Exploration vs. Exploitation: Reducing Uncertainty in Operational Problems

by

Yaron Shaposhnik

B.Sc., Technion – Israel Institute of Technology (2006)
M.Sc., Technion – Israel Institute of Technology (2011)

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2016

© Massachusetts Institute of Technology 2016. All rights reserved.

Author ................................................................. Sloan School of Management
June 30, 2016

Certified by ......................................................... Retsef Levi

J. Spencer Standish (1945) Professor of Management
Professor of Operations Management
Thesis Supervisor

Certified by .......................................................... Thomas L. Magnanti
Institute Professor
Professor of Operations Research
Thesis Supervisor

Accepted by ......................................................... Dimitris Bertsimas
Boeing Professor of Operations Research
Co-director of Operations Research Center
Exploration vs. Exploitation: Reducing Uncertainty in Operational Problems

by

Yaron Shaposhnik

Submitted to the Sloan School of Management
on June 30, 2016, in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

Motivated by several core operational applications, we introduce a class of multistage stochastic optimization models that capture a fundamental tradeoff between performing work under uncertainty (exploitation) and investing resources to reduce the uncertainty in the decision making (exploration/testing). Unlike existing models, in which the exploration-exploitation tradeoffs typically relate to learning the underlying distributions, the models we introduce assume a known probabilistic characterization of the uncertainty, and focus on the tradeoff of learning exact realizations.

In the first part of the thesis (Chapter 2), we study a class of scheduling problems that capture common settings in service environments in which the service provider must serve a collection of jobs that have a-priori uncertain processing times and priorities (modeled as weights). In addition, the service provider must decide how to dynamically allocate capacity between processing jobs and testing jobs to learn more about their respective processing times and weights. We obtain structural results of optimal policies that provide managerial insights, efficient optimal and near-optimal algorithms, and quantification of the value of testing.

In the second part of the thesis (Chapter 3), we generalize the model introduced in the first part by studying how to prioritize testing when jobs have different uncertainties. We model difference in uncertainties using the convex order, a general relation between distributions, which implies that the variance of one distribution is higher than the variance of the other distribution. Using an analysis based on the concept of mean preserving local spread, we show that the structure of the optimal policy generalizes that of the initial model where jobs were homogeneous and had equal weights.

Finally, in the third part of the thesis (Chapter 4), we study a broad class of stochastic combinatorial optimization that can be formulated as Linear Programs whose objective coefficients are random variables that can be tested, and whose constraint polyhedron has the structure of a polymatroid. We characterize the optimal policy and show that similar types of policies optimally govern testing decisions in this setting as well.
Thesis Supervisor: Retsef Levi
Title: J. Spencer Standish (1945) Professor of Management
Professor of Operations Management

Thesis Supervisor: Thomas L. Magnanti
Title: Institute Professor
Professor of Operations Research
Acknowledgments

The Hebrew poet Shaul Tchernichovsky wrote that human beings are molded by their environment (“Haadam eino ela tavnit nof moladeto...”). I am grateful that MIT and the ORC have been my environment in the last five years, and hope they remain part of me as I move forward in life.

An environment is mostly made of its people, and first I would like to express my deepest gratitude to my advisors Retsef Levi and Tom Magnanti for mentoring and guiding me in this journey. I appreciate everything you did for me and taught me about research, teaching, and life in general. I wouldn’t have gotten to this point without your support and encouragement throughout these years.

I would like to thank Georgia Perakis and David Shmoys, my thesis committee members. David, for your feedback and suggestions, and Georgia, for all your advice and help in the academic job search, and for being welcoming and kind to all of us at the ORC. I would also like to thank Robert Krauthgamer, for your careful feedback and advice about writing and research. Also, to Sloan faculty for whom I TA’ed and who taught me about various aspects of teaching: Rob Freund, Jim Orlin, Cynthia Rudin, Juan Pablo Vielma, Tauhid Zaman, and Karen Zheng. Karen and Tauhid, thank you for your help and advice about presenting my work. I would also like to thank Yale Herer, my Masters advisor from the Technion who introduced me to the academic world and encouraged me to pursue it ever since.

In addition, I would like to thank Laura Rose, Andrew Carvalho, Dimitris Bertsimas, and Patrick Jaillet, for making the ORC such an extraordinary place, and to Ariel Brander for all your help during these years. To my friends at the ORC, getting to know you was the most enjoyable and interesting part of this experience. To Ali Aouad, Chen Attias, Maxime Cohen, and Leon Valdes, I appreciate the countless hours we spent together and your support, and also Fernanda Bravo, Wang-Chi Cheung, Iain Dunning, David Fagnan, Paul Grigas, Swati Gupta, Nathan Kallus, John Kessler, Miles Lubin, Will Ma, Nikita Korolko, Peng Shi, John Silberholz, Joel Tay, He Wang, and Nataly Youssef.

Last but not least, I thank my family here and across the sea. Mostly, my parents Yossi
and Anna, my lovely wife Efrat, for “the kindness of thy youth, the love of thine espousals, when thou wentest after me in the wilderness, in a land that was not sown” (Jeremiah 2:2), and our son Gadi who made this effort worthwhile.
# Contents

1 Introduction ................................................................. 15
   1.1 Motivating Examples .................................................. 16
   1.2 Models ................................................................. 18
   1.3 Methods ............................................................... 20
   1.4 Results ............................................................... 21

2 Scheduling with Testing .................................................. 25
   2.1 Introduction ........................................................... 26
   2.2 Mathematical Formulation ......................................... 31
      2.2.1 The WSPT Rule .................................................. 32
      2.2.2 Marginal Cost Accounting ..................................... 33
      2.2.3 DP Formulation .................................................. 34
   2.3 Properties of the Optimal Policy ................................ 36
      2.3.1 Short Test Time \( (\rho_a < \rho) \) .................................. 38
      2.3.2 Long Test Time \( (\rho < \rho_a) \) .................................. 41
   2.4 Solutions and Algorithms ........................................... 41
      2.4.1 Low Dimensional DP Formulation ............................. 41
      2.4.2 Optimal Threshold Policy ...................................... 49
      2.4.3 FPTAS ............................................................ 50
   2.5 Optimal Myopic Policies ............................................. 52
   2.6 The Value of Testing ................................................ 56

7
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.6.1</td>
<td>Heuristics</td>
<td>56</td>
</tr>
<tr>
<td>2.6.2</td>
<td>The Initial Number of Jobs</td>
<td>59</td>
</tr>
<tr>
<td>2.6.3</td>
<td>The Testing Time</td>
<td>61</td>
</tr>
<tr>
<td>2.6.4</td>
<td>Numerical Illustration</td>
<td>61</td>
</tr>
<tr>
<td>2.7</td>
<td>Extensions</td>
<td>62</td>
</tr>
<tr>
<td>2.8</td>
<td>Conclusions</td>
<td>65</td>
</tr>
</tbody>
</table>

### 3 To Test or Not to Test: Reducing Uncertainty in Scheduling Diverse Jobs | 67

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>3.2</td>
<td>The Convex Order</td>
</tr>
<tr>
<td>3.3</td>
<td>Problem Formulation</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Notation and Assumptions</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Dynamic Programming Formulation</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Scheduling and Testing Homogeneous Jobs</td>
</tr>
<tr>
<td>3.4</td>
<td>Analysis</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Generalized testing thresholds</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Optimal Processing Order</td>
</tr>
<tr>
<td>3.4.3</td>
<td>Sub-optimal Testing</td>
</tr>
<tr>
<td>3.4.4</td>
<td>The Revised DP Formulation</td>
</tr>
<tr>
<td>3.4.5</td>
<td>Myopic Stopping Rule</td>
</tr>
<tr>
<td>3.4.6</td>
<td>The Optimal Testing Order</td>
</tr>
<tr>
<td>3.4.7</td>
<td>The Optimal Policy</td>
</tr>
<tr>
<td>3.5</td>
<td>Conclusions and Future Directions</td>
</tr>
</tbody>
</table>

### 4 Offline Testing in Combinatorial Optimization Problems | 99

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Submodular Set Functions and Polymatroids</td>
</tr>
<tr>
<td>4.2</td>
<td>Examples of LPPs</td>
</tr>
<tr>
<td>4.2.1</td>
<td>K-Selection</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Scheduling</td>
</tr>
</tbody>
</table>
B Proofs for Chapter 3 163

B.1 Proof of Lemma 3.4.7 ......................................................... 163
B.2 Proof of Lemma 3.4.9 ......................................................... 164
B.3 Proof of Lemma 3.4.10 ....................................................... 166
B.4 Proof of Lemma 3.4.11 ....................................................... 168
B.5 Proof of Lemma 3.4.13 ....................................................... 169
B.6 Proof of Lemma 3.4.14 ....................................................... 169
B.7 Proof of Lemma 3.4.15 ....................................................... 170
B.8 Proof of Theorem 3.4.16 .................................................... 174
B.9 Proof of Corollary 3.4.18 ................................................... 175
B.10 Proof of Theorem 3.4.19 ................................................... 175
B.11 Proof of Theorem 3.4.20 ................................................... 190

C Proofs for Chapter 4 191

C.1 Proof of Lemma 4.6.7 ......................................................... 191
C.2 Proof of Theorem 4.6.8 ....................................................... 191
## List of Figures

1-1 Thesis structure and main results. ........................................ 23
1-2 Notation for the thesis structure. ......................................... 24
2-1 Classification of jobs according to their ratio (assuming $\rho_a < \rho$). ............................ 37
2-2 Low-ratio jobs are immediately processed upon detection. ......................... 38
2-3 An example for the dynamics of the optimal policy ($\rho_a < \rho$). ...................... 40
2-4 LD DP: cost incurred under the control test one. .................................. 45
2-5 LD DP: cost incurred under the control process all. .................................. 46
2-6 An illustration of Assumption 2.5.4. ........................................ 54
2-7 The probability mass function of the Polar, Uniform, and, Binomial distributions. ............................. 62
2-8 The value of testing for different distributions and initial number of jobs. .... 63
2-9 The information revealed by testing in the original problem (a), and in the generalized problem (b). ......................... 64
3-1 Three normally distributed random variables with equal mean and different variance. .............................................. 74
3-2 Three types of discrete distributions increasing in convex order. ................... 75
3-3 An example of the mean preserving local spread. .................................... 76
3-4 A simple policy that processes three known jobs consecutively: (a) the processing order; (b) the completion time of each job (the total objective is the total grayed area). .......................... 80
3-5 The costs accumulated using different cost functions: (a) traditional cost accounting; (b) echelon cost accounting (the accumulated costs correspond to the grayed area). ........................................... 81

3-6 A classification of known jobs. Crosses and Circles denote known and unknown jobs, respectively. .............................................................. 85

3-7 The optimal control at state \((T, t + x)\) as a function of \(x\). ..................................................... 93

3-8 The structure of the optimal policy. Crosses and circles denote known and unknown jobs, respectively. ........................................... 96

4-1 Objective coefficients ordered from largest \((w_1)\) to smallest \((w_n)\). .................. 104

4-2 Example of subset of edges in a graphic matroid. .................................................. 110

4-3 A non-symmetric MST. .............................................................................................. 133

4-4 Examples of discrete (left) and continuous (right) distributions in convex order. 134

4-5 An example of the mean preserving local spread. .................................................. 135

A-1 Example of processing jobs in decreasing order of ratio. ................................. 140

A-2 The schedule under policy \(\pi\). .................................................................................. 141

A-3 Example: testing before processing low-ratio jobs .................................................. 142

A-4 Example: testing after processing a medium- or high-ratio job. .............................. 145

A-5 Example: testing when \(\rho < \rho_w\). ................................................................. 147

A-6 The solution to the model. ......................................................................................... 151

A-7 The Distribution \(\mathcal{D}_{T,W}\). .................................................................................. 151

B-1 Comparison of the policies \(\pi\) and \(\pi'\) (Lemma 3.4.9). ................................. 165

B-2 Comparison of the policies \(\pi\) and \(\pi'\) (Lemma 3.4.10). ................................. 167
Chapter 1

Introduction

Ever since Operations Research (OR) emerged and formed as a scientific discipline, the OR community has devoted great efforts to modeling and solving decision-making problems under uncertain input parameters (e.g., future demand). The challenge in these problems stems from the fact that decisions have to be made prior to knowing exact outcomes (realizations) of the unknown parameters. As a result, it is impractical to hope to find a solution or policy that are optimal with respect to each possible realization. The reigning paradigms for addressing such problems include Stochastic Optimization, in which one tries to find solutions that perform well on average or some other risk measure; Robust Optimization, in which one protects herself against the worst case within a restricted set of realizations; and Online Algorithms, in which one provides worst-case guarantees to solutions that operate under imperfect knowledge about realizations.

Each of these paradigms has generated huge body of work. However, in many practical applications one does not merely work or make decisions under uncertainty, but in fact, could also actively engage in reducing uncertainty. In this thesis, we explore common settings in which the decision-maker can either explicitly (by paying direct costs) or implicitly (by utilizing shared resources) invest in reducing uncertainty about input parameter to be realized. We refer to this investment as testing. We study the resulting tradeoff between proceeding without testing (with increased uncertainty) or reducing uncertainty (testing) to allow
the construction of improved solutions that utilize enhanced information at the expense of testing costs.

1.1 Motivating Examples

As a first example, consider aircraft engine maintenance management where the maintenance team has to repair a collection of engines. A key operational decision in this setting relates to the sequence in which engines are repaired (processed), which significantly impacts various performance measures. For example, prioritizing engines whose processing times are very long could hinder the turnover of repaired engines. Due to uncertainty about the exact nature of the breakdowns and the respective required processing times, the decision-maker cannot foretell the optimal sequence of repairs. Moreover, once the team starts working on an engine, the processing time becomes known, but due to high setup times, it is very expensive and impractical to preempt the work on an engine. Alternatively, the team could run diagnostics (tests), which requires time, but provides more accurate information about the processing times of the tested engine. Testing engines could inform better scheduling (or sequencing) decisions, but also increases the total amount of work performed by the maintenance team, which needs to decide if testing is worthwhile.

The second example is medical triage in emergency departments, a process through which patients are diagnosed in order to reveal their urgency (sensitivity to waiting), as well as the time and medical resources required to care for them. This is a considerably different area of application, but it exhibits a similar trade-off between resources (medical staff and time) used to triage patients (testing) or treat them (work). Medical triage is essentially a protocol (or a policy) to determine how resources should be allocated between treating patients and reducing uncertainties. Note that unlike maintenance, in which engines could be similar, patients could have different urgencies which could be captured by using relative weights.

The third example is concept selection in project management, where a planner needs to choose between different alternatives with uncertain time estimations (e.g., about suppliers,
designs, or implementations). The planner can commit to a certain alternative, or she could test one, that is, invest time in market research or feasibility study to reduce uncertainty about the respective alternative. The planner must therefore dynamically decide how much to invest in reducing uncertainty, as well as how to prioritize the uncertainty reduction of the different alternatives before committing to a certain alternative.

While having common theme and characteristics, these examples differ on some important aspects. The latter example of concept selection corresponds to an Offline setting in which decisions about uncertainty reduction precede the actual selection decision. In contrast, the first two examples correspond to Online settings, in which decisions about uncertainty reduction and work are intertwined (e.g., one could repair engines, then test, and repair once more). Moreover, the testing cost in the concept selection example is direct, as there is a fixed time investment for testing each alternative. In contrast, in the first two examples the testing cost is indirect, as there is no explicit cost associated with testing, but testing delays processing, which affects the overall performance indirectly.

The focus of this thesis is on stochastic optimization models with testing, a less studied, yet very common aspect that captures uncertainty reduction mechanism (also known as probing or querying (Guha et al. (2007) and Guha and Munagala (2007))). In the prevailing Bayesian framework, the evolution of uncertainty is modeled through sampling. That is, one may retrieve samples from an unknown distribution, and use them to generate a posterior distribution. Uncertainty is reduced when the number of samples increases and the unknown distribution converges to the true distribution. Similar dynamics exists in more recent works related to the tradeoff of exploration versus exploitation (for example, see Besbes and Zeevi (2009)). The common theme in these models is that individual instances are observed and used to learn the distribution of the population. Moreover, it is quite common in the Bayesian approach to assume a certain parametric structure (e.g., Gaussian or Dirichlet distributions) in order to obtain analytical and computational tractability. In contrast, the testing framework captures many practical applications in which the service process is repetitive (as is the case in both maintenance management and medical triage), and there is
abundance of data that can be used to construct reliable probability distributions of uncertain parameters (e.g., about engines and patients processing time). The issue then is not to learn unknown distributions, but rather to reduce the uncertainty in the specific realizations of various uncertain parameters drawn from the respective distributions.

1.2 Models

This thesis consists of three parts centered around interconnected yet distinct stochastic optimization problems.

The first part focuses on one of the core problems in scheduling theory, with a single server and the objective of minimizing the weighted sum of completion times of a given set of jobs. This objective reflects the goal of minimizing the weighted total (or average) wait time, which is realistic in the practical settings previously mentioned. While the distributions are known, the exact processing time and weight of each job is unknown. However, it can be revealed if the job is tested, an activity that requires a specified server time. Thus, at any stage a decision must be made whether to test another job or process a job (either one that was already tested, or one that still has uncertain processing time and weight.) Once a job is processed it must be completed (i.e., preemption is not allowed.) Without the option to test, the problem is well-studied (e.g., Smith (1956)) and can be solved optimally by processing jobs in an increasing order of their expected weighted processing times; this is known as the Weighted Shortest Processing Time rule (WSPT). In contrast, in our problem one has to dynamically decide how to allocate the server between testing and processing jobs. This is what we termed above as an online setting, in which testing decisions are intertwined with processing decisions. (We also consider more general settings where the testing time is random and where testing reveals only partial information about the realizations of the respective jobs.)

In the second part of the thesis, we generalize the model introduced in the first part and

\footnote{There is some overlap between the different chapters, such as the motivating examples and the literature review, so that each chapter could stand alone as an independent piece.}
study how to prioritize uncertainty reduction (through testing) when jobs are statistically different. For example, in the context of medical triage, two patients might be admitted for different reasons, which could indicate that the a-priori service time distributions of the two patients are different. We model the difference in uncertainties using convex orders, a general relation between distributions, which implies that the variance of one distribution is higher than the variance of the other distribution (e.g., Normal distributions with the same mean but different variance). Unlike the initial model, the decision-maker needs to decide not only whether to test or process jobs, but also which type of job to test when testing.

In the final part of the thesis\(^2\) we study uncertainty reduction in the context of a broad class of stochastic combinatorial optimization problems whose parameters are uncertain. We denote these problems as Linear Programs over Polymatroids with Testing (LPPT). These problems can be formulated as Linear Programs whose objective coefficients are random variables (from known distributions), and whose constraint polyhedron has the structure of a polymatroid\(^3\). In LPPTs, the decision-maker can either pay to test one of the random objective coefficients and observe its realization, or optimize under uncertainty (that is, determine irrevocably the value of all decision variables). The goal is to adaptively decide which objective coefficients to test before committing to a solution, so as to maximize the expected value of the LP solution minus the testing costs.

LPPTs capture a broad class of known combinatorial optimization problems. One example of LPPT is the offline variant of the scheduling problem discussed in the previous chapters (when there are no weights). In the offline variant, testing jobs must precede the processing of jobs (e.g., the maintenance team first diagnoses engines, and then creates a finalized work-plan for the repairs). While this is an offline setting, optimal testing decisions might still be adaptive. That is, testing could depend on the realizations of previously tested jobs. Other examples of LPPTs include: (1) concept selection in project management

\(^2\) The work in this chapter was carried out jointly with Chen Attias from the Weizmann Institute of Science. In particular, the results appear also in her MS thesis.

\(^3\) Polymatroid is a polytope associated with a submodular function, in which constraints bound the summation of each subset of decision variables by the submodular function applied to the respective subset. Despite having exponentially many constraints, polymatroid optimization problems do not require standard LP solution methods and can be solved efficiently using a greedy algorithm (see Schrijver (2003)).
- choosing the best (or \( k \) best) element from a set, when each element has a-priori random value that could be tested for a fixed cost. The goal is to maximize the expected value of the chosen element minus the testing costs; (2) the Maximal Spanning Tree (MST) problem (Graham and Hell (1985)) – given a weighted graph, the goal in MST is to find a tree that connects all nodes whose total edge weights are maximal. In MST with Testing, the weight of each edge is a random variable and one can test edges to observe their respective weights before forming the tree. The goal is to maximize the expected weight of the selected spanning tree minus the testing costs; and (3) queueing problems that satisfy Conservation Laws (Shanthikumar and Yao (1992)) – for example, prioritizing the processing of waiting jobs from different classes in an M/M/1 queue, with the objective of minimizing the average cost of waiting (a per-unit of time wait cost is associated with each of the job classes). When the wait costs are random variables that can be tested before the queue starts operating, the problem can be formulated as an LPPT. Note that, in contrast to models studied the first two chapters, the LPPT is an offline testing model (as illustrated by the prior examples).

1.3 Methods

We use Dynamic Programming (DP) as our modeling framework. While natural DP formulations are intractable due to high dimensionality, we use these formulations to derive structural properties about the optimal policy, and to obtain more compact formulations. Beyond using standard DP techniques (such as proof by induction, and rounding to obtain an approximate dynamic program), we make two important methodological contributions:

1. In Chapter 2, we propose a new marginal cost accounting scheme, which we leverage to reduce the dimensionality of the state space. Unlike traditional cost accounting schemes, in which the contribution of a job to the overall objective function is accounted for at the moment when the processing of this job is completed, in marginal cost accounting its contributions to the completion times of other jobs are accounted once the relative scheduling order between jobs is determined. This is an intuitive method
that seems natural in analyzing stochastic optimization problems with uncertainty reduction.

2. In Chapter 3, we use an analysis based on the concept of *mean preserving local spread* (Müller and Stoyan (2002)). This enables reducing the comparison of functions of random variables in convex order to a simpler comparison between locally different distributions. We believe that this general technique could be useful in the analysis of other problems, in particular for Dynamic Programs in which probability distributions are in convex order (and potentially also other types of stochastic orders).

1.4 Results

In each of the thesis chapters, we characterize the optimal policy of an operational problem which includes underlying uncertainty that can be reduced. We describe the optimal policies using intuitive thresholds and scheduling rules, and provide optimal and near-optimal algorithmic solutions. Moreover, we show some of the optimal scheduling rules can be interpreted as *myopic rules* which base decisions about testing by computing the expected gain from a single test (which can be computed efficiently, as opposed to computing the gain from an non-restricted number of tests). Moreover, the analysis proves that in many cases, one should consider testing only the job (or objective coefficient) with the highest myopic gain, and only when the respective gain is positive. We next highlight the main result of each chapter.

In Chapter 2, we show that while a natural formulation of the problem leads to a high-dimensional DP, we can provide structural analysis that obtains a characterization of optimal policies that is managerially intuitive. Specifically, we explicitly identify (and compute) two thresholds that induce a partition of the tested jobs into three groups. The first group should be processed immediately with no delay. The second group should be processed last after all other jobs are processed. Finally, unknown jobs can be potentially tested only before known jobs from the third group are processed. We also show that the optimal policy has a
structure of an optimal stopping time problem; it tests jobs and processes immediately jobs in the first group until at some point it switches to processing all remaining jobs using the WPST rule, and never tests again. Based on the structural characterization of the optimal policies together with the marginal cost accounting scheme, we propose a low-dimensional DP formulation that unlike the natural high-dimensional one can be solved efficiently. Moreover, the structural properties of the optimal policy lead to a low-dimensional DP formulation that can be solved near-optimally for any specified degree of accuracy. Specifically, we describe a fully polynomial approximation scheme (FPTAS). Moreover, under a certain condition (which includes the special case of equally weighted jobs), the optimal policy is shown to be a myopic rule that can be based merely on current state. In addition, the analysis provides insights into the value of testing as a function of the various parameters of the problem, as well as analytically assesses the performance of simpler policies. These could be used to assess when the testing functionality is worthwhile. Finally, the analysis extends to broader settings, in which testing might reveal only partial information about the latent attributes.

In Chapter 3, we show that the structure of the optimal policy when jobs have different distributions, generalizes the optimal policy of the base model with identical jobs that we analyzed in Chapter 2. We show that when the processing time distribution of each job is different (i.e., in convex order), there exist testing thresholds associated with each of the probability distributions (rather than a single testing threshold in the base model). The testing thresholds govern the timing of testing and processing in the following way. First, we process the jobs whose durations are shorter than the smallest testing threshold. Then, we either test the unknown job corresponding to the smallest testing threshold and return to step 1, or simply process all jobs without ever testing again. This structure generalizes the optimal stopping time interpretation for the base model. Moreover, we show that the decision about whether one should test or process all jobs can be determined optimally using the same myopic rule. The optimal policy is intuitive and suggests that one should test jobs in a decreasing convex order. That is, jobs with higher uncertainties are preferred for testing.

In Chapter 4, we analyze different cases of LPPTs and show that, similarly to the schedul-
<table>
<thead>
<tr>
<th>Ch.</th>
<th>Base (deterministic) Problem</th>
<th>Uncertain Quantities</th>
<th>Testing</th>
<th>Main Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$\sum w_i C_i$</td>
<td>$(T_i, W_i) \sim \mathcal{D}$</td>
<td>Online</td>
<td>Intuitive structural properties</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_i \leq_{cx} T_{i+1}$</td>
<td></td>
<td>Marginal cost accounting</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Low-dimensional DP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Optimality of a myopic policy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FPTAS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Quantifying the value of testing</td>
</tr>
<tr>
<td>3</td>
<td>$\sum C_i$</td>
<td>$T_i \sim \mathcal{D}$</td>
<td>Online</td>
<td>A generalized optimal myopic policy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$T_i \leq_{cx} T_{i+1}$</td>
<td></td>
<td>An analysis based on the mean preserving local spread</td>
</tr>
<tr>
<td>4</td>
<td>$\max_{x \in P_f} \mathbb{E} \left[ \sum W_i x_i \right]$</td>
<td>$W_i$</td>
<td>Offline</td>
<td>$W_i$ have the same mean - optimality of a myopic stopping rule</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MST - optimality of a myopic stopping and testing rules</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Symmetric $P_f$ and $W_i \leq_{cx} W_{i+1}$ - optimality of both stopping and testing rules</td>
</tr>
</tbody>
</table>

Figure 1-1: Thesis structure and main results.

In the previous chapters, the myopic policy also plays a pivotal role in describing the optimal policy for solving LPPTs. In particular, for LPPTs whose objective coefficients are independent random variables that have the same mean value, the decision when to stop testing can be determined myopically. In addition, for MST with Testing where the edge weights (i.e., the objective coefficients of the respective LPPT) are independent and identically distributed random variables, a myopic rule is optimal not only in deciding when to stop testing, but also in choosing which edge to test next. Finally, we show that these myopic rules are also optimal for LPPTs that satisfy a certain type of symmetry even when the objective coefficients relate in convex order. Table 1-1 outlines the thesis structure and summarizes the main results of each chapter (see Table 1-2 for the notation).
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_i$</td>
<td>Processing duration of job $i$</td>
</tr>
<tr>
<td>$C_i$</td>
<td>Completion (departure) time of job $i$</td>
</tr>
<tr>
<td>$w_i$</td>
<td>Weight of job $i$ (a per unit of time penalty for waiting)</td>
</tr>
<tr>
<td>$\leq_{cr}$</td>
<td>The convex order relation</td>
</tr>
<tr>
<td>$1</td>
<td></td>
</tr>
<tr>
<td>$1</td>
<td></td>
</tr>
<tr>
<td>$(T_i, W_i) \sim \mathcal{D}$</td>
<td>The joint processing time and weight distribution (the same for all jobs)</td>
</tr>
<tr>
<td>$T_i \sim \mathcal{D}_i$</td>
<td>The processing time distribution of job $i$ (varies by job)</td>
</tr>
<tr>
<td>$P_f$</td>
<td>A polymatroid (polytope) associated with the (submodular) function $f$</td>
</tr>
<tr>
<td>$W_i$</td>
<td>The objective coefficient associated with decision variable $x_i$</td>
</tr>
<tr>
<td>$\bar{x}$</td>
<td>The vector of decision variables</td>
</tr>
</tbody>
</table>

Figure 1-2: Notation for the thesis structure.
Chapter 2

Scheduling with Testing

In this chapter, we study a new class of scheduling problems that capture common settings in service environments, in which one has to serve a collection of jobs that have a-priori uncertain attributes (e.g., processing times and priorities) and the service provider has to decide how to dynamically allocate resources (e.g., people, equipment and time) between testing (diagnosing) jobs to learn more about their respective uncertain attributes and processing jobs. The former could inform future decisions, but could delay the service time for other jobs, while the latter directly advances the processing of the jobs but require making decisions under uncertainty. While a natural Dynamic Programming formulation has a high-dimensional state space, we derive structural properties of the optimal policies and use a new cost-accounting scheme to devise a surprisingly low-dimensional DP formulation. These ultimately lead to a complete characterization of the optimal policy when jobs are equally weighted, and to a fully polynomial time approximation scheme (FPTAS) in general.

This chapter is organized as follows. In Section 2.1 we introduce the problem, discuss the motivating examples, highlight our main results, and review related work. In Section 2.2 we describe the model, the new cost accounting scheme, and the resulting DP formulation. Section 2.3 contains an analysis of the model and the characterization of the optimal policy, a characterization that we then use in Section 2.4 to develop algorithms that solve the problem near-optimally. In Section 2.5 we study a myopic policy and prove its optimality under a
certain assumption. In Section 2.6 we discuss the value of testing, and in Section 2.7 we
generalize the model to a broader settings and show that the results of the basic model still
hold. We conclude in Section 2.8.

2.1 Introduction

Effective management of many service systems often relies on the ability to appropriately
classify and prioritize customers, tasks or jobs. However, in many settings the exact nature
of the various jobs is uncertain; for example, the time and amount of resources required
to process a given job and its relative priority might not be known exactly. While recent
advancements in information technologies enable obtaining far more accurate predictions
about each job, there are still many settings, in which collecting more information on a job
requires the allocation of the same resources used to process the job. This gives rise to oper-
tional tradeoffs of exploration versus exploitation, specifically, how to dynamically allocate
resources between diagnostic work called testing that aims to collect more information on
the arriving jobs, and processing work called working that simply serves the jobs (customers)
in the systems. In this chapter, we introduce new scheduling models that capture these
tradeoffs, and provide some structural results and insights on the optimal policies as well as
algorithmic results on how to obtain optimal and provably near-optimal policies. Surpris-
ingly, in many interesting cases the optimal policy can be described through myopic (local)
rules.

One relevant example for the type of tradeoff studied in this chapter arises in aircraft
maintenance. Engine repair requires disassembling and reassembling engines, which are
costly in terms of time. Alternatively, engines can be diagnosed using special testing equip-
ment, which can unveil the nature of breakdowns, and the required corrective measures and
processing times. The shared resource in this case between working and testing are the
maintenance personnel. Another example arises in emergency medical departments. In this
setting, patients undergo the process of triage that aims at collecting information about
their urgency (sensitivity to waiting), as well as the required activities and processing times.
This information allows prioritizing patients to ensure efficient allocation of limited medical resources. While these examples stem from considerably different practices, they both give rise to similar tradeoffs, specifically, how resources should be allocated between diagnostics and actual processing of jobs.

The chapter is focused on one of the core problems in scheduling theory, with a single server and the objective of minimizing the weighted sum of completion times of a given set of jobs. This objective reflects the goal of minimizing the weighted total (or average) wait time, which is realistic in the practical settings mentioned above and in many others, in which the tradeoff of testing versus working exists. The processing time and weight of the jobs is unknown but can be revealed if the job is tested, an activity that requires a specified server time. Thus, at any stage a decision has to be made whether to test another job or process a job (either one that was already tested, or one that still has uncertain processing time and weight.) Once a job is processed it has to be completed (i.e., preemption is not allowed.) We note that without the option to test, the problem is known to be solved optimally by processing jobs in an increasing order of their expected weighted processing times; this is known as the Weighted Shortest Processing Time rule (WSPT).

Contributions. This thesis chapter makes several important contributions. First, it introduces a new class of scheduling models that capture exploration versus exploitation tradeoffs in service environments. While it is widely recognized that understanding and controlling variability could be critical for sustaining uninterrupted operations, to the best of our knowledge, this is the first work that studies the extent to which resources should be utilized to collect information and reduce uncertainty. Second, while a natural formulation of the problem leads to a high-dimensional Dynamic Program (DP), the chapter provides structural analysis that obtains a characterization of optimal policies, which is managerially intuitive. Specifically, we explicitly identify (and compute) two thresholds that induce a partition of the tested jobs into three groups. The first group should be processed immediately with no delay. The second group should be processed last after all other jobs are processed. Finally, unknown jobs can be potentially tested only before known jobs from the third group are
processed. We also show that the optimal policy has a structure of an optimal *stopping time* problem; it tests jobs and processes immediately jobs in the first group until at some point it switches to processing all remaining jobs using the WPST rule, and never tests again. Third, based on the structural characterization of the optimal policies together with an innovative *marginal cost accounting scheme*, we propose a low-dimensional DP formulation that unlike the natural high-dimensional one can be solved efficiently. Unlike traditional cost accounting schemes, in which the contribution of a job to the overall objective function is accounted for at the moment when the processing of this job is completed, in marginal cost accounting its contributions to the completion times of other jobs are accounted once the relative scheduling order between jobs is determined. Moreover, the structural properties of the optimal policy lead to a low-dimensional DP formulation that can be solved near-optimally for any specified degree of accuracy using a *fully polynomial approximation scheme* (FPTAS). Fourth, under a certain condition (which includes the special case of equally weighted jobs), the optimal policy is shown to be a *myopic rule* that can be based merely on current state. Fifth, the analysis provides insights into the value of testing as a function of the various parameters of the problem, as well as analytically assesses the performance of simpler policies. This analysis could be used to better understand and assess when the testing functionality is indeed worthwhile. Finally, the analysis extends to broader settings, in which testing might reveal only partial information about the latent attributes.

**Literature review.** For more than half a century the research community has developed a rich and extensive literature in the area of scheduling. Nevertheless, despite the wide spectrum of problems that has been explored, the topic of testing per se seems to have not receive attention. Typical features of scheduling problems concern aspects such as individual job properties (e.g., processing times, due dates, release dates and preemption), dependencies between jobs (e.g., precedence constraints, families of jobs, setup times), server properties (e.g., multiple servers, control of speed, batch processing, and breakdowns), server-job settings (e.g., flow shop, job shop, and open shop problems), under a variety of objectives (e.g., makespan, flow-time, lateness, and tardiness). The literature review of this chapter does not
attempt to present a comprehensive survey of this enormous body of knowledge, but rather,
present the main research areas within the scheduling literature, and their relation to our
work. For a comprehensive treatment of the subject, the reader is referred to Pinedo (2012a)

A principal way of classifying scheduling work is according to the amount of information
known to the scheduler. The main categories are deterministic, stochastic, and online. The
three decrease in the availability of information. In deterministic problems, all the informa-
tion is known in advance, which implies that decisions can be made in advance, and that the
overall performance can be predicted. Stochastic scheduling assumes a probabilistic charac-
terization of uncertain data. On the far extreme lies online scheduling, where no knowledge
regarding the processing or the arrival time to the system is assumed, and information is
revealed gradually. The models studied in the chapter share properties of both stochastic
and online scheduling problems. On one hand, probability distributions are used to model
uncertainties, but on the other hand, testing is allowed as a means of learning about these
uncertainties in an online fashion.

Two domains that are closely related to our model are medical Triage, and maintenance.
Over the last decade, a series of papers appeared that challenge the current practices of Triage
(e.g., Sacco et al. (2005), Lerner et al. (2008) and Jenkins et al. (2008)). One of the main
critiques being that resource availability is not taken into consideration when deciding on
patients priorities. At the same time, new work on models and heuristics, offered insights and
alternatives for creating a better triage process (e.g., Sacco et al. (2005), Li and Glazebrook
(2010), Jacobson et al. (2012), and Mills et al. (2013)). Typically however, the basic premise
in these type of work is that information about patients is available (whether partial or
complete), and the goal is to optimize the allocation of resources given the information.
In our work, in addition to deciding on the allocation of resources to serve patients, we
also consider the process of determining the condition of each patient, which also consumes
resources (yet in a rather different setting). Alizamir et al. (2013) studied a model where
diagnosis accuracy can be controlled (e.g., by investing more time in diagnosing), and there
is a tradeoff between the diagnosis accuracy and patient delays. Whereas the latter work is concerned with diagnosis accuracy, our focus is on determining the sequence in which patients should be served, which is an important lever in improving various performance measures in many service systems.

In the literature on maintenance, a notable effort has been devoted to studying maintenance problems with inspections, in a class of models that are known as preparedness models (McCall (1965)). In these models, machines deteriorate over time in a process that is hidden unless inspection is employed, which reveals the true state of the machine. Most of the related literature on inspection models focused on single-component systems in an infinite horizon setting, where breakdowns are invisible without inspections. Moreover, the assumptions are that inspections are costly and that their time is negligible, and the objective is to minimize costs. That is, the focus is on costs rather than on allocation of limited capacity. In multi-component systems, the work has mostly been on harnessing economies of scale to reduce maintenance costs by simultaneously repairing multiple components. In addition, models were created for systems in which a correlation exists between the evolution of components, or where components are jointly maintained due to structural dependencies.


We also note several results that study the value of information in a single server queuing and scheduling settings. Bansal (2005), studied an $M/M/1$ queue, in which job durations are known upon arrival. The latter work quantifies the improvement of a policy that processes jobs in an increasing order of the remaining processing time, over the standard first comes first served policy. Wierman and Nuyens (2008) studied a class of policies that generalizes the shortest processing time rule. These are used in practice when jobs with different pro-
cessing times need to be grouped and assigned the same priority rule (which is similar to not having the exact information about the processing times). They derive bounds for multiple performance measures, and investigate how the bounds are affected by the accuracy of the information.

The tradeoff of exploration vs. exploitation has been studied in the context of several operational problems in revenue management and supply chain management (for example, see Besbes and Zeevi (2009) and Besbes and Muharremoglu (2013)). However, the typical assumption in this stream of work is that the underlying distributions are unknown and learnt from data. In contrast, in our setting the distributions are assumed known but specific instances from these distributions can be observed through testing.

The novelty of our work is in incorporating learning decisions into job scheduling problems. Traditionally, scheduling problems focused on determining the optimal sequence of jobs processing in a deterministic environment, or subject to uncertainties that are represented by probability distributions. However, to the best of our knowledge, the issue of testing has not been studied in published literature (the recent work of Sun et al. (2014) studied a very special case of this model).

2.2 Mathematical Formulation

Consider $N_0$ jobs that need to be processed by a single non-preemptive server. Each job $i$, is associated with a given processing time $t_i$ as well as a weight $w_i$ that represents the relative importance of the job. The duration $t_i$ and weight $w_i$ of job $i$ are a-priori random variables $(T, W)$ distributed according to a known joint distribution $\mathcal{D}$ with support $[1, D] \times [1, V]$, and are independent and identically distributed across jobs.

When the server becomes idle, the scheduler could do one of the following. It can process a job, in which case the the processing time $T$ and the weight $W$ of the job are realized. However, the job must be processed with no preemption. Alternatively, the server can be used to test a job, which requires a specified processing time $t_a$, and reveals the required processing time and weight of the specific job. After testing, the job could be put on hold.
and processed later. Thus, whenever the server becomes idle, three decisions are available: process one of the “known” jobs (i.e., process a job that was tested), process an “unknown” job (i.e., process a job that was not tested), or test an “unknown” job. Note that both known and unknown pertain to jobs that have not yet been processed.

The system’s state can be expressed as a vector \((N, [t_1, w_1, ..., t_n, w_n])\), where \(N\) and \(n\) denote the number of unknown and known jobs, respectively, and \(t_1, w_1, ..., t_n, w_n\) denote the realization of the processing times and weights of each of the \(n\) known jobs. When there are no known jobs, the system state is simply \((N, [])\). Without loss of generality, we always assume that the ratio \(t_i/w_i\) is non-decreasing in \(i\). We denote by \(\rho = \mathbb{E}[T]/\mathbb{E}[W]\) and \(\rho_i = t_i/w_i\) the processing time to weight (importance) ratio for an unknown job and a given tested job \(i\), respectively. The action space can be described by the set \(\{\text{test}, \text{process}_u, \text{process}_i\}\). The controls refer to testing an unknown job, processing an unknown job, and processing the known job \(i\). The goal is to find an adaptive scheduling policy that minimizes the expected weighted sum of completion times. This will be denoted as the S&T model (Scheduling with Testing).

Before presenting a DP formulation for the problem, we show in Section 2.2.1 below that a variant of the WSPT rule extends to our problem. This allows introducing a marginal cost accounting scheme in Section 2.2.2 which is then used to obtain a DP formulation for the problem (Section 2.2.3).

### 2.2.1 The WSPT Rule

The deterministic variant of the problem studied in this chapter is known to be solved optimally by the policy that processes jobs in a non-decreasing order of their ratio (a.k.a., the WSPT rule or Smith’s rule, see Smith (1956)). A different (dynamic) view of this rule is that the optimal policy always selects for processing the job with the lowest ratio.

In this section, it is shown that a weaker version of this property holds for the S&T model (this will later be extended in Section 2.3). Specifically, we show that when processing, it is optimal to process a known job when its ratio is less than the ratio of an unknown job.
Note that while this property is usually proven using a simple interchange argument, in the S&T model test actions can take place between processing of any two jobs. In such a case, interchanging the two jobs to form a decreasing ratio order might no longer guarantee an improved scheduling policy. Moreover, by testing, we observe the true ratio of jobs, which might also affect the optimal scheduling order.

In Lemma 2.2.1 below, we show that given that two jobs that are processed consecutively, their ratio must be non-decreasing. We then prove in Lemma 2.2.2 a stronger property that given two jobs do not undergo testing, it is sub-optimal to process the job with the higher ratio before processing the job with the lower ratio (even when the jobs are not processed consecutively).

**Lemma 2.2.1.** Processing consecutively jobs with a decreasing ratio is sub-optimal.

*Proof.* See Appendix A.1.

**Lemma 2.2.2.** Processing a job (known or unknown) with a ratio higher than the ratio of a known job is sub-optimal.

*Proof.* See Appendix A.2.

Note that Lemma 2.2.2 significantly reduces the actions space. At any state, we need to choose only between testing or processing an unknown job, and processing the known job with the smallest ratio.

### 2.2.2 Marginal Cost Accounting

Marginal cost accounting is related to the concept of Linear Ordering (see Queyranne and Schulz (1994) and the references therein). For the problem without testing, policies are described by linear ordering using the processing order of any pair of jobs. The objective value can then be written as $\sum_{i=1}^{n} t_i w_i + \sum_{i\neq j} (1_{i<j} t_i w_j)$, where $1_{i<j}$ is the indicator function for the event that job $i$ is processed before job $j$. We see that each job $i$ contributes its processing time to itself ($w_i t_i$), and to every job $j$ that is processed afterward ($t_i w_j$).
When jobs can be tested, one has to consider in addition the further delays caused by testing. For job \( i \), the delays due to testing are \( t_a \) times the number of tested jobs prior to job \( i \). With hindsight, we can write the objective value as follows:

\[
J_{\text{mrg}} (N_0, \emptyset) = \sum_{i=1}^{N_0} t_i w_i + \sum_{i \neq j} (1_{i<j} t_i w_j) + \sum_{i=1}^{N_0} w_i t_a (\# \text{ of tested jobs prior to job } i) . \tag{2.1}
\]

Lemma 2.2.2 implies that as jobs are tested and their respective values \( t_i, w_i \) become known, the optimal sequence of processing is partially determined. Therefore, some of the future costs can be computed at the time of testing. More generally, in marginal cost accounting, we charge all of the future costs that become known due to present action.

Specifically, when at state \((N, [t_1, w_1, ..., t_n, w_n])\) an unknown job \( l \not\in \{1..n\} \) is tested and the values \((t_l, w_l)\) are realized, then the delays caused by testing, \( t_a (\Sigma_{i=1}^{n} w_i + (N-1) \mathbb{E}[W] + w_l) \), and the ordering costs with respect to other known jobs, \( t_l w_l + \Sigma_{i=1}^{n} (1_{\rho_l < \rho_i} (t_l w_i) + 1_{\rho_l > \rho_i} (t_l w_l)) \) can be charged. Furthermore, if an unknown job is processed, the ordering costs with respect to all other jobs, \( \mathbb{E}[TW + \Sigma_{i \neq 1}^{n} Tw_i + (N-1) T\mathbb{E}[W]] \) can be charged. This include the “self imposing cost” \( TW \), the costs associated with known jobs \( \Sigma_{i=1}^{n} Tw_i \), and the costs associated with the other \( N-1 \) unknown jobs, which on expectation are \((N-1) T\mathbb{E}[W] \) (we use the independence between jobs).

Finally, when known job 1 is processed, the additional costs are the ordering costs of job 1 with respect to the unknown jobs: \( N t_1 \mathbb{E}[W] \). Note that other ordering costs have been already accounted for by the previous actions.

### 2.2.3 DP Formulation

In this section we describe a DP formulation for the problem. The state of the DP is represented by the vector \((N, [t_1, w_1, ..., t_n, w_n])\) defined previously. From Lemma 2.2.2 we can restrict the control space to \{test, process\_u, process\_1\}. The transitions are straightforward, whereas processing an unknown job decreases \( N \) by 1; processing job 1 removes job 1 from the state; testing an unknown job, decreases \( N \) by 1, and adds a known job to the state.
with the realizations of the processing time and weight \((T, W)\). This occurs with probability defined by the distribution \(D\). Using the marginal cost account scheme (Section [2.2.2]), we define the Bellman’s equation as follows:

\[
J_{\text{mrg}} (N, [t_1, w_1, ..., t_n, w_n]) = \min \begin{cases} 
\mathbb{E}[TW] + (\sum w_i + N\mathbb{E}[W]) t_a + & \text{test} \\
+\mathbb{E}\left[\sum_{i=1}^n \min \{W t_i, w_i T\}\right] \\
+\mathbb{E}[J_{\text{mrg}} (N-1, [t_1, w_1, ..., t_n, w_n] \cup \{T, W\})] \\
\end{cases} \\
+\mathbb{E}[TW] + (\sum w_i + (N-1) \mathbb{E}[W]) \mathbb{E}[T] + & \text{process}_u \\
+J_{\text{mrg}} (N-1, [t_1, w_1, ..., t_n, w_n]) \\
N\mathbb{E}[W] t_1 + J_{\text{mrg}} (N, [t_2, w_2, ..., t_n, w_n]) & \text{process}_1 \\
\end{cases}
\]

\[J_{\text{mrg}} (0, [t_1, w_1, ..., t_n, w_n]) = 0.\]

(2.2)

Since \(T\) and \(W\) are random, the transition to the next system state is random, and as a result, the cost-to-go is captured through expectation over all possible states. Observe that this DP formulation has a high-dimensional state space that is likely to explode, making it computationally intractable to solve.

