.96	57.33	69.30
	Minimum	74.72	14.42	15.12	76.43	15.01	14.38
	Maximum	838.51	673.03	6443.68	832.92	681.48	6448.06
Best Solution (out of 480)		0	32	448	0	38	442

From Tables 2.8 and 2.9, it is clear that both EACH and PAH are superior to NDH (with regard to both the expected inventory holding cost and the solution time), where NDH models current practice. To further compare the performance of EACH and PAH, we define U_2 as follows and compare the expected inventory holding costs by setting the expected makespan to be $1.005 \times U_2$ and $1.02 \times U_2$. Tables 2.10 and 2.11 summarize the results.

$$U_2 = \max\left(Z^{\mathcal{S}}_{SFCT}(\vec{f}_{\text{EACH}}), Z^{\mathcal{S}}_{SFCT}(\vec{f}_{\text{PAH}})\right) \tag{2.75}$$

Table 2.10: Expected inventory holding cost of iEACH and iPAH for instances with 30 activities.

	$\tilde{UB}$	$1.005 \times U_2$		$1.02 \times U_2$	
	Approach	iEACH	iPAH	iEACH	iPAH
Expected Total Inventory Holding Cost	Average	16.0999	12.9022	8.9599	7.7220
	Median	14.7334	11.4274	8.2339	6.9543
	Minimum	1.7970	0.2498	0.8456	0.1618
	Maximum	54.2814	50.5242	31.5095	30.4822
Total Solution Time (sec)	Average	8.01	11.00	8.45	10.79
	Median	6.15	9.33	6.50	8.95
	Minimum	1.67	2.32	1.88	2.28
	Maximum	184.64	185.89	184.46	188.74
Best Solution (out of 480)*		74	414	83	402

* For some instances more than one approach finds the best solution.

Table 2.11: Expected inventory holding cost of iEACH and iPAH for instances with 60 activities.

	$\tilde{UB}$	$1.005 \times U_2$		$1.02 \times U_2$	
	Approach	iEACH	iPAH	iEACH	iPAH
Expected Total Inventory Holding Cost	Average	33.2897	22.5116	17.3952	13.3990
	Median	30.3365	18.4800	15.8161	11.4682
	Minimum	3.5854	2.1721	1.4985	1.1205
	Maximum	107.1510	104.3751	57.0591	55.9068
Total Solution Time (sec)	Average	123.41	193.26	125.85	193.03
	Median	47.97	66.77	52.05	67.41
	Minimum	12.38	11.79	12.12	12.44
	Maximum	677.85	6421.83	679.57	6448.36
Best Solution (out of 480)		27	453	33	447

For any given resource flow, by solving AZāL($\mathcal{S}$) for different values of , we can create an efficient frontier that represents the trade-off between the expected inventory holding cost and the expected makespan of the project. Figure 2.17 represents the efficient frontier for instances $(4-2)$ and $(39-5)$ with 30 activities, and instances $(2-2)$ and $(22-5)$ with 60 activities. The same figures (in HTML format) for all 960 instances are available here.

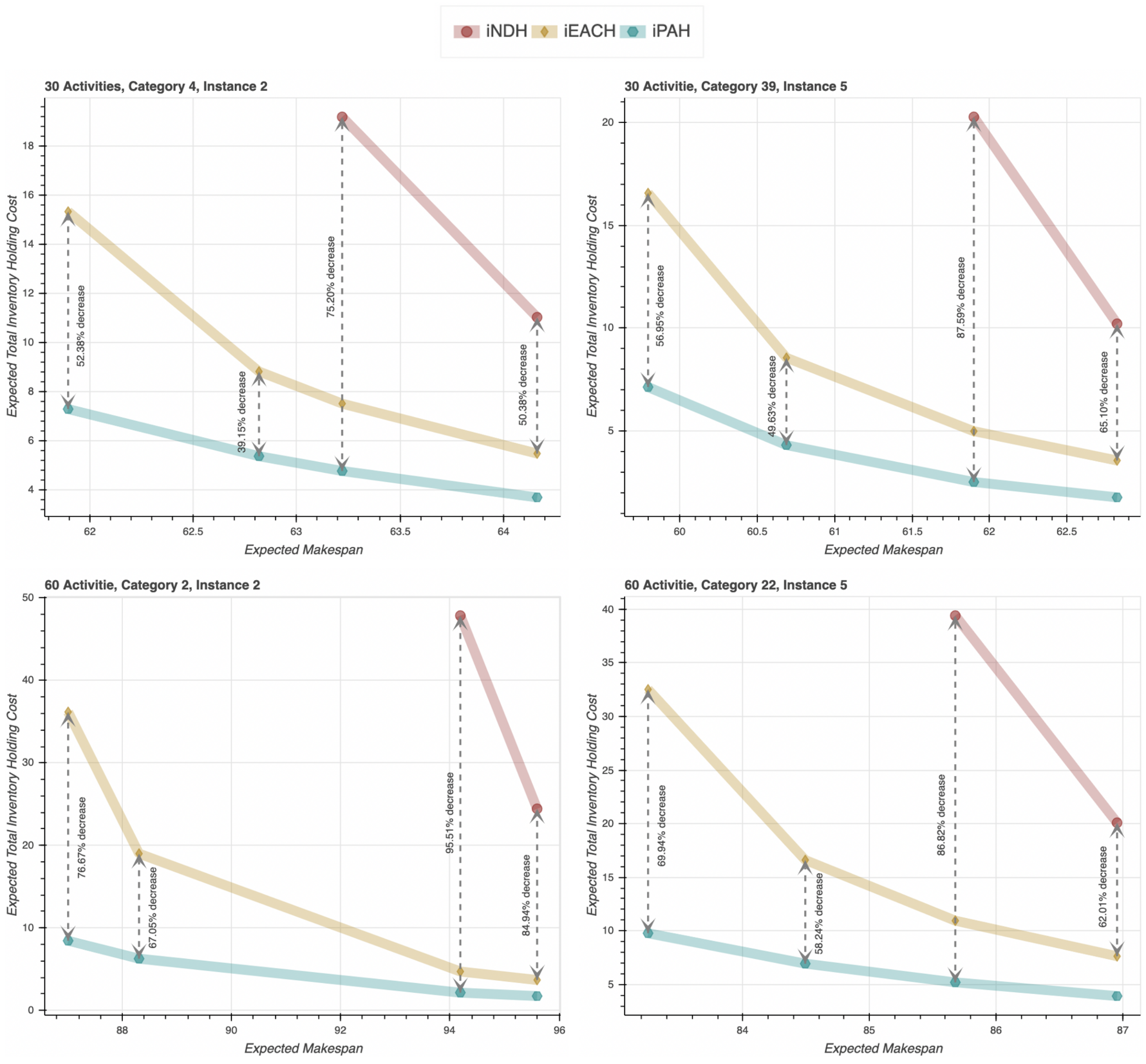


Figure 2.17: Efficient frontier of expected makespan vs. expected inventory holding cost.

Table 2.12 represents the average and median percentage of decrease in the expected inventory holding cost when the solution of one heuristic replaces the solution of the another heuristic. Note that for each instance, the percentage of decrease in the expected inventory holding cost when, for example, PAH solution replaces NDH solution is calculated as:

$$\frac{\mathcal{C}_{\text{NDH}} - \mathcal{C}_{\text{PAH}}}{\mathcal{C}_{\text{NDH}}},$$

where $\mathcal{C}_{\text{NDH}}$ and $\mathcal{C}_{\text{PAH}}$ are the expected inventory holding cost of NDH solution and PAH solution, respectively. Note that for most of the instances, when the expected makespan is set to be $U_2 \times 1.005$ or $U_2 \times 1.02$, NDH is unable to find a feasible solution, and that is why N/A appears in some fields of the table.

Table 2.12: Percentage of decrease in expected inventory holding cost when, given the fixed expected makespan, one heuristic replaces another one.

		$\tilde{UB}$	$1.005 \times U_2$	$1.02 \times U_2$	$1.005 \times U_3$	$1.02 \times U_3$
30 Activities	iPAH $\rightarrow^*$ iNDH	Average	N/A	N/A	69.94%	61.31%
		Median	N/A	N/A	74.08%	64.94%
	iPAH $\rightarrow$ iEACH	Average	21.02%	14.98%	13.14%	11.97%
		Median	19.80%	11.56%	9.68%	8.49%
	iEACH $\rightarrow$ iNDH	Average	N/A	N/A	65.55%	56.32%
		Median	N/A	N/A	69.11%	58.66%
60 Activities	iPAH $\rightarrow$ iNDH	Average	N/A	N/A	84.05%	77.86%
		Median	N/A	N/A	88.62%	82.96%
	iPAH $\rightarrow$ iEACH	Average	34.31%	24.82%	19.04%	17.96%
		Median	36.54%	24.08%	17.64%	16.47%
	iEACH $\rightarrow$ iNDH	Average	N/A	N/A	80.77%	73.61%
		Median	N/A	N/A	85.10%	78.42%

[*] $X \rightarrow Y$ denote X replaces Y.

Table 2.12 demonstrates that applying either EACH or PAH, on average, decreases the expected inventory holding cost of a project by approximately 55-90% compared to the current practice (NDH). When the choice of heuristics is limited to the deterministic ones, as Tables 2.10-2.12 show, although on average PAH approach perform better than EACH, for each project, it is best to use both approaches and select the best solution. However, if the computational power is limited, a practitioner should use PAH. Note that one might prefer EACH over PAH as the maximum solution time of EACH is significantly less than the maximum solution time of PAH. Other than the heuristic approach, a practitioner must trade-off the expected makespan vs. the expected inventory holding cost, and the efficient frontiers depicted in Figure 2.17 can help them to make such decisions.

All Heuristics: iNDH, iEACH, iPAH, iSAAD and iSAAD+

iSAAD solves iSFCT($\mathcal{S}'$) in its first stage. Recall that the objective function of iSFCT($\mathcal{S}'$) minimizes the expected cost of a project over $\mathcal{S}'$, while the objective function of SFCT($\mathcal{S}'$) minimizes the expected makespan of the project over $\mathcal{S}'$. A valid question is if solving iSFCT($\mathcal{S}'$) in the first stage adds value over solving SFCT($\mathcal{S}'$)?

To answer this question, we find the optimal resource flow of both iSFCT($\mathcal{S}'$) and SFCT($\mathcal{S}'$), and solve AZāL($\mathcal{S}$) for those resource flows, given the expected makespan of the project is set to be $1.005 \times U_2$, $1.02 \times U_2$, $1.005 \times U_3$ and $1.02 \times U_3$, where U_2 and U_3 are defined in Equations 2.75 and 2.74. In the iSFCT($\mathcal{S}'$), we set the value of γ to 10^{-4}. Table 2.13 depict the summary of the results.

Table 2.13: Impact of solving iSFCT($\mathcal{S}'$) rather than SFCT($\mathcal{S}'$) in the first stage of iSAAD.

	$\tilde{UB}$	$1.005 \times U_2$		$1.02 \times U_2$		$1.005 \times U_3$		$1.02 \times U_3$	
	Heuristic	iSAAD$_I$ [1]	iSAAD$_S$ [2]	iSAAD$_I$	iSAAD$_S$	iSAAD$_I$	iSAAD$_S$	iSAAD$_I$	iSAAD$_S$
Expected Total Inventory Holding Costs	Average	7.4263	7.9959	4.8626	5.1827	2.9051	3.0605	2.2129	2.3243
	Median	6.8454	7.3959	4.5436	4.9663	2.5729	2.7292	1.9652	2.1269
	Minimum	0.2322	0.2384	0.1468	0.1500	0.0621	0.0635	0.0451	0.0462
	Maximum	25.2049	25.3503	16.3477	16.4117	16.5518	16.6169	11.9965	12.0323
Best Solution (Out of 282) [3]		235	58	241	54	242	55	241	56
Avg Decrease in Inv Costs [4]		6.58%		6.22%		5.71%		5.58%	
P-value [5]		1.4440 e-27		1.0010 e-29		1.0439 e-29		1.9202 e-29	

[1] The iSFCT($\mathcal{S}'$) is solved in the first stage.

[2] The SFCT($\mathcal{S}'$) is solved in the first stage.

[3] For some instances more than one approach finds the best solution.

[4] Average decrease in expected inventory holding costs when in the first stage, the iSFCT($\mathcal{S}'$) is solved rather than the SFCT($\mathcal{S}'$).

[5] Wilcoxon signed-rank test.

The last row in Table 2.13 represents the P-value of Wilcoxon signed-rank test of whether solving iSFCT($\mathcal{S}'$) rather than SFCT($\mathcal{S}'$) in the first stage reduces the expected inventory holding cost. For all given expected makespan ($\tilde{U}$) in this table, the test suggests that we can strongly reject the hypothesis that solving iSFCT($\mathcal{S}'$) instead of SFCT($\mathcal{S}'$) in the first stage leads to no improvement. In other words, solving iSFCT($\mathcal{S}'$) in the first stage results in lower expected inventory holding cost.

To compare the performance of the proposed deterministic heuristics to SAA-based heuristics, similar to Table 2.13, we consider $\tilde{U}$ to be $1.005 \times U_2$, $1.005 \times U_2$, $1.005 \times U_2$ and $1.005 \times U_2$. Table 2.14 represents the expected inventory holding cost and the total solution times of all proposed heuristics when the expected makespan is set to be $\tilde{U}$.

Table 2.14: Expected inventory holding cost and solution times of all proposed heuristics.

		Best	expected inventory holding cost				Total Solution Time (sec)			
$\tilde{UB}$	Approach	Solution*	Average	Median	Min	Max	Average	Median	Min	Max
$1.005 \times U_2$	iEACH	2	12.9601	12.5400	1.7970	33.8279	5.49	5.22	1.91	14.87
	iPAH	14	9.6667	9.1708	0.2498	27.7585	8.95	8.62	2.32	21.67
	iSAAD	17	7.4263	6.8454	0.2322	25.2049	98.49	26.96	6.75	3645.11
	iSAAD+	266	6.7899	6.1724	0.2017	24.2902	5250.78	1423.95	382.79	49137.61
$1.02 \times U_2$	iEACH	2	7.2422	7.0847	0.8456	19.9954	5.92	5.74	1.88	15.00
	iPAH	16	5.9097	5.7028	0.1618	17.7944	8.54	8.16	2.28	18.97
	iSAAD	15	4.8626	4.5436	0.1468	16.3477	98.33	26.68	8.24	3651.59
	iSAAD+	264	4.5127	4.2057	0.1257	15.7704	5253.84	1394.80	368.03	48983.34
$1.005 \times U_3$	iNDH	0	16.3725	15.9639	3.5011	37.2681	21.70	21.11	11.65	39.52
	iEACH	2	4.1488	3.7591	0.1448	20.2580	6.32	5.97	2.56	18.17
	iPAH	12	3.4710	3.1626	0.0831	18.0166	8.94	8.77	2.36	18.86
	iSAAD	18	2.9051	2.5729	0.0621	16.5518	98.57	28.50	6.99	3648.66
	iSAAD+	268	2.7167	2.3787	0.0512	15.9665	5264.35	1392.55	407.25	49079.60
$1.02 \times U_3$	iNDH	0	8.9666	8.6093	1.7318	21.9254	19.89	18.99	12.34	38.31
	iEACH	2	3.0768	2.8163	0.0986	14.4517	6.55	6.20	2.51	18.45
	iPAH	11	2.6138	2.3534	0.0545	12.9981	9.23	8.99	2.47	19.37
	iSAAD	23	2.2129	1.9652	0.0451	11.9965	99.03	28.77	6.63	3645.26
	iSAAD+	269	2.0765	1.8253	0.0373	11.5391	5272.70	1435.51	348.78	49091.41

* Out of 282. For some instances more than one approach finds the best solution.

As in the previous subsection, for any given resource flow, by solving AZāL($\mathcal{S}$) for different values of $\tilde{UB}$, we can create an efficient frontier that represents the trade-off between the expected inventory holding cost and the expected makespan of the project. Figure 2.18 represents the efficient frontier for instances $(15-3)$ and $(31-6)$. The same figures (in HTML format) for all 282 instances are available here.

Figure 2.18 illustrates that for instance $(15-3)$, when the upper bound is set to be $1.005 \times U_3$, the expected inventory holding cost of SAAD+ solution is 86.74% less than NDH solution. Similarly, it shows when the upper bound is set to be $1.02 \times U_2$, SAAD+ solution results in 57.7% less expected inventory holding cost than PAH solution. Table 2.15 represents the average and median percentage of decrease in the expected inventory holding cost when one heuristic replaces another one. Note that for most of the instances, when the expected makespan is set to be $U_2 \times 1.005$ or $U_2 \times 1.02$, NDH is unable to find a feasible solution, and that is why N/A is appeared in some fields of the table.

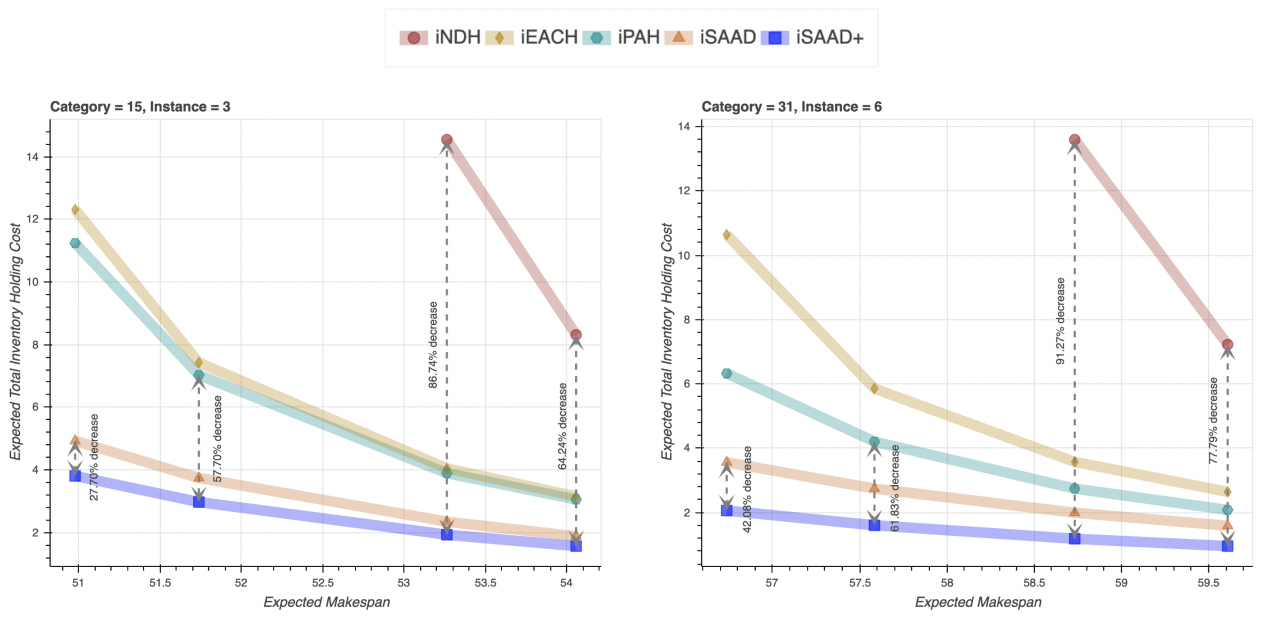


Figure 2.18: Efficient frontier of expected makespan vs. expected inventory holding cost.

Table 2.15: Percentage of decrease in expected inventory holding cost when, given the fixed expected makespan, one heuristic replaces another one.

	$\tilde{UB}$	$1.005 \times U_2$	$1.02 \times U_2$	$1.005 \times U_3$	$1.02 \times U_3$
iSAAD+ $\rightarrow$[1] iSAAD	Average	8.29%	7.4%	6.78%	6.61%
	Median	3.54%	3.56%	3.06%	2.99%
iSAAD+ $\rightarrow$ B-det [2]	Average	26.93%	22.19%	19.55%	18.71%
	Median	26.49%	21.17%	17.43%	15.95%
iSAAD+ $\rightarrow$ iNDH	Average	N/A	N/A	84.22%	78.03%
	Median	N/A	N/A	84.91%	78.54%

[1] $X \rightarrow Y$ denote X replaces Y.

[2] B-det denote the best solution among deterministic heuristics

Tables 2.14 and 2.15 and Figure 2.18 show that, when sufficient computational power is available, it is best for a practitioner to use iSAAD+, as it results in significantly lower expected inventory holding cost. The limitation of iSAAD+ is that it cannot be applied to projects with many activities. Among project with 30 activities, iSAAD+ is most suitable for those with large RS_r and small RF_r, as it is easier to optimally solve iSFCT($\mathcal{S}'$) for those projects.

The complexity of iSAAD+ increases linearly as the number of replications increases.

Thus, we can choose large numbers of replications without significantly increasing the required computational effort. Figure 2.19 shows the impact of increasing the number of replications (from 1 to 30) on the average expected inventory holding cost of iSAAD+ solution. This figure suggests that the impact of increasing the number of replications is higher when the given expected makespan is smaller.

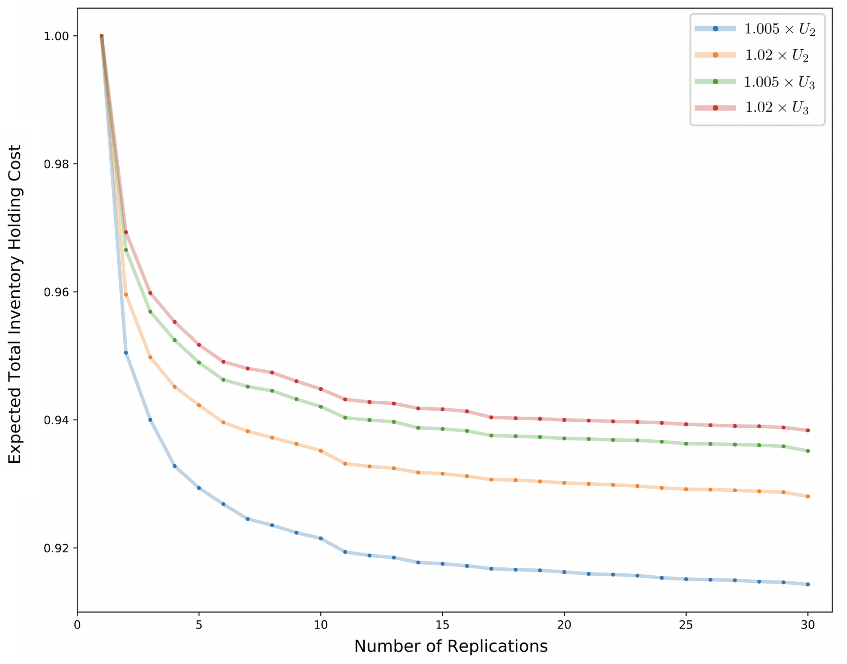


Figure 2.19: Average expected inventory holding cost of iSAAD+ solution for different given expected makespan and different number of replications.

Chapter 3

Scheduling of Oil Field Drilling Operations (A Case Study)

3.1 Problem Description

To create a well, an expensive drilling rig (which we subsequently call *the resource*) is required. As these resources are costly, only a limited number of them are available at each project, and each of them needs to be utilized multiple times to create multiple wells. So allocating these renewable resources to the wells is a key decision.

In a project, wells are positioned in different locations. After a resource finishes drilling a well, it is transferred to the location of the next well that it will process. Although all resources can process all wells at the same speed, their transportation times can be different. The transportation of resources is slow and expensive, and this time and cost depend on the start and end locations. This transportation also requires special licenses, and possibly building roads, which is very time-consuming. Hence, it is essential to determine at the start of the project the sequence of wells that each resource will process, as this sequence determines the resources' transportation route.

In addition to one resource (a drilling rig), unique materials and equipment (e.g., special pipe spools) are required to create a well. While the engineering team decides the specifics of materials and equipment required for each well, we need to determine the delivery time of this materials/equipment to each well. It is assumed that all materials/equipment required for a well are delivered together, and that processing on a well cannot start before the delivery. Also, as the materials/equipment is mostly make-to-order, it has long lead-times for which delivery dates need to be determined at the beginning of the project. Delivering too soon increases inventory holding cost of the project dramatically, while delivering too late delays the start of the process on wells.

Although there are engineering estimates regarding the duration of drilling for each well, these estimates are not necessarily accurate due to reasons such as human errors, weather, and technical complications. In this research, we assumed the duration of drilling a well has a known distribution.

Most research in the literature that considers problems similar to the scheduling of oil field drilling operations considers the objective function of minimizing the expected overall project delivery time (expected makespan). However, the goal here is maximizing the expected revenue generated by oil extraction. We frame this as a penalty, potentially different for each well, that captures missed revenue generation opportunities when production at a well is delayed. To mathematically describe the goal, for each well j, we define a weight w_j that represents a per unit timed missed revenue generation opportunity. Then, the objective function is to minimize the expected average weighted production start time of the wells. Let z_j denotes the production start time of well j, and N denotes the set of all wells. Then, the objective function is $\min \sum_{j \in N} w_j \times z_j$.

After a well is drilled, it is necessary to further process the well, in what is known as the *completion phase*. The duration of the completion phase is adequately predictable, and the required machinery is attainable. Hence, we assume production at a well starts a constant time (which is proportional to the duration of drilling) after it is drilled. Then:

$$\sum_{j \in N} w_j \times z_j = \sum_{j \in N} w_j \times (C_j + \alpha \times P_j) = \sum_{j \in N} w_j \times C_j + constant,$$

where P_j is the stochastic duration of drilling well j (which we subsequently call the *duration* of well j), $\alpha \times P_j$ is the duration of the completion phase of well j, and C_j the time of completion of the drilling of well j (which we subsequently call the *finish time*). From a modeling perspective, adding a constant to the objective function does not impact the solution; consequently, the objective function is equivalent to $\min \sum_{j \in N} w_j \times C_j$, which is the expected average weighted finish time.

In this problem, a solution is a static policy, which is a set of rules that are applied dynamically to determine the schedule of the project. The policy needs to proactively determine the sequence of wells that each resource visits, and the delivery time of materials/equipment for all wells at the beginning of the project, and dynamically determine the starting time of the drilling process at each well to minimize the expected average weighted finish time of wells. To better motivate why the expected average weighted finish time rather than the expected makespan best represents the objective function for this problem, consider Example 4.

Example 4. *Consider the project that is depicted in Figure 3.1. Well 1 extracts oil with an associated production rate of 2.5, and Wells 2 and 3 have production rates of 1.0 and 2.0, respectively. The duration of drilling Wells 1-3 are deterministic and equal to* 2.00, 0.75, *and* 2.50, *respectively, and the duration of the completion phase is zero for them. Also, all wells are in one location (so there is no transportation time).*

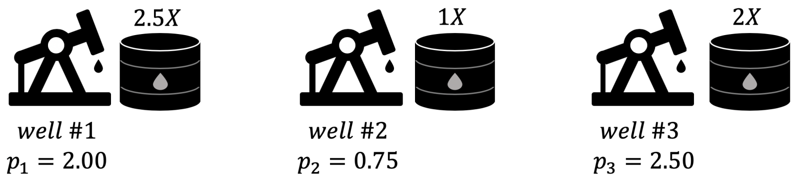


Figure 3.1: Details of the project that is introduced in Example 4.

Two schedules are suggested in Figure 3.2. In the first schedule, the makespan of the project is 2.75, while in the second schedule, the makespan is 3.25. However, when we consider the total amount of oil extracted by time 3.25, schedule one extracts a total of 5.125 units, while schedule two extracts a total of 5.625 units. In this example, although the makespan of schedule two is not as good as that of schedule one, it generates more revenue.

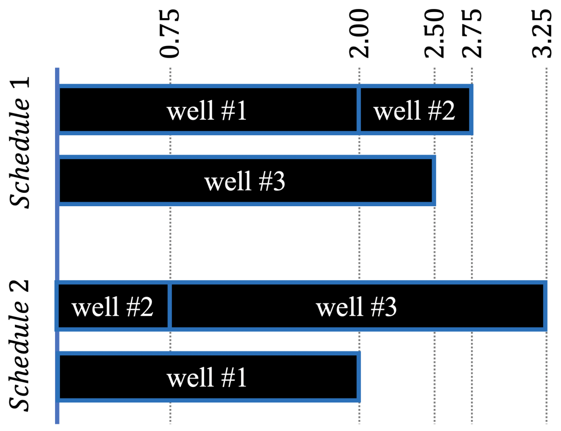


Figure 3.2: Two alternative schedules for the project.

We present the mathematical formulation of the integrated problem in Section 3.2. We describe the current practice that we refer to as the *decomposed approach* because it effectively relaxes the mathematical formulation and decomposes it into smaller problems, and our proposed heuristic approach, which we refer to as the *partially decomposed approach* in Section 3.3. Finally, in a case study, using the data from a real-world project, we compare the performance of our approach to current practice in Section 3.4.

3.2 Mathematical Model Formulation

As described in Section 3.1, in reality, the duration of the drilling processes is uncertain. We introduce the following notation to express the objective function of minimizing the expected average weighted finish times:

$$E_{\vec{P}}[F(\vec{P}, \nu)], \tag{3.1}$$

where

ν: Set of all decision variables;
$\vec{P}$: Vector of durations (each element has an associated probability distribution);
$F(.)$: A function that provides the average weighted finish times.

The expected value in Equation 3.1 is a high-dimensional integral. However, we can solve this problem numerically by sampling a set of independently and identically distributed (i.i.d.) realizations of the stochastic durations (S), which we call *scenarios*. Given this approach, the expectation can be written as:

$$E_{\vec{P}}[F(\vec{P}, \nu)] = \lim_{|S| \to \infty} \sum_{s \in S} F(\vec{p}_s, \nu)/|S|, \tag{3.2}$$

where

S: Set of sampled scenarios;
$\vec{p}_s$: The vector of realized durations in scenario s.

With this approach, as the number of stochastic parameters (durations) is finite, we can convert our stochastic problem into an equivalent massive deterministic scenario-based problem. Intuitively, a scenario can be interpreted as a plausible realization of the stochastic durations of all wells.

We propose the following mixed-integer linear programming (MILP) formulation to model the equivalent deterministic scenario-based version of the stochastic scheduling of oil field drilling operations. The problem contains three types of decisions, namely: assignment of resources to the wells, determination of the sequences of wells that each resource visits, and determination of the delivery times of materials/equipment. The following MILP formulation considers all of these decisions concurrently, and hence we call it the *Integrated Model*.

$$\min \sum_{j \in N} \sum_{s \in S} w_j C_j^s / (|S| \times \sum_{j \in N} w_j) \tag{3.3}$$

$$\sum_{\substack{j \in N_0 \\ j \neq k}} \sum_{i \in M} x_{ijk} = 1 \qquad \forall\, k \in N \tag{3.4}$$

$$\sum_{\substack{k \in N \\ j \neq k}} \sum_{i \in M} x_{ijk} = 1 \qquad \forall\, j \in N \tag{3.5}$$

$$\sum_{j \in N} x_{i0j} \leq 1 \qquad \forall\, i \in M \tag{3.6}$$

$$\sum_{\substack{k \in N_0 \\ j \neq k}} x_{ijk} = \sum_{\substack{h \in N_0 \\ h \neq j}} x_{ihj} \qquad \forall\, j \in N, i \in M \tag{3.7}$$

$$C_k^s - C_j^s + V(1 - x_{ijk}) \geq T_{ijk} + p_k^s \qquad \forall\, j \in N_0, k \in N, i \in M, s \in S \tag{3.8}$$

$$C_0^s = 0 \qquad \forall\, s \in S \tag{3.9}$$

$$I_j^s = C_j^s - O_j - p_j^s \qquad \forall\, j \in N, s \in S \tag{3.10}$$

$$\sum_{j \in N} \sum_{s \in S} \theta_j I_j^s / |S| \leq \bar{I} \tag{3.11}$$

$$x_{ijk} \in \{0, 1\} \qquad \forall\, j \in N_0, k \in N_0, j \neq k, i \in M$$

$$I_j^s \geq 0 \qquad \forall\, j \in N, s \in S$$

where:

Sets

M: Set of resources;
N: Set of wells;
N_0: N + a dummy node indexed by 0;
S: Set of scenarios;

Parameters and Constants

p_j^s for $j \in N$ and $s \in S$: Realized Duration of well j in scenario s;
T_{ijk} for $i \in M$ and $j, k \in N_0$: Transportation time of resource i from well j to well k;
w_j for $j \in N$: Weight of well j (per unit timed missed revenue generation opportunity);
θ_j for $j \in N$: The per unit of time inventory cost for materials/equipment of well j;
$\bar{I}$: Upper bound on the expected inventory holding cost;
V: A very large positive number (a.k.a. big-M);

Variables

x_{ijk} for $i \in M$ and $j, k \in N_0$: 1 if resource i process well k directly after well j, zero o/w;
C_j^s for $j \in N_0$ and $s \in S$: Finish time of well j in scenario s;
O_j for $j \in N$: Arrival time of materials/equipment of well j;
I_j^s for $j \in N$ and $s \in S$: Total time materials/equipment of well j stays in inventory in scenario s.

The objective function, Equation 3.3, is the expected average weighted finish time. Constraints 3.4-3.7 ensure that all resources visit a feasible sequence of wells, Constraint 3.8 ensures that the value of finish times are determined correctly, and Constraint 3.9 sets the start of the project to zero.

Constraint 3.10 defines a variable I_j^s for $j \in N$ and $s \in S$ to represent the total time that materials/equipment of each well j stays in inventory in scenario s. The fact that this variable is non-negative ensures the processing of each well cannot start before the delivery of its materials/equipment. Lastly, Constraint 3.11 ensures that the expected inventory holding cost is less than a predefined limit. Intuitively, increasing (decreasing) the limit decreases (increases) the expected average weighted finish times.

While our goal is to solve the integrated model, realistically sized instances of the scheduling of oil field drilling operations are intractable. For instance, Gurobi (version 8.1.1) is unable to find even a feasible solution for our case study in 3600 seconds.

3.3 Solution Approach

3.3.1 Decomposed Approach (Current Practice)

Based on our discussions with experts, current practice for addressing the scheduling of oil field drilling operations is to make the decisions listed in Section 3.2 sequentially. We depict this approach in Figure 3.3. Formally, this is a so-called "decomposition approach." However, in general experts do not explicitly consider and formally make this decomposition – this is just considered the "natural" way to solve this problem.

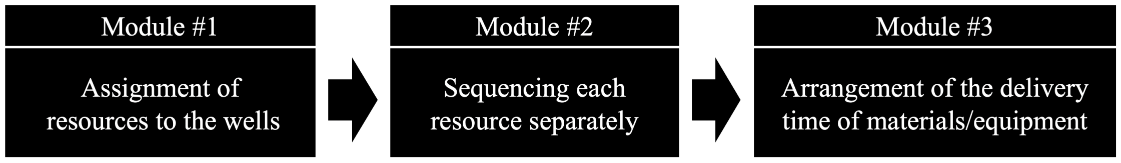

Module #1
Assignment of resources to the wells
Module #2
Sequencing each resource separately
Module #3
Arrangement of the delivery time of materials/equipment

Figure 3.3: Decomposed Approach – decomposing the integrated model into three modules.

In the decomposed approach, Module 1 assigns the resources to the wells. The assignment is determined by minimizing the expected average weighted finish time of wells, while the inventory holding costs and transportation times are ignored. The allocation of resources to the wells is an input to Module 2, which minimizes the expected average weighted finish time of wells for each resource while accounting for the transportation times and ignoring the inventory holding costs. Then, the assignment of activities along with the sequences of wells that each resource visits are given to Module 3, in which the delivery times of materials/equipment are determined. Module 3 minimizes the expected average weighted finish times in a way that the expected inventory holding cost be less than or equal to a predefined limit.

The mathematical formulations of these modules are presented in Subsections 3.3.1, 3.3.1, and 3.3.1, respectively. As we show in Lemma 12, sequentially solving Modules 1 and 2 is not necessarily as useful as solving them concurrently. Similarly, in Lemma 13, we show that solving Module 3 after Modules 1 and 2 might not result in the optimal solution of the integrated model. The decomposed approach is also relatively inflexible in terms of adding constraints on the transportation times because the assignment of resources to the wells is arranged before considering the transportation times.

Lemma 12. *Sequentially solving Modules 1 and 2 does not necessarily find the optimal solution.*

Proof. A simple counterexample is provided here. Consider three wells in three different locations as it is illustrated in Figure 3.4.

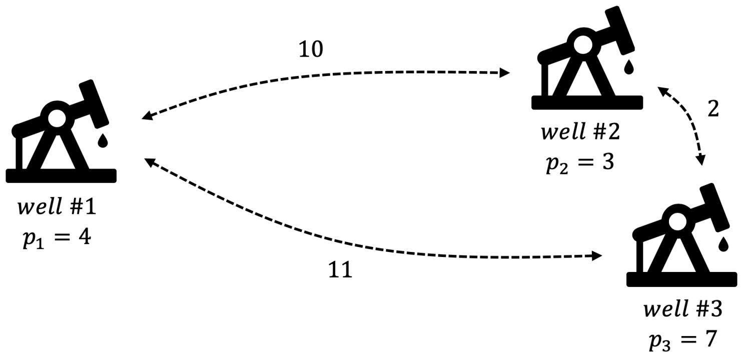


Figure 3.4: Illustration of the counterexample for Lemma 12

With two resources, the optimal solution of Module 1 is assigning one resource to wells one and two and the other resource to well three. Solving Module 2 subsequently shows the first resource should first dig well two and then well one, and the other resource should only dig well three. The value of the objective function for this solution is 27.

On the other hand, concurrently solving Modules 1 and 2 shows that it is optimal to create well two and then well three with the first resource and well one with the other resource. The objective value of this solution is 19. □

Lemma 13. *The optimal solution resulting from solving Module 3 with the given assignment of resources and sequence of wells that each resource visits is not guaranteed to be the same as the optimal solution of the integrated model.*

Proof. A simple counterexample is presented in Figure 3.5. The optimal sequence of wells for the resources is to process well one and then two with one of the resources, and well three and then four with the other one. The expected objective value of this solution is 22. In Module 3, when no inventory holding cost can be tolerated, the materials/equipment delivery time for wells one and two are zero and seven, respectively. Hence, given the sequence, the optimal objective value of Module 3 is 24. On the contrary, the optimal solution of the integrated model is to process well one and then well three with one resource, well two and then four with the other one, and delivering the materials/equipment of all wells at time zero. The objective value of this solution is 23.

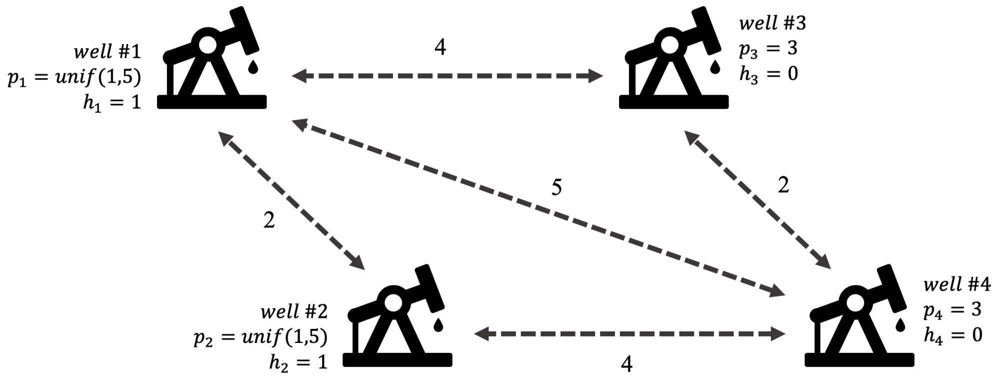


Figure 3.5: Illustration of a counterexample for Lemma 13

□

Modules 1 and 2 in the decomposed approach are both NP-hard, and the size of these problems are very large as we account for uncertainties in the durations by defining a massive set of scenarios. Hence, they are both intractable.

We refer to these problems with uncertainties in durations are considered as *the stochastic versions of the drilling problem (SDP)*. In contrast to the SDP, we define *the deterministic versions of the drilling problem (DDP)* by replacing the stochastic durations with their expected values.

Lemma 14 proves that the optimal solution to the SDP and the DDP versions of Modules 1 and 2 are the same. As a result, there is no need to factor in the uncertainties of durations to find the optimal solution of these parts, which significantly reduces their size and complexity.

Lemma 14. *For Modules 1, Module 2, and the Combined Modules 1-2 (which is presented in Subsection 3.3.2), the optimal solution to the SDP and the DDP versions are the same .*

Proof. Consider any solution to the SDP version of one of Module 1, Module 2, or the Combined Modules 1-2. The objective function of the SDP version (Z^s) and the DDP version (Z^d) are:

$$Z^s = \sum_{j\in N}\sum_{s\in S} w_j C_j^s / (|S| \times \sum_{j\in N} w_j)$$

$$Z^d = \sum_{j\in N} w_j C_j / \sum_{j\in N} w_j,$$

where C_j^s is the finish time of well $j \in N$ in scenario $s \in S$ in the SDP version of the problem, and C_j is the finish time of well $j \in N$ in the DDP version of the problem.

For each well j, let $seq(j)$ denote the sequence of wells that are drilled before well j (including itself) by the same resource, $seq^{-1}(j)$ denote the sequence of wells that are drilled after well j (including itself) by the same resource, and $res(j)$ denote the resource that drilled well j. As there is no inventory consideration in these problems, C_j for $j \in N$ and

C_j^s for $j \in N$ and $s \in S$ can be calculated solely from $seq(j)$ and $res(j)$ for $j \in N$. As a result, for each well $j \in N$, C_j and C_j^s for $s \in S$ can be expressed as:

$$C_j = \sum_{k \in seq(k)} (T_{res(j)(k-1)k} + E[P_k])$$

$$C_j^s = \sum_{k \in seq(k)} (T_{res(j)(k-1)k} + p_k^s).$$

Therefore,

$$Z^s = \sum_{j \in N} \sum_{s \in S} w_j \Big(\sum_{k \in seq(k)} T_{res(i)(k-1)k} + p_k^s \Big) / (|S| \times \sum_{j \in N} w_j) = \sum_{j \in N} \sum_{k \in seq(k)} w_j T_{res(i)(k-1)k} / \sum_{j \in N} w_j + \sum_{j \in N} \Big(w_j \sum_{k \in seq(k)} \sum_{s \in S} p_k^s / |S| \Big) / \sum_{j \in N} w_j$$

As $\sum_{s \in S} p_j^s / |S|$ is the expected value of P_j for $j \in N$, we conclude:

$$Z^s = \sum_{j \in N} \sum_{k \in seq(k)} w_j T_{res(i)(k-1)k} / \sum_{j \in N} w_j + \sum_{j \in N} \Big(w_j \sum_{k \in seq(k)} E[P_k] \Big) / \sum_{j \in N} w_j = \sum_{j \in N} w_j \Big(\sum_{k \in seq(k)} T_{res(i)(k-1)k} + E[P_k] \Big) / \sum_{j \in N} w_j = Z^d$$

By straightforward algebraic manipulation, it is easy to see that the proof is also correct in the other direction, so each solution results in the same objective value in both SDP and DDP versions. Therefore, the optimal solutions of SDP and DDP are the same.

□

Mathematical Optimization Formulation of the Module 1

As we proved in Lemma 14, there is no need to consider uncertainties of durations in Module 1. Hence, instead of the SDP version of the model, we create its equivalent DDP version by replacing the distributions of the durations with their expected values. This model determines the assignment of resources to the wells to minimize the average weighted finish times without accounting for the transportation times and the inventory constraints.

In this mixed-integer programming (MIP) formulation, we use the optimality property of weighted shortest processing time (WSPT) algorithm for the problem $1||\sum_j w_j C_j$ ([70]). In this formulation, without loss of generality, it is assumed that the numerical indices of wells are arranged in decreasing order of $w_j/E[P_j]$, so that $w_j/E[P_j] \geq w_{j+1}/E[P_{j+1}]$ for all $j \in N$.

$$\min \sum_{j \in N} w_j C_j / \sum_{j \in N} w_j \tag{3.12}$$

$$\sum_{i \in M} x_{ij} = 1 \qquad \forall\, j \in N \tag{3.13}$$

$$C_j = \sum_{i \in M} x_{ij}(E[P_j] + \sum_{k<j} E[P_k] x_{ik}) \qquad \forall\, j \in N \tag{3.14}$$

$$x_{ij} \in \{0,1\} \qquad \forall\, i \in M, j \in N$$

where:

Sets

M: Set of resources;

N: Set of wells;

Parameters and Constants

P_j for $j \in N$: The stochastic duration of well j;

$E[P_j]$ for $j \in N$: The expected duration of well j;

w_j for $j \in N$: Weight of well j (per unit timed missed revenue generation opportunity);

Variables

C_j for $j \in N_0$: Finish time of well j;

x_{ij} for $i \in M$ and $j \in N_0$: 1 if resource i process well j, zero otherwise.

The objective function in this model (Equation 3.12) minimizes the average weighted finish time. Constraint 3.13 ensures that precisely one resource is assigned to each well, and Constraint 3.14 defines the finish times of wells. Recall that wells are indexed based on decreasing $w_j/E[P_j]$, and based on the WSPT rule, among all wells that are assigned to the same resource, wells are processed in increasing index order.

This problem is equivalent to a machine scheduling problem, where each resource represents a machine, and each well denotes a job. With this representation, the problem is equivalent to the parallel machine scheduling problem, denoted $P_m||\sum w_j C_j$ in the machine scheduling literature.

Although Module 1 is in practice significantly more straightforward to solve than the original integrated problem, as [51] showed, it is still NP-hard in the strong sense for an arbitrary number of resources. In our case study, the default settings of Gurobi (version 8.1.1), is unable to solve it optimally in 3600 seconds. The best optimality gap it can achieve in 3600 seconds is 38.17%, where the optimality gap is defined as:

$$\textit{optimality gap} = \frac{\textit{the smallest upper bound} - \textit{the largest lower bound}}{\textit{the largest lower bound}}$$

A well-known approximation algorithm for this problem, the WSPT rule ([40]), achieves an objective value that is at most $(1+\sqrt{2})/2$ times the optimal objective value. In our case study, the solution generated by this heuristic is 3.53% more than the optimal objective value.

[14] suggested a column generation algorithm for solving $P_m||\sum w_j C_j$. In our case study, the column generation approach finds solutions with the optimality gap of less than 0.01% in a few seconds. In what follows, we consider a solution with an optimality gap of less than 0.01% as an optimal solution.

