

ABSTRACT

KHORRAMFAR, RAHMAN. Computational Optimization for Hierarchical Decision Making Problems. (Under the direction of Dr. Osman Ozaltin).

We consider the hierarchical decision making (HDM) which is a game-theoretical decision process in which single or multiple decision-makers (DMs) solve sequential interdependent optimization problems over single or multiple planning periods. Depending on the relationship between decision-makers, the mathematical structure of the various decision problems, and the sequence in which decisions are made, HDM models can be classified as bilevel programs, stochastic programs, and robust optimization. This dissertation addresses a multistage stochastic programming models and two bilevel models. We start with stochastic programming where decision-makers cooperatively work toward a common objective, which can be viewed as a single decision maker making sequential decisions. We propose a reformulation for general multistage stochastic programming problems that are particularly amenable to parallel computing. The reformulation establishes bounds by decomposing the scenario tree into independent subtrees. Unlike traditional decomposition methods that solely focus on the scenarios, the presented approach allows a more general decomposition pattern that, as evident by the computational results, enables the method to deal with much larger instances.

The last two chapters analyze bilevel mixed-integer models in the context of product transitions. Unlike stochastic programming, bilevel programming deals with problems where decision-makers are organized in a hierarchy with their own objectives. We model the hierarchical, decentralized nature of product transition by a multi-follower interdependent mixed-integer bilevel program. We propose an exact solution approach based on the cut-and-column generation algorithm. We then generalize this model by considering multiple product divisions that each operates in a specific market segment and require multiple resources to carry out their operations. Each product divisions is responsible for the development and production of new products. In fact, a product division contains both engineering and manufacturing units. This again results in a multi-follower interdependent lower level problem for which we develop a cut-and-column generation algorithm. We perform extensive numerical experiments to analyze the performance of the proposed algorithms and provide managerial insights.

Computational Optimization for Hierarchical Decision Making Problems

by
Rahman Khorramfar

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Industrial Engineering

Raleigh, North Carolina
2021

APPROVED BY:

Dr. Reha Uzsoy

Dr. Leila Hajibabai

Dr. Michael Kay

Dr. Osman Ozaltin
Chair of Advisory Committee

DEDICATION

To my wife, Nasrin

BIOGRAPHY

Rahman Khorramfar was born and raised in the city of Binab (Bonab) located in southwest of East Azerbaijan province, Iran. He earned his bachelor's degree in the University of Tabriz in 2014, and his master's degree from Sharif University of Technology in 2016, both in Industrial Engineering (IE). In August of 2017, he moved to the United States to pursue his PhD degree in Industrial and Systems Engineering where he was advised by Dr. Osman Ozaltin.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Osman Ozaltin for his advise and guidance, and above all, for his friendship. For me, Dr. Ozaltin is an outstanding role model with a rigorous and scholarly approach toward complex optimization problems. I am thankful of my defense committee, Dr. Reha Uzsoy, Dr. David Papp, Dr. Michael Kay, and Dr. Leila Hajibabai whose comments and suggestions improved the dissertation. I also want to thank my teaching mentor, Dr. Russell King, for his trust and constant assistance in my teaching experience.

I appreciate the Edward P. Fitts Department of Industrial and Systems Engineering for taking good care of its graduate students. I have been awarded Provost's Fellowship for my first year for which I am thankful. I also thank Dr. Julie Swann for making the department a vibrant place for original research. Our department enjoys a great staff who go beyond their responsibilities to help. Many thanks to all of them.

My friends are a big part of my life. I appreciate and value the friendships I established during my time as a PhD students. I was very lucky for having friends from all walks of life and from many regions of the world. I want to thank my fellow graduate friends in the department, and my dearest friends outside the department. Thank you all; you gave me wonderful memories and I learned so much from you.

Last but definitely not least, my highest acknowledgment goes to my wife and parents. I express my sincere gratitude to my wife, Nasrin, for her relentless support and her forbearance during our difficult initial years in the United States. I feel endlessly fortunate for having her in my life. In fact, Nasrin is my best friend, a friend for good times and difficult times, a friend for life. My parents are the best, and I would like to recognize their loving support during my PhD career. I am forever grateful of my mother, Malahat, and my father, Majid. I greatly regret that I was not able to spend much time with them due to my study abroad, but truly hope that my efforts would make them proud.