While the marginal cost accounting method may seem less straightforward than the traditional method in which completion times are added after jobs are processed, the marginal cost accounting method has the advantage of accounting costs at the earliest possible moment. Intuitively, this means that we need to encode less information in the DP states which would allow us to formulate a more compact DP. More specifically, using the marginal cost accounting method and several structural properties on the optimal policy (Section [2.3]), we formulate the problem as a low-dimensional DP (Section [2.4]), and show that 5 statistics of the unknown and known jobs are sufficient to account for all future costs (in contrast to the
2.3 Properties of the Optimal Policy

In this section, the DP formulation of described in Section 2.2.3 is leveraged to characterize structural properties of optimal policies. These are then used to devise a low-dimensional DP formulation.

We start by introducing a new quantity $\rho_a$, that together with $\rho = \mathbb{E}[T] / \mathbb{E}[W]$, will be key to characterizing the optimal policy.

**Definition 2.3.1.** The testing ratio $\rho_a$ is defined as the unique solution to the equation:

$$\rho_a = x : t_a - \mathbb{E}[(xW - T)^+] = 0.$$  \hfill (2.3)

Lemma 2.3.2 below shows that $\rho_a$ is well defined.

**Lemma 2.3.2.** (1) The function $f(x) = t_a - \mathbb{E}[(xW - T)^+]$ is non-increasing in $x$. Moreover, $f(x)$ is strictly decreasing in $x \geq \inf \{d/v : \text{Prob}(T = d, W = v) > 0\}$, where its value is $t_a > 0$.

(2) The solution to $t_a - \mathbb{E}[(xW - T)^+] = 0$ is unique.

(3) If $x < \rho_a$ then $t_a - \mathbb{E}[(xW - T)^+] > 0$; if $x > \rho_a$ then $t_a - \mathbb{E}[(xW - T)^+] < 0$.

(4) $\rho_a < \rho \iff t_a < \mathbb{E}[(\rho W - T)^+]$.

**Proof.** Straightforward.

The quantity $\rho_a$ has the intuitive meaning of the minimal job ratio, for which testing earlier is favored to testing later. As we will see, $\rho_a$ is a trigger point for testing unknown jobs; specifically, we will show that it is never optimal to test an unknown job after a known job $i$ with $\rho_a < \rho_i$ or before a known job with $\rho_a > \rho_i$. Similarly, $\rho$ will serve as a trigger point for processing unknown jobs, and we will show that it is never optimal to process unknown jobs after a known job $i$ with $\rho < \rho_i$, or before a known job $i$ with $\rho > \rho_i$. 

above $N$ dimensional DP formulation).
Using $\rho$ and $\rho_a$ we can divide known jobs into three groups: (i) **low-ratio** jobs ($\rho_i < \min(\rho, \rho_a)$); (ii) **medium-ratio** jobs ($\min(\rho, \rho_a) < \rho_i < \max(\rho, \rho_a)$); and (iii) **high-ratio** jobs ($\rho_i > \max(\rho, \rho_a)$). For ease of exposition we assume that $\rho_a \neq \rho$ (no loss of generality), and that for each $i$, we have $\rho_i \neq \rho_a$ and $\rho_i \neq \rho$. (This is with a slight loss of generality that can be easily resolved but hinders readability.) For state $(N, [t_1, w_1, ..., t_n, w_n])$, denote the set of low/medium/high ratio jobs by $S_{Low}/S_{Med}/S_{High}$, respectively. Observe that these sets are state-dependent.

Figure 2-1 illustrates this classification of jobs assuming that $\rho_a < \rho$. Jobs are ordered on the axis shown according to their ratio. Unknown jobs are denoted by a circle, and known jobs are denoted by “x”. In this example, there are four unknown jobs, two low-ratio job, three medium-ratio jobs, and two high-ratio jobs.

![Figure 2-1: Classification of jobs according to their ratio (assuming $\rho_a < \rho$).](image)

The next lemma shows that the low-ratio jobs have the highest priority and are the first to be processed with no further delays immediately after they were tested.

**Lemma 2.3.3.** For any state $(N, [t_1, w_1, ..., t_n, w_n])$, in which job 1 has a low-ratio, processing job 1 is the only optimal control.

*Proof.* See Appendix A.3.

Lemma 2.3.3 implies that low-ratio jobs should be processed immediately upon testing. For example, in the state illustrated in Figure 2-1 it is optimal to process the shortest two jobs and transition to the state illustrated in Figure 2-2. Without loss of generality, we can always assume that $\rho_a < \rho_1$ or $\rho < \rho_1$. That is, under an optimal policy there is never a state with a low-ratio job.
We now consider two cases separately: (1) $\rho_a < \rho$, and (2) $\rho_a > \rho$. As the parameter $\rho_a$ monotonically increases with the testing time (Definition 2.3.1), we denote the two cases by short and long testing times. In the first case (Section 2.3.1), we show that any testing must precede any processing of medium-ratio, high-ratio, and unknown jobs. Consequently, all testing should be done immediately after processing low-ratio jobs, and once we stop testing, all remaining jobs are processed in non-decreasing order of their ratio, essentially following the WSPT rule. For the second case (Section 2.3.2), we show that unknown jobs should never be tested. Therefore, the problem is reduced to the traditional problem (without testing), of minimizing the weighted sum of completion times.

### 2.3.1 Short Test Time ($\rho_a < \rho$)

Using the assumption that $\rho_a < \rho$, we first prove a local optimality condition that pertains to testing immediately after processing. Despite being local, this result together with the previous lemmas and properties impose a significant amount of structure on the optimal policy.

**Lemma 2.3.4.** If $\rho_a < \rho$, then testing immediately after processing jobs with ratio higher than $\rho_a$ is sub-optimal.

**Proof.** See Appendix A.4

As an example of Lemma 2.3.4 consider the state illustrated in Figure 2-2. Lemma 2.3.4 states that testing a job after processing any subset of the remaining known and unknown jobs is not optimal. We conclude that when $\rho_a < \rho$, the optimal policy always processes
all low-ratio jobs before testing, and generally operates in two phases. In the first phase, unknown jobs are tested, and processed immediately only if they have a low-ratio. In the second phase, all the jobs in the system are processed in a non-decreasing order of their ratio. This means that the problem can be seen as a stopping problem, where the decision to continue corresponds to testing an unknown job (and processing if the respective job has a low-ratio), and stopping corresponds to processing all remaining jobs.

Figure 2-3 illustrates the optimal policy through a sequence of actions that results in transitions between four states denoted by A-D. In Figure 2-3a we see state A – the current state of the system. The optimal policy then needs to decide between two courses of actions: either stop (process all jobs according to the WSPT rule) or test an unknown job. Suppose that it is optimal to test. In this case, we test an unknown job and transition to state B (Figure 2-3b). Observe that now we have one less unknown jobs, and that the tested job (numbered as 1) has medium-ratio. Once again, we need to decide between stopping and testing. Suppose that once again it optimal to test. We test the unknown job and transition to state C (Figure 2-3c). In state C there are only two unknown jobs left and an additional low-ratio job (numbered as 2). The low-ratio job is processed immediately which leads us to state D (Figure 2-3d). Once again we need to choose between stopping and testing. Assuming that stopping is optimal, we process all jobs according to their ratio, that is, according to their location on the axis from left to right.

These results are summarized in the following theorem.

**Theorem 2.3.5.** For $\rho_a < \rho$, the dynamics of the optimal policy are:

1. Process all jobs with ratio below $\rho_a$ in a non-decreasing order of their ratio;
2. Either process all remaining jobs in a non-decreasing order of ratio, or, test a job and go back to (1).

(when we process all the jobs in non-decreasing order of their ratio (in case 2), we use $\rho$ as the ratio for all unknown jobs and thus can process them all consequentially).

**Proof.** Immediate from Lemmas 2.3.3 and 2.3.4. ☐
Interestingly enough, the form for the optimal solution bears a close resemblance to current practices of emergency departments. The highest priority is given to urgent patients (high weight), and to cases that can be quickly resolved (low processing times). Other cases are triaged (tested) and put on hold. This may suggest that the triage model should be considered in other industries (possibly adjusted to the specific domains).

An alternative interpretation for lemmas 2.3.3 and 2.3.4 is that of an interchange property of testing. Similarly to the way the interchange argument reorders jobs in the problem without testing, the two lemmas reorder testing so that it is performed after processing low-ratio jobs, and before processing any other job.

Note that some of the questions are yet unanswered. Mainly, should we test or process all jobs after all low-ratio jobs have been processed (that is, when to stop).
2.3.2 Long Test Time ($\rho < \rho_a$)

When $\rho < \rho_a$, we show that testing is always sub-optimal, which implies that the problem reduces to the traditional problem without testing.

**Theorem 2.3.6.** If $\rho < \rho_a$, then for every state $(N, [t_1, w_1, ..., t_n, w_n])$ the optimal policy processes all jobs in a non-decreasing order of their ratio (i.e., testing is never optimal).

**Proof.** See Appendix A.5.

While a basic assumption of the model is that in the initial state of the system there are $N_0$ jobs, Theorem 2.3.6 (and all other lemmas and theorems) holds even when the starting state contains known jobs.

Note that when $\rho < \rho_a$, the optimal policy of processing all jobs in a non-decreasing order of their ratios is a special case of the optimal policy when $\rho_a < \rho$, in which testing should not be performed.

2.4 Solutions and Algorithms

In this section, we develop an efficient algorithmic solution to the problem. In Section 2.4.1 we use the properties of the optimal policies proven in Section 2.3 to obtain a low-dimensional DP formulation. In Section 2.4.2 we analyze the new formulation, and in Section 2.4.3 we use it to develop an approximation scheme.

2.4.1 Low Dimensional DP Formulation

We start by defining several statistics for an arbitrary state $(N, [t_1, w_1, ..., t_n, w_n])$:

- $\omega_M \triangleq \sum_{i \in S_{Med}} w_i$ (the total weight of medium-ratio jobs),
- $\omega_H \triangleq \sum_{i \in S_{High}} w_i$ (the total weight of high-ratio jobs),
- $\tau_M \triangleq \sum_{i \in S_{Med}} t_i$ (the total processing time of medium-ratio jobs), and
\[ \omega_T \triangleq \sum_{i \in S_{\text{Med}} \cup S_{\text{High}}} \mathbb{E} \left[ \min \left( Tw_i, t_iW \right) \right] \] (the expected ordering costs of a tested job and the known jobs).

Building on Theorem 2.3.3 and the stopping time interpretation for the S&T problem, we formulate an improved DP.

**Definition 2.4.1.** The Low Dimensional DP is defined as following

\[
J_{LD} \left( \begin{array}{c} N, \\ \omega_M, \omega_H, \\ \tau_M, \omega_T \end{array} \right) = \min \left( J_{\text{test-one}}^{LD} \left( \begin{array}{c} N, \\ \omega_M, \omega_H, \\ \tau_M, \omega_T \end{array} \right), J_{\text{process-all}}^{LD} \left( \begin{array}{c} N, \\ \omega_M, \omega_H, \\ \tau_M, \omega_T \end{array} \right) \right). \tag{2.4}
\]

\[
J_{\text{test-one}}^{LD} \left( \begin{array}{c} N, \\ \omega_M, \omega_H, \\ \tau_M, \omega_T \end{array} \right) = \mathbb{E} \left[ TW \right] + \left( \omega_M + \omega_H + N \mathbb{E} \left[ W \right] \right) t_a + \omega_T +
\]

\[
\sum_{(d,v) \in S_D} p_{d,v} \left( \begin{array}{c} 1_{\frac{d}{v} < \rho_a} d \left( N - 1 \right) \mathbb{E} \left[ W \right] + \\ 1_{\frac{d}{v} < \rho_a} J_{LD} \left( N - 1, \omega_M, \omega_H, \tau_M, \omega_T \right) + \\ 1_{\rho_a < \frac{d}{v} < \rho} J_{LD} \left( N - 1, \omega_M + v, \omega_H, \tau_M + d, \omega_T + \mathbb{E} \left[ \min \left( Tv, dW \right) \right] \right) \right) + \tag{2.5}
\]

\[
J_{\text{process-all}}^{LD} \left( \begin{array}{c} N, \\ \omega_M, \omega_H, \\ \tau_M, \omega_T \end{array} \right) = N \mathbb{E} \left[ TW \right] + \mathbb{E} \left[ W \right] N \tau_M + N \mathbb{E} \left[ T \right] \omega_H + \left( \begin{array}{c} N \end{array} \right) \mathbb{E} \left[ T \right] \mathbb{E} \left[ W \right]. \tag{2.6}
\]

\[
J_{LD} \left( \begin{array}{c} 0, \\ \omega_M, \omega_H, \\ \tau_M, \omega_T \end{array} \right) = 0.
\]
In these expressions, \(d\) and \(v\) are realizations of the processing time and weight of an unknown job.

There are only two controls in the LD DP: “test-one” and “process-all”. Test-one refers to testing an unknown job and processing the respective job if it has low-ratio. Process-all refers to processing all jobs in a non-decreasing order of their ratio. Observe that process-all has a closed form, and that test-one is defined recursively.

We next show that there is an equivalence between the DP formulation of Section 2.2.3 and the LD DP.

**Theorem 2.4.2.** For every state \((N, [t_1, w_1, ..., t_n, w_n])\), the following holds:

\[
J_{\text{merge}} (N, [t_1, w_1, ..., t_n, w_n]) = J_{\text{LD}} (N, \omega_M, \omega_H, \tau_M, \omega_T)
\]

**Proof.** We prove the lemma in three steps. First, we incorporate the structural results of Section 2.3 to the DP formulation of Section 2.2.3 to obtain a formulation with reduced control and state spaces. Second, we show that the new formulation is consistent with the objective (expected weighted sum of completion times). Lastly, we substitute terms to obtain an equivalent LD DP formulation.

**A modified DP formulation.**

In Section 2.3, we proved two important properties of the optimal policy: (1) low-ratio jobs are processed immediately; (2) the problem can be seen as a stopping time problem having only two controls: test-one and process-all. Using these two properties we reduce the control space (only two controls) and the state space (no low-ratio jobs). We use the principals of the marginal cost accounting method and the structure of the optimal policy to define a cost function that accounts for testing delays and ordering costs at the earliest moment when any order between two jobs is determined.

We analyze the costs under each of the two control, starting with test-one. Figure 2.4 illustrates the sources of different costs inflicted by the control test-one. Figure 2-4
illustrates the initial system state before performing the test-one action. There are two unknown jobs, two medium-ratio jobs, and two high-ratio jobs. There are six types of costs:

1. Self imposing costs of the tested job: $E\left[TW\right]$ (Figure 2-4b). The now realized duration of the tested job is part of the completion time of that same job which contributes the cost $E\left[TW\right]$ to the objective.

2. Testing delays: $\left(\sum_{i \in S_{Med} \cup S_{High}} w_i + N E\left[W\right]\right) t_a$ (Figure 2-4c). Each of the jobs in the system is being delayed by a duration of $t_a$.

3. Costs induced by pairs of known jobs and the tested job: $\sum_{i \in S_{Med} \cup S_{High}} E\left[\min\left(Tw_i, t_i W\right)\right]$ (Figure 2-4d). Once the tested job is realized (duration $T$), its relative order with respect to the known jobs is being determined as these jobs are always processed according to the WSPT rule. We can therefore immediately account for these costs. If the ratio of the tested job is smaller than the ratio of the known job ($T/W < t_i/w_i$), the tested job is processed first in which case the objective is increased by $Tw_i$; otherwise, the known job is processed first which adds $t_i W$ to the objective. This is equivalent to adding the smaller of these two terms: $\min\left(Tw_i, t_i W\right)$.

4. Costs induced by the tested job (realized as $(d, v)$) and the other unknown jobs when the tested job has a low-ratio ($d/v < \rho_a$, Figure 2-4e): $d\left(N - 1\right) E\left[W\right]$. The low-ratio job is processed immediately and its duration is added to the completion time of each of the unknown jobs, adding a total of $d\left(N - 1\right) E\left[W\right]$ to the objective. Observe that there is one less unknown job once we test which explains why this expression contains $N - 1$.

5. Future costs in the case that the tested job has a low-ratio: $J_{mrg}\left(N - 1, [t_1, w_1, ..., t_n, w_n]\right)$ (Figure 2-4b). The tested job was processed and therefore is not included in the future state.

6. Future costs in case the tested job has a medium- or high-ratio (Figure 2-4e and
The tested job was not processed and therefore is included in the future state.

When choosing to process all jobs, unknown jobs are being processed according their ratio $\rho$. At that point, the entire processing order is determined, and we can account for the ordering costs between the unknown jobs and the rest of the known jobs (the relative order of known jobs have been known prior to deciding to process all jobs and therefore the respective costs of those pairs of jobs has already been accounted for).

Figure 2-5 illustrates the different types of relative orders that are being determined when we decide to process-all. There are three types of costs arising from the new information about job ordering:

1. Ordering costs between pairs of unknown jobs: \( \binom{N}{2} \mathbb{E}[T] \mathbb{E}[W] + N \mathbb{E}[TW] \) (Figure 2-5h). There are \( \binom{N}{2} \) pairs of unknown jobs, and in each pair a job with expected duration \( \mathbb{E}[T] \) delays a job with expected weight \( \mathbb{E}[W] \). In addition, each of the \( N \) unknown jobs delays itself by its duration which result in total additional cost of \( N \mathbb{E}[TW] \);

2. Ordering costs between pairs of medium-ratio jobs and unknown jobs: \( \mathbb{E}[W] N \left( \sum_{i \in \mathcal{S}_{Med}} t_i \right) \)
The unknown jobs are processed after the medium-ratio jobs which means that each of the $N$ unknown jobs is being delays by the total duration of the medium-ratio jobs;

3. Ordering costs between pairs of high-ratio jobs and unknown jobs: $N \mathbb{E}[T]\left(\sum_{i \in S_{\text{high}}} w_i\right)$ (Figure 2-5c). High-ratio jobs are processed after the unknown jobs, therefore, each of the high-ratio jobs is delays by the total duration of the unknown jobs: $(N \mathbb{E}[T])$. The total contribution to the objective from these pairs is the total expected duration of unknown jobs multiplied by the total weight of high-ratio jobs.

We can now write the complete Bellman’s Equation for the modified DP:

$$J_{\text{mrq}}\left(\begin{array}{c}N, \\
\left[t_1, w_1, \ldots, t_n, w_n\right]\end{array}\right) = \min \left( J_{\text{test-one}}^{\text{mrq}}\left(\begin{array}{c}N, \\
\left[t_1, w_1, \ldots, t_n, w_n\right]\end{array}\right), J_{\text{mrq}}^{\text{PA}}\left(\begin{array}{c}N, \\
\left[t_1, w_1, \ldots, t_n, w_n\right]\end{array}\right)\right),$$

where $J_{\text{mrq}}^{\text{PA}}$ is defined as:

$$J_{\text{mrq}}^{\text{PA}}\left(\begin{array}{c}N, \\
\left[t_1, w_1, \ldots, t_n, w_n\right]\end{array}\right) = N \mathbb{E}[TW] + \mathbb{E}[W] N\left(\sum_{i \in S_{\text{med}}} t_i\right) + N \mathbb{E}[T]\left(\sum_{i \in S_{\text{high}}} w_i\right) + \left(\frac{N}{2}\right)\mathbb{E}[T] \mathbb{E}[W],$$
and \( J_{\text{test-one}} \) is defined as:

\[
J_{\text{test-one}} \left( N, \left[ t_1, w_1, \ldots, t_n, w_n \right] \right) = \mathbb{E} [TW] + \left( \sum_{i \in S_{\text{Med}} \cup S_{\text{High}}} w_i + N \mathbb{E} [W] \right) t_a + \sum_{i \in S_{\text{Med}} \cup S_{\text{High}}} \mathbb{E} [\min (T w_i, t_i W)] + \sum_{(d, v) \in S_D} p_{d, v} \left( \begin{array}{c}
1_{\frac{1}{2} \leq \rho < \frac{1}{2}} d (N - 1) \mathbb{E} [W] + \\
1_{\frac{1}{2} < \rho \leq 1} J_{\text{mr}} (N - 1, [t_1, w_1, \ldots, t_n, w_n]) + \\
1_{\rho < \frac{1}{2}} J_{\text{mr}} (N - 1, [t_1, w_1, \ldots, t_n, w_n]) \\
\end{array} \right) .
\]

The boundary condition is given by

\[
J_{\text{mr}} \left( 0, \left[ t_1, w_1, \ldots, t_n, w_n \right] \right) = 0.
\]

**Consistency with the objective value.**

We now show that for any policy the modified DP formulation returns the expected weighted sum of completion time. We argue that the three types of costs of Equation 2.1 are being accounted for in a similar way:

1. Self inflicting costs. Under any policy, the duration of any job is part of its completion time. Therefore, regardless of the policy the term \( N \mathbb{E} [TW] \) should be part of the final cost. Under the modified formulation, this cost it accounted for either upon testing, or when we process all jobs in which case we add the term \( \mathbb{E} [TW] \) for each of the unknown jobs. The term \( \mathbb{E} [TW] \) is added exactly \( N \) times.
2. Testing delays. A natural way to look at the testing delay costs (Equation 2.1) is that for any job we need to multiple the job’s weight by the total testing delay. An alternative view is that each time an unknown job is tested we add the total weight of the jobs yet unprocessed. The latter is the exact cost accounting done by the modified formulation. Each time we test (and only then) we add the multiply the testing time by the total weight of the jobs in the system.

3. Ordering costs. In the modified DP formulation we account for the ordering cost at the earliest possible moment that the order between a pair of jobs is determined. This guarantees that the ordering cost is accounted for exactly once. To see this, consider any unknown job $i$. At the beginning all jobs are unknown and therefore no ordering costs have been accounted for. If we test a job $j$ and it has a low-ratio, then job $j$ is process immediately and we account for the pair $(i, j)$. The job $j$ is not part of the state and we will not account for this pair again. If job $j$ does not have a low-ratio, the ordering cost of $(i, j)$ will be accounted for either if we (1) test job $j$, in which case we account for it immediately, and we ignore this pair in the future (whether we test-one or process-all); (2) process all jobs, in which case we account for the pair $(i, j)$ exactly once. This is true for any unknown job, therefore it is true for the entire ordering cost.

Variable substitution.

Observe that the cost function of the modified DP can be represented as a function of the following quantities: the number of unknown jobs ($N$), the total weight of medium-ratio jobs ($\omega_M = \sum_{i \in S_{Med}} w_i$), the total weight of high-ratio jobs ($\omega_H = \sum_{i \in S_{High}} w_i$), the sum of processing times of medium-ratio jobs ($\tau_M = \sum_{i \in S_{Med}} t_i$), a summation over all known jobs ($\omega_\tau = \sum_{i \in S_{Med} \cup S_{High}} \mathbb{E} [\min (T w_i, t_i W)]$), and a few other constants (e.g., $\mathbb{E} [TW]$). By substituting for these quantities, we establish the equivalence between the two DP formulations. That is,

$$J_{mrg}(N, [t_1, w_1, ..., t_n, w_n]) = J_{LD}(N, \omega_M, \omega_H, \tau_M, \omega_\tau).$$
Note that the dimension of the state space is now five (comparing with the initial DP formulation in Section 2.2.3, where the dimension was as high as $N_0$). Observe that while this significantly reduces the complexity of the problem, the number of states in the low-dimensional formulation could still be large as it contains pseudo-polynomial terms. For this reason we develop an FPTAS that provides an approximation for the optimal solution and guarantees a polynomial running time (with a certain dependency on the approximation level).

### 2.4.2 Optimal Threshold Policy

The LD formulation has several advantageous properties in addition to having a low dimension. It is monotonous in its parameters, but more importantly, its optimal solution admits a threshold structure. That is, for every value of $N, \omega_M, \omega_H$ and $\tau_M$ there exists a threshold value for $\omega_T$ that determines if the optimal action is to test or to process all jobs.

**Lemma 2.4.3.** The value function $J_{LD}$ is non-decreasing in $\tau_M$ and $\omega_T$.

**Proof.** See Appendix A.9.

**Lemma 2.4.4.** For every value of $N, \omega_M, \omega_H$ and $\tau_M$, there exists a threshold value $\overline{\omega_T}$ such that testing is the optimal control, if and only if, $\omega_T \leq \overline{\omega_T}$.

**Proof.** See Appendix A.10.

Lemma 2.4.4 implies that there is an efficient way of representing the optimal policy when the support of the distribution $\mathcal{D}$ is discrete. The number of different values that $N, \omega_M, \omega_H$, and $\tau_M$ can take is polynomial in $N_0$, $D$, and $V$, and for every value of $(N, \omega_M, \omega_H, \tau_M)$, exactly one value of $\overline{\omega_T}$ is sufficient to describe the optimal policy.

However, to find the actual values of these thresholds we need to solve the DP. This cannot be done using standard DP methods due to the exponential growth of the state space. In the next section we develop an approximation scheme for solving the low-dimensional.
Building upon the LD formulation of Section 2.4.1, we use a rounding technique (e.g., see Williamson and Shmoys (2011)) to formulate an Approximate Dynamic Program (ADP) which is then used as a basis of an FPTAS.

We start by describing the state space of the ADP formulation. The state space is similar to the state space in the low-dimensional formulation, except for having rounded values for the second through the fifth dimensions. The first dimension of the state space takes values from the set \( \{0, 1, \ldots, N_0\} \). The second and third dimensions of the LD state space, namely \( \omega_M \) and \( \omega_H \), can take values from the set \( \{ \sum_{i \in S} w_i \} \), where \( w_i \in \{1, \ldots, V\} \), and \( S \) represents a collection of known jobs. The minimal and maximal values of \( \omega_M \) and \( \omega_H \), are 0 and \( N_0V \), respectively. The discretized set of values is defined as:

\[
\mathcal{S}_1 = \{0, 1, (1 + \delta), (1 + \delta)^2, \ldots, \lceil N_0V \rceil \} .
\]

The operator \( \lceil \cdot \rceil \) rounds up to the next power of \( 1 + \delta \). Similarly, the maximal values that the dimension \( \tau_M \) and \( \omega_T \) can hold are \( N_0D \), and \( N_0DV \), respectively. The discretized sets of values for the dimensions \( \tau_M \) and \( \omega_T \) are defined as:

\[
\mathcal{S}_2 = \{0, 1, (1 + \delta), (1 + \delta)^2, \ldots, \lceil N_0D \rceil \} , \mathcal{S}_3 = \{0, 1, (1 + \delta), (1 + \delta)^2, \ldots, \lceil N_0DV \rceil \} .
\]

Bellman’s equation for the ADP can be written as follows:
\[
J_\delta \left( \begin{array}{c}
N, \\
\omega_M, \omega_H, \\
\tau_M, \omega_T
\end{array} \right) = \min \left\{ \begin{array}{c}
\mathbb{E}[TW] + (\omega_M + \omega_H + N\mathbb{E}[W]) t_a + \omega_T + \\
1_{\frac{g}{\delta} \leq \mu_a \leq \rho} d (N - 1) \mathbb{E}[W] + \\
1_{\frac{g}{\delta} < \mu_a < \rho} J_\delta (N - 1, [\omega_M], [\omega_H], [\tau_M], [\omega_T]) + \\
1_{\mu_a < \frac{g}{\delta}} J_\delta (N - 1, [\omega_M + \mu], [\omega_H], [\tau_M + d], [\omega_T + \mathbb{E}[\min(T v, d W)]] + \\
1_{\mu_a > \frac{g}{\delta}} J_\delta (N - 1, [\omega_M], [\omega_H + \mu], [\tau_M], [\omega_T + \mathbb{E}[\min(T v, d W)]]
\end{array} \right) \\
\mathbb{E}[W] N \tau_M + N \mathbb{E}[T]\omega_H + N \mathbb{E}[TW] + \left( \frac{N}{2} \right) \mathbb{E}[T] \mathbb{E}[W]
\right\}
\]

\[
J_\delta \left( \begin{array}{c}
0, \\
\omega_M, \omega_H, \\
\tau_M, \omega_T
\end{array} \right) = 0.
\]

The next lemma shows that the ADP can be used as a close approximation to the low-dimensional DP.

**Lemma 2.4.5.** For any state \((N, \omega_M, \omega_H, \tau_M, \omega_T)\), the ratio between the value functions of the ADP and the LD DP formulations is greater than or equal to one, and at most \((1 + \delta)^N\):

\[1 \leq \frac{J_\delta (N, \omega_M, \omega_H, \tau_M, \omega_T)}{J_{LD} (N, \omega_M, \omega_H, \tau_M, \omega_T)} \leq (1 + \delta)^N.\]

**Proof.** See Appendix A.11. \qed

The approximation algorithm is summarized in Algorithm 1. At each state, we use the ADP to make decisions in the low-dimensional DP problem space. The decision of testing or processing is done by evaluating the two controls using the approximated value function
$J_\delta$. When at state $(N, \omega_M, \omega_H, \tau_M, \omega_\tau)$, the decision we make is sub-optimal by a factor of at most $(1 + \delta)^N$ (Lemma 2.4.5). Our next decision is sub-optimal by a factor of at most $(1 + \delta)^{N-1}$. On the overall, the approximation algorithm is sub-optimal by a factor of at most $(1 + \delta)^{N^2}$. Therefore, setting the value of $\delta$ to $e^{-N^2_0} - 1$ ensures a $1 + \epsilon$ approximation.

**Algorithm 1** Approximation Algorithm.

1: Choose $\epsilon^{\frac{1}{\log(1+\delta)}}$
2: Set $\delta$ to $e^{\frac{N^2_0}{N_0}} - 1$
3: Solve the ADP: calculate $J_\delta(N, \omega_M, \omega_H, \tau_M, \omega_\tau)$ for $N \leq N_0$, $\omega_M, \omega_H \in S_1$, $\tau_M \in S_2$, and $\omega_\tau \in S_3$
4: Set the current state $S$ to $(N_0, 0, 0, 0, 0)$
5: while $S.N > 0$ do
6: Activate the control that minimizes the value function $J_\delta(S)$
7: Update the current state based on the selected control and the observed realization
8: end while

The size of each of the sets $S_1, S_2,$ and $S_3$ is polynomial in $N_0, \log D, \log V,$ and, $1/\log(1 + \delta)$. By plugging the value of $\delta$, we obtain the following:

$$\frac{1}{\log(1 + \delta)} = \frac{N^2_0}{\log(1 + \epsilon)} = O\left(\frac{N^2_0}{\epsilon}\right),$$

which means that the total number of states in the ADP formulation is polynomial in the input size and $1/\epsilon$, and we can solve the ADP in that same order of time.

The algorithm returns a $1 + \epsilon$ approximation for the optimal policy, and its running time is polynomial in the input size and $1/\epsilon$. Therefore, it is an FPTAS.

**2.5 Optimal Myopic Policies**

The optimal and near-optimal algorithmic solutions presented in the previous section allow us to solve the problem in a polynomial number of steps. However, from a practical point of view, $N^5$ may not be tractable for large instances. The heuristics on the other hand, can be efficiently implemented, but may not always have sufficiently good performance guarantees.
In this section, we study a myopic policy that is both efficient and optimal under a relatively general assumption (which includes the case where all jobs have equal weights). This seems to be quite surprising given the high dimensionality in the state space of the initial problem formulation (before analyzing and characterizing the optimal policy).

We start with a definition before stating the assumption and the main Theorem.

**Definition 2.5.1.** Process all (PA) is the policy in which all jobs are processed in a non-decreasing order of their expected ratio.

**Definition 2.5.2.** The Single test policy (STP) is the policy in which a single unknown job is tested before processing all jobs in a non-decreasing order of their expected ratio (assuming there is at least one unknown job).

**Lemma 2.5.3.** For any state $(N, [t_1, w_1, ..., t_n, w_n])$ with no low-ratio jobs and $N > 0$, the difference in the objective value between policies PA and STP is equal to:

$$J^{STP} - J^{PA} = \left( N\mathbb{E}[W] + \sum_i w_i \right) t_a - (N - 1)\mathbb{E}[(W\mathbb{E}[T] - \mathbb{E}[W]T)^+] - \sum_{i \in \text{Medium}} \mathbb{E}[(Wt_i - w_i T)^+] - \sum_{i \in \text{High}} \mathbb{E}[(w_i T - Wt_i)^+].$$

**Proof.** See Appendix A.12.

**Assumption 2.5.4.** For all jobs $i$ that have medium ratio: $t_i \leq \mathbb{E}[T]$, and for all jobs $i$ that have high ratio: $w_i \leq \mathbb{E}[W]$.

Figure 2-6 illustrates distributions $D$ that satisfy Assumption 2.5.4. These are distributions whose support reside within the shaded area.

There are two interesting special cases that satisfy Assumption 2.5.4. In the first special case, all jobs have the same weights but potentially different processing times. The low-, medium-, and high-ratio job classes correspond to jobs with short, intermediate, and long processing times, respectively. Moreover, the testing threshold corresponds to a certain duration (shorter than the mean processing time), such that all jobs whose processing times

53
are shorter than the testing threshold, should be processed immediately. All intermediate and long jobs are put on hold until the testing phase is completed.

In the second special case, all jobs have the same processing times, but could have different weights. In this case, the low-, medium-, and high-ratio job classes correspond to jobs with high, intermediate, and low weights, respectively. Here the testing threshold corresponds to a minimal weight (which is higher than the mean weight), such that all jobs whose weights are above this threshold should be processed immediately without further delay, while the rest of the known jobs with either low or intermediate weights are put on hold.

For all problem instances that satisfy Assumption 2.5.4, a simple myopic rule govern the decision to stop testing, as given by the following Theorem.

**Theorem 2.5.5.** Under Assumption 1, for any state \((N, [t_1, w_1, ..., t_n, w_n])\) with no low-ratio jobs and \(N > 0\), the optimal control is to process all jobs, if and only if, the following condition holds:

\[
(N \mathbb{E}[W] + \sum_i w_i) t_a \geq (N - 1) \mathbb{E}[(W \mathbb{E}[T] - \mathbb{E}[W]T)^+] + \sum_{i \in \text{Medium}} \mathbb{E}[(W t_i - w_i T)^+] + \sum_{i \in \text{High}} \mathbb{E}[(w_i T - W t_i)^+].
\]

**Proof.** See Appendix A.6 \(\square\)
The optimal policy under Assumption 2.5.4 is summarized in Algorithm 2. From a technical standpoint, under Assumption 2.5.4, there is monotonicity in Equation (2.9) in the sense that with every test, regardless of the realization of the tested job, the difference between the lefthand side and righthand side increases. This implies that once the lefthand side surpasses the righthand side (which from myopic standpoint means that we would like to stop) it will always stay higher. Under such condition we apply an inductive argument and show that any potentially optimal policy that tests must test exactly once, which means that this policy is an STP policy. This contradicts the optimality of such potentially optimal policy since that the myopic policy would test had the STP policy been better than processing all jobs.

From an intuitive perspective, the optimal policy is sensitive to realizations within the shaded area. On expectation, it might not worth testing once, but if the realization happens to be an outcome with exceptionally high processing time and weight, we suddenly might want to test again since the cost of a sub-optimal schedule increased comparing to the testing cost. Based on this intuition, we construct an example and show that when Assumption 2.5.4 is not satisfied the myopic policy need not always be optimal (see Appendix A.7).

Algorithm 2 The myopic policy algorithm.

1: Process all jobs with ratio below \( \rho_a \) in a non-decreasing order of their ratio.
2: \textbf{while} the following condition is satisfied
3: \begin{align*}
\left( N \mathbb{E}[W] + \sum_i w_i \right) t_a &< (N - 1) \mathbb{E}[(W \mathbb{E}[T] - W \mathbb{E}[T]^+)] \\
&+ \sum_{i \in \text{Medium}} \mathbb{E}[(W t_i - w_i T)^+] + \sum_{i \in \text{High}} \mathbb{E}[(w_i T - W t_i)^+];
\end{align*}
4: \textbf{do}
5: Test jobs and process them immediately if they have low-ratio.
6: \textbf{end while}
7: Process all jobs in a non-decreasing order of their ratio, where \( \rho \) is the ratio for all unknown jobs, and the processing order of any two jobs with equal ratio can be arbitrarily chosen.
2.6 The Value of Testing

In this section we address the question of how much can we actually gain from testing. We start by analyzing a few simple heuristics, and then examine how the different problem parameters affect the value of testing.

2.6.1 Heuristics

We analyze the performance of three simple policies:

- “Process-all” \( (PA\), which was introduced in Section 2.3.1\).
- “Test-all first” \( (TAF)\), and
- “Test-all process-low-ratio” \( (TAPL)\).

As their names suggest, in “test-all first”, we start by testing all jobs, and then process them in a non-decreasing order of job ratios. Under the policy “test-all process low-ratio”, we test all unknown jobs, but immediately process low-ratio jobs, and process other known jobs in a non-decreasing order of ratio only after all testing has been completed. Note that policies \( PA\) and \( TAPL\) correspond to the two extremes of the optimal policy: the first, when the stopping time is zero, and the latter, when the stopping time is \( N\). We denote by \( OPT\) the optimal policy.

Clairvoyant Solution (CL)

We use the clairvoyant solution \( (CL)\) as a lower bound for the optimal policy. That is, we calculate the objective value when the processing times and weights are known to the scheduler (or alternatively if the testing time is zero). Using marginal cost accounting, the objective value is

\[
J^{CL} (N, []) = E \left[ \sum_{i=1}^{N} T_i W_i + \sum_{i=1}^{N} \sum_{j=1}^{i-1} \min (W_i T_j, W_j T_i) \right] = NE [TW] + \left( \frac{N}{2} \right) E [\min (W_i T_j, W_j T_i)] .
\]

56
The objective value consists of the self imposing costs \( \sum_{i=1}^{N} T_i W_i \), and the ordering costs of the perfectly ordered jobs (as all the information is known in advance).

**The “Process All” (PA) Policy**

Like the clairvoyant solution, we use marginal cost accounting to calculate the objective value under policy PA:

\[
J^{PA}(N, \emptyset) = N \mathbb{E}[TW] + \left( \frac{N}{2} \right) \mathbb{E}[T] \mathbb{E}[W].
\]

The self imposing cost of every job is \( \mathbb{E}[TW] \) and every pair contributes a cost of \( \mathbb{E}[T] \mathbb{E}[W] \) to the objective (we use the independence between jobs). We obtain the following bound on the objective value of policy PA:

\[
\frac{J^{PA}(N, \emptyset)}{J^{OPT}(N, \emptyset)} \leq \frac{J^{PA}(N, \emptyset)}{J^{CL}(N, \emptyset)} \leq \frac{\mathbb{E}[T] \mathbb{E}[W]}{\mathbb{E}[\min(W_i T_j, W_j T_i)]}.
\]

The process all policy is of special interest to us, as it serves as basis for comparison when testing is not available. Put another way, comparing the optimal policy with the process all policy tells us the value of testing.

To get a better sense of the bound, we list below its values for a sample of distributions for the unweighted case \( W = 1 \). In favor of readability, the proofs are included in Appendix A.13.

**Lemma 2.6.1.** If \( T \sim \text{Uni}(a, b) \) and \( W = 1 \) the policy PA is a \( \frac{3(b+a)}{2(b+2a)} \)-approximation for the optimal policy.

**Lemma 2.6.2.** If \( T \sim \text{Exp}(\lambda) \) and \( W = 1 \) the policy PA is a 2-approximation for the optimal policy.

**Lemma 2.6.3.** If \( T \sim \begin{cases} a & \text{with probability } p \\ b & \text{with probability } 1 - p \end{cases} \) and \( W = 1 \) the policy PA is a \( \frac{p a + (1-p)b}{a + (1-p)b} \)-approximation for the optimal policy.
In the last example, when \( a = 0, p = 1/2 \), policy \( PA \) is a a 2-approximation, and for \( a = 0, b = M, p = 1 - 1/M \), \( PA \) is an \( M \)-approximation. This means that policy \( PA \) can do arbitrarily bad. When the testing time is relatively short, the clairvoyant solution is close to the optimal policy, and the bound becomes tight. In these examples, we see that testing can improve the objective by 33\% for the uniform distribution, by 50\% for the exponential distribution, and in some cases can do even better (e.g., the \( M \)-approximation).

The “Test-All First” (TAF) Policy

When the testing time is short, it makes sense to test all jobs before doing any processing so that processing will be performed in the optimal order. It is easy to see that:

\[
J^{TAF} (N, []) = t_a N^2 \mathbb{E} [W] + J^{CL} (N, []),
\]

which results in an approximation ratio of

\[
\frac{J^{TAF} (N, [])}{J^{OPT} (N, [])} \leq \frac{J^{TAF} (N, [])}{J^{CL} (N, [])} = \frac{t_a N^2 \mathbb{E} [W] + J^{CL} (N, [])}{J^{CL} (N, [])} \leq 1 + \frac{2t_a \mathbb{E} [W]}{\mathbb{E} [\min (W_i T_j, W_j T_i)]}.
\]

Indeed, as \( t_a \) approaches zero, the TAF policy becomes optimal.

The “Test-All Process Low-Ratio” (TAPL) Policy

The intuition behind policy \( TAPL \), is that one could try to improve policy \( TAF \) by processing low-ratio jobs immediately upon detection (in order to follow Lemma [A-3]).

The \( TAPL \) policy has two types of additional costs compared with the clairvoyant solution: due to testing delays (as it tests all jobs), and due to sub-optimal processing order. Denoting by \( N^l \) the number of low-ratio jobs, the total expected cost under policy \( TAPL \) can be written as follows:

\[
J^{TAPL} (N, []) = J^{CL} (N, []) + \mathbb{E} \left[ \binom{N^l}{2} \right] \mathbb{E} \left[ (W_2 T_1 - W_1 T_2)^+ \right] \frac{T_1}{W_1} \frac{T_2}{W_2} < \rho_a
\]
\[ +t_a \mathbb{E} [N - N^t] \mathbb{E} \left[ W | \frac{T}{W} > \rho_a \right] + t_a \mathbb{E} [N^t] \frac{1 + N}{2} \mathbb{E} \left[ W | \frac{T}{W} < \rho \right] \] (2.11)

See Appendix A.14 for the complete derivation. Comparing the costs under policies \( TAF \) and \( TAPL \), we observe that under policy TAF, the total testing time penalty \( (t_a N^2 \mathbb{E}[W]) \) is higher, but on the other hand, the processing order is optimal. In other words, policy TAPL trades the optimality of the processing order, in exchange for lower testing times. Using Equations (2.10) and (2.11), we can conclude in what cases the tradeoff is worthwhile.

### 2.6.2 The Initial Number of Jobs

We proceed to study how the value of testing is affected by the initial number of unknown jobs. Specifically, we use a sufficient condition on the state space for which testing is optimal, to find a lower bound on the number of times that the control test one is optimal (assuming \( \rho_a < \rho \)).

The following quantity is important for describing the lower bound on the optimal stopping time.

**Definition 2.6.4.** We define the stopping factor \( \beta \) as

\[
\beta := \frac{t_a^{max}}{t_a} = \frac{\mathbb{E} \left[ (\rho W - T)^+ \right]}{\mathbb{E} \left[ (\rho_a W - T)^+ \right]} = \frac{\mathbb{E} \left[ (\mathbb{E}[T] W - T \mathbb{E}[W])^+ \right]}{t_a \mathbb{E}[W]}.
\]

To develop intuition regarding the value \( \beta \), consider the case when an unknown job is tested immediately after processing another unknown job. The stopping factor \( \beta \) is the ratio between the savings from testing earlier (using an improved schedule), and the additional cost of testing (from having another job waiting while we test). Intuitively, the higher the values of \( \beta \), it is more beneficial to test. (Note that as the testing time decreases, \( \beta \) increases, which matches our intuition that for higher values of \( \beta \), testing is more beneficial.) Also note that when the ratio is less than 1, it is sub-optimal to test (since \( \rho_a < \rho \)). We can therefore assume that \( \beta \) is greater than 1.
Lemma 2.6.5. For every state \((N, \omega_M, \omega_H, \tau_M, \omega_T)\), where \(\beta > (N\mathbb{E}[W] + \omega_H) / (N - 1)\), the optimal control is to test.

Proof. See Appendix A.15.

Note that the lefthand side of the inequality in Lemma 2.6.5 above (i.e., \(\beta\)) is a constant for the problem, and that the righthand side dynamically evolves as a function of the state. Moreover, \(\beta > 1\), and the righthand side goes to 1 as \(N\) becomes larger. Therefore, when the number of initial unknown jobs is large, testing is optimal for an increasing number of jobs. From a practical point of view, in congested systems with many unknown jobs, testing is most beneficial.

In the next theorem, we use Lemma 2.6.5 to obtain a lower bound on the minimal number of tests.

Theorem 2.6.6. For finite and symmetrical distributions of jobs weight, the optimal policy tests for at least \(N_{\text{tests}}\) periods where

\[
N_{\text{tests}} = \left\lfloor N_0 \frac{\beta - 1}{\beta + 1} - \frac{\beta}{\beta + 1} \right\rfloor = N_0 - 1 - \left\lceil \frac{2N_0 + \beta}{\beta + 1} \right\rceil.
\]

Proof. See Appendix A.8.

Lemma 2.6.5 suggests that the higher the number of unknown jobs is, the more likely we are to test, that is the value of information increases with \(N_0\).

Another implication of Lemma 2.6.5 (that can be derived similarly to Theorem 2.6.6), is that when the starting state has \(N_0\) unknown jobs, to find the optimal policy we need to solve a DP with only \(N = \beta / (\beta - 1)\) jobs, that is, we only need to solve a constant size DP. While this quantity could still be high (especially when the testing time is short), it does not depend on \(N_0\), which indicates that the policy that tests all \(N_0\) jobs and processes them if they have low-ratio is asymptotically optimal (to see this, we briefly note that \(J_{LD}(N_0, 0, 0, 0) = \Theta(N_0^2)\), and that \(J_{LD}(\beta / (\beta - 1), \omega_M, \omega_H, \tau_M, \omega_T) = \Theta((\beta / (\beta - 1))^2 + (\beta / (\beta - 1))N_0)\), which means that testing and processing low-ratio jobs immediately can be sub-optimal up to a cost that decreases asymptotically with \(N_0\).
2.6.3 The Testing Time

Using the condition \( \rho < \rho_a \), we derive a threshold value for the testing time above which testing is never optimal.

**Theorem 2.6.7.** *If the testing time is greater than \( t_a^{\text{max}} := \mathbb{E} \left[ (\rho W - T)^+ \right] \), testing is never optimal.*

*Proof.* By definition, if \( t_a > t_a^{\text{max}} \), then \( t_a > \mathbb{E} \left[ (\rho W - T)^+ \right] \). Therefore, \( \rho_a > \rho \) (Lemma 2.3.2), and testing is not optimal (Theorem 2.3.6).