Mathematical Optimization Formulation of the Module 2

In Module 2, the assignment of resources to the wells is given. Therefore, the schedule of each resource can be determined independently from the schedule of other resources. Also, as we prove in Lemma 14, there is no need to consider uncertainties of the durations. The following MILP formulation determines the sequence of wells that resource $i \in M$ should drill to minimize the average weighted finish times. Note that this module does not account for inventory-related constraints.

$$\min \sum_{j \in N^i} w_j C_j / \sum_{j \in N^i} w_j \tag{3.15}$$

$$\sum_{\substack{j \in N_0^i \\ j \neq k}} x_{jk} = 1 \qquad \forall\, k \in N^i \tag{3.16}$$

$$\sum_{\substack{k \in N^i \\ j \neq k}} x_{jk} = 1 \qquad \forall\, j \in N_0^i \tag{3.17}$$

$$C_k - C_j + V(1 - x_{jk}) \geq T_{jk} + E[P_k] \qquad \forall\, j \in N_0^i, k \in N^i \tag{3.18}$$

$$C_0 = 0 \tag{3.19}$$

$$x_{jk} \in \{0,1\} \qquad \forall\, j \in N_0^i, k \in N_0^i, j \neq k$$

where:

Sets

M: Set of resources;
N^i for $i \in M$: Set of wells that are assigned to resource i in sub-model one;
N_0^i for $i \in M$: N^i + a dummy node indexed by 0;

Parameters and Constants

P_j for $j \in N$: The stochastic duration of well j;
$E[P_j]$ for $j \in N$: The expected duration of well j;
T_{ijk} for $i \in M$ and $j, k \in N_0^i$: Transportation time of resource i from well j to well k;

w_j for $j \in N^i$: Weight of well j (per unit timed missed revenue generation opportunity);
V: A very large positive number (a.k.a. big-M);

Variables

C_j for $j \in N_0^i$: Finish time of well j;
x_{jk} for $j \in N_0^i$ and $k \in N^i$: 1 if well k is processed directly after well j, zero otherwise.

The objective function (Equation 3.15) minimizes the average weighted finish time of wells that are assigned to the resource i. Constraints 3.16 and 3.17 ensure that wells are feasibly sequenced, and Constraints 3.18 and 3.19 define the finish time of each well with respect to the sequence, process durations and transportation times.

This problem can be represented as a machine scheduling problem, denoted $1|S_{j,k}|\sum w_j C_j$ in machine scheduling literature. In our case study, the problem size for each resource is relatively small: $\sim$50 binary variables, $\sim$7 continuous variables, and $\sim$50 constraints. As a result, the optimal solution can be found in a few seconds using the default settings of Gurobi (Version 8.1.1).

To analyze the impact of limiting the total transportation time in Section 3.4, we need to solve Module 2 for all resources concurrently. In that case, the model has $\sim$300 binary variables, $\sim$40 binary variables, and $\sim$300 constraints, and Gurobi (Version 8.1.1) is still capable of solving it optimally.

Mathematical Optimization Formulation of the Module 3

In Module 3, the assignment of resources to the wells and the sequence of wells that each resource drills are given. This module determines the delivery date of materials/equipment to minimize the expected average finish times in a way that the total expected inventory holding cost remains less than the maximum allowed.

In contrast to Modules 1 and 2, it is essential to consider the uncertainties of durations in Module 3, as uncertainty is what leads to inventory holding costs. To do this, as discussed before, we formulate the scenario-based equivalent of the stochastic problem, resulting in the following linear programming (LP) formulation for Module 3.

$$\min \sum_{j\in N} \sum_{s\in S} w_j C_j^s / (|S| \times \sum_{j\in N} w_j) \tag{3.20}$$

$$C_k^s - C_j^s \geq T_{ijk} + p_k^s \qquad \forall\, j \in N_0, k \in N, i \in M, s \in S \;\; if \;\; \hat{x}_{ijk} = 1 \tag{3.21}$$

$$C_0^s = 0 \qquad \forall\, s \in S \tag{3.22}$$

$$I_j^s = C_j^s - O_j - p_j^s \qquad \forall\, j \in N, s \in S \tag{3.23}$$

$$\sum_{j \in N} \sum_{s \in S} \theta_j I_j^s / |S| \leq \bar{I} \tag{3.24}$$

$$I_j^s \geq 0 \qquad \forall\, j \in N, s \in S$$

where:

Sets

M: Set of resources;
N: Set of wells;
N_0: N + a dummy node indexed by 0;
S: Set of scenarios;

Parameters and Constants

$\hat{x}_{ijk}$ for $i \in M$, $j \in N_0$ and $k \in N$: The fixed assignment and processing sequence of resources;
p_j^s for $j \in N$ and $s \in S$: The realized duration of well j in scenario s;
T_{ijk} for $i \in M$ and $j, k \in N_0$: Transportation time of resource i from well j to well k;
w_j for $j \in N$: Weight of well j (per unit timed missed revenue generation opportunity);
θ_j for $j \in N$: The per unit of time cost of holding materials/equipment of well j in inventory;
$\bar{I}$: Upper bound on the expected inventory holding cost;

Variables

C_j^s for $j \in N_0$ and $s \in S$: Finish time of well j in scenario s;
O_j for $j \in N_0$: Delivery time of materials/equipment of well j;
I_j^s for $j \in N_0$ and $s \in S$: Total time materials/equipment of well j stays in inventory in scenario s.

The objective function, Equation 3.20, minimizes the expected average weighted finish times. Constraints 3.21 and 3.22 define the finish time of each well, and Constraint 3.23 defines a variable that captures how long the material/equipment for each well in each scenario is stored in the inventory. Note that as variable I for $j \in N_0$ and $s \in S$ is non-negative, the processing of each well cannot start before the delivery of its materials/equipment. Constraint 3.24 ensures that the expected inventory holding cost is less than the given limit. Intuitively, increasing (decreasing) the limit decreases (increases) the expected average weighted finish times.

This mathematical formulation is an LP that is known to be solvable in polynomial time. The model size increases linearly with the increasing the number of scenarios, which allows us to consider thousands of scenarios and still find the optimal solution in a reasonable time using the default settings of Gurobi (Version 8.1.1).

3.3.2 Partially Decomposed Approach

There is a trade-off in heuristically decomposing a problem. Each module of a decomposed problem is more convenient to solve, but the final solution is more likely to be far from the globally optimal solution. On the other hand, without decomposition, the problem might be intractable, or the best achievable optimality gap might be unacceptable. Thus, we propose an intermediate approach – the partially decomposed approach, which is described in Figure 3.6. In this approach, the assignment of resources to the wells and the sequence of wells that each resource drills are determined concurrently via Combined Module 1-2, and Module 3 is solved sequentially after the Combined Module.

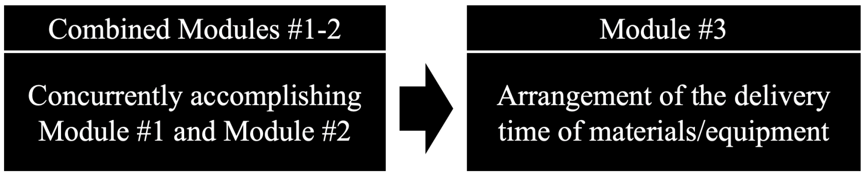

Combined Modules #1-2
Concurrently accomplishing Module #1 and Module #2
Module #3
Arrangement of the delivery time of materials/equipment

Figure 3.6: Partially Decomposed Approach – decomposing the integrated model into two modules.

The mathematical formulation of the Combined Modules 1-2 is presented in Subsection 3.3.2, and the mathematical formulation of Module 3 is shown in Subsection 3.3.1. As it is shown in the next section, the partially decomposed approach finds reasonable solutions in a reasonable amount of time. Additionally, its flexibility enables us to analyze the cost of imposing multiple types of constraints on the transportation of resources.

Mathematical Optimization Formulation of the Combined Modules 1-2

As discussed in Lemma 14, there is no need to consider the uncertainties of durations in the Combined Module 1-2. This model can be formulated as the following MILP:

$$\min \sum_{j \in N} w_j C_j / \sum_{j \in N} w_j \tag{3.25}$$

$$\sum_{i \in M} \sum_{\substack{j \in N_0 \\ j \neq k}} x_{ijk} = 1 \qquad \forall\, k \in N \tag{3.26}$$

$$\sum_{\substack{k \in N \\ j \neq k}} \sum_{i \in M} x_{ijk} = 1 \qquad \forall\, j \in N \tag{3.27}$$

$$\sum_{j \in N} x_{i0j} \leq 1 \qquad \forall\, i \in M \tag{3.28}$$

$$\sum_{\substack{k\in N\\ j\neq k}} x_{ijk} = \sum_{\substack{h\in N_0\\ h\neq j}} x_{ihj} \qquad \forall\, j\in N, i\in M \tag{3.29}$$

$$C_k - C_j + V(1-x_{ijk}) \geq T_{ijk} + E[P_k] \qquad \forall\, i\in M, j\in N_0, k\in N \tag{3.30}$$

$$C_0 = 0 \tag{3.31}$$

$$x_{ijk}\in\{0,1\} \qquad \forall\, i\in M, j\in N_0, k\in N$$

where:

Sets

M: Set of resources;
N: Set of wells;
N_0: N + a dummy node indexed by 0;

Parameters and Constants

P_j for $j\in N$: The stochastic duration of well j;
$E[P_j]$ for $j\in N$: The expected duration of well j;
T_{ijk} for $i\in M$ and $j,k\in N_0$: Transportation time of resource i from well j to well k;
w_j for $j\in N$: Weight of well j (per unit timed missed revenue generation opportunity);
V: A very large positive number (a.k.a. big-M);

Variables

C_j for $j\in N_0$: Finish time of well j;
x_{ijk} for $i\in M$, $j\in N_0$ and $k\in N$: 1 if resource i process well k directly after well j, zero otherwise.

The objective function (Equation 3.25) minimizes the average weighted finish times. Constraints 3.26-3.29 ensure that the assignment of resources and sequence of wells are feasible, and Constraints 3.30 and 3.31 define the finish time of each well.

For our case study, within 3600 seconds, the best solution Gurobi (Version 8.1.1) finds has an optimality gap of 95.77%. In the literature, few methods such as a branch and price algorithm (proposed by [54]), and branch and check algorithms (introduced by [74] and [7]) are available that can solve the proposed model for small instances. However, we are not aware of any approach that can optimize the proposed model for instances that are the size of our case study.

To solve this model, we use LocalSolver, which is a solver that *"combines local and direct search techniques, constraint propagation and inference techniques, linear and mixed-integer programming techniques, as well as nonlinear programming techniques, to solve the problem* *at best"*[1]. In our case study, the best solution that Gurobi finds within 3600 seconds is 2.68% higher than the solution that LocalSolver (Version 9.0) finds in a few seconds.

3.4 Case Study

In the decomposed approach, the assignment of resources to wells and the sequence of wells that each resource drills are determined sequentially, while in the partially decomposed approach, these decisions are determined concurrently. If we find the optimal decision while we determine those decisions concurrently, as it is discussed in Lemma 12, the final solution of the partially decomposed approach should, in general, be better than that of the decomposed approach. Nevertheless, we are not aware of any approach that can optimally solve the Combined Module 1-2, and solving this problem heuristically degrades the overall objective value. In Subsection 3.4.2, we analyze this trade-off and show the partially decomposed approach outperforms the decomposed approach in numerous ways, even though we cannot solve the Combined Module 1-2 optimally.

In practice, managing transportation of resources is challenging and expensive, requiring special licenses and possibly building or improving roads. As a result, there are always strong biases towards preventing any resource from having a long transportation time, and limiting the total transportation times. In Subsection 3.4.3, we discuss the impact of having such biases on project metrics. Additionally, in Subsection 3.4.4, we perform further analysis to estimate the value of providing additional resources.

3.4.1 Test Data Set

In this research, we used the data of a real-world project. Depending on the particular analysis we are performing, we process the data in appropriate ways. For instance, to generate scenarios, we estimate the distribution of the drilling durations based on the historical data and use the distributions to generate scenarios. Also, to determine transportation times between the real locations, we use the Google Maps API. For the purpose of confidentiality, we are not able to reveal additional details about the data.

3.4.2 Decomposed Approach vs. Partially Decomposed Approach

For the first comparison, we ignore the inventory-related constraints by assuming that any amount of expected inventory holding cost is tolerable, and force an upper bound on the expected average weighted finish times and minimize the total transportation time. Figure 3.7 depicts the performance of both the decomposition approach and the partially decomposed approach in this setting.

[1] https://www.LocalSolver.com

Figure 3.7 shows that by using the partially decomposed approach rather than the decomposed approach, the total transportation time decreases by 55% to 70% without affecting the expected average weighted finish times. We expect that the total transportation time of both approaches will decrease as the forced upper bound on the expected average weighted finish times increases. However, we can see for the decomposed approach (the brown line), the line flattens promptly, which is the result of the inflexibility of the current approach.

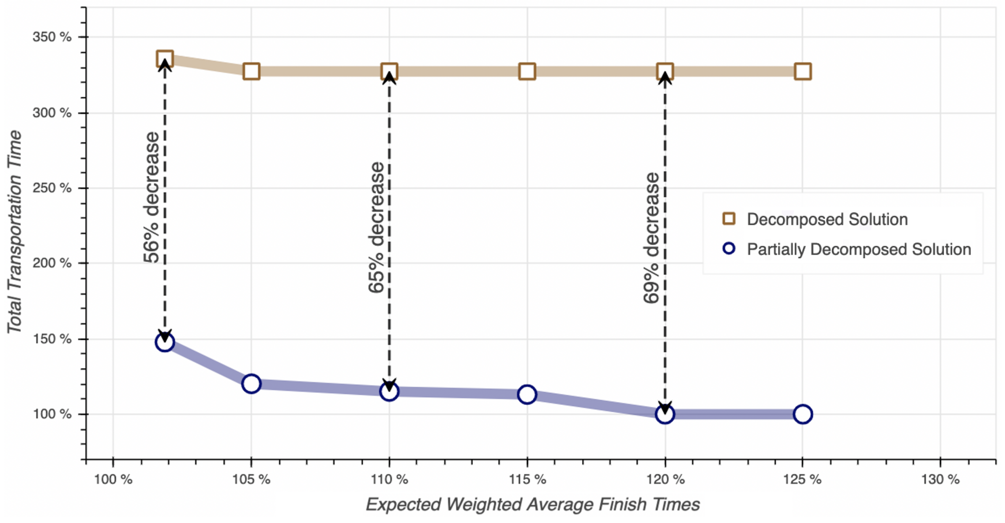


Figure 3.7: Minimum total transportation time with respect to the forced upper bound on the expected average weighted finish times

For the second comparison, we force an upper bound on the expected average weighted finish times and minimize the expected inventory holding cost. Figure 3.8 illustrates the minimum inventory holding cost that each of the approaches achieves while enforcing different upper bounds on the expected average weighted finish times.

As Figures 3.7 and 3.8 indicate, the partially decomposed approach outperforms the decomposed approach in terms of total transportation time and expected inventory holding cost. Furthermore, the partially decomposed approach outperforms the decomposed approach in other transportation time-related metrics. For instance, when it is desired to minimize the maximum transportation time of the resources, the partially decomposed approach finds a solution with a maximum transportation time that is 62% less than the best solution that the decomposed approach achieves.

The partially decomposed approach is capable of finding feasible solutions in the presence of more rigorous transportation-related constraints. To investigate the capacity of approaches on satisfying rigorous constraints, we add a constraint to force an upper bound on the total transportation time as well as a constraint to force an upper bound on the maximum transportation time of the resources. Figure 3.9 represents the range of values for which each of the approaches finds a feasible solution.

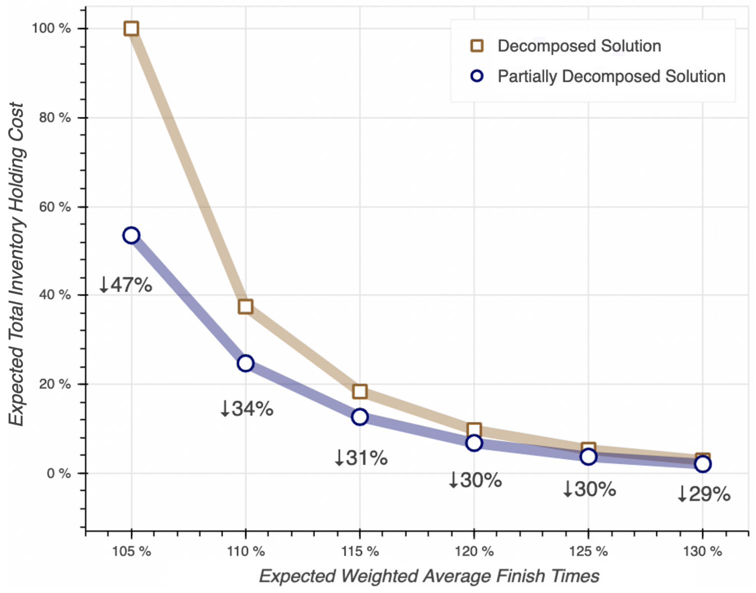


Figure 3.8: Minimum expected inventory holding cost with respect to the forced upper bound on the expected average weighted finish times

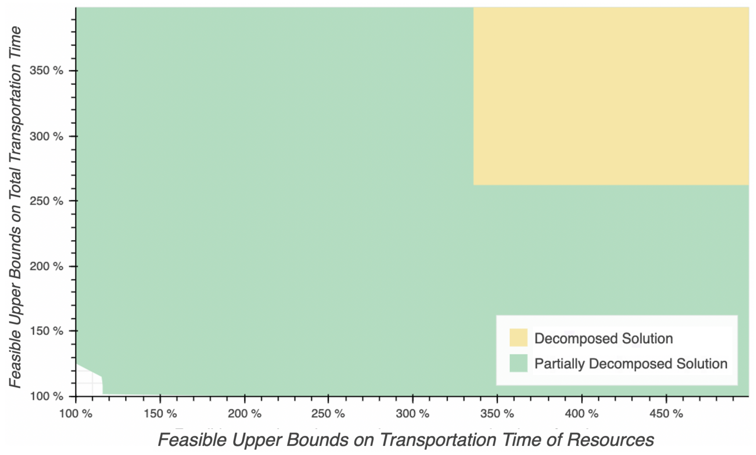


Figure 3.9: Range of feasible upper bounds on the total transportation times and the maximum transportation time of the resources

3.4.3 Impacts of Imposing Constraints on Transportation

In this subsection, the inventory-related constraints are ignored, and the results are based on the best solution that the partially decomposed approach can achieve.

It might be counter-intuitive that a solution with longer total transportation time leads to a lower expected average weighted finish times in a project; however, there is a trade-off between the expected average weighted finish times and the maximum allowed total transportation time. To illustrate the trade-off, we place upper bounds on the total transportation time and calculate the minimum expected average weighted finish times. The result is depicted in Figure 3.10, which shows that the expected average weighted finish times reduce 26% by allowing a 5% increase in the total transportation time compared to the schedule that achieves the minimum possible total transportation time. Also, increasing the total transportation time by 15% and 100% decrease the expected average weighted finish times 30% and 38%, respectively.

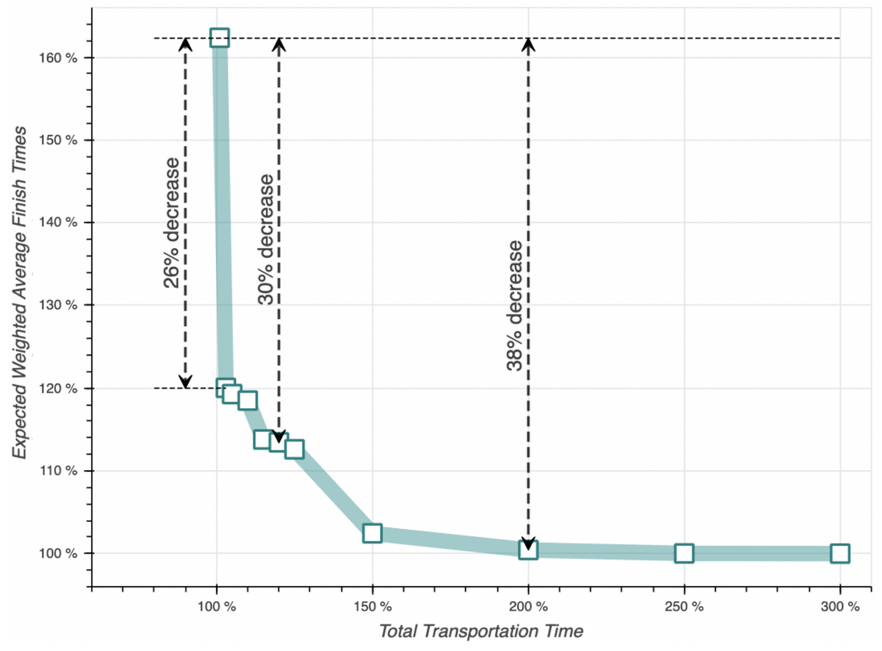


Figure 3.10: Efficient frontier of the upper bound on the total transportation time and the expected average weighted finish times

Limiting the total transportation time requires a holistic view; without it, approaches used to minimize it are likely to be myopic. For instance, in a project with several resources, a myopic approach tries to minimize the transportation time of one resource at a time. In comparison, an advanced holistic approach might choose to increase the transportation time of one resource to reduce to total transportation time of the whole project.

Myopic limits on the transportation of resources can be modeled by two types of constraints. First, one might prevent resources from having any trip longer than a threshold. In other words, an upper bound on the length of each trip might be enforced. Second, one might restrict the transportation time of each resource by enforcing an upper bound on it.

The efficient frontier of total transportation time versus the expected average weighted finish times in the presence of upper bounds on the maximum length of each trip is shown in Figure 3.11. Similarly, the efficient frontier in the presence of upper bounds on the transportation time of each resource is depicted in Figure 3.12.

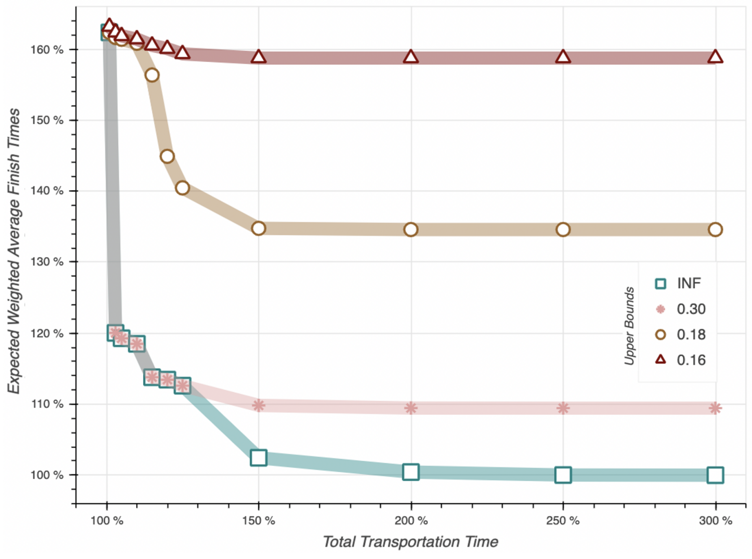


Figure 3.11: Efficient frontier of the upper bound on the total transportation time vs. the expected average weighted finish times. Distinct colors indicate different values of upper bound on the maximum length of trips.

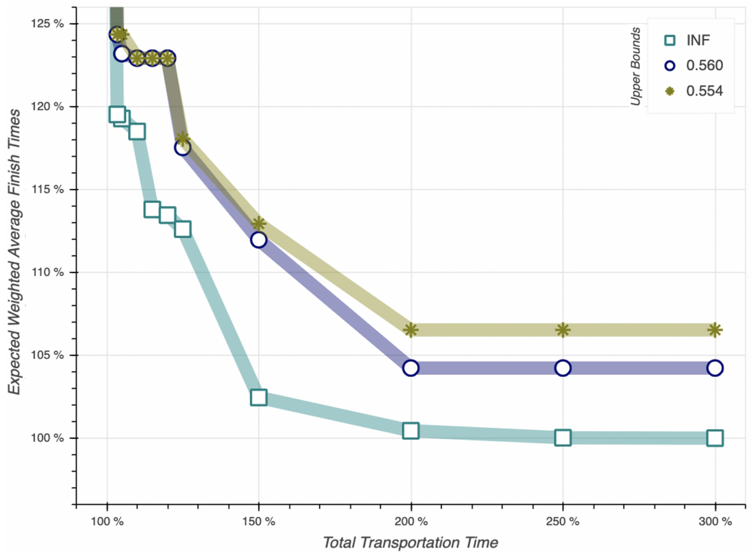


Figure 3.12: Efficient frontier of the upper bound on the total transportation time vs. the expected average weighted finish times. Distinct colors indicate different values of upper bound on the maximum transportation time of resources.

Figures 3.11 and 3.12 demonstrate that by imposing tighter upper bounds, the performance of the solution gets worst in terms of both the expected average weighted finish times and the total transportation time. Figure 3.11 shows that imposing a tight upper bound on the maximum trip length increases the expected average weighted finish times up to 60%, and Figure 3.11 indicates that forcing a tight upper bound on the transportation time of resources increases the expected average weighted finish times up to 10%. As a result, restricting the maximum length of trips has a more substantial detrimental impact than restricting the transportation time of resources.

3.4.4 Value of Acquiring Additional Resources

As mentioned above, the required resources for drilling operations are costly, and therefore, only a limited number of these resources are available at each project. To analyze the benefits of providing an additional resource, we calculate the incremental decrease in the expected average weighted finish times per additional resource in our case study. Figure 3.13 presents the results of these calculations. The marginal and cumulative decreases in the expected average weighted finish times shed light on the trade-off between the cost and benefit of acquiring new resources.

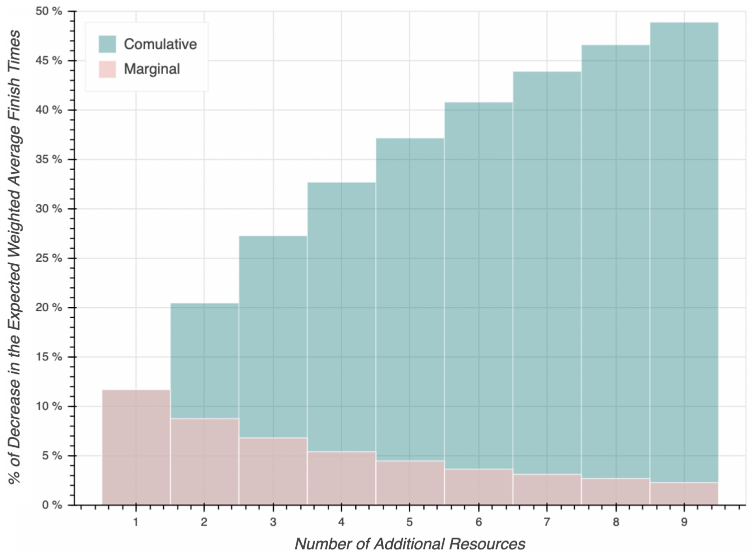


Figure 3.13: Percentage of decrease in the expected average weighted finish times per additional resource

Chapter 4

Online Scheduling to Minimize Total Weighted (Modified) Earliness and Tardiness Cost

4.1 Introduction and Problem Description

We consider the online scheduling of a single machine, to which jobs with distinct weights w_j, and due dates d_j arrive over time. The jobs must be processed one at a time on the machine without preemption, in order to minimize the total weighted earliness and tardiness cost. This problem is denoted $1|r_j, d_j, online|\sum_j w_j\ (\zeta_E\ E_j + \zeta_T T_j)$, where r_j is the arrival time of each job (also called the release date), E_j and T_j are the earliness and tardiness of job j, and ζ_E and ζ_T are the penalties per unit time associated with earliness and tardiness, respectively.

We focus on an online setting, where no information about future arrivals, even the number of jobs that will arrive, is available. The weight, due date, and processing time of each job are available upon that job's arrival, and our goal is to develop a scheduling policy that is effective in this online setting.

Typically, the performance of online scheduling policies is analyzed using the competitive ratio, which was first formally introduced by Sleator and Tarjan [69]. If the objective function value of the schedule produced by online policy ψ for the instance I is C_I^ψ, and the optimal offline objective function value of the instance is C_I^*, then the competitive ratio of policy ψ, ρ^ψ, is defined as:

$$\rho^\psi = \sup_{I \in \mathcal{I}} \{\rho_I^\psi\},$$

where $\mathcal{I}$ is the set of all possible instances and for each instance I, ρ_I^ψ is defined as:

$$\rho_I^\psi = \inf\{\rho \,|\, C_I^\psi \leq \rho \times C_I^*\}.$$

If there exists an online policy ψ with a competitive ratio of ρ for a problem, then the problem is called ρ-competitive. On the other hand, if no online policy can exist with a competitive ratio smaller than ρ', then ρ' is called a lower bound for the problem. Formally, ρ' is a lower bound if and only if:

$$\rho' \leq \inf_{\psi \in \Psi} \{\rho^{\psi}\},$$

where Ψ is the set of all online policies. Therefore, an online policy is called optimal for a problem if its competitive ratio matches a lower bound. Note that here, optimal indicates that no online policy can have a smaller competitive ratio. The goal of minimizing the competitive ratio can be interpreted as minimizing the maximum relative regret of the objective function.

In this chapter, we prove that the competitive ratio of $1|r_j, d_j, online|\sum_j w_j(\zeta_E E_j + \zeta_T T_j)$ is unbounded. Given this we consider a *modified* version of the objective function, which was first introduced by Kolliopoulos and Steiner [48], in which a constant, $\sum_j w_j(\zeta_d d_j)$, where d_j is the due date of job j and ζ_d is a constant coefficient, is added to the objective function. The modified problem is thus denoted $1|r_j, d_j, online|\sum_j w_j(\zeta_E \ E_j + \zeta_T T_j + \zeta_d d_j)$. Since a constant is added to the objective function, for each instance the optimal schedule (the schedule that minimizes the objective function) remains the same for this modified problem. However, adding this allows us to find a policy with a finite competitive ratio.

When all release dates are equal to zero, so that all information is available at the start of the horizon, the online and offline versions of this model are equivalent. For this case, the objective function can be viewed as minimizing the work-in-process (WIP) inventory cost as well as the tardiness cost (in a Just-In-Time (JIT) environment, for example). Since in a JIT environment no product can be delivered prior to its due date, $w_j\zeta_d$ can be viewed as the inventory cost of unprocessed product j, $w_j(\zeta_T - \zeta_d)$ can be viewed as the cost of delay for product j, and $w_j(\zeta_E + \zeta_d)$ can be viewed as the inventory cost of completed product j. In this case, the objective function of $\sum_j w_j(\zeta_E E_j + \zeta_T T_j + \zeta_d d_j)$ is WIP inventory cost and delay cost.

4.2 Literature Review

Pruhs, Sgall, and Torng [65] categorize common online scheduling frameworks. In the *online over time* framework, jobs are assumed to arrive over time, and no information about any job is available before its arrival. In addition, the total number of jobs is unknown. In this chapter, we consider the non-preemptive online over time scheduling problem detailed in the previous section.

Vestjens [82] showed that for the online over time version of problem of minimizing the total completion times, denoted $1|r_j, online|\sum_j C_j$, the Delayed Shortest Processing Time (DSPT) policy is optimal and its competitive ratio is two. The weighted version of that problem, $1|r_j, online| \sum_j w_j C_j$, was studied by Anderson and Potts [1], who showed that the

Delayed Weighted Shortest Processing Time (DWSPT) policy is optimal and its competitive ratio is two. The DSPT and DWSPT policy are described in detail in Section 4.3.

To the best of our knowledge, Liu et al. [53] is the first work that considered tardiness in an 'online over time' setting. They proved that the competitive ratio of $1|r_j, d_j, online|\sum_j w_j T_j$ is unbounded, and then minimized the modified total tardiness, denoted $1|r_j, d_j, online|\sum_j (T_j + d_j)$. They showed that two is a lower bound for this problem, and that the competitive ratio of DSPT policy is at most three for this problem. They extended their work by considering different weights for jobs, $1|r_j, d_j, online| \sum_j w_j(T_j + d_j)$, and proved the competitive ratio of DWSPT policy is at most three for this problem.

In this chapter, we introduce a new policy, the list-based delayed shortest processing time (LDWSPT) policy, and we develop lower and upper bounds on its performance for several related problems. We summarize the state-of-the-art in terms of bounds on the optimal competitive ratio of these problems, along with our new results, in Table 4.1.

Table 4.1: State of the art bounds on the optimal competitive ratio.

Objective Function	Literature			Our result		
	LB	UB	Gap%	LB	UB	Gap%
$\sum_j (T_j + d_j)$	2 [a]	3 [a]	50%	2	2	0%
$\sum_j w_j(T_j + d_j)$	2 [a]	3 [a]	50%	2	2	0%
$\sum_j (E_j + T_j + d_j)$	N/A	N/A	N/A	2	2	0%
$\sum_j w_j(E_j + T_j + d_j)$	N/A	N/A	N/A	2	2	0%
$\sum_j w_j(\zeta_E E_j + \zeta_T T_j + \zeta_d d_j)$	N/A	N/A	N/A	x [b]	y [c]	0% [d]

[a] Liu et al. [53]

[b] $x = \max\{2, 1 + \zeta_T/\zeta_d\}$

[c] $y = \max\{2, 1 + \zeta_E/\zeta_d, 1 + \zeta_T/\zeta_d\}$

[d] when $\zeta_E \leq \max\{\zeta_T, \zeta_d\}$

The remainder of this chapter is organized as follows. In Section 4.3, we review the DSPT and the DWSPT policies and introduce the LDWSPT policy. In Section 4.4, we investigate the problem of minimizing total weighted (modified) earliness and tardiness and provide its lower and upper bounds. In Section 4.4, we assume that the cost of earliness and tardiness are equal. In Section 4.5, we relax this assumption and investigate the problem of minimizing total weighted (modified) unequal earliness and tardiness cost and provide its lower and upper bounds.

4.3 DSPT, DWSPT, and LDWSPT Policies

For the sake of completeness, we review the DSPT policy, proposed by Vestjens [82] and the DWSPT policy. Then, we propose a new policy, the LDWSPT policy.

Algorithm 1 Delayed Shortest Processing Time (DSPT) Policy

```
Each time the machine becomes available:
  Step 1:
    If there is no available job, wait until a job arrives.
  Step 2:
    Among available jobs, SELECT a job, $J_j$, with the smallest processing time and where
    there are ties, SELECT the one with the lowest index where numerical indices are arbi-
    trarily assigned to the jobs.
  Step 3:
    If $p_j \leq t$, where $t$ is the time since the start of the time horizon, start processing on $J_j$.
    Otherwise, wait until $t = p_j$ or until a new job arrives, whichever happens first, and then,
    go to step 1.
```

Here, p_j is the processing time of J_j. Note that in general p_j will only be greater than t at the start of the time horizon, so after some initial period, this policy performs similarly to the Shortest Processing Time (SPT) policy. By delaying the processing of jobs at the beginning of the time horizon, we can avoid certain degenerate cases in our analysis.

Algorithm 2 Delayed Weighted Shortest Processing Time (DWSPT) Policy

```
Each time the machine becomes available:
  Step 1:
    If there is no available job, wait until a job arrives.
  Step 2:
    Among available jobs, SELECT a job, $J_j$, with the smallest ratio of $p_j/w_j$, and where
    there are ties, SELECT the job with the smallest processing times and then the lowest
    index where numerical indices are arbitrarily assigned to jobs.
  Step 3:
    If $p_j \leq t$, where $t$ is the time since the start of the time horizon, start processing on $J_j$.
    Otherwise, wait until $t = p_j$ or until a new job arrives, whichever happens first, and then,
    go to step 1.
```

We utilize a list in the LDWSPT policy in order to ensure that the relative sequence of any two jobs remains the same once they are added to the list. LDWSPT is a useful policy for analysis. Since ties are broken arbitrarily, stronger inductions can be used in proofs. Note that because ties are broken arbitrarily in the LDWSPT policy, an instance can have various LDWSPT schedules. In particular, for each instance, if we break ties based

Algorithm 3 List-based Delayed Weighted Shortest Processing Time (LDWSPT) Policy

```
Create an empty list, $L$.
Each time the machine becomes available:
  Step 1:
    INSERT all new jobs (jobs that have been released but have not yet been added to the
    list) to $L$ such that the list is arranged in order of increasing $p_j/w_j$. Break ties arbitrarily.
  Step 2:
    If $L$ is empty, wait until a job arrives and then, go to Step 1.
    If $L$ is not empty, SELECT the first job, $J_j$, in $L$.
  Step 3:
    If $p_j \leq t$, where $t$ is the time since the start of the time horizon, start processing on $J_j$
    and REMOVE it from $L$.
    Otherwise, wait until $t = p_j$ or until a new job arrives, whichever happens first, and then,
    go to step 1.
```

on the smallest processing time and then the lowest index, the LDWSPT policy generates the DWSPT schedule. Hence, any upper bound on the competitive ratio of LDWSPT policy is also an upper bound on the competitive ratio of DWSPT policy.

4.4 Total Weighted (Modified) Earliness and Tardiness

In the online framework, as we show in Lemma 15, there is no finite upper bound on the competitive ratio of the unmodified version of our problem:

Lemma 15. *There is no finite upper bound on the competitive ratio of* $1|r_j, d_j, online|\sum_j w_j (E_j + T_j)$.

Proof. Consider two instances I_1 and I_2:

Instance I_1: Only one job arrives, with:
$J_1 : r_1 = 0, d_1 = 1, p_1 = 1, w_1 = 1.$

Instance I_2: N jobs arrive, with:
$J_1 : r_1 = 0, d_1 = 1, p_1 = 1, w_1 = 1;$
J_j for $j \in \{2, ..., N\}$: $r_j = \epsilon, d_j = 0, p_j = \epsilon, w_j = 1;$
where $\epsilon = 1/N^2$.

In the online setting, for both instances a single job with identical characteristics arrives at time zero, so it is impossible to distinguish between the instances at the start of the horizon. The competitive ratio of any policy that starts processing after time zero with the given information at the start of the horizon goes to infinity, because for the instance I_1, the objective function value of the policy is infinitely worse than the optimal offline objective function value. Similarly, the competitive ratio of any policy that starts processing at time zero with the given information at the start of the horizon goes to infinity. This is because for the instance I_2, the objective function value of any policy that starts processing at time zero is infinitely worse than the optimal offline objective function value as N increases to infinity.

Hence, the competitive ratio of all policies goes to infinity, as for any given policy, there exists an instance for which the objective function value of the policy is infinite times worst than the optimal offline objective function value. $\square$

Thus, in the modified version of the problem, in order to allow the possibility of policies with finite competitive ratios, a constant, the weighted sum of the due dates, is added to the objective function, resulting in problem $1|r_j, d_j, online|\sum_j w_j(E_j + T_j + d_j)$. This modified objective function was first introduced by Kolliopoulos and Steiner [48], and is also used by Liu et al. [53]. Note that the additional term in the objective function is a constant independent of the policy, and so the optimal solution in this modified version of problem (the solution minimizing the value of the objective function) remains the same.

In Subsection 4.4.1, we show there is no online policy with a competitive ratio of less than two for this modified problem. In other words, we show that two is a lower bound for this problem. In Subsection 4.4.2, we show that for this problem, LDWSPT policy is 2-competitive. Therefore, it is an optimal policy for this problem.

4.4.1 Lower Bound

Our lower bound is based on two lemmas:

Lemma 16. $\sum_j w_j(T_j + d_j) = \sum_j w_j \max_j(C_j, d_j)$.

Proof. If $C_j \leq d_j$, then $T_j = 0$ and $\max_j(C_j, d_j) = d_j$. So, $w_j(T_j + d_j) = w_j d_j = w_j \max(C_j, d_j)$.

If $C_j > d_j$, then $T_j = C_j - d_j$ and $\max(C_j, d_j) = C_j$. So, $w_j(T_j + d_j) = w_j C_j = w_j \max(C_j, d_j)$.

The result follows from summing over all jobs. $\square$

Lemma 17. *Two is a lower bound for the minimization of* $1|r_j, d_j, online|\sum_j(E_j + T_j + d_j)$ *and consequently for the minimization of* $1|r_j, d_j, online|\sum_j w_j(E_j + T_j + d_j)$.

Proof. By Lemma 16,

$$\sum_j (E_j + T_j + d_j) = \sum_j \left(\max(C_j, d_j) + E_j \right).$$

When all due dates equal zero, the problem is equivalent to the minimization of $\sum_j C_j$, which Vestjens [82] show has a lower bound of two. Therefore, two is a lower bound for $\sum_j(E_j + T_j + d_j)$. Additionally, when all w_j equal one, $\sum_j w_j(E_j + T_j + d_j)$ is equivalent to to $\sum_j(E_j + T_j + d_j)$; thus, two is its lower bound. $\square$

4.4.2 Competitiveness of the LDWSPT Policy

We prove that the LDWSPT policy is 2-competitive. The proof is based on contradiction, where we assume there exists an instance, I, for which $\rho_I^{LDWSPT} > 2$.

Definition 1. *"LDWSPT schedule" refers to the schedule of jobs resulting from applying an LDWSPT policy and "LDWSPT sequence" refers to the sequence of jobs in the LDWSPT schedule.*

Definition 2. *"Counterexample" refers to an instance, I, for which $\rho_I^{LDWSPT} > 2$.*

Definition 3. *A "block" of jobs refers to a set of jobs that are processed consecutively (with no idle time between them).*

Lemma 18. *If there exists a counterexample, the LDWSPT schedule of the counterexample, which has the smallest possible number of jobs among all counterexamples, is composed of only one block of jobs.*

Proof. By contradiction, assume there exists a counterexample, I_0, with the smallest possible number of jobs for which the LDWSPT schedule is composed of at least two blocks of jobs (similar to Figure 4.1). Since I_0 is a counterexample:

$$\sum_{j\in I_0} w_j(E_j + T_j + d_j) = C_{I_0}^{LDWSPT} > 2C_{I_0}^* = 2\sum_{j\in I_0} w_j(E_j^* + T_j^* + d_j) \tag{4.1}$$

where E_j and T_j denote the earliness and tardiness of job j in the LDWSPT schedule of I_0, and E_j^* and T_j^* denote the earliness and tardiness of job j in the optimal offline schedule of I_0.

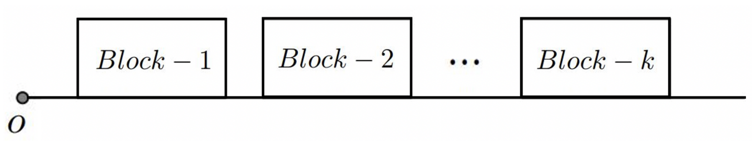


Figure 4.1: Illustration of an instance for which the LDWSPT schedule consist of k blocks of jobs.

Split the counterexample into two separate instances: I_1 and I_2. I_1 consists of all jobs in the first block, and I_2 consists of all other jobs.

There exists an LDWSPT schedule for I_1 in which the start times of jobs are the same as their start times in the LDWSPT schedule of I_0. In the LDWSPT schedule of I_0, there is idle time before the start of the first job in the second block. Immediately before the start of the first job in the second block, either there is no available job or the SELECTED job, J_j, cannot be processed because $p_j > t$. In the latter case, we create a new instance $I_{2'}$ by setting the release date of the SELECTED job (r_j) to zero. In the former case, $I_{2'}$ is the same as I_2. Our goal here is to ensure that the first job in the LDWSPT schedule of $I_{2'}$ is not processed before the first job in the second block of the LDWSPT schedule of I_0. Hence, we can claim there exists an LDWSPT schedule of $I_{2'}$ in which the start times of jobs are the same as their start times in the LDWSPT schedule of I_0. Thus,

$$C_{I_1}^{LDWSPT} = \sum_{j \in I_1} w_j(E_j + T_j + d_j)$$

$$C_{I_{2'}}^{LDWSPT} = \sum_{j \in I_{2'}} w_j(E_j + T_j + d_j)$$

In addition:

$$C_{I_1}^* \leq \sum_{j \in I_1} w_j(E_j^* + T_j^* + d_j)$$

$$C_{I_{2'}}^* \leq \sum_{j \in I_{2'}} w_j(E_j^* + T_j^* + d_j)$$

$$C_{I_1}^* + C_{I_{2'}}^* \leq C_{I_0}^*$$

Consequently, based on Equation 4.1:

$$C_{I_1}^{LDWSPT} + C_{I_{2'}}^{LDWSPT} = C_{I_0}^{LDWSPT} > 2C_{I_0}^* \geq 2C_{I_1}^* + 2C_{I_{2'}}^*$$

Therefore, either $C_{I_1}^{LDWSPT} > 2C_{I_1}^*$ or $C_{I_{2'}}^{LDWSPT} > 2C_{I_{2'}}^*$. In other words, either I_1 or $I_{2'}$ is a counterexample which contradicts our assumption that I_0 is a counterexample with the smallest possible number of jobs. □

Lemma 19. *If there exists a counterexample, there also exists a counterexample (with the same number of jobs) which has an optimal offline schedule in which all jobs finish at their due dates.*

Proof. Consider a counterexample, I, which has an optimal offline schedule with at least one job, J_j, that does not finish at its due date. Define C_j and Z_j as the completion time of J_j in the optimal offline schedule and the LDWSPT schedule, respectively. Create I_{new} by altering I as follows:

- If $C_j < d_j$ and $Z_j \geq d_j$, set $d_j^{new} = d_j - x$, where $x = d_j - C_j > 0$. I_{new} is also a counterexample because:

$$C_{I_{new}}^{LDWSPT} = C_I^{LDWSPT} > 2(C_I^* - 2w_j x) \geq 2C_{I_{new}}^*$$

- If $C_j > d_j$ and $Z_j \leq d_j$, set $d_j^{new} = d_j + x$, where $x = C_j - d_j > 0$. I_{new} is also a counterexample because:

$$C_{I_{new}}^{LDWSPT} = C_I^{LDWSPT} + 2w_j x > 2C_I^* \geq 2C_{I_{new}}^*$$

- If $C_j < d_j$ and $Z_j < d_j$, set $d_j^{new} = d_j - x - y$, where $y = \max(0, Z_j - C_j)$, and $x = d_j - C_j - y$. I_{new} is also a counterexample because:

$$C_{I_{new}}^{LDWSPT} = C_I^{LDWSPT} - 2w_jx > 2(C_I^* - 2w_jx - 2w_jy) \geq 2C_{I_{new}}^*$$

- If $C_j > d_j$ and $Z_j > d_j$, set $d_j^{new} = d_j + x + y$, where $y = \max(0, C_j - Z_j)$, and $x = C_j - d_j - y$. I_{new} is also a counterexample because:

$$C_{I_{new}}^{LDWSPT} = C_I^{LDWSPT} + 2w_jy > 2C_I^* \geq 2C_{I_{new}}^*$$

As a result, I_{new} is a counterexample in which there are fewer jobs that do not finish at their due dates in the optimal offline schedule than in I. By repeating this procedure, we can create a counterexample in which all jobs finish at their due dates in the resulting optimal offline schedule. □

Lemma 20. *If there exists a counterexample which has an optimal offline schedule in which all jobs finish at their due dates, there also exists a counterexample (with the same number of jobs) which has an optimal offline schedule in which all jobs finish at their due dates and there is no feasible way to start any job earlier without changing the sequence of jobs.*

Proof. Consider a counterexample, I, which has an optimal offline schedule in which all jobs finish at their due dates and at least one job, J_j, can start x_j time unit earlier without changing the sequence of jobs.