TABLE OF CONTENTS

List of Tables	vii
List of Figures	viii
Chapter 1 Introduction and Literature Review	1
1.1 Stochastic Programming	2
1.2 Bilevel Programming	6
1.3 Outline of the Dissertation	8
 Chapter 2 Subtree Decomposition for General Multistage Stochastic Program	 10
2.1 Introduction	11
2.2 Methodology	13
2.2.1 Multistage Stochastic Programming	13
2.3 Subtree Reformulation	16
2.3.1 Notation and Parameters	16
2.3.2 An Illustrative Example	17
2.3.3 Subtree Reformulation	19
2.3.4 Diameter Expansion	22
2.3.5 The Upper Bound Procedure	23
2.4 Computational Study	24
2.4.1 Multistage Capacity Acquisition Problem (MCAP)	24
2.4.2 Multistage Asset Management Problem (MSAM)	26
2.4.3 Instance Generation	28
2.4.4 Analysis of the Results	29
2.5 Summary	34
 Chapter 3 Managing Product Transitions in Semiconductor Manufacturing: A Bilevel Programming Approach	 36
3.1 Introduction	38
3.2 Literature Review	41
3.2.1 Product Transitions	41
3.2.2 Production Planning in Semiconductor Manufacturing	41
3.2.3 Generalized Nash Equilibrium Problem	42
3.2.4 Bilevel Programming	43
3.3 Model Formulation	44
3.4 Solution Approach	49
3.4.1 Single-level Reformulations of the EPTM	53
3.4.2 Constraint and Column Generation Algorithm	58
3.4.3 Lower Bound for BPTM	62
3.5 Numerical Experiments	63

3.5.1	Instance Generation	64
3.5.2	Computational Performance	64
3.5.3	The Impact of Product Mix	68
3.5.4	The Value of CORP's Leadership	69
3.6	Summary	71
Chapter 4 Coordinating Resource Allocation and Product Transitions in Semiconductor Manufacturing Using Bilevel Programming Model		72
4.1	Introduction	73
4.2	Literature Review	76
4.2.1	Production Transition	76
4.2.2	Capacity Planning in Semiconductor Firms	77
4.2.3	Generalized Nash Equilibrium Problem	77
4.2.4	Bilevel Programming	78
4.3	Model formulation	80
4.4	Solution Approach	84
4.4.1	Pseudo-Nash Equilibrium Games	84
4.4.2	Single-level Reformulation	87
4.4.3	Cut-and-Column Generation Algorithm	89
4.5	Computational Experiments	91
4.5.1	Instance Generation and Implementation Details	91
4.5.2	Results and Discussions	93
4.6	Summary	98
Chapter 5 Conclusion		99
References		104

LIST OF TABLES

Table 2.1	Parameters in MCAP	25
Table 2.2	Numerical results on small-size instances for MCAP	30
Table 2.3	Upper and lower bound gap for small-size MCAP instances. Rows corresponds to the actual gap for each class of instances. LB1 and LB2 results in the best bounds with less than 2% for all the instances.	31
Table 2.4	Numerical results for small-size instances of MSAM	32
Table 2.5	Gap between LB3 and UB for large-size instances of MCAP	32
Table 2.6	Percentage upper and lower bound gap for small-size MSAM instances.	33
Table 2.7	Upper and lower bound values for large-size instances of MCAP	34
Table 3.1	Parameters of the bilevel product transition model	46
Table 3.2	Decision variables of the bilevel product transition model	46
Table 3.3	Parameter values	64
Table 3.4	Number of constraints and variables in each instance. The number of binary variables is given in parenthesis. CORP is the leader in BPTM. MFG and ENG are followers.	66
Table 3.5	The performance of the CCG algorithm with Exact Reformulations 1 and 2.	67
Table 3.6	Average change in the optimal objective values of CORP, MFG, and ENG as the number of new products $ P $ increases.	68
Table 3.7	Change in each component of the MFG objective with number of new products.	69
Table 3.8	The difference in the optimal objective values of the CORP, MFG, and ENG between MFG-Leader and BPTM-Nash. The reported instances are in problem class C2.	70
Table 3.9	The difference in the optimal objective values of the CORP, MFG, and ENG between CORP-Release and BPTM-Nash. The reported instances are in problem class C2.	70
Table 4.1	Parameters of the bilevel product transition model for each PD_j	81
Table 4.2	Parameter values	92
Table 4.3	Demand Distributions	92
Table 4.4	Set of values and distributions considered to generate instances	93
Table 4.5	Te performance of the CCG algorithm	95
Table 4.6	Effect of competition on the average objective value of PDs	97