Theorem 2.6.7 provides an upper bound on the testing time for which testing is still beneficial. In Section 2.6.2 we saw that it is in fact the smallest possible bound (when \( \rho_a < \rho \), testing is always optimal for large enough values of \( N_0 \)).

2.6.4 Numerical Illustration

To illustrate the aforementioned observations, we plot in Figure 2-8 the value of testing for different problem parameters. Specifically, we plot the optimality gap between the optimal policy (that could test) and the policy that never tests (Process-all) for three probability distributions (see Figure 2-7), and three values of the initial number of jobs \( (N_0 = 6, 7, 8) \), as a function of the testing time \( t_a \). We summarize our main findings below.

**Testing time.** For all curves, we see that when the testing time is high, processing all jobs is optimal (that is the value of testing is zero, Theorem 2.6.7). As the testing time decreases, the performance of policy PA deteriorates, and the optimal policy is a two-phase policy that transitions between TAPL to PA. Policy PA is worst when \( t_a \to 0 \), where the optimal policy coincides with policies TAF and TAPL.

**Variability.** The probability distributions selected for this experiment share the same mean value but are different otherwise. In the figure, we see that the value of testing is highest for the Polar distribution, and lowest for the Binomial distribution. That is, in
the above example the value of testing is correlated with the variability of the processing times. This result is very intuitive, as we gain more information by testing jobs with a higher variability of the processing time distribution.

**The initial number of jobs.** We also see in the figure, that the value of testing increases with the number of initial jobs. This observation may seem unintuitive at first, because the cost of testing increases when there are more jobs in the system (as more jobs are being delayed by testing). It seems, however, that the benefit of testing when there are more jobs is more significant than the additional testing costs.

### 2.7 Extensions

In this section, we study a generalization of the problem and show that it can be solved using the analysis and methods presented in Sections 2.3 and 2.4. Specifically, we consider the case where testing does not reveal the exact processing times and weights, but rather the class of the respective job. For example, in the context of emergency departments, testing may reveal the type of treatment required by the patient but there may still be uncertainty associated with the actual service time, and severity.

We assume that there are $C$ classes, and that for class $i \in C$, the processing time and weight of every job $T_i, W_i$ are R.V.s from a known distribution $\mathcal{D}_i$ with expected processing
Figure 2-8: The value of testing for different distributions and initial number of jobs.

time $\bar{T}_i$, and expected weight $\bar{W}_i$. The probability that a job belongs to class $i$ is denoted by $p_i^c$ (which implies that the expected processing time and weight of an unknown job are $E[T] = \sum p_i^c \bar{T}_i$, and $E[W] = \sum p_i^c \bar{W}_i$ respectively). Testing a job reveals its class and requires $t_a$ units of time. Here we denote by unknown, jobs that were not yet tested, and by known jobs that were tested and which class is known (although actual processing time and weight may still be random).

Figure 2-9 illustrates the differences between the original and generalized problems. In the original problem (2-9a) the exact processing time is known once testing is performed. On the other hand, in the generalized problem (2-9b), only the class is realized by testing, and the exact processing time is known only after processing. Note that in the generalized problem, jobs can be tested at most once, that is, we can test a job to find its class and the respective distribution $D_i$ of the processing time and weight, but we cannot test it further for exact values. Also note, that if the distributions $D_i$ are degenerate, the generalized problem reduces to a discrete instance of the original problem. Finally, observe that this generalization can be used to model errors in testing, in which testing either reveals the true realization with some probability, or testing reveals false realizations with complementing probabilities (as captured by the model of Sun et al. (2014)). We note briefly that this can
be accomplished by defining job classes that correspond to each testing result, and specifying
the probabilities that a random variable from that distribution is realized to the true and
false values.

While the generalized problem is more complex, the analysis of Section 2.3 carries
through. In particular, we can show that this problem is equivalent to the initial model
where each class $C$ is replaced by a deterministic job whose processing time and weight are
equal to the expected processing time and the expected weight of jobs class $C$. Intuitively,
this follows from the linearity of the expectation operator and because the objective function
(weighted sum of completion time) is practically linear with respect to individual processing
times and weights (the self imposing costs do not depend on the policy and can be ignored).
See Appendix A.16 for a detailed discussion and proof.

We conclude this section by noting that while a basic assumption of the model was that
the testing time is a constant, it is easy to show that results follow even when the testing time
is a random variable $T_a$, which may be correlated with the processing time and weight of the
tested job. In this case, the testing threshold $\rho_a$ is similarly defined using the expectation
of $T_a$ (instead of using $t_a$). A small modification to the DP formulation of Section 2.2.3 is
needed, specifically, replacing the expression $N t_a \mathbb{E}[W]$ by the term $(N - 1) \mathbb{E}[T_a] \mathbb{E}[W] + \mathbb{E}[W T_a]$. We also note that if the testing time is part of the processing time, that is, testing
decreases the processing time by $t_a$, the structure of the optimal policy is preserved. In this
case however, the testing threshold $\rho_a$ is smaller than the threshold of the original problem.
2.8 Conclusions

In this chapter we introduced a new class of models that captures a principal exploration-exploitation tradeoff that is common in many scheduling problems. We analyzed the problem and found an intuitive characterization of the optimal policy. For a large number of cases, the optimal policy was given explicitly in the form of a stopping rule. For all other cases, a novel cost accounting scheme was used to formulate a low-dimension DP, which lead to optimal and near optimal algorithms. We studied the performance of several intuitive policies, and how the problem parameters affect the value of testing. Finally, it was shown that the properties and algorithms extend to a more general model.
Chapter 3

To Test or Not to Test: Reducing Uncertainty in Scheduling Diverse Jobs

In this chapter, we continue the study of the exploration-exploitation tradeoff that characterizes operational problems with uncertainty reduction controls. We generalize the model introduced in the first chapter by studying how to prioritize testing when jobs have different uncertainties. The difference in uncertainties is modeled using the convex order, a general relation between distributions, which implies that the variance of one distribution is higher than the variance of the other distribution. Unlike the initial model, here the decision-maker has to decide not only whether to test or process jobs, but also to choose which job to test next, from a set of non-homogeneous jobs. We show that the structure of the optimal policy perfectly generalizes that of the initial problem where jobs were homogeneous and with equal weights. Moreover, we show that when testing, it is optimal to test the job that is highest in convex order, which intuitively, corresponds to the job that has the highest uncertainty. In proving these results, we use a novel analysis based on the concept of mean preserving local spread (Müller and Stoyan (2002)), which is a special case of convex order where the probability distributions of jobs are marginally different. This enables reducing the comparison of functions of random variables in convex order to a simpler comparison between distributions that differ only locally. This is a general technique that could be useful
in the analysis of other problems, in particular in dynamic optimization problems where the underlying probability distributions are in convex order (and potentially also other types of stochastic orders).

The rest of the chapter is organized as follows. In Section 3.1 we introduce the problem and the applications, and summarize our contributions. Section 3.2 provides background on the convex order which we use as a modeling assumption. In Section 3.3 we introduce a scheduling with testing model in which jobs are non-homogeneous, and summarize the known result for the special case when jobs are homogeneous. In Section 3.4 we analyze the problem and describe the optimal policy, and in Section 3.5 we highlight several promising future research directions enabled by the analysis.

3.1 Introduction

In many operational settings with underlying uncertainty, there is a fundamental tradeoff between performing work (exploitation) and investing capacity (and time) to reduce the underlying uncertainty (exploration). For example, the common practice in emergency departments is to triage patients to better understand their condition, required care (and associated resources), and urgency. In this context, the medical staff is involved in two conceptually different activities. One activity is geared towards eliminating or reducing uncertainties, while the other activity is aimed at advancing work. Examining patients reveals their urgencies, but it also postpones the treatment of other patients. The emergency department operators must therefore decide whether to engage in examination or treatment of patients. The triage system is essentially one practical way of solving this exploration-exploitation tradeoff.

Other applications of this tradeoff include:

1. Diagnostic and repair work of engines. Repairing aircraft engines involves assembling and disassembling engines, two time-consuming activities that utilizes the maintenance crew, and result in high setup times that discourage preemption of repair work. In this environment, there is often uncertainty about the cause of failure and the overall
processing time. Moreover, the maintenance crews can perform diagnostic work on the engine (test) to reduce uncertainty and reveal the nature of breakdown and the required repair time. The fact that both repair and diagnostic work is done by the same resource (maintenance crew) results in a tradeoff between repairing engines under uncertainty (which may result in a suboptimal order of engine repairs), and investing time in reducing uncertainty and enables processing engines in an improved order (but with an additional testing delays).

2. Query optimization in databases \cite{Guha2007}. In database systems, a query optimizer prioritizes the execution order of a collection of query requests. The optimizer can also test queries to estimate the time needed to perform a certain query in order to optimize the processing sequence. Since that both testing and performing the actual processing of queries requires the exact same computational resources, we obtain the aforementioned tradeoff.

3. Multi-channel wireless networks \cite{Guha2007}. In wireless systems with multiple channels a user can choose the channel over which to transmit information. Before transmitting a significant amount of data, the user can assess the state of each channel by sending small control packets. A decision must then be made on whether to transmit data based on the available information, or to test channels which requires additional time, but could reduce the actual transition times.

In contrast to traditional exploration-exploitation models, in the settings studied in this thesis (and in particular, in this chapter), the issue is not to learn unknown distributions, but rather to reduce the uncertainty regarding the specific realizations of various uncertain parameters. We study models that capture the fundamental tradeoff in resource allocation between exploration and exploitation, as well as how to prioritize the exploration efforts in the presence of statistically different uncertain parameters (e.g., different patient classes).

Similar to recent work of \cite{Sun2014} and \cite{Levi2014}, we focus on an extension of one of the most fundamental models in scheduling theory, in which a single server must

\footnote{The paper is based on the first chapter.}
process a set of specified jobs, each associated with an a-priori random processing time specified by a given probability distribution. The goal is to minimize the expected sum of completion times (wait times). Unlike the traditional model, we consider the option of using the server for a specified duration of time to test jobs and reveal their exact processing time. Thus, at any stage, a decision must be made whether to test another job or process a job (either one that was already tested, or one that still has an uncertain processing time). Once a job is processed it must be completed (i.e., preemption is not allowed.) Without the option to test, the problem is known to be solved optimally by processing jobs in an increasing order of their expected processing times: the well-known Shortest Processing Time rule (SPT).

Unlike [Sun et al. (2014)] and [Levi et al. (2014)], we study a model in which jobs belong to classes with different stochastic characteristics. Intuitively, this captures cases in which the expected behavior of jobs could be similar, but where the magnitude of uncertainty varies across different classes of jobs. Thus, one has to decide not only whether to test an unknown job, but also which class of jobs should be tested (an issue that did not exist when all jobs are identically distributed). We consider broad problem settings where jobs are stochastically ordered according to the convex order (e.g., normal random variables with the same mean but with different variance are in convex order). For example, in emergency departments, the service provider may already be aware of differences in the service requirements of patients even before triaging them. Similarly, in engines maintenance, query optimization, and transmission over networks, the decision-makers may have prior information about the engines, queries, and channels, prior to engaging in testing activities.

**Contributions.** The contributions of this chapter are twofold. First, we derive a simple and intuitive optimal policy. Second, we provide a novel analysis that could be applied to other dynamic stochastic optimization problems.

More specifically, we characterize the structure of the optimal policy, and then use this to develop a novel Dynamic Programming formulation where the decisions epochs are moments in time where outcome of tests are realized. This allows us to characterize the set of feasible policies as a series of jobs processing followed by a test. We then find that any sequence of job
processing must be a prefix of a predetermined sequence. We also find that there exists an optimal order of testing where jobs that are higher in convex order (higher variance) are tested before jobs lower in convex order (smaller variance). Intuitively, this implies that it is always optimal to test the job with the highest uncertainty (as an extreme example, constants are smallest in the convex order and should be considered last for potential testing). Moreover, we show that every distribution is associated with a “testing threshold” that governs the timing of testing. In particular, at any point of time, there is a single candidate for testing, which is the job with the smallest testing threshold. When jobs are ordered according to the convex order, jobs with higher variability have lower testing thresholds (which is consistent with our finding about the optimal testing order). The optimal policy processes all the jobs that are shorter than the smallest testing threshold, and proceeds by either testing (the unique testing candidate), or processing all jobs without ever testing again. Finally, we show that the decision about whether to test or process all jobs can be determined optimally using a simple myopic (local) rule, which essentially computes the expected marginal improvement from a single test. It is optimal to test, if and only if, the expected improvement is positive.

The analysis is based on an equivalent yet not very common interpretation of convex orders ([Müller and Stoyan (2002)]). At the center of the analysis is the idea that stochastic orders (and in particular, the convex order) can be seen as a finite series of transformations of distributions. This series leads from a smaller distribution to a larger distribution (in the stochastic order sense) where each transformation makes a marginal difference to the previous distribution in the sequence. Together with a few other properties, we show that to prove the structural results on the testing order, it is sufficient to analyze the case when the jobs are only marginally different.

**Literature review.** A significant amount of the literature in Operations Research is dedicated to resource allocation problems, which traditionally have been divided to deterministic and stochastic models. The focus in most of these work, however, is on optimization, and seldom on uncertainty reduction or learning (see [Pinedo (2012b)] for sample work on Scheduling problems). One specific related domain is Maintenance where inspection models have been
studied. However, the common view of inspection is as a mandatory activity that needs to be scheduled under certain time constraints, and not as an activity with impact on learning (see Levi et al. (2014) and the reference therein).

Closest to our work are papers by Sun et al. (2014) and Levi et al. (2014) who consider the problem of scheduling with testing jobs with random processing times and weights by a single server with the objective of minimizing the expected weighted sum of completion times. In these two papers, unknown jobs belong to a single homogeneous class characterized by a joint distribution. A policy in these settings determines whether processing jobs or testing should take place. In contrast, we consider a model where each job belongs to a class that is associated with a processing times distribution. In this generalized setting, there is an additional layer of complexity where in addition to deciding on whether to test or not, one must also decide which job to test. As we shall see, generalizing the model assumptions requires a considerably different analysis comparing to that used in previous work.

Another related body of work deals with learning problems, such as exploration-exploitation tradeoffs (e.g., Besbes and Zeevi (2009)), multi-armed bandits (Gittins et al. (2011), Bubeck and Cesa-Bianchi (2012)), and optimal learning (Powell and Ryzhov (2012)). In this line of work, learning is embodied through sampling. For example, in the Bayesian framework, there is a prior distributions that is updated with every additional sample and becomes more accurate. In contrast, in our setting, distribution are known, and there exists an uncertainty reduction mechanism in which realizations from known distributions can be observed through testing. Put it differently, in the traditional exploration-exploitation framework, uncertain parameters are tested again and again in order to learn how a general population behaves, while in our setting the underlying assumption is that distributions are known, and testing reduces the uncertainty with respect to individuals instances.

Our work can be classified as a study of uncertainty reduction in stochastic optimization problems. Related work include Alizamir et al. (2013) who study a tradeoff between congestion and diagnostic accuracy of patients. They study an M/M/1 model where the server can perform diagnostics on patients. Patient’s assessment is described by a Markov chain
that evolves with tests, and at any point of time a decision needs to be made on whether to finalize a diagnosis based on current information, or spend more time on diagnosing, which in turn delays patients. While this work studies uncertainty reduction, it does not include scheduling decisions, which are important levers in many operational problems. Megow et al. (2015) considered the problem of finding the minimum spanning tree when edges’ weights are unknown. Testing an edge reveals its weight and the goal is to find a policy that minimizes the testing cost while guaranteeing finding the best tree. A worst case analysis approach is taken by the authors, and a randomized algorithm with a near-optimal guarantee is proposed. Another related work research was done by Golovin and Krause (2011) who developed a framework for stochastic optimization problems with a certain property of diminishing returns called Adaptive Submodularity. While to some extent this phenomenon can be observed in our work, we find the actual optimal policy while Golovin and Krause (2011) provides more limited results in the form of a near-optimal solution (for a broader class of problems).

3.2 The Convex Order

A basic assumption of our model is that there is a convex order relation between the processing time distributions associated with the different job classes. In this section we review the definition and main properties of the convex order. In addition, we describe the mean preserving local spread and show its relation to the convex order. These concepts and properties are key to the analysis in Section 3.4.

Definition

The convex order is a partial relation between probability distributions that is often used to model difference in risk profiles (see Müller and Stoyan (2002)). Intuitively, distributions that are higher in convex order are more variable or spread around their mean values, and therefore, have a higher risk of undesirable outcomes. Formally, we say that $X$ is less than
Figure 3-1: Three normally distributed random variables with equal mean and different variance.

$Y$ in convex order (written $X \leq_{cx} Y$), if $\mathbb{E}[u(X)] \leq \mathbb{E}[u(Y)]$ for all real convex functions $u$ such that the expectations exist.

**Examples**

Figure 3-1 illustrates an example of convex order between continuous random variables. Three normally distributed random variables with equal mean and different variance are monotonically increasing in the convex order. That is, distributions with a higher variance are relatively higher in convex order (e.g., the distribution denoted by 3 is higher than the distributions denoted by 1 and 2, in both variance and convex order). An example of a convex order relation in discrete distributions is presented in the left side of Figure 3-2. In the figure, we see three discrete distributions that have the same mean value (10.5), but are different otherwise. In particular, the two point distribution is higher in convex order than the uniform distribution, which is higher in convex order than the binomial distribution.

**Properties**

A fundamental property of convex order is that the mean value of two random variables that are in convex order is identical (using the convex function $u(x) = x$ and $u(x) = -x$).
The convex order also implies increasing order of variances. That is, if \( X \leq_{cx} Y \), then \( Var(X) \leq Var(Y) \) (the variance function is convex). Finally, if the function \( v \) is concave, then the following holds:

\[
X \leq_{cx} Y \iff \forall \text{concave function } v \ (\text{where expectations are well defined}) : \mathbb{E}[v(X)] \geq \mathbb{E}[v(Y)].
\]

Note that two random variables \( X \) and \( Y \) that are identically distributed (that is, \( X \overset{d}{=} Y \)) also satisfy convex order.

**Verification**

There are many advantages to using the convex order, both in terms of modeling and analysis. In addition, there are practical benefits as there are established ways of verifying the existence of a convex order relation. One way, is to compare the cumulative distribution functions (CDF). If two distributions have the same mean, and have single intersection in their CDF functions, then they relate in the convex order with the “inner” distribution being the smaller (see Shaked and Shanthikumar (2007)). Note that this is the case for the discrete and continuous distributions discussed earlier (e.g., the right side of Figure 3-2). (See Shaked and Shanthikumar (2007) and Müller and Stoyan (2002) for other sufficient conditions to when random variables satisfy convex order.)
The Mean Preserving Local Spread

Perhaps the simplest type of convex order between discrete random variables is the mean preserving local spread:

**Definition 3.2.1.** *(From Müller and Stoyan (2002))* Let $F$ and $G$ be distribution functions of discrete distribution whose common support is a finite set of points $x_1 < x_2 < \ldots < x_n$ with probability mass function $f$ and $g$ respectively. Then $G$ is said to differ from $F$ by a local spread, if there exists some $i \in 2, 3, \ldots, n-1$ such that $0 = g(x_i) \leq f(x_i), g(x_{i-1}) \geq f(x_{i-1}), g(x_{i+1}) \geq f(x_{i+1})$, and $g(x_j) = f(x_j)$ for all $j \notin i-1, i, i+1$. A local spread is said to be mean preserving if $F$ and $G$ have the same mean. Write $F \leq_{LS} G$ if $G$ is a mean preserving local spread of $F$.

Intuitively, if distribution $G$ can be obtained from a distribution $F$ by splitting the probability mass in one point of the support (which we call the focal point) to two adjacent points while preserving the mean, then $G$ is a mean preserving local spread of $F$. Figure 3-3 illustrates two distributions $F$ and $G$ defined over the support $\{1, 2, \ldots, 7\}$, where the probability mass in $x = 3$ (the focal point) was shifted to $x = 2$ and $x = 4$. Observe that the conditions for the local spread are satisfied: (1) for $x \in \{1, 5, 6, 7\}$ : $g_x = f_x$; (2) $g(2) = f(2) + \epsilon_1$; (3) $g(3) = 0$, $f(3) = \epsilon_1 + \epsilon_2$; (4) $g(4) = f(4) + \epsilon_2$.

We assert that analyzing distributions that are in local spread is significantly simpler...
than analyzing arbitrary distributions that are in convex order. For example, comparing the expected values of some function of two random variables that are in local spread is more tractable than comparing the expected values of arbitrary distributions since that many of the terms will cancel each other.

It is readily seen that local spread implies a convex order between distributions. Interestingly, the opposite direction also holds. That is, if two random variables satisfy convex order, then there exists a series of distributions leading from one distribution to another, where every two consecutive distributions are in local spread. This is stated formally in the following theorem.

**Theorem 3.2.2.** (From Müller and Stoyan (2002)) Let $F$ and $G$ be distribution functions of discrete distribution with finite support. Then $F \preceq_{cx} G$ holds if and only if there is a finite sequence $F_1, ..., F_k$ with $F_1 = F$ and $F_k = G$, such that $F_i \preceq_{ls} F_{i+1}$ for $i = 1, ..., k - 1$.

As we shall demonstrate, Theorem 3.2.2 will be critical to the analysis where we essentially reduce a model with general convex order relations to a model where there is a single pair of distributions that are in local spread.

### 3.3 Problem Formulation

In this section, we introduce the non-homogeneous scheduling with testing model. This is a generalization of the model studied by Levi et al. (2014), and we therefore use similar (but extended) notation. Throughout this section, we draw the similarities between the basic and our generalized models.

We start by describing the model and notation used in this work (Section 3.3.1). We then develop a Dynamic Programming formulation (Section 3.3.2) for the problem. Finally, we review the main results for the special case, in which jobs are homogeneous. This would be used in our work as boundary conditions (Section 3.3.3) in proving properties about the more general non-homogeneous model.
3.3.1 Notation and Assumptions

We consider the problem of scheduling $N$ jobs with random processing times on a single server with the objective of minimizing the expected sum of completion times (which is equivalent to minimizing the total and average wait times). Each job belongs to a class, that is characterized by a discrete distribution of the processing time. Without loss of generality, we assume that there are $N$ distributions, and use $T_j$ to denote the random processing time of job $j$.

Preemption is not allowed and so if a job is scheduled for processing, the server must work on that job for the entire (potentially random) processing time. After processing is complete the job leaves the system. Jobs can also be tested by the server, which we assume to take a fixed amount of time $t_a$, during which the server is working. After testing a job, the a-priori random processing time is observed and becomes known with certainty. Jobs can be tested one at a time, and there is independence between the processing times of different jobs. We refer to a job with random processing times as “unknown”, and to a tested job as “known”. Unknown jobs may be processed either after testing, in which case the processing time is deterministic, or may be processed without testing, in which case the processing time is random.

We use the convex order to model the different magnitude of uncertainty among the different jobs (see Section 3.2). That is, we assume that there is a complete ordering between unknown jobs, and use smaller indices to denote jobs that are higher in the convex order:

$$T_N \leq_T T_{N-1} \leq_T \ldots \leq_T T_1.$$

In addition to the $N$ unknown jobs, at any point of time there are $n$ known jobs, which durations had been previously observed via testing, but have not yet been processed. We denote the processing time of the known jobs by $t_1, \ldots, t_n$. We use the convention that lower indices correspond to shorter jobs, that is, $t_1 \leq t_2 \leq \ldots \leq t_n$.

Our goal is to find an optimal adaptive policy that decides on the next job to be tested
or processed, at any moment of time in which the server becomes available.

### 3.3.2 Dynamic Programming Formulation

The problem can be naturally described using Dynamic Programming. As we shall see, the resulting formulation will be highly-dimensional which makes it impractical to solve the problem as is. Nevertheless, the DP formulation can be used to derive structural properties that would eventually lead a complete characterization of the optimal policy.

We start with describing the state space.

**State space.**

The system state can then be described by the random variables representing unknown jobs, and the realizations of the known jobs. We use both vectors and enumerated elements to denote the system state:

\[ (T, t) \equiv ([T_1, \ldots, T_N], [t_1, \ldots, t_n]) . \]

**Action space.**

At any state \((T, t)\), one can take three types of actions:

1. Process one of the known jobs \([t_1, \ldots, t_n]\). For example, processing known job \(i\) will result in the transition \((T, t) \rightarrow (T, t - t_i)\). (We use “-” and “+” to denote exclusion and inclusion of elements to sets.)

2. Process one of the unknown jobs \([T_1, \ldots, T_N]\). For example, processing unknown job \(j\) will result in the transition \((T, t) \rightarrow (T - j, t)\).

3. Test one of the unknown jobs \([T_1, \ldots, T_N]\). For example, testing unknown job \(j\) will result in a transition to a state without the unknown job \(j\) and with an additional realization. When referring to a specific realization \(d\) of the random variable \(T_j\), we write this transition as \((T, t) \rightarrow (T - j, t + d)\). When referring to the random state obtained from testing job \(j\), we write the transition as \((T, t) \rightarrow (T - j, t + T_j)\).
Figure 3-4: A simple policy that processes three known jobs consecutively: (a) the processing order; (b) the completion time of each job (the total objective is the total grayed area).

Cost function.

We now define the cost accounting method, that is, the costs associated with taking actions at each state. The traditional way for accounting costs (denoted by TA) is by summing the completion time of each job when the respective job is processed. An alternative, is to use Echelon cost accounting (EA) where for every action, we add to the objective the duration of the respective action multiplied by the total number of jobs in the system.

Figure 3-4a illustrates a simple policy that processes three known jobs sequentially, with durations denoted as $t_1$, $t_2$, and $t_3$. In Figure 3-4b, we see the completion time of each of the respective jobs, and the resulting objective which is the total area of all blocks. Figure 3-5 illustrates how costs are accumulated under the traditional and the echelon cost accounting methods. Specifically, we see that when the processing of job 2 takes place, the traditional cost accounting method (Figure 3-5a) summed the completion times of the jobs which had been processed, with a total duration of $(t_1) + (t_1 + t_2)$, where the terms correspond to the first and second actions, respectively. In contrast, under the echelon cost accounting method (Figure 3-5b), the cost accounted for are: $(3t_1) + (2t_2)$, where each term once again corresponds to the first and second actions, respectively. Graphically, in TA, blocks are accumulated from top to bottom, whereas in EA, blocks are covered from left to right. When the server terminates, the exact same area is covered by the two methods.

Note that while the cost accounting method does not affect the optimal policy and the total costs accumulated by each policy, we find that an “Echelon” cost accounting method is most convenient for our particular analysis (in contrast to Levi et al. (2014) who defined
Figure 3-5: The costs accumulated using different cost functions: (a) traditional cost accounting; (b) echelon cost accounting (the accumulated costs correspond to the grayed area).

and used a different cost accounting method called the Marginal Cost Accounting).

**Bellman’s equation.**

The DP formulation can be written as follows:

\[
J(T, t) = \min \begin{cases} 
(N + n)t_a + \mathbb{E}[J(T - j, t + T_j)] & \text{test}_j \\
(N + n)\mathbb{E}[T_j] + J(T - j, t) & \text{process}^n_j \\
(N + n)t_i + J(T, t - t_i) & \text{process}^k_i 
\end{cases} 
\tag{3.1}
\]

\[
J(\emptyset, \emptyset) = 0,
\]

where the upper scripts \(k, u\) refer to known and unknown jobs, respectively; the lower scripts \(i, j\) refers to the index of the known or unknown job, respectively.

It is easy and useful to show that the following property hold:

**Lemma 3.3.1.** The value function \(J([T_1, ..., T_N], [t_1, ..., t_n])\) is concave in \(t_i\) (given that all other problem parameters are fixed).

**Proof.** We prove by induction on the number of actions. The base case \(N = n = 0\) trivially holds. For the step, observe that the value function \(J(T, t)\) is defined as the minimum of three functions, which are concave (from the induction hypothesis). Therefore, \(J(T, t)\) is a concave function of \(t_i\). \(\square\)
3.3.3 Scheduling and Testing Homogeneous Jobs

If all jobs have identical (homogeneous) distributions, there is effectively a single class with a corresponding processing time distribution. [Levi et al. 2014] characterized the optimal policy for the homogeneous model using two thresholds. The processing threshold defined to be the mean processing time $\mu$, and the testing threshold $\mu^a$ defined as the solution to the equation:

$$t_a = \mathbb{E}[(\mu^a - T)^+]$$

(3.2)

where $T$ is a random variable from the single class processing time distribution. The two thresholds divide the known jobs into three groups of jobs: short, intermediate, and long jobs, depending on whether the duration of the known job is lower, in between, or above the two thresholds.

The optimal policy for the homogeneous model can be described as follows:

- One should always process short jobs before processing or testing other jobs. This implies that short jobs are processed immediately upon detection,

- Long jobs should be processed only after all other jobs have been tested or processed,

- After processing short jobs, the optimal policy either tests an unknown job and processes the respective job if it happens to be short, or processes all jobs in a non-decreasing order of their expected processing times (without ever testing again),

- It is optimal to test if a single test outperforms processing all jobs (on expectation), and

- If $\mu \leq \mu^a$ processing all jobs is optimal.

Observe that unlike the homogeneous model, in our problem there are $N$ different distributions, which means that there could be as many as $N$ different testing thresholds. Some of the testing thresholds could be higher than $\mu$ while the other testing thresholds could be lower than $\mu$. It is a-priori not clear which of the properties would generalize and how.
Moreover, while in the homogeneous model the analysis relied on comparing testing and processing jobs, in our model one can test different jobs which increases the action space by an order of magnitude, and the state space by an exponential factor. This also means that to analyze this problem, we need to be able to compare different test actions, which from a technical point of view, translates to computing the expected value functions of two different random variables (a result of testing different jobs). Note that applying a sample path argument is not trivial because the evolution of the states could be considerably different when testing different jobs. As we shall see in Section 3.4, the local spread will be key to simplifying the analysis.

Finally, we recall some definitions and properties from Levi et al. (2014) that will be useful throughout the work.

**Definition 3.3.2.** Process all (PA) is the policy in which all jobs are processed in a non-decreasing order of their expected value.

Since processing all means that one would never test again, we will also use the term stopping for this policy (as in stop testing).

**Definition 3.3.3.** The Single test policy (STP) is the policy in which a single unknown job is tested before processing all jobs in a non-decreasing order of their expected value (assuming there is at least one unknown job).

**Lemma 3.3.4.** For any state \( (N, [t_1, ..., t_n]) \) with no short jobs and \( N > 0 \), the difference in the objective value between policies STP and PA is equal to:

\[
J^{STP} - J^{PA} = (N + n) t_a - (N - 1) E[(\mu - T)^+] - \sum_{i \in \text{Inter.}} E[(t_i - T)^+] - \sum_{i \in \text{Long}} E[(T - (\mu - T)^+)]
\]

Note that while the STP policy is well defined in the homogeneous model, in the non-homogeneous model it must to be extended to specify which of the unknown job is tested.
3.4 Analysis

In this section, we analyze the non-homogeneous scheduling with testing model. We gradually derive structural properties, which culminates in a complete characterization of an optimal policy.

We start by generalizing the definition of the testing threshold (Equation 3.2) to the non-homogeneous case, and the definitions of short, intermediate, and long jobs (Section 3.4.1). We then prove the optimality of certain processing priorities (Section 3.4.2), followed by a partial characterization about when testing is sub-optimal (Section 3.4.3). In Section 3.4.4 we summarize all the properties proved earlier and formulate a more compact Dynamic Programming formulation. In Section 3.4.5 we derive an optimal, necessary, and sufficient condition for processing all jobs without testing, and in Section 3.4.6 we find the optimal testing order. We summarize the optimal policy and discuss the managerial insights in Section 3.4.7.

3.4.1 Generalized testing thresholds

In the homogeneous model, there is a single processing time distribution, and as a result, there is a single testing threshold. Under the generalized model we could have as many as \( N \) different testing thresholds corresponding to each of the distributions:

**Definition 3.4.1.** The testing threshold \( \mu_j^a \) of unknown job \( j \) is defined as the solution to the equation:

\[
\mu_j^a : t_a = \mathbb{E}[(\mu_j^a - T_j)^+].
\]  

(3.4)

An interesting property of the testing threshold is that there is a decreasing monotonic relation between convex order and the testing threshold. That is, distributions that are higher in convex order are associated with lower testing thresholds. Using our convention that \( T_j \preceq_{cx} T_{j+1} \), we can say that the testing threshold \( \mu_j^a \) is monotonically increasing in \( j \) as given by the following lemma.

**Lemma 3.4.2.** If \( T_i \succeq_{cx} T_j \), then \( \mu_i^a \leq \mu_j^a \).
Proof. Using Definition 3.4.1

\[ t_a = \mathbb{E}[(\mu^a_j - T_j)^+)] = \mathbb{E}[(\mu^a_i - T_i)^+)] \geq \mathbb{E}[(\mu^a_i - T_j)^+)], \]

where the inequality follows from the convexity of \( f(X) = (t - X)^+ \), and the convex order \( T_i \geq_{cx} T_j \). The monotonicity of \( f(X) \) implies that \( \mu^a_j \geq \mu^a_i \).

Having multiple testing threshold necessitates redefining the groups of jobs (Section 3.3.3). We define as short jobs the known jobs which duration is smaller than or equal to \( \mu^a_1 \) and \( \mu \); we define as intermediate jobs the known jobs which duration lies within \( \mu^a_1 \) and \( \mu \); and we define long jobs to be the known jobs which durations are strictly higher than both \( \mu^a_1 \) and \( \mu \). At each system state, we denote by \( n_s, n_i, \) and \( n_l \) the number of short, intermediate, and long jobs, respectively. Note that unlike the the homogeneous case, in this model the definition of short jobs is dynamic because testing thresholds may disappear when the corresponding unknown jobs are tested.

Figure 3-6 illustrates the aforementioned definitions. The axis represents the processing time, and jobs are placed on the axis according to their expected processing times. That is, unknown jobs (denoted by circles) are placed at their mean value \( \mu \), while the known jobs (denoted by crosses) are placed according to their exact processing times. The figure illustrates a state with 4 unknown jobs and 9 known jobs, where jobs 1 and 2 are short, jobs 3,4, and 5 are intermediate, and jobs 6,7,8, and 9 are long.
3.4.2 Optimal Processing Order

We now prove properties that describe the optimal order in which jobs should be processed. Some of these properties can be proved using a simple interchange argument and are therefore omitted.

We start with a property that describes the optimal order of processing known jobs:

**Lemma 3.4.3.** There exists an optimal policy that for each state $(T, t)$ processes known job 1 before any other known job.

**Proof.** A simple interchange argument.

According to Lemma 3.4.3, we can discard the control $\text{Process}_i^k$ where $i > 1$ without hindering optimality.

The following two lemmas characterize the optimal order between processing the shortest known job, and processing unknown jobs.

**Lemma 3.4.4.** There exists an optimal policy that at each state $(T, t)$ in which $t_1 \leq \mu$, the policy either process job 1, or tests one of the unknown jobs.

**Proof.** A simple interchange argument.

**Lemma 3.4.5.** At each state $(T, t)$ where $\mu < t_1$, processing a known job is sub-optimal.

**Proof.** A simple interchange argument.

Lemmas 3.4.3, 3.4.4, and 3.4.5 imply the following corollary:

**Corollary 3.4.6.** There exists an optimal policy that at each state $(T, t)$ does one of two:

1. Test an unknown job,

2. Processes job 1 when $t_1 \leq \mu$, and processes an unknown job when $\mu < t_1$.

Next, we describe the optimal order of processing unknown jobs:

**Lemma 3.4.7.** There exists an optimal policy in which unknown jobs are processed in an increasing convex order.
Proof. See Appendix [B.1].

The latter implies that the controls Process \( j \) can be discarded for each \( j < N \) (by definition, \( T_N \) is smallest in convex order).

We can synthesize the previous properties into a unified theorem:

**Theorem 3.4.8.** For each state \((T, t)\), any sequential processing of jobs is a prefix of a predetermined series \( S(T, t) \):

\[
S(T, t) = \begin{cases} 
    \{t_1, t_2, ..., t_{n_s+n_1}, T_N, T_{N-1}, ..., T_1, t_{n_s+n_1+1}, ..., t_n\}, & \text{for } \mu_1^a \leq \mu \\
    \{t_1, t_2, ..., t_{n_s}, T_N, T_{N-1}, ..., T_1, t_{n_s+1}, ..., t_n\}, & \text{for } \mu < \mu_1^a
\end{cases}
\]

**Proof.** Immediate from lemmas [3.4.3] [3.4.4] [3.4.5] and [3.4.7].

The theorem maintains that at every state there is exactly one candidate for processing, which is given by the theorem. Note that the sequence depends on the state, which evolves as one tests and processes jobs.

A complete characterization must also specify which of the unknown coefficients should be tested (when testing), and how to decide between processing and testing. A partial answer to the latter is given in the next section.

### 3.4.3 Sub-optimal Testing

In this section, we identify several cases in which processing a job is optimal.

**Lemma 3.4.9.** For each state \((T, t)\) where \( t_1 \leq \min(\mu_1^a, \mu) \), there exists an optimal policy that processes job 1 at that state.

**Proof.** See Appendix [B.2].

Lemma [3.4.9] implies that short jobs should be processed immediately upon detection. Without loss of generality, we can therefore assume that there are never short jobs in the system. That is, we can assume that \( n_s = 0 \).
Lemma 3.4.10. At each state \((T, t)\), if \(\mu_j^a \leq t_1\), then for each policy that first processes the known job 1 and then tests job \(j\), there exists a dominating policy that first tests unknown job \(j\) before processing job 1.

Proof. See Appendix B.3

Intuitively, Lemma 3.4.10 implies that the testing thresholds of unknown jobs could be comparable with the processing times of known jobs. If we place on an axis jobs according to their expected processing times and testing thresholds (of unknown jobs), we observe that it is optimal to process the shortest jobs in increasing order until we reach the smallest testing threshold (Lemma 3.4.9). Moreover, testing an unknown job after processing a known job whose duration exceeds the respective testing threshold is suboptimal (Lemma 3.4.10). While these two lemmas partially describe the optimal testing and processing policy, in later sections, we will see that there is, in fact, an optimal policy that processes and tests jobs according to the order of processing times and testing thresholds.

We next show that when all testing thresholds exceed the mean processing time, testing is never optimal. In this case, processing all jobs according to their expected processing times is optimal.

Lemma 3.4.11. At any state \((T, t)\), if for all \(j\) \(\mu_j^a \geq \mu\), then PA is an optimal policy.

Proof. See Appendix B.4

Lemma 3.4.11 specifies the optimal course of actions when the testing thresholds exceed the mean processing time. This, for example, may happen when the unknown jobs in the initial state of the system have high testing thresholds. It could also happen for an initial state that contain testing thresholds that are both below and above the mean processing time, after the unknown jobs corresponding to the lower testing thresholds were tested.

Without loss of generality, from now on we will assume that \(\mu_1^a < \mu\) (otherwise policy PA is optimal).
3.4.4 The Revised DP Formulation

In this section we leverage the optimality properties proven earlier to formulate a more compact DP problem denoted as the Revised Dynamic Programming Formulation. This is an intuitive formulation that consolidates properties proved earlier, which we use in later sections to further characterize the optimal policy.

We start with an observation. Every policy to our problem describes a series of testing and process actions. When we test a job, the realization of the respective job could affect future decisions as new information is revealed. On the other hand, processing a job have no impact on future decisions because the respective job leaves the system (there is no new information from processing a job that could affect our future decisions). This means that, the decision-maker makes at most \( N \) “effective” decisions, each about a sequence of jobs to process, and a job to test.

We can therefore formulate a new DP over the same state space, where every control corresponds to a sequence of jobs to process and a job to test (in addition to the control that corresponds to processing all jobs). From Theorem 3.4.8 we know that there exists an optimal policy in which every sequential processing of jobs is a prefix of a unique sequence. Therefore, the set of jobs to processed before testing can be described simply by stating the size of the prefix, which varies between 0 to \( N + n - 1 \). We denote by \((k, j)\) the control in which a prefix of size \( k \) is chosen prior to testing unknown job \( j \).

Moreover, the optimal policy need not process all unknown jobs before testing. To see this, observe that when \( k > n_i + N - 1 \), all unknown jobs but one are processed, in which case unknown job \( j \) is the last remaining job. If \( \mu^a_j \geq \mu \) testing \( j \) is not optimal (according to the homogeneous model), and if \( \mu^a_j < \mu \), then we know that there exists an optimal policy that tests prior to process a job which duration is longer than \( \mu^a_j \) (Lemma 3.4.10).

We can therefore write Bellman’s Equation for the Revised Dynamic Programming Problem as follows:

\[
J(T, t) = \min \{J_{k,j}(T, t), J_{PA}(T, t)\} 
\] (3.5)
Recall that there are no short jobs, and therefore there are two possible ranges based on the value of $k$:

1. $0 \leq k \leq n_i$: only intermediate jobs are processed,

2. $n_i + 1 \leq k \leq n_i + N - 1$: all intermediate jobs, and some or all of the unknown jobs are processed.

One can think about the Revised DP as an optimal stopping time problem. At each decision epoch, one can either choose to continue (process a collection of jobs and test) or to stop (process all). This is similar to the interpretation given by Levi et al. (2014) for the homogeneous model, with the difference that in our model, there is more than one way to continue, as one can choose different subsets of jobs to process and a job to test. By the end of Section 3.4 we will see that there is a single value of $(k, j)$ that should be considered when deciding to test. The model then perfectly reduces to an optimal stopping time problem with two controls for continuing and stopping.

### 3.4.5 Myopic Stopping Rule

Each policy to problem formulated by the Revised DP can be decomposed into two decisions:
1. Decide whether to stop and process all jobs or continue and perform more tests (choose between PA and \((k, j)\));

2. When testing, decide on the prefix length and which job to test (choose the control \(\text{Test}_{k,j}\)).

In this section we address the first question, and show that a simple myopic rule can be used to optimally decide when to process all jobs. We start with a few definitions.

**Definition 3.4.12.** At any state \((T, t)\), the Single Test Policy \(\text{STP}_{k,j}\) selects the control \((k, j)\) and stops. We denote by \(J^{\text{STP}}_{k,j}\) the expected cost of policy \(\text{STP}_{k,j}\).

Note that the above definition perfectly generalizes Definition 3.3.3 for the homogeneous model, where there is exactly one type of unknown job (and therefore testing any job \(j\) is equivalent), and where the optimality conditions enforce the value of \(k\) to be zero. For this reason, in the homogeneous model there is exactly one STP policy.

**Lemma 3.4.13.** At any state \((T, t)\) with no short jobs, the difference in the objective between policies PA and \(\text{STP}_{k,j}\) is given by the following expression:

\[
J_{\text{PA}}(T, t) - J^{\text{STP}}_{k,j}(T, t) = -(N + n - k)t_a + \sum_{m=k+1}^{n_i} \mathbb{E}[(t_m - T_j)^+] \\
+ \sum_{m=n_i+1}^{n} \mathbb{E}[(T_j - t_m)^+] + (N - 1 - (k - n_i)^+)\mathbb{E}[(T_j - \mu)^+].
\]  

**Proof.** See Appendix B.5

An intuitive way of thinking about the difference between policies PA and STP is as the expected marginal improvement from a single test. In the next lemma (Lemma 3.4.14), we show that testing unknown job 1 without processing any of the known jobs (i.e., the control \((k = 0, j = 1)\)) obtains the highest marginal improvement. We then show that once we reach a state in which the highest marginal improvement is non-negative, it will remain non-negative in the next state if we test any of the unknown jobs (Lemma 3.4.15). The latter
property is used in Theorem 3.4.16 to show that it is optimal to stop testing and process all jobs, if and only if, the maximal expected marginal improvement is non-negative.

Lemma 3.4.14. At any state \((T,t)\) with no short jobs, policy \(STP_{0,1}\) obtains the highest marginal increase to the objective:

\[
J_{PA}(T,t) - J_{STP_{0,1}}(T,t) \geq J_{PA}(T,t) - J_{STP_{k,j}}(T,t),
\]

or equivalently,

\[
J_{0,1}^{STP}(T,t) \leq J_{k,j}^{STP}(T,t).
\]

Proof. See Appendix B.6

We next show that under the optimal policy, a certain monotonicity is being preserved. Specifically, that if processing all jobs is better than any SPT policy, then processing all jobs will remain better than any SPT policy if we still decide to test.

Lemma 3.4.15. At any state \(s\) with no short jobs, if processing all jobs is better than any admissible \(SPT_{k,j}\) policy, then the same will hold for any state \(s'\) reached after choosing the control \(k,j\):

\[
J_{PA}(s) \leq J_{STP_{0,1}}(s) \iff J_{PA}(s') \leq J_{STP_{0,1}}(s').
\]

Proof. See Appendix B.7

We now use the monotonicity property in the previous lemma to show that it is optimal to process all jobs if policy \(PA\) outperforms all STP policies.

Theorem 3.4.16. At any state \((T,t)\) with no short jobs, policy \(PA\) is optimal, if and only if, it is better than policy \(STP_{0,1}\):

\[
J_{PA}(T,t) \leq J_{0,1}^{STP}(T,t) \iff J_{PA}(T,t) = J(T,t).
\]

Proof. See Appendix B.8
Theorem 3.4.16 provides an optimal, sufficient, and necessary condition for processing all jobs. This is a myopic rule, in which the condition can be computed based on the current state without solving the entire DP formulation. The theorem provides a closed form formula to evaluate if the condition is met.

Note that this optimal stopping rule depends only on the threshold of the job that is highest in convex order, and not on the distributions of the other random variables. This seems to suggest that unknown job 1 may be the only candidate for testing. In the next section, we show that this is indeed the case. Before we move on to the next section, we conclude with a few insightful corollaries that will be used later in the analysis.

**Corollary 3.4.17.** At every state with no short jobs \((T, t)\), if the optimal policy is to process all jobs, then processing all jobs will be optimal also at states \((T - j, t)\) and \((T - j, t + x_m)\), for any value of \(j\) and \(x_m\).

*Proof.* Immediate from Lemma 3.4.15 and Theorem 3.4.16.

**Corollary 3.4.18.** At any state \((T, t + x)\), the support could be divided into four consecutive ranges \(A, B, C, D\), which determine the next optimal decision: (Figure 3-7)

1. \(x \in A\): Process \(x\)
2. \(x \in B\): Process all
3. \(x \in C\): Process and test
4. \(x \in D\): Process all

*Proof.* See Appendix B.9.
Corollary \textit{3.4.18} and Theorem \textit{3.4.16} tell us that the closer the realization of a tested job is to $\mu$, the more likely we are to test. Intuitively, testing is most valuable when jobs are closer to each other, in which case testing allows us to differentiate the jobs and process them in the right order. After some testing, the jobs become more distinguishable and the value from testing decreases. The monotonicity of STP policies implies that at a certain point the expected value of testing becomes smaller than the respective cost, in which case we stop testing and process all jobs.