We create I_{new} by setting $d_j^{new} = d_j - x_j$ in I. Note that J_j starts x_j time unit earlier in the optimal offline schedule of I_{new} than in the optimal offline schedule of I; hence, it finishes at its due date in I_{new}. In addition, I_{new} is a counterexample, because:

- If $Z_j < d_j$:

$$C_{I_{new}}^{LDWSPT} \geq C_I^{LDWSPT} - 2w_jx_j > 2(C_I^* - w_jx_j) = 2C_{I_{new}}^*$$

- If $Z_j \geq d_j$:

$$C_{I_{new}}^{LDWSPT} = C_I^{LDWSPT} > 2(C_I^* - w_jx_j) = 2C_{I_{new}}^*$$

As a result, I_{new} is a counterexample that has an optimal offline schedule in which all jobs finish at their due dates. Also, $\sum_j d_j^{new} < \sum_j d_j$. Since there is a lower bound on the summation of the the due dates of all jobs, repeatedly applying this procedure results in a counterexample in which all jobs finish at their due dates in the optimal offline schedule and there is no feasible way without changing the sequence to start any job earlier.

□

Definition 4. *We define "distinctive counterexample" to be a counterexample with the following properties:*

1. *The number of jobs is the smallest possible in a valid counterexample.*
2. *Jobs finish at their due dates in the optimal offline schedule.*
3. *In the optimal offline schedule, there is no feasible way, without changing the sequence of jobs, to start any job earlier.*

It follows from Lemmas 19 and 20 that a distinctive counterexample exists if and only if any counterexample exists. In Subsection 4.4.2, we prove that for the special case, in which all jobs have equal ratios of p_j/w_j, no distinctive counterexample exists. By extending the result of this special case, in Subsection 4.4.2, we prove that no distinctive counterexample exists for which jobs are arranged in increasing ratios of p_j/w_j in the LDWSPT schedule. Lastly, in Subsection 4.4.2 we extent the results of the special cases and prove that no distinctive counterexample and consequently, no counterexample exists in the general case, which proves the LDWSPT policy is 2-competitive for the minimization of $1|r_j, d_j, online|\sum_j w_j(E_j+T_j+d_j)$.

Special Case: When All Jobs Have Equal Ratios of p_j/w_j

Consider an instance I in which all jobs have equal ratios of p_j/w_j. For simplicity, from now on, we refer to the ratio of p_j/w_j as the "*ratio.*" Let κ represents the greatest common divisor of processing times of all jobs in instance I. We create a new instance, I', that we refer to as the "*split version*", in which we divide each job of instance I (e.g., J_j) into $n_j = p_j/\kappa$ jobs ($J'_{j1}, J'_{j2}, ..., J'_{jn_j}$), and we refer to them as *sub-jobs*. For each of the sub-jobs, we set the processing time to $p' = \kappa$, the weight to $w' = \kappa.w_j/p_j$, and the release date to zero. Also, the due date of each sub-job that is created by dividing job j of instance I is d_j. Note that the weights and processing time of all sub-jobs of instance I' are equal. As an illustration, consider Example 1.

Example 1. *There exist an instance, I, with three jobs: $p_1 = 2, w_1 = 4, d_1 = 9$; $p_2 = 4, w_2 = 8, d_2 = 7$; and $p_3 = 3, w_3 = 6, d_3 = 3$. Note that the ratio of all jobs are equal ($\frac{p_j}{w_j} = 0.5 \quad \forall j$).*

To create instance I', job one is divided into two sub-jobs, J'_{11}, J'_{12}; job two is divided into four sub-jobs, J'_{21}, J'_{22}, J'_{23}, J'_{24}; and job three is divided into three sub-jobs, J'_{31}, J'_{32}, J'_{33}. Note that all sub-jobs have equal processing times $p' = \kappa = 1$, and equal weights $w' = \kappa.w_j/p_j = 2$, and their release dates are zero. The due date of sub-jobs J'_{11}, J'_{12} is $d_1 = 9$, sub-jobs J'_{21}, J'_{22}, J'_{23}, J'_{24} is $d_2 = 7$, and sub-jobs J'_{31}, J'_{32}, J'_{33} is $d_3 = 3$.

Definition 5. *Consider instance I in which all jobs have equal ratios, and its split version, I'. The "LDWSPT-equivalent schedule" for the instance I' is a schedule in which the start time of sub-job J'_{jk} is $S_j + p'(k-1)$, where S_j is the start time of J_j in the LDWSPT schedule for instance I. Similarly, the "optimal-equivalent schedule" for the instance I' is a schedule in which the start time of sub-job J'_{jk} is $S^*_j + p'(k-1)$, where S^*_j is the start time of J_j in the optimal offline schedule of instance I.*

As an illustration, for Example 1, the LDWSPT schedule for the instance I as well as the LDWSPT-equivalent schedule for the instance I' are depicted in schedule (a) of Figure 4.2. Also, the optimal offline schedule of the instance I as well as the optimal-equivalent schedule for the instance I' are illustrated in the schedule (b) of Figure 4.2.

Similar to Figure 4.2, let Z_j denotes the completion time of J_j in the LDWSPT schedule, C_j denotes the completion time of J_j in the optimal offline schedule, z'_{jk} denotes the completion time of sub-job J'_{jk} in the LDWSPT-equivalent schedule, and c'_{jk} denotes the completion time of sub-job J'_{jk} in the optimal-equivalent schedule. Note that in Figure 4.2, the indices are assigned based on the sequence of appearance of jobs in the LDWSPT schedule. In addition, let s refers to the start time of the first job in the LDWSPT schedule, and q refers to s minus the start time of the first job in the optimal offline schedule.

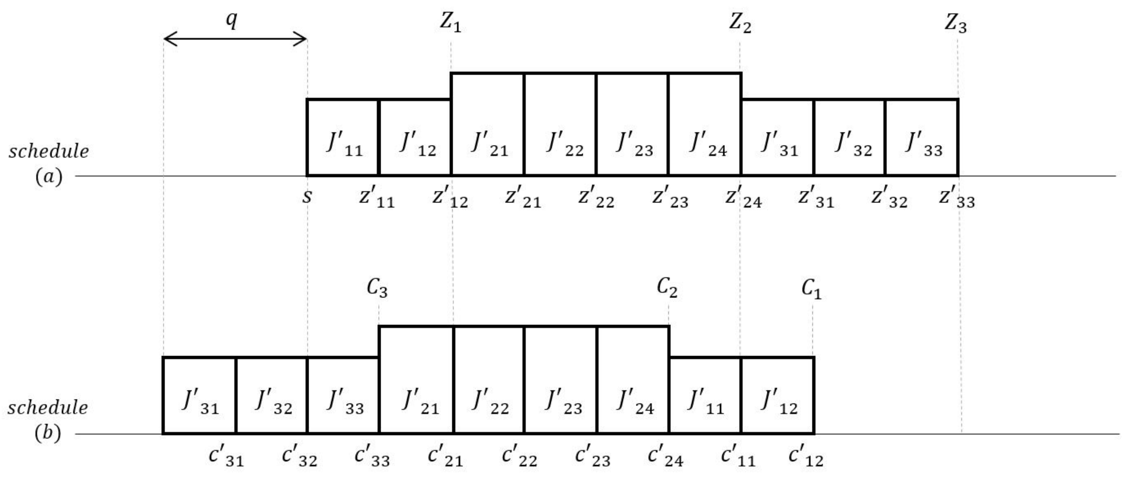


Figure 4.2: Illustration of an instance of three jobs with equal ratios of p_j/w_j, and its split version.

Lemma 21. *$q \leq p_{max}$ where $p_{max} = \max_j p_j$.*

Proof. Since jobs are ordered in a list in the LDWSPT policy, the relative position of any two jobs does not change after they join the list. Hence, there are two possibilities for the start time of the first job (J_j) in the LDWSPT schedule. First, $s = p_j$. In this case, $q \leq p_j \leq p_{max}$. Second, $s = r_j$. In this case, when $r_j \leq p_{max}$, it is trivially true that $q \leq r_j \leq p_{max}$; and when $r_j > p_{max}$, we conclude that no job was available before r_j; otherwise, it would processed before r_j. So, the processing starts at least at time r_j in the optimal offline schedule and in this case $q \leq 0 \leq p_{max}$.

□

Definition 6. *For any given instance, I, in which jobs have equal ratios, and its split version, I', a "sub-job schedule" refers to any arbitrary schedule for sub-jobs of instance I'. We call* *a sub-job schedule "feasible" if the first sub-job does not start before* $\max\{0, s - p_{max}\}$*. We define a new metric,* $\hat{\rho}_{I'}(\phi)$*, for any feasible sub-job schedule,* ϕ*, for the instance* I' *as:*

$$\hat{\rho}_{I'}(\phi) = \frac{\sum_j \left(\sum_{j'=1}^{n_j} w'(\mid z'_{jj'} - c_{jj'} \mid + c_{jj'}) + A(p_j) \right)}{\sum_j \left(\sum_{j'=1}^{n_j} w' c_{jj'} + A(p_j) \right)} = 1 + \frac{\sum_j \sum_{j'=1}^{n_j} w' \mid z'_{jj'} - c_{jj'} \mid}{\sum_j \left(\sum_{j'=1}^{n_j} w' c_{jj'} + A(p_j) \right)},$$

where $c_{jj'}$ *is the completion time of sub-job* $J'_{jj'}$ *in the sub-job schedule* ϕ *for the instance* I'*, and* $A(p_j) = \frac{w' p_j (p_j - p')}{2p'}$*.*

Lemma 22. *For any distinctive counterexample,* I*, in which jobs have equal ratios, and its split version,* I'*, there exists a feasible sub-job schedule,* ϕ*, for which* $\hat{\rho}_{I'}(\phi) \geq \rho_I^{LDWSPT}$*.*

Proof. To prove there exists a feasible sub-job schedule, ϕ, for which $\hat{\rho}_{I'}(\phi) \geq \rho_I^{LDWSPT}$, we show for the optimal-equivalent schedule, ϕ', $\hat{\rho}_{I'}(\phi') = \rho_I^{LDWSPT}$.

The optimal-equivalent schedule for I' is a feasible sub-job schedule because the first job in the optimal offline schedule of I starts at $s - q$, which is based on Lemma 21 greater than or equal to $\max\{s - p_{max}, 0\}$.

In the optimal-equivalent schedule of I':

$$E_j + T_j = \mid Z_j - d_j \mid = \mid z'_{jj'} - c'_{jj'} \mid \quad \forall j, j'$$

$$w_j(E_j + T_j) = \sum_{j'=1}^{n_j} w' \mid z'_{jj'} - c'_{jj'} \mid \quad \forall j$$

$$\sum_j w_j(E_j + T_j) = \sum_j \sum_{j'=1}^{n_j} w' \mid z'_{jj'} - c'_{jj'} \mid .$$

In addition:

$$d_j = C_j = c'_{jn_j} = c'_{jj'} + p'(n_j - j')$$

$$w_j d_j = \sum_{j'=1}^{n_j} w' c'_{jj'} + w'p' \sum_{j'=1}^{n_j} (n_j - j')$$

$$w'p' \sum_{j'=1}^{n_j} (n_j - j') = \frac{{p'}^2 w_j (n_j^2 - n_j)}{2p_j} = \frac{w_j(p_j - p')}{2} = A(p_j)$$

$$w_j d_j = \sum_{j'=1}^{n_j} w' c'_{jj'} + A(p_j)$$

$$\sum_j w_j d_j = \sum_j \big(\sum_{j'=1}^{n_j} w' c'_{jj'} + A(p_j)\big);$$

hence:

$$\hat{\rho}_{I'}(\phi') = 1 + \frac{\sum_j \sum_{j'=1}^{n_j} w' \mid z'_{jj'} - c'_{jj'} \mid}{\sum_j \big(\sum_{j'=1}^{n_j} w' c'_{jj'} + A(p_j)\big)} = 1 + \frac{\sum_j w_j (E_j + T_j)}{\sum_j w_j d_j} = \rho_I^{LDWSPT}$$

□

Definition 7. *For any given sequence of sub-jobs, we define an "efficient sub-job schedule" as a specific feasible sub-job schedule in which the first sub-job of the sequence starts at time* $\max(0, s - p_{max})$*, the second sub-job of the sequence starts at the completion of the first sub-job, the third sub-job of the sequence starts at the completion of the second sub-job, and so on.*

Lemma 23. *For any given instance, I, in which jobs have equal ratios, its split version, I', and any feasible sub-job schedule, ϕ, for which $\hat{\rho}_{I'}(\phi) > 2$:*

$$\hat{\rho}_{I'}(\varphi) \geq \hat{\rho}_{I'}(\phi),$$

where φ is an efficient sub-job schedule for the instance I', in which the sequence of sub-jobs is the same as the sequence of sub-jobs in ϕ.

Proof. When both the efficient sub-job schedule and any arbitrary feasible sub-job schedule have the same sequence of sub-jobs, the start time of each sub-job in the efficient sub-job schedule is less than or equal to its start time in the arbitrary feasible sub-job schedule.

The remainder of the proof is similar to the proof of Lemma 20. □

Lemma 24. *For any given instance, I, in which jobs have equal ratios, and its split version, I':*

$$\hat{\rho}_{I'}(\varphi^*) = \max_{\varphi \in \Phi} \hat{\rho}_{I'}(\varphi),$$

where φ^* is the efficient sub-job schedule in which the sequence of sub-jobs is the reverse of the sequence of sub-jobs in the LDWSPT-equivalent schedule, and Φ represent the set of all possible efficient sub-job schedules for the instance I'.

Proof. By contradiction, assume:

$$\hat{\rho}_{I'}(\varphi^*) \neq \max_{\varphi \in \Phi} \hat{\rho}_{I'}(\varphi) = \hat{\rho}_{I'}(\varphi').$$

Clearly, there exists at least a pair of sub-jobs (e.g., $J'_{jj'}$ and $J'_{kk'}$) that $J'_{jj'}$ appears before $J'_{kk'}$ in both φ' and the LDWSPT-equivalent schedule. We create a new efficient sub-job schedule, φ'', by substituting sub-jobs $J'_{jj'}$ and $J'_{kk'}$ in the φ'. With straightforward algebra it is easy to see that:

$$\hat{\rho}_{I'}(\varphi'') \geq \hat{\rho}_{I'}(\varphi').$$

By repeating this procedure until no pair of sub-jobs exists that has the same relative sequence in both efficient sub-job schedule and the LDWSPT-equivalent schedule, we can shows that $\hat{\rho}_{I'}(\varphi^*) = \max_{\varphi \in \Phi} \hat{\rho}_{I'}(\varphi)$.

□

Lemma 25. *For any given instance, I, in which jobs have equal ratios, its split version, I', and any sub-job schedule, ϕ:*

$$\hat{\rho}_{I'}(\phi) \leq 2.$$

Proof. Based on Lemma 24:

$$\hat{\rho}_{I'}(\varphi^*) = \max_{\varphi \in \Phi} \hat{\rho}_{I'}(\varphi),$$

where φ^* is the efficient sub-job schedule in which the sequence of sub-jobs is the reverse of the sequence of sub-jobs in the LDWSPT-equivalent schedule. For φ^*:

When $(\sum_j p_j + p_{max})/p'$ is an even number:

$$\sum_j \sum_{j'=1}^{n_j} w' \mid z'_{jj'} - c_{jj'} \mid \leq \frac{(\sum_j p_j)^2 + p_{max}^2}{2p'/w'},$$

and when $(\sum_j p_j + p_{max})/p'$ is an odd number:

$$\sum_j \sum_{j'=1}^{n_j} w' \mid z'_{jj'} - c_{jj'} \mid \leq \frac{(\sum_j p_j)^2 + p_{max}^2 - p'^2}{2p'/w'}.$$

In addition:

$$\sum_j \sum_{j'=1}^{n_j} w' c_{jj'} \geq \frac{(\sum_j p_j)^2 + (\sum_j p_j)p'}{2p'/w'}$$

$$A(p_{max}) = \frac{p_{max}(p_{max} - p')}{2p'/w'};$$

hence:

$$\sum_j (\sum_{j'=1}^{n_j} w' c_{jj'} + A(p_j)) \geq A(p_{max}) + \sum_j \sum_{j'=1}^{n_j} w' c_{jj'} \geq \sum_j \sum_{j'=1}^{n_j} w' \mid z'_{jj'} - c_{jj'} \mid,$$

which conclude:

$$\hat{\rho}_{I'}(\varphi^*) = 1 + \frac{\sum_j \sum_{j'=1}^{n_j} w' \mid z'_{jj'} - c_{jj'} \mid}{\sum_j (\sum_{j'=1}^{n_j} w' c_{jj'} + A(p_j))} \leq 2$$

Since $\hat{\rho}_{I'}(\varphi^*) \leq 2$, based on Lemma 23, for any sub-job schedule, ϕ, $\hat{\rho}_{I'}(\phi) \leq 2$.

□

Theorem 1. *$\rho_I^{LDWSPT} \leq 2$ for any instance, I, in which all jobs have equal ratios of p_j/w_j.*

Proof. By Lemma 22, for any distinctive counterexample, I, in which jobs have equal ratios, and its split version, I', there exists a sub-job schedule, ϕ, for which:

$$\hat{\rho}_{I'}(\phi) \geq \rho_I^{LDWSPT} > 2.$$

By Lemma 25, for any sub-job schedule, ϕ, $\hat{\rho}_{I'}(\phi)$ cannot be greater than two. As a result, no distinctive counterexample in which all jobs have equal ratios exists. Consequently, by Lemmas 19 and 20, no counterexample in which all jobs have equal ratios exists. □

Special Case: When Jobs are Arranged in Increasing Ratios of p_j/w_j in the LDWSPT Sequence

For each job, J_j, in the LDWSPT schedule of an instance, I, let

$$V_j^I = E_j + T_j - d_j$$

and

$$\hat{V}(I) = \sum_{j \in I} w_j V_j^I.$$

Note that for any instance, I, when all jobs (similar to the distinctive counterexamples) complete at their due dates in the optimal offline schedule, $\rho_I^{LDWSPT} > 2$ is equivalent to $\hat{V}(I) > 0$.

Definition 8. *For any distinctive counterexample, I_0, in which jobs are arranged in increasing ratios of p_j/w_j in its LDWSPT sequence, if $\exists\, k' = \min\{k : p_k/w_k < p_{k+1}/w_{k+1}\}$, we create a new instance, $I_{0'}$, and we refer to it as the "mutated version", by making the following changes to the instance I_0:*

$$w_j^{new} = \frac{p_j \times w_{k'+1}}{p_{k'+1}} \quad \forall j \in \{1, 2, ..., k'\},$$

where the numerical indices of jobs are assigned based on their appearance in the LDWSPT schedule for I_0.

Lemma 26. *For any distinctive counterexample, I_0, in which jobs are arranged in increasing ratios of p_j/w_j in its LDWSPT sequence and $\exists\, k' = \min\{k : p_k/w_k < p_{k+1}/w_{k+1}\}$, its mutated version, $I_{0'}$, is a distinctive counterexample.*

Proof. In the proof, the numerical indices of jobs are assigned based on their appearance in the LDWSPT schedule for I_0. Create a new instance, I_1, that only contains $J_1, J_2, ..., J_{k'}$. As the ratio of all jobs in instance I_1 are equal, by Theorem 1, $\rho_{I_1}^{LDWSPT} \leq 2$.

It is easy to see that there exists an LDWSPT schedule for I_1 that is the same as the schedule of the first k' jobs in the LDWSPT schedule for I_0. Therefore, for the instance I_0:

$$\sum_{j=1}^{k'} w_j V_j^{I_0} \leq 0. \tag{4.2}$$

In I_0, since $\rho_{I_0}^{LDWSPT} > 2$:

$$\hat{V}(I_0) = \sum_{j=1}^{N} w_j V_j^{I_0} = \sum_{j=1}^{k'} w_j V_j^{I_0} + \sum_{j=k'+1}^{N} w_j V_j^{I_0} > 0,$$

where N denotes the number of jobs in I_0. As the relative sequence of jobs in a sorted list of ratios are the same in I_0 and $I_{0'}$, there exist an LDWSPT schedule for $I_{0'}$ that is the same as the LDWSPT schedule for I_0. In addition:

$$\frac{p_{k'}.w_{k'+1}}{p_{k'+1}.w_{k'}} < 1;$$

therefore:

$$0 < \sum_{j=1}^{k'} w_j V_j^{I_0} + \sum_{j=k'+1}^{N} w_j V_j^{I_0} \leq \frac{p_{k'}.w_{k'+1}}{p_{k'+1}.w_{k'}} \sum_{j=1}^{k'} w_j V_j^{I_0} + \sum_{j=k'+1}^{N} w_j V_j^{I_0} = \hat{V}(I_{0'})$$

The optimal offline schedules of I_0 and $I_{0'}$ are the same; thus, all jobs finish at their due dates in the optimal offline schedule of $I_{0'}$. As a result, from $\hat{V}(I_{0'}) > 0$, we conclude that $I_{0'}$ is a distinctive counterexample.

Note that as there exists an LDWSPT for $I_{0'}$ that is the same as the LDWSPT for I_0, jobs in the LDWSPT schedule for $I_{0'}$ are arranged in increasing ratios of p_j/w_j. □

Theorem 2. *$\rho_I^{LDWSPT} \leq 2$ for any instance, I, in which jobs are arranged in increasing ratios of p_j/w_j in the LDWSPT sequence.*

Proof. Assume there exists a distinctive counterexample, I_0, in which jobs are arranged in increasing ratios of p_j/w_j in the LDWSPT sequence.

If $\nexists\, k' = \min\{k : p_k/w_k < p_{k+1}/w_{k+1}\}$, the ratio of all jobs in I_0 are equal.

If $\exists\, k' = \min\{k : p_k/w_k < p_{k+1}/w_{k+1}\}$, the ratio of the first k' jobs that appear in the LDWSPT schedule for I_0 are equal. By Lemma 26, there exists another distinctive counterexample (the mutated version of I_0), in which jobs are arranged in increasing ratio of p_j/w_j in its LDWSPT sequence, and the ratio of at least the first $k'+1$ jobs that appear in its LDWSPT schedule are equal. By repeating this procedure, we can create a distinctive counterexample in which the ratio of all jobs are equal.

By Theorem 1, no counterexample exists in which the ratio of all jobs are equal. Therefore, the assumption of existence of a distinctive counterexample, in which jobs are arranged in increasing ratio of p_j/w_j in its LDWSPT sequence, is not true. Consequently, by Lemmas 19 and 20, no counterexample in which jobs are arranged in increasing ratio of p_j/w_j in its LDWSPT schedule exists.

□

General Case

Definition 9. *Define a "sub-block" to be a set of all jobs that appear consecutively and are arranged in increasing ratios of p_j/w_j in an LDWSPT sequence. Represent the i^{th} sub-block by B_i, the i'^{th} job in B_i by $J_{i,i'}$, the number of jobs in B_i by N_i, the total number of sub-blocks by l, and the start time of $J_{l-1,N_{l-1}}$ by r'.*

Note that in this Subsection, to represent jobs, we use two notations interchangeably: J_j for $j \in \{1, ..., N\}$ where N denotes the total number of jobs, and $J_{i,i'}$ for $i \in \{1, ..., l\}$ and $i' \in \{1, ..., N_i\}$. We transform the indices by function $[.,.]$. For example, if J_5 is the third job in the fourth sub-block of an LDWSPT schedule, $[4, 3] = 5$.

For illustration, Figure 4.3 captures the general formation of an LDWSPT sequence for an arbitrary instance I. In this figure, each bar represents a job. Jobs are positioned on the X-axis based on their position in the LDWSPT sequence for I, and the Y-axis represents the ratios.

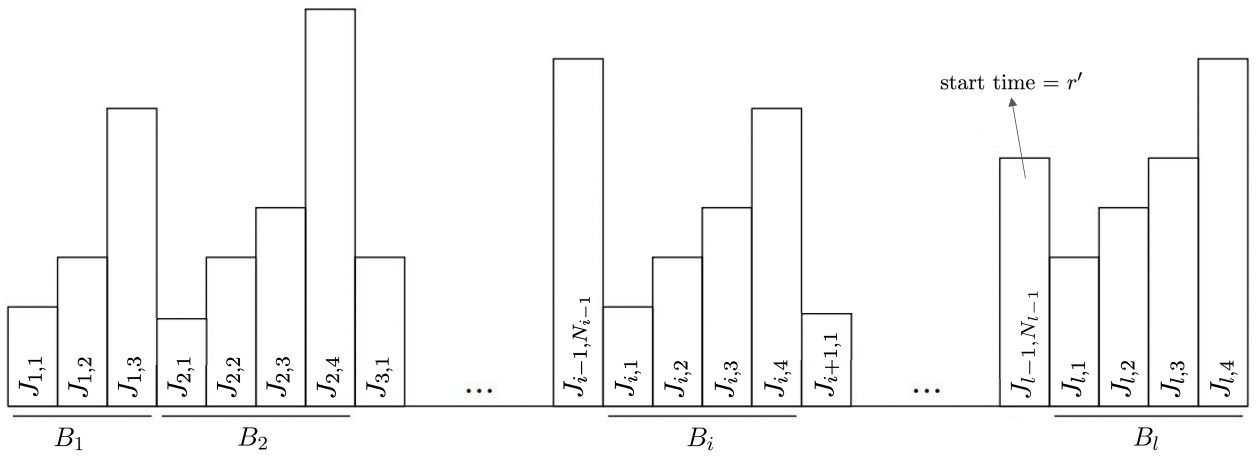


Figure 4.3: General formation of the LDWSPT sequence that contains l sub-blocks of jobs.

Note that when all jobs have equal ratios (Subsection 4.4.2), and when jobs are arranged in increasing ratios of p_j/w_j in the LDWSPT sequence (Subsection 4.4.2), only one sub-block of jobs exists, while in the general form, similar to Figure 4.3, there can be many sub-blocks.

Definition 10. *Define set A as the set of all jobs in B_l (last sub-block) that have ratios strictly less than $p_{[l-1,N_{l-1}]}/w_{[l-1,N_{l-1}]}$, which is the ratio of the last job in the second to last sub-block.*

Lemma 27. *If there exists a distinctive counterexample, I, we can create another distinctive counterexample, I', by setting the release date of all jobs in set A as well as all other jobs in B_l that have greater release date than r' to $r^{new} = r' + \xi$, where $\xi \to 0^+$.*

Proof. The release dates of all jobs in set A are greater than r'; otherwise, they would have appeared before $J_{l-1,N_{l-1}}$ in the LDWSPT schedule of I. Since after r' jobs are arranged increasingly based on their ratios, setting the release dates of jobs in set A as well as all other jobs of B_l that have greater release date than r' to r^{new} does not affect the LDWSPT schedule. Hence the LDWSPT schedule for I and for I' are the same. In addition, the optimal offline schedule of I and I' are the same since all jobs complete at their due dates. Thus, I' is also a distinctive counterexample, because:

$$\rho_{I'}^{LDWSPT} = \rho_{I}^{LDWSPT} > 2$$

□

Definition 11. *We define $F_j(k)$ to represent the set of jobs among the first k jobs of set A that appear before J_j (including itself) in an LDWSPT schedule. Similarly, we define $H_j(k)$ to represent the set of jobs among the first k jobs of set A that appear before J_j (including itself) in an optimal offline schedule. Lastly, we define $\tilde{P}_j(k)$ and $P'_j(k)$ as follows:*

$$\tilde{P}_j(k) = \sum_{i \in F_j(k)} p_i$$

and

$$P'_j(k) = \sum_{i \in H_j(k)} p_i.$$

Lemma 28. *For any distinctive counterexample, I_0, in which the release date of each job in B_l is at most $r' + \xi$ where $\xi \to 0^+$, there exist small enough $\delta_1 > 0$ and $\delta_2 > 0$ for which at least one of the following statements is true.*

Statement 1: *A distinctive counterexample, I_1, can be created by altering the processing times of the first k' jobs of set A as: $p_j^{new} = (1 - \delta_1)p_j$, and their due dates as: $d_j^{new} = d_j - \delta_1 P'_j(k')$;*

Statement 2: *A distinctive counterexample, I_2, can be created by altering the processing times of the first k' jobs of set A as: $p_j^{new} = (1 + \delta_2)p_j$, and their due dates as: $d_j^{new} = d_j + \delta_2 P'_j(k')$;*
where:

$$k' = \min\{\, \min\{k : \frac{p_{[l,k]}}{w_{[l,k]}} < \frac{p_{[l,k+1]}}{w_{[l,k+1]}}\}, |A|\}.$$

Proof. Define v_j^1 and v_j^1 as follows:

$$v_j^1 = V_j^{I_1} - V_j^{I_0}$$

$$v_j^2 = V_j^{I_2} - V_j^{I_0}$$

In Table 4.2, for each job j, the values of v_j^1 and v_j^2 based on whether J_j is tardy (T), is early (E) or is neither (-) in the LDWSPT schedule for I_0 is presented.

Table 4.2: Values of v_j^1 and v_j^2 for each job.

Job	T/E	v_j^1	v_j^2
j	T	$+(P_j'(k') - \tilde{P}_j(k'))\delta_1 + P_j'(k')\delta_1$	$-(P_j'(k') - \tilde{P}_j(k'))\delta_2 - P_j'(k')\delta_2$
	E	$-(P_j'(k') - \tilde{P}_j(k'))\delta_1 + P_j'(k')\delta_1$	$+(P_j'(k') - \tilde{P}_j(k'))\delta_2 - P_j'(k')\delta_2$
	-	$+\lvert P_j'(k') - \tilde{P}_j(k')\rvert\delta_1 + P_j'(k')\delta_1$	$+\lvert P_j'(k') - \tilde{P}_j(k')\rvert\delta_2 - P_j'(k')\delta_2$

In I_0, I_1, and I_2, all jobs finish at their due dates in the optimal offline schedule. Therefore, $\rho_{I_0}^{LDWSPT} > 2$, $\rho_{I_1}^{LDWSPT} > 2$, and $\rho_{I_2}^{LDWSPT} > 2$ are equivalent to $\hat{V}(I_0) > 0$, $\hat{V}(I_1) > 0$, and $\hat{V}(I_2) > 0$, respectively. Furthermore:

$$\hat{V}(I_1) = \hat{V}(I_0) + \sum_j w_j v_j^1$$

$$\hat{V}(I_2) = \hat{V}(I_0) + \sum_j w_j v_j^2$$

As demonstrated in Table 4.2, for each j, $v_j^1/\delta_1 \geq -v_j^2/\delta_2$. Therefore, if $\sum_j w_j v_j^2 < 0$, then $\sum_j w_j v_j^1 > 0$. Hence, since $\hat{V}(I_0) > 0$, at least one of $\hat{V}(I_1)$ or $\hat{V}(I_2)$ is greater than zero, and consequently, at least one of I_1 or I_2 is a distinctive counterexample. □

Lemma 29. *δ_1 and δ_2 in Lemma 28 can be any arbitrary positive numbers as long as the LDWSPT sequences for I_1 and I_2 remain the same as the LDWSPT sequence for I_0.*

Proof. For any $\delta_1 > 0$ and $\delta_2 > 0$, as long as the LDWSPT sequences of I_1 and I_2 remain the same as the LDWSPT sequence for I_0, $v_j^1 \geq -v_j^2$ remains valid, and thus, at least one of $\hat{V}(I_1)$ or $\hat{V}(I_2)$ is greater than zero. Therefore, for any $\delta_1 > 0$ and $\delta_2 > 0$, as long as the LDWSPT sequences for I_1 and I_2 remain the same as the LDWSPT sequence of I_0, at least one of the statements one and two is true.

□

Definition 12. *For any distinctive counterexample, I_0, in which the release date of each job in B_l is at most $r' + \xi$ where $\xi \to 0^+$, and $\exists k' = \min\{k : p_{[l,k]}/w_{[l,k]} < p_{[l,k+1]}/w_{[l,k+1]}\}$, the "adjusted version" of I_0 is created by altering the processing times and due dates of the first k' jobs of set A of I_0 as follows:*

$$p_{[l,j]}^{new} = p_{[l,k'+1]} \times w_{[l,j]}/w_{[l,k'+1]},$$

and

$$d^{new}_{[l,j]} = d_{[l,j]} + (\frac{p_{[l,k'+1]} \cdot w_{[l,j]}}{p_{[l,j]} \cdot w_{[l,k'+1]}} - 1) \times P'_{[l,j]}(k').$$

for all $j \in \{1, 2, ..., k'\}$. Note that if k' exists, by Definition 10, $k' \le |A|$.

Lemma 30. *For any distinctive counterexample, I_0, in which the release date of each job in B_l is at most $r' + \xi$ where $\xi \to 0^+$, and $\exists k' = \min\{k : p_{[l,k]}/w_{[l,k]} < p_{[l,k+1]}/w_{[l,k+1]}\}$, the adjusted version of I_0 is a distinctive counterexample.*

Proof. For any distinctive counterexample, I_0, in which the release date of each job in B_l is at most $r' + \xi$ where $\xi \to 0^+$, when setting the values of δ_1 and δ_2 in Lemma 28 to 1 and $\frac{p_{[l,k'+1]} \cdot w_{[l,k']}}{p_{[l,k']} \cdot w_{[l,k'+1]}} - 1$, respectively, by Lemma 29, at least one of the following statements is true:

Statement 1: A distinctive counterexample, I_1, can be created by altering the processing times of the first k' jobs of set A as: $p^{new}_j = 0$, and their due dates as: $d^{new}_j = d_j - P'_j(k')$;

Statement 2: The adjusted version of I_0 is a distinctive counterexample.

To prove the adjusted version of I_0 is a distinctive counterexample, in the following, we prove the first statement is wrong.

Assume the first statement is true and I_1 is a distinctive counterexample. Create a new instance, $I_{1'}$ by removing $J_{l,1}, J_{l,2}, ..., J_{l,k'}$ from I_1. Note that the start and end times of all other jobs in the LDWSPT schedules for I_1 and for $I_{1'}$ are the same. Therefore:

$$\hat{V}(I_{1'}) = \hat{V}(I_1) - \sum_{j \in \{1,2,...,k'\}} w_{[l,j]}(E_{[l,j]} + T_{[l,j]} - d_{[l,j]}) \tag{4.3}$$

In I_1, $r_{[l,j]} > r'$ for $j \in \{1, 2, ..., k'\}$. Therefore, the due dates of $J_{l,1}, J_{l,2}, ..., J_{l,k'}$ which are equal to their completion times in the optimal offline schedule of I_1 are grater than r'.

As the processing times of $J_{l,1}, J_{l,2}, ..., J_{l,k'}$ are zero, in the LDWSPT schedule for I_1, their start and end times are equal to the completion time of $J_{l-1,N_{l-1}}$, which is at most $2r'$. So if any of those jobs is tardy, its tardiness is at most r'. Therefore, for $j \in \{1, 2, ..., k'\}$, whether $J_{l,j}$ has earliness, tardiness or neither:

$$E_{[l,j]} + T_{[l,j]} \le d_{[l,j]}$$

and thus:

$$\sum_{j \in \{1,2,...,k'\}} w_{[l,j]}(E_{[l,j]} + T_{[l,j]} - d_{[l,j]}) \le 0. \tag{4.4}$$

From Equations 4.3 and 4.4, we conclude that $\hat{V}(I_{1'}) > 0$. As a result, $I_{1'}$ is a distinctive counterexample. However, this contradicts the fact that a distinctive counterexample has the minimum number of jobs among all counterexamples, as $I_{1'}$ has fewer jobs than I_1.

Therefore, the assumption made on the correctness of the first statement is not valid, and thus, we conclude that the adjusted version of I_0 is a distinctive counterexample. □

Theorem 3. *The LDWSPT policy is 2-competitive for the minimization of* $1|r_j, d_j, online| \sum_j w_j(E_j + T_j + d_j)$.

Proof. Assume a distinctive counterexample exists. Consider any distinctive counterexample that has the minimum number of sub-blocks among all counterexamples. By Lemma 27, there exists a distinctive counterexample, I_0, in which the release date of each job in B_l is at most $r' + \xi$ where $\xi \to 0^+$, and that has the minimum number of sub-blocks among all counterexamples. In I_0, the number of sub-blocks are greater than one, because otherwise, all jobs are arranged in increasing ratios of p_j/w_j in its LDWSPT schedule, which is not possible based on Theorem 2.

To prove no distinctive counterexample exists, we prove if I_0 exists, another distinctive counterexample with fewer number of sub-blocks than I_0 exists, which contradicts to the assumption that I_0 is a distinctive counterexample that has the minimum number of sub-blocks among all counterexamples. To do this, we first prove there exists a distinctive counterexample in which the ratio of all jobs in set A are equal.

For the instance I_0, if $\nexists k' = \min_{k\in\{1,2,\ldots,|A|-1\}}\{k : p_{[l,k]}/w_{[l,k]} < p_{[l,k+1]}/w_{[l,k+1]}\}$, the ratios of all jobs in set A are equal. Also, if $\exists k' = \min_{k\in\{1,2,\ldots,|A|-1\}}\{k : p_{[l,k]}/w_{[l,k]} < p_{[l,k+1]}/w_{[l,k+1]}\}$, the ratios of the first k' jobs of set A that appear in the LDWSPT schedule for I_0 are equal. In this case, by Lemma 30, there exists another distinctive counterexample (the adjusted version of I_0), in which the ratios of at least the first $k' + 1$ jobs of set A that appear in its LDWSPT schedule are equal. By repeating this procedure, we can create a distinctive counterexample in which the ratios of all jobs in set A are equal.

Consider a distinctive counterexample, I_1, in which the release date of each job in B_l is at most $r' + \xi$ where $\xi \to 0^+$, that has the minimum number of sub-blocks among all counterexamples, and the ratios of all its jobs in set A are equal.

If $|A| < N_l$ in I_1, by Lemma 30, there exists a distinctive counterexample (the adjusted version of I_1) in which the ratios of the first $|A|$ jobs in the last sub-block are equal to the ratio of $J_{l,|A|+1}$. As by Definition 10, the ratio of $J_{l,|A|+1}$ is larger than the ratio of $J_{l-1,N_{l-1}}$, the adjusted version of I_1 has one sub-block less than I_1.

If $|A| = N_l$ in I_1, by setting the values of δ_1 and δ_2 in Lemma 28 to 1 and $\frac{p_{[l-1,N_{l-1}]}\cdot w_{[l,|A|]}}{p_{[l,|A|]}\cdot w_{[l-1,N_{l-1}]}} - 1$, respectively, with a similar proof to Lemma 30, we can show that there exists a distinctive counterexample that has one sub-block less than I_1.

As a conclusion, if I_0 exists, there exists a distinctive counterexample that has one sub-block less than I_0. This contradicts the fact that I_0 has the smallest number of sub-blocks among counterexamples. Therefore, we conclude that no distinctive counterexample, and consequently, by Lemmas 19 and 20, no counterexample exists. □

Corollary 1. *The LDWSPT policy is an optimal policy for the minimization of* $1|r_j, d_j,$ *online*$|\sum_j (E_j + T_j + d_j)$ *and* $1|r_j, d_j, online|\sum_j w_j(E_j + T_j + d_j)$.

Proof. From Theorem 3, for both problems, the LDWSPT policy is 2-competitive. In addition, from Lemma 17, two is a lower bound for both problems. □

Corollary 2. *The DWSPT policy is an optimal policy for the minimization of* $1|r_j, d_j,$ *online*$|\sum_j (E_j + T_j + d_j)$ *and* $1|r_j, d_j,$ *online*$|\sum_j w_j(E_j + T_j + d_j)$.

Proof. Since the LDWSPT policy is the general form of the DWSPT policy, all schedules produced by the DWSPT policy can be reproduced by LDWSPT policy. Hence, the DWSPT policy is also 2-competitive and optimal for both problems. □

4.5 Total Weighted (Modified) Unequal Earliness and Tardiness Cost

Up to this point, we have assumed that earliness and tardiness penalties are equal. In this section, we relax that assumption by extending our analysis to the problem of minimizing $1|r_j, d_j, online|\sum_j w_j(\zeta_E E_j + \zeta_T T_j + \zeta_d d_j)$, where ζ_E is the per unit time cost of earliness, ζ_T is the per unit time cost of tardiness, and ζ_d is the per unit time due date cost. In what follows, we refer to this problem as the "extended problem".

In Subsection 4.5.1, we prove that $\max(2, 1 + \zeta_T/\zeta_d)$ is a lower bound for the extended problem and in Subsection 4.5.2, we show that the LDWSPT policy is $\max(2, 1 + \zeta_E/\zeta_d, 1 + \zeta_T/\zeta_d)$-competitive for the extended problem. Thus, when $\zeta_E \leq \max\{\zeta_T, \zeta_d\}$, the LDWSPT policy is optimal for the extended problem.

Note that for the extended problem, when $\zeta_d < \zeta_T$, the lower bound is a function of ζ_d and ζ_T. Also, when $\zeta_d < \zeta_E$ or $\zeta_d < \zeta_E$, the upper bound is a function of ζ_d and $\max\{\zeta_E, \zeta_T\}$. This is in contrast to the previous results in this chapter, in which both lower and upper bounds are independent of the parameters of the problem.

4.5.1 Lower Bound

Lemma 31. $\max(2, 1+\zeta_T/\zeta_d)$ *is a lower bound on the minimization of* $1|r_j, d_j, online|\sum_j w_j$ $(\zeta_E\ E_j + \zeta_T T_j + \zeta_d d_j)$.

Proof. We define Instances I_1, I_2, I_3 and I_4 as follows:
Instance I_1: jobs = $\{J_1\}$
J_1: $r_1 = 0, p_1 = 1, w_1 = 1, d_1 = 0$.
Instance I_2: jobs = $\{J_1, J_2, ..., J_N\}$
J_1: $r_1 = 0, p_1 = 1, w_1 = 1, d_1 = 0$;
J_j for $j \in \{2, 3, ..., N\}$: $r_j = a + \epsilon$, $p_j = \epsilon$, $w_j = 1, d_j = 0$;
where $\epsilon = 1/N^2$ and a is a non-negative scalar.
Instance I_3: jobs = $\{J_1\}$
J_1: $r_1 = 0, p_1 = 1, w_1 = 1, d_1 = 1$.

Instance I_4: jobs $= \{J_1, J_2, ..., J_N\}$
J_1: $r_1 = 0, p_1 = 1, w_1 = 1, d_1 = 1$;
J_j for $j \in \{2, 3, ..., N\}$: $r_j = a + \epsilon$, $p_j = \epsilon$, $w_j = 1, d_j = a$;
where $\epsilon = 1/N^2$ and a is a non-negative scalar.

In the online setting, for both instances I_1 and I_2, a single job with identical characteristics arrives at time zero, so it is impossible to distinguish between the instances at the start of the horizon. Similarly, it is impossible to distinguish between the instances I_3 and I_4 at the start of the horizon. In the following, we separately prove two and $1 + \zeta_T/\zeta_d$ are lower bounds.

The competitive ratio of any possible policy is at least two, because for the instance I_1, when a is the starting time of process on J_1, the objective function value of the policy is $1+a$ times its optimal offline objective function value. Also, for the instance I_2, when a is the starting time of the process on J_1, the objective function value of the policy, when $N \to \infty$, is $1 + 1/a$ times its optimal offline objective function value. Therefore, the competitive ratio of any policy is at least $\max(1 + a, 1 + 1/a)$, which is at least equal to two.

Similarly, the competitive ratio of any possible policy is at least $1 + \zeta_T/\zeta_d$, because, for the instance I_3, when a is the starting time of the process on J_1, the objective function value of the policy is $1 + \frac{a.\zeta_T}{\zeta_d}$ times its optimal offline objective function value. Also, for the instance I_4, when a is the starting time of the process on J_1, the objective function value of the policy, when $N \to \infty$, is $1 + \frac{\zeta_T}{a.\zeta_d}$ times its optimal offline objective function value. Therefore, the competitive ratio of any policy is at least $\max(1 + \frac{a.\zeta_T}{\zeta_d}, 1 + \frac{\zeta_T}{a.\zeta_d})$, which is at least equal to $1 + \zeta_T/\zeta_d$.

□

4.5.2 Competitiveness of the LDWSPT Policy

In this subsection, "counterexample" refers to an instance, I, with

$$\rho_I^{LDWSPT} > \alpha,$$

where $\alpha \geq 2$. (Note that this is different from the "counterexample" of Subsection 4.4.2.) Additionally, for any instance, I, in which jobs have equal ratios, and its split version, I', the metric $\hat{\rho}_{I'}(\phi)$, for any feasible sub-job schedule ϕ of the instance I', is defined as:

$$\hat{\rho}_{I'}(\phi) = 1 + \frac{\sum_j \sum_{j'=1}^{n_j} w'\zeta_T \max\{z'_{jj'} - c_{jj'}, 0\}}{\sum_j \zeta_d(\sum_{j'=1}^{n_j} w'c_{jj'} + A(p_j))} + \frac{\sum_j \sum_{j'=1}^{n_j} w'\zeta_E \max\{c_{jj'} - z'_{jj'}, 0\}}{\sum_j \zeta_d(\sum_{j'=1}^{n_j} w'c_{jj'} + A(p_j))}.$$

For any instance, I, V_j^I is defined as:

$$V_j^I = \zeta_E E_j + \zeta_T T_j - (\alpha - 1)\zeta_d d_j,$$

and for any instance, I, when all jobs (similar to the distinctive counterexamples) complete at their due dates in the optimal offline schedule, $\rho_I^{LDWSPT} > \alpha$ is equivalent to $\hat{V}(I) > 0$.

Lemma 32. *When $\zeta_T \geq \zeta_d$, the LDWSPT policy is α-competitive for the minimization of $1|r_j, d_j, online|\sum_j w_j(\zeta_E E_j + \zeta_T T_j + \zeta_d d_j)$, where $\alpha = \max\{1 + \zeta_E/\zeta_d, 1 + \zeta_T/\zeta_d\}$.*

Proof. Below, we present the conditions under which all proofs that are presented in Subsection 4.4.2 are valid for the extended problem. These results follow in a straightforward way from the proofs already presented in this chapter.

For any ζ_E, ζ_T, and ζ_d, Lemmas 18, 21, 22, 24, 27, 28, and 29 are valid for the extended problem. Note that the proof of the validity of Lemma 28 follows from the the proof of Lemma 28 in Subsection 4.4.2, substituting Table 4.3 for Table 4.2. In Table 4.3, $\tilde{\zeta}_j$ is ζ_T when $P'_j(k') \geq \tilde{P}_j(k')$, and is ζ_E when $P'_j(k') < \tilde{P}_j(k')$.

Table 4.3: Values of v_j^1 and v_j^2 for each job in the extended problem.