LIST OF FIGURES

Figure 2.1	Scenario tree representation of a MSP where the stochastic process $\xi = \{\xi_1, \dots, \xi_T\}$ form a Markov chain. The scenario tree has N nodes starting from the root node 0 ending at node $N - 1$. Time marches ahead from the current time $t = 1$ until the end of the planning horizon $t = T$ in discrete steps. \mathcal{N}_t is associated with each time stage and contains the set of corresponding nodes.	14
Figure 2.2	A sample notation on a scenario tree with $T = 4$ and $N = 17$	15
Figure 2.3	An example of a subtree decomposition on a scenario tree with $T = 4$ and $N = 15$	17
Figure 2.4	Example of an invalid subtree decomposition	18
Figure 2.5	Examples of DEP with different diameters and covers	23
Figure 2.6	Lower bound gap for small-size MCAP instances	31
Figure 2.7	Lower bound gap for small-size MSAM instances	32
Figure 3.1	Schematic view of the proposed bilevel model with CORP as the leader and MFG and ENG as interdependent followers who play a generalized Nash equilibrium game over factory capacity. Arrows represent "decision variables" communicated between CORP and the followers. The demand estimates and due date for development of products are parameters made available for the followers from outside sources.	45
Figure 4.1	General flow of resources and information among decision units in the proposed bilevel model. Given the upper-level budget allocation and capacity release decisions, PDs compete among themselves over the factory and engineering resources as depicted by solid arrows in the lower level problem. In each period, the availability of engineering and factory capacity, and demand for each products are given as parameters. Once PDs reach an equilibrium, they communicate their objective values to the leader.	81
Figure 4.2	The impact of Bc on the total objective value of the leader (a), and average objective value of PDs (b)	97
Figure 4.3	Percentage Change in the Average Obj. Value of PDs Under Competition	98

Chapter 1

Introduction and Literature Review

This dissertation deals with problems arising in hierarchical decision making (HDM) processes. HDM is a game-theoretical decision process in which one or more decision makers (DMs) solve sequential interdependent optimization problems over single or multiple planning periods. The decision problem of each DM in each time period is parameterized by the decisions of other DMs who, in turn, compete to optimize their own optimization problem. From an optimization point of view, problems involving HDM are extensively studied under the terms *multistage optimization* and *multilevel programming*. In multistage optimization problems DMs are cooperative players, each of whom makes decisions for multiple time periods while taking into account their (expected) impact for the future periods. Two-stage and multi-stage stochastic programmings, are prime examples of multistage optimization problems. Multilevel program, however, refers to HDM where decision makers are noncooperative players who seek to optimize their own objective function in a sequential manner. The Stackelberg game, especially its static version which is known as bilevel programming, is the best known multilevel programming problem. Despite the broad applicability of HDM problems, most multistage and multilevel problems are not amenable to solution methods developed for single-level and single-period decision models. Hence the majority of the HDM literature imposes idealized assumptions in which the subsequent stages for multistage problems or lower level problems for multilevel problems are continuous linear optimization models without any discrete variables. This allows the solution methods to exploit strong duality theorem to derive either the single level reformulation or utilize decomposition paradigms such as column

generation algorithms and cutting plane approaches. It has been only recently that the HDM problems with discrete variables are embraced earnestly, hence paving the way to address many real world problems.

This dissertation contributes to the theory of HDM and extends its applications to new domains. We present new algorithmic approaches to tackle two challenging HDM problems and their applications. Our solution methods do not assume linearity nor convexity for the subsequent states for multistage problems. We also do not require linearity of the lower level problems for the multilevel case. We first study multistage stochastic problems (MSP) where we propose a solution approach that provides lower and upper bounds to wide range of such problems. We consider an application of multilevel programs as the second class of HDM problems and model product transitions in semiconductor manufacturing companies as a bilevel program with multiple interdependent followers. Our proposed solution approach for this class of HDM effectively handles the mixed-integer interdependent follower's problem. In the remainder of this chapter we first provide a brief review of the stochastic optimization literature including solution approaches and applications. We then proceed to bilevel programs and briefly cover solution approaches for variants of the problem as well as sample applications. We present the structure of this dissertation at the end of the chapter.