\subsection*{3.4.6 The Optimal Testing Order}

In this section we prove that there exists an optimal policy which tests unknown jobs in a decreasing convex order. That is, the only testing control one should consider is $(k, j = 1)$. As we shall see in Section \textit{3.4.7}, the next theorem together with previously proved properties would provide a complete characterization of the optimal policy.

\textbf{Theorem 3.4.19.} There exists an optimal policy in which unknown jobs are tested in an increasing order of their testing threshold (or in a decreasing convex order).

\textit{Proof.} To assist the flow of reading, the proof has been moved to Appendix \textit{B.10}. Nevertheless, the reader is encouraged to walk through the proof for an interesting proof technique. In particular, the proof involves a reduction of a stochastic optimization problem with a general convex order relation, to a problem where there is a more simple mean preserving local spread relation between probability distributions. \hfill $\square$

Theorem \textit{3.4.19} states that we would always prefer testing unknown with higher convex order before testing unknown jobs with lower convex order. An intuitive explanation is that it is better to test jobs from which we can gain more information. For example, consider the extreme case of an unknown job that has a constant value of $\mu$. This job is smallest in convex order and we would not gain any benefit from testing that job. Another explanation is that the objective of minimizing the sum of completion times drives us to process short jobs first. In our model, testing enables identifying short jobs, and we are more likely to
identify a short job when we test a job with higher variability or convex order (which is why we should test jobs that are higher in convex order first).

### 3.4.7 The Optimal Policy

We now synthesize the properties proved thus far to describe the optimal policy (Algorithm 3).

**Algorithm 3** The optimal policy.

1: while $\mu_1^a < \mu$ do
2:   Process jobs whose duration is shorter than or equal to $\mu_1^a$ in a non-decreasing order of processing times.
3:   if the following condition is satisfied:
   
   $$(N + n)t_a \leq \sum_{m=1}^{i} \mathbb{E}[(t_m - T_1)^+] + \sum_{m=i+1}^{n} \mathbb{E}[(T_1 - t_m)^+] + (N - 1)\mathbb{E}[(\mu - T_1)^+], \quad (3.9)$$

   then
4:     Test unknown job 1.
5:     Update the state (decrease the index of unknown jobs by 1, and add the realization of the tested job to the known jobs vector).
6:   else
7:     Exit loop.
8: end if
9: end while
10: Process all jobs in a non-decreasing order of their expected processing times.

**Theorem 3.4.20.** Algorithm 3 describes an optimal policy.

*Proof.* See Appendix B.11

Figure 3-8 illustrates the optimal policy for a certain sample path. At the initial state (Figure 3-8a), there are 9 known jobs (denoted by crosses) whose durations are $t_1, ..., t_9$, and 4 unknown jobs (denoted by circles) corresponding to each of the 4 testing thresholds. Since that $\mu_1^a < \mu$ we proceed to Step 2 where the short jobs $t_1, t_2$ are processed (Figure 3-8b). In Step 3, we check if the condition is satisfied. Suppose that it is, in which case we test unknown job 1. The testing threshold $\mu_1^a$ disappears as there is no longer an unknown job
Figure 3-8: The structure of the optimal policy. Crosses and circles denote known and unknown jobs, respectively.

associated with that threshold. In addition, a realization from distribution $\mathcal{D}_1$ is added to the axis (Figure 3-8c). The now smallest testing threshold $\mu^a_2$ is smaller than $\mu$ and we continue to Step 2 and process the short jobs $T_1, t_3, t_4$ (short jobs are defined with respect to the threshold $\mu^a_2$, Figure 3-8d). Going back to step 3, we need to decide if we should test unknown job 2 or process all jobs. Suppose that the condition of Step 3 does not hold. In this case, we simply process all the job in an increasing order of their expected processing times: $t_5, T_4, T_3, T_2, t_6, t_7, t_8, t_9$, which determines the final schedule.

Note that the algorithm perfectly generalizes the optimal policy of the homogeneous problem (Section 3.3.3). Similarly, to the homogeneous model, the optimal policy can be interpreted as a greedy algorithm, where at any state, we choose a control (that is, a set of jobs to be processed and an unknown job to be tested) that maximizes the marginal expected improvement. If the expected marginal improvement is non-positive, we do not test at all and simply process all jobs.

Finally, we note another interesting outcome of the optimal policy that is given in the
following corollary.

**Corollary 3.4.21.** There exists an optimal policy in which unknown jobs with testing thresholds higher than the mean processing times are never tested.

*Proof.* Immediate from Algorithm 3.

### 3.5 Conclusions and Future Directions

In this chapter, we studied the problem of scheduling jobs with random processing times, in which the server can be used not only to process jobs, but also to test jobs to observe the actual processing times of the respective jobs. Unlike existing work that focused on a homogeneous model where the random processing times of jobs is identically distributed, we studied a model where jobs could be statistically different by satisfying a general convex order relation. We analyzed the problem and found that the optimal policy for the homogeneous model is a special case of our model. In particular, we defined thresholds and scheduling rules that perfectly generalize the optimal policy for the homogeneous model. Moreover, the optimal policy for our model is very intuitive and provides insights on when to test, and why testing certain jobs is preferred over testing other jobs. In our analysis, we used a novel technique, which in essence, reduces a stochastic optimization problem with random variables that are in convex order, to a stochastic optimization problem where the probability distribution of the random variables are only marginally different (i.e., satisfy a mean preserving local spread relation).

As our algorithm perfectly generalizes the optimal solution for the homogeneous model, an interesting question is whether an even more general algorithm exists that can optimally solve an even more general setting. In particular, a scheduling with testing model with arbitrary probability distributions. A first step would be to explore a more structured relation between distribution, such as the stochastic order where the probability distributions need not have the same mean value. (In the stochastic order, the cumulative distribution function (CDF) of one random variable is above the CDF of another random variable.)
Finally, we note that while our analysis relied on the assumption that the processing times distributions are in convex order, a similar approach can be used for other relations between distributions. For example, for the stochastic order, an equivalent for the mean preserving local spread is to shift a probability mass from a smaller to a larger value. Even more generally, one can connect any two arbitrary distributions by a series of distributions that are marginally different, where probability mass is shifted to smaller and higher values. We believe that this approach would prove to be useful in the analysis of other Dynamic Programming problems beyond the context of the specific problem studied in this chapter.
Chapter 4

Offline Testing in Combinatorial Optimization Problems

In the chapter\(^1\) we study a broad class of stochastic combinatorial optimization problems whose parameters are uncertain. These are problems that can be formulated as Linear Programs whose objective coefficients are random variables that can be tested, and where the constraint polyhedron has the structure of a polymatroid (Schrijver (2003)). We call these problems Linear Programs over Polymatroids with Testing (LPPT). At each step, the decision-maker can either test one of the random coefficients (which incurs a fixed cost), or optimize under uncertainty (that is, finalize the value of the decision variables). These are offline testing problems where testing decisions precede other controls (such as fixing decision variables). The goal of the decision-maker is to maximize the expected optimization value (i.e., the outcome of the linear program) minus the testing costs.

One example of an LPPT is concept selection in project management. A planner who attempts to minimize costs, needs to choose between different alternatives with random cost estimates (e.g., about suppliers, designs, or implementations). The planner can commit to a certain alternative potentially with random cost, or he could test one, that is, pay a fixed fee to reduce uncertainty about the respective alternative. The objective is to dynamically

\(^1\) The work in this chapter was carried out jointly with Chen Attias from the Weizmann Institute of Science. In particular, the results appear also in her MS thesis.
decide how much to invest in reducing uncertainty, as well as how to prioritize the uncertainty reduction of the different alternatives before committing to a certain alternative. This problem can be formulated as an LPPT, where each of the objective coefficients represent the (possibly random) valuations of different alternatives.

Another example is the Maximal Spanning Tree (MST) with Testing. In the MST problem, the decision-maker is given a weighted graph and has to find a tree that connects all nodes (i.e., a spanning tree) whose total edge weights are maximal. In the MST with Testing, the edge weights are random, and the decision-maker may pay a fixed cost to test edges, which in turn, reveals their respective realizations. Every policy for the problem has to specify how to adaptively choose edges to test, decide when to stop testing, and determine the resulting tree once the testing phase completed (both tested and random edges can be selected for the resulting tree). The goal is to find a policy that maximizes the expected weight of the resulting tree minus the testing costs. The MST problem with Testing can be formulated as an LPPT, in which each of the objective coefficients represent the weight of an edge, and the polymatroid represents the graph structure (which is not affected by testing). (See Section 4.2 for several additional examples.)

We begin by reviewing known results about polymatroid optimization problems (Section 4.1), and provide a few representative problems that can be modeled as polymatroids (Section 4.2). In Section 4.3 we formally formulate the LPPT problem. In Section 4.4 we show that when the random coefficients have equal means, a myopic stopping rule that compares stopping to testing once is optimal in deciding when to stop testing in LPPTs. We then show that in some special cases it is optimal to decide on the testing order myopically based on the expected improvement to the underlying optimization problem when we are limited to a single test. This includes the Maximum Spanning Tree problem when edge weights are independent and identically distributed (Section 4.5), and a certain type of LPPT we denote as symmetric, in which the underlying optimization problem is symmetric with respect to objective coefficients (Section 4.6). In the symmetric case, we allow the random coefficients to be drawn from different distributions that satisfy a convex order, which is a general re-
lation between probability distributions that have the same mean but different magnitude of uncertainty (we discuss the convex order in Section 4.6). We conclude in Section 4.7 and discuss future research directions.

4.1 Submodular Set Functions and Polymatroids

We now review some basic definitions and properties of submodular set functions and polymatroids, as stated in Schrijver (2003) and Yao and Zhang (1997).

Submodular set functions

Given a finite ground set $N$, a set function $f : 2^N \rightarrow \mathbb{R}$ is called submodular if for all subsets $A, B \subseteq N$,

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B).$$

(4.1)

Equivalently (see Schrijver (2003)), $f$ is submodular if for all subsets $A \subseteq B \subseteq N$ and for all elements $e \in N \setminus B$,

$$f(A + e) - f(A) \geq f(B + e) - f(B).$$

(4.2)

Similarly to concavity, submodularity is a property of diminishing marginal returns. In the second definition, we see that when we add an element $e$ to a smaller set $A$, a submodular function $f$ increases at by same or higher value than when the same element is added to a larger set $B$ (where $A \subseteq B$). In fact, when the submodular function $f$ depends only on the cardinality of the set $A$, the function $f$ must be a concave function of the number of elements in the set $A$.

Supermodular function are defined similarly to submodular functions. $f$ is called supermodular if $-f$ is submodular, i.e., $f$ satisfies equations (4.1) and (4.2) with the opposite inequality sign. $f$ is modular if $f$ is both submodular and supermodular, i.e., if $f$ satisfies (4.1) and (4.2) as equalities. Note that the relation between submodular and supermodular functions is similar to the relation between concave and convex functions (for example, in
A set function \( f \) on \( N \) is called non-decreasing if \( f(A) \leq f(B) \) whenever \( A \subseteq B \subseteq N \) and non-increasing if \( f(A) \geq f(B) \) whenever \( A \subseteq B \subseteq N \). We say that the function \( f \) is normalized if \( f(\emptyset) = 0 \).

**Polymatroids**

Let \( x_e \) be a decision variable associated with a ground set element \( e \in N \). For each subset \( S \subseteq N \), we define:

\[
x(S) := \sum_{e \in S} x_e.
\]

(4.3)

Using this notation, define a polyhedron \( P_f \) associated with a set function \( f \) on \( 2^N \):

\[
P_f = \{ x \in \mathbb{R}^N \mid x(S) \leq f(S) \ \forall S \subseteq N \text{ and } x \geq 0 \}.
\]

(4.4)

\( P_f \) is called the polymatroid associated with \( f \) if the following properties hold (from Yao and Zhang (1997)):

1. (normalized) \( f(\emptyset) = 0 \);
2. (non-decreasing) \( f(A) \leq f(B) \) whenever \( A \subseteq B \subseteq N \);
3. (submodular) \( f(A) + f(B) \geq f(A \cap B) + f(A \cup B) \), for all \( A, B \subseteq N \).

Observe that all polymatroids are bounded (since \( 0 \leq x_e \leq f(\{e\}) \) for each \( e \in N \)), and hence they are polytopes.

**Optimization over Polymatroids using the Greedy Algorithm**

Let \( f : 2^N \rightarrow \mathbb{R} \) be a normalized, non-decreasing, and submodular function, given by a value giving oracle, that is, by an oracle that returns \( f(S) \) for any \( S \subseteq N \). Given \( f \) and a weight function \( w : N \rightarrow \mathbb{R}_+ \), the goal is to solve the maximization problem

\[
\max_{x \in P_f} \sum_{e \in N} w(e)x_e.
\]

(4.5)
We denote this optimization problem as the Linear Optimization over Polymatroids (LPP). Despite the fact that polymatroids are defined using exponentially many constraints, LPPs can be optimized rather efficiently using a simple greedy algorithm (Yao and Zhang [1997]):

**Greedy optimization algorithm.**

1. Order the coefficients \( e_1, ..., e_{|N|} \in N \) such that \( w(e_1) \geq w(e_2) \geq ... \geq w(e_{|N|}) \).

2. For each \( e_i \in N \) let

\[
x_{e_i} = f \left( \{e_1, ..., e_{i-1}, e_i\} \right) - f \left( \{e_1, ..., e_{i-1}\} \right).
\]

(4.6)

Note that the greedy algorithm is a strongly polynomial-time algorithm since it requires a linear number of calls to the value oracle. Also observe, that when the function \( f \) is integer valued, the solution \( x \) given by Equation (4.6) is integral.

**Properties of LPPs**

Let \( \varphi(\bar{w}) \) denote the optimal value of an LPP with known objective coefficients vector \( \bar{w} \), that is:

\[
\varphi(\bar{w}) = \max_{x \in P} \bar{w}^T x.
\]

(4.7)

As shown in the lemma below, the function \( \varphi \) inherits properties from both linear programs, and the greedy algorithm.

**Lemma 4.1.1.** The function \( \varphi(\bar{w}) \) satisfies the following properties:

1. The function \( \varphi(\bar{w}) \) is continuous, convex, and piece-wise linear,

2. The function \( \varphi(\bar{w}) \) is linear in each \( w_t \in \bar{w} \) in any segment defined by two consecutive values in the monotonically ordered set of coefficients: \( \bar{w} \setminus \{w_t\} \).
Figure 4-1: Objective coefficients ordered from largest \((w_1)\) to smallest \((w_n)\).

3. The derivative of \(\varphi(\bar{w})\) with respect to \(w_t\) in any linear segment is constant and equals:

\[
\frac{\partial}{\partial w_t} \varphi(\bar{w}) = f(\{j : w_j > w_t\} \cup \{t\}) - f(\{j : w_j > w_t\}).
\]

Proof. (1) is a known result (Theorem 5.3 in Bertsimas and Tsitsiklis (1997)) for polytopes; (2) and (3) follow directly from Theorem 5.3 in Bertsimas and Tsitsiklis (1997), which states that the derivative with respect to an objective coefficient is equal to the value of the respective decision variable (given by the greedy algorithm).

Figure 4-1 illustrates the objective coefficients of an LPP that are ordered from largest \((w_1)\) to smallest \((w_n)\). This order induces a certain optimal solution vector \(\bar{x}\). Consider an objective coefficient \(w_t\) that satisfies \(w_{t-1} < w_t < w_{t+1}\). If we change the value of \(w_t\) within the range \((w_{t-1}, w_{t+1})\), the optimal solution vector does not change (due to the greedy algorithm). This implies that within this range the objective value \(\varphi(\bar{w})\) is equal to \(\varphi(\bar{w}) = \bar{w}^T \bar{x}\).

Therefore, the function \(\varphi\) is linear in \(w_t\), and its derivative with respect to \(w_t\) is equal to the respective decision variable \(x_t\) (given by the greedy algorithm). This holds as long as the order of objective coefficients does not change. Therefore, the only values in which the function \(\varphi\) is non-smooth in \(w_t\) are the set of objective coefficients.

A Note on Contra-Polymatroids

Closely related to the polyhedral structure of polymatroid, is the Contra-Polymatroid. Whereas polymatroid is defined using a normalized, non-decreasing, and submodular function, the Contra-Polymatroid is defined using a normalized, non-decreasing, and supermodular func-
\[ P_f^c = \left\{ x \in \mathbb{R}^N \mid x(S) \geq f(S) \ \forall S \subseteq N \text{ and } x \geq 0 \right\}, \] (4.8)

where the inequality that defines the set of constraints is in a reverse order compared with the definition of polymatroid.

Similarly to maximization of linear objectives over a polymatroid, there exists a greedy algorithm that efficiently solves linear minimization objectives over contra-polymatroids:

\[ \min_{x \in P_f^c} \sum_{e \in N} w(e)x_e. \] (4.9)

The relation between the polymatroid maximization (LPP), and contra-polymatroid minimization is close to the relation between minimization of convex functions and the maximization of concave functions. Due to this equivalence and to avoid redundancy, we focus on polymatroid maximization (while all results hold for contra-polymatroid minimization as well).

Next, we provide examples of combinatorial optimization problems that can be written as LPPs.

### 4.2 Examples of LPPs

Linear programs over polymatroids capture many core problems in graph theory, linear algebra, and other branches of mathematics and computer science. We now review several representative examples.

#### 4.2.1 K-Selection

Suppose a manager needs to select (exactly) one out of several competing projects. Using graph terminology, these projects can be modeled as parallel edges between two vertices, with the edge weights representing revenues. Different levels of uncertainty regarding the revenue of each project are modeled as different probability distributions, possibly estimated based
on past experience or preliminary examination. To reduce this uncertainty the manager can “test” any alternative, for example by performing market research, to obtain the realization of its revenue, and such a test incurs a fixed cost. The goal of the manager is to maximize the expected profit, where profit in this example is the revenue of the selected project minus all the testing costs. If the testing cost is high, then the manager might not want to test many projects (and if it is extremely high he may not wish to test at all). So the first decision the manager must make is whether to perform any test at all, or to select a project based on the current information. If he decides that testing is worthwhile, the manager must also choose which project to test. Then, after observing the realization of the tested project, the manager must make another decision based on her updated information – whether to test another project, and if so which one, and so forth. These decisions define an adaptive policy the manager follows, and our goal is to find such a policy with optimal expected profit.

Without testing, the more general problem of selecting the best K of N projects can be written as the following linear program:

$$\begin{align*}
\max & \quad w^T x \\
\text{s.t.} & \quad \sum_{i=1}^{N} x_i \leq K, \\
& \quad x_i \leq 1, \\
& \quad x_i \geq 0
\end{align*}$$

(4.10)

where $w_t$ denotes the revenue from project $t$, and $x_t$ relates to the decision to choose project $t$. The problem can be also written as an LPP using the submodular function:

$$f(S) = \begin{cases} 
0 & S = \emptyset \\
\min(K, |S|) & \text{otherwise.}
\end{cases}$$

(4.11)

To see this, we divide the constraints based on the cardinality of the set $S$:

1. $|S| = 1$: $S = \{x_i\}$ and the resulting constraints are $x_i \leq 1$,
2. \( |S| = N \): \( S = \{1\ldots N\} \) and the resulting constraint is \( \sum_{i=1}^{N} x_i \leq K \),

3. \( |S| \leq K \): the resulting constraint can be written as \( \sum_{i \in S} x_i \leq |S| \), which is redundant by (1),

4. \( |S| > K \): the resulting constraint can be written as \( \sum_{i \in S} x_i \leq K \) and is redundant by (2).

### 4.2.2 Scheduling

In chapter 3, we studied the problem of scheduling \( N \) jobs for processing with the objective of minimizing the sum of completion times. This problem can formulated as the following linear program:

\[
\begin{align*}
& \text{min} & & \sum_{i=1}^{N} t_i + \sum_{i=1}^{N} \sum_{j > i} (x_{ij} t_i + x_{ji} t_j) \\
& \text{s.t.:} & & x_{ij} + x_{ji} = 1 \quad \forall i, j \in \{1..N\} \\
& & & x_{ij} \geq 0 \quad \forall i, j \in \{1..N\},
\end{align*}
\]

where \( x_{ij} \) denotes the relative order between jobs \( i \) and \( j \) (that is, \( x_{ij} = 1 \) if job \( i \) is processed before job \( j \), and 0 otherwise).

It is easy to see that the following formulation obtains the same optimal solution:

\[
\begin{align*}
& \text{min} & & \sum_{i=1}^{N} \sum_{j > i} (x_{ij} t_i + x_{ji} t_j) \\
& \text{s.t.:} & & x_{ij} + x_{ji} \geq 1 \quad \forall i, j \in \{1..N\} \\
& & & x_{ij} \geq 0 \quad \forall i, j \in \{1..N\},
\end{align*}
\]

where a constant term was removed from the objective, and a constraint was altered from equality to inequality (which does not change the optimal solution due to the minimization, that is, for every solution with a strict inequality, we can generate a dominating solution where all constraints are satisfied with equality).

However, the same polyhedron can be written as a contra-polymatroid with the following
supermodular function $f$:

$$f(U) = \sum_{i<j} I_{\{x_{ij}, x_{ji}\} \in U}. \tag{4.14}$$

To see this, observe that for any set $U = \{(i, j), (j, i)\}$, the resulting constraint is: $x_{ij} + x_{ji} \geq 1$. Moreover, all other constraints are redundant:

- If $(i, j) \in U$ and $(j, i) \notin U$, then the constraint $X(U) \geq f(U)$ is redundant due to the constraint generated by $X(U \setminus \{(i, j)\}) \geq f(U \setminus \{(i, j)\})$. For example, $x_{12} + x_{21} + x_{34} \geq 1$ is redundant due to the constraint $x_{12} + x_{21} \geq 1$, and

- Any constraint generated by a set $U$ that includes both ordering of any pair of jobs (that is $(i, j) \in U \rightarrow (j, i) \in U$), is redundant due to the constraints generated by each of the pairs. For example, the constraint $x_{12} + x_{21} + x_{34} + x_{43} \geq 2$ (which includes both ordering of the pairs $(1, 2)$ and $(3, 4)$) is redundant due to the constraints $x_{12} + x_{21} \geq 1$ and $x_{34} + x_{43} \geq 1$.

Note that the testing model studied in this chapter does not precisely cover the above model where a single objective coefficient (e.g., processing time $t_i$) is associated with multiple decision variables (e.g., $x_{i,1}$ and $x_{i,2}$).

### 4.2.3 Matroid Optimization

A matroid is a combinatorial structure that captures and generalizes the notion of independent sets in vector spaces (Wikipedia (2016)). This structure is of particular interest since many subset selection problems can be formulated using matroids (either directly or using composition), and due to desirable algorithmic properties associated with solving the resulting optimization problems.

Formally, a matroid is a pair $(\mathcal{S}, \mathcal{I})$, where $\mathcal{S}$ is a finite set of elements (called the ground set), and $\mathcal{I}$ is a non-empty collection of subsets of $\mathcal{S}$ (the collection of independent sets), satisfying the following properties (Schrijver (2003)):

1. if $I \in \mathcal{I}$, and $J \subset I$, then $J \in \mathcal{I}$ (closure to taking subsets),

108
2. if $I, J \in \mathcal{I}$ and $|I| < |J|$, then $I + z \in \mathcal{I}$ for some $z \in J \setminus I$ (all inclusion-wise maximal independent sets have the same size).

We define the rank function of a set $U \subset \mathcal{S}$ to be the number of elements in the largest independent subset of $U$:

$$r(U) := \max\{|Z| : Z \in \mathcal{I}, Z \subset U\}.$$ 

Edmonds (1970) showed that given a matroid and a weight function on the ground set $W : \mathcal{S} \to \mathbb{R}^+$, finding an independent set whose total weight is maximal can be formulated as an LPP with the submodular function $f$ being the rank function $r$:

$$\begin{align*}
\max & \sum_{e \in \mathcal{S}} w(e)x_e \\
\text{s.t.} & \quad x(U) \leq r(U) \quad \forall U \subset \mathcal{S} \\
\quad & \quad x_e \geq 0.
\end{align*}$$

(4.15)

In this formulation, $x_e$ represents the decision to include edge $e \in E$ in the resulting set. Note that the rank of any independent set $\{e\}$ is equal to 1, and therefore $x_e \leq 1$. Moreover, the integrality of the submodular function defining the polymatroid (i.e., the rank function) implies that the every decision variable $x_e$ is integer (and in fact, binary).

### 4.2.4 Graphic Matroids and the Maximum Spanning Tree (MST)

One particularly important class of matroids is the Graphic Matroid (see Schrijver (2003)). Let $G = (V, E)$ be a connected graph with vertices $V = \{v_1, \ldots, v_n\}$ and edges $E = \{e_1, \ldots, e_m\}$. For any subset $E'$ of edges, denote by $V(E') \subset V$ the subset of vertices covered by $E'$:

$$V(E') = \{v_1 \in V : \exists v_2 : (v_1, v_2) \in E'\}.$$ 

In the graphic matroid $(\mathcal{S}, \mathcal{I})$ defined over $G$, the ground set $\mathcal{S}$ contains all edges (that is, $\mathcal{S} = E$), and the collection of independent sets $\mathcal{I}$ contains the subsets of $E$ that form a forest.
(i.e., there are no cycles in $E$). The rank function $r$ for the graphic matroid can then be written as follows. For each subset $E'$ of $E$, let $\kappa(V, E')$ denote the number of components in the graph $G(V, E')$. Then, for each $E' \subset E$:

$$r(E') = |V| - \kappa(V, E').$$

An interesting interpretation of the resulting LPP is that for any subset of edges $E' \subset E$ that connects a single component there is a constraint associated with $E'$ that restricts the number of selected elements in $E'$ to be at most $|V(E')| - 1$:

$$x(E') \leq |V| - \kappa(V, E') = |V(E')| - 1.$$  

Moreover, all of the constraints of the LPP associated with edges $E'$ that connect multiple components are redundant and can be expressed using constraints about edges that connect a single component. Figure 4-2 illustrates a subset of two edges $e_1, e_2$, in a graphic matroid that includes 6 vertices. The rank function associated with the sets $\{e_1\}, \{e_2\}$, and $\{e_1, e_2\}$ is:

$$r(\{e_1\}) = 6 - 5 = 1, r(\{e_2\}) = 6 - 5 = 1, r(\{e_1, e_2\}) = 6 - 4 = 2.$$  

The resulting constraints are:

$$x_1 \leq 1, x_2 \leq 1, x_1 + x_2 \leq 2.$$  

The edges in of the set $\{e_1, e_2\}$ connect two components and the constraint associated with the set $\{e_1, e_2\}$ is indeed redundant.

Given edge weights $w_1, \ldots, w_m$, the problem of finding a spanning tree of the graph $G$ (i.e., a subset of edges that connect every vertex in the graph and does not contain a cycle) with
maximal total weight is therefore a matroid optimization problem and can be solved using
the following LPP:
\[
\begin{align*}
\text{max} & \quad \sum_{e \in E} w(e)x_e \\
\text{s.t.} & \quad x(E') \leq r(E') \quad \forall E' \subset E, \\
& \quad x_e \geq 0
\end{align*}
\] (4.16)
where \(x_e\) represents the decision to include edge \(e \in E\) in the maximum spanning tree.

Note that the Selection problem of a single element can be viewed as a special case of
the Maximum Spanning Tree problem, where \(G\) has two vertices and parallel edges between
them.

4.2.5 Conservations Laws

In addition to subset selection problems, polymatroids play an important role in stochastic
optimization. Shanthikumar and Yao (1992) showed that many queueing control problems
can be formulated as polymatroid optimization problems. Specifically, if one can define a
performance vector \(\bar{x}_\pi\) (which depends on the policy \(\pi\)) such that:

1. the objective of the queueing control problem can be written as a linear function of
   performance vectors, and

2. performance vectors satisfy certain properties called conservation laws,

then the space of all performance vectors

\[ P = \{\bar{x}_\pi : \text{where } \pi \text{ is an admissible policy}\}, \]

is a polymatroid. (More precisely, it is the intersection of a polymatroid and a half space,
for which the greedy algorithm solution is also optimal.) For these queueing problems, the
optimal objective value can be found by solving a polymatroid optimization problem.

As an example, Shanthikumar and Yao (1992) consider a single preemptive server (i.e.,
processing of a job could be paused while the server switches to another job) with \(J\) classes
of arriving jobs. Jobs are independent, and every job class $i$ is associated with different distributions for the service and inter-arrival times, and a waiting cost $c_i$ defined per unit of time. They look at the problem of prioritizing the processing of jobs so as to minimize the long-run average waiting costs. Formally:

- $\mu_i$ is the service rate of jobs of class $i$,
- $c_i$ is the cost per unit of time for having a job of class $i$ in the system,
- $E[N_i]$ is the expected number of jobs of class $i$ in the system, and
- the objective is to minimize $\sum_i c_i E[N_i] = \sum_i c_i \mu_i (E[N_i]/\mu_i)$.

For this problem, they show that the performance vector $\bar{x}$ where $x_i = E[N_i]/\mu_i$ satisfies conservation laws. Therefore, the optimal objective value can be formulated as a polymatroid optimization problem. In addition, they provide an intuitive interpretation for the optimal policy in the form of the well known $c\mu$ rule (which corresponds to the value of the objective coefficients in the polymatroid optimization problem).

### 4.3 The LPPT Problem Formulation

Linear Programs over Polymatroids with Testing (LPPTs) are LPPs, in which the objective coefficients $\bar{W}$ are independent random variables from known distributions. Without testing, the vector $x$ that maximizes the expected objective value is the same vector $x$ that solves the LPP when the objective coefficients are $E[\bar{W}]$:

$$\max_x E[\bar{W}^T x] = \max_x E[\bar{W}]^T x. \quad (4.17)$$

This is a direct consequence of the linearity of the expectation operator.

In LPPT we assume that one could test the coefficients prior to optimizing. Testing an objective coefficient $W_i$ reveals the exact realization of that coefficient $w_i$, but incurs a cost of $c$. After testing, we can either test one of the untested (or unknown) coefficients, or
we could stop testing and simply optimize by taking the expected values of the objective coefficients. The goal is to develop an adaptive policy that maximizes the expected objective value of the optimization problem minus the testing costs. A policy for the LPPT problem decides adaptively on whether to continue testing or solve the respective polymatroid with respect to the expected values of the objective coefficients. If testing continues, the policy chooses the next objective coefficient to be tested. This decision could depend on the values of already tested coefficients. To put things in context of the examples of Section 4.2 in the K-Selection problem, the objective coefficients \( \bar{W} \) represent the unknown valuation of every alternative. In the MST problem, the objective coefficients represent the unknown random weights of the edges.

The system state of LPPT can be described as a tuple \((\bar{W}, \bar{w})\), where \( \bar{W} \) denotes the vector of unknown coefficients, and \( \bar{w} \) denotes the vector of tested (known) coefficients. We assume \( \bar{W} \) to be a vector of independent discrete random variables from known distributions.

After testing coefficient \( W_i \in \bar{W} \), we transition to state \((\bar{W} - W_i, \bar{w} + W_i)\), where we use '-' and '+' to denote exclusion and inclusion of elements from a set \( (\bar{W} - W_i \equiv \bar{W} \setminus \{W_i\} \) \) and \( \bar{w} + W_i \equiv \bar{w} \cup \{W_i\} \). With a slight abuse of notation, we use \( \varphi(\mathbb{E}[\bar{W}], \bar{w}) \) to denote the expected value of the optimal solution to the LPP without future testing. This is based on the observation of Equation \( (4.17) \), in which the optimal solution to the LPP with random objective coefficients can be obtained from the LPP by replacing the untested coefficients with their expected values \( \mathbb{E}[\bar{W}] \). The Dynamic Programming formulation of the LPPT problem can be then written as:

\[
J^{\text{opt}} (\bar{W}, \bar{w}) = \max \begin{cases} 
\varphi(\mathbb{E}[\bar{W}], \bar{w}) & \text{stop} \\
-c + \mathbb{E}_{W_i} \left[ J^{\text{opt}} (\bar{W} - W_i, \bar{w} + W_i) \right] & \text{test}_i \text{ for all } i 
\end{cases}
\]  

(4.18)

In this formulation, \( J^{\text{opt}} \) denotes the value function of the optimal policy at state \((\bar{W}, \bar{w})\). Observe that any state in this formulation corresponds to a product of a subset of unknown coefficients, times all possible realizations of the tested coefficients. To get a sense of the complexity of the state space, consider the case where the unknown coefficients are drawn
from a single distribution with a support of size \( s \). There are \( s^N \) different states in which all coefficients are known (in addition to the many other states where only subsets of coefficients are known). Therefore, we cannot hope to solve this problem to optimality in an efficient way without characterizing the optimal policy first.

To motivate why testing could be valuable, observe that \( \varphi(\mathbb{E}[\bar{W}], \bar{w}) \) computes a solution that is optimal with respect to the mean of the unknown coefficients. However, this solution is not necessarily optimal with respect to each realization, and revealing the exact values through testing could therefore increase the total expected objective value function. In the extreme case, we can obtain the optimal solution for any specific objective vector realization by testing all coefficients, however, this might be too costly because of the testing costs. The goal is to find a policy that optimally balances the benefit from reducing the uncertainty associated with the objective vector and the cost incurred by testing.

One can interpret the dynamic programming formulation of the LPPT as a composition of LPPs. As a consequence, some of the properties of LPPs carry through to LPPTs as given by the following lemma.

**Lemma 4.3.1.** The value function \( J^{opt}(\bar{W}, \bar{w}) \) is continuous, convex, and piece-wise linear for every \( w_t \in \bar{w} \).

**Proof.** Proceed by induction on \( N \). When \( N = 0 \), then \( \bar{W} = \emptyset \) and the value function of \( \Pi^{opt} \) is

\[
J^{opt}(\emptyset, \bar{w}) = \varphi(\emptyset, \bar{w})
\]

which is continuous, convex, and piece-wise linear in \( w_t \in \bar{w} \) (Lemma 4.1.1).

For \( N \geq 1 \), the value function of \( \Pi^{opt} \) is

\[
J^{opt}(\bar{W}, \bar{w}) = \max \begin{cases}
\varphi(\mathbb{E}[\bar{W}], \bar{w}) & \text{stop} \\
-c + \mathbb{E}_i \left[ J^{opt}(\bar{W} - W_i, \bar{w} + W_i) \right] & \text{test}_i \text{ for all } i
\end{cases}
\]

From the induction hypothesis, \( J^{opt}(\bar{W} - W_i, \bar{w} + W_i) \) is continuous, convex, and piece-wise linear in \( w_t \) for any random coefficient \( W_i \in \bar{W} \) and any realization of \( W_i \). Since summation
and the maximum operator preserve continuity, convexity, and the piece-wise linearity, the function $J_{\text{opt}}(\bar{W}, \bar{w})$ is continuous, convex, and piece-wise linear for every $w_i \in \bar{w}$.

Myopic Policies

We define a few terms that will be useful in the discussion of myopic policies:

**Definition 4.3.2.** (Stop Policy) $\Pi_{\text{stop}}$ is the policy that at any state $(\bar{W}, \bar{w})$ stops testing and solves the optimization problem using the expected values of the untested parameters. The value function $J_{\text{stop}}$ of policy $\Pi_{\text{stop}}$ satisfies:

$$J_{\text{stop}}(\bar{W}, \bar{w}) = \varphi(\mathbb{E}[\bar{W}], \bar{w}).$$  \hspace{1cm} (4.19)

**Definition 4.3.3.** (Test $i$ Policy) $\Pi_{i\text{test}}$ is the policy that any state $(\bar{W}, \bar{w})$ tests parameter $W_i \in \bar{W}$ and continues according to the optimal policy $\Pi_{\text{opt}}$. The value function $J_{i\text{test}}$ of policy $\Pi_{i\text{test}}$ satisfies:

$$J_{i\text{test}}(\bar{W}, \bar{w}) = -c + \mathbb{E}_{W_i}[J_{\text{opt}}(\bar{W} - W_i, \bar{w} + W_i)].$$  \hspace{1cm} (4.20)

**Definition 4.3.4.** $\Pi_{i\text{my}}$ is the policy that at any state $(\bar{W}, \bar{w})$ tests parameter $W_i \in \bar{W}$ and stops. The value function $J_{i\text{my}}$ of policy $\Pi_{i\text{my}}$ satisfies:

$$J_{i\text{my}}(\bar{W}, \bar{w}) = -c + \mathbb{E}_{W_i}[\varphi(\mathbb{E}[\bar{W} - W_i], \bar{w} + W_i)].$$  \hspace{1cm} (4.21)

**Definition 4.3.5.** (Myopic Gain) For state $(\bar{W}, \bar{w})$ and untested parameter $W_i \in \bar{W}$ denote by $\Delta_i(\bar{W}, \bar{w})$ the myopic gain from testing $W_i$,

$$\Delta_i(\bar{W}, \bar{w}) = J_{i\text{my}}(\bar{W}, \bar{w}) - J_{\text{stop}}(\bar{W}, \bar{w}).$$  \hspace{1cm} (4.22)

**Remark 4.3.6.** Since policy $\Pi_{i\text{my}}(\bar{W}, \bar{w})$ observes only one more realization and the stopping policy obtains no new information, $\Delta_i$ is sensitive only to $W_i$, and treats all other untested coefficients as their expectations. Hence, $\Delta_i(\bar{W}, \bar{w}) = \Delta_i(\bar{W} - W_i, \bar{w} + \mathbb{E}[W_i])$ for all $W_i \in$
\{\bar{W} - W_i\}.

Using the above notation, we can now introduce myopic polices.

**Definition 4.3.7. (Myopic Stopping Rule)** We say that policy \(\pi\) adheres to the Myopic Stopping Rule if at every state \((\bar{W}, \bar{w})\), policy \(\pi\) stops, if and only if, all myopic gains are non-positive, that is:

\[ \forall W_i \in \bar{W} : \Delta_i(\bar{W}, \bar{w}) \leq 0. \]

**Definition 4.3.8. (Myopic Testing Rule)** We say that a policy \(\pi\) adheres to the Myopic Testing Rule if at every state \((\bar{W}, \bar{w})\), the policy may only test coefficients whose myopic gains are maximal. That is, if at state \((\bar{W}, \bar{w})\) policy \(\pi\) tests \(W_i \in \bar{W}\), then

\[ \forall W_j \in \bar{W} : \Delta_j(\bar{W}, \bar{w}) \leq \Delta_i(\bar{W}, \bar{w}). \]

**Definition 4.3.9. (Myopic Policy)** Any policy \(\pi\) that adheres to the myopic stopping and testing rules, is said to be a myopic policy.

### 4.4 An Optimal Stopping Rule for LPPT with Coefficients with Equal Means

We now prove that when the unknown coefficients in the LPPT are discrete random variables that have equal means, the myopic stopping rule (Definition [13.7]) is optimal in determining when to stop testing. We start by proving a certain monotonicity property, which we then use to show that the myopic stopping rule is indeed optimal.

**Lemma 4.4.1.** At each state \((\bar{W}, \bar{w})\) where \(W_i \in \bar{W}, w_t \in \bar{w}\), and for all \(j \in \bar{W}: \mathbb{E}[W_j] = \mu\), the function \(\Delta_i(\bar{W}, \bar{w})\) is unimodal in \(w_t\) and maximized at \(w_t = \mu\).

**Proof.** To prove this result, we first show that the function \(\Delta_i\) is continuous and piece-wise linear in \(w_t \in \bar{w}\). We then show that in every linear segment where \(w_t < \mu\), the derivative of \(\Delta_i\) with respect to \(w_t\) is non-negative, and that in every linear segment where \(w_t > \mu\), the
The derivative is non-positive. The continuity of $\Delta_i$ would then imply that $\Delta_i$ is unimodal in $w_t$ and achieves its maximal value at $w_t = \mu$.

By definition (Equation (4.22)) the function $\Delta_i (\bar{W}, \bar{w})$ can be written as follows:

$$\Delta_i (\bar{W}, \bar{w}) = J_{i}^{my} (\bar{W}, \bar{w}) - J_{i}^{stop} (\bar{W}, \bar{w}) = -c + \mathbb{E}_{W_i} [\varphi (\bar{W} - W_i, \bar{w} + W_i)] - \varphi (\bar{W}, \bar{w})$$

$$= -c + \sum_{i} \text{Prob}(W_i = w_i) \varphi (W - W_i, \bar{w} + w_i) - \varphi (W - W_i, \bar{w} + \mu). \quad (4.23)$$

In this expression, $w_i$ denotes specific realizations of the random coefficient $W_i$. In the last equality we use the fact that under the function $\varphi$, random coefficients are replaced by their expected value (which is why we replaced $\varphi (\bar{W}, \bar{w})$ with $\varphi (\bar{W} - W_i, \bar{w} + \mu)$).

In Equation (4.23), we see that the function $\Delta_i$ is constituted from the function $\varphi$ applied to different states that share the same unknown and known coefficients, with the exception of having different values for the known parameter $w_i$. From Lemma 4.1.1 we know that the function $\varphi$ is continuous in $w_t \in \bar{w}$ and therefore so is the function $\Delta_i$. Moreover, Lemma 4.1.1 asserts that the function $\varphi$ is piece-wise linear with the non-smooth points being the set of coefficients $\bar{w} \setminus \{w_t\} \cup \{\mu\}$. Therefore, the summation in Equation (4.23) is a piece-wise linear function of $w_t$ where the non-smooth points are the union of all coefficients values:

$$\mathcal{S} = \bar{w} \setminus \{w_t\} \cup \{\mu\} \cup \{w_i\}.$$  

This implies that the function $\Delta_i$ is not only continuous, but also piece-wise linear in $w_t$. Therefore, the function $\Delta_i$ is linear in every segment defined by two consecutive points in $\mathcal{S}$ (that is, between any two points $x, y \in \mathcal{S}$ such that $\not\exists z \in \mathcal{S}$ for which $x < z < y$).

We now take the derivative of $\Delta_i (\bar{W}, \bar{w})$ with respect to $w_t$ to show that the function $\Delta_i (\bar{W}, \bar{w})$ is non-decreasing in $w_t$ in segments where $w_t < \mu$, and that it is non-increasing in $w_t$ in segments where $w_t > \mu$ (by definition $\mu \in \mathcal{S}$ and therefore all segments are either
below or above $\mu$). The derivative of $\Delta_i$ with respect to $w_i$ can be written as follow:

\[
\frac{\partial}{\partial w_t} \Delta_i (W, \bar{w}) = \frac{\partial}{\partial w_t} \left( -c + \sum_i \text{Prob}(W_i = w_i) \varphi (W - W_i, \bar{w} + w_i) - \varphi (W - W_i, \bar{w} + \mu) \right) \\
= \sum_i \text{Prob}(W_i = w_i) \frac{\partial}{\partial w_t} \varphi (W - W_i, \bar{w} + w_i) - \frac{\partial}{\partial w_t} \varphi (W - W_i, \bar{w} + \mu) \\
= \sum_{i: w_i < w_t} \left( \text{Prob}(W_i = w_i) \frac{\partial}{\partial w_t} \varphi (W - W_i, \bar{w} + w_i) \right) \\
+ \sum_{i: w_i > w_t} \left( \text{Prob}(W_i = w_i) \frac{\partial}{\partial w_t} \varphi (W - W_i, \bar{w} + w_i) \right) \\
- \frac{\partial}{\partial w_t} \varphi (W - W_i, \bar{w} + \mu). \\
\tag{4.24}
\]

The second equality follows from the linearity of the derivative operator, and because $c$ is a constant. In the third equality we split the support of $W_i$ to values that are higher and lower than $w_i$. Since that the support of $W_i$ is included in $S$, there is no value $w_i$ where $w_i = w_t$.

Let $w_i^-$ be a realization of $W_i$ that satisfies $w_i^- < w_t$, and let $x_i^-$ denote the derivative of $\varphi (\bar{W} - W_i, \bar{w} + w_i^-)$ with respect to $w_i$:

\[
x_i^- = \frac{\partial}{\partial w_t} \varphi (\bar{W} - W_i, \bar{w} + w_i^-).
\]

From Lemma 4.1.1 (Property (3)), we know that $x^-$ is insensitive to the value of $w_i$ as long as it remains below $w_t$:

\[
x_i^- = \frac{\partial}{\partial w_t} \varphi (\bar{W} - W_i, \bar{w} + w_i^-), \text{ for all } w_i < w_t.
\]

Similarly, we denote by $x_i^+$ the derivative of $\varphi (\bar{W} - W_i, \bar{w} + w_i)$ with respect to $w_t$ for any realization $w_i > w_t$:

\[
x_i^+ = \frac{\partial}{\partial w_t} \varphi (\bar{W} - W_i, \bar{w} + w_i), \text{ for all } w_i > w_t.
\]

118
We can then write Equation \[4.24\] as follows:

\[
\frac{\partial}{\partial w_t} \Delta_i (\bar{W}, \bar{w}) = \sum_{i : w_i < w_t} \text{Prob}(W_i = w_i) x^-_i + \sum_{i : w_i > w_t} \text{Prob}(W_i = w_i) x^+_i - x^\text{stop}_i \\
= \sum_{i : w_i < w_t} \text{Prob}(W_i = w_i) (x^-_i - x^\text{stop}_i) + \sum_{i : w_i > w_t} \text{Prob}(W_i = w_i) (x^+_i - x^\text{stop}_i) \\
= \text{Prob}(W_i < w_t) (x^-_i - x^\text{stop}_i) + \text{Prob}(W_i > w_t) (x^+_i - x^\text{stop}_i). \tag{4.25}
\]

In this expression, \(x^\text{stop}_i\) denotes the derivative of \(\varphi(\bar{W} - W_i, \bar{w} + \mu)\) with respect to \(w_t\). Recall that we compute the derivative for each segment independently and that by construction \(w_t \neq w_i\) and \(w_t \neq \mu\). Moreover, in every segment \(x^-_i, x^+_i,\) and \(x^\text{stop}_i\) are constants.

Now that we have an expression for the derivative of \(\Delta_i (\bar{W}, \bar{w})\) with respect to \(w_t \in \bar{w}\), we show that it non-negative when \(w_t < \mu\), and that it is non-positive when \(w_t > \mu\).