Job	T/E	v_j^1	v_j^2
j	T	$+\zeta_T(P'_j(k') - \tilde{P}_j(k'))\delta_1 + (\alpha-1)\zeta_d P'_j(k')\delta_1$	$-\zeta_T(P'_j(k') - \tilde{P}_j(k'))\delta_2 - (\alpha-1)\zeta_d P'_j(k')\delta_2$
	E	$-\zeta_E(P'_j(k') - \tilde{P}_j(k'))\delta_1 + (\alpha-1)\zeta_d P'_j(k')\delta_1$	$+\zeta_E(P'_j(k') - \tilde{P}_j(k'))\delta_2 - (\alpha-1)\zeta_d P'_j(k')\delta_2$
	-	$+\tilde{\zeta}\,\lvert P'_j(k') - \tilde{P}_j(k')\rvert\,\delta_1 + (\alpha-1)\zeta_d P'_j(k')\delta_1$	$+\zeta_E\zeta_T/\tilde{\zeta}\,\lvert P'_j(k') - \tilde{P}_j(k')\rvert\,\delta_2 - (\alpha-1)\zeta_d P'_j(k')\delta_2$

Lemmas 19, 20, 23, 25, 26, and 30 are not valid for all α, ζ_E, ζ_T, and ζ_d. However, for each of these Lemmas, if values of α, ζ_E, ζ_T, and ζ_d meet the following conditions, they are valid.

Lemma 19 is valid when $\zeta_T \geq \zeta_d$, and Lemma 20 is valid when $\alpha \geq 1 + \zeta_E/\zeta_d$. When $\alpha \geq 1 + \zeta_E/\zeta_d$, similar to Lemma 23, it can be shown that $\hat{\rho}_{I'}(\varphi) > \alpha$ if $\hat{\rho}_{I'}(\phi) > \alpha$, and when $\alpha \geq 1 + \frac{\zeta_E + \zeta_T}{2\zeta_d}$, similar to Lemma 25, it can be shown that $\hat{\rho}_{I'}(\phi) \leq \alpha$. Lemma 26 is valid when $\zeta_T \geq \zeta_d$, $\alpha \geq 1 + \zeta_E/\zeta_d$ and $\alpha \geq 1 + \frac{\zeta_E + \zeta_T}{2\zeta_d}$. Lastly, when $\alpha \geq 1 + \zeta_E/\zeta_d$ and $\alpha \geq 1 + \zeta_T/\zeta_d$, Lemma 30 is valid.

In sum, when $\zeta_T \geq \zeta_d$, $\alpha \geq 1 + \zeta_E/\zeta_d$, and $\alpha \geq 1 + \zeta_T/\zeta_d$, similar to Theorems 1, 2, and 3, it can be shown that the LDWSPT policy is α-competitive for the extended problem. In other words, when $\zeta_T \geq \zeta_d$, the LDWSPT policy is $\max\{1 + \zeta_E/\zeta_d, 1 + \zeta_T/\zeta_d\}$-competitive for the extended problem.

□

Theorem 4. *The LDWSPT policy is $\max\{2, 1 + \zeta_E/\zeta_d,\ 1 + \zeta_T/\zeta_d\}$-competitive for the minimization of $1|r_j, d_j,\ online|\sum_j w_j(\zeta_E E_j + \zeta_T T_j + \zeta_d d_j)$.*

Proof. Consider any instance, I, in which $\zeta_d > \zeta_T$. For the instance I:

$$\sum_j w_j(\zeta_E E_j + \zeta_T T_j + \zeta_d d_j) = \sum_j w_j((\frac{\zeta_E \zeta_T}{\zeta_d})E_j + \zeta_T T_j + \zeta_T d_j) + \sum_j w_j((\zeta_E - \frac{\zeta_E\zeta_T}{\zeta_d})E_j + (\zeta_d - \zeta_T)d_j) \quad (4.5)$$

Create a new instance, I_1, by altering I and setting the per unit time cost of earliness and due date to $\zeta_E^1 = \zeta_E\zeta_T/\zeta_d$ and $\zeta_d^1 = \zeta_T$, respectively. Then, the objective function value for I_1 is:

$$\sum_j w_j(\zeta_E^1 E_j + \zeta_T T_j + \zeta_d^1 d_j) = \sum_j w_j((\frac{\zeta_E\zeta_T}{\zeta_d})E_j + \zeta_T T_j + \zeta_T d_j).$$

By Lemma 32:

$$\rho_{I_1}^{LDWSPT} \leq \max\{1 + \zeta_E^1/\zeta_d^1, 1 + \zeta_T/\zeta_d^1\} = \max\{2, 1 + \zeta_E/\zeta_d\}.$$

Create a new instance, I_2, by altering I and setting the per unit time cost of earliness, tardiness, and due date to $\zeta_E^2 = (\zeta_E - \zeta_E\zeta_T/\zeta_d)$, $\zeta_T^2 = 0$ and $\zeta_d^2 = \zeta_d - \zeta_T$, respectively. Then, the objective function value for I_2 is:

$$\sum_j w_j(\zeta_E^2 E_j + \zeta_T^2 T_j + \zeta_d^2 d_j) = \sum_j w_j((\zeta_E - \frac{\zeta_E\zeta_T}{\zeta_d})E_j + (\zeta_d - \zeta_T)d_j)$$

The optimal objective function value of the optimal offline schedule for I_2 is at least $\sum_j w_j(\zeta_d - \zeta_T)d_j$. Therefore,

$$\rho_{I_2}^{LDWSPT} \leq \frac{\sum_j w_j((\zeta_E - \frac{\zeta_E\zeta_T}{\zeta_d})E_j + (\zeta_d - \zeta_T)d_j)}{\sum_j w_j(\zeta_d - \zeta_T)d_j} \leq 1 + \zeta_E/\zeta_d.$$

As all jobs are the same in the instances I, I_1 and I_2, their LDWSPT schedules are the same. By Equation 4.5, the objective function value of I is equal to the sum of the objective function values of I_1 and I_2. Thus:

$$\rho_I^{LDWSPT} \leq \max\{\rho_{I_1}^{LDWSPT}, \rho_{I_2}^{LDWSPT}\} \leq \max\{2, 1 + \zeta_E/\zeta_d\}. \quad (4.6)$$

From Equation 4.6, we conclude that the LDWSPT policy is $\max\{2, 1+\zeta_E/\zeta_d\}$-competitive when $\zeta_T < \zeta_d$. Also by Lemma 32, the LDWSPT policy is $\max\{1 + \zeta_E/\zeta_d, 1 + \zeta_T/\zeta_d\}$-competitive when $\zeta_T \geq \zeta_d$. Hence, for any ζ_E, ζ_T and ζ_d, the LDWSPT policy is $\max\{2, 1 + \zeta_E/\zeta_d, 1 + \zeta_T/\zeta_d\}$-competitive for the extended problem.

□

Corollary 3. *When* $\max\{\zeta_T, \zeta_d\} \geq \zeta_E$, *the LDWSPT policy is an optimal policy for the minimization of* $1|r_j, d_j, online|\sum_j w_j(\zeta_E E_j + \zeta_T T_j + \zeta_d d_j)$, *and its competitive ratio is* $\max\{2, 1 + \zeta_T/\zeta_d\}$.

Proof. By Lemma 4.5.1, $\max\{2, 1 + \zeta_T/\zeta_d\}$ is a lower bound on the competitive ratio, and also by Theorem 4, the competitive ratio of LDWSPT policy is $\max\{2, 1 + \zeta_T/\zeta_d\}$ when $\max\{\zeta_T, \zeta_d\} \geq \zeta_E$. □

Corollary 4. *For the minimization of* $1|r_j, d_j, online| \sum_j (T_j + d_j)$ *and* $1|r_j, d_j, online|\sum_j w_j(T_j + d_j)$, *the LDWSPT policy is 2-competitive and optimal.*

Proof. By Theorem 4, for both problems, the LDWSPT policy is 2-competitive. Also, by a straightforward algebraic manipulation of the proof of Lemma 17, two is a lower bound on the competitive ratio of both problems. Thus, the LDWSPT policy is optimal for both problems and its competitive ratio is two. □

4.6 Future Research

When $\zeta_E > \max\{\zeta_T, \zeta_d\}$, the LDWSPT policy is not optimal for the minimization of $1|r_j, d_j, online|\sum_j w_j\ (\zeta_E E_j + \zeta_T T_j + \zeta_d d_j)$. We conjecture that the following online policy has a competitive ratio of $\max\{2, 1 + \zeta_T/\zeta_d\}$ for this problem. We leave the proof of this conjecture for future work.

Algorithm 4 Double Delayed Weighted Shortest Processing Time (DDWSPT) Policy

```
Create two empty lists, $L_1$ and $L_2$.
Each time the machine becomes available:
 Step 1:
  INSERT all new jobs (jobs that have been released but have not yet been added to $L_1$)
  into $L_1$ such that the list is arranged in order of increasing $d_j - p_j$. Break ties arbitrarily.
 Step 2:
  Define $k' = \max\{k : d_{L_1[k]} - p_{L_1[k]} \leq t\}$, where $t$ is the time since the start of the time
  horizon and $L_1[k]$ is the $k^{th}$ job in $L_1$.
  REMOVE the first $k'$ jobs of $L_1$ and INSERT them into $L_2$ such that $L_2$ is arranged in
  order of increasing $p_j/w_j$. Break ties arbitrarily.
 Step 3:
  If $L_2$ is empty, wait until $t = d_{L_1[1]} - p_{L_1[1]}$ or until a new job arrives, whichever happens
  first, and then go to *Step 1*.
  If $L_2$ is not empty, SELECT the first job, $J_j$, in $L_2$.
 Step 4:
  If $p_j \leq t$, REMOVE $J_j$ from $L_2$ and start processing it.
  Otherwise, wait until $t = p_j$ or until $t = d_{L_1[1]} - p_{L_1[1]}$ or until a new job arrives, whichever
  happens first, and then, go to *step 1*.
```

Chapter 5

Workload Leveling Based on Work Space Zoning for Takt Planning[1]

5.1 Introduction

This chapter focuses on a location-based method that is used in takt planning, specifically the Work Density Method (WDM) that is outlined next. The contribution to knowledge of this chapter is the problem formulation and presentation of an optimization algorithm to level so-called work density across trades engaged in a sequence of process steps necessary to realize a phase of construction work.

The following Section 5.2 introduces the concept of work density and describes a takt planning method based on it. It also formulates the workload leveling with zoning (WoLZo) problem that is the focus of this chapter. Section 5.3 then highlights the literature on existing mathematical optimization algorithms that informed the method we developed to solve the WoLZo problem. Section 5.4 presents the problem formulation and Section 5.5 describes the algorithm and solution method. The subsequent Section 5.6 illustrates the algorithm's application to example data and shows results. Finally, Section 5.7 expands on the discussion of the results and offers insights gained from running the model with various inputs.

5.2 Work Density Method for Takt Planning

5.2.1 Definition of Work Density

To articulate the production planning method presented in this chapter, the following terms will be used. The **master schedule** of a construction project may comprise one or multiple phases. A **phase** is defined so as to include the work of all trades who will work on site more-or-less concurrently in a certain **work space**. A phase's start and end are delimited by

[1]This work is published in [38] and [39].

milestones in the master schedule. Phases comprise one or several processes. Each **process** is made up of steps. Each **step** is performed by one trade so that the process sequence of steps forms a Parade of Trades. Trades may be responsible for performing one or multiple steps in a process. However, to keep explanations simple, we here assume that a phase comprises only a single process and that each trade performs only one step in a process.

How much time a trade needs to perform the work in their step depends on the project's specifics and management decision making. For example: *What scope is included in any one step?*, *Where is that work located?*, and *What means-and-methods will be used?* Such questions are addressed in the course of work structuring [79], but that is a topic beyond the scope of this chapter. To describe this duration, Tommelein ([23], [75], [76]) coined the term **work density**, expressed as a unit of time per unit of area. She defined it as follows: Given a certain work area, work density describes how much time a trade will require to do their work in that area, based on:

1. the product's design, i.e., what is in the construction project drawings and specifications,
2. the scope of the trade's work,
3. the specific task in their schedule (depending on work already in place and work that will follow later in the same or another process),
4. the means and methods the trade will use (e.g., when prefabricating off-site, the work density on-site is expected to decrease),
5. while accounting for crew capabilities and size.

Accordingly, work density serves as a metric common to all trades, that characterizes at what speed their work will progress through space. It is an expression pertaining to workload. It is also related to cycle time but augments that concept in that work density explicitly references where work takes place, with space modeled in 2 dimensions. Knowledge of work densities informs the task of production planning, for example using takt.

5.2.2 Takt Planning with the Work Density Method

Takt is the German word for beat. It is the "unit of time within which a product must be produced (supply rate) in order to match the rate at which that product is needed (demand rate)" (e.g., [34], [77]). Takt is a parameter used in designing a production system.

The goal of takt planning is to produce a production plan for a certain scope of work and with a level work flow. That plan is then used to steer and control construction work on- and off-site. Takt planning can have multiple objectives. An objective essential to the workload leveling problem as formulated here, is that trade stacking is avoided, i.e., no two trades can be working in the same location at any one time.

Takt is widely used to pace manufacturing assembly-line processes. In contrast with manufacturing where the assembly moves, in construction the product is stationary and the process requires that trades move from one work location to the next. Thus, takt is defined as the maximum amount of time allowed for each trade's step in a process to finish their work in their designated work location. Emulating an assembly line, each trade gets the same amount of time to do the work for their step in the process, so the step that will need the largest amount of time in any of their given work locations constrains the takt of the line.

Aiming to keep the discussion simple, we refer to this largest amount of time as the takt. However, when there is process time variability in the system, determination of the takt must include some buffer time being added to this largest amount of time required. While the elimination of process time variability and subsequent buffering for any that remains are of utmost importance to the potential success of a takt plan's implementation, discussion of how to cope with process time variability is beyond the scope of this chapter.

Assuming the takt has been determined for a given process, if the work area associated with a phase is undivided, the phase comprises only one process, and trades are not stacked, then the duration of the phase equals the number of steps in the process multiplied by the takt.

To speed up the process, the production planning team may divide the work area into **zones** (work areas that are mutually exclusive and collectively exhaustive), and develop a takt plan while continuing to ensure that each trade has their own zone to work in at any time, never stacking them. In general—although not always—as zones get smaller, steps in each zone will then have less scope for each trade to complete, and thus require less time for that scope to be completed prior to the trade moving to the next zone. Less time needed means that the takt can possibly be reduced, so the achievable takt will vary based on how zones are defined. We therefore represent takt by $T(Z)$ to emphasize it is a function of the number of zones Z, as shown in Figure 5.1. Zones may differ in size and shape from one another, but any work in them must be completed within the takt of the Parade as it progresses.

Hypothetically, then, the duration of the process corresponds to the sum of (1) the number of trade steps S in the process (i.e., the steps in the Parade of Trades) and (2) the number of zones Z minus 1, multiplied by the takt $T(Z)$, as calculated with Equation 5.1. The increase in zones allows for greater concurrency of process steps. As mentioned, as a work space gets divided more (i.e., Z increases), generally one might expect zones to become smaller and the corresponding takt $T(Z)$ will therefore decrease.

$$D = (S + Z - 1) \times T(Z) \tag{5.1}$$

Extending this equation for the case where the process repeats over multiple floors F, the duration is calculated with Equation 5.2.

$$D = (S + F \times Z - 1) \times T(Z) \tag{5.2}$$

Figures 5.1 and 5.2 illustrate this hypothetical reduction in phase completion time when the number of zones increases ($S = 4$ and $Z = 1$, 2, or 3 in Figure 5.1), and when the process gets repeated over multiple floors ($S = 4$, $Z = 4$, and $F = 3$ in Figure 5.2).

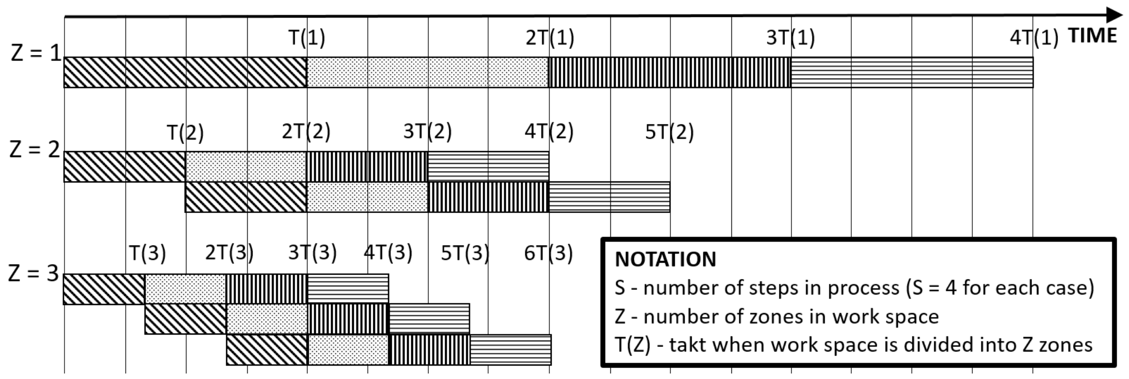

TIME
Z = 1
T(1)
2T(1)
3T(1)
4T(1)
Z = 2
T(2)
2T(2)
3T(2)
4T(2)
5T(2)
Z = 3
T(3)
2T(3)
3T(3)
4T(3)
5T(3)
6T(3)
NOTATION
S - number of steps in process (S = 4 for each case)
Z - number of zones in work space
T(Z) - takt when work space is divided into Z zones

Figure 5.1: Process with 4 steps, scheduled when work space is divided respectively in 1, 2, or 3 zones

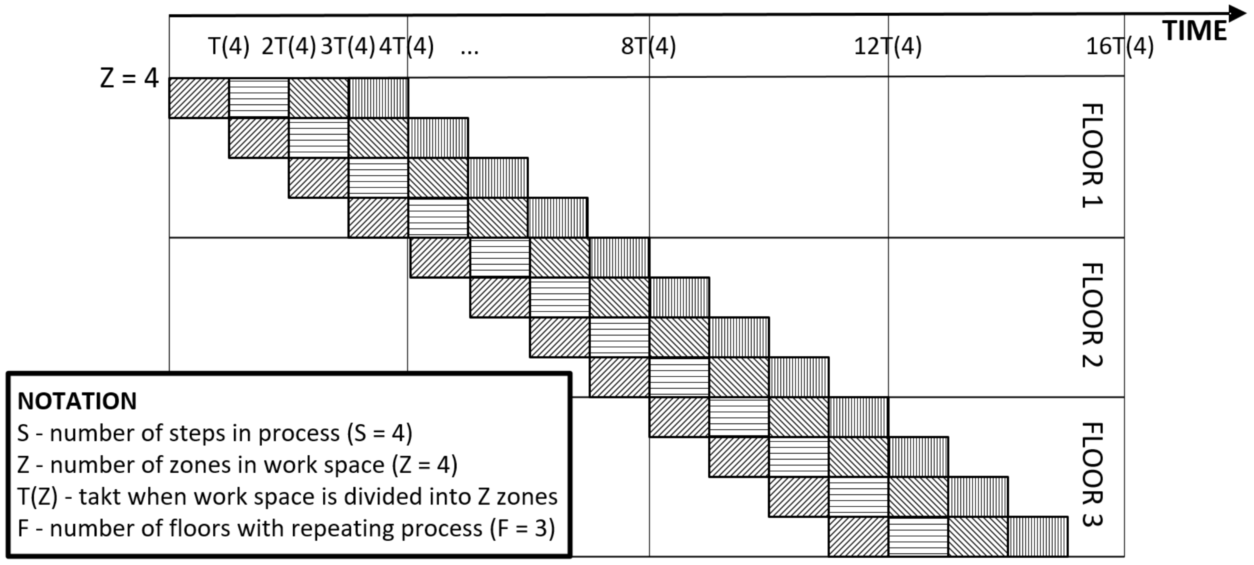
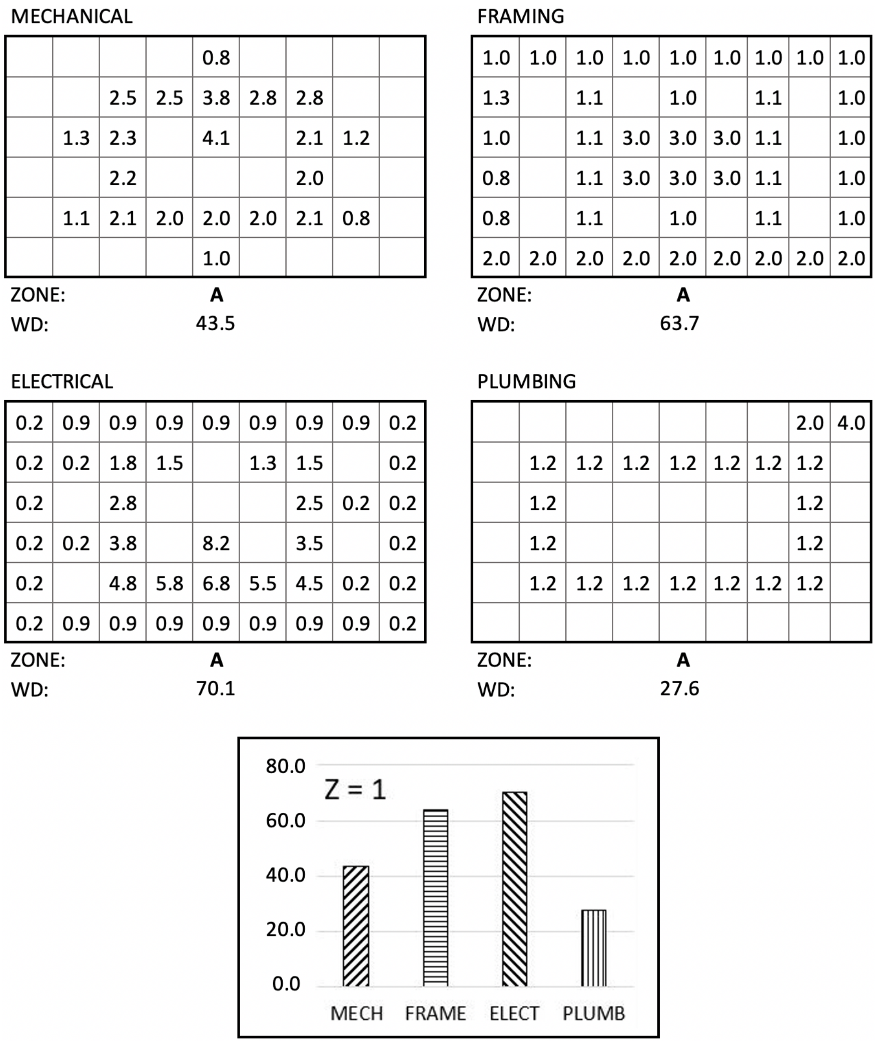

TIME
T(4) 2T(4) 3T(4) 4T(4) ...
8T(4)
12T(4)
16T(4)
Z = 4
FLOOR 1
FLOOR 2
FLOOR 3
NOTATION
S - number of steps in process (S = 4)
Z - number of zones in work space (Z = 4)
T(Z) - takt when work space is divided into Z zones
F - number of floors with repeating process (F = 3)

Figure 5.2: Process with 4 steps that repeats over 3 floors, scheduled when work space is divided in 4 zones

However, these calculations may not work in reality for one or multiple reasons, e.g., work density is unevenly distributed over the work space, and workloads are not divisible across multiple zones. In any case, the optimization problem is to find a zoning that will minimize the takt.

5.2.3 Workload Leveling and Zoning (WoLZo) Problem

To illustrate the Workload Leveling and Zoning (WoLZo) problem to be solved, Figure 5.3 offers a toy problem example. It was inspired by the overhead construction work done on the pilot project described by Dunnebier et al. [23]. The process is characterized by means of work density maps of four different trades: (1) Mechanical, (2) Framing, (3) Electrical, and (4) Plumbing. Each work density map describes work the trade will perform in a certain work space (e.g., overhead work in a floor).


MECHANICAL

				0.8				
		2.5	2.5	3.8	2.8	2.8		
	1.3	2.3		4.1		2.1	1.2	
		2.2				2.0		
	1.1	2.1	2.0	2.0	2.0	2.1	0.8	
				1.0				

ZONE: A
WD: 43.5

FRAMING

1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.3		1.1		1.0		1.1		1.0
1.0		1.1	3.0	3.0	3.0	1.1		1.0
0.8		1.1	3.0	3.0	3.0	1.1		1.0
0.8		1.1		1.0		1.1		1.0
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0

ZONE: A
WD: 63.7

ELECTRICAL

0.2	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.2
0.2	0.2	1.8	1.5		1.3	1.5		0.2
0.2		2.8				2.5	0.2	0.2
0.2	0.2	3.8		8.2		3.5		0.2
0.2		4.8	5.8	6.8	5.5	4.5	0.2	0.2
0.2	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.2

ZONE: A
WD: 70.1

PLUMBING

							2.0	4.0
	1.2	1.2	1.2	1.2	1.2	1.2	1.2	
	1.2						1.2	
	1.2						1.2	
	1.2	1.2	1.2	1.2	1.2	1.2	1.2	

ZONE: A
WD: 27.6

Figure 5.3: Work density for four trades and one zone

This example is realistic in that it illustrates a case where each trade has some grid cells with a higher work density than others (e.g., a real-world example may be electrical work to be done in a lobby vs. in an electrical room). In addition, different trades have different work densities in certain grid cells (e.g., a real-world example may be an operating room, where the work density of the mechanical crew will likely differ from the work density of the electrical crew).

These work density maps may be used in takt planning according to the following steps of the Work Density Method. Assume that all overhead work is to be done by trades following each other in a Parade, with only one trade at a time working in that work space. The time needed by the Mechanical trade is computed by adding the work densities shown in each grid cell of their work density map. This adds up to a work density [WD] of 43.5 time units/floor. Likewise, the aggregate work densities for Framing, Electrical, and Plumbing are respectively 63.7, 70.1, and 27.6 time units/floor. When compared with one another (see histogram at the bottom of Figure 5.3), it is clear that the Electrical trade needs the greatest amount of time to complete their work in the work space. Assuming that this time defines the takt (see earlier mention of process time variability), the process will take 4 * 70.1 time units/floor = 280.4 time units to complete all work in the work space (per Equation 5.1 with $S = 4$, $Z = 1$, and $T(Z) = 70.1$). This is a long time and some trades will have a lot of idle time because of work density variation.

We premise is that, through data collection with consideration given to what determines work density, and judicious design of the zones, this peak in work density when considering all zones and all trades can be reduced. That is, the workload can be leveled and, as this peak informs the determination of the takt, the resulting takted system will take less time to complete than would be the case without workload leveling.

Figure 5.4 illustrates this leveling. Whereas Figure 5.3 showed the entire work area as a single zone (Zone A), Figure 5.4 shows the work area divided into three zones (zones A, B, and C).

The four rectangles form matrices that have grid cells with work densities of each of the four trades in the process. Superimposing a zoning pattern over these work density matrices, one can compute the amount of time each trade will take to complete their task by zone. A histogram (bottom of Figure 5.4) shows these workloads by zone for each trade in the process.

With three zones now, the time needed by each trade is computed zone by zone. For example, the work densities shown in each grid cell of the Mechanical work density map add up to 11.5 time units/zone A, 21.0 time units/zone B, and 11.0 time units/zone C. Likewise, the aggregate work densities for Framing, Electrical, and Plumbing are respectively 17.3, 18.4, and 7.2 time units/zone A. Similar computations yield time units/zone B and time units/zone C. Figure 5.4 illustrates that the Electrical work needing 34.5 time units in zone B defines the peak in work density when considering all zones and all trades. Assuming this time defines the takt (see earlier mention of process time variability), the process will take (4 + 3 - 1) * 34.5 time units = 207.0 time units to complete all work in the work space (per Equation 5.1 with $S = 4$, $Z = 3$, and $T(Z) = 34.5$). The process takes less time than was needed in the case of a single zone because, with three zones, trades can work concurrently in different zones.

MECHANICAL

				0.8				
		2.5	2.5	3.8	2.8	2.8		
	1.3	2.3		4.1		2.1	1.2	
		2.2				2.0		
	1.1	2.1	2.0	2.0	2.0	2.1	0.8	
				1.0				

ZONE: A B C
WD: 11.5 21.0 11.0

FRAMING

1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1.3		1.1		1.0		1.1		1.0
1.0		1.1	3.0	3.0	3.0	1.1		1.0
0.8		1.1	3.0	3.0	3.0	1.1		1.0
0.8		1.1		1.0		1.1		1.0
2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0

ZONE: A B C
WD: 17.3 29.0 17.4

ELECTRICAL

0.2	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.2
0.2	0.2	1.8	1.5		1.3	1.5		0.2
0.2		2.8				2.5	0.2	0.2
0.2	0.2	3.8		8.2		3.5		0.2
0.2		4.8	5.8	6.8	5.5	4.5	0.2	0.2
0.2	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.2

ZONE: A B C
WD: 18.4 34.5 17.2

PLUMBING

							2.0	4.0
	1.2	1.2	1.2	1.2	1.2	1.2	1.2	
	1.2						1.2	
	1.2						1.2	
	1.2	1.2	1.2	1.2	1.2	1.2	1.2	

ZONE: A B C
WD: 7.2 7.2 13.2

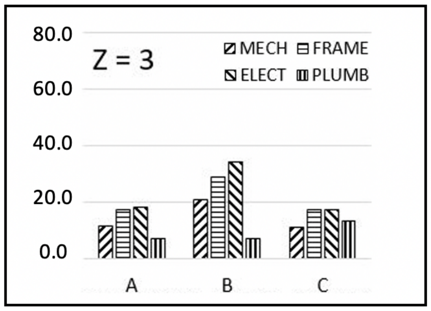


Figure 5.4: Work density for four trades and work space divided into three zones

When for a selected zoning the corresponding histogram is deemed sufficiently level, the planning team sets the takt, namely the time within which each step in the process must be completed (see earlier mention of process time variability). Using that takt and zoning, the planning team can finalize the takt plan by deciding on the order in which trades will traverse the zones. The planning team must ensure that the takt plan meets the master schedule and possibly other requirements, or otherwise adjust the plan until it does (e.g., modify takt, change the number of zones to allow for increased concurrency of steps, etc.).

The Work Density Method is used to even out- and takt the flow of work. It will take iteration to change work densities, rezone, and then redraw the histogram to find one that is deemed better, with no guarantee that a better one will be found when this is done manually. A 100% level workflow would be best, but in general that will be just about impossible to achieve.

The so-called WoLZo algorithm introduced later in this chapter addresses the WoLZo problem. The WoLZo problem involves leveling work density across trades engaged in a sequence of steps necessary to realize a process or phase of construction work. The problem formulation assumes that work density maps and the number of zones in which to divide the work space are given. In other words, the problem is to determine the zoning that minimizes the work density peak (the maximum work density) across trades and across zones. A lower peak corresponds to a lower takt, and thus a shorter duration of the process or phase. As soon as the zoning is determined, project scheduling and control become straightforward as trades progress in an orderly fashion at the same, regular pace governed by the takt.

5.3 Literature Review

A literature search did not reveal any prior work that articulated the WoLZo problem. Closely related to the WoLZo problem is the rectangular tiling problem (also known as rectangle tiling problem in the literature). The latter has applications in 2-dimensional histograms, database mining, and data partitioning.

In the rectangular tiling problem, a matrix needs to be partitioned into at most p rectangles such that the weight of the rectangle with the largest weight is minimized, where the weight of a rectangle is the summation of all array elements inside it. (Later in this chapter, the weight will be expressed as a work density).

Khanna, Muthukrishnan, and Paterson [43] proved that an approximation within a factor of 5/4 for the rectangular tiling problem is NP-hard. In other words, finding a feasible solution whose objective value is at most 25% more than the optimal objective value is proved to be NP-hard. To the best of our knowledge, the tightest approximation algorithm for this problem is given byPaluch [60]: it is a 17/8 approximation with a running time that is linear in the size of the matrix.

As the rectangular tiling problem can be reduced to the WoLZo problem (which we formally define in the next section), that problem is also NP-hard.

Although the rectangular tiling and the WoLZo problem are similar in some respects, they differ in other respects. First, in rectangular tiling, each array element has one weight, whereas in the WoLZo problem, each cell has multiple weights with each weight representing the work density of a trade. In other words, each trade has a separate work density matrix in which each cell represents how much time that trade will require to do their work in the associated cell in the work space. Second, in rectangular tiling, each partition/zone must be rectangular, whereas in the WoLZo problem, the shape of zones can be more complex (e.g., L-shapes).

Note also that the WoLZo problem differs from the multidimensional rectangular tiling problem that was discussed in multiple papers (e.g., [61]). In the WoLZo problem, although multiple matrices of weights exist, the goal is to find only a single zoning plan, one that levels the work densities per zone for all trades. In contrast, in the multidimensional rectangular tiling problem, each 2-dimensional slice of the multidimensional array can have a distinct partition.

The following section describes two formulations of the WoLZo problem.

5.4 The Models

We model the WoLZo problem as a MILP because zones are demarcated by grid cells and the grid is defined using integer values. Given a work density map of the type illustrated in Figure 5.3, the WoLZo problem involves partitioning the work space into zones in order to minimize the maximum workload per zone (expressed as work density) across trades engaged in a sequence of process steps necessary to realize a phase of construction work. For a given number of zones, Z, and the appropriately subdivided work density map, the problem can equivalently be seen as partitioning a $M \times N$ grid, into Z disjoint zones, in order to minimize the maximum amount of work density in a zone over all zones and all trades – in other words, to find the minimum takt for a given number of zones.

Recall that each trade gets the same amount of time to do their work in each zone, so the minimum takt is equal to the maximum work density of any trade in any zone. For instance in Figure 5.4, the Electrical crew needs 34.5 time units in zone B, which is the maximum among all trades and all zones. Hence, takt can be defined as 34.5 time units for the depicted partition. Thus, to minimize the takt, we minimize the maximum work density of any trade in any zone.

In order to model this problem as a MILP, and to divide the work area into reasonable zones, we restrict the possible set of zones in several ways. First, we require each zone to be a single connected region, since a zone divided into several parts might require the travel of a trade back and forth between zones, which might disrupt the work of other trades.

In addition, we require zones to have relatively simple shapes. For example, Figure 5.5(a) shows that choosing complex and irregular shapes for the zones—although it might help reduce the takt—is probably not practical. Without additional detail about the project, it seems intuitively clear that zoning 5.5(a) is less practical or implementable than zoning 5.5(b).

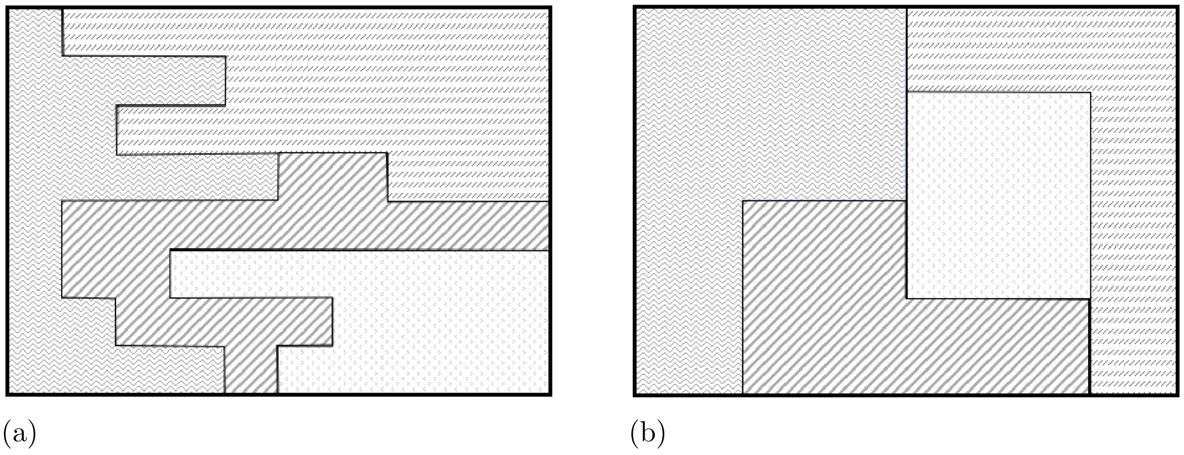


Figure 5.5: Shapes of the zones: (a) irregular complex shapes and (b) simple shapes

To facilitate our optimization model, in Model R (which we formally define in the Section 5.4.1), we define a simple shape as a rectangle, and in Model L (which we formally define in the Section 5.4.2), we define a simple shape as a shape with at most six facets, where all facets are horizontal or vertical (perpendicular). Simple shapes are thus both rectangle and L-shapes, where L-shapes are regions that take the shape of a rotated or flipped "L". These shapes are relatively simple and implementable in practice, and also amenable to modeling in a MILP. Figure 5.6 illustrates shapes that meet these restrictions.

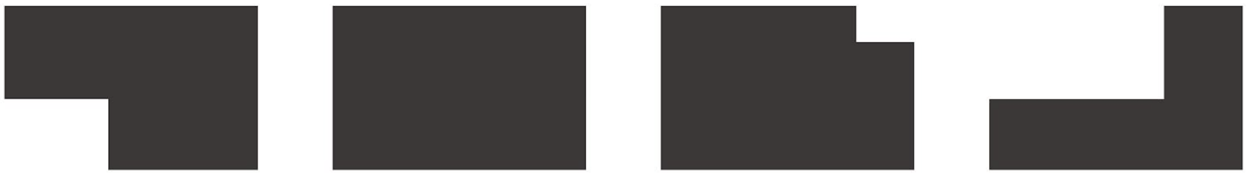


Figure 5.6: Shapes with at most six facets, where all facets are perpendicular

A trade-off exists between the quality of the solution and the simplicity of the shapes. As the set of allowable shapes become constrained, the optimal takt increases. Hence, we expect the optimal takt of Model L to be smaller (which means better, as this value is to be minimized) than that of Model R.

To develop our mathematical optimization model, we first assume that we are optimizing in an $M \times N$ grid. We define set I to be $\{0, 1, 2, ..., M\}$, set $I(i)$ to be $\{i+1, i+2, ..., M\}$, and set $\bar{I}(i)$ is to be $\{0, 1, 2, ..., i\}$. Similarly, we define sets J, $J(j)$, and $\bar{J}(j)$ to be $\{0, 1, ..., N\}$, $\{j+1, ..., N\}$, and $\{0, 1, ..., j\}$, respectively.

We introduce binary variables x_{i_1,j_1,i_2,j_2} for each possible rectangular shape, and $y_{i_1,j_1,i_2,j_2,i_3,j_3,q}$ for each possible L-shape, where these variables equal 1 if the corresponding zone is in the solution, and 0 otherwise.

Specifically, for rectangle shapes and corresponding variables x_{i_1,j_1,i_2,j_2}, indices (i_1, j_1) represent the coordinates of the lower left corner of the rectangle, and indices (i_2, j_2) represent coordinates of the upper right corner of the rectangle.

Any L-shape can be viewed as a smaller rectangle subtracted from a larger rectangle. To capture these shapes, for corresponding variables $y_{i_1,j_1,i_2,j_2,i_3,j_3,q}$, indices (i_1, j_1) represent the coordinates of the lower left corner of the larger rectangle, indices (i_2, j_2) represent coordinates of the upper right corner of the larger rectangle, indices (i_3, j_3) represent coordinates of the reflex (interior) angle of the L-shape, and index q represents the part of the rectangle that is missing, where q is in the set $Q = \{1, 2, 3, 4\}$. q can be viewed as representing the missing quadrant assuming (i_3, j_3) is the origin, as illustrated in Figure 5.7.

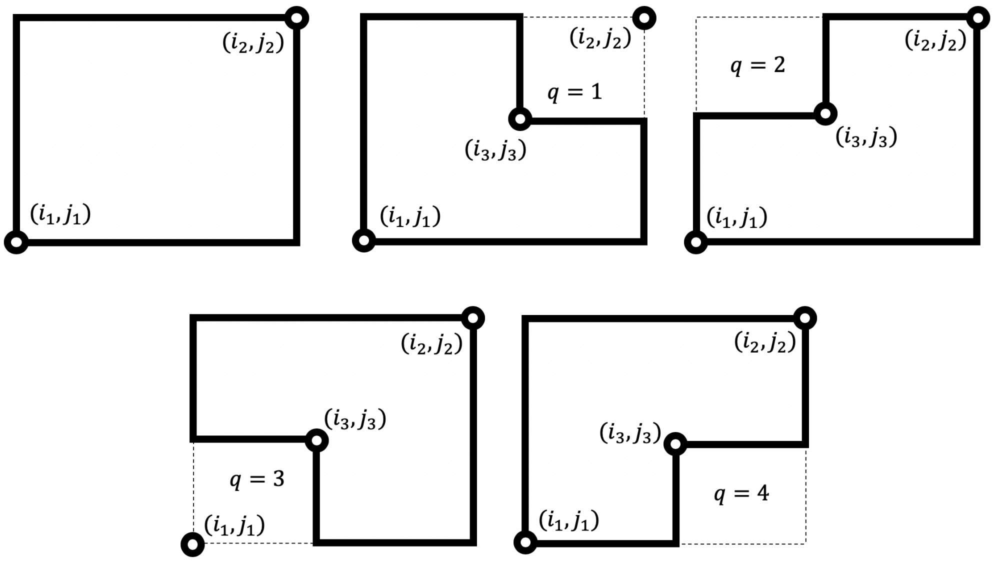


Figure 5.7: Notation for rectangular and L-shaped zones

The cost of including a particular zone in the solution is the maximum work density in that zone, where the maximum is taken over all steps taking place in that zone. Specifically, we define C_{i_1,j_1,i_2,j_2} to be the cost of including the rectangular zone x_{i_1,j_1,i_2,j_2} in the solution, and $C_{i_1,j_1,i_2,j_2,i_3,j_3,q}$ to be the cost of including the L-shaped zone $y_{i_1,j_1,i_2,j_2,i_3,j_3,q}$ in the solution.

As depicted in Figure 5.8, the work density in each cell is represented by parameter $w_{i,j}$, where (i, j) are the coordinates of the lower left corner of the cell.

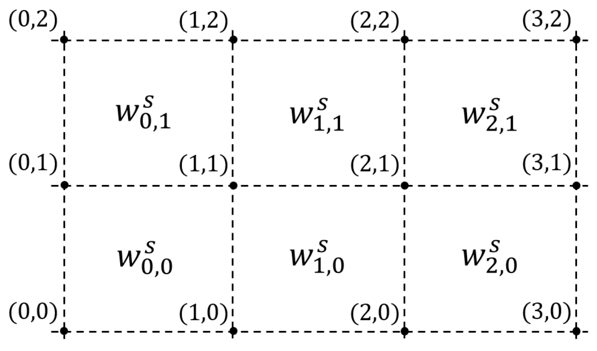


Figure 5.8: Notation tying work density to each cell

We define S to be the set of steps, and for every L-shaped region and $s \in S$, we define A^s to be the sum of work densities for s in each potential 'cut-out' rectangle of the L-shaped region:

$$A^s(1) = \sum_{i=i_3}^{i_2-1} \sum_{j=j_3}^{j_2-1} w^s_{i,j}$$

$$A^s(2) = \sum_{i=i_1}^{i_3-1} \sum_{j=j_3}^{j_2-1} w^s_{i,j}$$

$$A^s(3) = \sum_{i=i_1}^{i_3-1} \sum_{j=j_1}^{j_3-1} w^s_{i,j}$$

$$A^s(4) = \sum_{i=i_3}^{i_2-1} \sum_{j=j_1}^{j_3-1} w^s_{i,j}$$

Given these definitions, the cost of including a particular zone in the solution is captured by the following expressions:

$$C_{i_1,j_1,i_2,j_2} = \max_{s \in S} \left(\sum_{i=i_1}^{i_2-1} \sum_{j=j_1}^{j_2-1} w^s_{i,j} \right) \tag{5.3}$$

$$C_{i_1,j_1,i_2,j_2,i_3,j_3,q} = \max_{s \in S} \left(\sum_{i=i_1}^{i_2-1} \sum_{j=j_1}^{j_2-1} w^s_{i,j} - A^s(q) \right) \tag{5.4}$$

Finally, we define parameter Z to represent the number of zones we are identifying, $T(Z)$ to be a continuous variable equal to the takt, and intermediate variables $U^q_{i_1,j_1,i_2,j_2}(i,j)$ and V_{i_1,j_1,i_2,j_2} for clarity of exposition. $U^q_{i_1,j_1,i_2,j_2}(i,j)$ equals the number of L-shaped regions in the solution with larger rectangle (i_1, j_1) and (i_2, j_2) whose missing part is quadrant q and the missing part contains the cell identified by (i, j) and $(i + 1, j + 1)$, and V_{i_1,j_1,i_2,j_2} is the total number of selected L-shapes whose larger rectangle is formed with (i_1, j_1) and (i_2, j_2).

Given these definitions, we can write the following MILPs of Model R (in Subsection 5.4.1) and Model L (in Subsection 5.4.2).

5.4.1 Model R: All Shapes Must be Rectangles

Model R constrains partitions so that they are divided into only simple, rectangular shapes.

$$(WoLZo) \quad Minimize\, T(Z) \tag{5.5}$$

$$\sum_{i_1 \in I} \sum_{j_1 \in J} \sum_{i_2 \in I(i_1)} \sum_{j_2 \in J(j_1)} x_{i_1,j_1,i_2,j_2} = Z \tag{5.6}$$

$$\sum_{i_1 \in \bar{I}(i)} \sum_{j_1 \in \bar{J}(j)} \sum_{i_2 \in I(i)} \sum_{j_2 \in J(j)} x_{i_1,j_1,i_2,j_2} = 1 \qquad \forall i \in I - \{M\}, j \in J - \{N\} \tag{5.7}$$

$$T(Z) \geq C_{i_1,j_1,i_2,j_2} \times x_{i_1,j_1,i_2,j_2} \qquad \forall i_1 \in I, j_1 \in J, i_2 \in I(i_1), j_2 \in J(j_1) \tag{5.8}$$

$$x_{i_1,j_1,i_2,j_2} \in \{0, 1\} \qquad \forall\, i_1, i_2 \in I, j_1, j_2 \in J \tag{5.9}$$

Constraint 5.6 ensures that Z zones are selected. Constraint 5.7 ensures that every cell is in exactly one zone (zones are mutually exclusive and collectively exhaustive). Constraint 5.8 ensures that the takt is greater than or equal to the cost of all zones that are included in the solution and Constraint 5.9 ensures variables x_{i_1,j_1,i_2,j_2} for all $i_1, i_2 \in I, j_1, j_2 \in J$ only get binary values. Finally, Equation 5.5, minimizes the takt.