1.1 Stochastic Programming

Stochastic programs (SP) are a subset of HDM problems suitable for modeling sequential decisions that involve parameters whose exact values are unknown a priori due to uncertainty of future events. Many SP assume a finite number of possible scenarios that uncertain parameters could adopt as the future events gradually unfold. Based on the number of planning horizon, SPs are generally divided into two-stage and multistage stochastic programs (MSP). Two-stage SPs are static planning problems where an initial set of decisions is made at the time of planning and then, depending on the realization of uncertainty, recourse decisions are taken to account for the new situation. MSPs, however, offer a more flexible modeling framework by allowing multiple periods of plannings. In other words, MSPs incorporate a time component making the planning dynamic to properly handle the gradual realization of future events (Kall et al. 1994; Birge and Louveaux 2011).

The solution methods for two-stage SPs usually depends on the structure of the second stage problem, known as the recourse problem. In the case of linear recourse problems, many approaches such as the continuous L-shape method use the convexity of the second stage problem and enforce its optimality by the strong duality property (Schultz et al. 1996). However, the presence of discrete variables in the second stage immediately renders this property inapplicable. Over the years, two-stage mixed-integer stochastic programs have captured the attention of many researchers due to its immense applications. Laporte and Louveaux (1993) pioneer the study of two-stage SPs with integer variables in the recourse problem where they adapt the L-shaped method and develop a branch-and-cut algorithm when the second stage is a mixed-binary program. Carøe and Tind (1998) generaliz the L-shape method to the mixed-integer recourse case. The method relies on the dual functions of integer programs to approximate the value function of the recourse problem. Ahmed et al. (2004) develop a finite branch-and-bound scheme for the two-stage SPs with mixed-integer first stage and pure integer second stage problems. They reformulate the problem via the variable transformation that makes the discontinuities associated with the value function of the second stage problem orthogonal to the variables. The authors then develop a branch-and-bound scheme that eliminates discontinuities and provides en exact representation of the second stage value function. (Sherali and Zhu 2006) consider the general mixed-integer two-stage SP and propose a modified Benders decomposition based branch-and-bound procedure in which the resulting Benders subproblem are sequentially convexified using reformulation-linearization and lift-and-project techniques. A similar approach based on branch-and-cut framework is proposed in (Sen and Sherali 2006). The case of binary first stage variables and general integer variables in the second stage is considered in (Gade et al. 2014). The authors develop a decomposition algorithm that uses valid inequalities that sequentially provides tighter approximation of the second stage problem.

Two-stage stochastic programs has numerous applications in a multitude of domains (Sahinidis 2004). Nürnberg and Römisich (2002) model cost optimal electric power production planning as a two-stage SP in which the demand of the hydro-thermal plant as well as fuel price are uncertain parameters. The authors propose a Lagrangian relaxation scheme equipped with dynamic programming to solve subproblems. Barahona et al. (2005) develop a multi-period two-stage SP for capacity planning in semiconductor manufacturing where the demand is uncertain and the goal is to minimize the expected value of unmet demand. The resulting model is solved approximately by a heuristic cutting

plane algorithm. Li et al. (2011) utilize this modeling framework to address the pooling problem in natural gas production. Another application of two-stage SP in power systems is presented in (Shi and Oren 2016) to evaluate the cost-effectiveness of integrating wind power into the power grids. The resulting two-stage stochastic unit commitment formulation is solved by commercial solver. Two-stage SPs have many application in logistics and supply chain management including humanitarian logistics (Grass and Fischer 2016). Barbarosoğlu and Arda (2004) model the transportation of vital commodities in post-disaster response as a multi-commodity multi-modal stochastic network flow where the resource mobilization is the stochastic parameters. The model also accounts for disruption in the form of random arc capacities.