Consider first the case where \(w_t > \mu\). From Lemma \[4.1.1\] the value of the derivative of \(\Delta_i (\bar{W} - W_i, \bar{w} + w_i)\) with respect to \(w_t \in \bar{w}\) is the same for all realizations \(w_i\) of \(W_i\) that are smaller than \(w_t\), including \(w_t = \mu\), which is why \(x^\text{stop}_i = x^-_i\). We can then write Equation \[4.25\] as follows:

\[
\frac{\partial}{\partial w_t} \Delta_i (\bar{W}, \bar{w}) = \text{Prob}(W_i < w_t) (x^-_i - x^\text{stop}_i) + \text{Prob}(W_i > w_t) (x^+_i - x^\text{stop}_i) \\
= \text{Prob}(W_i < w_t) (0) + \text{Prob}(W_i > w_t) x^+_i - \text{Prob}(W_i > w_t) x^\text{stop}_i \\
= \text{Prob}(W_i > w_t) (f(\{j : w_j > w_t\} \cup \{t, i\}) - f(\{j : w_j > w_t\} \cup \{i\})) \\
- \text{Prob}(W_i > w_t) (f(\{j : w_j > w_t\} \cup \{t\}) - f(\{j : w_j > w_t\})) \\
\leq 0.
\]

The third equality follows from Lemma \[4.1.1\] and the fact that \(w_t > \mu\), and because \(x^+_i\) corresponds to realizations of \(W_i\) that are higher than \(w_t\). The inequality results from the submodularity of the function \(f\).

Similarly, when \(w_t < \mu\), the derivative of \(\Delta_i (\bar{W} - W_i, \bar{w} + w_i)\) with respect to \(w_t\) is equal for all the realizations of \(W_i\) that are larger than \(w_t\) including \(\mu\). This implies that \(x^\text{stop}_i = x^+_i\),
and Equation (4.25) can be written as follows:

\[
\frac{\partial}{\partial w_t} \Delta_i(W, \bar{w}) = \text{Prob}(W_i < w_t) (x_i^- - x_i^{stop}) + \text{Prob}(W_i > w_t) (x_i^+ - x_i^{stop})
\]

\[
= \text{Prob}(W_i < w_t)x_i^- - \text{Prob}(W_i < w_t)x_i^{stop} + 0
\]

\[
= \text{Prob}(W_i < w_t) \left( f(\{j : w_j > w_t\} \cup \bar{W} \cup \{t}\} - f(\{j : w_j > w_t\} \cup \bar{W}) \right)
\]

\[
- \text{Prob}(W_i < w_t) \left( f(\{j : w_j > w_t\} \cup \bar{W} \cup \{t, i\}) - f(\{j : w_j > w_t\} \cup \bar{W} \cup \{i\}) \right)
\]

\[
\geq 0,
\]

which holds for similar reasons.

The function \(\Delta_i\) is decreasing when \(w_t > \mu\), and increasing when \(w_t < \mu\), and is therefore unimodal and obtains its maximal value where \(w_t = \mu\).

Intuitively, Lemma 4.4.1 implies that the myopic gain of testing is monotonically decreasing when we test, as coefficients drift from their expected value. We formalize this in the following corollary.

**Corollary 4.4.2.** At each state \((\bar{W}, \bar{w})\) and for each untested parameter \(W_j \in \bar{W}\), the myopic gain \(\Delta_j(\bar{W}, \bar{w})\) of an LPPT with equal mean \(\mu\) does not increase by testing another untested coefficient \(W_i \in \bar{W} - W_j\), that is:

\[
\Delta_j(\bar{W}, \bar{w}) \geq \Delta_j(\bar{W} - W_i, \bar{w} + w_i), \text{ for every coefficient } W_i \in \bar{W} - W_j, \text{ and realization } w_i.
\]

**Proof.** Immediate from Lemma 4.4.1

Corollary 4.4.2 implies that once all myopic gains are non-positive, they will remain non-positive in future steps. We use this fact to prove that the myopic stopping rule is optimal.

**Theorem 4.4.3.** The myopic stopping rule is optimal for LPPTs with untested coefficients that have equal means.
Proof. One direction is straightforward since it is readily verified that if for some untested parameter \( W_i \in \bar{W} \), \( \Delta_i (\bar{W}, \bar{w}) > 0 \) then stopping is not optimal since a single test of coefficient \( W_i \) outperforms stopping.

We prove the other direction by induction on the number of untested coefficients \( k \). When \( k = 1 \) the myopic stopping rule is optimal by definition. We prove the step \( k > 1 \) by contradiction. Suppose that at state \((\bar{W}, \bar{w})\) the myopic gains are non-positive (\( \forall W_i \in \bar{W} : \Delta_i (\bar{W}, \bar{w}) \leq 0 \)) and that the optimal policy \( \pi \) tests the unknown coefficient \( W_j \). Using Corollary 4.4.2 in the next state \((\bar{W} - W_j, \bar{w} + W_j)\) the myopic gains remain non-positive regardless of the realization of \( W_j \), that is:

\[
\forall W_i \in \bar{W} - W_j : \Delta_i (\bar{W} - W_j, \bar{w} + W_j) \leq 0.
\]

Therefore, the induction hypothesis implies that it is optimal to stop in state \((\bar{W} - W_j, \bar{w} + W_j)\). Consequently, policy \( \pi \) tests exactly once, and the following holds:

\[
J^\pi (\bar{W}, \bar{w}) = J^{MY}_j (\bar{W}, \bar{w}) \leq J^{STOP} (\bar{W}, \bar{w}).
\]

The equality holds because policy \( \pi \) tests exactly once, and the inequality holds because it is equivalent to \( \Delta_j (\bar{W}, \bar{w}) \leq 0 \). This is a contradiction to the suboptimality of stopping in state \((\bar{W}, \bar{w})\).

\[ \square \]

**Corollary 4.4.4.** For an LPPT with untested coefficients that have the same mean, if stopping at state \((\bar{W} + W_i, \bar{w})\) is optimal, then stopping is also optimal at state \((\bar{W}, \bar{w} + v)\), for any realization \( v \) of \( W_i \).

Proof. Immediate from Corollary 4.4.2 and Theorem 4.4.3

\[ \square \]

An interesting consequence of Lemma 4.4.1 is the following. Suppose that the myopic gain for testing coefficient \( i \) is positive. If we test coefficient \( j \) and the realization of \( W_j \) happens to be close to \( \mu \), then the myopic gain of testing coefficient \( i \) remains positive. That is, we can think about untested coefficients as being a-priori at the mean value of their
respective distributions; once we test, the coefficients drift away from the mean values. The closer the realizations are to the mean, the more likely we are to test again. On the other hand, if the realizations are farther away from the mean, then we are less likely to test again. That is, for every state, there exists an interval around the mean value where testing is optimal in the next step, as given by the following corollary.

**Corollary 4.4.5.** For an LPPT with untested coefficients that have the same mean, for each state \((\bar{W}, \bar{w} + v)\), it is either optimal to stop regardless of the value \(v\), or there exists \(v_1, v_2\), such that it is optimal to test if and only if \(v_1 \leq v \leq v_2\).

*Proof.* From Lemma 4.4.1 we know that the function \(\Delta_i (\bar{W}, \bar{w} + v)\) is unimodal in \(v\) and maximal in \(v = \mu\). Thus if \(\Delta_i (\bar{W}, \bar{w} + \mu) \leq 0\) for each \(i\), then by Theorem 4.4.3 it is not optimal to test for any value of \(v\). If, on the other hand, there exists an unknown coefficient \(W_i\) such that \(\Delta_i (\bar{W}, \bar{w} + \mu) > 0\), then we define \(v_1\), and \(v_2\) as follows:

\[
v_1 = \inf \{ v : \exists j \text{ s.t. } \Delta_j (\bar{W}, \bar{w} + v) > 0 \},
\]

and,

\[
v_2 = \sup \{ v : \exists j \text{ s.t. } \Delta_j (\bar{W}, \bar{w} + v) > 0 \}.
\]

From Theorem 4.4.3 we know that it is optimal to test, if and only if, \(v_1 \leq v \leq v_2\). □

We have completely characterized the decision to stop testing for LPPTs with identical mean value of untested coefficients. In the next two sections, we consider which coefficient to test when stopping is not optimal.

### 4.5 An Optimal Myopic Policy for MST with Testing

We now focus on a special LPPT, Maximum Spanning Tree with Testing, or MST with Testing for short. The formulation of the MST problem as an LPP is described in Section 4.2. Theorem 4.4.3 implies that when the cost coefficients (edge weights in MST) have the same mean, the myopic stopping rule (Definition 4.3.7) is optimal. In this section, we show that
when the edge weights are also identically distributed, then a myopic policy that consists of a myopic stopping rule and a myopic testing rule (Definition 4.3.8), i.e., the order in which edges are tested is determined myopically, achieves optimal expected profit. It is well-known that a minimum spanning tree can be found by negating all edge weights and finding a maximum spanning tree, so we will only address the maximization problem.

Formally, let $G = (V, E)$ be a connected graph with vertices $V = \{v_1, ..., v_n\}$ and edges $E = \{e_1, ..., e_m\}$. The edge weights denoted $W_{e_i}$, or $W_i$ for short, are i.i.d., drawn from a known distribution $W_i \sim W$, with a finite mean $E[W_i] = \mu < \infty$. The cost of testing an edge is $c > 0$. The goal is to determine an optimal policy for finding a maximum profit spanning tree of the graph $G$, with profit being the weight of the selected spanning tree minus testing costs.

Remark 4.5.1 (Tie breaking). Throughout this section, we will use Kruskal’s algorithm (Cormen et al. (2001)) to construct MSTs. Kruskal’s algorithm orders the edges according to their weights in non-increasing order and adds them to the MST if they do not close a cycle with existing tree edges. We will assume that ties between edge weights are broken in a consistent manner, i.e., if there are several possible MSTs, the algorithm will always return the same MST for the same weights. Furthermore, for simplicity, we will assume that if an edge weight changes, and becomes tied with an existing edge weight, the tie will be broken by ordering the new value after the old value in Kruskal’s algorithm.

When a policy stops testing, it optimizes based on tested edge weights and expectations of untested edge weights $(\bar{w}, \mathbb{E}[\bar{W}])$. Thus, the MST is computed on a deterministic graph, i.e., a graph with deterministic edge weights. We start by considering a deterministic graph and define the concept of a substituting edge that will be useful for the subsequent proofs. Intuitively, the substituting edge is the best candidate for swapping with $e_i$ into (or out of) the tree after testing it.

Definition 4.5.2. Let $T$ be a maximum spanning tree constructed by Kruskal’s algorithm in a deterministic graph $H = (V, E)$. For each edge $e_i \in E$, the unique substituting edge $e_{\text{sub}(i)}$, and the unique cycle $\text{Cycle}_i$ that contains both $e_i$ and $e_{\text{sub}(i)}$, are defined as follows:
• If the edge $e_i \notin T$, then $\text{Cycle}_i$ is the unique cycle that is created by adding $e_i$ to the tree. The substituting edge $e_{\text{sub}(i)}$ is defined as the tree edge with smallest weight in $\text{Cycle}_i \setminus \{e_i\}$, i.e., is latest in the ordering of Kruskal’s algorithm, and

• If the edge $e_i \in T$, then consider all the non-tree edges that, with the tree, form a cycle that contains $e_i$. The substituting edge $e_{\text{sub}(i)}$ is the edge that has maximal weight among them, i.e., is first in the ordering of Kruskal’s algorithm. Denote by $\text{Cycle}_i$ the unique cycle that is created by adding $e_{\text{sub}(i)}$ to the tree.

We can now prove a lemma concerning the effect of changing one edge weight in a deterministic graph.

**Lemma 4.5.3.** Let $T$ with edges $e_1, e_2, \ldots, e_m$ be a maximum spanning tree constructed by Kruskal’s algorithm in a deterministic graph $H$. Consider changing the weight of edge $e_i$ from $w_i$ to $w_i'$ and let the resulting MST be $T'$. Then the tree $T'$ is one out of three possible trees, the same tree $T' = T$, or $T' = T - e_i + e_{\text{sub}(i)}$, or $T' = T + e_i - e_{\text{sub}(i)}$. Moreover, $e_{\text{sub}(i)}$ depends only on $H, T, e_i$ and weights $w_1, \ldots w_m$ (but not on $w_i'$).

**Proof.** We use the following well-known "Cycle Properties" of the MST:

1. For an edge $e_l \notin T$, and for each edge $e_k \in \text{Cycle}_l$, it holds that $w_k \geq w_l$.

2. For each cycle $C$ in $H$, if $e_l \in C$ and $w_k > w_l$ for every other edge $e_k \in C$, then edge $e_l \notin T$. Notice that by our tie breaking assumption, there is a unique such $e_l$ even if the former strong inequality is replaced by its weak version $w_k \geq w_l$.

Consider first the case $e_i \notin T$. If $w_i' \leq \min\{w_k : e_k \in \text{Cycle}_l\}$, then by the first cycle property also $e_i \notin T'$. Since $e_i \notin T$ and $e_i \notin T'$, then the only cycle $e_i$ can close with $T'$ is $\text{Cycle}_i$. Hence, the two executions of Kruskal’s algorithm make the same decisions (also) for all other edges and therefore $T' = T$. Otherwise, $w_i' > w_{\text{sub}(i)}$. Hence, $e_i$ is before $e_{\text{sub}(i)}$ in the ordering and Kruskal’s algorithm will place $e_i \in T'$ and $e_{\text{sub}(i)} \notin T'$. Notice that no other edge can be affected by this change because there can only be one cycle formed by adding $e_i$ to the tree $T$. Therefore $T' = T + e_i - e_{\text{sub}(i)}$. 

124
Now consider the case $e_i \in T$. By the first cycle property, $w_k \geq w_{\text{sub}(i)}$ for all $e_k \in \text{Cycle}_i$. If $w'_i > w_{\text{sub}(i)}$ their relative order in Kruskal’s algorithm remains the same, and by the second cycle property $e_{\text{sub}(i)} \notin T'$ and $e_i \in T'$. Because $e_{\text{sub}(i)}$ is the first in the ordering among all other non-tree edges that form a cycle that contained $e_i$ with $T$, there is no cycle containing $e_i$ in which relative orders switch, hence Kruskal’s algorithm will return the same tree $T' = T$. Otherwise, $w'_i \leq w_{\text{sub}(i)}$, the relative orders of $e_i$ and $e_{\text{sub}(i)}$ switch (and possibly relative orders of $e_i$ and other non-tree edges that form a cycle that contained $e_i$ with $T$ also switch). Recall that $e_{\text{sub}(i)}$ is ordered after all the (tree) edges in $\text{Cycle}_i \setminus \{e_i, e_{\text{sub}(i)}\}$, and before all (non-tree) edges that form a cycle that contained $e_i$ with $T$. Thus $e_{\text{sub}(i)}$ does not form a cycle with edges that are before it in the ordering and are in $T'$; so $e_{\text{sub}(i)} \notin T'$ and $e_i$ does form a cycle with tree edges since $\text{Cycle}_i \setminus \{e_i\} \subset T'$, so $e_i \notin T'$. Also, no other edge is affected by the change because every other non-tree edge that formed a cycle that contained $e_i$ with $T$, now forms a cycle that contains $e_{\text{sub}(i)}$ with $T'$ (by transitivity). Hence $T' = T - e_i + e_{\text{sub}(i)}$.

In conclusion, $T$ and $T'$ differ by at most one swap, and this swap can be of only $e_i$ and its substituting edge $e_{\text{sub}(i)}$. Furthermore, $e_{\text{sub}(i)}$ is determined by the structure of the tree and the weights of edges before the change, but not by $w'_i$.

Going forward, we return to the setting of MST with testing, in which some of the edge weights are random. Recall that we denote the random edge weights by $\bar{W}$ and the known edge weights by $\bar{w}$. Recall also Definitions 4.3.2-4.3.4 of policies $\Pi_\text{stop} (\bar{W}, \bar{w})$, $\Pi_\text{test} (\bar{W}, \bar{w})$, and $\Pi_\text{my} (\bar{W}, \bar{w})$. For each state $(\bar{W}, \bar{w})$, if a policy chooses to stop, it returns a MST of the graph $G$, denoted by $T_\text{stop} (\bar{W}, \bar{w})$, which is constructed by Kruskal’s algorithm using expectations to replace random variables. For an untested edge $e_i$, the substituting edge $e_{\text{sub}(i)} (\bar{W}, \bar{w})$ and $\text{Cycle}_i (\bar{W}, \bar{w})$ are defined similarly to $e_{\text{sub}(i)}$ and $\text{Cycle}_i$ in Definition 4.5.2 with regard to the MST $T_\text{stop} (\bar{W}, \bar{w})$. Finally, recall definition 4.3.5 of the myopic gain of testing edge $e_i$, denoted by $\Delta_i (\bar{W}, \bar{w})$. For simplicity, when the state is clear from the context, we omit it from the notation.

We can now state the main result of this section. We show that when edge weights are
i.i.d, a myopic policy for the MST with testing achieves the optimal expected profit.

**Theorem 4.5.4.** If all the edge weights $W_i$ are independent and identically distributed with mean $\mu$, then at each state $(\bar{W}, \bar{w})$ the following policy is optimal: test an edge with the highest positive myopic gain $\Delta_i(\bar{W}, \bar{w})$, and stop when all myopic gains are non-positive.

In Section 4.2.4, we formulated the MST with testing problem as an LPPT and, therefore, by Section 4.4, it is optimal to stop testing when all myopic gains are non-positive. It remains to show that a policy that tests in non-increasing order of myopic gains is optimal. We do so by showing that testing an edge can affect only edges that have the same myopic gain value. This implies that if a myopic gain value is positive at some state, it remains positive as long as no other edge with the same myopic value is tested. We then use this to show that testing by descending myopic gains is optimal. First, we prove a couple of useful technical results.

**Lemma 4.5.5.** At every state $(\bar{W}, \bar{w})$, testing a yet-untested edge $e_i$ can cause at most one difference between $T_{\text{stop}}(\bar{W}, \bar{w})$ and $T_{\text{stop}}(\bar{W} - W_i, \bar{w} + w_i)$, which is swapping between $e_i$ and its substituting edge $e_{\text{sub}(i)}(\bar{W}, \bar{w})$.

**Proof.** $T_{\text{stop}}(\bar{W}, \bar{w})$ and $T_{\text{stop}}(\bar{W} - W_i, \bar{w} + w_i)$ are constructed using the deterministic values $(E[\bar{W}], \bar{w})$ and $(E[\bar{W} - W_i], \bar{w} + w_i)$ respectively. The only difference between these two executions of Kruskal’s algorithm is the weight of $e_i$ that changes from $E[W_i]$ to $w_i$. Hence, by Lemma 4.5.3, the MST can change by at most one edge, and that change is exactly swapping between $e_i$ and $e_{\text{sub}(i)}(\bar{W}, \bar{w})$. $\square$

We can now use the previous lemma to simplify the expression for the myopic gain of edge $e_i$. For consistency we use the notation $E[W_{\text{sub}(i)}]$ to denote the weight of edge $e_{\text{sub}(i)}$, although it can potentially be a known value $w_{\text{sub}(i)}$ if the edge $e_{\text{sub}(i)}$ was already tested.

**Corollary 4.5.6.** At each state $(\bar{W}, \bar{w})$ and for each untested edge $e_i$:

1. If $e_i \notin T_{\text{stop}}(\bar{W}, \bar{w})$, then $E[W_{\text{sub}(i)}] \geq \mu$ and the myopic gain of $e_i$ is

$$\Delta_i(\bar{W}, \bar{w}) = E[(W_i - E[W_{\text{sub}(i)}])^+] - c. \quad (4.26)$$
2. If \( e_i \in T^{\text{stop}} (\bar{W}, \bar{w}) \), then \( \mathbb{E} [W_{\text{sub}(i)}] \leq \mu \) and the myopic gain of \( e_i \) is

\[
\Delta_i (\bar{W}, \bar{w}) = \mathbb{E} \left[ (\mathbb{E} [W_{\text{sub}(i)}] - W_i)^+ \right] - c. \tag{4.27}
\]

**Proof.** If \( e_i \notin T^{\text{stop}} (\bar{W}, \bar{w}) \), then \( e_{\text{sub}(i)} \in T^{\text{stop}} (\bar{W}, \bar{w}) \) by Definition 4.5.2. By the cycle properties \( \mathbb{E} [W_j] \geq \mathbb{E} [W_i] = \mu \) for all \( e_j \in \text{Cycle}_i \). In particular, also \( \mathbb{E} [W_{\text{sub}(i)}] \geq \mu \). The myopic gain will be, by Definition 4.3.5 and Lemma 4.5.5.

\[
\Delta_i (\bar{W}, \bar{w}) = J_i^{\text{my}} (\bar{W}, \bar{w}) - J^{\text{stop}} (\bar{W}, \bar{w})
\]

\[
= -c + \mathbb{E}_{W_i} \left[ \varphi \left( \mathbb{E} [W - W_i], \bar{w} + W_i \right) - \varphi \left( \mathbb{E} [W], \bar{w} \right) \right] + \mathbb{E}_{W_i} \left[ \varphi \left( \mathbb{E} [W_i], \bar{w} \right) | W_i > \mathbb{E} [W_{\text{sub}(i)}] \right] \mathbb{P} \left( W_i > \mathbb{E} [W_{\text{sub}(i)}] \right)
\]

\[
= \mathbb{E}_{W_i} \left[ W_i - \mathbb{E} [W_{\text{sub}(i)}] \right] \mathbb{P} \left( W_i > \mathbb{E} [W_{\text{sub}(i)}] \right) - c
\]

Similarly, if \( e_i \in T^{\text{stop}} (\bar{W}, \bar{w}) \), then \( e_{\text{sub}(i)} \notin T^{\text{stop}} (\bar{W}, \bar{w}) \) by Definition 4.5.2. By the cycle properties \( \mathbb{E} [W_{\text{sub}(i)}] \leq \mu \). The myopic gain will be, by Definition 4.3.5 and Lemma 4.5.5.

\[
\Delta_i (\bar{W}, \bar{w}) = J_i^{\text{my}} (\bar{W}, \bar{w}) - J^{\text{stop}} (\bar{W}, \bar{w})
\]

\[
= -c + \mathbb{E}_{W_i} \left[ \varphi \left( \mathbb{E} [W - W_i], \bar{w} + W_i \right) - \varphi \left( \mathbb{E} [W], \bar{w} \right) \right] + \mathbb{E}_{W_i} \left[ \varphi \left( \mathbb{E} [W_i], \bar{w} \right) | W_i < \mathbb{E} [W_{\text{sub}(i)}] \right] \mathbb{P} \left( W_i < \mathbb{E} [W_{\text{sub}(i)}] \right)
\]

\[
= \mathbb{E}_{W_i} \left[ W_i - \mathbb{E} [W_{\text{sub}(i)}] \right] \mathbb{P} \left( W_i < \mathbb{E} [W_{\text{sub}(i)}] \right) - c
\]

\[
\square
\]

We can now show the independence between testing edges with different myopic gains.
Lemma 4.5.7. Assume that at state \((\bar{W}, \bar{w})\) the two untested edges \(e_i, e_j\) have different myopic gain values \(\Delta_i (\bar{W}, \bar{w}) \neq \Delta_j (\bar{W}, \bar{w})\). Then for each realization \(w_j\) of \(W_j\), we have \(\Delta_i (\bar{W} - W_j, \bar{w} + w_j) = \Delta_i (\bar{W}, \bar{w})\). Similarly, for each realization \(w_i\) of \(W_i\), we have \(\Delta_j (\bar{W} - W_i, \bar{w} + w_i) = \Delta_j (\bar{W}, \bar{w})\).

*Proof.* Assume without loss of generality that at state \((\bar{W}, \bar{w})\) a policy tests \(e_j\) and reveals its realization \(w_j\). Assume towards contradiction that \(\Delta_i (\bar{W} - W_j, \bar{w} + w_j) \neq \Delta_i (\bar{W}, \bar{w})\).

Recall that by Corollary \textbf{4.5.6} if \(e_i \notin T_{\text{stop}} (\bar{W}, \bar{w})\) then

\[
\Delta_i (\bar{W}, \bar{w}) = \mathbb{E} \left[ \left( W_i - \mathbb{E} \left[ W_{\text{sub}(i)} (\bar{W}, \bar{w}) \right] \right)^+ \right] - c,
\]

and otherwise \(\Delta_i (\bar{W}, \bar{w}) = \mathbb{E} \left[ (\mathbb{E} \left[ W_{\text{sub}(i)} (\bar{W}, \bar{w}) \right] - W_i)^+ \right] - c\), and that by Lemma \textbf{4.5.5}, the only change that can happen to the MST is that \(e_j\) and its substituting edge \(e_{\text{sub}(j)} (\bar{W}, \bar{w})\) swap. Therefore, the only cases where testing \(e_j\) can affect \(\Delta_i\) are:

1. If the substituting edge of \(e_i\) does not change, but the value \(\mathbb{E} \left[ W_{\text{sub}(i)} (\bar{W}, \bar{w}) \right] \neq w_{\text{sub}(i)} (\bar{W} - W_j, \bar{w} + w_j)\), which implies that \(e_{\text{sub}(i)} (\bar{W}, \bar{w}) = e_j\);

2. Else if the tree \(T_{\text{stop}}\) changes such that if \(e_i \notin T_{\text{stop}} (\bar{W}, \bar{w})\) then \(e_i \in T_{\text{stop}} (\bar{W} - W_j, \bar{w} + w_j)\) or the other way around, which implies that \(e_i = e_{\text{sub}(j)} (\bar{W}, \bar{w})\); or

3. Otherwise, if the substituting edge of \(e_i\) changes such that

\[
e_{\text{sub}(i)} (\bar{W}, \bar{w}) \neq e_{\text{sub}(i)} (\bar{W} - W_j, \bar{w} + w_j)
\]

For case 1, assume towards contradiction that \(e_j = e_{\text{sub}(i)} (\bar{W}, \bar{w})\). Then \(e_i\) and \(e_j\) are on the same cycle, one of them is in the tree and the other is out of the tree. Assume first that \(e_i \in T_{\text{stop}} (\bar{W}, \bar{w})\), which implies that \(e_j \notin T_{\text{stop}} (\bar{W}, \bar{w})\). Then by Corollary \textbf{4.5.6} \(\Delta_i (\bar{W}, \bar{w}) = \mathbb{E} \left[ (\mu - W_i)^+ \right] - c\). Since \(e_j\) is outside the tree with expected weight \(\mu\), then every edge in \(\text{Cycle}_j \setminus \{e_j\}\) must have weight greater or equal to \(\mu\), specifically \(\mathbb{E} [W_{\text{sub}(j)}] \geq \mu\).

On the other hand, the substituting edge is by Definition \textbf{4.5.2} the smallest edge in the
Cycle_j \{e_j\}, which contains e_i. Therefore, \(E[W_{sub(j)}] \leq \mu\), and hence \(E[W_{sub(j)}] = \mu\). So the myopic gain of edge \(e_j\) is \(\Delta_j (\bar{W}, \bar{w}) = E[(W_j - \mu)^+] - c\). Using the fact that \(W_i\) and \(W_j\) are independent and identically distributed, we obtain \(\Delta_i (\bar{W}, \bar{w}) = \Delta_j (\bar{W}, \bar{w})\), in contradiction to our assumption. Now assume that \(e_i \notin T^{stop} (\bar{W}, \bar{w})\), which implies that \(e_j \in T^{stop} (\bar{W}, \bar{w})\).

Then by Corollary 4.5.6, \(\Delta_i (\bar{W}, \bar{w}) = E[(W_i - \mu)^+] - c\). Since \(e_j\) is in the tree with expected weight \(\mu\), then \(e_{sub(j)} (\bar{W}, \bar{w}) \notin T^{stop} (\bar{W}, \bar{w})\) and it is the minimal such edge that closes a cycle with \(e_j\). By similar reasons as before it must be that \(E[W_{sub(j)}] = \mu\). So the myopic gain of edge \(e_j\) is \(\Delta_j (\bar{W}, \bar{w}) = E[(\mu - W_j)^+] - c\). Similarly, \(\Delta_i (\bar{W}, \bar{w}) = \Delta_j (\bar{W}, \bar{w})\), in contradiction to our assumption.

Case 2 is symmetric to case 1, and thus also leads to contradiction.

For case 3, since \(e_j\) is the edge whose weight changes, then either \(e_{sub(i)} (\bar{W}, \bar{w}) = e_j\) or \(e_{sub(i)} (\bar{W} - W_j, \bar{w} + w_j) = e_j\). The former case was covered by case 1, so it remains to prove the latter. Assume towards contradiction that \(e_{sub(i)} (\bar{W} - W_j, \bar{w} + w_j) = e_j\). First consider the case \(e_i \notin T^{stop} (\bar{W}, \bar{w})\). Edge \(e_{sub(i)} (\bar{W}, \bar{w}) \in T^{stop} (\bar{W}, \bar{w})\) is the minimal edge in Cycle_\(i\) (\(\bar{W}, \bar{w}\) \{e_i\}), so \(e_i\) can only replace it as substituting edge if \(w_j < E[W_{sub(i)} (\bar{W}, \bar{w})]\). However, for \(e_j\) to enter the tree instead of \(e_{sub(j)} (\bar{W}, \bar{w})\), then \(E_W [W_{sub(j)} (\bar{W}, \bar{w})] < w_j\). Together we get \(E_W [W_{sub(j)} (\bar{W}, \bar{w})] < E[W_{sub(i)} (\bar{W}, \bar{w})]\), in contradiction to the minimality of \(e_{sub(i)} (\bar{W}, \bar{w})\). Now consider the case \(e_i \in T^{stop} (\bar{W}, \bar{w})\).

In this case, assume towards contradiction that \(e_{sub(i)} (\bar{W} - W_j, \bar{w} + w_j) = e_j\). This implies that adding \(e_j\) to the tree \(T^{stop} (\bar{W} - W_j, \bar{w} + w_j)\) will close a cycle with \(e_i\). Since \(E[W_{sub(i)} (\bar{W}, \bar{w})] \neq E[W_{sub(j)} (\bar{W}, \bar{w})]\), then the edges \(e_{sub(j)} (\bar{W}, \bar{w}) \notin T^{stop} (\bar{W}, \bar{w})\) and \(e_{sub(i)} (\bar{W}, \bar{w}) \notin T^{stop} (\bar{W}, \bar{w})\) cannot form the same cycle with the tree (if they did then only the heavier of them would have been the substituting edge for both \(e_j\) and \(e_i\)). However, this implies that the tree \(T^{stop} (\bar{W}, \bar{w})\) had a cycle containing \(e_j\) and \(e_i\), in contraction to it being a tree.

In conclusion, testing \(e_j\) cannot have any affect on \(\Delta_i\). \(\square\)

**Proof of Theorem 4.5.4.** Proceed by induction on \(N\), the number of untested edges. For \(N = 1\), the base case, the myopic policy is trivially an optimal policy. For the inductive step,
consider $N > 1$ untested edges. Assume by the inductive hypothesis that for any subset of $\bar{W}$ of size less or equal to $N - 1$, an optimal policy tests according to descending myopic gains. We shall show that it is also optimal for $N$ untested edges. Order the untested edges such that $\Delta_1 (\bar{W}, \bar{w}) \geq \Delta_2 (\bar{W}, \bar{w}) \geq \ldots \geq \Delta_N (\bar{W}, \bar{w})$. Denote by $\Pi^{opt}$ an optimal policy. Clearly if $\Pi^{opt}$ stops then our policy does the same by Theorem 4.4.3 and the proof follows. Also, if there exists a single edge with positive myopic gain, then this edge is clearly $e_1$ and $\Pi^{opt}$ tests it, so the optimal policy tests in descending order of non-negative myopic gains (because it is the only order for one edge), which completes the proof. Thus, assume there is more than one edge with positive myopic gain. Let $e_i$ be the first edge that $\Pi^{opt}$ tests. Clearly, if $\Pi^{opt}$ tests $e_i$ such that $\Delta_i (\bar{W}, \bar{w}) = \Delta_1 (\bar{W}, \bar{w})$, then by the induction hypothesis, the rest of the testing order of the optimal policy is in descending order of myopic gains and the proof follows. We remain with the case that $\Pi^{opt}$ tests some edge $e_i$ such that $\Delta_1 (\bar{W}, \bar{w}) > \Delta_i (\bar{W}, \bar{w}) \geq 0$. In policy $\Pi^{opt}$, testing $e_i$ will possibly lead to some change in the optimal tree, but by Lemma 4.5.7 and Corollary 4.4.2 no matter what this change is, the myopic gain $\Delta_1$ will remain the highest non-negative myopic gain. Hence, an edge obtaining a myopic gain that equals $\Delta_1$ has to be tested next by the induction hypothesis. Without loss of generality, assume it is $e_1$. Denote by $\Pi^{my}$ a myopic policy that tests according to descending myopic gains, and tests $e_1$ first. Assume towards contradiction that $\Pi^{my}$ is strictly suboptimal, i.e., $J^{opt} (\bar{W}, \bar{w}) > J^{my} (\bar{W}, \bar{w})$.

We define a suboptimal policy $\Pi^{sub}$ that tests $e_1$ first, $e_i$ second, and then mimics the optimal policy. Clearly, $J^{my} (\bar{W}, \bar{w}) \geq J^{sub} (\bar{W}, \bar{w})$ because they preform the same first test and, according to the induction hypothesis, $\Pi^{my}$ preforms optimally after the first test.

By Lemma 4.5.7 testing $e_i$ does not influence $\Delta_1$ and vice versa. Since both myopic gains are non-negative, both policies $\Pi^{sub}$ and $\Pi^{opt}$ always test both edges $e_1$ and $e_i$. After conducting both tests the sample path will be the same for both policies by the induction hypothesis. Hence, $J^{sub} (\bar{W}, \bar{w}) = J^{opt} (\bar{W}, \bar{w})$, which concludes the proof. □
4.6 An Optimal Myopic Testing Rule for Symmetric LPPTs

In this section, we study another class of LPPTs for which myopic rules are optimal, both for stopping and testing. Specifically, we study LPPT problems that are symmetric, and in which there is a general relation between the distributions of the unknown coefficients known as the convex order. This case is different from the MST problem which in general is a-symmetric (as will be shown below), and more general than the MST in the sense that the objective coefficients are in convex order and need not be identically distributed.

We start in Section 4.6.1 by defining symmetric problems, and discuss the symmetry of the selection problem and the a-symmetry of the MST problem. In Section 4.6.2 we define the convex order, and present some of its properties. Finally, in Section 4.6.3 we define the myopic policy and show that it is optimal for the class of symmetric LPPTs where unknown coefficients satisfy convex order.

4.6.1 Symmetric LPPTs

Intuitively, we say that an optimization problem is symmetric if the only defining characteristic of an unknown parameter is its value. For example, the selection problem presented in Section 4.2 is symmetric, as we can permute the value of parameters and obtain an equivalent problem. In contrast, the MST problem is not symmetric, because every parameter is associated with an edge on a graph, and permuting the parameters values can result in a considerably different optimal value. Before proving these statements, we start with a definition of symmetric LPPTs:

Definition 4.6.1. An LPPT is symmetric if the associated function $\varphi(\bar{w})$ is a symmetric function.

That is, any permutation on the value of the parameters results with the same optimal value for the optimization problem.

The next lemma characterizes an interesting family of symmetric problems. It includes the Selection problem discussed in Section 4.2
Lemma 4.6.2. Every LPPT that is defined by a submodular function \( f(S) \) that only depends on the cardinality of the set \( S \) defines a symmetric LPPT problem.

Proof. Let \( P \) denote an LPP with objective coefficients \( \bar{w} \), and let \( P_r \) denote an identical LPP with a permutation \( \bar{w}_r \) over the coefficients. We need to show that for every permutation \( \bar{w}_r \) of the coefficients \( \bar{w} \), the resulting value of the optimal solution for the two LPPs is identical, that is \( \varphi(\bar{w}) = \varphi(\bar{w}_r) \). We do this in two steps:

1. Show that the feasible region of the two problems is identical, and that every permutation \( \bar{x}_r \) of a solution \( \bar{x} \) is also feasible;
2. Show that for every solution \( \bar{x} \) to \( P \), there exists a permutation \( \bar{x}_r \) that achieves the same objective value for the problem \( P_r \).

To see (1), we look at all the constraints associated with some set cardinality \( l \), which can be written as:

\[
\forall U \subseteq S \text{ s.t. } |U| = l : \sum_{i \in U} x_i \leq f(l).
\]

These constraints are symmetric with respect to the order of the permutation, for every value of \( l \). Therefore for every solution \( \bar{x} \), and a permutation \( r \), the solution \( \bar{x}_r \) is also feasible.

To see why (2) holds, observe that for every permutation \( r \), the following holds:

\[
\bar{w}_r^T \bar{x}_r = \bar{w}^T \bar{x}.
\]

Corollary 4.6.3. The K-Selection problem is a symmetric LPPT.

Lemma 4.6.4. There exists an MST that is a non-symmetric LPPT.

Proof. Figure 4-3 illustrates a non-symmetric MST. We see that for the same graph (an LPP instance) there are different optimal solution values for different permutations of edge weights (objective coefficients). In Figure 4-3A the value of the MST is equal to 21, whereas in Figure 4-3B the value of the MST is equal to 30.
4.6.2 The Convex Order

We now review the definition and main properties of the convex order. In particular, we describe the mean preserving local spread and show its relation to the convex order. These concepts and properties are key to the analysis in Section 4.6.3.

The convex order is a mathematical relation between probability distributions, that is often used to model difference in risk profiles (see Müller and Stoyan (2002)). Intuitively, distributions that are higher in convex order are more variable or spread around their mean values. Formally, we say that $X \leq_{cx} Y$ (X is smaller in convex order than Y), if the expected value of any convex function $u$ on the two random variables results in a lower value for $X$, that is

$$X \leq_{cx} Y \Leftrightarrow \mathbb{E}[u(X)] \leq \mathbb{E}[u(Y)],$$

for any convex function $u$ for which expectations exist.

Figure 4-4 illustrates the convex order in discrete and continuous distributions. In the left side of Figure 4-4 is an example of the convex order relation in discrete distributions. In the figure are three discrete distributions that have the same mean value, but are different otherwise. In particular, the two-point distribution is higher in convex order than the uniform distribution, which is higher in convex order than the binomial distribution. The right side of Figure 4-4 depicts three normally distributed random variables with equal means and different variance. These distributions are monotonically increasing in the convex order where a higher variance corresponds to a higher convex order (e.g., the distribution denoted by 3 is higher both in variance and convex order than the distributions denoted by 1 and 2).
This is not a coincidence as variance is a convex function, which means that convex order implies monotonicity in variance.

### The Mean Preserving Local Spread

Perhaps the simplest type of convex order is the mean preserving local spread as given by the following definition:

**Definition 4.6.5.** *(From Müller and Stoyan (2002))* Let $F$ and $G$ be distribution functions of discrete distribution whose common support is a finite set of points $x_1 < x_2 < \ldots < x_n$ with probability mass function $f$ and $g$ respectively. Then $G$ is said to differ from $F$ by a local spread, if there exists some $i \in 2, 3, \ldots, n - 1$ such that $0 = g(x_i) \leq f(x_i), g(x_{i-1}) \geq f(x_{i-1}), g(x_{i+1}) \geq f(x_{i+1})$, and $g(x_j) = f(x_j)$ for all $j \notin i - 1, i, i + 1$. A local spread is said to be mean preserving if $F$ and $G$ have the same mean. Write $F \leq_{LS} G$ if $G$ is a mean preserving local spread of $F$.

Intuitively, if distribution $G$ can be obtained from a distribution $F$ by shifting some of the probability mass in one point of the support (which we call the focal point) to two adjacent points while preserving the mean, then $G$ is a mean preserving local spread of $F$. Figure 4-5 illustrates two distributions $F$ and $G$ defined over the support $\{1, 2, \ldots, 7\}$. Observe that the conditions for the local spread are satisfied:

- for $i \in 1, 5, 6, 7: g_i = f_i$,
Figure 4-5: An example of the mean preserving local spread.

- $g_2 = f_2 + \epsilon_1,$
- $g_3 = 0, \ f_3 = \epsilon_1 + \epsilon_2$ ($i = 3$ is the focal, and point), and
- $g_4 = f_4 + \epsilon_2.$

We argue that comparing random variables that are ordered in local spread is significantly simpler than analyzing arbitrary distributions that are in convex order. For example, comparing the expected values of some function of two random variables that are in local spread is easier than comparing the expected values of arbitrary distributions, because many of the terms cancel each other.

It is easy to show that local spread implies a convex order between distributions. Interestingly, the reverse direction holds as well. That is, if two distributions are in convex order, then there exists a series of distributions leading from one distribution to another, where every two consecutive distributions are in local spread. This will be critical to the analysis where we essentially reduce the model with general convex order relations to a model where there is only one pair of distributions that are in local spread. Formally:

**Theorem 4.6.6.** (From Müller and Stoyan (2002)) Let $F$ and $G$ be distribution functions of discrete distribution with finite support. Then $F \leq_{cx} G$ holds if and only if there is a finite sequence $F_1, ..., F_k$ with $F_1 = F$ and $F_k = G,$ such that $F_i \leq_{LS} F_{i+1}$ for $i = 1, ..., k - 1.$
4.6.3 An Optimal Myopic Policy

In this section, we show that, similarly to the MST, the myopic policy (Definition 4.3.9) is optimal for symmetric LPPTs that have the same mean. The optimality of the myopic stopping rule (Definition 4.3.7) for the symmetric case, is a special case of Theorem 4.4.3, and we are left to show the myopic testing rule (Definition 4.3.8) is also optimal.

We first observe an interesting property of the myopic policy, that for symmetric LPPTs, when the unknown coefficients are in convex order, the myopic policy may only choose to test the coefficient that is highest in convex order:

**Lemma 4.6.7.** For symmetric LPPTs with unknown coefficients that are in convex order $W_1 \leq_{cx} W_2 \ldots \leq_{cx} W_k$, testing coefficient $W_k$ obtains the highest myopic gain:

$$k = \arg\max_{i \in [K]} \Delta_i(\bar{W}, \bar{w}), \quad \text{for all states } (\bar{W}, \bar{w}).$$

**Proof.** See Appendix C.1.

We are now ready to present the main result of this section.

**Theorem 4.6.8.** For symmetric LPPTs with unknown coefficients that are in convex order $W_1 \leq_{cx} W_2 \ldots \leq_{cx} W_k$, the myopic policy is optimal.

**Proof.** See Appendix 4.6.8

Intuitively, this tells us that when the only distinguishing factor between two unknown parameters is their distributions (as is the case in symmetric problems), we should favor testing the coefficient that is “more variable” than others (e.g., the parameter that is highest in convex order). This is the parameter that gives us most information, and is more likely to improve the optimization outcome. As an extreme example, consider the constant $\mu$ which is a trivial distribution that is smaller in convex order than any other distribution with mean $\mu$. Testing this distribution benefits less than testing any other distribution (and in fact, it is never optimal to test it). Note that in the absence of symmetry (such as in MSTs), the
specific structure of the problem must also be taken into account when deciding on which parameter to test, in addition to uncertainty reduction.

4.7 Conclusions and Future Directions

In this chapter, we studied offline testing in the context of a broad class of stochastic combinatorial optimization problems. These are problems that can be formulated as linear programs where the constraints polyhedron is a polymatroid, and where the objective coefficients are random variables. We showed that when the objective coefficients share the same mean value, a simple myopic rule optimally determines when to stop testing objective coefficients. In addition, we studied several interesting cases in which myopic rules are also optimal in deciding which coefficient to test.

What seems to be the most restrictive assumption in our analysis, is that random coefficients share the same mean value. It would be interesting to study a more general case where random coefficients are associated with arbitrary distributions. One may wonder whether the myopic policy remains optimal in such cases.

Another interesting direction to peruse is to study an online testing problem of polymatroids where decision variables can be fixed progressively, and where the testing cost decreases with the number of remaining unknown objective coefficients. This could potentially generalize the scheduling problems studied in the first two chapters of the thesis (which can be formulated as polymatroid optimization, but are online).

Finally, while polymatroid optimization represent a general class of problems, a natural direction to proceed is to generalize it further. For example, to general linear programs, or even to problems where the objective can be expressed as a convex function of the random coefficients.
Appendix A

Proofs for Chapter 2

A.1 Proof of Lemma 2.2.1

Proof. We prove by contradiction. Let $\pi$ denote the optimal policy, and assume that at some point jobs 1 and 2 are processed one after the other, and that the ratio of job 1 is greater than the ratio of job 2. We introduce a policy $\pi'$ which is identical to policy $\pi$ except for processing job 2 immediately before processing job 1 (see Figure A-1). It is easy to see that the interchange only affects jobs 1 and 2, as the expected completion time of any other job remains the same.

We denote by $T_i, W_i$ the random processing time and weight of job $i$. One of the jobs refers to a known job with ratio $t_i/w_i$, while the other may refer to either another known job, or to an unknown job with ratio $\rho$. The completion time of job 1 increased by $T_2$, and the completion time of job 2 decreased by $T_1$. Therefore, the objective decreased by $E[T_1 W_2 - T_2 W_1]$. We then obtain the following:

$$E[T_1 W_2 - T_2 W_1] = E[T_1] E[W_2] - E[T_2] E[W_1]$$
$$= E[W_1] E[W_2] \left( \frac{E[T_1]}{E[W_1]} - \frac{E[T_2]}{E[W_2]} \right)$$
$$> 0,$$
whereas we use the independence between the two jobs for the first equality, and the assumption that the ratio of job 1 is higher than that of job 2 to obtain the inequality. This contradicts the optimality of policy \( \pi \).

\[ \square \]

**A.2 Proof of Lemma 2.2.2**

*Proof.* By contradiction, assume that at some point of time the optimal policy \( \pi \) processes a job \( i \), s.t. \( \mathbb{E}[T_i]/\mathbb{E}[W_i] > \rho_1 \) (job 1 is a known job). Job \( i \) can be either a known or an unknown job. The resulting schedule is illustrated in Figure A.2. We denote by \( L \) the time interval between the processing of jobs \( i \) and 1 during which testing and processing of other jobs might take place. Let \( W_L \) represent the total weight of jobs processed in the time intervals \( L \), and let \( T_L \) denote the total duration of the time interval \( L \). Note that for any policy \( \pi \), \( T_L \) and \( W_L \) random variables (as a special case these could be constants). In what follows we ignore the part of the objective corresponding to the elapsed time, as this can be seen as a sunk cost independent of future actions.

Under policy \( \pi \), the expected weighted completion time of jobs \( i \) and 1 are \( \mathbb{E}[T_iW_i] \), and \( \mathbb{E}[(T_i + T_L + t_1)w_1] \). If \( C_L \) denotes the expected weighted completion times of the jobs processed in the time intervals \( L \) under policy \( \pi \), then the part of the objective value corresponding to jobs \( i, 1 \) and the jobs processed in \( L \), is

\[
C_{\pi} = \mathbb{E}[T_iW_i + C_L + (T_i + T_L + t_1)w_1].
\]  

(A.1)

We now show that we can construct a policy \( \pi' \) that achieves a lower expected objective value. Policy \( \pi' \) imitates policy \( \pi \) over the interval \( L \), while changing the order in which the
processing of jobs 1 and \(i\), and the actions in the interval \(L\) are performed.