5.4.2 Model L: All Shapes Must be Either Rectangles or L-shapes

Model L constrains partitions so that they are divided into simple shapes that are either rectangular or L-shaped.

$$(WoLZo) \quad Minimize\, T(Z) \tag{5.10}$$

$$V_{i_1,j_1,i_2,j_2} = \sum_{i_3=i_1+1}^{i_2-1} \sum_{j_3=j_1+1}^{j_2-1} \sum_{q \in Q} y_{i_1,j_1,i_2,j_2,i_3,j_3,q} \qquad \forall i_1 \in I, j_1 \in J, i_2 \in I(i_1), j_2 \in J(j_1) \tag{5.11}$$

$$\sum_{i_1 \in I} \sum_{j_1 \in J} \sum_{i_2 \in I(i_1)} \sum_{j_2 \in J(j_1)} (x_{i_1,j_1,i_2,j_2} + V_{i_1,j_1,i_2,j_2}) = Z \tag{5.12}$$

$$\sum_{i_1 \in \bar{I}(i)} \sum_{j_1 \in \bar{J}(j)} \sum_{i_2 \in I(i)} \sum_{j_2 \in J(j)} (x_{i_1,j_1,i_2,j_2} + V_{i_1,j_1,i_2,j_2} - \sum_{q \in Q} U^q_{i_1,j_1,i_2,j_2}(i,j)) = 1$$
$$\forall i \in I - \{M\}, j \in J - \{N\} \tag{5.13}$$

$$U^1_{i_1,j_1,i_2,j_2}(i,j) = \sum_{i_3=i_1+1}^{i} \sum_{j_3=j_1+1}^{j} y_{i_1,j_1,i_2,j_2,i_3,j_3,1}$$
$$\forall i \in I, j \in J, i_1 \in \bar{I}(i), j_1 \in \bar{J}(j), i_2 \in I(i), j_2 \in J(j) \tag{5.14}$$

$$U^2_{i_1,j_1,i_2,j_2}(i,j) = \sum_{i_3=i+1}^{i_2-1} \sum_{j_3=j_1+1}^{j} y_{i_1,j_1,i_2,j_2,i_3,j_3,2}$$
$$\forall i \in I, j \in J, i_1 \in \bar{I}(i), j_1 \in \bar{J}(j), i_2 \in I(i), j_2 \in J(j) \tag{5.15}$$

$$U^3_{i_1,j_1,i_2,j_2}(i,j) = \sum_{i_3=i+1}^{i_2-1} \sum_{j_3=j+1}^{j_2-1} y_{i_1,j_1,i_2,j_2,i_3,j_3,3}$$
$$\forall i \in I, j \in J, i_1 \in \bar{I}(i), j_1 \in \bar{J}(j), i_2 \in I(i), j_2 \in J(j) \tag{5.16}$$

$$U^4_{i_1,j_1,i_2,j_2}(i,j) = \sum_{i_3=i_1+1}^{i} \sum_{j_3=j+1}^{j_2-1} y_{i_1,j_1,i_2,j_2,i_3,j_3,4}$$
$$\forall i \in I, j \in J, i_1 \in \bar{I}(i), j_1 \in \bar{J}(j), i_2 \in I(i), j_2 \in J(j) \tag{5.17}$$

$$T(Z) \geq C_{i_1,j_1,i_2,j_2} \times x_{i_1,j_1,i_2,j_2} \qquad \forall i_1 \in I, j_1 \in J, i_2 \in I(i_1), j_2 \in J(j_1) \tag{5.18}$$

$$T(Z) \geq C_{i_1,j_1,i_2,j_2,i_3,j_3,q} \times y_{i_1,j_1,i_2,j_2,i_3,j_3,q}$$
$$\forall i_1 \in I, j_1 \in J, i_2 \in I(i_1), j_2 \in J(j_1),$$
$$i_3 \in I(i_1) \cap \bar{I}(i_2 - 1), j_3 \in J(j_1) \cap \bar{J}(j_2 - 1), q \in Q \tag{5.19}$$

$$x_{i_1,j_1,i_2,j_2}, y_{i_1,j_1,i_2,j_2,i_3,j_3,q} \in \{0,1\} \qquad \forall\, i_1,i_2,i_3 \in I, j_1,j_2,j_3 \in J, q \in Q \tag{5.20}$$

Constraint 5.11 defines variables V_{i_1,j_1,i_2,j_2} and Constraint 5.12 ensures that Z zones are selected. Constraint 5.13 ensures that every cell is in exactly one zone (zones are mutually exclusive and collectively exhaustive). Constraints 5.14 to 5.17 define $U^q_{i_1,j_1,i_2,j_2}(i,j)$. Constraints 5.18 and 5.19 ensure that the takt is greater than or equal to the cost of all zones that are included in the solution and Constraint 5.20 ensures variables x_{i_1,j_1,i_2,j_2} and $y_{i_1,j_1,i_2,j_2,i_3,j_3,q}$ for all $i_1,i_2,i_3 \in I, j_1,j_2,j_3 \in J, q \in Q$ only get binary values. Finally, Equation 5.10, minimizes the takt.

5.5 Solution Method

As mentioned, the WoLZo problem is NP-hard. Standard solvers, e.g., Gurobi (version 9.0), cannot directly find the optimal solution of the formulation (5.10)-(5.19) for a problem with a realistic size (e.g., 108 cells in the grid) within 3,600 seconds. The WoLZo algorithm was therefore developed to help solve the WoLZo problem efficiently.

The WoLZo algorithm focuses on reducing the size of the problem by removing the redundant variables. To identify the redundant variables, the algorithm makes it necessary to find an upper bound on the optimal takt. Clearly, any zone with a cost higher than the upper bound will not be selected as the optimal solution; hence, their corresponding variables can be removed from the model.

A tight upper bound can significantly reduce the size of the problem; however, such an upper bound is not a-priori known. This inspired us to implement a procedure that attempts to search for a tight upper bound by first estimating a lower bound, and systematically increasing estimates of an upper bound as a function of that lower bound, starting at a small amount above the lower bound, until a feasible solution is found.

Figure 5.9 depicts the flowchart of the WoLZo algorithm. The algorithm starts with a simple lower bound, LB, computed by dividing the highest [among all trades] total work density of the entire work space by the number of zones:

$$LB = \frac{C_{0,0,M,N}}{Z} \tag{5.21}$$

To illustrate, using the data depicted in Figure 5.3 where Electrical has the highest work density, and assuming $Z = 3$:

$$LB = \frac{70.1}{3} = 23.367$$

Next, we select a step ratio, α, for the procedure. This step ratio must be greater than 1 (e.g., 1.05). Then our initial candidate for an upper bound is $\hat{UB} := \alpha \times LB$, and we add Constraint 5.22 to the model.

$$T(Z) \leq \hat{UB} \tag{5.22}$$

Similar to what is discussed by Mingozzi and Morigi [56], this constraint limits the solution to zones whose costs are less than or equal to the $\hat{UB}$. Hence, variables corresponding to the zones with higher costs can be eliminated, and doing so significantly reduces the problem size. Given this constraint, two outcomes are possible.

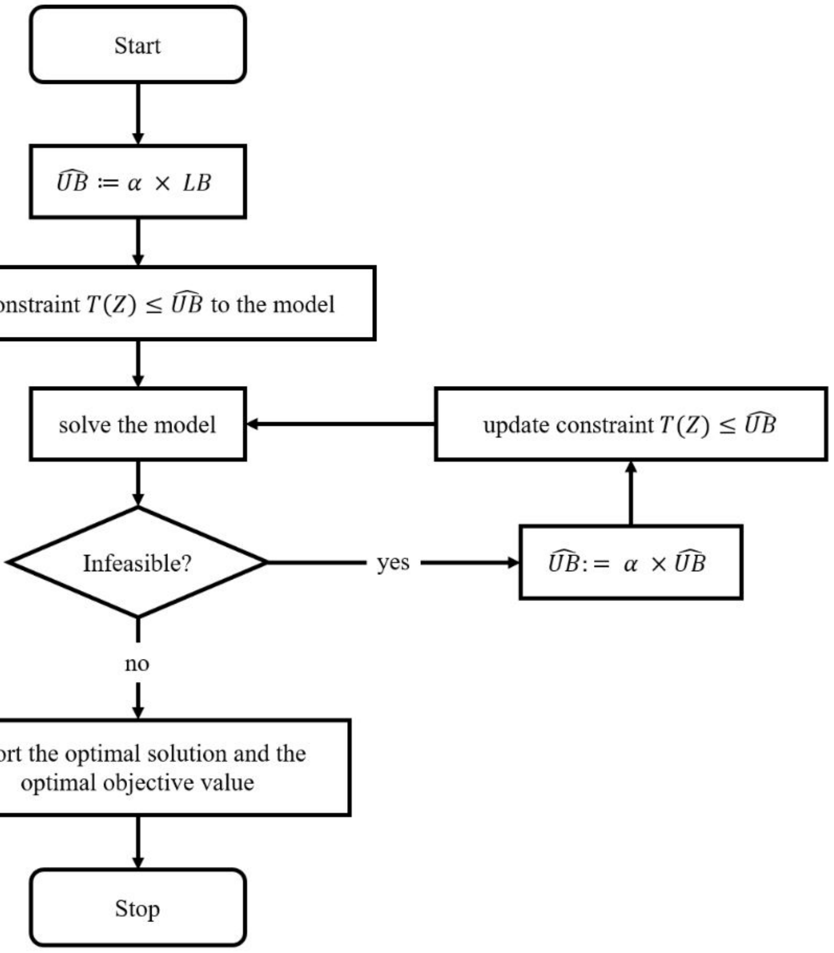

Start
$\widehat{UB} \coloneqq \alpha \times LB$
add constraint $T(Z) \leq \widehat{UB}$ to the model
solve the model
update constraint $T(Z) \leq \widehat{UB}$
Infeasible?
yes
$\widehat{UB}: = \alpha \times \widehat{UB}$
no
report the optimal solution and the optimal objective value
Stop

Figure 5.9: Solution procedure flowchart

One outcome is that, if the candidate upper bound ($\hat{UB}$) is not in fact an actual upper bound on the optimal takt, but is strictly less than it, the model will be infeasible. Because the number of variables in this infeasible model is relatively small, standard solvers can determine this infeasibility in a matter of seconds. In this case, the procedure iterates: we guess a new upper bound for the model as $\hat{UB} := \alpha \times \hat{UB}$. As *alpha* is greater than one, the new upper bound is larger than the previous one. After updating Constraint 5.22 and consequently removing the redundant variables, the model is solved again. This iteration continues until a model is reached that has a feasible solution.

The other outcome is that, if the candidate $\hat{UB}$ is in fact an actual upper bound, the model is again typically relatively small, and it can be optimally solved using standard solvers. The optimal solution of this restricted model is also the optimal solution to the original problem.

In the WoLZo algorithm, if we choose a relatively tiny step ratio (e.g., $\alpha = 1.001$), then many iterations may need to occur before the algorithm reaches a model with a feasible solution. In contrast, if the step ratio is large (e.g., $\alpha = 2.0$), the upper bound might not be tight, which can result in a model with a large number of variables that is not efficient to solve. Hence, finding an appropriate step ratio can speed up the algorithm. In practice, if the number of iterations needed to keep increasing $\hat{UB}$ is noticeably high when using the WoLZo algorithm, increasing *alpha* is helpful. In contrast, if the final model (the one with a feasible solution) takes a long time to solve, decreasing *alpha* might help reduce the total run time of the algorithm.

5.6 Application Example

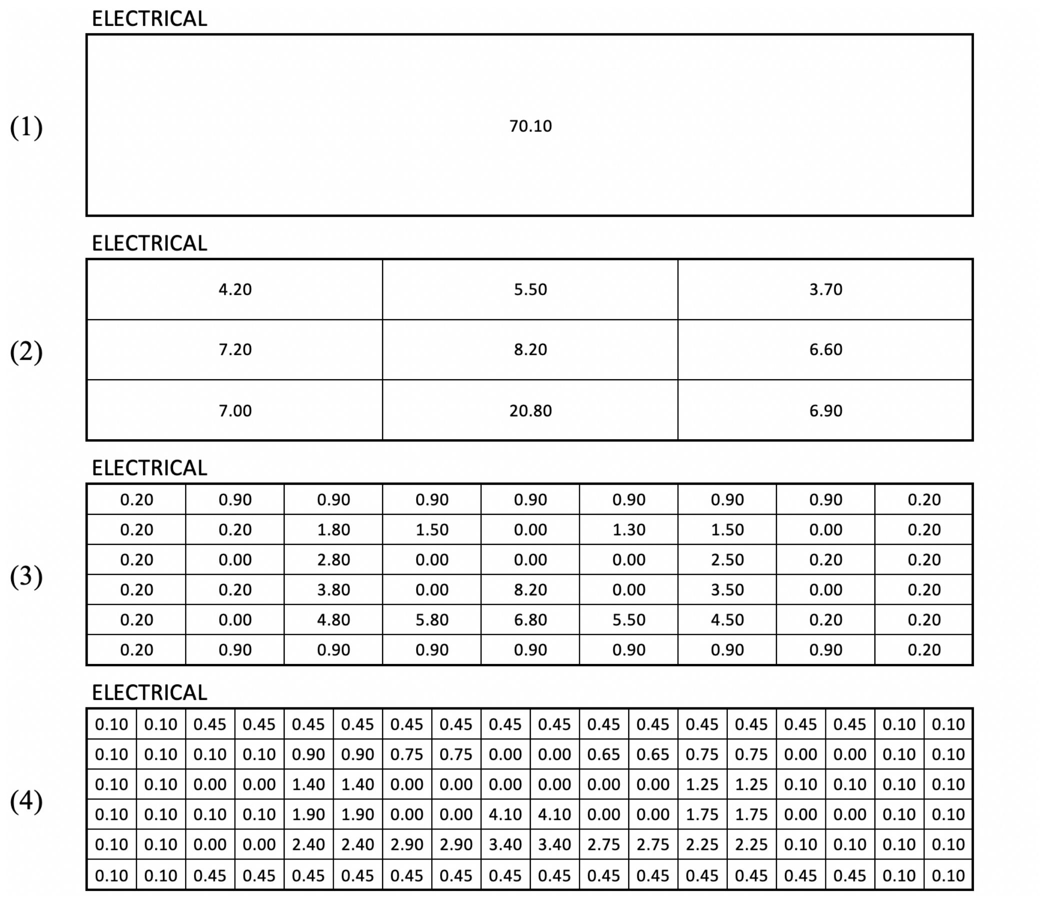
5.6.1 Input Data Set

The WoLZo algorithm was used to study four scenarios that differ in the granularity at which work density is depicted. The goal was to highlight how the number of cells dividing a given work area impacts the solution space.

The scenarios all use input data regarding work density of the 4-step process illustrated in Figures 5.3 and 5.4. Using the Electrical trade as an example and given its 6×9 matrix (corresponding to Scenario 3), Figure 5.10 shows the work densities for the other three scenarios. In this example, the densities were computed by aggregating cells into larger ones or by dividing cells evenly into smaller ones.

Note that in reality, when work density data can be collected directly from project data, a computation using addition or division might not be in order. For example, a small cell with a high work density may not be divisible due to the nature of the work (e.g., installation of a large assembly) or that cell may well be surrounded by cells with 0 work density, so that aggregating them would result in a larger cell with the same high work density. The four scenarios are the following:

Scenario 1: 1×1 matrix (1 cell covers the entire work area)
Scenario 2: 3×3 matrix (9 cells)
Scenario 3: 6×9 matrix (54 cells)
Scenario 4: 6×18 matrix (108 cells)



(1)

ELECTRICAL

70.10

(2)

ELECTRICAL

4.20	5.50	3.70
7.20	8.20	6.60
7.00	20.80	6.90

(3)

ELECTRICAL

0.20	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.20
0.20	0.20	1.80	1.50	0.00	1.30	1.50	0.00	0.20
0.20	0.00	2.80	0.00	0.00	0.00	2.50	0.20	0.20
0.20	0.20	3.80	0.00	8.20	0.00	3.50	0.00	0.20
0.20	0.00	4.80	5.80	6.80	5.50	4.50	0.20	0.20
0.20	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.20

(4)

ELECTRICAL

0.10	0.10	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.10	0.10
0.10	0.10	0.10	0.10	0.90	0.90	0.75	0.75	0.00	0.00	0.65	0.65	0.75	0.75	0.00	0.00	0.10	0.10
0.10	0.10	0.00	0.00	1.40	1.40	0.00	0.00	0.00	0.00	0.00	0.00	1.25	1.25	0.10	0.10	0.10	0.10
0.10	0.10	0.10	0.10	1.90	1.90	0.00	0.00	4.10	4.10	0.00	0.00	1.75	1.75	0.00	0.00	0.10	0.10
0.10	0.10	0.00	0.00	2.40	2.40	2.90	2.90	3.40	3.40	2.75	2.75	2.25	2.25	0.10	0.10	0.10	0.10
0.10	0.10	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.45	0.10	0.10

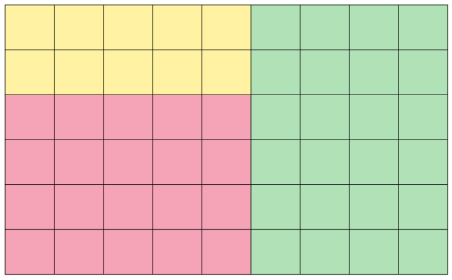
Figure 5.10: Electrical work density based on a (1) 1×1 matrix, (2) 3×3 matrix, (3) 6×9 matrix, and (4) 6×18 matrix

5.6.2 Optimization Results

Optimization of Takt $T(Z)$ for Single Work Space

The results obtained using the WoLZo algorithm to partition the work space are illustrated next, based on Scenario 3 and $Z = 3$. Appendixes B and C include detailed results from all scenarios.

The partition depicted in Figure 5.4—plausibly derived manually—resulted in a takt $T(3) = 34.5$. Use of the WoLZo algorithm on Model R (Figure 5.11a) resulted in a partition with a takt $T(3) = 26.4$, which is a 23% decrease in time from the initial solution of 34.5. Similarly, using the WoLZo algorithm on Model L (Figure 5.11b) results in a partition with a takt $T(3) = 23.5$, a 32% decrease.


(a)

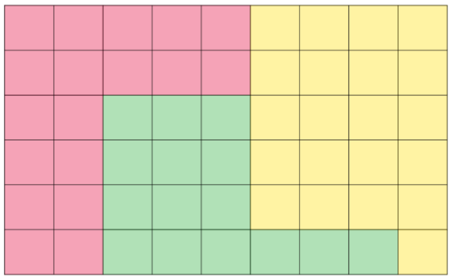
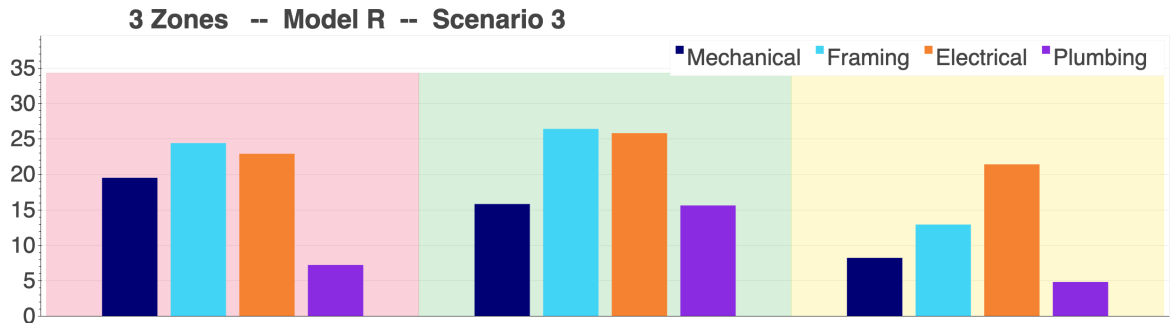
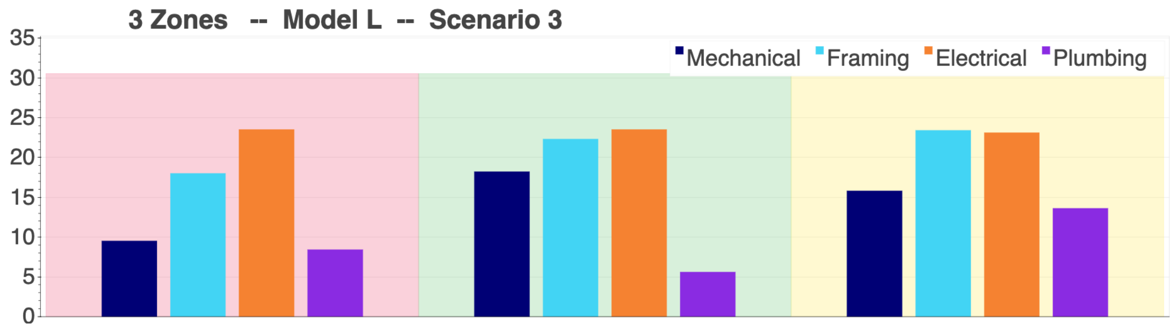

(b)

Figure 5.11: Optimal partition suggested by the WoLZo algorithm based on (a) Model R and (b) Model L (NOTE: each zone has its own color)

Figure 5.12 depicts the histograms that correspond to the partitions that the WoLZo algorithm provided, respectively based on Model R and Model L. The histograms show the workloads by zone for each step in the process, using as background the same colors as those used to depict zones in Figures 5.11a and 5.11b.



(a)



(b)

Figure 5.12: Histogram of workloads by zone for each step in the process based on (a) Model R and (b) Model L (NOTE: the background color matches the color depicting the zone in Figure 5.11)

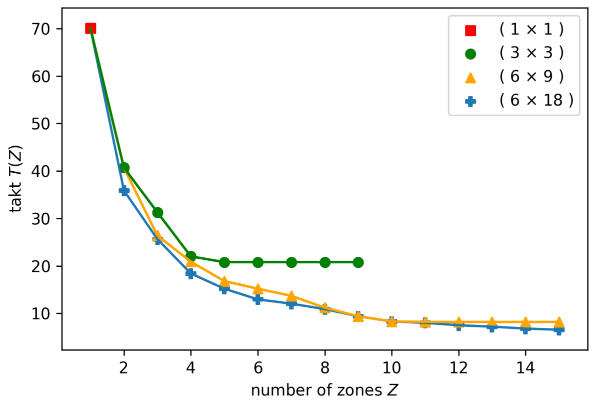
Table 5.1 shows the takt computed by the WoLZo algorithm using Model R for each of the four scenarios while varying Z from 1 to 15. Similarly, Table 5.2 shows the takt computed by WoLZo using Model L. Figure 5.13 plots this data, including the zoning that was shown in Figure 5.11 ($Z = 3$ and in Model R, $T(3) = 26.4$, and in Model L, $T(3) = 23.5$). This plot supports the following observations:

1. Due to the granularity of any work density matrix, a limit exists on the number of zones one can choose. At most, Z can be equal to the dimension of the work density matrix. By construction, a zone cannot be smaller than the size of a cell in that matrix. For example, in Model L, Scenario 2 with 9 cells reaches its limit at $Z = 9$, and Scenario 1 with 1 cell reaches its limit at $Z = 1$.

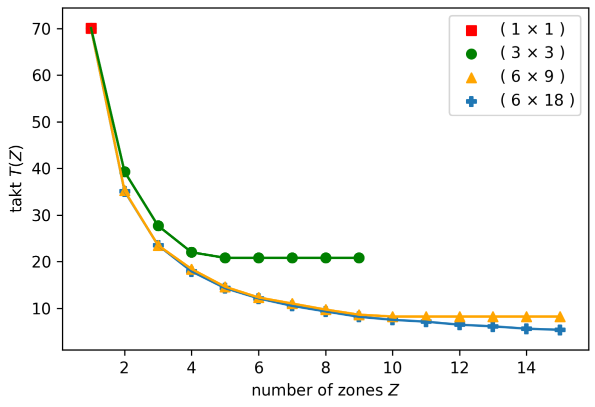
2. When the entire work area is a single zone ($Z = 1$), the solution ($T(1) = 70.1$) for the 1×1 matrix of Scenario 1 is indifferent to work densities being specified more finely. However, boundaries of zones may shift when Z is 2 or larger and matrices with finer granularity are used.

3. $T(Z)$ does not necessarily become smaller as Z increases. When the number of zones Z increases, the takt $T(Z)$ decreases or levels out, until the limit is reached. For example, Scenario 2 illustrates that increasing the number of zones does not keep on reducing the takt: in Model L, the takt remains constant from $Z = 5$ onward. This is the consequence of work density being distributed unevenly across the cells in the matrix.

4. When the granularity of the work density matrix decreases (cells in the matrix become smaller in area), the takt can become smaller. For example, for any value of Z, the takt for Scenario 4 is always smaller than or equal to the takt of Scenario 3. The same is true for Scenario 3 vs. 2, and 2 vs. 1.



(a)



(b)

Figure 5.13: Number of zones Z vs. takt $T(Z)$ in (a) Model R and (b) Model L

Table 5.1: Number of zones Z vs. takt $T(Z)$ computed with WoLZo algorithm for Model R

Number	Takt $T(Z)$			
of Zones Z	Scenario 1 (1×1)	Scenario 2 (3×3)	Scenario 3 (6×9)	Scenario 4 (6×18)
1	70.1	70.1	70.1	70.1
2		40.7	40.7	35.9
3		31.3	26.4	25.7
4		22.0	20.9	18.4
5		20.8	16.8	15.2
6		20.8	15.2	13.0
7		20.8	13.7	12.1
8		20.8	11.2	10.9
9		20.8	9.4	9.4
10			8.3	8.3
11			8.2	8.0
12			8.2	7.5
13			8.2	7.2
14			8.2	6.8
15			8.2	6.6

Table 5.2: Number of zones Z vs. takt $T(Z)$ computed with WoLZo algorithm for Model L

Number	Takt $T(Z)$			
of Zones Z	Scenario 1 (1×1)	Scenario 2 (3×3)	Scenario 3 (6×9)	Scenario 4 (6×18)
1	70.1	70.1	70.1	70.1
2		39.3	35.2	35.1
3		27.7	23.5	23.5
4		22.0	18.4	17.9
5		20.8	14.6	14.3
6		20.8	12.3	12.1
7		20.8	11.0	10.5
8		20.8	9.7	9.3
9		20.8	8.6	8.2
10			8.2	7.5
11			8.2	7.1
12			8.2	6.5
13			8.2	6.1
14			8.2	5.6
15			8.2	5.3

The details of using the WoLZo algorithm, including the run-times, for both Model R and Model L for every scenario while varying Z from 1 to 15 are presented in Appendix B.

Optimization of Process Duration D when Work Space is Replicated over Multiple Floors

The previous results focused on the takt obtained by zoning a single work space. Now assume that this work space represents a floor and gets replicated over multiple floors, and focus on the phase duration D. Based on the data from Tables 5.1 and 5.2 (illustrated in Figure 5.13), the duration D of a phase of work that extends over respectively 1, 2, 5, and 10 floors is computed by means of Equation 5.2 (these numbers of floors were selected just for illustrative purposes; the computation can be done likewise for any other number of floors). The takt times were first rounded up to the nearest integer, because in actuality they may represent a day or a week and decimal values of such units may be hard to implement. In practice, the decision of whether and how to round results from the WoLZo algorithm is up to the project team.

Figures 5.14 and 5.15 show the results for Model R and Model L, respectively. These support the following observations:

1. Dividing the work space into more zones than 1 tends to reduce the duration D (unless work density is exceptionally uneven, e.g., concentrated in a small area down to a single cell). Then, at some point, D starts to increase. Indeed, Equation 5.2 shows that F gets multiplied by the number of zones Z while (as discussed in Subsection 5.6.2) $T(Z)$ levels out when Z continues to increase.

2. The duration D may oscillate when Z increases (see for example Scenario 4 (6 x 18 matrix)). This is observable for any number of floors, even when $F = 1$ where it is known that the takt $T(Z)$ decreases or levels out as the number of zones Z increases (Figure 5.13). This is due to variation in the work density of adjacent cells combined with the imposition of rectangle or L-shapes when zoning the work space.

3. The benefit of dividing a work space into smaller zones (allowing for more concurrency), diminishes when the number of floors increases. Indeed, in Scenario 4 in Model R, the minimum D for $F = 1$ occurs at $Z = 11$, for $F = 2$ at $Z = 6$, for $F = 5$ also at $Z = 6$, and for $F = 10$ at $Z = 4$. Similarly, in Scenario 4 in Model L, the minimum D for $F = 1$ occurs at $Z = 14$, for $F = 2$ at $Z = 10$, for $F = 5$ at $Z = 4$, and for $F = 10$ also at $Z = 4$. In other words, the benefit of dividing a work space in more zones is greater relatively-speaking when the work area is replicated less.

Furthermore, practical considerations such as simplicity of a zoning also favor the selection of a value for Z on the lower end. For example, one may argue for the use of $Z = 4$ in the case of $F = 1$, 2, and 5, even though that does not result in the shortest duration D. Planners must consider the trade-off between a relatively-small incremental reduction in duration vs. selecting a lower Z, recognizing that a lower Z has benefits. These benefits include: requiring fewer hand-offs overall between trades involved in consecutive process steps, requiring fewer setups and movement of trades from one location to the next, allowing for larger areas to work in, etc.

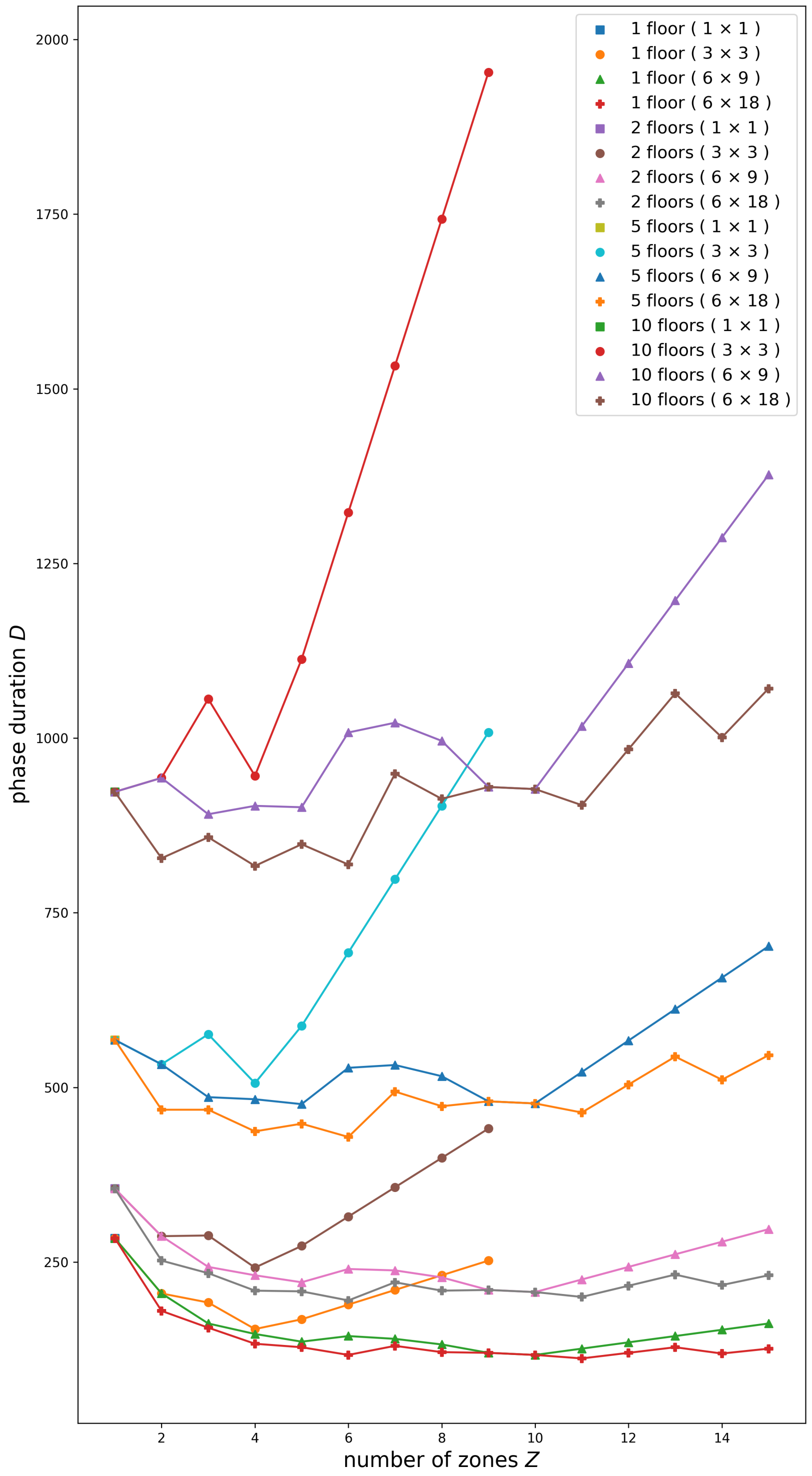


Figure 5.14: Phase duration vs. number of zones for 1, 2, 5, and 10 floors in Model R

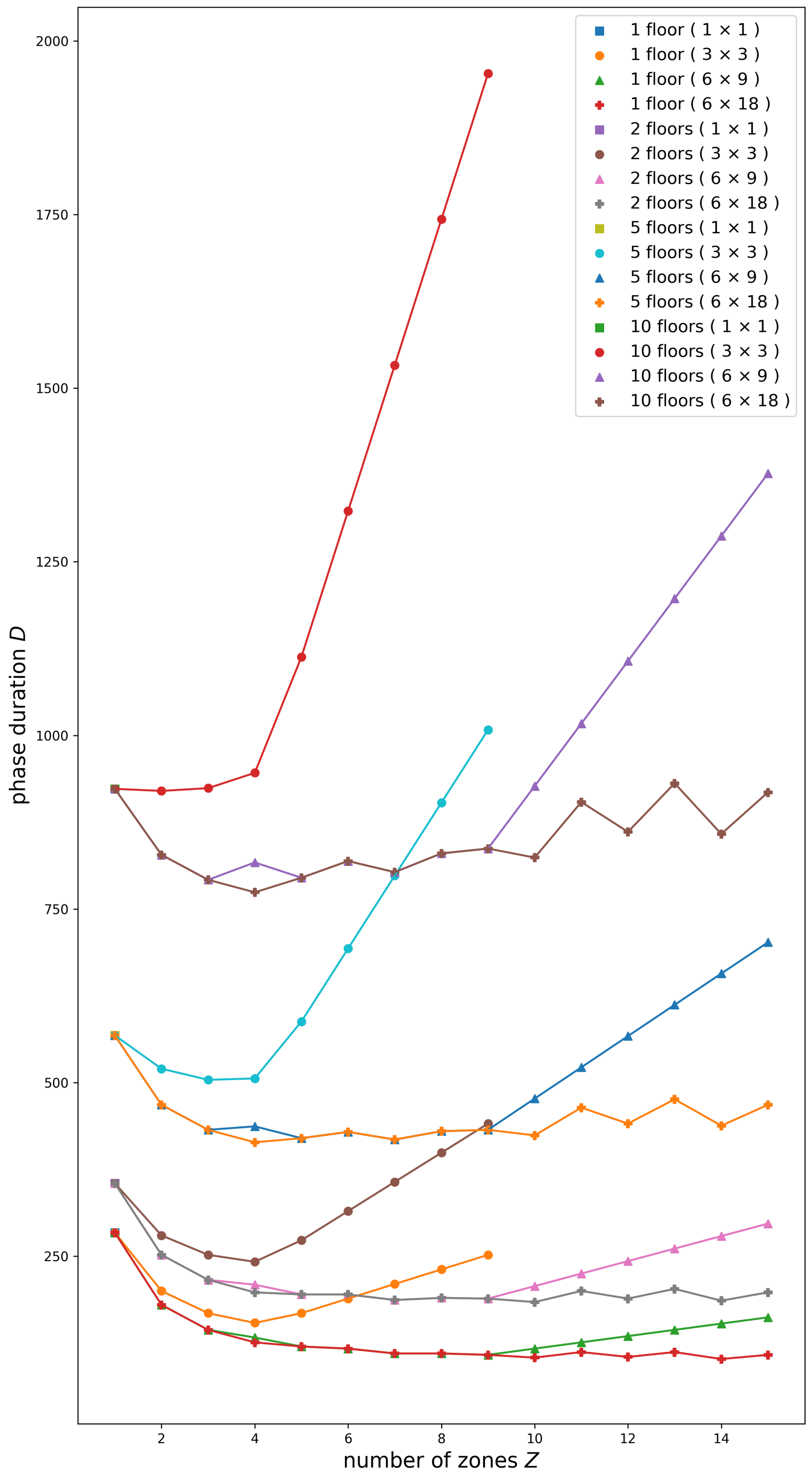


Figure 5.15: Phase duration vs. number of zones for 1, 2, 5, and 10 floors in Model L

5.6.3 Computer Implementation

The WoLZo algorithm is coded in Python 3 and solved by Gurobi version 9.0 Using a 2.3 GHz Intel Xeon E5-2670 v3. The average run-time of the algorithm for Z ranging from 1 to 15 (or up to the maximum number of cells in the grid in Scenarios 1 and 2) in different scenarios are summarized in Table 5.3, and the average number of iterations of the model to find the optimal solution is presented in Table 5.4. The results shown in Figure 5.11 took (a) 0.2 s and (b) 25.9 s when *alpha* is 0.05.

Table 5.3: Average run-time of the algorithm in different scenarios when acceptable shapes are either only rectangles or rectangles as well as L-shapes, and different choices of *alpha*

Model	Scenario (grid size)	Average run-time (seconds)				
		$\alpha = 1.005$	$\alpha = 1.01$	$\alpha = 1.05$	$\alpha = 1.25$	$\alpha = 1.5$
R	1 (1×1)	0.0	0.0	0.0	0.0	0.0
	2 (3×3)	0.2	0.1	0.0	0.0	0.0
	3 (6×9)	1.5	0.9	0.3	0.2	0.2
	4 (6×18)	6.8	3.7	1.3	1.1	1.2
L	1 (1×1)	0.0	0.0	0.0	0.0	0.0
	2 (3×3)	0.8	0.5	0.3	0.3	0.3
	3 (6×9)	54.0	37.2	26.3	25.6	37.3
	4 (6×18)	419.5	268.6	201.0	270.5	N/A*

* Some of the instances cannot be solved optimally in 3,600 s

Table 5.4: Average number of iterations of the algorithm in different scenarios when acceptable shapes are either only rectangles or rectangles as well as L-shapes, and different choices of *alpha*

Model	Scenario (grid size)	Average number of iterations				
		$\alpha = 1.005$	$\alpha = 1.01$	$\alpha = 1.05$	$\alpha = 1.25$	$\alpha = 1.5$
R	1 (1×1)	1.0	1.0	1.0	1.0	1.0
	2 (3×3)	94.4	47.6	10.1	2.7	1.7
	3 (6×9)	52.2	26.5	5.9	1.7	1.2
	4 (6×18)	34.1	17.3	3.9	1.3	1.0
L	1 (1×1)	1.0	1.0	1.0	1.0	1.0
	2 (3×3)	91.0	45.8	9.8	2.6	1.7
	3 (6×9)	36.1	18.5	4.3	1.5	1.2
	4 (6×18)	12.3	6.3	1.8	1.0	N/A*

* Some of the instances cannot be solved optimally in 3,600 s

For comparison in terms of processing time, on the same computer we ran Gurobi using its default settings to solve the model presented in Equations 5.10-5.20, that is, without the algorithm presented in Section 5.9. The solution times were similar for Scenarios 1 and 2, and for Scenario 3 with rectangles only. However, as the problem gets harder, as is the case for Scenario 3 considering both rectangles as well as L-shapes, the algorithm is faster by orders of magnitude. For example, for Scenario 4 and considering rectangles and L-shapes, (1) for the number of zones ranging from 2 to 4, Gurobi's solutions had an optimality gap of 100% (optimality gap defined as the ratio of the solver's (upper bound - lower bound) / upper bound), and (2) for the number of zones ranging from 5 to 15 Gurobi could not find a feasible solution within a 3,600 second time limit.

The individual run-times of each instance, when the algorithm is used and when it is not used, are shown in Appendix B.

5.7 Discussion

A discussion follows of how the modeling of work density affects the WoLZo problem formulation and optimization algorithm.

5.7.1 Assumptions and Limitations of using Work Density

The problem as formulated is based on the assumption that work density data can be obtained. Indeed, Tommelein [75] described how such data can be captured and represented cell by cell. While the results from the WoLZo algorithm demonstrate that a finer granularity (a work space divided in more grid cells) will likely result in a lower takt, there is a practical limit to how small zones (and thus cells) can be. Trades need space to work, e.g., to stage and handle materials in the zone, maneuver a scissor lift, etc. Each zone must provide sufficient space to each trade that is to work in it.

How small zones can be relates to how large Z can get and, as mentioned in Subsection 5.6.2, practical considerations will encourage planners to choose the number Z to be smaller rather than larger. But so far, no mention was made of the physical dimensions of each cell. Is the cell 1 m x 1 m, 10 m x 15 m, or does it have another length and width? Further study is in order.

The work density matrix is an abstraction with no mention of physical dimensions of cells. The WoLZo algorithm applies as long as the topology of the grid is maintained. Accordingly, the physical dimensions of a column (or row) of cells can differ from the physical dimensions of the adjacent columns (or rows) of cells.

In terms of the method followed to determine the takt, i.e., lowering the workload peak(s) across trades by re-zoning the work space, of note is that other methods exist. Work density maps are an abstract characterization of workloads by zone, reflecting the settings of a number of throttles for production: e.g., scope of work included, product to be put in place, means and methods, crew size, skill of crew members. These throttles may also be used to lower workload peaks.

A number of other methodological considerations come into play when developing a takt plan. They pertain to work structuring (e.g., how to phase a project and define Parades of Trades) including deciding on the sequencing of steps that make up a process (e.g., a workload peak may be lowered by allowing the crew to do their work not in one but in two consecutive steps). Further discussion of these methodological considerations is beyond the scope of this chapter.

5.7.2 Extending or Customizing the WoLZo Model

The WoLZo model is designed for rectangular work spaces but it has broader applicability. Figure 5.16 shows that it can be generalized to work spaces with any shape by fitting the given work space inside a rectangle and assuming the work density of all trades is 0 for all the added grid cells (e.g., when vertical work such as the elevator core and stair well are planned as a process separate from other work on a floor). In Model L, in case the algorithm then yields an L-shaped zone with a discontinuity in the middle due to the added grid cells with 0 work density, then remove those L-shapes from the model.

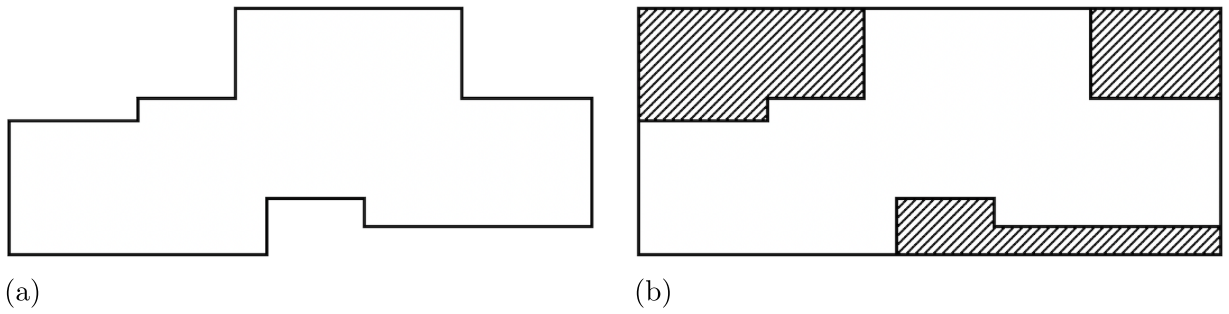


Figure 5.16: Transforming (a) any work space to (b) a rectangular work space (hashed area has work density of 0 for all trades)

For various reasons, it may be desirable to further constrain the problem. For instance, one might wish to add some or all of the following constraints:

- Forcing a few cells of the grid to be in the same zone:

  To do so, remove from Model R every rectangle, x_{i_1,j_1,i_2,j_2}, remove from Model L every rectangle, x_{i_1,j_1,i_2,j_2}, and L-shape, $y_{i_1,j_1,i_2,j_2,i_3,j_3,q}$, that contains some of those cells but not all of them.

- Forcing some of the cells to be in separate zones:

To do so, remove from Model R every rectangle, x_{i_1,j_1,i_2,j_2}, and remove from Model L every rectangle, x_{i_1,j_1,i_2,j_2}, and L-shape, $y_{i_1,j_1,i_2,j_2,i_3,j_3,q}$, that contains more than one of those cells.

- Forcing the zones to have a minimum/maximum size/height/width:

  Figure 5.17 clarifies the definition of size, width, and height of a zone. To add a constraint on those, remove from Model R every rectangle, x_{i_1,j_1,i_2,j_2}, and remove from Model L every rectangle, x_{i_1,j_1,i_2,j_2}, and L-shape, $y_{i_1,j_1,i_2,j_2,i_3,j_3,q}$, that does not follow the defined standards pertaining to minimum/maximum size/height/width.

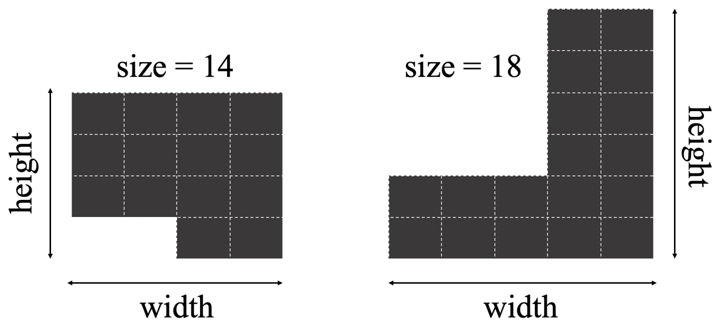


Figure 5.17: Definition of size, height, and width of a zone

Such constraints may help make zones somehow "better" to work in. This notwithstanding, added constraints will likely result in a higher takt. They will not likely impact the WoLZo algorithm's run-time, however, since we can implement these constraints in pre-processing by removing some of the variables.

5.7.3 Model R Compared to Model L

A comparison between Model R and Model L is in order. On the one hand, Model R is faster to solve and it is scalable to larger grid sizes than Model L is. The zones in Model R have more straightforward shapes compared to those in Model L (rectangles vs. both rectangles as well as L-shapes). On the other hand, for the same grid size, Model L is likely to find a better takt than Model R because Model L allows for a greater variety of shapes.

5.7.4 Optimal Solutions Compared with Manual Solutions

It is common practice for construction site managers to divide work spaces into smaller areas. Superintendents may break up a large floor into quadrants, for example, to make their work more manageable and speed up the schedule by planning for concurrency. However, their rationale, if not informed by work density, causes unevenness in the workflow or requires that trades crew up or down. People also tend to use very simple shapes (e.g., rectangles) instead of looking for perhaps more complex ones (e.g., L-shapes) that can be better from a workload leveling perspective.

These practices are understandable because, depending on the granularity at which work density is considered, the number of possible zoning arrangements grows exponentially and there may seem to be infinitely many of them. Constrained by cognitive ability and time, this is why people settle on simple divisions. It is just about impossible to manually determine the mathematical optimum.

From our experience working with practitioners to explore zoning options and using takt planning, we know that people tend to think of zoning the work space with earlier process steps in mind, rather than considering all steps in the process (that is, we tend to be greedy). In contrast, the WoLZo model gives equal consideration to the workload associated with each and every step in any given process. The WoLZo model does not reflect any bias towards earlier vs. later work.