As a natural generalization of two-stage planning, multistage stochastic programs offer more flexible planning framework, albeit with added challenges for modeling and solution. The earliest systematic effort to efficiently solve the multistage models dates back to 1980s. Birge (1985) extend the L-shape method to linear multistage case and propose a decomposition scheme that partition the time stages. To tackle large scenario trees, Pereira and Pinto (1991) apply a sampling based variant of the nested Benders decomposition, and called stochastic dual dynamic programming (SDDP). This method is based on building the cost function of future stages by piecewise linear outer approximation. The random sampling of SDDP allows the algorithm to effectively avoid the curse of dimensionality. Chen and Powell (1999) propose a convergent cutting plane and partial sampling scheme for the linear MSP in which the cutting plane mechanism approximates the expected recourse functions. Rebennack (2016) considers the case when the uncertainty is assumed to be stagewise dependent and uses the sampling procedure of the SDDP algorithm in a nested Benders decomposition framework. More recently, Zou et al. (2019) extend the SDDP to MSP with binary state variables. They reformulate the nodal subproblems by making copies of state variables. The Lagrangian dual of the reformulation is then solved to generate a new type of cut, and the convergence of the algorithm is established.

As an alternative for exact methods, some studies has focused on generating statistical bounds, especially when the MSP contains discrete variables. Shapiro (2006) estimate the size of samples required by the Sample Averaging Approximation (SAA) approach to solve the problem to a given accuracy. They argue that the exponential growth of the scenario tree makes SAA relatively ineffective for these problems. Zenarosa et al. (2014) propose

vertex cuts to partition the scenario nodes by separating the root node from the scenario nodes. For a given partition, they formulate subproblems by choosing a pre-determined number of scenarios from each member of the partition. This method provides a lower bound by taking an expectation over all such group subproblems, and an upper bound by completing the solution of subproblems to construct a feasible solution. Sandikçi and Özaltın (2017) develop a generic bounding method to the general MSP (i.e. not requiring the convexity of functions or variables). They propose a scenario decomposition scheme that results in independent subproblems whose solutions provide a lower bound. Some researchers derive bounds from the polyhedral description of the problem. Daryalal et al. (2020) consider mixed-integer MSP and propose bounding method based on the Lagrangian relaxation of the problem. They propose stagewise Lagrangian duals to relax state equations coupling successive stages and nonanticipativity Lagrangian duals to create copies of variables and relax the nonanticipativity constraints. Ryan et al. (2020) consider scenario grouping and propose a optimization framework to optimally group scenario nodes. Their algorithm relies on the information obtained from a given candidate solutions to maximize a metric that reflects the quality of bounds.

The literature of MSP enjoys many successful applications since many real world problems are dynamic problems and contain uncertainty, and can naturally be modeled as MSPs. Sen et al. (2006) model a decision support system that integrates unit commitment constraints with financial decision for wholesale power market. Ahmed et al. (2003) model the multi-period expansion of production capacity as a mixed-integer MSP in which the demand and investment cost are uncertain parameters. They propose a variable disaggregation technique to exploit the lot-sizing structure of subproblems. de Matos and Finardi (2012) consider a long-term scheduling of hydrothermal power systems under water resource uncertainty. The open pit mine production scheduling problem is modeled as a mixed-integer MSP in (Boland et al. 2008) in which the uncertain parameters depend on the observation of parameters in earlier stages, thus creating an endogenous uncertainty. Özaltın et al. (2011) formulate the optimal timing of a flu shot and its composition with mixed-integer MSP and use the block structure of the problem to decompose the model and solve the resulting subproblems by dynamic programming. Gul et al. (2012) formulate the assignment of surgeries to operating rooms as a mixed-integer MSP and approximate the optimal solution by the progressive hedging algorithm. Stochastic unit commitment is considered in (Zou et al. 2018) for which the authors apply stochastic dual dynamic integer programming approach of (Zou et al. 2019). More recently, the

application of MSP has been extended to biology in (Kıbaş et al. 2021). The authors formulate the joint optimization of surveillance and control of an invasive insect in cities to help authorities better target the insects and remove the infested trees.