We start with the cases when \(\mathbb{E}[W_L] = 0\), that is, when no processing takes place in \(L\). If \(\mathbb{E}[T_L] = 0\), then the interval \(L\) is empty, and jobs \(i\) and 1 are processed consecutively. From Lemma 2.2.1 we know that policy \(\pi\) is not optimal. If \(\mathbb{E}[W_L] = 0\) and \(\mathbb{E}[T_L] > 0\), then the policy \(\pi\) always performs tests (and only tests) in \(L\). In this case \(C_L = 0\) (but \(T_L > 0\)), and so \(C_x = \mathbb{E}[T_iW_i + (T_i + T_L + t_1)w_1]\). An improved policy \(\pi'\) is identical to policy \(\pi\) except for processing job 1 before performing the test actions. It is easy to see that only job 1 is affected by this change and that its completion time decreases.

Assume now that under policy \(\pi\), \(\mathbb{E}[W_L] > 0\). We define the ratio \(\rho_L = \mathbb{E}[T_L]/\mathbb{E}[W_L]\). We are in one of three cases:

1. \(\rho_1 < \mathbb{E}[T_i]/\mathbb{E}[W_i] \leq \rho_L\);
2. \(\rho_1 < \rho_L < \mathbb{E}[T_i]/\mathbb{E}[W_i]\);
3. \(\rho_L \leq \rho_1 < \mathbb{E}[T_i]/\mathbb{E}[W_i]\).

Note that we can separate to different cases due to the independence of \(W_i\) and \(T_i\). For any policy \(\pi\), the ratio \(\rho_L\) is fixed and can be determined in advance before taking any action. Calculating \(\rho_L\) might be complicated and computationally challenging, but nonetheless, can be done in finite time.

Case 1, \(\rho_1 < \mathbb{E}[T_i]/\mathbb{E}[W_i] \leq \rho_L\). We construct the policy \(\pi'\) as follows: first process job 1, process job \(i\), and then perform the actions in the interval \(L\). The relevant part of the objective value under policy \(\pi'\) is

\[
C_{\pi'} = \mathbb{E}[t_1w_1 + (t_1 + T_i) W_i + (C_L + t_1W_L)],
\]
whereas the weighted time of job 1 is $t_1w_1$, the weighted time of job $i$ is $(t_1 + T_i)W_i$, and the weighted times of the jobs in interval $L$ is $C_L + t_1W_L$. The latter is due to a delay of length $t_1$ to the jobs in $L$, which increased the objective value by $t_1W_L$. The difference in the objective values between the two policies is therefore

$$C_\pi - C_{\pi'} = \mathbb{E}[T_iW_i + C_L + (t_1 + T_L + t_1)w_1] - \mathbb{E}[t_1w_1 + (t_1 + T_i)W_i + (C_L + t_1W_L)]$$

$$= \mathbb{E}[T_iw_1 + T_Lw_1 - t_1W_i - t_1W_L]$$

$$= \mathbb{E}[T_iw_1 - t_1W_i] + \mathbb{E}[T_Lw_1 - t_1W_L]$$

$$= (\mathbb{E}[T_i]w_1 - t_1\mathbb{E}[W_i]) + (\mathbb{E}[T_L]w_1 - t_1\mathbb{E}[W_L])$$

$$= w_1\mathbb{E}[W_i]\left(\frac{\mathbb{E}[T_i]}{\mathbb{E}[W_i]} - \frac{t_1}{w_1}\right) + w_1\mathbb{E}[W_L]\left(\frac{\mathbb{E}[T_L]}{\mathbb{E}[W_L]} - \frac{t_1}{w_1}\right).$$

Since $\rho_1 < \mathbb{E}[T_i]/\mathbb{E}[W_i] \leq \rho_L$, we obtain that $C_\pi - C_{\pi'} > 0$ and that policy $\pi'$ is sub-optimal.

Similarly to Case 1, one can construct policies that outperform policy $\pi$ under the assumptions of Cases 2 and 3. For Case 2, policy $\pi'$ is similar to policy $\pi$ with an interchange between the processing of jobs $i$ and 1. For Case 3, in policy $\pi'$ we first process the jobs in $L$, then we process jobs 1, and finally we process job $i$ and imitate policy $\pi$ thereafter. This contradicts the assumption that processing a job $i$ when $\rho_1 < \mathbb{E}[T_i]/\mathbb{E}[W_i]$ is optimal.

\[\Box\]

### A.3 Proof of Lemma 2.3.3

**Proof.** Using Lemma 2.2.2 we know that processing any other job is sub-optimal. It is left to show that testing is also sub-optimal. We prove this by induction on the number of unknown jobs $N$. **Base, $N = 1$:** Assume by contradiction that the under the optimal policy $\pi$, the control at state $(N, [t_1, w_1, ..., t_n, w_n])$ is to test. We construct a policy $\pi'$, which processes
job 1, tests the unknown job, and then follows policy $\pi$ (Figure A-3). From Lemma 2.2.1, we know that after testing, both policies process all jobs according to the WSPT rule. The difference in the objective value between the two policies is due to (1) an additional testing delay under policy $\pi$, and (2) sub-optimal processing order under policy $\pi'$ when the ratio of the tested job is smaller than $\rho_1$. We obtain the following:

$$J_{mrg}^\pi (1, [t_1, w_1, ...]) - J_{mrg}^\pi' (1, [t_1, w_1, ...])$$

$$= t_a w_1 + \text{Prob} \left( \frac{T}{W} < \frac{t_1}{w_1} \right) \mathbb{E} \left[ Tw_1 - t_1W \left| \frac{T}{W} < \frac{t_1}{w_1} \right. \right]$$

$$= t_a w_1 - \text{Prob} \left( \frac{T}{W} < \frac{t_1}{w_1} \right) \mathbb{E} \left[ t_1W - Tw_1 \left| \frac{T}{W} < \frac{t_1}{w_1} \right. \right]$$

$$= t_a w_1 - \mathbb{E} \left[ (t_1W - Tw_1)^+ \right]$$

$$= w_1 \left( t_a - \mathbb{E} \left[ (\rho_1W - T)^+ \right] \right) > 0,$$

which contradict the optimality of policy $\pi$.

Step, $N > 1$: By contradiction, assume that the optimal control under policy $\pi$ is test. Using the induction hypothesis one of two scenarios materializes: (1) the ratio of the tested job is greater than $\rho_1$ in which case job 1 is processed immediately after testing; (2) the ratio of the tested job is less than $\rho_1$ in which case, the tested job is processed immediately, followed by the processing of job 1. We compare the objective of policy $\pi$ with a policy $\pi'$ that tests only after processing job 1 (see Figure A-3). In both cases, the difference in the objective is an outcome of additional testing times (under policy $\pi$ there is one more job in the system when testing takes place), and there are savings to policy $\pi$ from an improved processing order of job 1 and the tested job. We obtain the following expression for the difference:

$$J_{mrg}^\pi (N, [t_1, w_1, ...]) - J_{mrg}^\pi' (N, [t_1, w_1, ...])$$

$$= t_a w_1 + \text{Prob} \left( \frac{T}{W} < \frac{t_1}{w_1} \right) \mathbb{E} \left[ -t_1W + Tw_1 \left| \frac{T}{W} < \frac{t_1}{w_1} \right. \right]$$

$$= t_a w_1 - \mathbb{E} \left[ (t_1W - Tw_1)^+ \right]$$
\begin{align*}
&= t_a w_1 - w_1 \mathbb{E} \left[ \left( \frac{t_1 W - T}{w_1} \right)^+ \right] \\
&= w_1 \left( t_a - \mathbb{E} \left[ (\rho_1 W - T)^+ \right] \right) \\
&> 0.
\end{align*}

Since policy $\pi'$ has a lower objective, we obtain a contradiction to the optimality of policy $\pi$, which proves that testing and processing unknown jobs before processing job 1 are not optimal. \hfill \Box

### A.4 Proof of Lemma 2.3.4

**Proof.** By contradiction, assume that under the optimal policy $\pi$ testing is performed immediately after processing job $i$, such that $\rho_a < \rho_i$ (Figure A-4). Compare policy $\pi$ with a policy $\pi'$ that tests before processing job $i$. If the ratio of the tested job is lower than $\rho_a$ it is processed immediately (Lemma 2.3.3), otherwise, policy $\pi'$ processes job $i$ and imitates policy $\pi$ thereafter. Using marginal cost accounting, we obtain the following difference in their objective:

\[ J_{\text{mrg}}^\pi \left( N, [t_1, w_1, \ldots] \right) - J_{\text{mrg}}^\pi' \left( N, [t_1, w_1, \ldots] \right) \]

\[ = \mathbb{E}_{T_i, W_i} \left[ t_a W_i - \text{Prob} \left( \frac{T}{W} < \rho_a \right) \mathbb{E}_{T, W} \left[ T_i W - T W_i \left| \frac{T}{W} < \rho_a \right. \right] \right] \]

\[ = t_a \mathbb{E} [W_i] - \text{Prob} \left( \frac{T}{W} < \rho_a \right) \mathbb{E}_{T, W_i} \left[ E_{T_i} W_i \left| T_i W - T \mathbb{E} W_i \left| \frac{T}{W} < \rho_a \right. \right] \right] \]

\[ = \mathbb{E} [W_i] \left( t_a - \text{Prob} \left( \frac{T}{W} < \rho_a \right) \mathbb{E} \left[ E_{T_i} W_i \left| T_i W - T \mathbb{E} W_i \left| \frac{T}{W} < \rho_a \right. \right] \right) \]

\[ < \mathbb{E} [W_i] \left( t_a - \text{Prob} \left( \frac{T}{W} < \rho_a \right) \mathbb{E} \left[ \rho_a W - T \left| \frac{T}{W} < \rho_a \right. \right] \right) \]

\[ = \mathbb{E} [W_i] \left( t_a - \mathbb{E} \left[ (\rho_a W - T)^+ \right] \right) \]

\[ = 0. \]
Figure A-4: Example: testing after processing a medium- or high-ratio job.

We obtain the first equality by comparing the objective of the two policies, conditioning on the ratio of the tested job. If the tested job has low-ratio, it is processed immediately after testing. The difference between the two policies is in the additional testing time, and the processing order of job \( i \) and the tested job. The difference in the objective is: 

\[
\text{Prob} \left( \frac{T}{W} < \rho_a \right) \mathbb{E}_{T_i,W_i} \left[ t_a W_i - \mathbb{E}_{T,W} \left[ T_i W - TW_i \left| \frac{T}{W} < \rho_a \right] \right] \right) ; \text{otherwise, if the ratio of the tested job is not low, the processing order is the same under both policies, while there is still the additional testing cost of: } \text{Prob} \left( \frac{T}{W} > \rho_a \right) \mathbb{E}_{T_i,W_i} [t_a W_i]. \]

The third equality results from the independence of job \( i \) and the tested job, and the following inequality holds since \( \rho_a < \rho_i \). Note that job \( i \) is allowed to have a random processing time and weight to cover the case when job \( i \) is an unknown job that is processed without testing. \( \square \)

### A.5 Proof of Theorem 2.3.6

**Proof.** We prove by induction on \( N \), that for any state \( s = (N, [t_1, w_1, ..., t_n, w_n]) \), testing is not optimal. Denote by \( \text{test} \) the policy that tests the unknown job at state \( s \).

Starting with \( N = 1 \), the case with no known jobs is trivial. If \( \rho_1 < \rho \), then Lemma 2.3.3 implies that policy \( \text{test} \) is not optimal and the claim holds. Otherwise, if \( \rho < \rho_1 \), we compare policy \( \text{test} \) with the the policy \( \text{proc - all} \), which processes all jobs in non-decreasing order of their ratio (in which case the unknown job has the lowest ratio of all jobs). Using marginal cost accounting, the difference in the objective value between the two policies results from the additional testing time under policy \( \text{test} \), and the sub-optimal ordering of policy \( \text{proc - all} \).

The difference is therefore:

\[
J_{\text{mrq}}^{\text{test}} \left( 1, [t_1, w_1, ...] \right) - J_{\text{mrq}}^{\text{proc-all}} \left( 1, [t_1, w_1, ...] \right)
\]
\[= t_a \sum w_i + \sum \mathbb{E} [\min (W t_i, w_i T)] - \sum w_i \mathbb{E} [T] \]
\[= t_a \sum w_i + \sum w_i \mathbb{E} \left[ \min \left( W \frac{t_i}{w_i}, T \right) \right] - \sum w_i \mathbb{E} [T] \]
\[= \sum w_i (t_a - \mathbb{E} [T - \min (W \rho_i, T)]) \]
\[= \sum w_i (\mathbb{E} \left[ (\rho_a W - T)^+ \right] - \mathbb{E} \left[ (T - \rho W)^+ \right]) \]
\[> \sum w_i (\mathbb{E} \left[ (\rho W - T)^+ \right] - \mathbb{E} \left[ (T - \rho W)^+ \right]) \]
\[= 0. \]

In the fourth equality we use Definition 2.3.1 to substitute for \( t_a \) with an equivalent expression. The last equality results from simple arithmetic. This shows that \( J_{test}^{mr} (1, [t_1, w_1, ...]) > J_{proc-all}^{mr} (1, [t_1, w_1, ...]) \), which means that testing is not optimal for the base case.

Now, assume that \( N > 1 \) and the claim holds for \( N - 1 \). Once again, if \( \rho_1 < \rho \) the claim follows from Lemma 2.3.3. For \( \rho < \rho_1 \), the proof follows by contradiction. Assume that testing is optimal under the optimal policy \( \pi \) (see Figure A-5). By the induction hypothesis, after testing we have \( N - 1 \) unknown jobs and the optimal policy \( \pi \) processes all jobs in a non-decreasing order of their ratio. Thus, if the tested job’s ratio is smaller than \( \rho \), it will be processed before processing the next unknown job (there is at least one such job since \( N > 1 \)). If the tested job has a greater ratio, then it is processed at some point after processing the first unknown job. We compare the objective value of policy \( \pi \) with that of policy \( \pi' \) that first processes an unknown job, then tests, and then processes all jobs in a non-decreasing order of their ratio. The difference in the objective value between the two policies includes an ordering costs, and testing costs. We denote by \( T, W, T_1, W_1 \) the processing times and the weights of the tested and the processed (unknown) job respectively. The difference is:

\[ J_{mr}^\pi (N, [t_1, w_1, ...]) - J_{mr}^{\pi'} (N, [t_1, w_1, ...]) \]
\[= \mathbb{E}_{T_1, W_1} \left[ t_a W_1 + \text{Prob} \left( \frac{T}{W} < \rho \right) \mathbb{E}_{T, W} \left[ TW_1 - WT_1 \left| \frac{T}{W} < \rho \right] \right] \]
\[= t_a \mathbb{E} [W_1] - \text{Prob} \left( \frac{T}{W} < \rho \right) \mathbb{E}_{T_1, W_1, T, W} \left[ WT_1 - TW_1 \left| \frac{T}{W} < \rho \right] \right] \]
The first equality is a direct outcome of marginal cost accounting. For the third equality we use the independence between the tested and processed unknown jobs. Finally, the inequality results from Definition 2.3.1 using the assumption that $\rho < \rho_a$. The objective value under policy $\pi$ exceeds the one under policy $\pi'$, which is a contradiction to the optimality of testing.

### A.6 Proof of Theorem 2.5.5

**Proof.** There are two directions to be proved. If

$$
\left( N\mathbb{E}[W] + \sum_i w_i \right) t_a < (N-1)\mathbb{E}[(W\mathbb{E}[T] - \mathbb{E}[W]T^+) + \\
+ \sum_{i\in\text{Medium}} \mathbb{E}[(Wt_i - w_iT)^+] + \sum_{i\in\text{High}} \mathbb{E}[(w_iT - Wt_i)^+],
$$

Figure A-5: Example: testing when $\rho < \rho_a$. 

$$
= t_a\mathbb{E}[W_1] - \text{Prob} \left( \frac{T}{W} < \rho \right) \mathbb{E}_{T,W} \left[ W\mathbb{E}[T_1] - T\mathbb{E}[W_1] \mid \frac{T}{W} < \rho \right] 
$$

$$
= t_a\mathbb{E}[W_1] - \mathbb{E}[W_1] \text{Prob} \left( \frac{T}{W} < \rho \right) \mathbb{E} \left[ W\frac{\mathbb{E}[T_1]}{\mathbb{E}[W_1]} - T\frac{T}{W} < \rho \right] 
$$

$$
= t_a\mathbb{E}[W_1] - \mathbb{E}[W_1] \text{Prob} \left( \frac{T}{W} < \rho \right) \mathbb{E} \left[ W\rho - T\frac{T}{W} < \rho \right] 
$$

$$
= \mathbb{E}[W_1] \left( t_a - \mathbb{E}[(W\rho - T)^+] \right) 
$$

$$
> 0.
$$
then it is straightforward from Lemma 2.5.3 that policy $STP$ outperforms policy $PA$, and therefore by Theorem 2.3.5 the optimal control is test.

We prove the second direction by induction on $N$. For the base case $N = 1$, there are only two possibilities at state $(N, [t_1, w_1, ..., t_n, w_n])$: (1) process the job with the smallest expected ratio, which implies processing all of the jobs (Theorem 2.3.5), or (2) test the unknown job, in which case there are no additional unknown jobs left, and all jobs will be processed after a single test was done. These correspond to the policies $PA$ and $STP$. We obtain the condition of Equation (2.9) directly from Lemma 2.5.3.

We now assume that the assumption holds for any $N - 1$, and show that it holds for $N$. We first show that if

$$
\left(NE[W] + \sum_i w_i\right) t_a \geq (N - 1)E[(WE[T] - E[W]T)^+] \\
+ \sum_{i \in \text{Medium}} E[(Wt_i - w_iT)^+] + \sum_{i \in \text{High}} E[(w_iT - Wt_i)^+],
$$

then after testing a job, we reach a state that satisfies the same condition.

There are three cases for the realization of the tested job:

1. Low-ratio job

2. Medium-ratio job

3. High-ratio job

Case 1: the low-ratio job is immediately processed in which case the lefthand side (LHS) decreases by $E[W]t_a$. The righthand side (RHS), decreases by $E[(WE[T] - E[W]T)^+]$. Since

$$
t_a = E[(\rho_a W - T)^+] \\
< E[(\rho W - T)^+] \\
= \frac{1}{E[W]}E[(WE[T] - E[W]T)^+],
$$

Case 2: In this case, LHS increases by \((-\mathbb{E}[W] + w_i) t_a\), and RHS increases by:

\[-\mathbb{E}[(\mathbb{W}E[T] - \mathbb{E}[W]T)^+] + \mathbb{E}[(Wt_i - w_iT)^+],\]

where \(\rho_i < \rho\). A sufficient condition for \(LHS \geq RHS\) is

\[
(-\mathbb{E}[W] + w_i) t_a \geq -\mathbb{E}[(\mathbb{W}E[T] - \mathbb{E}[W]T)^+] + \mathbb{E}[(Wt_i - w_iT)^+]
\]

\[
\iff (w_i) t_a - \mathbb{E}[(Wt_i - w_iT)^+] \geq (\mathbb{E}[W]) t_a - \mathbb{E}[(\mathbb{W}E[T] - \mathbb{E}[W]T)^+]
\]

\[
\iff w_i (t_a - \mathbb{E}[(W\rho_i - T)^+]) \geq \mathbb{E}[W] (t_a - \mathbb{E}[(W\rho - T)^+])
\]

\[
\iff w_i (\mathbb{E}[(W\rho_i - T)^+] - t_a) \leq \mathbb{E}[W] (\mathbb{E}[(W\rho - T)^+] - t_a).
\]

We now look at the last inequality for the realization of the medium-ratio job that satisfy the inequality. The RHS does not depend on the realization. The LHS is increasing in \(t_i\), and decreasing in \(w_i\). This means that if a realization \((t_i, w_i)\) satisfy the inequality, then the points \((t_i, w_i + c), (t_i - c, w_i)\), and \((ct_i, cw_i)\) (where \(c < 1\) will also satisfy the inequality. The inequality holds as an equality when \((t_i, w_i) = (\mathbb{E}[T], \mathbb{E}[W])\), therefore it will also hold for all the points \((t, w_i)\) such that \(\rho_a < t/w < \rho\) and \(t_i \leq \mathbb{E}[T]\).

Case 3: As in case 2, LHS increases by \((-\mathbb{E}[W] + w_i) t_a\), and RHS increases by:

\[-\mathbb{E}[(\mathbb{W}E[T] - \mathbb{E}[W]T)^+] + \mathbb{E}[(w_i T - Wt_i)^+],\]

where \(\rho_i > \rho\). A sufficient condition for \(LHS \geq RHS\) is

\[
(-\mathbb{E}[W] + w_i) t_a \geq -\mathbb{E}[(\mathbb{W}E[T] - \mathbb{E}[W]T)^+] + \mathbb{E}[(w_i T - Wt_i)^+]
\]

\[
\iff (w_i) t_a - \mathbb{E}[(w_i T - Wt_i)^+] \geq (\mathbb{E}[W]) t_a - \mathbb{E}[(\mathbb{W}E[T] - \mathbb{E}[W]T)^+]
\]

\[
\iff w_i (t_a - \mathbb{E}[(T - W\rho_i)^+]) \geq \mathbb{E}[W] (t_a - \mathbb{E}[(T - W\rho)^+])
\]

\[
\iff w_i (\mathbb{E}[(T - W\rho_i)^+] - t_a) \leq \mathbb{E}[W] (\mathbb{E}[(T - W\rho)^+] - t_a)
\]

\[
\iff w_i (\mathbb{E}[(T - W\rho_i)^+] - t_a) \leq \mathbb{E}[W] (\mathbb{E}[(W\rho - T)^+] - t_a).
\]
Similarly to Case 2, we want to find the range of values \((t_i, w_i)\) for which the inequality holds. Observe that the RHS is always positive, and does not depend on the realization of the high-ratio job. The LHS is monotonically increasing in \(w_i\), and monotonically decreasing in \(t_i\). This means that if a point \((t_i, w_i)\) satisfies the inequality, then so will the points \((t_i, w_i - \epsilon), (t_i + \epsilon, w_i), (ct_i, cw_i)\) where \(c < 1\). Since the point \((t_i, w_i) = (\mathbb{E}[T], \mathbb{E}[W])\) satisfy the inequality (with equality), we deduce that all the high-ratio jobs with \(w_i \leq \mathbb{E}[W]\) also satisfy the inequality.

By the induction hypothesis, when the condition of Equation (2.9) is met, a policy that tests a job necessarily processes all jobs in the next state, and therefore is the \(STP\) policy. Lemma 2.5.3 guarantees that if

\[
\left( N\mathbb{E}[W] + \sum_i w_i \right) t_a \geq (N - 1)\mathbb{E}[(W\mathbb{E}[T] - \mathbb{E}[W]T^+) + \sum_{i \in \text{Medium}} \mathbb{E}[(Wt_i - w_iT)^+] + \sum_{i \in \text{High}} \mathbb{E}[(w_iT - Wt_i)^+],
\]

then policy \(PA\) is better than policy \(STP\), which means that processing all jobs is the optimal control. \(\square\)

A.7 Example for a model where the myopic policy that is not optimal

Consider the following model: \(N_0 = 2\), \(t_a = 0.53\), and where the distribution \(\mathcal{D}_{T,W}\) is given by: \(P(3, 1) = 0.5\), \(P(1, 3) = 0.49\), \(P(100, 110) = 0.01\). The parameters \(\rho\) and \(\rho_a\) are equal to 0.97, and 0.69, respectively (see Figure A-7 for an illustration).

The dynamic programming solution to the model is summarized in Table A-6. For every state, it shows the value function of the optimal policy \((J_{mrg})\), the cost of processing all jobs \((J_{PA})\), and the optimal controls. While the myopic policy chooses to process all jobs at the initial state \((2, [], [])\), the optimal policy performs a test, and based on the realization, either processes all jobs or performs an additional test.
<table>
<thead>
<tr>
<th>State</th>
<th>$J_{PA}$</th>
<th>$J_{mrg}$</th>
<th>Optimal control</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, [3, 100], [1, 110])</td>
<td>0</td>
<td>0</td>
<td>process all</td>
</tr>
<tr>
<td>(0, [100, 3], [110, 1])</td>
<td>0</td>
<td>0</td>
<td>process all</td>
</tr>
<tr>
<td>(0, [100, 100], [110, 110])</td>
<td>0</td>
<td>0</td>
<td>process all</td>
</tr>
<tr>
<td>(0, [100], [110])</td>
<td>0</td>
<td>0</td>
<td>process all</td>
</tr>
<tr>
<td>(1, [3], [1])</td>
<td>115.96</td>
<td>115.96</td>
<td>process all</td>
</tr>
<tr>
<td>(2, [], [])</td>
<td>235.11</td>
<td>234.93</td>
<td>test one</td>
</tr>
<tr>
<td>(0, [3, 3], [1, 1])</td>
<td>0</td>
<td>0</td>
<td>process all</td>
</tr>
<tr>
<td>(1, [], [])</td>
<td>112.97</td>
<td>112.97</td>
<td>process all</td>
</tr>
<tr>
<td>(0, [], [])</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>(0, [3], [1])</td>
<td>0</td>
<td>0</td>
<td>process all</td>
</tr>
<tr>
<td>(1, [100], [110])</td>
<td>419.97</td>
<td>386.79</td>
<td>test one</td>
</tr>
</tbody>
</table>

Figure A-6: The solution to the model.

Figure A-7: The Distribution $\mathcal{D}_{T,W}$
A.8 Proof of Theorem 2.6.6

Proof. From Lemma 2.6.5 we know that it is optimal to test as long as the following holds:

\[ \beta > \frac{N + \omega_H/E[W]}{N - 1}. \]

In the initial state of the system, \( N = N_0 \) and \( n = 0 \). Each time we test, the denominator \((N - 1)\) decreases by 1, and the nominator \((N + \omega_H/E[W])\) increases by at most 1 (since \( w_i \leq 2E[W] \)). In the worst case, after \( N_{\text{tests}} \) tests (and processing of low-ratio jobs), the right-hand side is at most \((N_0 + N_{\text{tests}}) / (N_0 - 1 - N_{\text{tests}})\). A sufficient condition for testing after \( N_{\text{tests}} \) periods is

\[ \beta > \frac{(N_0 + N_{\text{tests}})}{(N_0 - 1 - N_{\text{tests}})}, \]

which implies that the control \textit{test one} is optimal for at least

\[ N_{\text{tests}} = \left\lfloor \frac{N_0 (\beta - 1)}{(\beta + 1)} - \frac{\beta}{\beta + 1} \right\rfloor \]

periods. \hfill \square

A.9 Proof of Lemma 2.4.3

Proof. By induction on \( N \). Base: for \( N = 0 \) the claim holds as \( J_{LD}(N, \omega_M, \omega_H, \tau_M, \omega_{\tau}) = 0 \).

Now assume the claim holds true for some \( N - 1 \). The value function is the minimum of two expressions. Since \( \tau_M \) and \( \omega_{\tau} \) are non-negative, the first (test-one) is non-decreasing in \( \tau_M \) and \( \omega_{\tau} \), and \( J_{LD}(N - 1, \cdot, \cdot, \cdot, \cdot) \) is also non-decreasing in \( \tau_M \) and \( \omega_{\tau} \) by the induction hypothesis. The second term (process-all) is also non-decreasing in \( \tau_M \) and \( \omega_{\tau} \). \hfill \square
A.10 Proof of Lemma 2.4.4

Proof. By induction on $N$. Base: $N = 1$. Recall that:

$$J_{LD}(N, \omega_M, \omega_H, \tau_M, \omega_T) = \min \left\{ \begin{array}{ll}
\mathbb{E}[TW] + (\omega_M + \omega_H + \mathbb{E}[W])t_a + \omega_T & \text{test one} \\
\mathbb{E}[TW] + \mathbb{E}[W]\tau_M + \mathbb{E}[T]\omega_H & \text{process all.}
\end{array} \right. \tag{A.2}
$$

The value function under the control test-one is increasing in $\omega_T$, and the value function under process-all is unaffected by changes in $\omega_T$. This is the form of a threshold policy.

Step: once again, the value function under the control process all is unaffected by changes in $\omega_T$, while the value function under the control test one is strictly increasing in $\omega_T$ (using Lemma 2.4.3). □

A.11 Proof of Lemma 2.4.5

Proof. We want to show that for any state $(N, \omega_M, \omega_H, \tau_M, \omega_T)$, the following holds:

$$1 \leq \frac{J_\delta(N, \omega_M, \omega_H, \tau_M, \omega_T)}{J_{LD}(N, \omega_M, \omega_H, \tau_M, \omega_T)} \leq (1 + \delta)^N.$$

We prove this by induction on $N$. We show that for each state and under each control, the ratio in the value functions between the two DP formulation is greater than or equal to 1, and smaller than or equal to $(1 + \delta)^N$.

Starting with the base case, $N = 0$, the control test-one is not feasible (no unknown jobs). Under the control process-all the costs are identical for both formulations and the lemma holds.

Before we continue to the step, we prove two properties about the ADP formulation. □

Lemma A.11.1. The value function $J_\delta(N, \omega_M, \omega_H, \tau_M, \omega_T)$ is non-decreasing in $\omega_M, \omega_H, \tau_M, \omega_T$.

Proof. The proof is similar to the proof of Lemma 2.4.3 and thus omitted. □
Lemma A.11.2. The following holds:

\[
\frac{J_\delta (N, \omega_M', \omega_H', \tau_M', \omega \tau')}{J_\delta (N, \omega_M, \omega_H, \tau_M, \omega \tau)} \leq 1 + \delta,
\]

where \(x \leq x' \leq (1 + \delta)x\) for \(x \in \{\omega_M, \omega_H, \tau_M, \omega \tau\}\).

Proof. We prove by induction on \(N\), starting with the base case \(N = 0\). Testing is not allowed (no unknown jobs), therefore the value function is equal to the cost-to-go when choosing the control process-all:

\[
\frac{J_\delta (N, \omega_M', \omega_H', \tau_M', \omega \tau')}{J_\delta (N, \omega_M, \omega_H, \tau_M, \omega \tau)} = \frac{\mathbb{E}[W] N \tau_M + N \mathbb{E}[T] \omega_H + N \mathbb{E}[TW] + \left(\frac{N}{2}\right) \mathbb{E}[T] \mathbb{E}[W]}{\mathbb{E}[W] N \tau_M + N \mathbb{E}[T] \omega_H + N \mathbb{E}[TW] + \left(\frac{N}{2}\right) \mathbb{E}[T] \mathbb{E}[W]}.
\]

This expression is bounded by \(1 + \delta\) as \(\tau_M\) and \(\omega_H\) satisfy: \(\tau_M' \leq (1 + \delta)\tau_M, \omega_H' \leq (1 + \delta)\omega_H\).

For the step, we assume the induction hypothesis holds for \(N - 1\) and prove that it holds for \(N\). We show that for each of the controls the ratio between the value functions under the control is bounded by \(1 + \delta\).

For the control process-all the ratio is bounded similarly to the base case. For the control test-one we observe that the cost function part satisfies:

\[
\mathbb{E}[TW] + (\omega_M' + \omega_H' + N \mathbb{E}[W]) t_a + \omega \tau' \leq (1 + \delta) (\mathbb{E}[TW] + (\omega_M + \omega_H + N \mathbb{E}[W]) t_a + \omega \tau),
\]

and that the terms \(d (N - 1) \mathbb{E}[W]\) and \(J_\delta (N - 1, [\omega_M], [\omega_H], [\tau_M], [\omega \tau])\) are the same under both formulations. We are left to show that the following two inequalities hold:

\[
J_\delta \left(\begin{array}{c}
N - 1, [\omega_M' + v], [\omega_H'], [\tau_M' + d], \\
[\omega \tau' + \mathbb{E} \min (Tv, dW)]
\end{array}\right) \leq (1 + \delta) J_\delta \left(\begin{array}{c}
N - 1, [\omega_M + v], [\omega_H], [\tau_M + d], \\
[\omega \tau + \mathbb{E} \min (Tv, dW)]
\end{array}\right),
\]

and

\[
J_\delta \left(\begin{array}{c}
N - 1, [\omega_M'], [\omega_H' + v], [\tau_M'], \\
[\omega \tau' + \mathbb{E} \min (Tv, dW)]
\end{array}\right) \leq (1 + \delta) J_\delta \left(\begin{array}{c}
N - 1, [\omega_M], [\omega_H + v], [\tau_M], \\
[\omega \tau + \mathbb{E} \min (Tv, dW)]
\end{array}\right).
\]
These hold by the induction hypothesis, and using the following inequality which holds for $v \geq 0$:

$$\left\lceil \frac{x' + v}{x + v} \right\rceil \leq \left\lceil \frac{x + v}{x + v} \right\rceil \leq 1 + \delta.$$  

The latter holds since that

$$\frac{x + v}{x + v} \leq \frac{x}{x} \leq 1 + \delta,$$

which implies that

$$\frac{\lceil x + v \rceil}{x + v} \leq 1 + \delta.$$

We are now ready to prove the step of the main lemma. Once again, we compare the value functions under each of the controls. For the control process-all the two value functions are identical by definition.

For the test-one control, we look at the terms composing the cost-to-go function:

$$\mathbb{E}[TW] + (\omega_M + \omega_H + N\mathbb{E}[W])t_a + \omega \tau,$$

and,

$$d(N - 1)\mathbb{E}[W]$$

are identical in both formulations.

Using Lemma \textbf{A.11.2} and the induction hypothesis:

$$J_\delta(N - 1, [\omega_M], [\omega_H], [\tau], [\omega\tau]) \leq (1 + \delta)J_\delta(N - 1, \omega_M, \omega_H, \tau, \omega \tau)$$

$$= (1 + \delta)(1 + \delta)^{N-1}J_{LD}(N - 1, \omega_M, \omega_H, \tau, \omega \tau)$$

$$= (1 + \delta)^N J_{LD}(N - 1, \omega_M, \omega_H, \tau, \omega \tau).$$

Similarly,

$$J_\delta\left( N - 1, [\omega_M + v], [\omega_H], [\tau + d], \left[\omega \tau + \mathbb{E}[\min(Tv, dW)]\right] \right).$$
\[
\leq (1 + \delta) J_\delta \left( \frac{N - 1, \omega_M + v, \omega_H, \tau_M + d,}{\omega_T + \mathbb{E} \left[ \min (Tv, dW) \right]} \right)
\]

\[
\leq (1 + \delta)(1 + \delta)^{N-1} J_{LD} \left( \frac{N - 1, \omega_M + v, \omega_H, \tau_M + d,}{\omega_T + \mathbb{E} \left[ \min (Tv, dW) \right]} \right)
\]

\[
= (1 + \delta)^N J_{LD} \left( \frac{N - 1, \omega_M + v, \omega_H, \tau_M + d,}{\omega_T + \mathbb{E} \left[ \min (Tv, dW) \right]} \right)
\]

and,

\[
J_\delta \left( \frac{N - 1, [\omega_M], [\omega_H + v], [\tau_M],}{[\omega_T + \mathbb{E} \left[ \min (Tv, dW) \right]]} \right) \leq (1 + \delta) J_\delta \left( \frac{N - 1, \omega_M, \omega_H + v, \tau_M,}{\omega_T + \mathbb{E} \left[ \min (Tv, dW) \right]} \right)
\]

\[
\leq (1 + \delta)(1 + \delta)^{N-1} J_{LD} \left( \frac{N - 1, \omega_M, \omega_H + v, \tau_M,}{\omega_T + \mathbb{E} \left[ \min (Tv, dW) \right]} \right)
\]

\[
= (1 + \delta)^N J_{LD} \left( \frac{N - 1, \omega_M, \omega_H + v, \tau_M,}{\omega_T + \mathbb{E} \left[ \min (Tv, dW) \right]} \right),
\]

which implies that

\[
\frac{J_\delta (N, \omega_M, \omega_H, \tau_M, \omega_T)}{J_{LD} (N, \omega_M, \omega_H, \tau_M, \omega_T)} \leq (1 + \delta)^N.
\]

Using Lemma [A.11.1] and the fact that rounding only increases the parameters, it is easy to see that the lower bound on the above ratio holds as well. \(\square\)

### A.12 Proof of Lemma 2.5.3

**Proof.** Using marginal cost accounting, testing an unknown job carries an additional expected weighted time of \((N\mathbb{E}[W] + \sum_i \omega_i) t_a\), but, in return, it improves the scheduling order. For any medium-ratio job, the ordering cost of the unknown job and a medium-ratio job \(i\), is by default \(W t_i\) (since the medium-ratio job comes before the unknown job). The savings occur when the realization of the tested job has a lower ratio, in which case the improved ordering cost is \(w_i T\). The expected savings of the pair is therefore \(\mathbb{E}[(W t_i - w_i T)^+]\). Similarly,
the expected savings of high-ratio jobs is \( \mathbb{E}[(w_i T - Wt_i)^+] \), and the savings with respect to other unknown jobs is equal to \( \mathbb{E}[(W\mathbb{E}[T] - W\mathbb{E}[T]^+)\) (and also to \( \mathbb{E}[(\mathbb{E}[W]T - W\mathbb{E}[T])^+] \). □

A.13 Proof of Lemmas 2.6.1, 2.6.2, and 2.6.3

We look at the expression \( \mathbb{E}[\min (T_1, T_2)] \). Let \( Z \) be the r.v. defined as \( Z = \min (T_1, T_2) \). Then, \( Pr(Z \leq z) = 1 - (1 - F(z))^2 \), and \( f_Z(z) = 2(1 - F(z))f(z) \), which means that \( \mathbb{E}[Z] = \int 2(1 - F(z))f(z)zdz \). We get the following equivalent bound on the “process all” policy:

\[
\frac{J_{PA}^N}{J_{OPT}^N} \leq \frac{\mu}{\mathbb{E}[Z]} = \frac{\mu}{\int 2(1 - F(z))f(z)zdz}.
\]

For uniform distribution:

\[
\mathbb{E}[Z] = \int_a^b 2(1 - F(z))f(z)zdz
= \int_a^b 2 \left(1 - \frac{z - a}{b - a}\right) \frac{1}{b - a} zdz
= \frac{2}{(b - a)^2} \left[b z^2 - \frac{z^3}{3}\right]_a^b
= \frac{1}{3 (b - a)^2} (b + 2a) (b - a)^2,
\]

which implies an approximation of

\[
APX = \frac{(a + b)}{2} \frac{3 (b - a)^2}{(b + 2a) (b - a)^2} = \frac{3 (b + a)}{2 (b + 2a)},
\]

this last is monotonically decreasing in \( a \).

For exponential distribution:

\[
\mathbb{E}[Z] = \int_0^\infty 2 (1 - F(z)) f(z) z dz
= \int_0^\infty 2 (1 - 1 + e^{-\lambda z}) \lambda e^{-\lambda z} z dz
\]
\[ \int_{0}^{\infty} 2\lambda e^{-2\lambda z}dz = \frac{1}{2\lambda}, \]

which implies a 2-approximation.

For \( T \sim \begin{cases} a & p \\ b & 1-p \end{cases} \):

\[ \frac{\mu}{E[Z]} = \frac{ap + b(1-p)}{b(1-p)^2 + a(1-(1-p)^2)} = \frac{ap + b(1-p)}{a + (b-a)(1-p)^2}, \]

By plugging in \( a = 0, p = 1/2 \), we get a \( 1/(1-p) = 2 \) approximation. For \( a = 0, b = M, p = 1 - 1/M \), we get an \( M \) approximation.

### A.14 Derivation of the Expected Costs under Policy \( TAPL \)

As noted in Section 2.6.1, the \( TAPL \) policy has two types of additional costs compared with the clairvoyant solution: due to testing delays (as it tests all jobs), and due to sub-optimal processing order.

In terms of testing delays costs, the completion time of all medium- and high-ratio jobs include the testing time of all \( N \) jobs (because these jobs are process only after all jobs were tested). This results in an additional cost of \( t_aE[W|T/W > \rho_a] N \) per job. Low-ratio jobs carry average testing delay costs of \( t_aE[W|T/W < \rho_a] (1+N)/2 \).

In terms of ordering costs, medium- and high-ratio jobs are scheduled optimally with respect to all other jobs. Low-ratio jobs are optimal with respect to medium- and high-ratio jobs, but could be sub-optimal with respect to other low-ratio jobs. This yields an expected additional cost of \( E\left[(W_2T_1 - W_1T_2)^+ | T_1/W_1, T_2/W_2 < \rho_a] \right) \) to every pair of low-ratio jobs.

Using \( N^l \) to denote the number of low-ratio jobs, the total expected cost under policy
$TAPL$ can be written as:

$$J^{TAPL}(N, []) = J^{CL}(N, []) + \mathbb{E} \left[ \binom{N}{2} \right] \mathbb{E} \left[ (W_2T_1 - W_1T_2)^+ \frac{T_1}{W_1}, \frac{T_2}{W_2} < \rho_a \right]$$

$$+ t_a \mathbb{E} [N - N] \mathbb{E} \left[ W \frac{T}{W} > \rho_a \right] + t_a \mathbb{E} [N] \frac{1 + N}{2} \mathbb{E} \left[ W \frac{T}{W} < \rho_a \right].$$

### A.15 Proof of Lemma 2.6.5

**Proof.** A sufficient condition for the optimality of testing is $J^{STOP} - J^{PA} < 0$. Using Lemma 2.5.3, this is equivalent to:

$$\left( N \mathbb{E}[W] + \sum_i w_i \right) t_a \leq (N - 1) \mathbb{E}[(W \mathbb{E}[T] - \mathbb{E}[W]T)^+]$$

$$+ \sum_{i \in \text{Medium}} \mathbb{E}[(Wt_i - w_iT)^+]$$

$$+ \sum_{i \in \text{High}} \mathbb{E}[(w_iT - Wt_i)^+] ,$$

and to

$$\left( N \mathbb{E}[W] + \sum_{i \in \text{High}} w_i \right) t_a \leq (N - 1) \mathbb{E}[(W \mathbb{E}[T] - \mathbb{E}[W]T)^+]$$

$$+ \sum_{i \in \text{Medium}} w_i \left( \mathbb{E}[(W \rho_i - T)^+] - t_a \right)$$

$$+ \sum_{i \in \text{High}} w_i \mathbb{E}[(T - W \rho_i)^+].$$

By plugging $\beta$, we obtain

$$\left( N \mathbb{E}[W] + \sum_{i \in \text{High}} w_i \right) t_a \leq (N - 1) \beta t_a \mathbb{E}[W]$$

$$+ \sum_{i \in \text{Medium}} w_i \left( \mathbb{E}[(W \rho_i - T)^+] - t_a \right)$$

$$+ \sum_{i \in \text{High}} w_i \mathbb{E}[(T - W \rho_i)^+].$$
Since the last two summations are positive ($\rho_i > \rho_a$ implies that $\mathbb{E}[(W \rho_i - T)^+] > t_a$), the following condition is sufficient for the inequality to hold:

$$N\mathbb{E}[W] + \sum_{i \in \text{High}} w_i \leq (N-1)\beta \mathbb{E}[W],$$

or equivalently:

$$\beta > \frac{N + \frac{\omega_H}{\mathbb{E}[W]}}{N-1}.$$

\[\square\]

A.16 An Equivalence between the Initial and the Generalized Model

In this section of the appendix, we show that the generalized model presented in Section 2.7, where testing reveals the class of the respective job, and where every class has an associated joint distribution for the processing time and weight, is equivalent to the initial model. In particular, every job class in the generalized model is replaced by the a deterministic job in the equivalent model, in which the processing time and weight are equal to the expected processing time and expected weight of the respective job class.

To prove this, we take advantage of two properties: (1) the linearity of the expectation operator and the objective function; and (2) the independence between jobs. In what follows, we discuss how to modify the proofs and algorithms to accommodate the general model.

The definition of jobs ratio perfectly extends to the generalized model, with the ratio of a known job $i$ being $\rho_i = \mathbb{E}[T_i] / \mathbb{E}[W_i]$. We define the processing and testing ratios in the same way. Lemma 2.2.1 applies regardless to whether jobs are deterministic or stochastic, and it is easy to see that Lemma 2.2.2 still holds if we replace the quantities $t_1, w_1$ with the random variables $T_1, W_1$ and take expectation. Therefore, we can once again limit the control set to testing or processing the job with the smallest ratio.

The DP formulation for the generalized problem can be developed similarly to the orig-
inal formulation (Section 2.2.3). The state space can be described by $N$, the number of unknown jobs, and by $\bar{T}_1, \bar{W}_1, ..., \bar{T}_n, \bar{W}_n$ which are the expected values of the random variables corresponding to the tested jobs (similarly, $\rho_i$ is assumed to be non-decreasing in $i$).

The generalized DP formulation can now be written as following:

$$J_{\text{ext}}(N, [\bar{T}_1, \bar{W}_1, ..., \bar{T}_n, \bar{W}_n]) = \min \begin{cases} E[TW] + (\sum \bar{W}_i + N\bar{W}) t_a + \\ +E \left[ \sum_{i=1}^n \min \{ W\bar{T}_i, \bar{W}_i T \} \right] + \\ +E \left[ J_{\text{ext}} \left( N - 1, [\bar{T}_1, \bar{W}_1, ..., \bar{T}_n, \bar{W}_n] \cup \{ T, W \} \right) \right] \quad \text{test} \\
E[TW] + (\sum \bar{W}_i + (N - 1) \bar{W}) \bar{T} + \\ +J_{\text{ext}} \left( N - 1, [\bar{T}_1, \bar{W}_1, ..., \bar{T}_n, \bar{W}_n] \right) + \\ N\bar{W} \bar{T}_1 + J_{\text{ext}} \left( N, [T_2, W_2, ..., T_n, W_n] \right) \quad \text{process_u} \\
0 \quad \text{process_1} \end{cases}$$

The value function for the control process unknown, and process job 1 can be derived using the fact the known jobs are independent from each other (e.g., $E[W\bar{T}_1] = E[W] E[\bar{T}_1]$). For the test control, observe that the job ordering cost (i.e., $E \left[ \min \{ W\bar{T}_i, \bar{W}_i T \} \right]$) between the tested job and the already known job $i$, is achieved as following. If the the ratio of the tested job ($T/W$) is smaller than the ratio of job $i$ (i.e., $W\bar{T}_i > \bar{W}_i T$), a cost of $\bar{W}_i T$ is added. Otherwise, if $W\bar{T}_i < \bar{W}_i T$, we add the cost $W\bar{T}_i$. This is equivalent to adding the cost $\min \{ W\bar{T}_i, \bar{W}_i T \}$. The other components of the test control are derived similarly to the original problem.

Considering the DP formulation for the generalized problem, observe that it is identical to a discrete instance of the original problem, where jobs are realized to their expected values.

This means that we can find the value function of the generalized problem using the solution 161
of an instance of the original problem. Practically speaking, the generalized problem reduces to the original problem. This implies that all the properties from Section 2.3 still hold, and that we can use the algorithms developed in Section 2.4 to solve the generalized problem efficiently.
Appendix B

Proofs for Chapter 3

B.1 Proof of Lemma 3.4.7

Proof. Assume by contradiction that there is a unique optimal policy $\pi$ that processes unknown job $j < N$ at some initial state $(T, t)$. We show that there exists a policy $\pi'$ that processes unknown job $N$ at that state and achieves the same or lower objective value.