Computational methods can thus augment construction site managers' decision making by helping them pursue clear planning objectives and consider trade-offs. As computational prowess continues to increase, thanks to the combination of increasing hardware capabilities and inventive algorithm design, new construction planning methods such as the Work Density Method will gain adoption in practice.

Chapter 6

Conclusion

In this dissertation, we developed models and approaches that are particularly useful for managing different elements of mega-projects, primarily in the oil and gas industry. These tools enable the concurrent determination of project schedules and inventory delivery times in order to efficiently manage the project supply chain, and to effectively control project delivery time and cost.

We integrated the stochastic resource-constrained project scheduling problem with inventory considerations and developed comprehensive models and multiple heuristics to optimize the problem over the proactive class of policies. We also introduced an approach to find a lower bound and probabilistic lower bounds to evaluate the performance of heuristics. Our results indicate that the advantages of our comprehensive model are two-fold. First, implementing the stochastic version of the model both decreases the expected makespan of the project and also dramatically lowers expected inventory holding cost, even when the variability of activity durations is small. Second, simultaneously optimizing the schedule and procurement deliveries significantly reduces the project cost. We also developed techniques to provide the inventory cost versus timeliness efficient frontier, which can serve as a fruitful managerial decision-making tool that sheds light on the relationship between the cost and timeliness of projects.

For oil field drilling operations, we developed a comprehensive model and proposed a decomposition-based heuristic approach to solve the model. Using data from a real-world project, we demonstrated that adopting our approach leads to significant improvements in terms of time and cost compared to the current practice in the industry. We used our model to evaluate how myopic constraints on the transportation of resources (typically resulting from the application of rules-of-thumb) impacts project performance. Lastly, we measured the value of acquiring additional resources on the expected revenue generated by oil extraction. The insights from this analysis can be used as a tool for making managerial decisions.

Motivated by our interviewees, we presented a stylized model for suppliers' decision making, where we consider sequencing decisions on a single processor, here representing a supplier, in an online setting where no data about the future incoming opportunities is available. With the goal of minimizing total weighted (modified) earliness and tardiness cost, we introduce a new policy, the list-based delayed shortest processing time policy, and we develop lower and upper bounds on its performance for several related problems. In the theoretical aspect, we closed the optimality gap that previously exists in the literature for several variants of single machine online scheduling problems in the presence of due dates by proving that our proposed policy is an optimal online algorithm for these variants.

Finally, we presented a novel formulation and algorithm to address the NP-hard problem of dividing a work space into distinct zones in order to level workloads by zone across process steps and for all zone, and made it possible to find the optimal solution rapidly for practical problem sizes. We described current practices related to dividing construction work areas into smaller ones and contrasted these with the zoning determined using our approach.

While we believe we have significantly advanced the state-of-the-art in those areas considered in this dissertation, our work has limitations. We do not consider uncertainty in deliveries as we assumed that the suppliers deliver materials on time. We focused on average outcomes without considering risk measures. Many of our solution approaches are NP-hard in the strong sense, which is not an issue for the projects with 30-60 activities as we leveraged the immense amount of technology that is developed for solving specific problems, but clearly is not scalable to projects with many more activities.

These observations suggest a variety of questions that can be addressed in future research. How can we integrate suppliers into a comprehensive model, and optimize a project over the entire supply chain rather than considering one stakeholder at a time? How does varying the level of granularity for activity definition impact a project? Note that for a project, for example, we can define drilling a well as a single activity, or many smaller activities. How can we develop more scalable approaches that address the needs of mega-projects and that can be applied to projects with many more activities (projects with a significantly more detailed level of granularity)? How can we incorporate risk into our models. These are a few of the many related questions that we hope to address in future research.

References

[1] Edward J. Anderson and Chris N. Potts. "Online Scheduling of a Single Machine to Minimize Total Weighted Completion Time". In: *Mathematics of Operations Research* 29.3 (2004), pp. 686–697. DOI: 10.1287/moor.1040.0092.

[2] Christian Artigues, Philippe Michelon, and Stéphane Reusser. "Insertion techniques for static and dynamic resource-constrained project scheduling". In: *European Journal of Operational Research* 149.2 (2003). Sequencing and Scheduling, pp. 249–267. ISSN: 0377-2217. DOI: 10.1016/S0377-2217(02)00758-0.

[3] Behzad Ashtiani, Roel Leus, and Mir-Bahador Aryanezhad. "New competitive results for the stochastic resource-constrained project scheduling problem: exploring the benefits of pre-processing". In: *Journal of Scheduling* 14.2 (2011), pp. 157–171. DOI: 10.1007/s10951-009-0143-7.

[4] Francisco Ballestién. "When it is worthwhile to work with the stochastic RCPSP?" In: *Journal of Scheduling* 10.3 (2007), pp. 153–166. DOI: 10.1007/s10951-007-0012-1.

[5] Philippe Baptiste and Sophie Demassey. "Tight LP bounds for resource constrained project scheduling". In: *OR Spectrum* 26.2 (2004), pp. 251–262. DOI: 10.1007/s00291-003-0155-1.

[6] Howard H. Bashford et al. "Implications of even flow production methodology for US housing industry". In: *Journal of Construction Engineering and Management* 129.3 (2003), pp. 330–337. DOI: 10.1061/(ASCE)0733-9364(2003)129:3(330).

[7] J. Christopher Beck. "Checking-Up on Branch-and-Check". In: *Principles and Practice of Constraint Programming – CP 2010*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 84–98. ISBN: 978-3-642-15396-9.

[8] Mario Brčić and Danijel Mlinarić. "Tracking Predictive Gantt Chart for Proactive Rescheduling in Stochastic Resource Constrained Project Scheduling". In: *Journal of Information and Organizational Sciences* 42.2 (2018), pp. 179–192. DOI: 10.31341/jios.42.2.2.

[9] Peter Brucker et al. "A branch and bound algorithm for the resource-constrained project scheduling problem". In: *European Journal of Operational Research* 107.2 (1998), pp. 272–288. ISSN: 0377-2217. DOI: 10.1016/S0377-2217(97)00335-4.

[10] Alvin F. Burkhart. "The use of SIPS as a productivity improvement tool". In: *Excellence in the Constructed Project*. ASCE. 1989, pp. 381–386. ISBN: 978-0-87262-740-6.

[11] Ripon K Chakrabortty, R Sarker, and D Essam. "Resource constrained project scheduling: A branch and cut approach". In: *Proceedings of the 45th international conference on computers and industrial engineering. Metz, France*. Vol. 132. 2015.

[12] Sriram Changali, Azam Mohammad, and Mark van Nieuwland. *The construction productivity imperative — McKinsey & Company*. 2015. URL: http://www.mckinsey.com/industries/capital-projects-and-infrastructure/our-insights/the-construction-productivity-imperative (visited on 12/18/2019).

[13] Shi Chen and Hau Lee. "Incentive Alignment and Coordination of Project Supply Chains". In: *Management Science* 63.4 (2017), pp. 1011–1025. DOI: 10.1287/mnsc.2015.2373.

[14] Zhi-Long Chen and Warren B. Powell. "Solving Parallel Machine Scheduling Problems by Column Generation". In: *INFORMS Journal on Computing* 11.1 (1999), pp. 78–94. DOI: 10.1287/ijoc.11.1.78.

[15] Zhi Chen et al. "Efficient priority rules for the stochastic resource-constrained project scheduling problem". In: *European Journal of Operational Research* 270.3 (2018), pp. 957–967. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2018.04.025.

[16] Nicos Christofides, R. Alvarez-Valdes, and J.M. Tamarit. "Project scheduling with resource constraints: A branch and bound approach". In: *European Journal of Operational Research* 29.3 (1987), pp. 262–273. ISSN: 0377-2217. DOI: 10.1016/0377-2217(87)90240-2.

[17] Peter F. Court. "Transforming traditional mechanical and electrical construction to a modern process of assembly". PhD thesis. Loughborough University, UK, May 2009. URL: https://repository.lboro.ac.uk/articles/Transforming_traditional_mechanical_and_electrical_construction_to_a_modern_process_of_assembly/9579350 (visited on 12/18/2019).

[18] Stefan Creemers. "Minimizing the expected makespan of a project with stochastic activity durations under resource constraints". In: *Journal of Scheduling* 18.3 (2015), pp. 263–273. DOI: 10.1007/s10951-015-0421-5.

[19] Morteza Davari and Erik Demeulemeester. "The proactive and reactive resource-constrained project scheduling problem". In: *Journal of Scheduling* 22.2 (2019), pp. 211–237. DOI: 10.1007/s10951-017-0553-x.

[20] Sophie Demassey, Christian Artigues, and Philippe Michelon. "Constraint-Propagation-Based Cutting Planes: An Application to the Resource-Constrained Project Scheduling Problem". In: *INFORMS Journal on Computing* 17.1 (2005), pp. 52–65. DOI: 10.1287/ijoc.1030.0043.

[21] Erik L Demeulemeester and Willy S Herroelen. *Project scheduling: a research handbook*. 1st ed. Vol. 49. Springer US, 2002. ISBN: 978-1-4020-7051-8. DOI: 10.1007/b101924.

[22] U. Dorndorf, E. Pesch, and T. Phan-Huy. “A branch-and-bound algorithm for the resource-constrained project scheduling problem”. In: *Mathematical Methods of Operations Research* 52.3 (2000), pp. 413–439. DOI: 10.1007/s001860000091.

[23] Doug Dunnebier et al. “Presentation: An Experiment in Takt Time”. In: *Proc. 16th Annual Lean Construction Congress* (San Francisco, CA, Oct. 7–10, 2014), pp. 1–26. URL: https://www.leanconstruction.org/media/docs/congress/2014/H5B_LCI2014%5C%20Presentation%5C%20Draft_Takt%5C%20Time.pdf (visited on 12/18/2019).

[24] Elfving and Tommelein. “Impact of multitasking and merge bias on procurement of complex equipment”. In: *Proceedings of the 2003 Winter Simulation Conference, 2003.* Vol. 2. 2003, 1527–1533 vol.2. DOI: 10.1109/WSC.2003.1261598.

[25] EY Research and Analysis. *Spotlight on oil and gas megaprojects; Oil and gas capital projects series*. URL: http://www.ey.com/Publication/vwLUAssets/EY-spotlight-on-oil-and-gas-megaprojects/%5C$FILE/EY-spotlight-on-oil-and-gas-megaprojects.pdf (visited on 12/18/2019).

[26] Bent Flyvbjerg. *The Oxford Handbook of Megaproject Management*. Oxford University Press, Sept. 2017. ISBN: 9780198732242.

[27] Bent Flyvbjerg, Nils Bruzelius, and Werner Rothengatter. *Megaprojects and risk: An anatomy of ambition*. Cambridge University Press, 2003. ISBN: 0521804205.

[28] Adam G. Frandson, Olli Seppänen, and Iris D. Tommelein. “Comparison between location based management and takt time planning”. In: *Proc. 23rd Annual Conference of the International Group for Lean Construction*. Perth, Australia, 2015, pp. 3–12. URL: http://www.iglc.net/papers/details/1181 (visited on 12/18/2019).

[29] Adam Frandson, Klas Berghede, and Iris D. Tommelein. “Takt time planning for construction of exterior cladding”. In: *Proc. 21st Annual Conference of the International Group for Lean Construction*. Fortaleza, Brazil, 2013, pp. 527–536. URL: http://www.iglc.net/papers/details/902 (visited on 12/18/2019).

[30] M. R. Garey and D. S. Johnson. “Complexity Results for Multiprocessor Scheduling under Resource Constraints”. In: *SIAM Journal on Computing* 4.4 (1975), pp. 397–411. DOI: 10.1137/0204035.

[31] Nicholas G Hall. “Project management: Recent developments and research opportunities”. In: *Journal of Systems Science and Systems Engineering* 21.2 (2012), pp. 129–143. DOI: 10.1007/s11518-012-5190-5.

[32] *Handbook on Project Management and Scheduling Vol.1*. Springer, 2015. ISBN: 978-3-319-05443-8.

[33] Robert B. Harris and Photios G. Ioannou. “Scheduling Projects with Repeating Activities”. In: *Journal of Construction Engineering and Management* 124.4 (1998), pp. 269–278. DOI: 10.1061/(ASCE)0733-9364(1998)124:4(269).

[34] Wallace J Hopp and Mark L Spearman. *Factory physics*. Waveland Press, 2011. ISBN: 978-1577667391.

[35] Oya Icmeli and Walter O. Rom. “Solving the resource constrained project scheduling problem with optimization subroutine library”. In: *Computers & Operations Research* 23.8 (1996), pp. 801–817. ISSN: 0305-0548. DOI: 10.1016/0305-0548(95)00074-7.

[36] G. Igelmund and F. J. Radermacher. “Preselective strategies for the optimization of stochastic project networks under resource constraints”. In: *Networks* 13.1 (1983), pp. 1–28. DOI: 10.1002/net.3230130102.

[37] Arman Jabbari and Philip Kaminsky. “Research Digest: Preliminary Investigations into Capital Projects Supply Chain Management”. In: *Journal of Project Production Management* 2 (2017), pp. 83–89. URL: https://projectproduction.org/journal/research-digest-preliminary-investigations-into-capital-projects-supply-chain-management/ (visited on 12/18/2019).

[38] Arman Jabbari, Iris D. Tommelein, and Philip M. Kaminsky. *Models and algorithm for workload leveling based on work space zoning to support takt planning*. Berkeley, CA: Technical Report, Project Production Systems Laboratory, UC Berkeley, Mar. 2020. DOI: https://doi.org/10.34942/P2159C.

[39] Arman Jabbari, Iris D. Tommelein, and Philip M. Kaminsky. “Workload Leveling based on Work Space Zoning for Takt planning”. In: *Automation in Construction* 000.0 (2020), pp. 000–000.

[40] Tsuyoshi Kawaguchi and Seiki Kyan. “Worst Case Bound of an LRF Schedule for the Mean Weighted Flow-Time Problem”. In: *SIAM Journal on Computing* 15.4 (1986), pp. 1119–1129. DOI: 10.1137/0215081.

[41] Russell Kenley and Olli Seppänen. *Location-based management for construction: Planning, scheduling and control*. 1st ed. London: Routledge, 2006. ISBN: 9780203030417. DOI: 10.4324/9780203030417.

[42] Mohammad Khalilzadeh, Fereydoon Kianfar, and Mohammad Ranjbar. “A Scatter Search Algorithm for the RCPSP with Discounted Weighted Earliness-Tardiness Costs”. In: *Life Science Journal* 8.2 (2011).

[43] Sanjeev Khanna, S. Muthukrishnan, and Mike Paterson. “On Approximating Rectangle Tiling and Packing”. In: *Proc. Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. San Francisco, California, USA: Society for Industrial and Applied Mathematics, 1998, pp. 384–393. ISBN: 0-89871-410-9. URL: http://dl.acm.org/citation.cfm?id=314613.314768 (visited on 12/18/2019).

[44] Robert Klein. “Project scheduling with time-varying resource constraints”. In: *International Journal of Production Research* 38.16 (2000), pp. 3937–3952. DOI: 10.1080/00207540050176094.

[45] Anton J. Kleywegt, Alexander Shapiro, and Tito Homem-de-Mello. "The Sample Average Approximation Method for Stochastic Discrete Optimization". In: *SIAM Journal on Optimization* 12.2 (2002), pp. 479–502. DOI: 10.1137/S1052623499363220.

[46] R Kolisch and R Padman. "An integrated survey of deterministic project scheduling". In: *Omega* 29.3 (2001), pp. 249–272. ISSN: 0305-0483. DOI: 10.1016/S0305-0483(00)00046-3.

[47] Rainer Kolisch and Sönke Hartmann. "Experimental investigation of heuristics for resource-constrained project scheduling: An update". In: *European Journal of Operational Research* 174.1 (2006), pp. 23–37. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2005.01.065.

[48] Stavros G. Kolliopoulos and George Steiner. "Approximation algorithms for scheduling problems with a modified total weighted tardiness objective". In: *Operations Research Letters* 35.5 (2007), pp. 685–692. ISSN: 0167-6377. DOI: 10.1016/j.orl.2006.12.002.

[49] Oumar Koné et al. "Event-based MILP models for resource-constrained project scheduling problems". In: *Computers & Operations Research* 38.1 (2011). Project Management and Scheduling, pp. 3–13. ISSN: 0305-0548. DOI: 10.1016/j.cor.2009.12.011.

[50] Philippe Laborie. *Solving Resource-Constrained Project Scheduling Problems with CP Optimizer*. 2019. URL: https://www.linkedin.com/pulse/solving-resource-constrained-project-scheduling-problems-laborie/ (visited on 12/18/2019).

[51] J. K Lenstra. *Sequencing by enumerative methods*. A revision of the author's thesis. Amsterdam : Mathematisch Centrum, 1977. ISBN: 9061961254.

[52] Olivier Liess and Philippe Michelon. "A Boolean satisfiability approach to the resource-constrained project scheduling problem". In: *Annals of Operations Research* 181.1 (2010), pp. 89–107. ISSN: 0377-2217. DOI: 10.1007/s10479-010-0693-2.

[53] Ming Liu et al. "Online scheduling to minimize modified total tardiness with an availability constraint". In: *Theoretical Computer Science* 410.47 (2009), pp. 5039–5046. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2009.07.055.

[54] Manuel J. [Pereira Lopes] and J.M. Valério [de Carvalho]. "A branch-and-price algorithm for scheduling parallel machines with sequence dependent setup times". In: *European Journal of Operational Research* 176.3 (2007), pp. 1508–1527. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2005.11.001.

[55] Edward W. Merrow. *Industrial Megaprojects: Concepts, Strategies, and Practices for Success*. 1st ed. John Wiley & Sons, 2011, p. 384. ISBN: 1118067509.

[56] Aristide Mingozzi and Serena Morigi. "Partitioning a matrix with non-guillotine cuts to minimize the maximum cost". In: *Discrete Applied Mathematics* 116.3 (2002), pp. 243–260. ISSN: 0166-218X. DOI: 10.1016/S0166-218X(00)00286-9.

[57] Rolf H. Möhring et al. “Solving Project Scheduling Problems by Minimum Cut Computations”. In: *Management Science* 49.3 (2003), pp. 330–350. DOI: 10.1287/mnsc.49.3.330.12737.

[58] Daniel Morillo-Torres, Luis Fernando Moreno-Velásquez, and Francisco Javier Díaz-Serna. “A branch and bound hybrid algorithm with four deterministic heuristics for the resource constrained project scheduling problem (RCPSP)”. In: *DYNA* 82 (Apr. 2015), pp. 198–207. ISSN: 0012-7353. URL: http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0012-73532015000200025&nrm=iso (visited on 12/18/2019).

[59] Klaus Neumann, Christoph Schwindt, and Jürgen Zimmermann. *Project Scheduling with Time Windows and Scarce Resources: Temporal and Resource-Constrained Project Scheduling with Regular and Nonregular Objective Functions*. 2nd ed. Springer-Verlag Berlin Heidelberg, 2003. ISBN: 9783540401254. DOI: 10.1007/978-3-540-24800-2.

[60] Katarzyna Paluch. “A $2\frac{1}{8}$-Approximation Algorithm for Rectangle Tiling”. In: *Automata, Languages and Programming*. Springer Berlin Heidelberg, 2004, pp. 1054–1065. ISBN: 978-3-540-27836-8. DOI: 10.1007/978-3-540-27836-8_88.

[61] Katarzyna Paluch. “A New Approximation Algorithm for Multidimensional Rectangle Tiling”. In: *Algorithms and Computation*. Berlin, Heidelberg: Springer, 2006, pp. 712–721. ISBN: 978-3-540-49696-0.

[62] Cyril N Parkinson and Robert C Osborn. *Parkinson’s law, and other studies in administration*. Vol. 24. Houghton Mifflin Boston, 1957.

[63] Radivoj Petrović. “Optimization of Resource Allocation in Project Planning”. In: *Operations Research* 16.3 (1968), pp. 559–568. DOI: 10.1287/opre.16.3.559.

[64] A Alan B Pritsker and Lawrence J Watters. *A Zero-One Programming Approach to Scheduling with Limited Resources*. Tech. rep. Santa Monica, Calif.: RAND Corporation, RM-5561-PR, Apr. 1968. URL: https://www.rand.org/pubs/research_memoranda/RM5561.html (visited on 12/18/2019).

[65] Kirk Pruhs, Jiıé Sgall, and Eric Torng. “Online Scheduling”. In: *Handbook of Scheduling - Algorithms, Models, and Performance Analysis*. Chapman and Hall/CRC, 2004. DOI: 10.1201/9780203489802.

[66] FJ Radermacher. “Cost-dependent essential systems of ES-strategies for stochastic scheduling problems”. In: *Methods of Operations Research* 42 (1981), pp. 17–31.

[67] S Rostami. *New models and methods for sequencing and project scheduling*. 2019. URL: https://lirias.kuleuven.be/retrieve/547784 (visited on 12/18/2019).

[68] Alexander Shapiro and Andy Philpott. *A tutorial on stochastic programming*. Atlanta, GA: Technical Report, Georgia Institute of Technology, 2007. URL: https://www2.isye.gatech.edu/people/faculty/Alex_Shapiro/TutorialSP.pdf (visited on 12/18/2019).

[69] Daniel D. Sleator and Robert E. Tarjan. "Amortized Efficiency of List Update and Paging Rules". In: *Commun. ACM* 28.2 (Feb. 1985), pp. 202–208. ISSN: 0001-0782. DOI: 10.1145/2786.2793.

[70] Wayne E. Smith. "Various optimizers for single-stage production". In: *Naval Research Logistics Quarterly* 3.1-2 (1956), pp. 59–66. DOI: 10.1002/nav.3800030106.

[71] Alexander Tesch. "Compact MIP Models for the Resource-Constrained Project Scheduling Problem". MA thesis. Technische Universität Berlin, 2015. URL: urn:nbn:de:0297-zib-60208 (visited on 12/18/2019).

[72] Walid Y. Thabet and Yvan J. Beliveau. "HVLS: Horizontal and vertical logic scheduling for multistory projects". In: *Journal of Construction Engineering and Management* 120.4 (1994), pp. 875–892. DOI: 10.1061/(ASCE)0733-9364(1994)120:4(875).

[73] Walid Y. Thabet and Yvan J. Beliveau. "SCaRC: Space-Constrained and Resource-Constrained scheduling system". In: *Journal of Computing in Civil Engineering* 11.1 (1997), pp. 48–59. DOI: 10.1061/(ASCE)0887-3801(1997)11:1(48).

[74] Erlendur S. Thorsteinsson. "Branch-and-Check: A Hybrid Framework Integrating Mixed Integer Programming and Constraint Logic Programming". In: *Principles and Practice of Constraint Programming — CP 2001*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 16–30. ISBN: 978-3-540-45578-3.

[75] Iris D. Tommelein. "Collaborative takt time planning of non-repetitive work". In: *Proc. 25th Annual Conference of the International Group for Lean Construction*. Heraklion, Greece, 2017, pp. 745–752. DOI: 10.24928/2017/0271.

[76] Iris D. Tommelein. *Presentation: Collaborative takt time planning of non-Repetitive work*. July 2017. URL: https://www.youtube.com/watch?v=500y23NrNms%5C&t=604s (visited on 12/18/2019).

[77] Iris D. Tommelein and Glenn Ballard. *P2SL Lean Construction Glossary*. 2.9. Berkeley, California: Project Production Systems Laboratory (P2SL), 2019, p. 72. DOI: 10.34942/P2WC7F.

[78] Iris D. Tommelein, David R. Riley, and Greg A. Howell. "Parade game: Impact of work flow variability on trade performance". In: *Journal of Construction Engineering and Management* 125.5 (1999), pp. 304–310. DOI: 10.1061/(ASCE)0733-9364(1999)125:5(304).

[79] Cynthia C. Y. Tsao et al. "Work Structuring to Achieve Integrated Product–Process Design". In: *Journal of Construction Engineering and Management* 130.6 (2004), pp. 780–789. DOI: 10.1061/(ASCE)0733-9364(2004)130:6(780).

[80] Vicente Valls, Francisco Ballestıén, and Sacramento Quintanilla. "Justification and RCPSP: A technique that pays". In: *European Journal of Operational Research* 165.2 (2005). Project Management and Scheduling, pp. 375–386. ISSN: 0377-2217. DOI: 10.1016/j.ejor.2004.04.008.

[81] Stijn Van de Vonder et al. “Proactive-Reactive Project Scheduling Trade-Offs and Procedures”. In: *Perspectives in Modern Project Scheduling*. Boston, MA: Springer US, 2006, pp. 25–51. ISBN: 978-0-387-33768-5. DOI: 10.1007/978-0-387-33768-5_2.

[82] A.P.A. Vestjens. “On-line machine scheduling”. PhD thesis. Eindhoven: Department of Mathematics and Computer Science, 1997. ISBN: 90-386-0571-4. DOI: 10.6100/IR500043.

[83] Carol Willis and Donald Friedman. *Building the Empire State*. 1st ed. W. W. Norton & Company, 1998. ISBN: 978-0393732313.

[84] Pierrette P. Zouein and Iris D. Tommelein. “Improvement algorithm for limited space scheduling”. In: *Journal of Construction Engineering and Management* 127.2 (2001), pp. 116–124. DOI: 10.1061/(ASCE)0733-9364(2001)127:2(116).

Appendix A

Interviews

Below, we summarize responses to our interview questions. Note that this is not intended to be a verbatim transcription of interview responses. Rather, we present sense of typical or particularly interesting responses to our questions. We divide these into three categories. The first relates to the efficiency and reliability of the tools that are used to manage projects, the second focuses on the impact of those tools and the resulting decisions on suppliers, and the last concerns the ways decision get made.

A.1 Balancing Efficiency and Reliability

Question: *What tools and techniques do you utilize for analyzing the impacts of delay on overall project performance?*

Answers: Companies commonly use scheduling tools (e.g. Primavera, MS Project) and develop estimating, productivity, and economic models using these tools to enable them to quantify the impact of delays, and that prior to sanctioning a project, most businesses (owner and contractor) will also have prepared schedule and cost models, cash flow curves, and detailed staffing plans. Some interviewees mentioned that project planning and scheduling techniques came from the Project Planning and Estimating department (or the equivalent), and that the contracting and procurement schedule is developed based on that schedule.

Question: *How reliable are those tools and techniques? Can you describe projects where these tools and techniques have improved project reliability?*

Answers: We meets 50% of our project schedules, however, my own view is that the schedules that are met are not world class and tend to be longer than the competition.

This suggests that although companies set their targets according to a baseline derived from past experience, they fail to meet those targets 50% of the time. Another interviewee told us:

In my experience, the tools themselves are generally reliable. My confidence in their output increases with the skills and experience of the planners, estimators and project managers who develop and utilize the tools.

This suggests that interviewees do not believe that tools are a cause of delays, although there doesn't seem to be evidence to support this point.

Question: *Do you consider the cost of project acceleration as well as the cost of delay?*

Answer: Yes. Both have a significant impact on revenue and project profitability. While we can try to accelerate projects, we focus on avoiding delay. We work to frame the project comprehensively *(it isn't clear to us precisely what this means)* and use standard designs and standard equipment packages to minimize delays in engineering and construction. In the oil and gas industry, there is a strong correlation in the slip to achieving "first oil" to the slip in completing engineering. The average slippage is 6 months and 25% of industry projects slip more than 10 months which directly impacts NPV.

Question: *Does excessive storing, staging, and moving increase the likelihood of defective or unsuitable components? By how much? What does this practice cost? What is the impact of this on project performance?*

Answer: Suppliers will deliver materials and equipment to staging areas as defined in the contract or purchase order. The storage and handling are often haphazard (i.e., store as delivered - not as required). This can cause excessive movement and increase the likelihood of damaged, defective or unsuitable materials and equipment (components). However, we do not track this data yet. This amount is considered to be negligible on multi-billion dollar projects, but it certainly impacts the project team and contractors.

Question: *What are the costs and other potential problems and negative impacts of mandating that parts and materials are all delivered far ahead of when they are needed, or even before the project starts?*

Answer: This results in large laydown yards, expensive handling, and double handling plus large leftover surplus after project completion. Storage of complex equipment also requires sophisticated storage facilities. In some projects where this has not been done well, it has required the suppliers to come to the site for rework.

Question: *What extra steps are necessary to preserve parts and components during long storage? What do these steps cost?*

Answer: Depending on the type of parts and components this can vary, for instance:

- Electrical measurement and instrumentation, process instrumentation, select CAO (computer assisted operations) and similar sensitive parts and equipment must be stored in air-conditioned and sealed rooms or mini-warehouses to avoid corrosion and irreparable damage.
- Rotating equipment (i.e., compressors, turbines, pumps, motors, etc.) must not sit idle and requires workers to turn the shafts to avoid flat spots on the crankshafts or windings.
- Specialty valves and piping must have protective covers on the open flanges, pin and box ends etc. to prevent corrosion, rodents or insects, etc. from getting inside – these covers can deteriorate over time.

- Some rotating equipment may need to be run periodically to ensure that it will operate when required and some may require lubricant changes depending on the temperature variation it may go through while in temporary storage.

- These items must be cataloged and have a preventative maintenance schedule defined and executed with trained workers.

- Segregating materials based on chemical composition (e.g., carbon steel from stainless steel, specialty alloys, duplex, chrome, etc.).

  These steps would typically cost 1-2 full-time equivalent technicians in addition to the material handlers (depending on the number and complexity of equipment types).

Question: *How does this impact financial measures? Quality? Rework?*
Answer: This impacts financial metrics by adding costs for the staging and preservation as well as potential repair costs for damaged materials and equipment.

Rework due to construction defects can be extremely costly, and the cost increases exponentially depending on where and when these defects are remedied. For example, weld failures are often discovered during the weld x-ray or hydro-testing phase. If they are remedied on the ground the cost is rework, if they are remedied after installation on the deck 300' in the air at quay-side the cost triples and if they are remedied offshore the cost triple again. This has significant impact on the financial performance of the project, particularly if the installation cannot be commissioned on time. Delay in starting up a facility will have a significant negative impact on project NPV.

Question: *Given that the goal is to ensure (with very high probability) that parts and material availability do not delay work, is this practice [requiring delivery far ahead of time] the best way to ensure this end result? Is it possible that current practices lead to more delays than alternative practices? To put this another way, if we graph system costs versus the likelihood that material and part availability lead to delays, one would assume that if the system is run as efficiently as possible, the likelihood of delays increases with decreasing system costs. Is it true that capital projects operate on or near this so-called "efficient frontier"?*

Answer: There is probably a smarter way to do it to really understand the economic and the trade-off of the linear process versus maybe some overlap and flexibility.

Question: *What are the costs and other potential problems associated with parts and material delays? How much flexibility is there typically built into the system to absorb these delays? Is the cost of delays more or less linear with the delay time, or is the relationship more complex?*

Answer: The cost of delay is not linear with the delay time – it can be exponentially increasing if we have other contractors engaged. For example, if the hull and topsides are scheduled for sail away and installation on a certain date, but it is delayed – and we have contracted for a crane vessel or deep water construction vessel then the cost is committed and their availability may also be constrained – so we may have to wait until it become available again (we experience stranded cost for the crane, contracting for the crane again, lost revenue, etc.). The cost to perform work offshore is significantly higher than performing the same work in the construction yard.

Question: *What likelihood of delay is acceptable? For instance, is a 1% chance of delay acceptable for a 20% decrease in costs? Is this a tradeoff that the industry considers?*

Answer: The industry focuses predominantly on safety first and some companies focus on the quality second. Development teams have team members who manage delays via expediting, rushed delivery (hot shot transportation), etc. and use contingency to cover the cost impact. There is rarely an attempt to quantify (in advance) the tradeoff between lower pricing and schedule – we typically set the delivery time and expect the contractor to simply 'make it happen'. Delays are never acceptable but we understand that they will happen. We try to work with our contractors to mitigate those delays, through adjusting the construction sequence, installation sequence, etc. This is where having the right relationship with our preferred contractors will help with mitigating schedule delays and potentially cost impacts.

A.2 Impact on Suppliers

Note that all of our interviewees are project managers and consultants. In the future, we intend to interview suppliers to get a sense of their views on these issues.

Question: *How are suppliers impacted by mandates that parts and materials be delivered far in advance of when they are needed, and often before the project starts? How does this impact quality of parts? The amount of required rework?*

Answer: I do not see a frequent impact to the suppliers on the scheduled demanded by us. Goods are usually delivered in the suppliers estimated delivery times. However very frequently goods are delivered far ahead of the required use for construction to ensure the construction activities are not held up in any way by late material or equipment. In general, if suppliers have sufficient time to bid, plan and execute their work, the correlation between their delivery date and the project start date shouldn't be an issue. Suppliers will take into consideration their overall workload, resources required to execute, etc. Problems arise when there is either insufficient lead time or (and most commonly) when design changes require reworking on something that's already in the manufacturing cycle or complete.

Question: *What challenges have suppliers communicated to you?*

Answer: Suppliers have communicated that they experience problems with engineering design changes, equipment specification changes, third party quality inspector preferences and differences of opinion on requirements documents. They shared that would prefer clear communication (straight talk) from the ultimate user (e.g., fabrication or construction contractor) to eliminate confusion and rework. Now there is much more open conversation about manufacturing schedules, long-lead ordering, workforce, impacts. And the suppliers say if you want us to get better, if you want us to be able to have have what you need when you needed, involve us early enough in your planning cycles so that we can respond and plan practically.

Question: *How do engineering changes affect suppliers? How often do these happen?*

Answer: Engineering changes (and changes in requirements) affect a variety of suppliers, and this happens many times in a project. This can be the result of regulatory requirements, operational learnings, functional acceptance testing results, etc. Suppliers often have to make mid-course adjustments or corrections as a result of these changes. This can then manifest itself in materials and equipment order changes to meet the changed requirements (often delaying delivery or adding cost), specialty materials may have to be manufactured and then fabricated, etc. Engineering changes occur a number of times per project and are frustrating to the supplier. This is probably the biggest contributor to cost and schedule growth.

There are many reasons. You probably rebaseline the schedule a couple of times a year but change on a project is inevitable. You'll have design development or as you progress on the engineering you have to make changes to the manufacturing or you find quality issues with the equipment or you have a contractor who falls behind. There are a number of reasons and things that introduce change on projects. So, you have to leave a reasonable amount of flexibility to accommodate that risks.

Question: *What do suppliers do to guard against changing delivery dates? How does this impact their costs and the prices they charge?*

Answer: Delivery dates are not frequently changed due to construction requirements with the exception of some bulk materials. A more frequent change of delivery occurs due to late changes by the client or EPC contractor and the cost can increase significantly, on some equipment even greater than a 100% increase.

Two things: First, most suppliers expect changes to occur. To mitigate their exposure, they will commonly include cost and schedule contingency in their quotes. For example, in some deep-water offshore projects, 48 month lead times have been quoted for manufacturing sub-sea trees where the manufacturing from raw materials to delivery takes 18 months with no contingencies. Second, prudent suppliers will also address this in their contracts. They will request contractual relief by their customer (e.g, cost, schedule and perhaps other terms – Liquidated damage/ Bonus, warranty, etc). if the client modifies the agreed upon delivery date for reasons not attributable to the supplier.

Question: *Have you ever mapped the value stream between your firm and your suppliers? What have you observed?*

Answer: We have created value stream maps with a number of suppliers and discovered a tremendous amount of waste, rework and non-value added work. We have been able to identify and eliminate more than 60% of the cycle time and reduce rework from 9 to 10 times to less than 1 time. Motion and repetitive checking are often confused with value added time. We have not reached critical mass, but this is a huge area for savings.

A.3 Decision-Making

Question: *How do you or your firm divide a project into components or subprojects?*

Answer: It mostly depends on the company and the culture of that company, the project, and its complexity. But it is not uncommon for a large mega-project to have several different areas or components and to have a project director who is ultimately responsible for the whole of the project and the area or subproject managers who are accountable for sub-area within the project. Ultimately, there will be an integrated plan and schedule for the whole project and the project director is accountable for that.

Question: *Which decisions are made by managers overseeing the entire project, and which are made by the managers of the subprojects?*

Answer: Assuming the project is lump sum turnkey the EPC contractor is making the decisions related to delivery dates and construction schedules with the EPC Project Manager delegating to his team leads. I think your questions is getting at how well is their decision making coordinated on the project. I think we have a wide range of experience on this, but strive for strong coordination across the decision making to ensure no one manager is making decisions which impact the schedule of others (not sure they do this as well as they would like).

These divisions of responsibility should be clearly established before the project commences and can vary depending on company culture and the project type and risk. The project director has overall accountability for execution. If he/she directs a change that impacts a sub-area, that subproject managers duty is to advise on the impact and the risk and to determine the resources (manpower, cost, equipment, schedule) required to meet the project manager's direction.

The subproject or components managers make decisions based on their deliverables. The overall project manager manages key stakeholders, partners, and contractors and makes decisions for the white-space (i.e., between the subprojects or components) or for cross-project issues (e.g., safety performance, costs tradeoffs, etc.)

Note that we were unable to get at the issue of what types of decisions need to be coordinated across sub-projects.

Question: *How do higher level managers overseeing the entire project communicate with subprojects managers and how is feedback transferred?*

Answer: We have a routine 'operating rhythm' for all projects, typically weekly for progress with a seven-day look-a-head and monthly focusing on overall progress, reviewing the risk register, and performing a 90-day look-a-head for progress against milestones. These meetings occur in an Obeya room where each subproject or component leader participates and is encouraged to highlight the gaps and what they need help with to ensure we meet Safety, Quality, Delivery and Cost (SQDC) targets.

We have weekly meetings in an Obeya room where each component leader provides updates on Safety, Quality, Delivery and Cost with a focus on hotspots that require help from other teams and highlights. The Obeya room is refreshed with measures, photographs, and visual controls to help the entire team assess project status and risks.

Question: *What overall project management strategies are communicated to subproject managers?*

Answer: We have something called work breakdown structures (WBS). You can divide the project into different areas based on your WBS and then you look at your equipment deliveries and your construction sequencing and you analyze which equipment needs to be engineered or procured soonest based on the time requirements for manufacture and delivery and sequencing of your construction. So, you need to get foundations in and major equipment in. That's all driven by WBS and the project manager would be an approver of that WBS.

Question: *How procurement/supply chain decisions get made? How do the various levels of project management affect those decisions?*

Answer: Each project has a Construction and Planning (C&P) lead on the PM leadership team who is heavily involved in working with the estimating and scheduling team to ensure delivery dates match the required schedule. The C&P team will get pressure from each of the sub-managers to ensure their material and equipment is delivered to the schedule or earlier. Naturally C&P will negotiate with the suppliers on schedule but usually, the relevant discipline manager is also present in those meetings.

This is heavily dependent on the company and project culture. My bias is to have a project structure in which the procurement lead reports directly to the project manager (most sophisticated firms do this). In addition, a good practice is to have a governance model that includes two types of approval:procurement and financial. In this model, before any commercial commitment can be made (eg. a supplier contract), the procurement manager must first concur that it is commercially acceptable. Only someone with procurement authority can do this. Once that happens, the individual with financial authority (i.e. the project manager) is authorized to enter into the contractual arrangement. The further removed supply chain is from the project leadership, the less likely this is to happen and the weaker their voice will be in commercial matters. This could lead to bias in bidders lists and award decisions and suboptimal management of the resulting contract.

The "Procurement Category Manager" works closely with the subproject or components managers to define the strategy and to collaborate with the key suppliers to create a team environment and clearly define expectations and performance metrics.

Question: *How are procurement/supply chain delivery time due dates set? Who sets them? How is the amount of float determined?*

Answer: That depends on the type of equipment and how much confidence you have in manufacturing. If you have a big offshore installation vessel you want to have a little bit more float in your schedule to make sure you are not paying a lot of money for a vessel to sit there and wait. It also depends on the weather. If you want to store a material in a very unfriendly environment like snow, you may not want that material to sit out at the site for a long time because you have to store and preserve it. So, there is lots of scenario planning that has to go into the different type of projects, the different type of material, the different contracting strategies, etc., to really determined how much float you build in your shipping schedule. Generally, you want some float into your schedule. You want to build about 60 days of a float or to have your material be there 60 days prior to when they are needed.

Other interviewees gave slightly different answers us:

We need to make sure the packages are available on site, so we have some float. Both the operator paying for construction of the facility as well as construction company typically want the material on site 30-60 days in advance. But it's not very scientific; it is more of an industry rule of thumb.

Every project based on the design of the project and based upon the economics merits of the project, the organization, the project team, the supply chain team, together with PNL will take a look at what's needed to get the project completed. If there is equipment with a long lead-time, it doesn't necessary mean that it will be ordering 4 month or 6 months or 8 months early. It really depends on what's needed and when and whether or not the team believes the sufficient capacity the market to get what is needed in terms of equipment and materials. In some cases, there will be a determination made to order certain equipment and materials that are in the critical path or have the ability to affect the project schedule be on site early. How far in advance really depends on the project's need, the project's requirement and the supplier's ability to get the material onsite when the teams need to.

Question: *Do you calculate these floats using some sort of models?*

Answer: In our scheduling tools, we have different calendars and scenarios. We don't really have models other than different calendars, and we don't have simulation capability. And then you need to understand what's the volume of material you're going to store. Does it need to be held in a climate control warehouse? Does it need to be secure? Can you store it in a less expensive open lay-down area? How much will it cost to store it? How much will it cost to preserve it? What's your consumption rate on the site? There are many variables that you need to take into consideration.

This is based upon the project plan, the project schedule, the risk and contingency associated with that project, and whether or not again specific item of equipment or materials is or could be on the market during that time. In the company where I work, ultimately the project manager makes the decisions to authorize if equipment or material needs to be released early.

Question: *Who decides on shipping times/lead times/etc. for subprojects? Who is responsible for delay and cost overrun on subproject? Who balances this trade-off between subprojects?*

Answer: In our company, the "Procurement Category Manager" works closely with the subproject leaders to define shipping times, lead times, delivery sequencing, quality inspections etc. for each subproject.

The project director is accountable for the overall project including safety, quality, delivery and cost including any delays or cost overruns. The project director also balances this trade-off between subprojects.

Question: *If certain materials and parts are requested further in advanced than others, what governs these decisions. Are the service levels, i.e., the ability to deliver within agreed lead time, of each supplier being measured and incorporated into how far in advance it is required? Or is it more so based on the criticality of the materials and parts?*

Answer: Long lead time items are specified and AFE'd separately and must be approved by executive leadership. These items (materials and equipment) are defined by the project team together with procurement. The decision drivers are the criticality of the materials and equipment, the manufacturing lead time required to assure industry leading acceptable quality.

Question: *Does your scheduling software consider all of these cost and factors? How do you link your schedules with all the cost analysis that you have in Excel? How do you consider cost in your schedules?*

Answer: No, you have to build many cost models in Excel or something outside of the scheduling software. You would have to estimate the cost and everything outside of the scheduling software and make the decision. For example, it's going to cost me so many dollars per month, I can consume so much per week onsite and you have to do the calculation and then build the float. The only thing that the scheduling software is really useful for is time, and sequencing and scheduling. It's not a costing tool. As far as I have ever been experienced, you kind of have to do some work outside of the scheduling software like Excel or a simulation modeling tool, and then use that data in the scheduling software.

Question: *What is the reason your company does not use something closer to Just in Time delivery (JIT)?*

Answer: Because it never works. It's a fallacy. Because there are always changes in construction projects. The manufacturers usually are late. There are changes to the schedule. There are changes to the sequencing of constructions in the field and if you rely on something to be delivered just in time to install it most of the time it does not work out that way and it costs you a lot of money to either pay the manufacturer to hold the material or to store and preserve the material on-site. That's why in the construction industry, they like to build up work in progress (WIP) and have an inventory of materials prior to mobilizing the field to make that the labor force can be utilized productively and efficiently.

I think in theory it could work. Unfortunately, what you have in this industry is less transparency between the owner organization or the customer and the supplier organization or contractor, because both are profit-driven enterprises. In order to have the the best execution, you need to have full transparency. For example, companies in this industry do not even share their drilling scheduling with their partner.

Question:*Do you think that you have a problem with early delivery from suppliers? Have you seen materials that are are on the site far sooner than the time they are needed?*

Answer: Generally speaking, I would say no. There are some times when you have equipment there long before you need it because of the aggressive schedule that we have in the industry these days. I think that often times you don't have the material on time, you don't have JIT delivery and that's why you have to have so many work-arounds in the field to accommodate construction based on the equipment that is available. So, I think it would be helpful to have a more reasonable schedule where you have the time to do the engineering and to do the equipment and to have more of it available to support the construction. From what I see, often times we are waiting on materials, more often that having it there just waiting for installation.

Question: *What is the most uncertainty that you have (among supplier manufacturing, shipping duration, construction or installation?*

Answer: The shipping is fairly consistent. There is not as much variation in the shipping times. I would say the manufacturing time or the time required to do the installation on the site depending upon if the construction productivity is good.

Question: *Is it efficient that you ask delivering 30-60 days prior to ROS? Do you think that procurement works efficiently when deciding float times?*

Answer: *Interviewees gave a variety of very different answers:*

Not at all. We are working to change all that as a company. Ultimately, we want to do be like the manufacturing world where delivery is JIT.

If we delay a project, that has far greater cost than sitting equipment for 30 days. To be clear, our company plans and schedules as much material and equipment as it can get to the site ahead of starting construction or any activity.

I do think that the systems are fairly efficient. There are complex models that are used to develop and sanction projects. So, I don't think that it's inefficient. I am not sure that there is a better way. I know that there are many criteria that go into the various stages of developing and sanctioning projects at least on an operating site. I don't think it is the question is it efficient or not, for me it really goes to what are the trade-off between what you need on site and when and whether the project or site prepared to accept the goods or services or materials and if they can arrive at site and what kind of constraints do you put on supply chain to make sure that will happen.

Question: *Why do others think that current systems are working efficiently?*

Answer: The traditional approach to contracting for big projects is EPC and all the risk is on the contractor, which is absolutely not the most efficient way to do things. IPA collects data on all capital projects including oil and gas. The majority of capital projects are delivered with delay and over budget which shows they are not efficient. There is certainly a huge opportunity. We are working on how to reduce the cycle time. We need to understand how supply chain management, material handling, contracting, and so on contribute on that.

Appendix B

Details of Running the WoLZo Algorithm

This Appendix B presents the details of using the WoLZo algorithm, including the run-times and the number of iterations, for both Model R and Model L for every scenario while varying Z from 1 to 15. In the following tables, $alpha$ is selected to be 1.05.