1.2 Bilevel Programming

Bilevel programming (BP) is a special case of multilevel problems that provide an framework to model HDM problems with multiple competing decision makers in each planning period. BP can be analyzed in both game theoretical and mathematical modeling terms. From the perspective of game theory, BP represents a static Stackelberg game between a leader and a follower (Bard 1998; Dempe 2002). First, the leader chooses a strategy based on its anticipation of the follower’s response. The follower responds to the leader, and the game ends (Colson et al. 2007). In mathematical optimization terms, bilevel programming is a nested optimization problem over a feasible region where the follower’s optimization problem defines a subset of leader’s constraints (Bracken and McGill 1973).

Depending on their structure, BPs are generally classified into linear BPs (LBP) and mixed-integer BPs (MIBP). At least one of the leader’s or follower’s problems in the MIBPs contain discrete variables. LBPs, on the other hand, lack any discrete variables, allowing the strong duality property can be employed to enforce the optimality of the follower’s problem (Bertsimas and Tsitsiklis 1997). The resulting single-level problem is a nonlinear program which exploits algorithmic developments for nonlinear programs.

Solution methods for the BPs with the linear follower’s problem are abundant as the lower level problem can be replaced by its Karush-KuhnTucker (KKT) conditions to yield the single-level reformulation. Bard and Falk (1982) apply a branch-and-bound method that branches on the complementarity constraints. Hansen et al. (1992) incorporate the branch-and-bound scheme with penalty anticipation for fixing a variable that is widely used in mixed-integer programming. Campelo et al. (2000) propose a method based on the penalization of the duality gap for the lower level problem. A branch-and-cut algorithm is proposed in (Audet et al. 2007). The authors exploit the equivalence between LBP with mixed-binary program and embed Gomory-like cuts in a branch-and-bound algorithm.

The first systematic effort to solve MIBP is presented in (Bard and Moore 1992) in

which discrete variables in the lower-level problem are considered. The authors apply a branch-and-bound algorithm that branches on the discrete variables of the lower level problem. DeNegre and Ralphs (2009) extend this work and develop a branch-and-cut algorithm for a pure integer BP; for the same problem, Caramia and Mari (2016) introduce a nonlinear cut and a branch-and-cut algorithm. Xu and Wang (2014) consider a MIBP and develop a branch-and-bound algorithm where branching occurs on the slack variables of the follower’s constraints. (Wang and Xu 2017) embed multiway disjunction cuts to remove bilevel infeasible solutions in a branch-and-bound framework. Lozano and Smith (2017) consider the optimal value function of the lower level problem and reformulate the MIBP as a single-level optimization problem, and develop an algorithm that generates a lower bound from the relaxed problem and upper bound from the feasible solutions generated so far. Another branch-and-cut algorithm is proposed in (Fischetti et al. 2018) that employs intersection cuts. Yue et al. (2019) derive the single-level reformulation of MIBP by considering it as a constrained mathematical program, and develop a projection-based decomposition approach that implicitly enumerates the lower level’s integer variables.

Defense problems were the the earliest application of BP (Bracken and McGill 1973), but the applications of bilevel programming quickly expanded to other domains (Kleinert et al. 2021). Gumus and Ciric (1997) use BP to model a chemical reactive distillation process that involves thermodynamic equilibria. Machine learning and statistics is another context where researchers find BP useful. Bennett et al. (2008) discuss the applicability of BP in machine learning models and propose a LBP for hyper-parameter tuning of statistical learning. Franceschi et al. (2018) unify the gradient-based hyperparameter optimization and meta-learning as a BP. MacKay et al. (2019) employ BP to map the hyperparameters to optimal weight for a neural network. More recently, Hong et al. (2020) apply BP to sample fresh data as an actor-critic algorithm runs. Healthcare problems are among the emerging applications of the bilevel programs. Robbins and Lunday (2016) propose a bilevel pediatric vaccine pricing problem to determine the optimal pricing strategies for a vaccine firm operating in a an oligopolistic environment. Özaltın et al. (2018) formulate the seasonal flu shot design problem as a multistage stochastic MIBP to determine the composition of flu shots and the timing of the strain selections.