We define policy $\pi'$ as follows. Policy $\pi'$ imitates policy $\pi$ except for operating on job $j$ (testing or processing) whenever policy $\pi$ operates on job $N$.

Consider the (future) state $s$ in which policy $\pi$ operates on job $N$. By the time we reach state $s$, some of the unknown and known jobs could have been tested and processed, which means that at state $s$, the vector of unknown jobs is a subset of $T$ (that includes $N$), while the vector of known jobs could be entirely different from $t$. Let $T'$ denote the vector of unknown jobs in state $s$ that excludes job $N$, and let $t'$ denote the vector of known jobs in state $s$. The state $s$ can be written as: $s = (T' + N, t')$. Similarly, before processing job $j$, policy $\pi'$ reaches the state $(T' + j, t')$. This is because policy $\pi'$ mimics policy $\pi$ with the difference of having unknown job $j$ instead of $N$ in the respective state.

Using the echelon cost accounting, we know the expected costs incurred by the two policies before reaching the temporary states are the same. Therefore, we only need to compare the value functions at those states.
If job \( N \) is processed under policy \( \pi \), then job \( j \) is processed under policy \( \pi' \), and the expected cost incurred by the two policies is the same:

\[
J_{\text{Process}_N}^\pi(T' + N, t') = \mathbb{E}[T_N] (|t'| + |T'| + 1) + J^\pi(T', t')
\]

\[
J_{\text{Process}_j}^\pi(T' + j, t') = \mathbb{E}[T_j] (|t'| + |T'| + 1) + J^\pi(T', t')
\]

Otherwise, the two policies test jobs \( N \) and \( j \), respectively. The two value functions can be written as:

\[
J_{\text{Test}_N}^\pi(T' + N, t') = t_a (|t'| + |T'| + 1) + \mathbb{E}J^\pi(T', t' + T_N),
\]

and,

\[
J_{\text{Test}_j}^{\pi'}(T' + j, t') = t_a (|t'| + |T'| + 1) + \mathbb{E}J^{\pi'}(T', t' + T_j).
\]

Using the concavity of the value function \( J \) (Lemma 3.3.1), and the fact that \( T_N \leq_c x T_j \), we obtain that the policy \( \pi' \) is as good or better than policy \( \pi \). \( \Box \)

B.2 Proof of Lemma 3.4.9

Proof. We prove the lemma by induction on \( N \). The base of the induction is given by the homogeneous model. For the step, we compare the policy that tests the unknown job \( j \) (denoted by \( \pi \)), with a policy that processes job 1 (denoted by \( \pi' \)). Note that from Lemma 3.4.3 we know that processing any other job cannot be optimal.

Under policy \( \pi \), after testing job \( j \), the optimal policy processes all jobs that are shorter than \( \mu_1^a \) (according to the induction hypothesis). If the tested job is shorter than \( t_1 \) it will be processed first, followed by the processing of job 1. If on the other the tested job turns out to be longer than \( t_1 \), then job 1 is to be processed first.
We define policy $\pi'$ as follows. It starts by processing job 1, and then tests the unknown job $j$. If the duration of job $j$ is shorter than $\mu_1^a$ it is processed immediately, otherwise, policy $\pi'$ imitates policy $\pi$ ever after.

We now compare the expected objective value under the two policies. Figure B-1 shows the sequence of actions under each of the policies depending on the realization of $T_j$.

**Case 1:** $T_j \leq t_1$.

When $T_j \leq t_1$, the two policies coincide after completing the first three actions, which implies that the cost difference in the objective results exactly from these three actions. Using echelon cost accounting:

$$J^\pi(T,t) - J^{\pi'}(T,t) = (N + n)t_a + T_j(N + n) + t_1(N + n - 1) + J^\pi(T - j, t - t_1) - (t_1(N + n) + (N + n - 1)t_a + T_j(N + n - 1) + J^\pi(T - j, t - t_1))$$

$$= t_a + T_j - t_1$$

**Case 2:** $T_j \geq t_1$.

In this case, the two policies coincide after two actions (Figure B-1). Using marginal cost accounting, the cost difference can be written as:

$$J^\pi(T,t) - J^{\pi'}(T,t) = (N + n)t_a + t_1(N + n) + J^\pi(T,t - t_1) - (t_1(N + n) + (N + n - 1)t_a + J^\pi(T,t - t_1))$$

$$= t_a$$
On expectation, the cost difference between the two policies is equal to:

\[
J^\pi(T, t) - J^{\pi'}(T, t) = \Prob(T_j \leq t_1)(t_a + T_j - t_1) + \Prob(T_j \geq t_1)(t_a) \\
= t_a - \E[(t_1 - T_j)^+] \\
\geq t_a - \E[(\mu_j^a - T_j)^+] \\
= 0,
\]

where the latter holds since that \( t_1 \leq \mu_j^a \), which contradicts the uniqueness of the optimality of policy \( \pi \).

\[ \]
Case 1: $T_j \leq t_1$.

In this case, the testing delays under policy $\pi'$ are higher by $t_a$, since that in the first test action there is an additional job that is being delayed (compared with policy $\pi$). Any actions that do not involve jobs 1 and $j$ results in the same immediate costs under both policies (since that the duration of these actions are the same under both policies and the number of jobs being in the system is identical).

We denote by $N'$ and $n'$ the number of unknown and known jobs, respectively, when the second of the two jobs 1 and unknown $j$ are processed (these quantities are identical by the definition of the policy $\pi'$). The cost difference between the two policies can therefore be written as follows:

$$J^\pi(T, t) - J^{\pi'}(T, t) = t_1(N + n) + t_a(N + n - 1) + T_j(N' + n') - (t_a(N + n) + T_j(N + n) + t_1(N' + n'))$$

$$= -t_a + t_1(N + n - N' - n') - T_j(N + n - N' - n')$$

$$= -t_a + (t_1 - T_j)(N + n - N' - n')$$

$$\geq -t_a + (t_1 - T_j),$$

where we use the fact that $N + n > N' + n'$, since there is at least one less job in the system.

Case 2: $T_j \geq t_1$.

In this case, the only difference between the two policies is an addition testing cost $t_a$ for policy $\pi'$. 

167
On expectation, the cost difference between the two policies can be written as:

\[
J^\pi(T, t) - J^{\pi'}(T, t) \geq \text{Prob}(T_j \leq t_1)(-t_a + (t_1 - T_j)) + \text{Prob}(T_j > t_1)(-t_a)
\]

\[
= -(t_a - \mathbb{E}[(t_1 - T_j)^+])
\]

\[
\geq 0,
\]

where the first inequality is obtained from joining the two cases. The second inequality results from the fact that the function \(f(t) = t_a - \mathbb{E}[(t - T_j)^+]\) is decreasing in \(t\), and intersect zero at \(t = \mu_j\). Since that \(\mu_j \leq t_1\), the value of \(f(t_1)\) is negative, implying that \(-f(t_1)\) is non-negative, and the total costs under policy \(\pi\) are high than or equal to the costs under policy \(\pi'\).

\[\square\]

### B.4 Proof of Lemma 3.4.11

**Proof.** We prove by induction on the number of unknown jobs. When \(N = 1\), the lemma holds from the single class model.

Consider now the case where \(N > 1\). Assume by contradiction that the optimal policy \(\pi\) tests job \(j\) (if policy \(\pi\) does not test at all, it is exactly the policy PA). We compare the policy that test job \(j\) \((\pi_{test_j})\), with the policy PA. Using the induction hypothesis we know that after testing the unknown job \(j\), policy \(\pi\) will process all the jobs without further testing (since that the testing thresholds are non-decreasing in our actions and remain above \(\mu\)). Comparing these two policies under the non-homogeneous and homogeneous models are therefore the same (since that the remaining \(N - 1\) unknown job can be replaced by \(N - 1\) known jobs of duration \(\mu\) without affecting the objective). Using the homogeneous model, we know that processing all jobs is optimal, which means that under the non-homogeneous model, the following holds:

\[
J_{PA}(T, t) \leq J_{test_j}(T, t).
\]

\[\square\]
B.5 Proof of Lemma 3.4.13

Proof. The difference between the STP and PA policies is equal to that same difference under the homogeneous model in which there is a single unknown job $j$ (since that all other unknown jobs are never tested and essentially treated as known jobs with duration $\mu$). Equation 3.8 can therefore be obtained directly from Lemma 3.3.4.

B.6 Proof of Lemma 3.4.14

Proof. We prove the lemma in two steps:

1. We show that $J_{0,1}^{STP}(T,t) \leq J_{k,1}^{STP}(T,t)$,

2. We show that $J_{k,1}^{STP}(T,t) \leq J_{k,j}^{STP}(T,t)$.

We start with step 1:

$$J_{0,1}^{STP}([T_1, ..., T_N],[t_1, ..., t_n]) = J_{0,1}^{STP}([T_1], [t_1, ..., t_n] \cup \{\mu, \mu, ..., \mu\})^{N-1} \leq J_{k,1}^{STP}([T_1], [t_1, ..., t_n] \cup \{\mu, \mu, ..., \mu\})^{N-1} = J_{k,1}^{STP}([T_1, ..., T_N],[t_1, ..., t_n]),$$

where the equalities follow from the definition of policy $STP_{k,j}$ where job $j$ is the only job tested, and all other unknown jobs are processed without testing, which means that these unknown jobs are equivalent to known jobs of duration $\mu$ (easy to establish using a sample path argument). This reduces to the problem to a homogeneous model. The inequality then follows from the the fact that there are no short jobs in state $(T,t)$, and under the homogeneous model it is better to test before processing any of the known jobs (Section 3.3.3).
Step 2:

\[
J_{k,1}^{\text{STP}}([T_1, ..., T_N], [t_1, ..., t_n]) = J_{k,1}^{\text{STP}}([T_1], [t_1, ..., t_n] \cup \{\mu, \mu, ..., \mu\}) \\
\leq J_{k,j}^{\text{STP}}([T_j], [t_1, ..., t_n] \cup \{\mu, \mu, ..., \mu\}) \\
= J_{k,j}^{\text{STP}}([T_1, ..., T_N], [t_1, ..., t_n]),
\]

where the equalities hold for similar reasons to step 1. The inequality results from the convex order \(T_j \leq_{cx} T_1\), and the concavity of the function \(J_{k,j}^{\text{STP}}\) (which is an average of functions \(J_{P,A}\) that are concave in \(T_j\)). □

B.7 Proof of Lemma 3.4.15

Proof. Let \(\tau\) denote the realization of the tested job. \(s = ([T_1, ..., T_N], [t_1, ..., t_n])\) is the current state, and the next temporary state is \(([T_1, ..., T_N] \setminus T_j, [t_{\min(k,n_i)+1}, ..., t_n] \cup \{\tau\})\). At this state, all short jobs would be processed immediately after which we transition to state \(s'\).

From Lemma 3.4.13 we obtain that at state \(s\) the following holds:

\[
(N + n) t_a \geq \sum_{m=1}^{n_i} \mathbb{E}[(t_m - T_1)^+] + \sum_{m=n_i+1}^{n} \mathbb{E}[(T_1 - t_m)^+] + (N - 1)\mathbb{E}[(\mu - T_1)^+]. \tag{B.1}
\]

To prove the lemma we need to show that for any control \(k, j\) and any realization of the tested job the following inequality holds:

\[
J_{P,A}(s') \leq J_{k,j}^{\text{STP}}(s'). \tag{B.2}
\]

Case A: \(j = 1\). If unknown job 1 is tested then from Lemma 3.4.10 we know that \(k\) must be zero, and that \(\mu_a^2\) would be the lowest testing threshold in state \(s'\). When \(\mu \leq \mu_a^2\)
then processing all jobs is optimal and Equation [B.2] holds:

\[ J_{PA}(s') = J(s') \leq J_{STP}^{STP}(s') \]

If \( \mu^2 < \mu \) we denote by \( k_2 \) the number of known jobs in state \( s \) that fall the interval \( (\mu_1^a, \mu_2^a] \). Once we test unknown job 1, these jobs will become short and will be processed immediately before reaching state \( s' \).

The tested job \( \tau \) could be in one of three intervals:

1. \( \tau \leq \mu_2^a \),
2. \( \mu_2^a < \tau < \mu \),
3. \( \mu \leq \tau \).

\textbf{A1.} If \( \tau \leq \mu_2^a \), then the unknown job \( j \) will be processed immediately before reaching the next state \( s' \). The next state is therefore \( s' = ([T_2, ..., T_N], [t_{k_2+1}, ..., t_n]) \) and the condition \( J_{PA}(s') \leq J_{STP}^{STP}(s') \) can be written as follows:

\[ (N+n-k_2-1)a \geq \sum_{m=k_2+1}^{n_1} \mathbb{E}[(t_m - T_2)^+] + \sum_{m=n_1+1}^{n} \mathbb{E}[(T_2 - t_m)^+] + (N-2)\mathbb{E}[(\mu - T_2)^+] \]. \hspace{1cm} (B.3)

Using the convex order \( T_2 \leq_{cx} T_1 \) a sufficient condition is:

\[ (N+n-k_2-1)a \geq \sum_{m=k_2+1}^{n_1} \mathbb{E}[(t_m - T_1)^+] + \sum_{m=n_1+1}^{n} \mathbb{E}[(T_1 - t_m)^+] + (N-2)\mathbb{E}[(\mu - T_1)^+] \], \hspace{1cm} (B.4)

which is equivalent to:

\[ (N+n)a \geq \sum_{m=1}^{n_1} \mathbb{E}[(t_m - T_1)^+] + \sum_{m=n_1+1}^{n} \mathbb{E}[(T_1 - t_m)^+] + (N-1)\mathbb{E}[(\mu - T_1)^+] \]

\[ + (k_2 + 1)a - \sum_{m=1}^{k_2} \mathbb{E}[(t_m - T_1)^+] - \mathbb{E}[(\mu - T_1)^+] \]
\[
\begin{align*}
&= \sum_{m=1}^{n_i} \mathbb{E}[(t_m - T_1)^+] + \sum_{m=ni+1}^{n} \mathbb{E}[(T_1 - t_m)^+] + (N-1)\mathbb{E}[(\mu - T_1)^+] \\
&\quad + (k_2 + 1)\mathbb{E}[(\mu_1^a - T_1)^+] - \sum_{m=1}^{k_2} \mathbb{E}[(t_m - T_1)^+] - \mathbb{E}[(\mu - T_1)^+] ,
\end{align*}
\]

where we use simple arithmetic to obtain the first inequality, and substitute the value of \( t_a \) to obtain the second inequality. The last inequality holds since that expression \( E_2 \) is non-positive (\( \mu_1^a \) is smaller than \( \mu \) and any \( t_m \)), and expression \( E_1 \) is smaller than \((N+n)t_a\) (from Equation [B.1]).

**A2.** If \( \mu_1^a < \tau < \mu \), then the next state \( s' \) is \( s' = ([T_2, ..., T_N], [t_{k_2+1}, ..., t_n] \cup \{\tau\}) \). We want to show that:

\[
(N+n-k_2)t_a \geq \mathbb{E}[(\tau - T_2)^+] + \sum_{m=k_2+1}^{n_i} \mathbb{E}[(t_m - T_2)^+] + \sum_{m=ni+1}^{n} \mathbb{E}[(T_2 - t_m)^+] + (N-2)\mathbb{E}[(\mu - T_2)^+],
\]

for which a sufficient condition is:

\[
(N+n-k_2)t_a \geq \mathbb{E}[(\tau - T_1)^+] + \sum_{m=k_2+1}^{n_i} \mathbb{E}[(t_m - T_1)^+] + \sum_{m=ni+1}^{n} \mathbb{E}[(T_1 - t_m)^+] + (N-2)\mathbb{E}[(\mu - T_1)^+].
\]

(B.5)

Similarly to case A1, we can rewrite the latter to the following equivalent condition:

\[
(N + n)t_a \geq \sum_{m=1}^{n_i} \mathbb{E}[(t_m - T_1)^+] + \sum_{m=ni+1}^{n} \mathbb{E}[(T_1 - t_m)^+] + (N-1)\mathbb{E}[(\mu - T_1)^+] \\
+ k_2t_a + \mathbb{E}[(\tau - T_1)^+] - \sum_{m=1}^{k_2} \mathbb{E}[(t_m - T_1)^+] - \mathbb{E}[(\mu - T_1)^+],
\]

which holds for similar reasons.

**A3.** If \( \mu \leq \tau \), then the next state \( s' \) is \( s' = ([T_2, ..., T_N], [t_{k_2+1}, ..., t_n] \cup \{\tau\}) \) (similar to
A2). We want to show that:

\[(N+n-k_2)t_a \geq \mathbb{E}[(T_2-\tau)^+] + \sum_{m=k_2+1}^{n_i} \mathbb{E}[(t_m-T_2)^+] + \sum_{m=n_i+1}^{n} \mathbb{E}[(T_2-t_m)^+] + (N-2)\mathbb{E}[(\mu-T_2)^+], \tag{B.7}\]

for which a sufficient condition is:

\[(N+n-k_2)t_a \geq \mathbb{E}[(T_1-\tau)^+] + \sum_{m=k_2+1}^{n_i} \mathbb{E}[(t_m-T_1)^+] + \sum_{m=n_i+1}^{n} \mathbb{E}[(T_1-t_m)^+] + (N-2)\mathbb{E}[(\mu-T_1)^+]. \tag{B.8}\]

We can then rewrite the latter to the following equivalent condition:

\[(N+n)t_a \geq \sum_{m=1}^{n_i} \mathbb{E}[(t_m-T_1)^+] + \sum_{m=n_i+1}^{n} \mathbb{E}[(T_1-t_m)^+] + (N-1)\mathbb{E}[(\mu-T_1)^+] + \sum_{m=1}^{k_2} \mathbb{E}[(t_m-T_1)^+] - \mathbb{E}[(\mu-T_1)^+] + k_2t_a + \mathbb{E}[(T_1-\tau)^+] - \sum_{m=1}^{k_2} \mathbb{E}[(t_m-T_1)^+] - \mathbb{E}[(\mu-T_1)^+], \]

which holds for similar reasons.

**Case B:** \(j > 1\). In this case after testing the lowest testing threshold remains \(\mu_1^a\), which means that only \(\tau\) could become a short job. Unlike case A, here we could process known jobs before testing unknown job \(j\) (that is, \(k\) could be positive).

Once again, the tested job \(\tau\) could be in one of three intervals:

1. \(\tau \leq \mu_1^a\),
2. \(\mu_1^a < \tau < \mu\),
3. \(\mu \leq \tau\).

**B1.** When \(\tau \leq \mu_1^a\), the next state is \(s' = ([T_1, \ldots, T_N] \setminus \{T_j\}, [t_{k+1}, \ldots, t_n])\), and we want to show that:

\[(N+n-k-1)t_a \geq \sum_{m=k+1}^{n_i} \mathbb{E}[(t_m-T_1)^+] + \sum_{m=n_i+1}^{n} \mathbb{E}[(T_1-t_m)^+] + (N-2)\mathbb{E}[(\mu-T_1)^+] . \tag{B.9}\]

This follows from the exact same arguments as in case A1.

173
B2. If $\mu_1^a < \tau < \mu$, then the next state is $s' = ([T_1, ..., T_N] \setminus \{T_j\}, [t_{k+1}, ..., t_n] \cup \{\tau\})$, and we want to show that:

\[(N+n-k)t_a \geq \mathbb{E}[(\tau-T_1)^+] + \sum_{m=k+1}^{n} \mathbb{E}[(t_m-T_1)^+] + \sum_{m=n+1}^{n} \mathbb{E}[(T_1-t_m)^+] + (N-2)\mathbb{E}[(\mu-T_1)^+],\]

which follows from the same arguments used in proving case A2.

B3. If $\mu \leq \tau$, then the next state is $s' = ([T_1, ..., T_N] \setminus \{T_j\}, [t_{k+1}, ..., t_n] \cup \{\tau\})$. We want to show that:

\[(N+n-k)t_a \geq \mathbb{E}[(T_1-\tau)^+] + \sum_{m=k+1}^{n} \mathbb{E}[(t_m-T_1)^+] + \sum_{m=i+1}^{n} \mathbb{E}[(T_1-t_m)^+] + (N-2)\mathbb{E}[(\mu-T_1)^+],\]

which again follows from the same arguments used in proving case A3.

We showed that for any admissible control $k, j$ and a realization $\tau$ the lemma holds which concludes the proof. □

B.8 Proof of Theorem 3.4.16

Proof. Direction $\leftarrow$: Straightforward. If the cost of policy PA is higher than another policy, it is clearly not optimal to process all jobs.

Direction $\rightarrow$: We prove by induction on $N$, the number of unknown jobs in state $s = (T, t)$. The base case $N = 1$ is given to us by the homogeneous case.

Assume now that the induction hypothesis holds for any state with $N - 1$ unknown jobs. Assume by contradiction that there exists an optimal policy $\pi$ which performs at least one additional test. We denote the expected cost of policy $\pi$ by $J_\pi$. The optimality of policy $\pi$ implies that $J_\pi < J_{PA}$. Without loss of generality we may assume that at state $s$ policy $\pi$ would process $k$ jobs and then test unknown job $j$ before transitioning to a random state $s'$ which has $N - 1$ unknown jobs.
From Lemma 3.4.15 we know that for every state $s'$ the following holds:

$$J_{PA}(s') \leq J_{STP_{b,1}}(s').$$

Using the induction hypothesis, this implies that for every state $s'$ it is optimal to process all, and that policy $\pi$ must therefore be the policy $STP_{k,j}$. This is contradicts to the optimality of policy $\pi$ since that

$$J_{PA}(s) \leq J_{STP_{b,1}}(s) \leq J_{STP_{k,j}}(s) = J_{\pi}(s),$$

where the first inequality follows from the induction hypothesis, and the second inequality follows from Lemma 3.4.14.

B.9 Proof of Corollary 3.4.18

Proof. The proof follows directly from Lemma 3.4.13. Figure 3-7 illustrates the four intervals. When $x \leq \mu_1^a$ it is optimal to process $x$. Otherwise the difference $J_{STP}^{STP} - J_{PA}$ is a unimodal function that reaches its minimal value at $x = \mu$. Since it is optimal to test if and only if $J_{STP}^{STP} - J_{PA} \leq 0$, there is a convex range $C$ in which it is optimal to test. In the remaining two intervals $B,D$ it is optimal to process all jobs. Observe that some of the ranges may be empty sets.

B.10 Proof of Theorem 3.4.19

Proof. We prove the lemma by induction on the number of unknown jobs $N$. When $N = 1$ the lemma trivially holds. For $N > 1$, assume by contradiction that at some state $(T' + T_j + T_1, t)$ with at least two unknown jobs with random processing times $T_1$ and $T_j$, the optimal policy $\pi_j$ tests unknown job $T_j$ (where $j > 1$), and that testing the unknown job $T_1$ at the same state is suboptimal. We show that there exists a policy $\pi_1$ that tests unknown job $T_1$
at state \((T' + T_j + T_1, t)\) and which performs equally well or better than policy \(\pi_j\).

Let \(J_{\pi_j}\) and \(J_{\pi_1}\) denote the value functions under policies \(\pi_j\) and \(\pi_1\), respectively. To obtain a contradiction we must show that the following holds:

\[
J_{\pi_j}(T' + T_j + T_1, t) - J_{\pi_1}(T' + T_j + T_1, t) \geq 0
\]

\[
\Leftrightarrow (N + n)t_a + \mathbb{E}[J(T' + T_1, t + T_j)] - ((N + n)t_a + \mathbb{E}[J(T' + T_j, t + T_1)]) \geq 0
\]

\[
\Leftrightarrow \mathbb{E}[J(T' + T_1, t + T_j)] - \mathbb{E}[J(T' + T_j, t + T_1)] \geq 0,
\]

(B.12)

where the first equivalence is obtained from the DP formulation of Section 3.3.2 and the second equivalence is a result of canceling the term \((N + n)t_a\).

Proving that Equation (B.12) is satisfied consists of multiple steps. We therefore start with the proof overview:

1. We rewrite Equation (B.12) by adding and subtracting the value function applied to state \((T' + T_1, t + T_1)\):

\[
\mathbb{E}[J(T' + T_1, t + T_j)] - \mathbb{E}[J(T' + T_1, t + T_1)] + \mathbb{E}[J(T' + T_1, t + T_1)] - \mathbb{E}[J(T' + T_j, t + T_1)] \geq 0.
\]

(B.13)

This transforms Equation (B.12) into a summation of two differences between states that either share the same unknown jobs (the two leftmost terms in Equation (B.13)), or the same known jobs (the two rightmost terms in Equation (B.13). This is in contrast to Equation (B.12) that compares the value function applied to two states that are different both in terms of the known and unknown jobs).

2. We introduce the suboptimal policy “Y” which we use in Step 3 to create a lower bound for Equation (B.13). \(J_Y\) denotes the value function under policy Y.

3. We use the mean preserving local spread to write the convex order between the random variables \(T_1\) and \(T_j\) as a series of random variables that are in local spread (Theorem 3.2.2):

\[
T_j \overset{d}{=} Y_k \leq_{LS} Y_{k-1} \leq_{LS} \ldots \leq_{LS} Y_1 \overset{d}{=} T_1.
\]

176
Using the series of random variables and policy $Y$, we create a lower bound for Equation [B.13] in the form of a summation of the telescopic series:

$$
\sum_{l=1}^{k-1} \left( + \mathbb{E}J(T' + T_1, t + Y_{l+1}) - \mathbb{E}J(T' + T_1, t + Y_l) \right).
$$

Therefore, a sufficient condition for Equations [B.13] to hold is that every term in the telescopic series is non-negative. That is:

$$
\underbrace{\mathbb{E}J(T' + T_1, t + Y_{l+1}) - \mathbb{E}J(T' + T_1, t + Y_l)}_{L} \geq 0,
$$

$$
\underbrace{\mathbb{E}J_Y(T' + Y_l, t + T_1) - \mathbb{E}J_Y(T' + Y_{l+1}, t + T_1)}_{R} \geq 0. \tag{B.14}
$$

Note that Equation [B.14] is close to Equation [B.13] with main difference being that the general convex order relation $T_j \leq_{cx} T_1$ is replaced by the local spread $Y_{l+1} \leq_{LS} Y_l$. We denote by $L$ and $R$ the difference in the upper and lower rows in Equation [B.14] respectively (we later analyze each of the terms separately).

4. We compute Equation [B.14] by explicitly writing expectations using probability mass functions. Eventually, we rewrite Equation [B.14] using a function we denote as the concave function $\phi$:

$$
(\epsilon_1 + \epsilon_2) \left( \phi \left( \frac{\epsilon_1}{\epsilon_1 + \epsilon_2} y_{s-1} + \frac{\epsilon_2}{\epsilon_1 + \epsilon_2} y_{s+1} \right) - \frac{\epsilon_1}{\epsilon_1 + \epsilon_2} \phi(y_{s-1}) - \frac{\epsilon_2}{\epsilon_1 + \epsilon_2} \phi(y_{s+1}) \right) \geq 0.
$$

(where the function $\phi$ is defined in Equation [B.16].) The values $y_1, ..., y_z$ represent the support of the distributions $Y_l$ and $Y_{l+1}$, and $s$ is the focal point of the local spread $Y_l \leq_{LS} Y_{l+1}$ where $\epsilon_1(y_s - y_{s-1}) = \epsilon_2(y_{s+1} - y_s)$. A sufficient condition for the proof to hold is that the function $\phi$ is concave.

We continue to writing each of the steps in details.
Step 1: Rewriting Equation \textbf{B.12}

Equation \textbf{B.12} can be written as follows:

\[
\begin{align*}
\mathbb{E}[J(T' + T_1, t + T_j)] - \mathbb{E}[J(T' + T_j, t + T_1)] \\
= & \mathbb{E}J(T' + T_1, t + T_j) - (\mathbb{E}J(T' + T_1, t + T_1) - \mathbb{E}J(T' + T_1, t + T_1)) - \mathbb{E}J(T' + T_j, t + T_1) \\
= & \mathbb{E}J(T' + T_1, t + T_j) - \mathbb{E}J(T' + T_1, t + T_1) + \mathbb{E}J(T' + T_1, t + T_1) - \mathbb{E}J(T' + T_j, t + T_1),
\end{align*}
\]

(B.13)

where in the first equality, we add and subtract the term \(\mathbb{E}J(T' + T_1, t + T_1)\), and by rearranging the terms we obtain the last expression.

Note that by adding and subtracting a common term, we transformed the difference in Equation \textbf{B.12} that compared the value function applied to states with different known and unknown jobs, to a comparison between states the either share the same unknown jobs (the two leftmost terms in Equation \textbf{B.13}), or the same known jobs (the two rightmost terms in Equation \textbf{B.13}).

Step 2: Policy Y.

We define policy Y as follows:

**Definition B.10.1.** At each state \((T' + Y, t)\) (with \(Y\) denoting any unknown job, such as \(T_j\), \(T_1\), or possibly a different random variable), policy Y tests unknown job Y at state \((T' + Y, t)\), if the optimal policy tests at state \((T' + T_1, t)\), and otherwise stops (i.e., processes all jobs). If policy Y tests, it later imitates the optimal policy.

We make the following observations about policy Y:

1. Policy Y is defined with respect to the unknown jobs in \(T'\).

2. Using the induction hypothesis, we know that at state \((T' + T_1, t)\) the optimal policy may test only \(T_1\) as it is highest in convex order.
3. At state \((T' + Y_l, t)\), policy \(Y\) does similar thing for all values of \(l\). The policy either stops testing (regardless of the value of \(l\)), or tests job \(Y_l\), in which case it transitions to state \((T', t + Y_l)\).

4. At state \((T' + T_1, t)\) policy \(Y\) is in fact the optimal policy.

Step 3: Using the Mean Preserving Local Spread to lower bound Equation \(\text{B.13}\) by the summation of a telescopic series.

Using \(T_j \leq c_1 T_1\) and Theorem 3.2.2, we know that there exists a sequence of random variables \(Y_1, Y_2, ..., Y_k\) that satisfy: (1) \(Y_1 \overset{d}{=} T_1\); (2) \(Y_k \overset{d}{=} T_j\); (3) \(Y_k \leq_{LS} Y_{k-1} \leq_{LS} ... \leq_{LS} Y_1\). We can write a lower bound for Equation \(\text{B.13}\) as follows:

\[
\mathbb{E}J(T' + T_1, t + T_1) - \mathbb{E}J(T' + T_1, t + T_1) + \mathbb{E}J(T' + T_1, t + T_1) - \mathbb{E}J(T' + T_j, t + T_1) \\
= \mathbb{E}J(T' + T_1, t + Y_k) - \mathbb{E}J(T' + T_1, t + Y_1) + \mathbb{E}J_Y(T' + Y_1, t + T_1) - \mathbb{E}J(T' + Y_k, t + T_1) \\
\geq \sum_{l=1}^{k-1} (\mathbb{E}J(T' + T_1, t + Y_{l+1}) - \mathbb{E}J(T' + T_1, t + Y_l)) \\
+ \sum_{l=1}^{k-1} (\mathbb{E}J_Y(T' + Y_l, t + T_1) - \mathbb{E}J_Y(T' + Y_{l+1}, t + T_1)) \\
\]

where the first row repeats Equation \(\text{B.13}\). In the first equality we replace the random variables \(T_1\) and \(T_j\) with \(Y_1\) and \(Y_k\), respectively, and use the optimality of policy \(Y\) in state \((T' + T_1, t + T_1)\) (fourth observation about policy \(Y\)) to replace \(\mathbb{E}J(T' + T_1, t + T_1)\) with \(\mathbb{E}J_Y(T' + T_1, t + T_1)\). The inequality follows from the suboptimality of policy \(Y\) in state \((T' + Y_k, t + T_1)\). In the last equality we write the preceding expression as a sum of a telescopic series.

A sufficient condition for Equation \(\text{B.12}\) to hold is that for every \(1 \leq l \leq k - 1\) the
following holds:

\[
\begin{align*}
&L = \mathbb{E}J(T' + T_1, t + Y_{l+1}) - \mathbb{E}J(T' + T_1, t + Y_{l}) \\
&+ \mathbb{E}J_Y(T' + Y, t + T_1) - \mathbb{E}J_Y(T' + Y_{l+1}, t + T_1) \\
&\geq 0. \quad \text{(B.14)}
\end{align*}
\]

**Step 4: An explicit computation of Equation (B.14)**

We now delve into Equation (B.14) and explicitly write the expectations using probability distributions. We use the fact that the random variables \(Y_l\) and \(Y_{l+1}\) are in local spread, which means that their distributions are identical in all but three points. Specifically, from Definition 3.2.1, we know that there exists a support \(y_1 < y_2 < \ldots < y_z\), a focal point \(s\) (\(1 < s < z\)), and probabilities \(\epsilon_1\) and \(\epsilon_2\), which satisfy:

- \(\text{Prob}(Y_l = y_{s-1}) = \text{Prob}(Y_{l+1} = y_{s-1}) + \epsilon_1\).
- \(\text{Prob}(Y_l = y_{s+1}) = \text{Prob}(Y_{l+1} = y_{s+1}) + \epsilon_2\).
- \(\text{Prob}(Y_l = y_s) = \text{Prob}(Y_{l+1} = y_s) - \epsilon_1 - \epsilon_2 = 0\).
- \(\text{Prob}(Y_l = y_i) = \text{Prob}(Y_{l+1} = y_i), \text{for } i \notin \{s-1, s, s+1\}\).
- \(\epsilon_1(y_s - y_{s-1}) = \epsilon_2(y_{s+1} - y_s), \text{ or equivalently:}\)
  \[y_s = \frac{\epsilon_1}{(\epsilon_1 + \epsilon_2)} y_{s-1} + \frac{\epsilon_2}{(\epsilon_1 + \epsilon_2)} y_{s+1}.\] \hspace{1cm} (B.15)

The reader is referred to Figure 3-3 for an intuitive illustration. In addition, we use \(v_m\) to denote values from the support of the random variables \(T_1\).

We can now express the expressions \(L\) and \(R\) using the support and probabilities of the distributions \(Y_l\) and \(Y_{l+1}\).
Step 4.1: Simplifying the term $L$.

The term $L$ represents the difference in the expected value function of two states that have the same unknown jobs and and all but one different known job. Using the definition of local spread, we can write the difference in expectations as follows:

$$
L = \mathbb{E}J(T' + T_1, t + Y_{i+1}) - \mathbb{E}J(T' + T_1, t + Y_i)
$$

$$
= \sum_{m=1}^{z} (\text{Prob}(Y_{i+1} = y_m)J(T' + T_1, t + y_m) - \text{Prob}(Y_i = y_m)J(T' + T_1, t + y_m))
$$

$$
= \text{Prob}(Y_{i+1} = y_{s-1})J(T' + T_1, t + y_{s-1}) - \text{Prob}(Y_i = y_{s-1})J(T' + T_1, t + y_{s-1})
$$

$$
+ \text{Prob}(Y_{i+1} = y_s)J(T' + T_1, t + y_s) - \text{Prob}(Y_i = y_s)J(T' + T_1, t + y_s)
$$

$$
+ \text{Prob}(Y_{i+1} = y_{s+1})J(T' + T_1, t + y_{s+1}) - \text{Prob}(Y_i = y_{s+1})J(T' + T_1, t + y_{s+1})
$$

$$
= -\epsilon_1 J(T' + T_1, t + y_{s-1})
$$

$$
+ (\epsilon_1 + \epsilon_2)J(T' + T_1, t + y_s)
$$

$$
- \epsilon_2 J(T' + T_1, t + y_{s+1}),
$$

where the first equality is from the definition of the term $L$, and in the second equality we use the definition of the expectation operator. In the third equality, we eliminate identical terms from the summation, and in the fourth equality we use the local spread to cancel out identical probabilities. We obtain the last equality by rearranging terms.

Note that while it is easy to verify the term $L$ is non-negative (from the local (and convex) order $Y_{i+1} \leq_{cx} Y_i$), our goal is to show that $L + R$ is non-negative. Moreover, the term $R$ is non-positive (for similar reasons), which is why we need to compute and sum to two terms.

Step 4.2: Simplifying the term $R$.

The term $R$ is defined to be the difference in the expected value function of policy $Y$ at two similar states that contain the same known jobs, and unknown jobs, with the exception of a single unknown job. From the definition of policy $Y$ (and observation 3 about policy...
Y), we know that for each realization of $T_1$, policy Y selects the same control at states $(T' + Y_{i+1}, t + T_1)$ and $(T' + Y_i, t + T_1)$. Specifically, if in state $(T' + T_1, t + T_1)$, the optimal policy stops, then policy Y stops at both states. Alternatively, if the optimal policy tests in state $(T' + T_1, t + T_1)$ then policy Y tests unknown jobs $Y_{i+1}$ and $Y_i$ in state $(T' + Y_{i+1}, t + T_1)$ and $(T' + Y_i, t + T_1)$, respectively.

Using Corollary 3.4.18 we know that in state $(T' + T_1, t + T_1)$ there are four ranges of values for any realization $v_m$ of $T_1$ where different controls are optimal:

1. $v_m \in A$: Process $v_m$,

2. $v_m \in B$: Process all,

3. $v_m \in C$: Process and test,

4. $v_m \in D$: Process all.

We can therefore write the term $R$ as follows:

$$R = \mathbb{E}J_Y(T' + Y_i, t + T_1) - \mathbb{E}J_Y(T' + Y_{i+1}, t + T_1)$$

$$= \sum_{v_m} Prob(T_1 = v_m) \left( J_Y(T' + Y_i, t + v_m) - J_Y(T' + Y_{i+1}, t + v_m) \right)$$

$$= \sum_{v_m \in A} Prob(T_1 = v_m) \left( J_Y(T' + Y_i, t + v_m) - J_Y(T' + Y_{i+1}, t + v_m) \right)$$

$$+ \sum_{v_m \in B,D} Prob(T_1 = v_m) \left( J_Y(T' + Y_i, t + v_m) - J_Y(T' + Y_{i+1}, t + v_m) \right)$$

$$+ \sum_{v_m \in C} Prob(T_1 = v_m) \left( J_Y(T' + Y_i, t + v_m) - J_Y(T' + Y_{i+1}, t + v_m) \right)$$

$$= \sum_{v_m \in A} Prob(T_1 = v_m) \left( v_m(N + n) + J_Y(T' + Y_i, t) - v_m(N + n) - J_Y(T' + Y_{i+1}, t) \right)$$

$$+ \sum_{v_m \in B,D} Prob(T_1 = v_m) \left( J_Y(T' + Y_i, t + v_m) - J_Y(T' + Y_{i+1}, t + v_m) \right)$$

$$= \sum_{v_m \in A} Prob(T_1 = v_m) \left( (N + n - 1)t_a + \mathbb{E}J(T', t + Y_i) \right)$$

$$- \sum_{v_m \in A} Prob(T_1 = v_m) \left( (N + n - 1)t_a + \mathbb{E}J(T', t + Y_{i+1}) \right)$$

182
\[
\begin{align*}
&+ \sum_{v_m \in C} \text{Prob}(T_1 = v_m) ((N + n)t_a + \mathbb{E}J(T', t + v_m + Y_i)) \\
&- \sum_{v_m \in C} \text{Prob}(T_1 = v_m) ((N + n)t_a + \mathbb{E}J(T', t + v_m + Y_{i+1})) \\
&= (5) \sum_{v_m \in A} \text{Prob}(T_1 = v_m) (\mathbb{E}J(T', t + Y_i) - \mathbb{E}J(T', t + Y_{i+1})) \\
&+ \sum_{v_m \in C} \text{Prob}(T_1 = v_m) (\mathbb{E}J(T', t + v_m + Y_i) - \mathbb{E}J(T', t + v_m + Y_{i+1})) \\
&= (6) \sum_{v_m \in A} \text{Prob}(T_1 = v_m) \begin{pmatrix}
\epsilon_1 J(T', t + y_{s-1}) \\
+\epsilon_2 J(T', t + y_{s+1}) \\
-(\epsilon_1 + \epsilon_2) J(T', t + y_s)
\end{pmatrix} \\
&+ \sum_{v_m \in C} \text{Prob}(T_1 = v_m) \begin{pmatrix}
\epsilon_1 J(T', t + v_m + y_{s-1}) \\
+\epsilon_2 J(T', t + v_m + y_{s+1}) \\
-(\epsilon_1 + \epsilon_2) J(T', t + v_m + y_s)
\end{pmatrix}
\end{align*}
\]

where equality (1) is obtained from the definition of expectation, and in equality (2) the expectation is derived from the definition of the sets \(A, B, C, D\). For equality (3), we observe that if policy \(Y\) stops (the sets \(B, D\)), then the value function is the same for states \((T' + Y_i, t + v_m)\) and \((T' + Y_{i+1}, t + v_m)\), as the different unknown jobs are processed without testing, which means that on expectation their contribution to the objective is identical. Under the set \(A\), both policies process the job \(v_m\) which incurs the same immediate costs as there are exactly the same number of jobs in the system.

Equality (4) is obtained by incurring the testing costs and transitioning to the next state, where policy \(Y\) is the same as the optimal policy (therefore we remove the subscript \(Y\)). Reordering the terms takes us to equality (5). Similar to the development of term \(L\), we observe that the two rows in equality (5) represent differences in the expected value functions in states that share the same unknown jobs and known jobs, except for one known job that is different in local spread. We can then eliminate terms from that difference which takes us to equality (6).

We are now ready to sum the terms \(L\) and \(R\).
Step 4.3: The summation of the terms L and R.

The summation of $L + R$ can be written as follows:

\[
(\epsilon_1 + \epsilon_2)J(T' + T_1, t + y_s) - \epsilon_1 J(T' + T_1, t + y_{s-1}) - \epsilon_2 J(T' + T_1, t + y_{s+1}) + \sum_{v_m \in A} \text{Prob}(T_1 = v_m) \begin{pmatrix}
\epsilon_1 J(T', t + y_{s-1}) \\
+ \epsilon_2 J(T', t + y_{s+1}) \\
- (\epsilon_1 + \epsilon_2) J(T', t + y_s)
\end{pmatrix}
\]

\[
+ \sum_{v_m \in C} \text{Prob}(T_1 = v_m) \begin{pmatrix}
\epsilon_1 J(T', t + v_m + y_{s-1}) \\
+ \epsilon_2 J(T', t + v_m + y_{s+1}) \\
- (\epsilon_1 + \epsilon_2) J(T', t + v_m + y_s)
\end{pmatrix}
\]

\[
= (\epsilon_1 + \epsilon_2) \begin{pmatrix}
J(T' + T_1, t + y_s) \\
- \sum_{v_m \in A} \text{Prob}(T_1 = v_m) J(T', t + y_{s-1}) \\
- \sum_{v_m \in C} \text{Prob}(T_1 = v_m) J(T', t + y_{s-1})
\end{pmatrix}
\]

\[
- \epsilon_1 \begin{pmatrix}
J(T' + T_1, t + y_{s+1}) \\
- \sum_{v_m \in A} \text{Prob}(T_1 = v_m) J(T', t + y_{s+1}) \\
- \sum_{v_m \in C} \text{Prob}(T_1 = v_m) J(T', t + y_{s+1})
\end{pmatrix}
\]

\[
- \epsilon_2 \begin{pmatrix}
J(T' + T_1, t + y_{s-1}) \\
- \sum_{v_m \in A} \text{Prob}(T_1 = v_m) J(T', t + y_{s-1}) \\
- \sum_{v_m \in C} \text{Prob}(T_1 = v_m) J(T', t + y_{s-1})
\end{pmatrix}
\]

\[
= (\epsilon_1 + \epsilon_2) \phi(y_s) - \epsilon_1 \phi(y_{s-1}) - \epsilon_2 \phi(y_{s+1})
\]

\[
= (\epsilon_1 + \epsilon_2) \left( \phi \left( \frac{\epsilon_1}{\epsilon_1 + \epsilon_2} y_{s-1} + \frac{\epsilon_2}{\epsilon_1 + \epsilon_2} y_{s+1} \right) - \frac{\epsilon_1}{\epsilon_1 + \epsilon_2} \phi(y_{s-1}) - \frac{\epsilon_2}{\epsilon_1 + \epsilon_2} \phi(y_{s+1}) \right),
\]

where the function $\phi(x)$ is defined as follows:

\[
\phi(x) = J(T' + T_1, t + x) - \sum_{v_m \in A} \text{Prob}(T_1 = v_m) J(T', t + x) - \sum_{v_m \in C} \text{Prob}(T_1 = v_m) J(T', t + v_m + x).
\]

(B.16)

To prove that the summation is non-negative, we need to show that $\phi(x)$ is a concave function.
Step 4.4: The function $\phi(x)$ is concave.

We evaluate the function $\phi$ for each of the intervals A,B,C, and D defined by Corollary 3.4.18:

A. $x$ is short,

B. $x$ is not short, and at state $(T' + T_1, t + x)$ it is optimal to process all jobs,

C. $x$ is not short, and at state $(T' + T_1, t + x)$ it is optimal to test,

D. $x$ is not short, and at state $(T' + T_1, t + x)$ it is optimal to process all jobs.

The function $\phi(x)$ is continuous, therefore, in order to establish concavity it is sufficient to show that the slope of $\phi(x)$ is decreasing in $x$. For the purpose of obtaining the derivative of $\phi(x)$ in $x$, we are only interested in terms that depend on $x$, therefore we will ignore all constants and simply write those as part of a constant term denoted as $\text{const}$.

Step 4.4.1: Evaluating the function $\phi$ in interval A.