We used the following notations:

Z: Number of zones
$T(Z)$: Minimum possible takt (units of time per zone)
$\lceil T(Z) \rceil$: Smallest integer larger than $T(Z)$
$iter$: Number of iterations of the WoLZo algorithm
t: Run-time of the WoLZo algorithm (seconds)
D_1: Phase duration when there is only 1 floor
D_2: Phase duration when there are 2 floors
D_5: Phase duration when there are 5 floors
D_{10}: Phase duration when there are 10 floors
T_G: Solution time when using Gurobi alone (bounded by 3,600 seconds)
gap_G: Optimality gap provided by Gurobi ((upper bound - lower bound) / upper bound)

Table B.1: Results of WoLZo Computation for Model R, Scenario 1

Z	$T(Z)$	$\lceil T(Z) \rceil$	$iter$	t	D_1	D_2	D_5	D_{10}	T_G	gap_G
1	70.1	71	1	0.0	284	355	568	923	0.0	0.00 %

Table B.2: Results of WoLZo Computation for Model L, Scenario 1

Z	$T(Z)$	$\lceil T(Z) \rceil$	$iter$	t	D_1	D_2	D_5	D_{10}	T_G	gap_G
1	70.1	71	1	0.0	284	355	568	923	0.0	0.00 %

Table B.3: Results of WoLZo Computation for Model R, Scenario 2

Z	$T(Z)$	$\lceil T(Z)\rceil$	$iter$	t	D_1	D_2	D_5	D_{10}	T_G	gap_G
1	70.1	71	1	0.0	284	355	568	923	0.0	0.00%
2	40.7	41	4	0.0	205	287	533	943	0.0	0.00%
3	31.3	32	6	0.0	192	288	576	1056	0.0	0.00%
4	22.0	22	5	0.0	154	242	506	946	0.0	0.00%
5	20.8	21	9	0.0	168	273	588	1113	0.0	0.00%
6	20.8	21	12	0.0	189	315	693	1323	0.0	0.00%
7	20.8	21	15	0.0	210	357	798	1533	0.0	0.00%
8	20.8	21	18	0.0	231	399	903	1743	0.0	0.00%
9	20.8	21	21	0.0	252	441	1008	1953	0.0	0.00%

Table B.4: Results of WoLZo Computation for Model L, Scenario 2

Z	$T(Z)$	$\lceil T(Z)\rceil$	$iter$	t	D_1	D_2	D_5	D_{10}	T_G	gap_G
1	70.1	71	1	0.3	284	355	568	923	0.3	0.00%
2	39.3	40	3	0.3	200	280	520	920	0.2	0.00%
3	27.7	28	4	0.3	168	252	504	924	0.3	0.00%
4	22.0	22	5	0.3	154	242	506	946	0.3	0.00%
5	20.8	21	9	0.3	168	273	588	1113	0.3	0.00%
6	20.8	21	12	0.3	189	315	693	1323	0.2	0.00%
7	20.8	21	15	0.4	210	357	798	1533	0.3	0.00%
8	20.8	21	18	0.4	231	399	903	1743	0.2	0.00%
9	20.8	21	21	0.4	252	441	1008	1953	0.2	0.00%

Table B.5: Results of WoLZo Computation for Model R, Scenario 3

Z	$T(Z)$	$\lceil T(Z) \rceil$	$iter$	t	D_1	D_2	D_5	D_{10}	T_G	gap_G
1	70.1	71	1	0.1	284	355	568	923	0.1	0.00%
2	40.7	41	4	0.2	205	287	533	943	0.2	0.00%
3	26.4	27	3	0.2	162	243	486	891	0.4	0.00%
4	20.9	21	4	0.2	147	231	483	903	0.6	0.00%
5	16.8	17	4	0.2	136	221	476	901	1.0	0.00%
6	15.2	16	6	0.3	144	240	528	1008	1.9	0.00%
7	13.7	14	7	0.5	140	238	532	1022	1.7	0.00%
8	11.2	12	6	0.3	132	228	516	996	1.8	0.00%
9	9.4	10	4	0.2	120	210	480	930	1.6	0.00%
10	8.3	9	4	0.3	117	207	477	927	1.9	0.00%
11	8.2	9	6	0.3	126	225	522	1017	1.7	0.00%
12	8.2	9	7	0.4	135	243	567	1107	1.8	0.00%
13	8.2	9	9	0.4	144	261	612	1197	1.8	0.00%
14	8.2	9	11	0.6	153	279	657	1287	0.8	0.00%
15	8.2	9	12	0.5	162	297	702	1377	1.6	0.00%

Table B.6: Results of WoLZo Computation for Model L, Scenario 3

Z	$T(Z)$	$\lceil T(Z) \rceil$	$iter$	t	D_1	D_2	D_5	D_{10}	T_G	gap_G
1	70.1	71	1	22.1	284	355	568	923	28.8	0.00%
2	35.2	36	1	30.4	180	252	468	828	3594.4	0.00%
3	23.5	24	1	25.9	144	216	432	792	3600.0	100.00%
4	18.4	19	1	23.7	133	209	437	817	3600.0	100.00%
5	14.6	15	1	22.9	120	195	420	795	3600.0	inf
6	12.3	13	2	23.6	117	195	429	819	3600.0	99.20%
7	11.0	11	2	23.6	110	187	418	803	3600.0	100.00%
8	9.7	10	3	25.4	110	190	430	830	2683.8	0.00%
9	8.6	9	3	24.3	108	189	432	837	3600.0	inf
10	8.2	9	4	25.2	117	207	477	927	3600.0	inf
11	8.2	9	6	26.5	126	225	522	1017	3600.0	100.00%
12	8.2	9	7	28.2	135	243	567	1107	1528.0	100.00%
13	8.2	9	9	28.9	144	261	612	1197	3600.0	99.40%
14	8.2	9	11	31.6	153	279	657	1287	3183.0	0.00%
15	8.2	9	12	31.7	162	297	702	1377	2774.5	0.00%

Table B.7: Results of WoLZo Computation for Model R, Scenario 4

Z	$T(Z)$	$\lceil T(Z) \rceil$	$iter$	t	D_1	D_2	D_5	D_{10}	T_G	gap_G
1	70.1	71	1	0.5	284	355	568	923	0.5	0.00 %
2	35.9	36	1	0.6	180	252	468	828	0.8	0.00 %
3	25.7	26	2	0.6	156	234	468	858	5.2	0.00 %
4	18.4	19	1	0.6	133	209	437	817	9.5	0.00 %
5	15.2	16	2	1.1	128	208	448	848	19.2	0.00 %
6	13.0	13	3	0.8	117	195	429	819	34.0	0.00 %
7	12.1	13	4	2.2	130	221	494	949	26.1	0.00 %
8	10.9	11	5	1.9	121	209	473	913	35.6	0.00 %
9	9.4	10	4	1.5	120	210	480	930	33.0	0.00 %
10	8.3	9	4	1.6	117	207	477	927	36.9	0.00 %
11	8.0	8	5	1.4	112	200	464	904	35.5	0.00 %
12	7.5	8	6	1.6	120	216	504	984	28.9	0.00 %
13	7.2	8	6	1.6	128	232	544	1064	32.0	0.00 %
14	6.8	7	7	1.7	119	217	511	1001	38.4	0.00 %
15	6.6	7	7	1.6	126	231	546	1071	34.4	0.00 %

Table B.8: Results of WoLZo Computation for Model L, Scenario 4

Z	$T(Z)$	$\lceil T(Z) \rceil$	$iter$	t	D_1	D_2	D_5	D_{10}	T_G	gap_G
1	70.1	71	1	170.5	284	355	568	923	191.5	0.00 %
2	35.1	36	1	301.5	180	252	468	828	3600.0	100.00 %
3	23.5	24	1	236.1	144	216	432	792	3600.0	100.00 %
4	17.9	18	1	200.3	126	198	414	774	3600.0	100.00 %
5	14.3	15	1	195.5	120	195	420	795	3600.0	inf
6	12.1	13	1	185.1	117	195	429	819	3600.0	inf
7	10.5	11	1	175.0	110	187	418	803	3600.0	inf
8	9.3	10	2	215.7	110	190	430	830	3600.0	inf
9	8.2	9	1	173.9	108	189	432	837	3600.0	inf
10	7.5	8	2	188.7	104	184	424	824	3600.0	inf
11	7.1	8	3	200.2	112	200	464	904	3600.0	inf
12	6.5	7	3	196.1	105	189	441	861	3600.0	inf
13	6.1	7	3	188.8	112	203	476	931	3600.0	inf
14	6.0	6	3	193.1	102	186	438	858	3600.0	inf
15	5.4	6	3	194.6	108	198	468	918	3600.0	inf

Appendix C

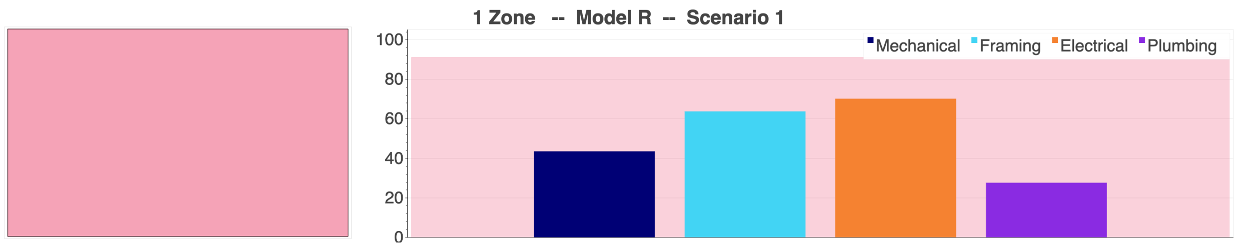
Optimal Partitions Suggested by WoLZo Algorithm

Appendix C offers graphical representations of the numerical results (already shown in tabular format in Appendix B) obtained using WoLZo for both Model R and Model L for every scenario while varying Z from 1 to 15 and with *alpha* set at 1.05.

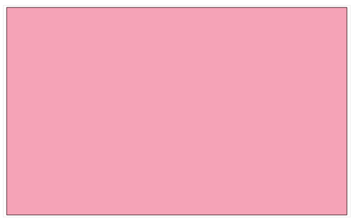
Shown on the left of each figure is the optimal partition selected by the WoLZo algorithm, and on the right is the corresponding histogram that depicts the workloads by zone for each trade in the process. In the histogram, different steps in different zones are shown along the x-axis and the workload (units of time per zone) is represented along the y-axis. Note that the various trades in the process are color-coded: Mechanical in navy blue, Framing in turquoise, Electrical in orange, and Plumbing in purple. Note also that the different zones are color-coded in the partition on the left, and that the same colors are used as backgrounds in the histogram on the right to demarcate those zones.

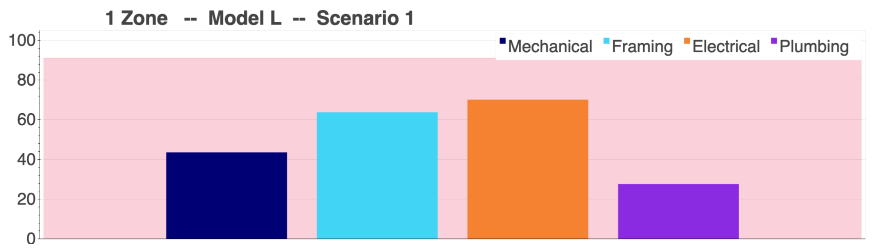
C.1 Scenario One: 1×1 matrix (1 cell)

C.1.1 Model R



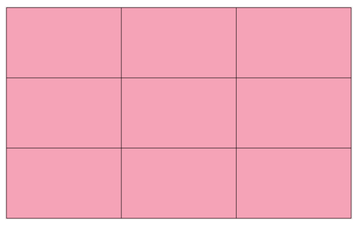
C.1.2 Model L

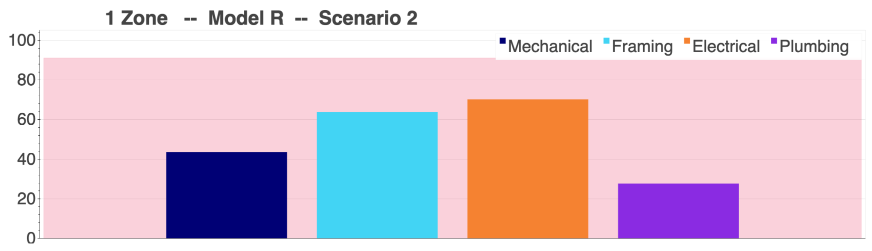


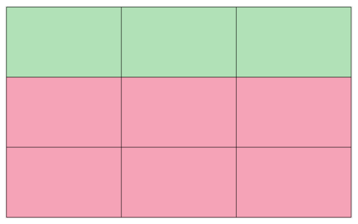


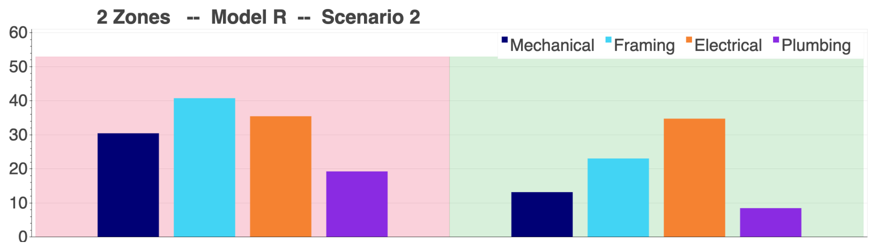
C.2 Scenario Two: 3×3 matrix (9 cells)

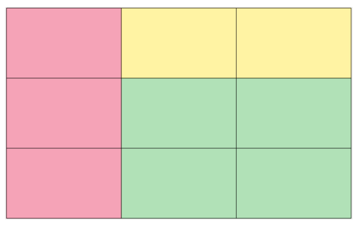
C.2.1 Model R

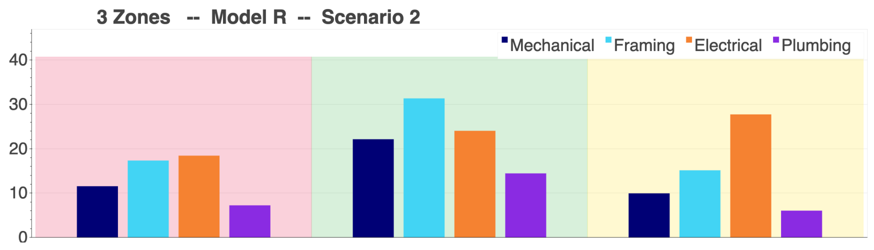


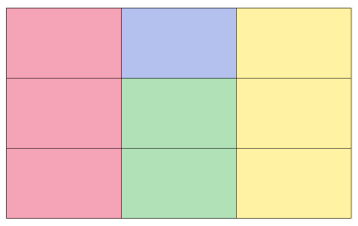


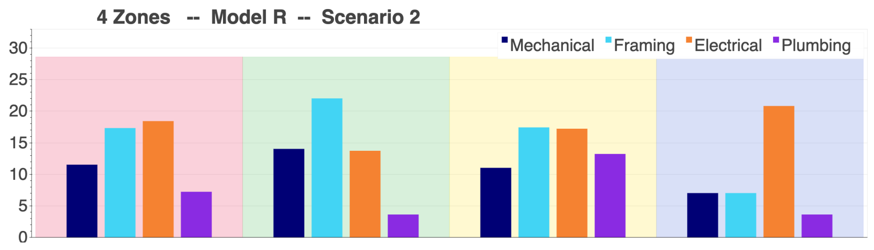


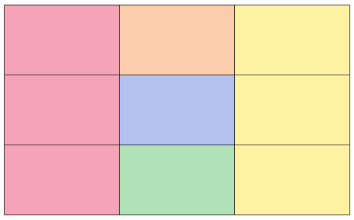


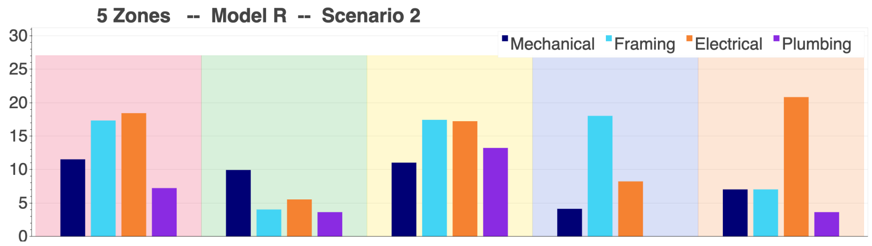


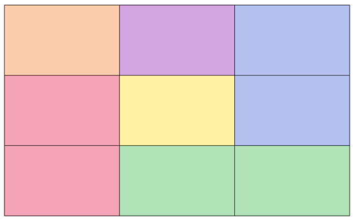


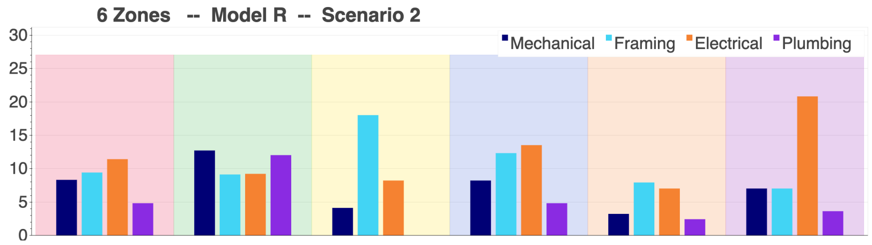


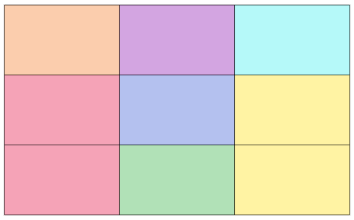


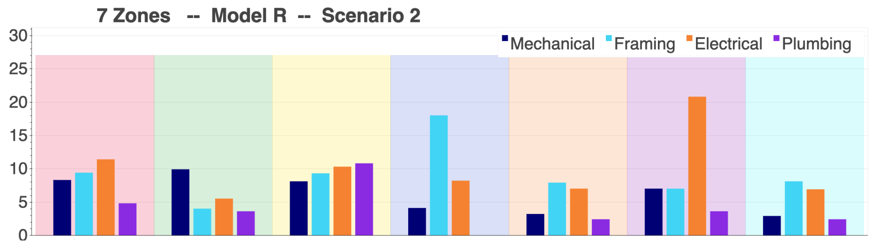


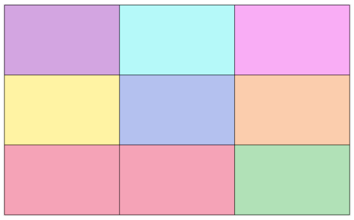

5 Zones -- Model R -- Scenario 2
Mechanical
Framing
Electrical
Plumbing

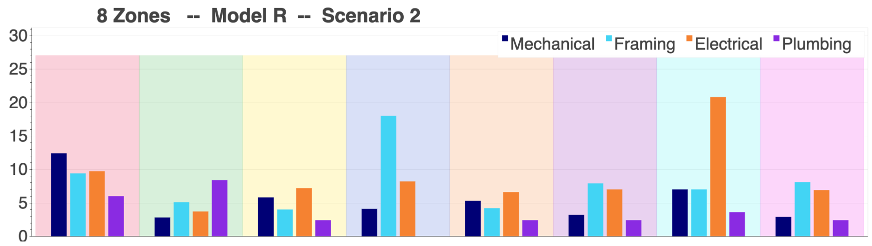


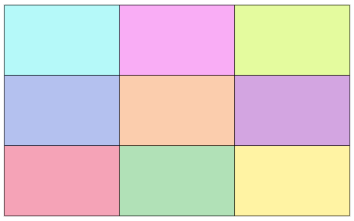

6 Zones -- Model R -- Scenario 2
Mechanical
Framing
Electrical
Plumbing

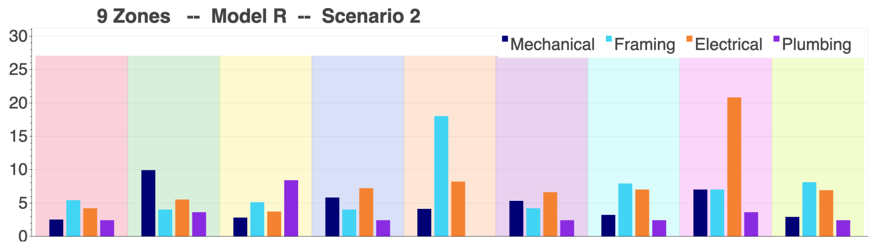



7 Zones -- Model R -- Scenario 2
Mechanical
Framing
Electrical
Plumbing

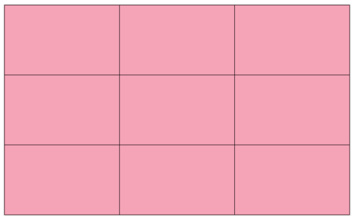


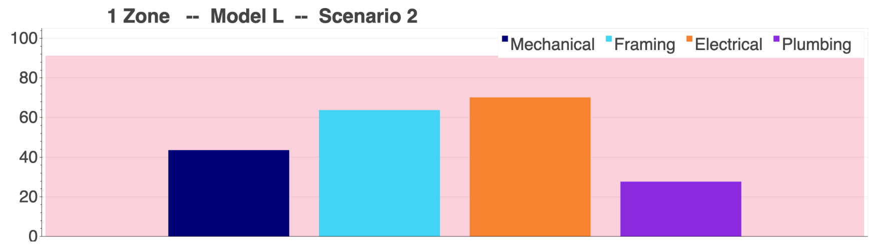

8 Zones -- Model R -- Scenario 2
Mechanical
Framing
Electrical
Plumbing

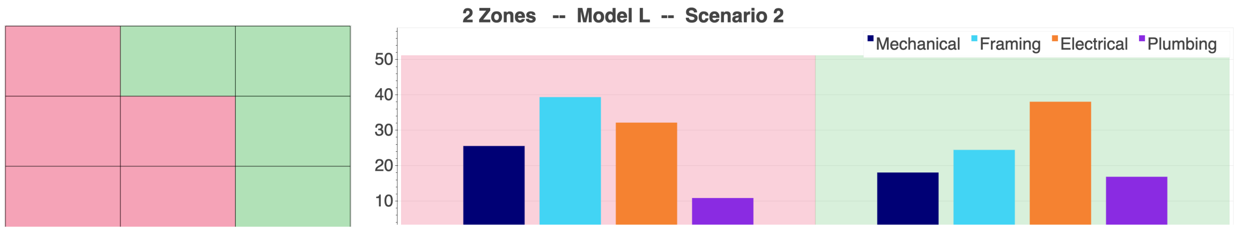


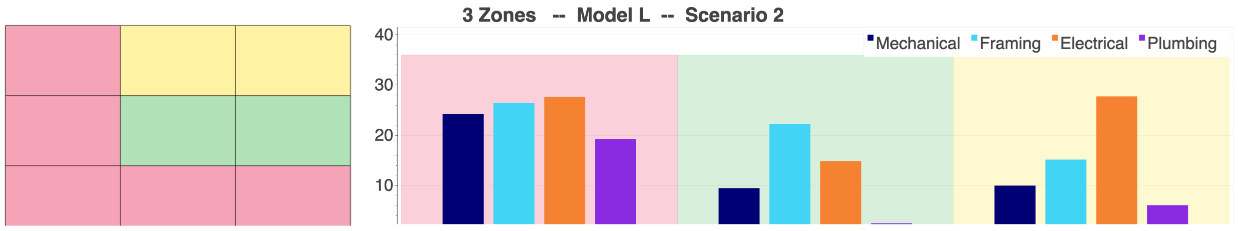

9 Zones -- Model R -- Scenario 2
Mechanical
Framing
Electrical
Plumbing

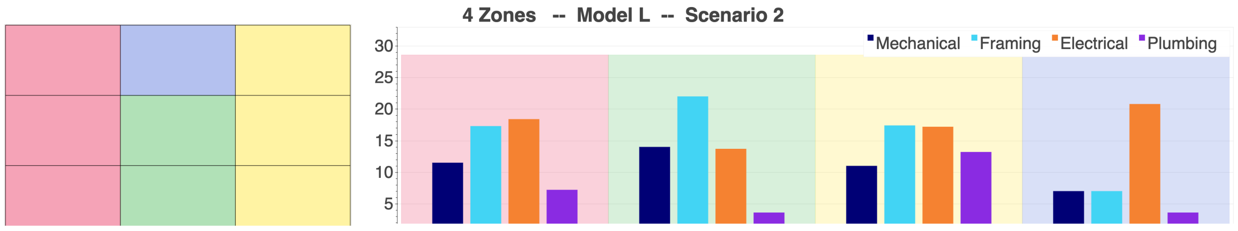
C.2.2 Model L

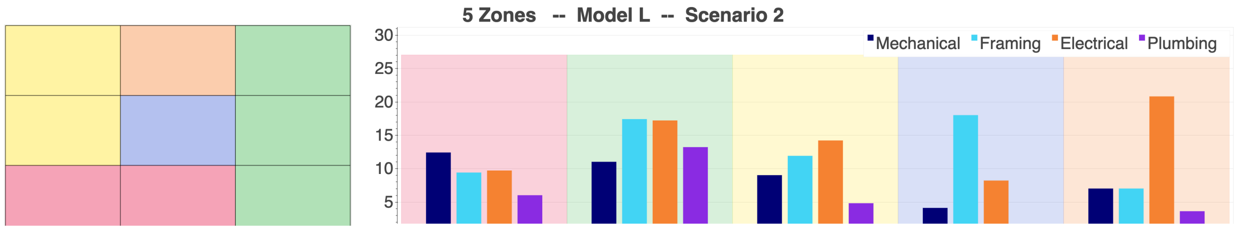


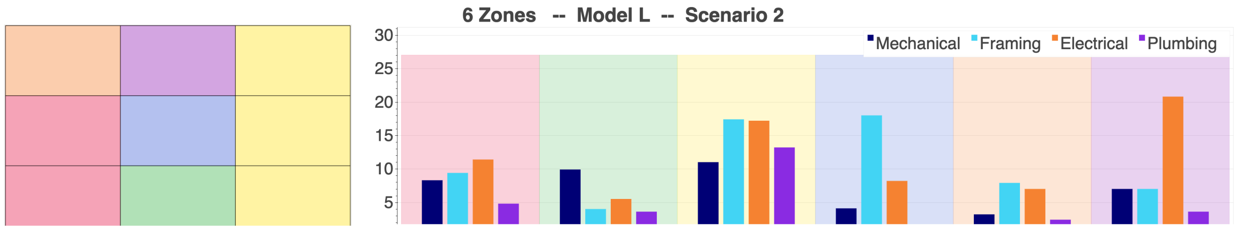

1 Zone -- Model L -- Scenario 2
Mechanical
Framing
Electrical
Plumbing

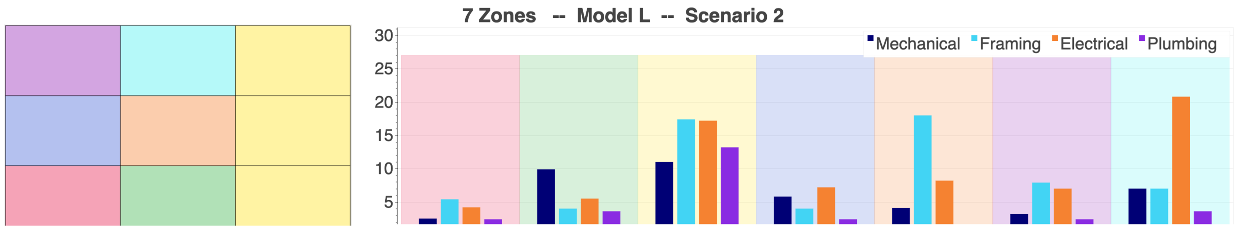

2 Zones -- Model L -- Scenario 2
Mechanical
Framing
Electrical
Plumbing
50
40
30
20
10

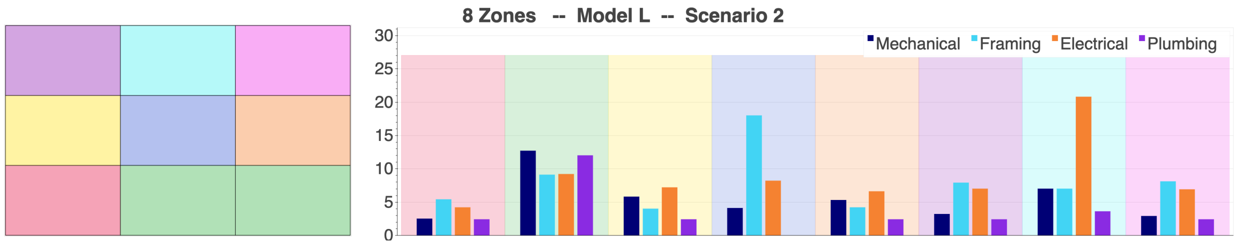

3 Zones -- Model L -- Scenario 2
Mechanical
Framing
Electrical
Plumbing
40
30
20
10

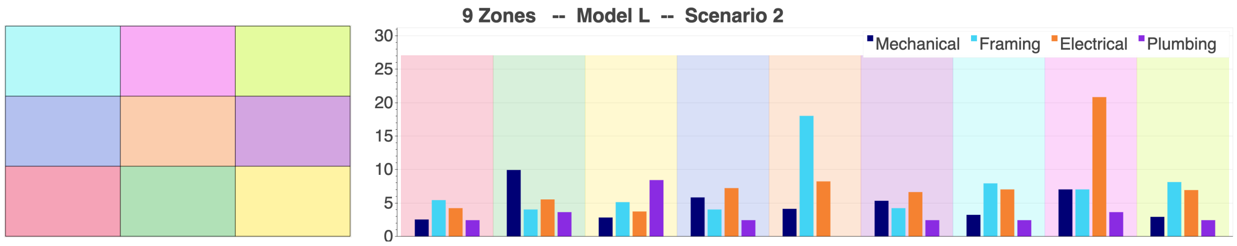

4 Zones -- Model L -- Scenario 2
Mechanical
Framing
Electrical
Plumbing
30
25
20
15
10
5


5 Zones -- Model L -- Scenario 2
Mechanical
Framing
Electrical
Plumbing
30
25
20
15
10
5


6 Zones -- Model L -- Scenario 2
Mechanical
Framing
Electrical
Plumbing
30
25
20
15
10
5

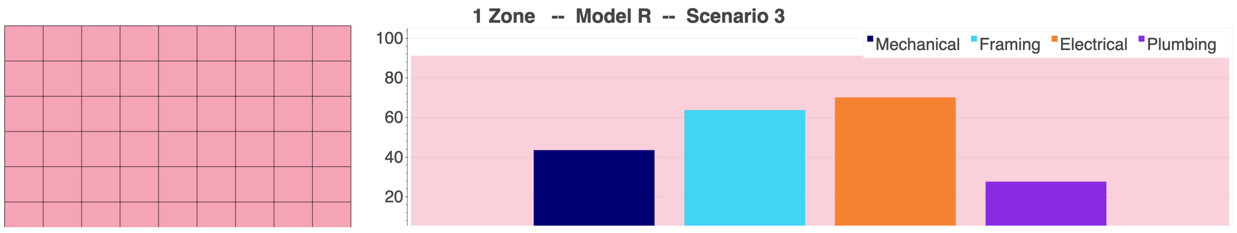

7 Zones -- Model L -- Scenario 2
Mechanical
Framing
Electrical
Plumbing
30
25
20
15
10
5

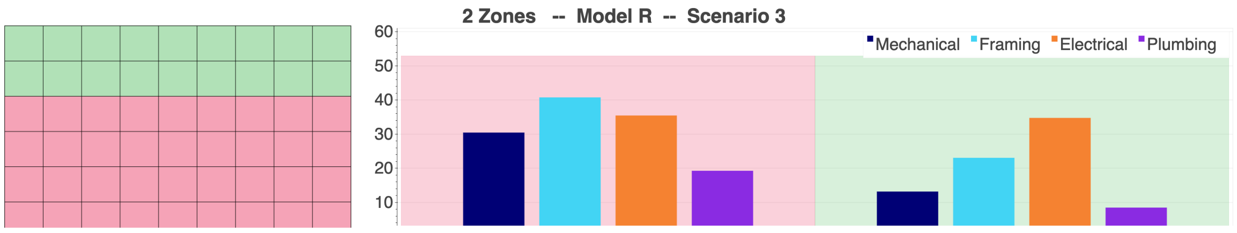

8 Zones -- Model L -- Scenario 2
Mechanical
Framing
Electrical
Plumbing
30
25
20
15
10
5
0

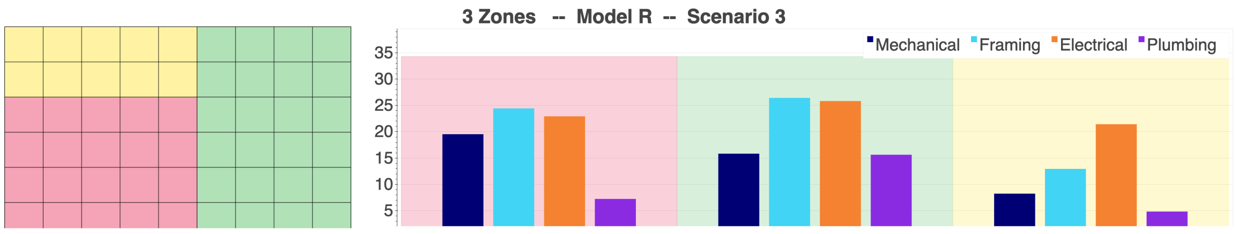


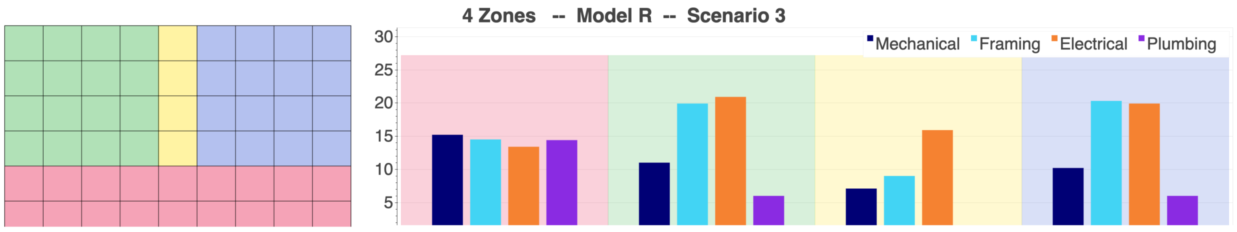
C.3 Scenario Three: 6×9 matrix (54 cells)

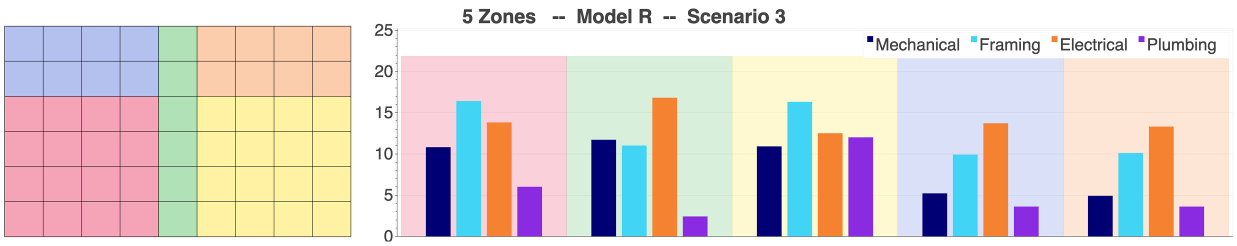
C.3.1 Model R

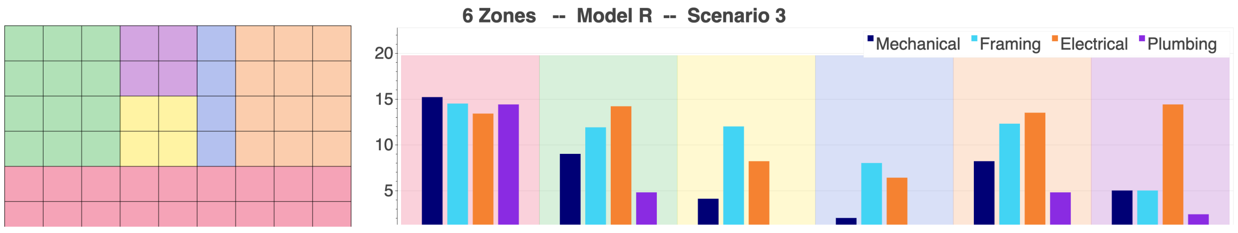


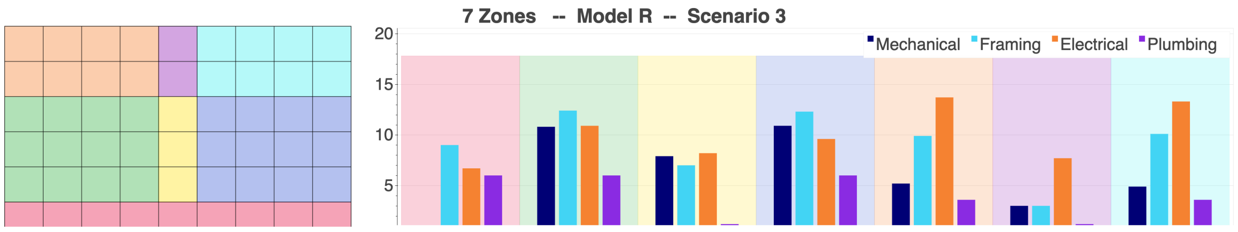


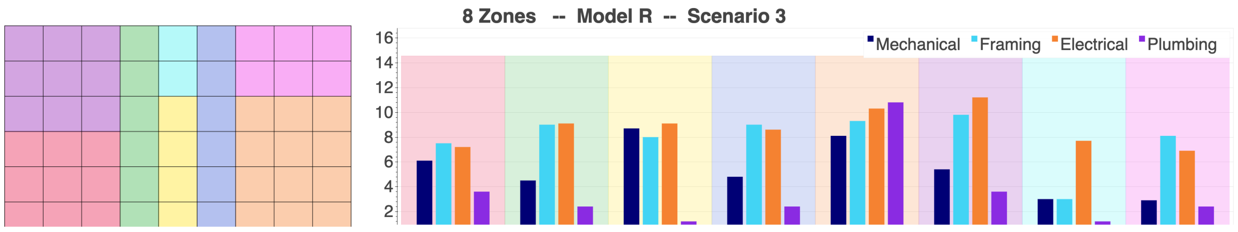


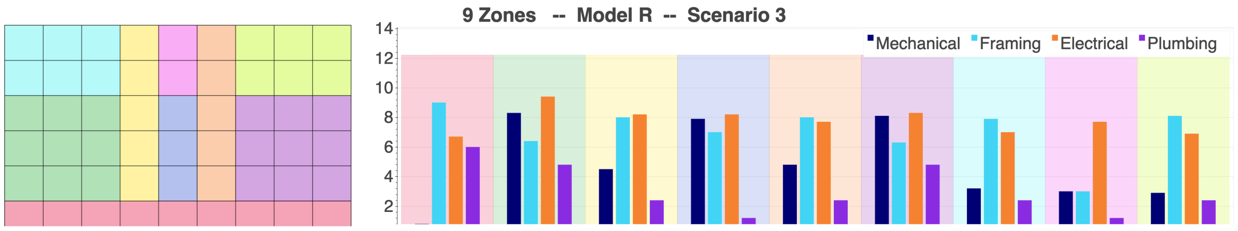


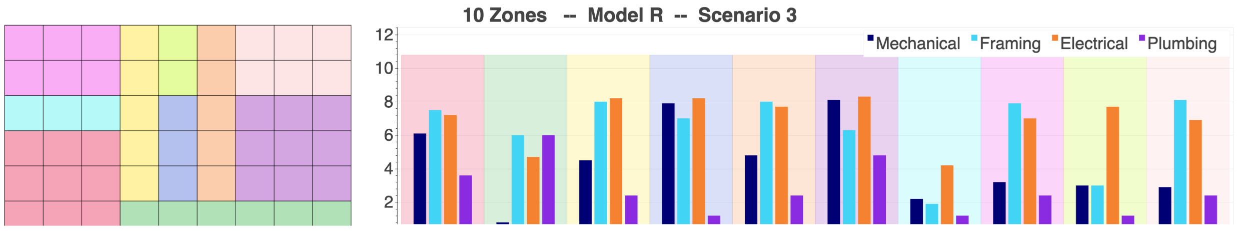


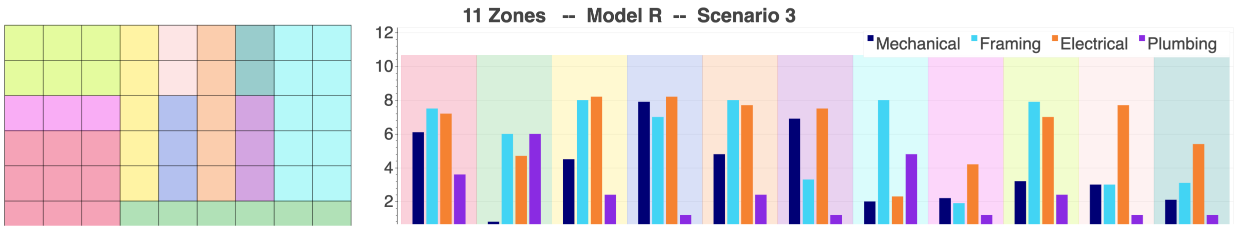

6 Zones -- Model R -- Scenario 3
Mechanical
Framing
Electrical
Plumbing

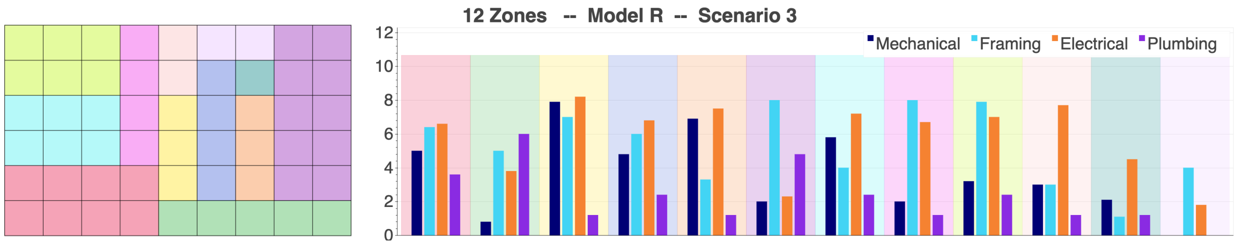

7 Zones -- Model R -- Scenario 3
Mechanical
Framing
Electrical
Plumbing

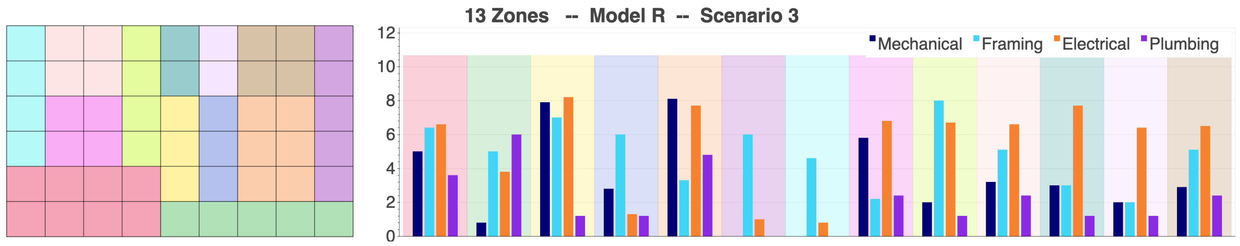

8 Zones -- Model R -- Scenario 3
Mechanical
Framing
Electrical
Plumbing

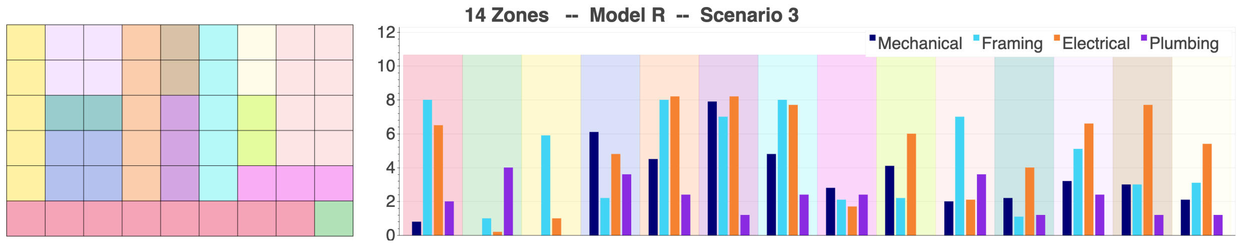

9 Zones -- Model R -- Scenario 3
Mechanical
Framing
Electrical
Plumbing

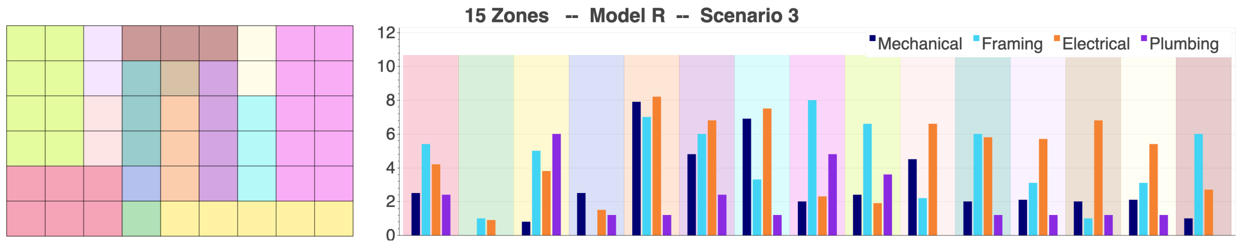

10 Zones -- Model R -- Scenario 3
Mechanical
Framing
Electrical
Plumbing


11 Zones -- Model R -- Scenario 3
Mechanical
Framing
Electrical
Plumbing

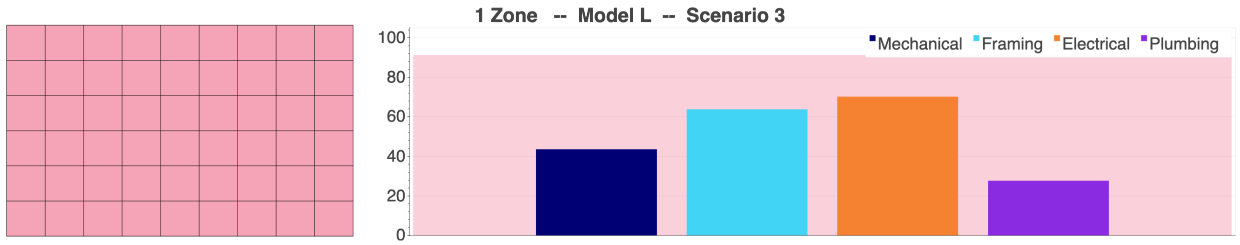

12 Zones -- Model R -- Scenario 3
Mechanical
Framing
Electrical
Plumbing

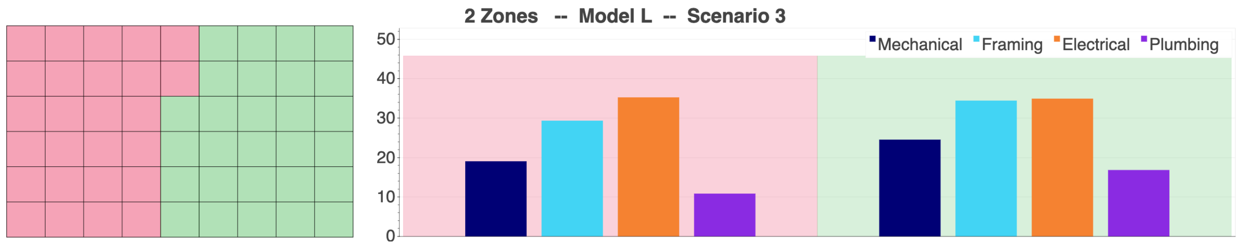

13 Zones -- Model R -- Scenario 3
Mechanical
Framing
Electrical
Plumbing

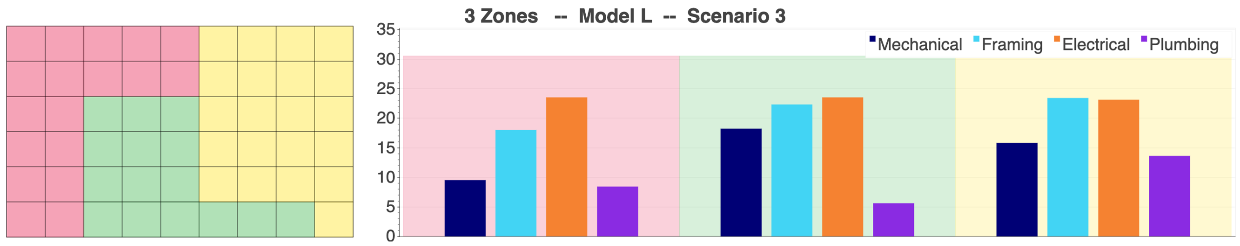

14 Zones -- Model R -- Scenario 3
Mechanical
Framing
Electrical
Plumbing

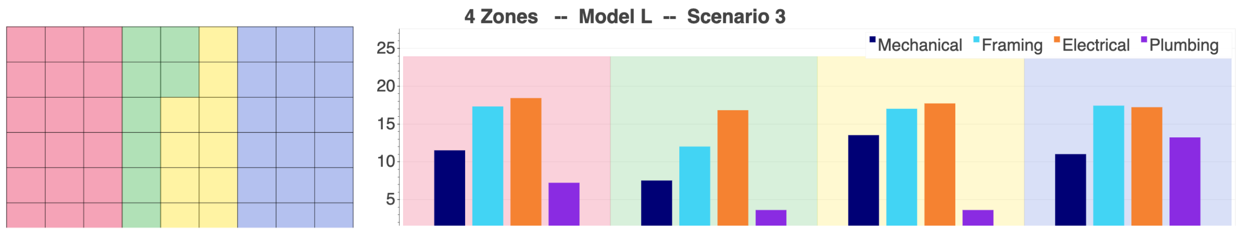

15 Zones -- Model R -- Scenario 3
Mechanical
Framing
Electrical
Plumbing

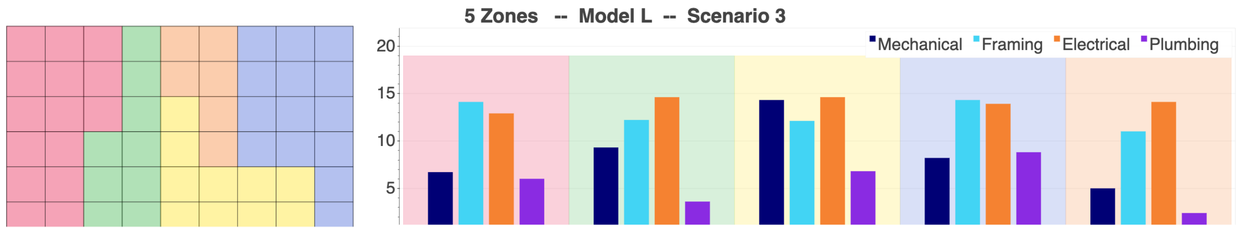
C.3.2 Model L

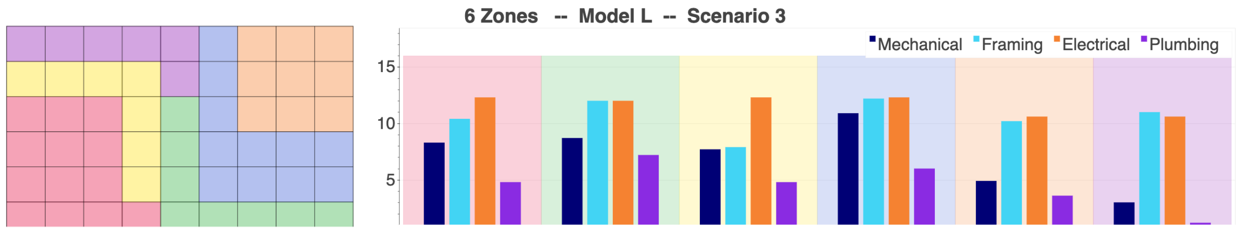

1 Zone -- Model L -- Scenario 3
Mechanical
Framing
Electrical
Plumbing

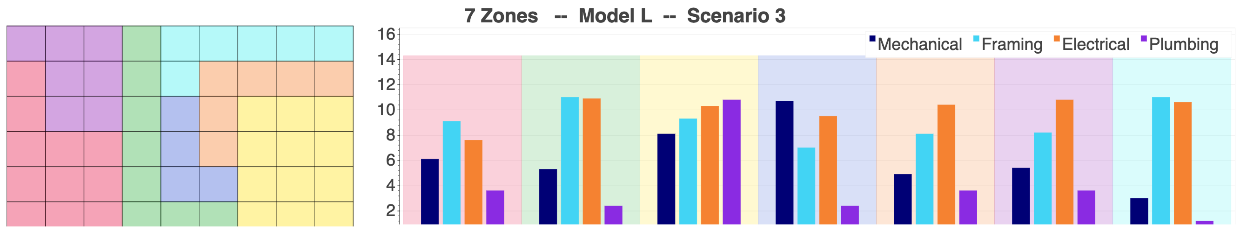

2 Zones -- Model L -- Scenario 3
Mechanical
Framing
Electrical
Plumbing

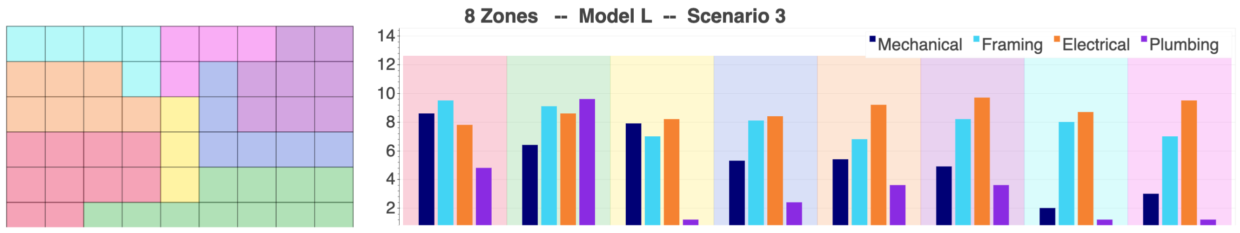

3 Zones -- Model L -- Scenario 3
Mechanical
Framing
Electrical
Plumbing

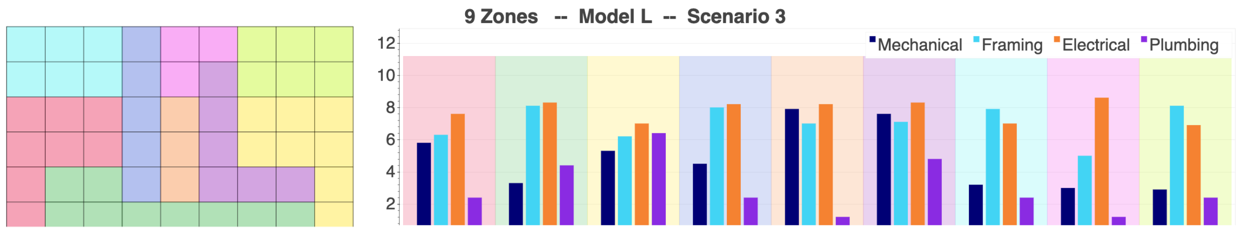

4 Zones -- Model L -- Scenario 3
Mechanical
Framing
Electrical
Plumbing

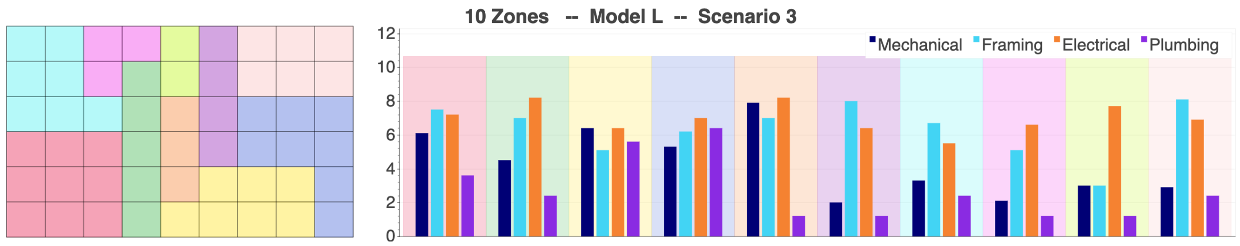

5 Zones -- Model L -- Scenario 3
Mechanical
Framing
Electrical
Plumbing

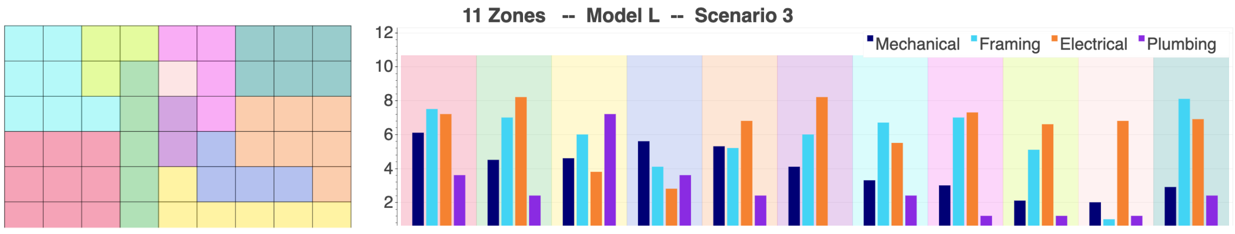

6 Zones -- Model L -- Scenario 3
Mechanical
Framing
Electrical
Plumbing

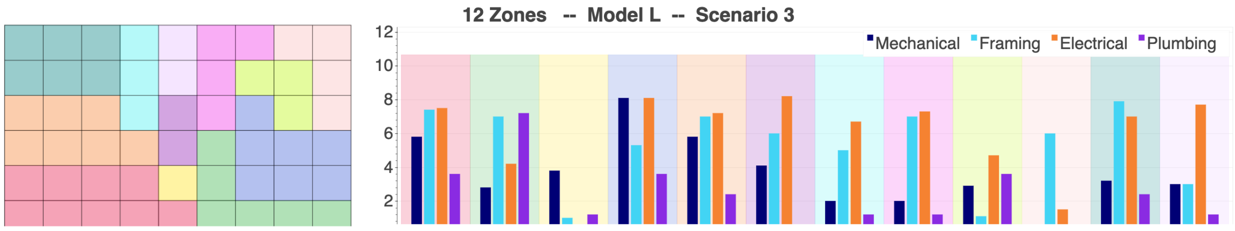

7 Zones -- Model L -- Scenario 3
Mechanical
Framing
Electrical
Plumbing

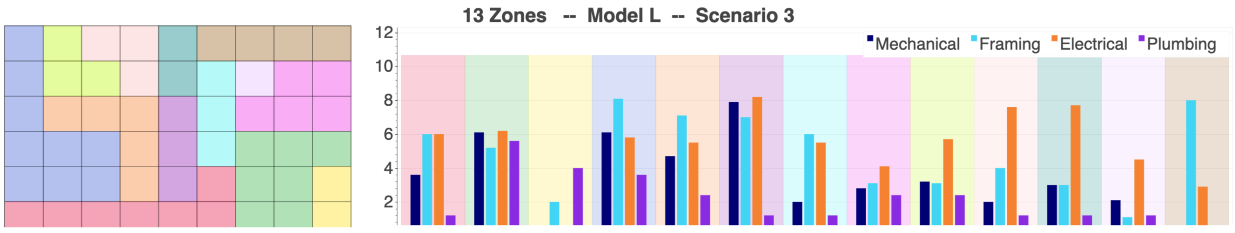

8 Zones -- Model L -- Scenario 3
Mechanical
Framing
Electrical
Plumbing

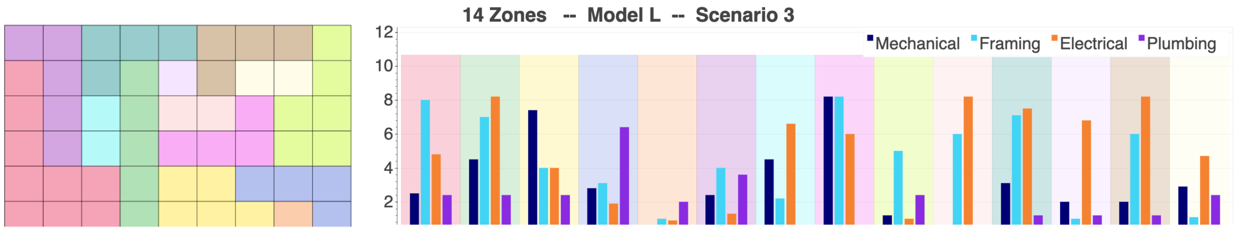

9 Zones -- Model L -- Scenario 3
Mechanical
Framing
Electrical
Plumbing

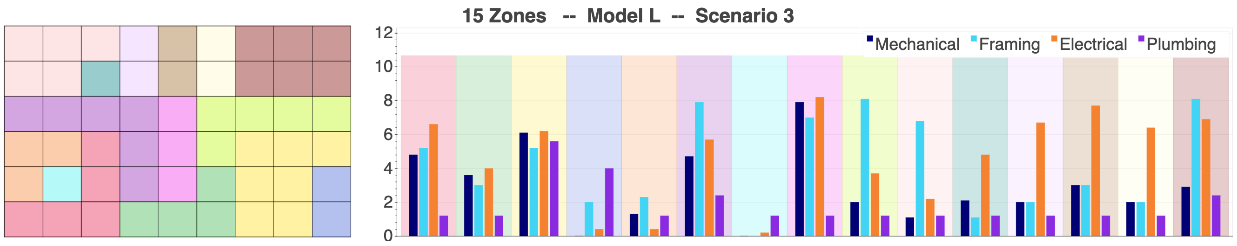

10 Zones -- Model L -- Scenario 3
Mechanical
Framing
Electrical
Plumbing





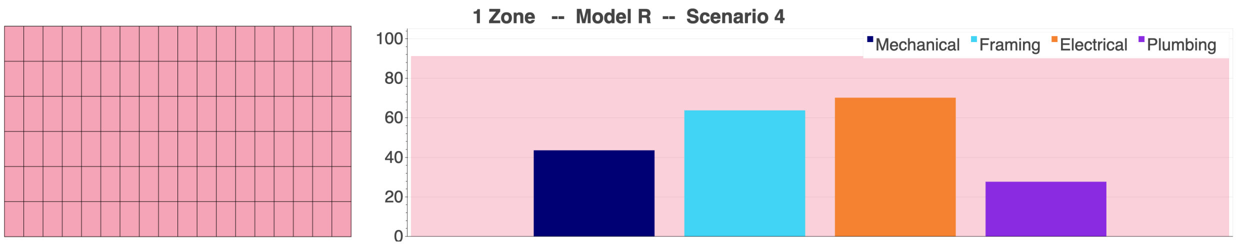


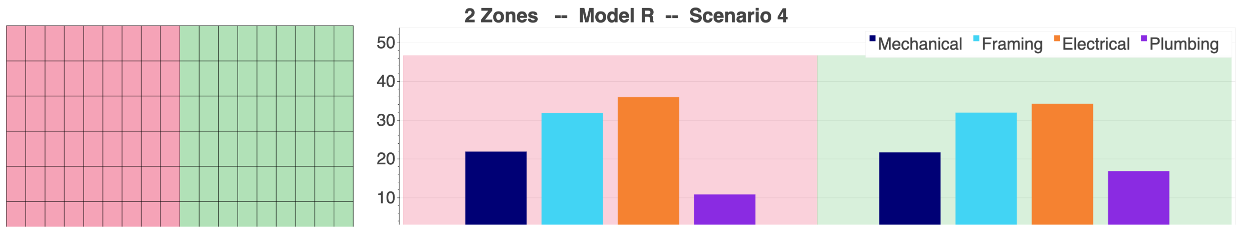


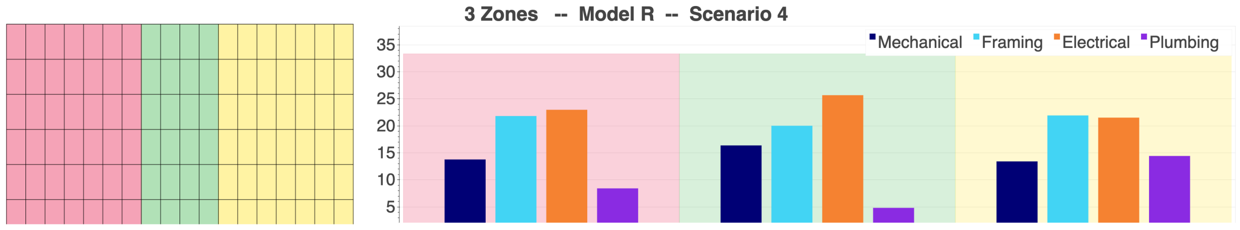


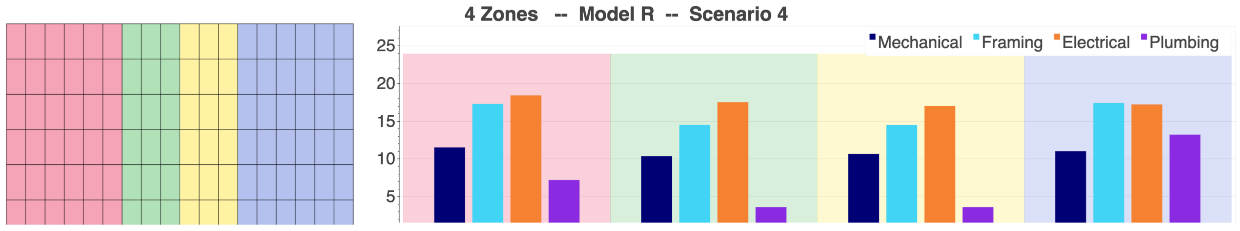
C.4 Scenario Four: 6×18 matrix (108 cells)

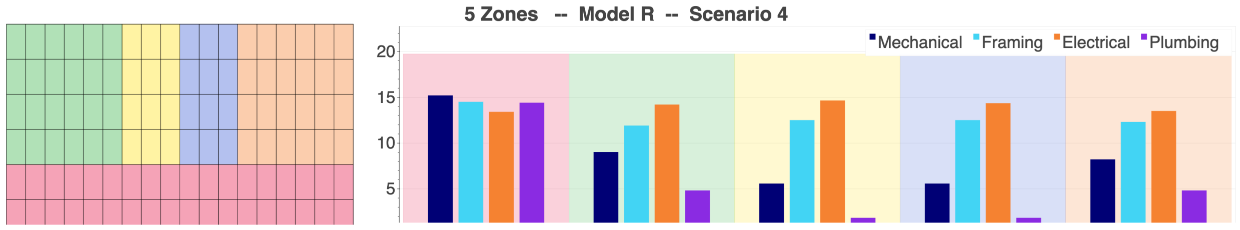
C.4.1 Model R

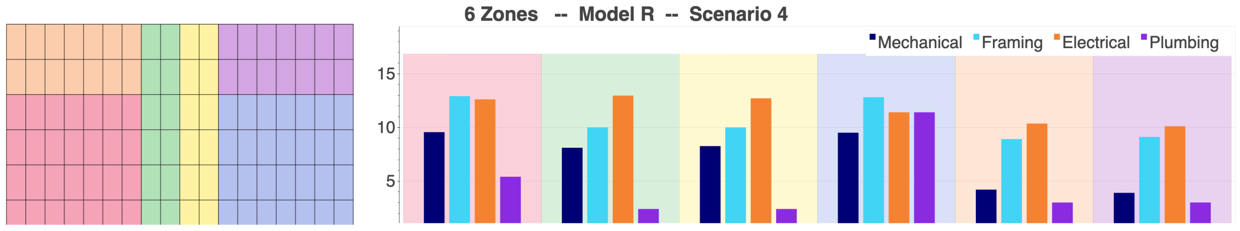


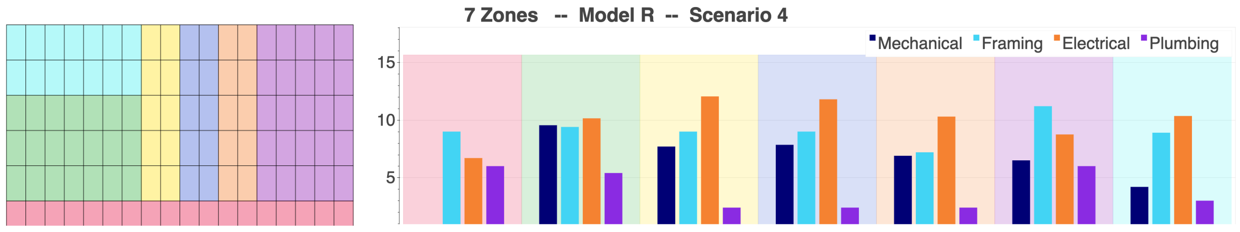

2 Zones -- Model R -- Scenario 4
Mechanical
Framing
Electrical
Plumbing

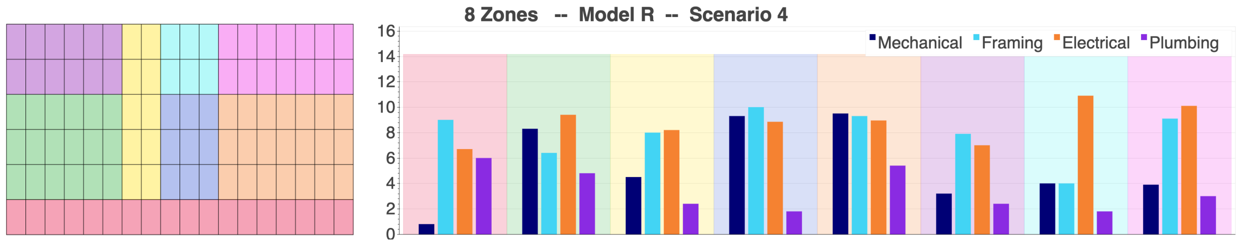

3 Zones -- Model R -- Scenario 4
Mechanical
Framing
Electrical
Plumbing

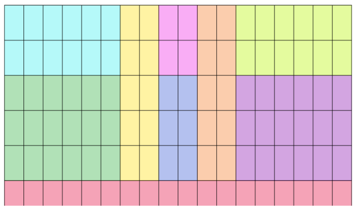
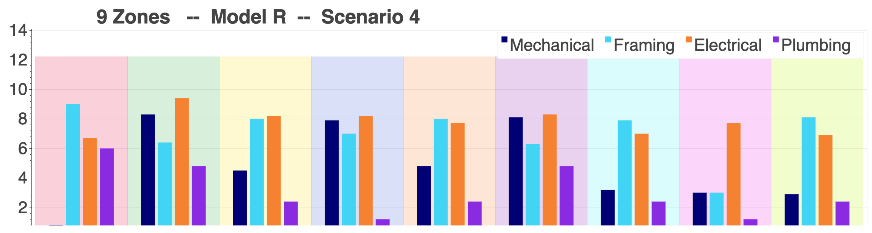

4 Zones -- Model R -- Scenario 4
Mechanical
Framing
Electrical
Plumbing

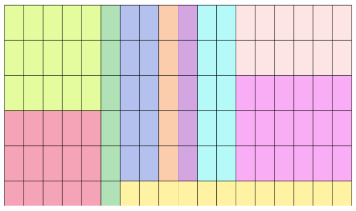
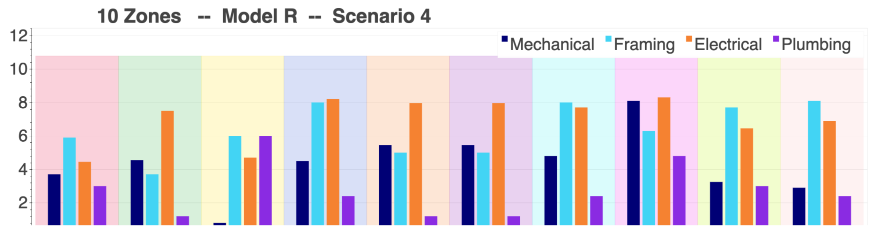

5 Zones -- Model R -- Scenario 4
Mechanical
Framing
Electrical
Plumbing

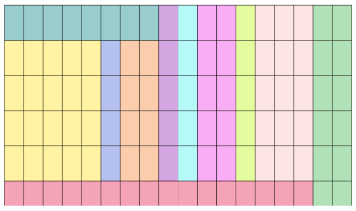
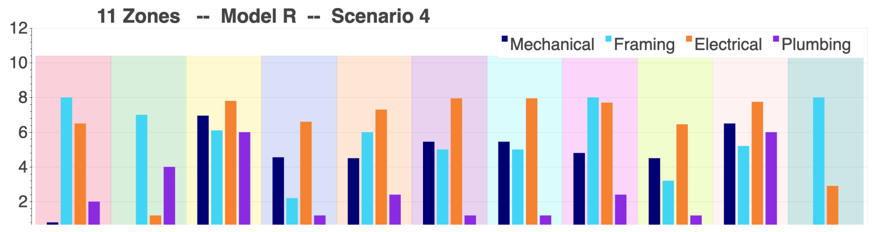

6 Zones -- Model R -- Scenario 4
Mechanical
Framing
Electrical
Plumbing

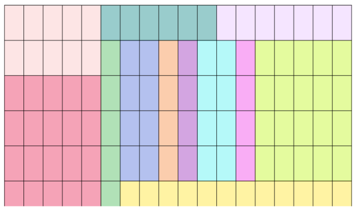
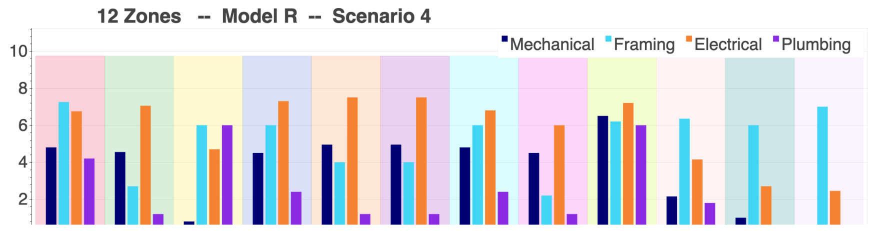

7 Zones -- Model R -- Scenario 4
Mechanical
Framing
Electrical
Plumbing

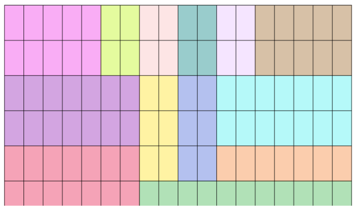
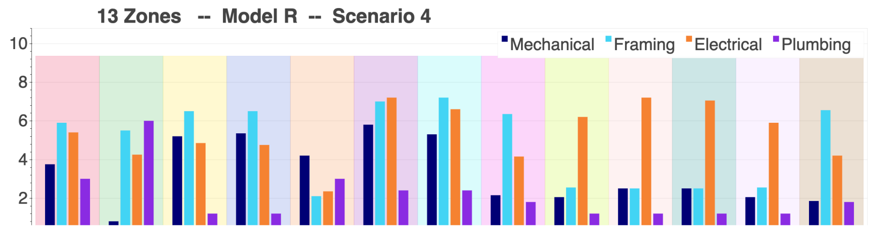

8 Zones -- Model R -- Scenario 4
Mechanical
Framing
Electrical
Plumbing

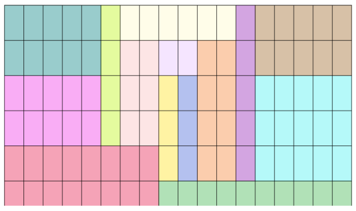
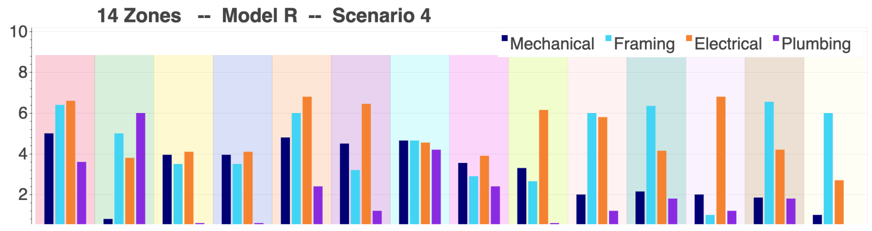


9 Zones -- Model R -- Scenario 4
Mechanical
Framing
Electrical
Plumbing

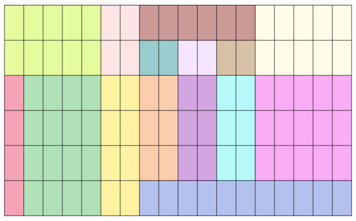
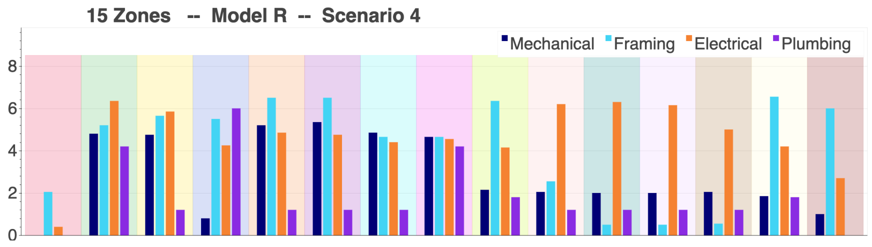


10 Zones -- Model R -- Scenario 4
Mechanical
Framing
Electrical
Plumbing



11 Zones -- Model R -- Scenario 4
Mechanical
Framing
Electrical
Plumbing

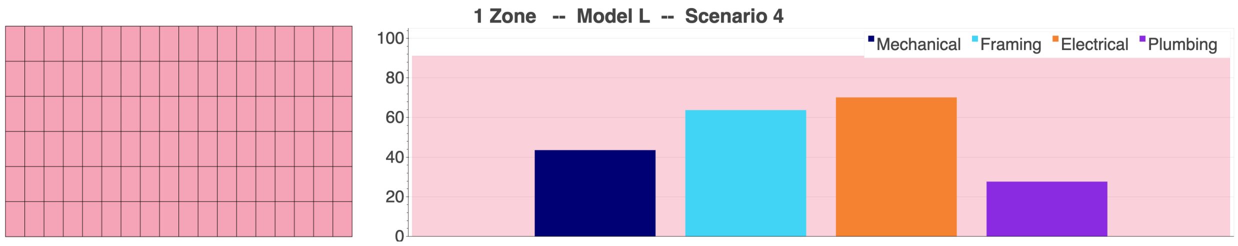


12 Zones -- Model R -- Scenario 4
Mechanical
Framing
Electrical
Plumbing

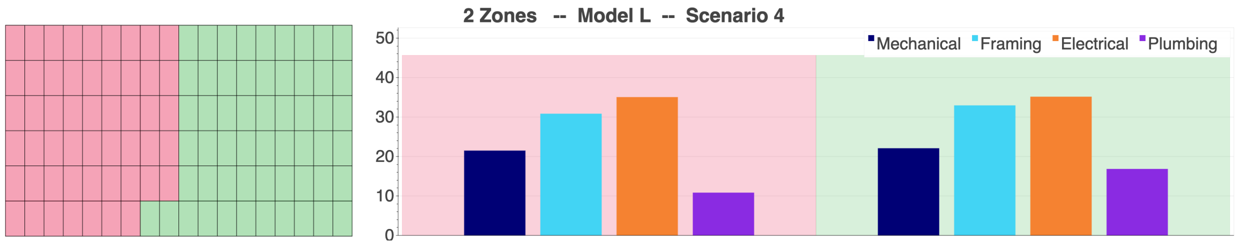


13 Zones -- Model R -- Scenario 4
Mechanical
Framing
Electrical
Plumbing

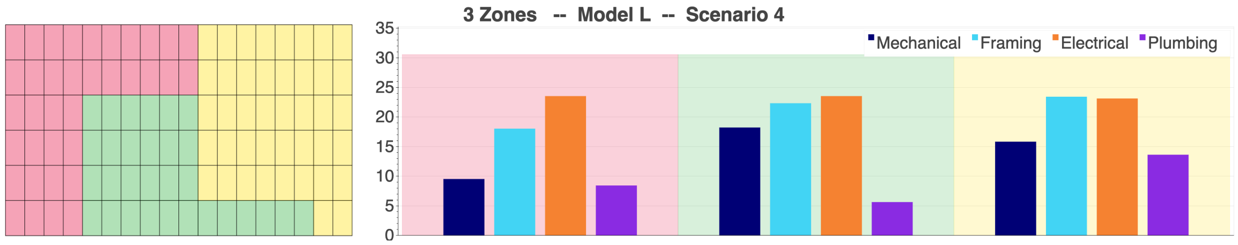


14 Zones -- Model R -- Scenario 4
Mechanical
Framing
Electrical
Plumbing

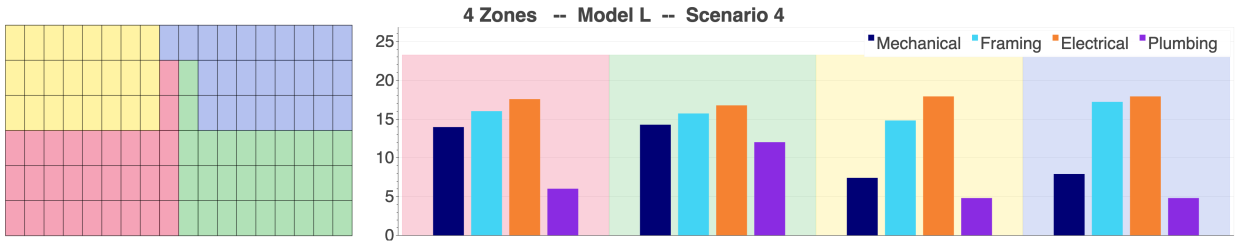


15 Zones -- Model R -- Scenario 4
Mechanical
Framing
Electrical
Plumbing

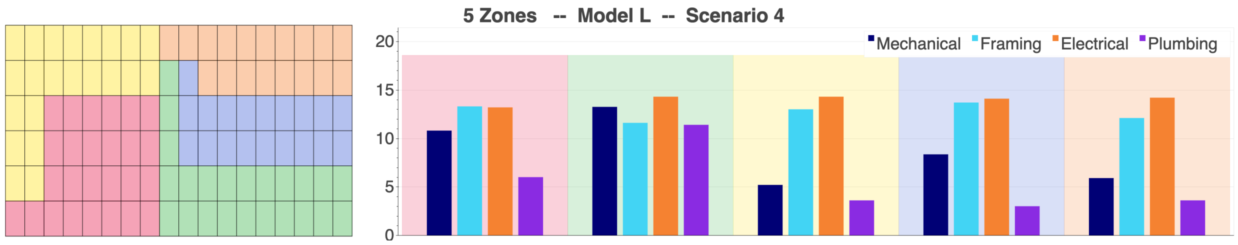
C.4.2 Model L

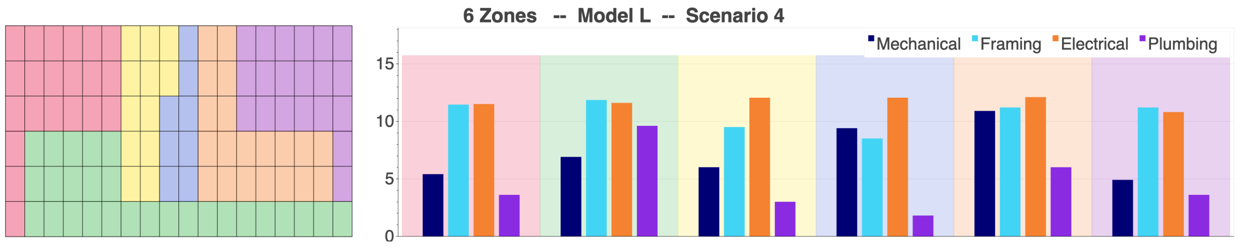

1 Zone -- Model L -- Scenario 4
Mechanical
Framing
Electrical
Plumbing
0
20
40
60
80
100

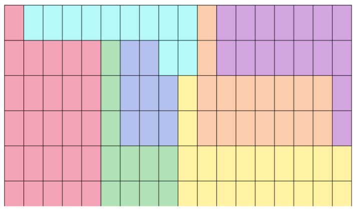
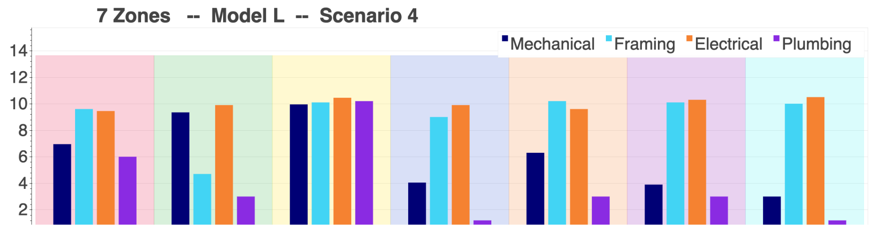

2 Zones -- Model L -- Scenario 4
Mechanical
Framing
Electrical
Plumbing
0
10
20
30
40
50

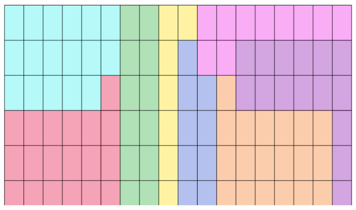
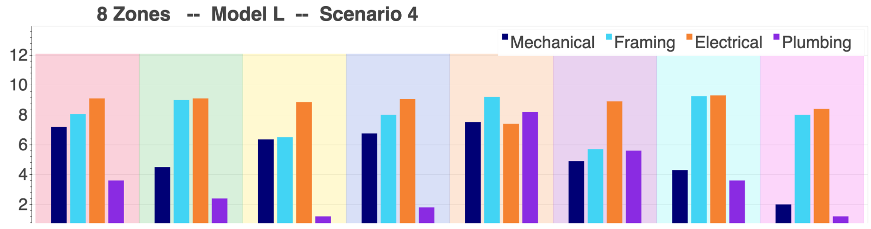

3 Zones -- Model L -- Scenario 4
Mechanical
Framing
Electrical
Plumbing
0
5
10
15
20
25
30
35

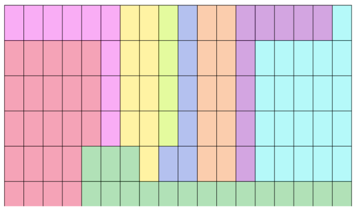
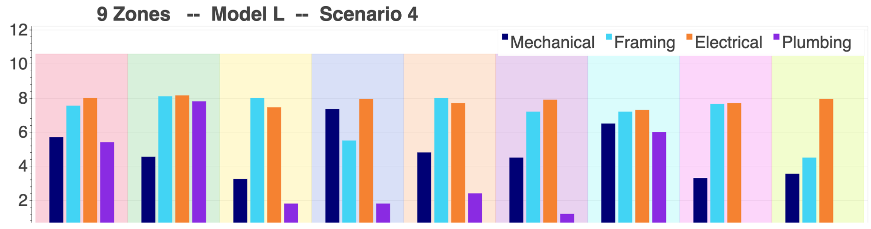

4 Zones -- Model L -- Scenario 4
Mechanical
Framing
Electrical
Plumbing
0
5
10
15
20
25

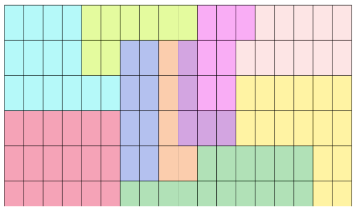
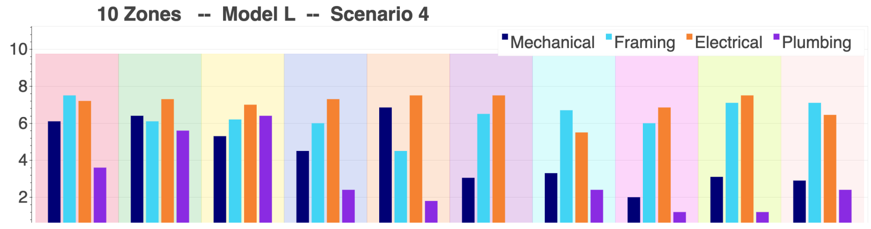

5 Zones -- Model L -- Scenario 4
Mechanical
Framing
Electrical
Plumbing
0
5
10
15
20

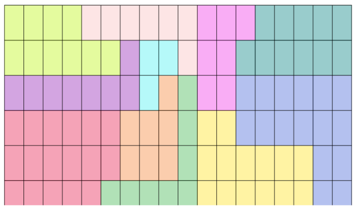
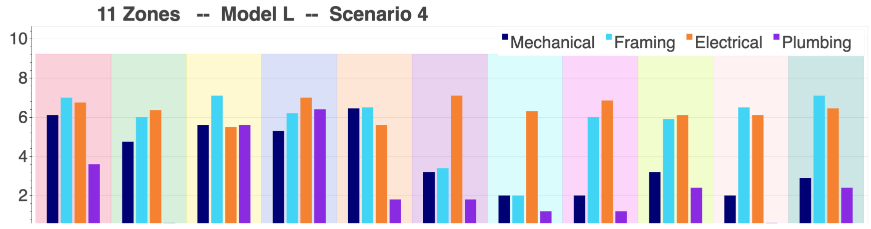

6 Zones -- Model L -- Scenario 4
Mechanical
Framing
Electrical
Plumbing
0
5
10
15

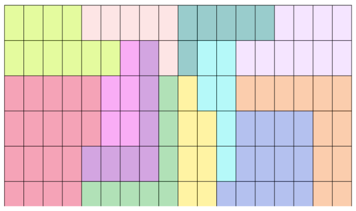
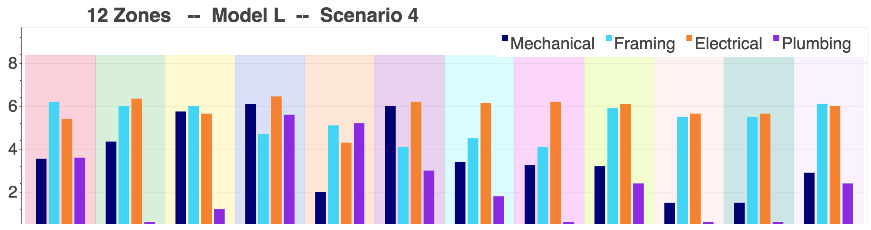


7 Zones -- Model L -- Scenario 4
Mechanical
Framing
Electrical
Plumbing
2 4 6 8 10 12 14

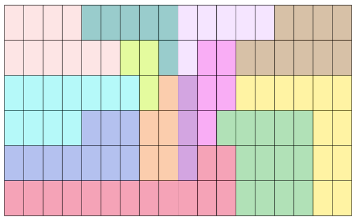
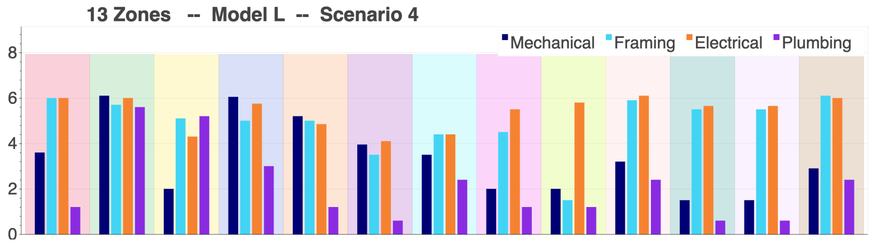


8 Zones -- Model L -- Scenario 4
Mechanical
Framing
Electrical
Plumbing
2 4 6 8 10 12

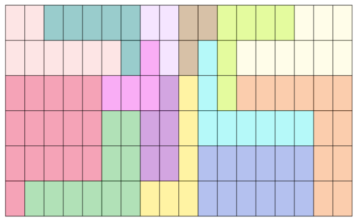


9 Zones -- Model L -- Scenario 4
Mechanical
Framing
Electrical
Plumbing
2 4 6 8 10 12

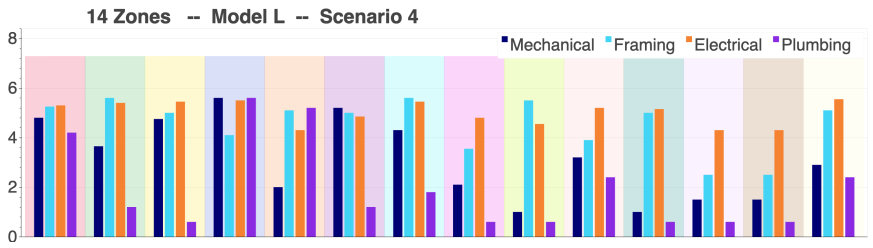


10 Zones -- Model L -- Scenario 4
Mechanical
Framing
Electrical
Plumbing
2 4 6 8 10

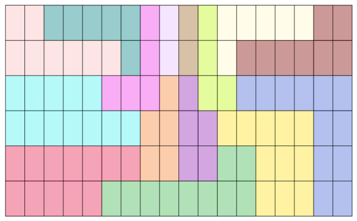


11 Zones -- Model L -- Scenario 4
Mechanical
Framing
Electrical
Plumbing
2 4 6 8 10

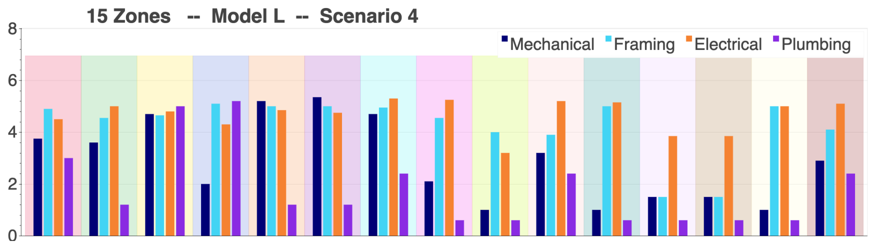


12 Zones -- Model L -- Scenario 4
Mechanical
Framing
Electrical
Plumbing
2 4 6 8



13 Zones -- Model L -- Scenario 4
Mechanical
Framing
Electrical
Plumbing
0 2 4 6 8




14 Zones -- Model L -- Scenario 4
Mechanical
Framing
Electrical
Plumbing
0
2
4
6
8




15 Zones -- Model L -- Scenario 4
Mechanical
Framing
Electrical
Plumbing
0
2
4
6
8