$x$ is short and processed immediately. The function $\phi$ in the interval A can be written as:

$$
\phi(x_A) = J(T' + T_1, t + x) - \sum_{v_m \in A} \text{Prob}(T_1 = v_m) J(T', t + x) - \sum_{v_m \in C} \text{Prob}(T_1 = v_m) J(T', t + v_m + x)
$$

$$
= x(N + n) + J(T' + T_1, t) - \sum_{v_m \in A} \text{Prob}(T_1 = v_m) (x(N + n - 1) + J(T', t))
$$

$$
- \sum_{v_m \in C} \text{Prob}(T_1 = v_m) (x(N + n) + J(T', t + v_m))
$$

$$
= x \left( N + n - \sum_{v_m \in A} \text{Prob}(T_1 = v_m)(N + n - 1) - \sum_{v_m \in C} \text{Prob}(T_1 = v_m)(N + n) \right)
$$

$$
+ J(T' + T_1, t) - \sum_{v_m \in A} \text{Prob}(T_1 = v_m) J(T', t) - \sum_{v_m \in C} \text{Prob}(T_1 = v_m) J(T', t + v_m)
$$

$$
= x \left( N + n - 1 \right) \left( 1 - \sum_{v_m \in A,C} \text{Prob}(T_1 = v_m) \right) + 1 - \sum_{v_m \in C} \text{Prob}(T_1 = v_m)
$$

$$
+ \text{const}
$$

$$
= x \left( N + n - 1 \right) \sum_{v_m \in B,D} \text{Prob}(T_1 = v_m) + \sum_{v_m \in A,B,D} \text{Prob}(T_1 = v_m) + \text{const}
$$
\[ \begin{align*}
&= \text{const} + \sum_{v_m \in B, D} \text{Prob}(T_1 = v_m) ((N - 1)x) \\
&+ (nx) \sum_{v_m \in B, D} \text{Prob}(T_1 = v_m) + x \sum_{v_m \in A, B, D} \text{Prob}(T_1 = v_m). 
\end{align*} \]

where the first equality is by the definition of \( \phi(x) \), and the second equality is a direct result of applying the control \( \text{Process}_k \) in interval A \((x \text{ is a short job})\). We obtain the next two equalities by rearranging terms and substituting all values that do not depend on \( x \) with const. In the last two transitions, we use the fact that the sum of probabilities equals 1: \( \sum_{v_m \in A, B, C, D} \text{Prob}(T_1 = v_m) = 1. \)

**Step 4.4.2: Evaluating the function \( \phi \) in intervals B and D.**

In both intervals B and D the optimal control at state \((T' + T_1, t + x)\) is to process all jobs. Corollary \( 3.4.17 \) implies that in states \((T', t + x)\), and \((T', t + x + v_m)\) it is also optimal to process all jobs.

We can write the function \( \phi \) in the intervals B and D as follows:

\[ \phi(x) = J(T' + T_1, t + x) - \sum_{v_m \in A} \text{Prob}(T_1 = v_m)J(T', t + x) \]

\[ - \sum_{v_m \in C} \text{Prob}(T_1 = v_m)J(T', t + v_m + x) \]

\[ = J_{P_A}(T' + T_1, t + x) - \sum_{v_m \in A} \text{Prob}(T_1 = v_m)J_{P_A}(T', t + x) \]

\[ - \sum_{v_m \in C} \text{Prob}(T_1 = v_m)J_{P_A}(T', t + v_m + x) \]

\[ = \text{const} + x + (N - 1) \min(\mu, x) + \sum_{i=1}^{n} \min(t_i, x) \]

\[ - \sum_{v_m \in A} \text{Prob}(T_1 = v_m) \left( \text{const} + x + (N - 2) \min(\mu, x) + \sum_{i=1}^{n} \min(t_i, x) \right) \]

\[ - \sum_{v_m \in C} \text{Prob}(T_1 = v_m) \left( \text{const} + x + (N - 2) \min(\mu, x) + \sum_{i=1}^{n} \min(t_i, x) + \min(v_m, x) \right) \]

\[ = \text{const} + \min(\mu, x) + \left( x + (N - 2) \min(\mu, x) + \sum_{i=1}^{n} \min(t_i, x) \right) \]
\[
- \sum_{v_m \in A,C} \text{Prob}(T_1 = v_m) \left( x + (N - 2) \min(\mu, x) + \sum_{i=1}^{n} \min(t_i, x) \right) \\
- \sum_{v_m \in C} \text{Prob}(T_1 = v_m) \min(v_m, x) \\
= \text{const} + (x + (N - 2) \min(\mu, x)) \sum_{v_m \in B,D} \text{Prob}(T_1 = v_m) \\
+ \left( \sum_{i=1}^{n} \min(t_i, x) \right) \sum_{v_m \in B,D} \text{Prob}(T_1 = v_m) + \left( \min(\mu, x) - \sum_{v_m \in C} \text{Prob}(T_1 = v_m) \min(v_m, x) \right)
\]

where the first equality is the definition of \( \phi(x) \), and the second equality results from the definition of intervals B and D where it is optimal to process all jobs. For the third equality we use the fact that the sum of completion times can be written as the ordering cost of all jobs; the ordering cost associated with job \( x \) and any other job is equal to the minimum of \( x \) and the duration of the respective job. The ordering cost of any pair of jobs that does not include \( x \) does not depend on the duration \( x \) and therefore can be written as part of \( \text{const} \). Using arithmetic we obtain the consecutive equalities.

In interval B, \( x < \mu \), and also \( x < v_m \) for any \( v_m \in C \) (Figure 3-7). The function \( \phi_B(x) \) can be written as:

\[
\phi_B(x) = \text{const} + (x + (N - 2)x) \sum_{v_m \in B,D} \text{Prob}(T_1 = v_m) \\
+ \left( \sum_{i=1}^{n} \min(t_i, x) \right) \sum_{v_m \in B,D} \text{Prob}(T_1 = v_m) + \left( x - \sum_{v_m \in C} \text{Prob}(T_1 = v_m) x \right) \\
= \text{const} + \sum_{v_m \in B,D} \text{Prob}(T_1 = v_m) ((N - 1)x) \\
+ \left( \sum_{i=1}^{n} \min(t_i, x) \right) \sum_{v_m \in B,D} \text{Prob}(T_1 = v_m) + x \sum_{v_m \in A,B,D} \text{Prob}(T_1 = v_m),
\]

where in the second equality we simply rearrange the terms.
In interval D, \( \mu < x \) and \( v_m < x \) for \( x \in C \), and the function \( \phi_D(x) \) can be written as:

\[
\phi_D(x) = \text{const} + (x + (N-2)\mu) \sum_{v_m \in B,D} \text{Prob}(T_1 = v_m) \\
+ \left( \sum_{i=1}^{n} \min(t_i, x) \right) \sum_{v_m \in B,D} \text{Prob}(T_1 = v_m) + \left( \mu - \sum_{v_m \in C} \text{Prob}(T_1 = v_m)v_m \right) \\
= \text{const} \\
+ \left( \sum_{i=1}^{n} \min(t_i, x) \right) \sum_{v_m \in B,D} \text{Prob}(T_1 = v_m) + x \sum_{v_m \in B,D} \text{Prob}(T_1 = v_m),
\]

where in the second equality we simply rearrange the terms.

**Step 4.4.3: Evaluating the function \( \phi \) in interval C.**

For \( x \in C \), \( x \) is non-short and it is optimal to test in state \((T' + T_1, t + x)\). We can write the function \( \phi_C \) as follows:

\[
\phi_C(x) = J_{\text{Test}}(T' + T_1, t + x) \\
- \sum_{v_m \in A} \text{Prob}(T_1 = v_m)J(T', t + x) - \sum_{v_m \in C} \text{Prob}(T_1 = v_m)J(T', t + v_m + x) \\
= \text{const} + \sum_{v_m \in A,B,C,D} \text{Prob}(T_1 = v_m)J(T', t + x + v_m) \\
- \sum_{v_m \in A} \text{Prob}(T_1 = v_m)J(T', t + x) - \sum_{v_m \in C} \text{Prob}(T_1 = v_m)J(T', t + v_m + x) \\
= \text{const} + \sum_{v_m \in B,D} \text{Prob}(T_1 = v_m)J(T', t + x + v_m) \\
+ \sum_{v_m \in A} \text{Prob}(T_1 = v_m)J(T', t + x + v_m) + \sum_{v_m \in C} \text{Prob}(T_1 = v_m)J(T', t + x + v_m) \\
- \sum_{v_m \in A} \text{Prob}(T_1 = v_m)J(T', t + x) - \sum_{v_m \in C} \text{Prob}(T_1 = v_m)J(T', t + v_m + x) \\
= \text{const} + \sum_{v_m \in B,D} \text{Prob}(T_1 = v_m)J(T', t + x + v_m) \\
+ \sum_{v_m \in A} \text{Prob}(T_1 = v_m)(v_m(N + n) + J(T', t + x)) + \sum_{v_m \in C} \text{Prob}(T_1 = v_m)J(T', t + x + v_m) \\
- \sum_{v_m \in A} \text{Prob}(T_1 = v_m)J(T', t + x) - \sum_{v_m \in C} \text{Prob}(T_1 = v_m)J(T', t + v_m + x)
\]
\[
\begin{align*}
&= \text{const} \\
&+ \sum_{v_m \in B, D} \Pr(T_1 = v_m) \left( \text{const} + x + \min(v_m, x) + \sum_{i=1}^{n} \min(t_i, x) + \min(\mu, x)(N - 2) \right) \\
&= \text{const} + \sum_{v_m \in B, D} \Pr(T_1 = v_m) (\min(\mu, x)(N - 2) + \min(v_m, x)) \\
&+ \left( \sum_{i=1}^{n} \min(t_i, x) \right) \sum_{v_m \in B, D} \Pr(T_1 = v_m) + x \sum_{v_m \in B, D} \Pr(T_1 = v_m),
\end{align*}
\]

where in the first equality we use the fact the in interval C it is optimal to test unknown job 1. In the second equality we split the support according to the realization \( v_m \) of \( T_1 \), and in the third equality we use the definition of interval A where realizations \( v_m \) are immediately processed. We eliminate and rearrange terms to obtain the next two equalities.

**Step 4.4.4: Comparing the derivative of \( \phi(x) \) in intervals A,B,C, and D.**

We now summarize the function \( \phi(x) \) in each of the intervals A, B, C, D:

\[
\begin{align*}
\phi_A(x) &= \text{const} + \sum_{v_m \in B, D} \Pr(T_1 = v_m) ((N - 1)x) \\
&+ \left( \sum_{i=1}^{n} \min(t_i, x) \right) \sum_{v_m \in B, D} \Pr(T_1 = v_m) + x \sum_{v_m \in A, B, D} \Pr(T_1 = v_m), \\
\phi_B(x) &= \text{const} + \sum_{v_m \in B, D} \Pr(T_1 = v_m) ((N - 1)x) \\
&+ \left( \sum_{i=1}^{n} \min(t_i, x) \right) \sum_{v_m \in B, D} \Pr(T_1 = v_m) + x \sum_{v_m \in A, B, D} \Pr(T_1 = v_m), \\
\phi_C(x) &= \text{const} + \sum_{v_m \in B, D} \Pr(T_1 = v_m) (\min(\mu, x)(N - 2) + \min(v_m, x))
\end{align*}
\]
where we mark by (1), (2), and (3) the terms that constitute the functions \(\phi_A(x), \phi_B(x), \phi_C(x),\) and \(\phi_D(x).\) We first observe that the functions are concave in \(x\) (\(\text{min}\) is a concave function), which means that the derivative of each of the functions is non-increasing. Moreover, by comparing the terms (1), (2), and (3) in each of the functions, it is easy to see that \(\phi'_A(x) \geq \phi'_B(x) \geq \phi'_C(x) \geq \phi'_D(x).\) The function \(\phi\) is continuous, and has derivative that is non-increasing in \(x\), which means that it is also concave. This implies that Equation B.12 holds, and that policy \(\pi_1\) achieves the same or lower objective than policy \(\pi_j.\) That is, for every policy that tests unknown job \(j > 1\) there exists a policy which tests unknown job 1 and achieves the same or lower objective.

\[\phi_D(x) = \text{const}\]

\begin{align*}
+ \left( \sum_{i=1}^{n} \min(t_i, x) \right) \sum_{v_m \in B, D} \text{Prob}(T_1 = v_m) + x \sum_{v_m \in B, D} \text{Prob}(T_1 = v_m), \quad (2) \\
+ \left( \sum_{i=1}^{n} \min(t_i, x) \right) \sum_{v_m \in B, D} \text{Prob}(T_1 = v_m) + x \sum_{v_m \in B, D} \text{Prob}(T_1 = v_m), \quad (3)
\end{align*}

B.11 Proof of Theorem 3.4.20

Proof. The external loop of the algorithm (Step 1) is an immediate result of Lemma 3.4.11. Lemma 3.4.9 implies that we always test short jobs before any tests (Step 2). Theorem 3.4.19 then states that we may only consider unknown job for testing which means that we either test unknown job 1 immediately (Lemma 3.4.10 prohibits testing unknown job 1 after any processing), or simply process all jobs. Theorem 3.4.16 then perfectly characterize which of the two possibilities are optimal using the condition specified at Step 3.
Appendix C

Proofs for Chapter 4

C.1 Proof of Lemma 4.6.7

Proof. Using Definition 4.3.5,

\[
\Delta_k(\bar{W}, \bar{w}) = J^\text{my}_{k}(\bar{W}, \bar{w}) - J^\text{stop}(\bar{W}, \bar{w}) \geq J^\text{my}_i(\bar{W}, \bar{w}) - J^\text{stop}(\bar{W}, \bar{w}) = \Delta_i(\bar{W}, \bar{w})
\]

where the inequality follows from the definition of the convex order and the fact that \( J^\text{my}_i(\bar{W}, \bar{w}) \) can be written as the expectation of the following expression:

\[-c + \varphi(\mathbb{E}[\bar{W} - W_i], \bar{w} + W_i),\]

which is convex function of \( W_i \) (Lemma 4.3.1). \(\square\)

C.2 Proof of Theorem 4.6.8

Proof. We prove the theorem by induction on \( k \). When \( k = 1 \) there is a single unknown coefficient and the theorem trivially holds. For the case \( k > 1 \), recall that the myopic gain from testing coefficient \( k \) is the highest (Lemma 4.6.7). This implies that if \( \Delta_k \leq 0 \), then for every \( j \) the myopic gain is non-positive (\( \Delta_j \leq 0 \)). According to Theorem 4.4.3 stopping is
optimal, in which case, the myopic policy is also optimal. We therefore only need to consider the case where \( \Delta_k > 0 \), in which testing is optimal. We show that there exists an optimal policy that tests coefficient \( k \).

Assume by contradiction that at some state testing coefficient \( k \) is not optimal, and that there exists an optimal policy that tests coefficient \( j \). Let \((\bar{W} + W_j + W_k, \bar{w})\) denote any state that has two or more unknown jobs, where \( \bar{W} \) is the set of untested jobs that are neither \( W_k \) nor \( W_j \) (\( \bar{W} \) could be the empty set). Moreover, let \( \pi_j \) denote the optimal policy that starts by testing coefficient \( j \). We construct a policy \( \pi_k \) which starts by testing coefficient \( k \), and show that the value function under policy \( \pi_k \) is equal to or higher than the value function under policy \( \pi_j \). That is, we show that the following holds:

\[
J_{\pi_j} (\bar{W} + W_j + W_k, \bar{w}) - J_{\pi_k} (\bar{W} + W_j + W_k, \bar{w}) \leq 0. \tag{C.1}
\]

Before proving Equation \( \text{(C.1)} \), we briefly outline the remainder of the proof:

1. We define a sub-optimal policy denoted as policy \( Y \).

2. We reformulate Equation \( \text{(C.1)} \) by adding and subtracting the term \( \mathbb{E}J_Y (\bar{W} + W_k, \bar{w} + W_k) \) (where \( J_Y \) denotes the value function under policy \( Y \)). This simplifies the analysis by allowing us to compare expressions that are more similar to each other. The new expression is given by Equation \( \text{(C.2)} \).

3. We then use the mean preserving local spread (Theorem \( \text{4.6.6} \)) to create an upper bound for Equation \( \text{(C.2)} \) using a sum of a telescopic series. Proving that every term in the telescopic series is non-positive is a sufficient condition for the proof to hold. This essentially reduces the problem from arbitrary convex orders to a mean preserving local spread. This term is given by Equation \( \text{(C.4)} \).

4. We explicitly write Equation \( \text{(C.4)} \). Since this term rather complex, we divide it to two parts denoted by ‘L’ and ‘R’. First we derive the term ‘L’, then derive the term ‘R’, and then sum the two. We present this summation using a function we denote.
by \( \phi \), and argue that in order to complete the proof it is sufficient to show that the function \( \phi \) is convex.

5. We establish the convexity of the function \( \phi \).

**Step 1: Policy “Y”**.

We start by defining policy Y as follows:

**Definition C.2.1.** At any state \((\bar{W}' + Y, \bar{w})\) (with \( Y \) denoting any unknown coefficient, such as \( W_j, W_k, \) or possibly a different random variable), policy Y tests coefficient \( Y \) at state \((\bar{W}' + Y, \bar{w})\), if the optimal policy tests at state \((\bar{W}' + W_k, \bar{w})\), and otherwise stops. If policy Y tests, it later imitates the optimal policy.

We make the following observations about policy Y:

1. Policy Y is defined with respect to the unknown coefficients in \( \bar{W}' \).

2. Using the induction hypothesis, we know that at state \((\bar{W}' + W_k, \bar{w})\) the optimal policy may test only \( W_k \) as it highest in convex order.

3. Under policy Y the same control is selected at states: \((\bar{W}' + Y_1, \bar{w})\) and \((\bar{W}' + Y_2, \bar{w})\) (where \( Y_1 \) and \( Y_2 \) correspond to two different random variables). At both states the policy either stops, or tests coefficients \( Y_1 \) and \( Y_2 \), respectively. This means that when testing, the two states transition to states \((\bar{W}', \bar{w} + Y_1)\) and \((\bar{W}', \bar{w} + Y_2)\), respectively.

4. At state \((\bar{W}' + W_k, \bar{w})\) policy Y is in fact the optimal policy.

We next use policy Y to reformulate the sufficient condition for the optimality of policy \( \pi_k \) given by Equation (C.1).
Step 2: Reformulating Equation (C.1).

Using the DP formulation of Equation (4.18), we can write Equation (C.1) as follows:

\[
J_{\pi_j} (\bar{W}' + W_j + W_k, \bar{w}) - J_{\pi_k} (\bar{W}' + W_j + W_k, \bar{w}) = -c + EJ (\bar{W}' + W_k, \bar{w} + W_j) - (-c + EJ (\bar{W}' + W_j, \bar{w} + W_k))
\]

\[
= EJ (\bar{W}' + W_k, \bar{w} + W_j) - EJ (\bar{W}' + W_j, \bar{w} + W_k)
\]

\[
- (EJ_Y (\bar{W}' + W_k, \bar{w} + W_k) - EJ_Y (\bar{W}' + W_k, \bar{w} + W_k))
\]

\[
= EJ (\bar{W}' + W_k, \bar{w} + W_j) - EJ_Y (\bar{W}' + W_k, \bar{w} + W_k)
\]

\[
+ EJ_Y (\bar{W}' + W_k, \bar{w} + W_k) - EJ (\bar{W}' + W_j, \bar{w} + W_k),
\]

(C.2)

where the first equality follow directly from Equation (4.18), and in the second equality we cancel the common term \(c\). In the third equality we introduce two terms that sum to zero, where the subscript \(Y\) denotes that the value function is computed under policy \(Y\). By rearranging the terms we obtain the last equality.

Observe that unlike Equation (C.1) in which we subtract the value function of two states that are different in both unknown and known coefficients, in Equation (C.2) we subtract terms that share either the same known or unknown coefficients (corresponding to the lower and upper rows in Equation (C.2)).

Step 3: Bounding Equation (C.2) by a sum of a telescopic series.

Using Theorem 4.6.6 and the convex order \(W_j \leq_{cx} W_k\), we know that there exists a series of random variables \(Y_1, ..., Y_m\) such that: (1) \(Y_1 \leq_{LS} ... \leq_{LS} Y_m\); (2) \(Y_1 \overset{d}{=} W_j\); and (3) \(Y_m \overset{d}{=} W_k\). We can then express Equation (C.2) using the series \(Y_1, ..., Y_m\):

\[
EJ (\bar{W}' + W_k, \bar{w} + W_j) - EJ_Y (\bar{W}' + W_k, \bar{w} + W_k)
\]

\[
+ EJ_Y (\bar{W}' + W_k, \bar{w} + W_k) - EJ (\bar{W}' + W_j, \bar{w} + W_k)
\]

194
\[
\begin{align*}
&= \mathbb{E}J(\tilde{W}'+W_k, \tilde{\omega}+Y_1) - \mathbb{E}J_Y(\tilde{W}'+W_k, \tilde{\omega}+Y_m) \\
&+ \mathbb{E}J_Y(\tilde{W}'+Y_m, \tilde{\omega}+W_k) - \mathbb{E}J(\tilde{W}'+Y_1, \tilde{\omega}+W_k) \\
&= \mathbb{E}J(\tilde{W}'+W_k, \tilde{\omega}+Y_1) - \mathbb{E}J(\tilde{W}'+W_k, \tilde{\omega}+Y_m) \\
&+ \mathbb{E}J_Y(\tilde{W}'+Y_m, \tilde{\omega}+W_k) - \mathbb{E}J_Y(\tilde{W}'+Y_1, \tilde{\omega}+W_k) \\
&\leq \mathbb{E}J(\tilde{W}'+W_k, \tilde{\omega}+Y_1) - \mathbb{E}J(\tilde{W}'+W_k, \tilde{\omega}+Y_m) \\
&+ \mathbb{E}J_Y(\tilde{W}'+Y_m, \tilde{\omega}+W_k) - \mathbb{E}J_Y(\tilde{W}'+Y_1, \tilde{\omega}+W_k) \\
&= \sum_{l=1}^{m-1} (\mathbb{E}J(\tilde{W}'+W_k, \tilde{\omega}+Y_l) - \mathbb{E}J(\tilde{W}'+W_k, \tilde{\omega}+Y_{l+1})) \\
&+ \sum_{l=1}^{m-1} (\mathbb{E}J_Y(\tilde{W}'+Y_{l+1}, \tilde{\omega}+W_k) - \mathbb{E}J_Y(\tilde{W}'+Y_l, \tilde{\omega}+W_k)), \quad (C.3)
\end{align*}
\]

where in the first equality we substitute \(W_j\) and \(W_k\) with \(Y_1\) and \(Y_m\), respectively. In the second equality, we remove the subscript \(Y\) because policy \(Y\) and \(\pi_j\) coincide (by definition of policy \(Y\)). In the first inequality, we use the fact that policy \(Y\) is sub-optimal, which we then rewrite as a sum of a telescopic series to obtain Equation (C.3).

Therefore, a sufficient condition for equations (C.1) and (C.2) to hold is that every term in the telescopic series given by Equation (C.3) is non-positive. That is, for every \(W_j \leq_{CS} Y_l \leq_{LS} Y_{l+1} \leq_{CS} W_k\) the following inequality holds:

\[
\underbrace{\mathbb{E}J(\tilde{W}'+W_k, \tilde{\omega}+Y_l) - \mathbb{E}J(\tilde{W}'+W_k, \tilde{\omega}+Y_{l+1})}_L \\
+ \underbrace{\mathbb{E}J_Y(\tilde{W}'+Y_{l+1}, \tilde{\omega}+W_k) - \mathbb{E}J_Y(\tilde{W}'+Y_l, \tilde{\omega}+W_k)}_R \leq 0, \quad (C.4)
\]

where \(L\) denotes the summation in the upper row, and \(R\) denotes the summation in the lower row.

**Step 4: Using local spread to simplify Equation (C.4).**

We now delve into Equation (C.4) and explicitly write the expectations using probability distributions. We use the fact that the random variables \(Y_l\) and \(Y_{l+1}\) are in local spread,
which means that their distributions is identical in all but three points in the support of their distributions. Specifically, from Definition 4.6.5, we know that there exists a support $y_1 < y_2 < \ldots < y_z$, a focal point $s$ ($1 < s < z$), and probabilities $\epsilon_1$ and $\epsilon_2$, which satisfy:

- $\text{Prob}(Y_{i+1} = y_{s-1}) = \text{Prob}(Y_i = y_{s-1}) + \epsilon_1,$
- $\text{Prob}(Y_{i+1} = y_{s+1}) = \text{Prob}(Y_i = y_{s+1}) + \epsilon_2,$
- $\text{Prob}(Y_{i+1} = y_s) = \text{Prob}(Y_i = y_s) - \epsilon_1 - \epsilon_2 = 0$,
- $\text{Prob}(Y_{i+1} = y_i) = \text{Prob}(Y_i = y_i)$, for $i \notin \{s - 1, s, s + 1\}$, and
- $\epsilon_1(y_s - y_{s-1}) = \epsilon_2(y_{s+1} - y_s)$, or equivalently:

$$y_s = \frac{\epsilon_1}{(\epsilon_1 + \epsilon_2)} y_{s-1} + \frac{\epsilon_2}{(\epsilon_1 + \epsilon_2)} y_{s+1}. \quad (C.5)$$

The reader is referred to Figure 4-5 for an intuitive illustration. In addition, we use $v_m$ to denote values from the support of the random variables $W_k$.

We can now express the expressions $L$ and $R$ using the support and probabilities of the distributions $Y_i$ and $Y_{i+1}$.

**Step 4.1: The term L.**

The term $L$ represents the difference in the expected value function of two states that have the same unknown jobs and and all but one different known job. Using the definition of local spread, we can write the difference in expectations as follows:

$$L = \mathbb{E}J(\bar{W} + W_k, \bar{w} + Y_i) - \mathbb{E}J(\bar{W} + W_k, \bar{w} + Y_{i+1})$$

$$= \sum_{m=1}^z \text{Prob}(Y_i = y_m)J(\bar{W} + W_k, \bar{w} + y_m)$$

$$- \sum_{m=1}^z \text{Prob}(Y_{i+1} = y_m)J(\bar{W} + W_k, \bar{w} + y_m)$$

196
\[
\sum_{m=1}^{z} (\text{Prob}(Y_i = y_m) - \text{Prob}(Y_{i+1} = y_m)) J (\bar{W}' + W_k, \bar{w} + y_m)
\]

\[
= -\epsilon_1 J (\bar{W}' + W_k, \bar{w} + y_{s-1}) + (\epsilon_1 + \epsilon_2) J (\bar{W}' + W_k, \bar{w} + y_s) - \epsilon_2 J (\bar{W}' + W_k, \bar{w} + y_{s+1}),
\]

\[\text{(C.6)}\]

where the first equality is the definition of L, and in the second equality we explicitly write expectations. The third equality follows from simple arithmetics, and in the fourth equality, we eliminate identical terms using the local spread and the fact that for \(i \notin \{s - 1, s, s + 1\}\): \(\text{Prob}(Y_i = y_i) = \text{Prob}(Y_{i+1} = y_i)\), and for \(i \in \{s - 1, s, s + 1\}\), we can express the difference in the probability mass function using \(\epsilon_1\) and \(\epsilon_2\).

**Step 4.2: The term \(R\).**

The term \(R\) is the difference of the value functions under policy Y applied to two very similar states that share the same known jobs, and are different only by a single unknown job. Under policy Y, at both states, the same control is chosen according to what policy \(\pi_j\) does at state \((\bar{W}' + W_k, \bar{w} + W_k)\). We therefore divide the support of the random variable \(W_k\) into two sets based on whether policy \(\pi_j\) stops (the set \(\mathcal{S}_1\)) or tests (the set \(\mathcal{S}_2\)) at state \((\bar{W}' + W_k, \bar{w} + W_k)\):

\[\mathcal{S}_1 = \{m : J_{\pi_j} (\bar{W}' + W_k, \bar{w} + v_m) = \varphi_P (\bar{W}' + W_k, \bar{w} + v_m)\}\]

and

\[\mathcal{S}_2 = \{m : J_{\pi_j} (\bar{W}' + W_k, \bar{w} + v_m) = -c + \mathbb{E}_k [J_{\pi_j} (\bar{W}' + W_k, \bar{w} + v_m)]\}\]

We can then rewrite the term \(R\) as follows:

\[
R = \mathbb{E} J_Y (\bar{W}' + Y_{i+1}, \bar{w} + W_k) - \mathbb{E} J_Y (\bar{W}' + Y_i, \bar{w} + W_k)
\]

\[
= \sum_{m} \text{Prob}(W_k = v_m) (J_Y (\bar{W}' + Y_{i+1}, \bar{w} + v_m) - J_Y (\bar{W}' + Y_i, \bar{w} + v_m))
\]
\[
\begin{align*}
&= (2) \sum_{m \in S_1} \text{Prob}(W_k = v_m) \left( J_Y \left( \bar{W}' + Y_{l+1}, \bar{w} + v_m \right) - J_Y \left( \bar{W}' + Y_l, \bar{w} + v_m \right) \right) \\
&\quad + \sum_{m \in S_2} \text{Prob}(W_k = v_m) \left( J_Y \left( \bar{W}' + Y_{l+1}, \bar{w} + v_m \right) - J_Y \left( \bar{W}' + Y_l, \bar{w} + v_m \right) \right) \\
&= (3) \sum_{m \in S_1} \text{Prob}(W_k = v_m) \left( \varphi \left( \mathbb{E}[\bar{W}' + Y_{l+1}, \bar{w} + v_m] \right) - \varphi \left( \mathbb{E}[\bar{W}' + Y_l, \bar{w} + v_m] \right) \right) \\
&\quad + \sum_{m \in S_2} \text{Prob}(W_k = v_m) \left( J_Y \left( \bar{W}' + Y_{l+1}, \bar{w} + v_m \right) - J_Y \left( \bar{W}' + Y_l, \bar{w} + v_m \right) \right) \\
&= (4) \sum_{m \in S_1} \text{Prob}(W_k = v_m) \left( -c + \mathbb{E} J \left( \bar{W}', \bar{w} + v_m + Y_{l+1} \right) \right) \\
&\quad + \sum_{m \in S_2} -\text{Prob}(W_k = v_m) \left( -c + \mathbb{E} J \left( \bar{W}', \bar{w} + v_m + Y_l \right) \right) \\
&= (5) \sum_{m \in S_2} \text{Prob}(W_k = v_m) \left( \mathbb{E} J \left( \bar{W}', \bar{w} + v_m + Y_{l+1} \right) - \mathbb{E} J \left( \bar{W}', \bar{w} + v_m + Y_l \right) \right) \\
&= (6) \sum_{m \in S_2} \text{Prob}(W_k = v_m) \left( \mathbb{E} J \left( \bar{W}', \bar{w} + v_m + Y_{l+1} \right) - \mathbb{E} J \left( \bar{W}', \bar{w} + v_m + Y_l \right) \right) \\
&\quad - \sum_{m \in S_2} \text{Prob}(W_k = v_m) \left( \epsilon_1 + \epsilon_2 \right) J \left( \bar{W}', \bar{w} + v_m + y_s \right) \\
&\quad + \sum_{m \in S_2} \text{Prob}(W_k = v_m) \epsilon_2 J \left( \bar{W}', \bar{w} + v_m + y_{s+1} \right) . \quad (C.7)
\end{align*}
\]

In equality (1) we explicitly write the expectation with respect to \( W_k \), and in equality (2) we divide the support to the sets \( S_1 \), and \( S_2 \). From the definition of the set \( S_1 \), we know that policy \( Y \) stops when \( v_m \in S_1 \), which allows us to replace the value function \( J_Y \) with the stopping cost \( \varphi \), and obtain equality (3). Using the symmetry of the function \( \varphi \), and the convex order which implies that \( \mathbb{E}[Y_l] = \mathbb{E}[Y_{l+1}] \), we obtain that the difference in the summation of over the set \( S_1 \) is equal to zero, which brings us to equality (4). In equality (5), we use the fact that when \( v_m \in S_2 \) testing is optimal, which we simplify to obtain equality (6). Finally, and similarly to the derivation of the term \( L \), we can write the expectations using probability mass functions, and cancel identical terms using the similarity of the distributions of \( Y_l \) and \( Y_{l+1} \).
Now that we derived the terms for L and R, we can sum them to obtain Equation (C.4), which we try to prove is non-positive.

**Step 4.3: The term \( L + R \).**

We can sum the derived terms for L and R (equations C.6 and C.7 respectively):

\[
L + R = -\epsilon_1 J (\bar{W}' + W_k, \bar{w} + y_{s-1}) + (\epsilon_1 + \epsilon_2) J (\bar{W}' + W_k, \bar{w} + y_s) \\
- \epsilon_2 J (\bar{W}' + W_k, \bar{w} + y_{s+1}) \\
+ \sum_{m \in S_2} \text{Prob}(W_k = v_m) \epsilon_1 J (\bar{W}', \bar{w} + v_m + y_{s-1}) \\
- \sum_{m \in S_2} \text{ Prob}(W_k = v_m) (\epsilon_1 + \epsilon_2) J (\bar{W}', \bar{w} + v_m + y_s) \\
+ \sum_{m \in S_2} \text{Prob}(W_k = v_m) \epsilon_2 J (\bar{W}', \bar{w} + v_m + y_{s+1}) \\
= (\epsilon_1 + \epsilon_2) \left( J (\bar{W}' + W_k, \bar{w} + y_s) - \sum_{m \in S_2} \text{Prob}(W_k = v_m) J (\bar{W}', \bar{w} + v_m + y_s) \right) \\
- \epsilon_1 \left( J (\bar{W}' + W_k, \bar{w} + y_{s-1}) - \sum_{m \in S_2} \text{ Prob}(W_k = v_m) J (\bar{W}', \bar{w} + v_m + y_{s-1}) \right) \\
- \epsilon_2 \left( J (\bar{W}' + W_k, \bar{w} + y_{s+1}) - \sum_{m \in S_3} \text{Prob}(W_k = v_m) J (\bar{W}', \bar{w} + v_m + y_{s+1}) \right) \\
= (\epsilon_1 + \epsilon_2) \phi(y_s) - \epsilon_1 \phi(y_{s-1}) - \epsilon_2 \phi(y_{s+1}) \\
= (\epsilon_1 + \epsilon_2) \left( \phi(y_s) - \frac{\epsilon_1}{\epsilon_1 + \epsilon_2} \phi(y_{s-1}) - \frac{\epsilon_2}{\epsilon_1 + \epsilon_2} \phi(y_{s+1}) \right) \quad (C.8)
\]

where the function \( \phi(y) \) is defined as follows:

\[
\phi(y) = J (\bar{W}' + W_k, \bar{w} + y) - \sum_{m \in S_2} \text{ Prob}(W_k = v_m) J (\bar{W}', \bar{w} + v_m + y).
\]
This implies that the sufficient condition for the optimality of policy $\pi_k$ (Equation (C.4)), can be written as follows:

$$\phi(y_s) - \frac{\epsilon_1}{\epsilon_1 + \epsilon_2} \phi(y_{s-1}) - \frac{\epsilon_2}{\epsilon_1 + \epsilon_2} \phi(y_{s+1}) \leq 0. \quad (C.9)$$

From the definition of local spread and Equation C.5, we know that $y_s$ is a convex combination of the points $y_{s-1}$ and $y_{s+1}$:

$$y_s = \frac{\epsilon_1}{(\epsilon_1 + \epsilon_2)} y_{s-1} + \frac{\epsilon_2}{(\epsilon_1 + \epsilon_2)} y_{s+1},$$

and therefore to prove that Equation [C.9] holds, it is sufficient to show that the function $\phi(y)$ is convex in $y$. Observe that the convexity of $\phi(y)$ does not trivially holds since as it is the difference of two convex functions.

We next show that the function $\phi(y)$ is indeed convex and complete our proof.

**Step 5: Establishing the convexity of the function $\phi(y)$.**

To prove that the function $\phi(y)$ is convex, we calculate the derivative of $\phi(y)$ in the three consecutive intervals, denoted as intervals 1, 2, and 3. We show that the function $\phi(y)$ is continuous, piece-wise linear, whose derivative is piece-wise constant and increasing, and therefore the function $\phi(y)$ is convex.

We start by defining the three intervals. Corollary 4.4.5 implies that there are three possible ranges of values for $y$:

1. $y < v_1$: it is optimal to stop at state $(\bar{W}' + W_k, \bar{w} + y)$
2. $v_1 \leq y \leq v_2$: it is optimal to test at state $(\bar{W}' + W_k, \bar{w} + y)$
3. $v_2 < y$: it is optimal to stop at state $(\bar{W}' + W_k, \bar{w} + y)$

Recall that the set $S_1$ is defined to be the values of the support $y_1, ..., y_z$ at which stopping is optimal, and therefore $S_1$ is equal to the union of intervals 1 and 3. Similarly, the set $S_2$ is the equal to interval 2.

200
We now calculate the derivative in each of the intervals.

**Interval 1.** By definition of interval 1, it is optimal to stop at state \((\tilde{W}' + W_k, \tilde{w} + y)\), which implies that it is also optimal to stop at any state \((\tilde{W}' + y + v_m)\) (Corollary 4.4.4).

We can therefore write the function \(\phi\) in interval 1 as follows:

\[
\phi_1(y) = J_{\text{Stop}} (\tilde{W}' + W_k, \tilde{w} + y) - \sum_{m \in S_2} \text{Prob}(W_k = v_m) J_{\text{Stop}} (\tilde{W}' + \tilde{w} + v_m + y) \quad (C.10)
\]

We are interested in \(\phi'_1(y)\) in the range \(y < v_1\):

\[
\phi'_1(y) = \frac{d}{dy} \left( J_{\text{Stop}} (\tilde{W}' + W_k, \tilde{w} + y) - \sum_{m \in S_2} \text{Prob}(W_k = v_m) J_{\text{Stop}} (\tilde{W}' + \tilde{w} + v_m + y) \right)
\]

\[
\begin{align*}
&\overset{(1)}{=} \frac{d}{dy} \left( \phi (\mathbb{E}[\tilde{W}' + W_k], \tilde{w} + y) - \sum_{m \in S_2} \text{Prob}(W_k = v_m) \phi (\mathbb{E}[\tilde{W}'], \tilde{w} + v_m + y) \right) \\
&\overset{(2)}{=} \frac{d}{dy} \left( \phi (\mathbb{E}[\tilde{W}'], \tilde{w} + y + \mu) - \sum_{m \in S_2} \text{Prob}(W_k = v_m) \phi (\mathbb{E}[\tilde{W}'], \tilde{w} + v_m + y) \right) \\
&\overset{(3)}{=} \frac{d}{dy} \left( \phi (\mathbb{E}[\tilde{W}'], \tilde{w} + y + v_1) - \sum_{m \in S_2} \text{Prob}(W_k = v_m) \phi (\mathbb{E}[\tilde{W}'], \tilde{w} + v_1 + y) \right) \\
&\overset{(4)}{=} \frac{d}{dy} \left( \phi (\mathbb{E}[\tilde{W}'], \tilde{w} + v_1 + y) \left( 1 - \sum_{m \in S_2} \text{Prob}(W_k = v_m) \right) \right) \\
&\overset{(5)}{=} \frac{d}{dy} \phi (\mathbb{E}[\tilde{W}'], \tilde{w} + v_1 + y) \left( \sum_{m \in S_1} \text{Prob}(W_k = v_m) \right) \\
&\overset{(6)}{=} \frac{d}{dy} \phi (\mathbb{E}[\tilde{W}'], \tilde{w} + v_1 + y) \sum_{m: v_m < v_1} \text{Prob}(W_k = v_m) \\
&\quad + \frac{d}{dy} \phi (\mathbb{E}[\tilde{W}'], \tilde{w} + v_1 + y) \sum_{m: v_m > v_2} \text{Prob}(W_k = v_m) \\
&\overset{(7)}{=} \frac{d}{dy} \phi (\mathbb{E}[\tilde{W}'], \tilde{w} + v_1 + y) \sum_{m: v_m < v_1} \text{Prob}(W_k = v_m) \\
&\quad + \frac{d}{dy} \phi (\mathbb{E}[\tilde{W}'], \tilde{w} + v_2 + y) \sum_{m: v_m > v_2} \text{Prob}(W_k = v_m) \quad (C.11)
\]

where the first transition we use DP formulation (Equation 4.18), and in transition (2) we...
use symmetry. In transition (3) we use the fact that \( \varphi \) is solved by the greedy algorithm, and that in the resulting solution, the value of the variable corresponding to coefficient \( y \), is not affected by changes in coefficients with higher values. Since that in interval 1 \( y < v_1 \), changing any coefficient that is larger or equal to \( v_1 \) to \( v_1 \) does not change the value of the variable corresponding to \( y \), and therefore does not affect the derivative of \( \varphi \) with respect to \( y \). This allows us to replace \( \mu \) by \( v_1 \). Similarly, we may replace any coefficient \( v_m \) in the set \( S_2 \) (which values are greater than \( v_1 \), by definition) with \( v_1 \).

With simple arithmetic we obtain transitions (4) and (5). In transition (6) we split the values of \( v_m \) in the set \( S_1 \) to values that are smaller than \( v_1 \), and to values that are higher than \( v_2 \). Finally with similar arguments to transition (3) we obtain transition (7).

**Interval 2.** In this interval, it is optimal to perform a test while being at state \((\bar{W} + W_k, \bar{w} + y)\). We can write the derivative of \( \phi(y) \) in this interval as follows:

\[
\frac{d}{dy} \left( J_{Test} (\bar{W}' + W_k, \bar{w} + y) - \sum_{m \in S_2} \text{Prob}(W_k = v_m)J (\bar{W}', \bar{w} + v_m + y) \right)
\]

\[
= \frac{d}{dy} \left( -c + \sum_{m \in S_1 \cup S_2} \text{Prob}(W_k = v_m)J (\bar{W}', \bar{w} + v_m + y) - \sum_{m \in S_2} \text{Prob}(W_k = v_m)J (\bar{W}', \bar{w} + v_m + y) \right)
\]

\[
= \frac{d}{dy} \left( \sum_{m \in S_1} \text{Prob}(W_k = v_m)J (\bar{W}', \bar{w} + v_m + y) \right)
\]

\[
= \frac{d}{dy} \left( \sum_{m \in S_1} \text{Prob}(W_k = v_m)J_{Stop} (\bar{W}', \bar{w} + v_m + y) \right)
\]

\[
= \frac{d}{dy} \left( \sum_{m \in S_1} \text{Prob}(W_k = v_m)\varphi (E[\bar{W}'], \bar{w} + v_m + y) \right)
\]

\[
= \frac{d}{dy} \left( \sum_{m: v_m < v_1} \text{Prob}(W_k = v_m)\varphi (E[\bar{W}'], \bar{w} + v_m + y) \right) + \frac{d}{dy} \left( \sum_{m: v_m > v_2} \text{Prob}(W_k = v_m)\varphi (E[\bar{W}'], \bar{w} + v_m + y) \right)
\]

\[
= \frac{d}{dy} \left( \sum_{m: v_m < v_1} \text{Prob}(W_k = v_m)\varphi (E[\bar{W}'], \bar{w} + v_1 + y) \right)
\]
\[ + \frac{d}{dy} \left( \sum_{m:v_m > v_2} \text{Prob}(W_k = v_m) \varphi \left( \mathbb{E}[W'], \bar{w} + v_2 + y \right) \right) \]

\[ = \frac{d}{dy} \varphi \left( \mathbb{E}[W'], \bar{w} + v_1 + y \right) \sum_{m:v_m < v_1} \text{Prob}(W_k = v_m) + \frac{d}{dy} \varphi \left( \mathbb{E}[W'], \bar{w} + v_2 + y \right) \sum_{m:v_m > v_2} \text{Prob}(W_k = v_m) \quad (C.12) \]

where equality (1) follows from the definition of interval 2. In (2) we use the definition of expectation, and in (3) we eliminate similar terms and the constant \(c\) which does not affect the derivative. In (4) we use the fact that when \(v_m \in S_1\) it is optimal to stop at state \((\bar{w}' + W_k, \bar{w} + v_m)\), which implies that it is also optimal to stop at state \((\bar{w}', \bar{w} + y + v_m)\) (Corollary 4.4.4). We can then rewrite the last expression using the function \(\varphi\) (transition (5)). In (6) we split the summation to two, and in (7) we use Lemma 4.3.1 and the fact that \(y\) is in interval 2 which allows us to change any coefficients that are outside of interval 2, that is, the values of \(v_m\). In (8) we simply rearrange the terms.

**Interval 3.** Similarly to to interval 1, it is optimal to stop at states \((\bar{w}' + W_k, \bar{w} + y)\) and \((\bar{w}', \bar{w} + y + v_m)\) when \(y > v_2\), and we can write the derivative of the function \(\phi(y)\) in this interval as follows:

\[
\phi_3'(y) = \frac{d}{dy} \left( J_{\text{Stop}} \left( \mathbb{E}[W' + W_k], \bar{w} + y \right) - \sum_{m \in S_2} \text{Prob}(W_k = v_m) J_{\text{Stop}} \left( \mathbb{E}[W'], \bar{w} + v_m + y \right) \right) \\
= \frac{d}{dy} \left( \varphi \left( \mathbb{E}[W' + W_k], \bar{w} + y \right) - \sum_{m \in S_2} \text{Prob}(W_k = v_m) \varphi \left( \mathbb{E}[W'], \bar{w} + v_m + y \right) \right) \\
= \frac{d}{dy} \left( \varphi \left( \mathbb{E}[W'], \bar{w} + y + \mu \right) - \sum_{m \in S_2} \text{Prob}(W_k = v_m) \varphi \left( \mathbb{E}[W'], \bar{w} + v_m + y \right) \right) \\
= \frac{d}{dy} \left( \varphi \left( \mathbb{E}[W'], \bar{w} + y + v_1 \right) - \sum_{m \in S_2} \text{Prob}(W_k = v_m) \varphi \left( \mathbb{E}[W'], \bar{w} + v_1 + y \right) \right) \\
= \frac{d}{dy} \left( \varphi \left( \mathbb{E}[W'], \bar{w} + y + v_1 \right) \left( 1 - \sum_{m \in S_2} \text{Prob}(W_k = v_m) \right) \right) \\
= \frac{d}{dy} \varphi \left( \mathbb{E}[W'], \bar{w} + v_1 + y \right) \left( \sum_{m \in S_1} \text{Prob}(W_k = v_m) \right)
\]

203
\[ = \frac{d}{dy} \phi(\mathbb{E}[\bar{W}'], \bar{w} + v_1 + y) \sum_{m: v_m < v_1} \text{Prob}(W_k = v_m) + \frac{d}{dy} \phi(\mathbb{E}[\bar{W}'], \bar{w} + v_2 + y) \sum_{m: v_m > v_2} \text{Prob}(W_k = v_m), \] 

(C.13)

where all the transitions follow for similar arguments as in interval 1.

We see that the derivative of \( \phi(y) \) is the same in all three intervals (equations C.11, C.12, and C.13 are identical). Moreover, this is the derivative of a non-decreasing piece-wise linear and convex function (Lemma 4.3.1), which means that \( \phi'(y) \) is piece-wise constant and non-decreasing. The continuity of \( \phi(y) \), and the fact that its derivative is piece-wise constant, and non-decreasing, implies that \( \phi(y) \) is convex in \( y \).

The convexity of \( \phi(y) \), implies that the inequalities in equations C.9, C.4, and C.1 hold, and therefore policy \( \pi_k \) is optimal, which contradicts the suboptimality of policy \( \pi_k \). \( \square \)
Bibliography


Levi, Retsef, Thomas Magnanti, Yaron Shaposhnik. 2014. Scheduling with testing.

Li, Dong, Kevin D. Glazebrook. 2010. An approximate dynamic programming approach to the development of heuristics for the scheduling of impatient jobs in a clearing system. *Naval Research Logistics (NRL)* 57(3) 225–236.


Sun, Zhankun, Nilay Tanik Argon, Serhan Ziya. 2014. Priority scheduling of jobs with unknown types.