Bilevel programs are also explored in the context of logistics and production planning. Ryu et al. (2004) propose a bilevel model to address hierarchical decision problems under

uncertainty in the enterprise-wise supply chain in which the upper level corresponds to a plant planning problem and the lower level problem formulates the distribution network. Garcia-Herreros et al. (2016) address the capacity expansion problem by a MIBP where the leader’s problem maximizes the producer profit and determines expansion plans, while the follower’s problem determines the demand assignment while minimizing the cost paid by the market. In (Dan et al. 2020), the authors study service facilities in a competitive market considering customer behavior and propose mixed-integer nonlinear BP that determines location, service rate, and prices offer by facilities.

Power and energy systems are the largest area of BP application (see, for example (Gabriel et al. 2012)). Motto et al. (2005) propose a MIBP to analyze the security of electricity grid under disruption threats. Power generation and transmission expansion planning problem are formulated as a BP in (Garcés et al. 2009). Lavigne et al. (2000) apply BP to analyze the impact of various pricing mechanism in the electricity market. A joint wind power investment and transmission reinforcement model is proposed in (Baringo and Conejo 2012). Shelar et al. (2021) propose a generalized Benders method to tackle the bilevel problem of evaluating resiliency of electricity distribution network against disruptions.

1.3 Outline of the Dissertation

The rest of the dissertation presents hierarchical decision making problems we consider. Chapter 2 examines the convex multistage stochastic problems and proposes a reformulation to establish bounds by systematically decomposing the scenario tree into subproblems who can be solved by commercial solvers in parallel. The focus of Chapter 3 and 4 are the applications of bilevel programming to product transition in a semiconductor firm. Chapter 3 considers production transitions when the transition is managed by three entities, namely corporate management, engineering and manufacturing divisions. Corporate management leads the company towards its strategic goals. Engineering and manufacturing entities form the followers problem. Engineering division is responsible for all the activities concerning production development, whereas manufacturing division determines the production decisions. Chapter 4 generalizes the same problem to incorporate multiple resources and multiple markets. More specifically, our model in Chapter 4 contains multiple product divisions each of which is responsible for a specific market

segment. Each product division is responsible for the development and production of new products, containing both engineering and manufacturing units. The novel aspect of the problems in Chapter 3 and 4 is that the lower level problem in both models has multiple interdependent followers who compete over resources. This creates several theoretical challenges for which we develop suitable solution approaches.

Chapter 2

Subtree Decomposition for General Multistage Stochastic Program

Abstract:

Multistage stochastic programming offers a powerful framework for modeling complex sequential decision making problems involving uncertainty. In many problem settings, however, its practicality is hampered by the problem's size or structure, rendering it impossible to solve for the required instance size. Nonlinear problem structure, and memory limitation due to problem size are the main factors rendering an instance difficult to solve. We proposed a reformulation of the general multistage program to establish bounds by systematically decomposing the scenario tree into subproblems that can be solved by commercial solvers. Not only is our method broadly applicable to any multistage stochastic problem under mild assumptions, it fits naturally into multithreaded computing environments as well. We test the performance of bounds on instances of two well-known problems from the literature. The numerical results indicate that the method has the potential to tackle large instances for which no previous method could generate any bound.

2.1 Introduction

Stochastic programming is an approach for modeling sequential optimization problems whose parameters are not fully known to the decision maker. A stochastic program (SP) assumes that the decision maker knows the probability distribution of the uncertain parameters. Due to its relatively nonrestrictive approach toward modeling uncertainty, SP has gained increasing attention in the research community (Kall et al. 1994; Birge and Louveaux 2011). These studies aim to develop efficient algorithms for special classes of SPs (Laporte and Louveaux 1993; Shapiro and Ahmed 2004; Ahmed et al. 2020; Dowson et al. 2020; Boland et al. 2018; Shapiro and Ding 2020) and extend its application to new problem domains in healthcare (Özaltın et al. 2011; Castaing et al. 2016; Gul 2018), logistics and supply chain (Santoso et al. 2005; Grass and Fischer 2016; Powell and Topaloglu 2003), energy (Linderoth and Wright 2005; Carrión et al. 2007; Zhan and Zheng 2018), and finance (Infanger 2008) among others (Wallace and Ziemba 2005). Depending on the planning horizon, SPs can broadly be divided into two-stage and multistage models. In two-stage models, an initial set of decisions is followed by recourse decisions after observing realization of uncertain parameters. In multistage stochastic programs (MSP), this process is repeated multiple times. Unlike two-stage models, MSP offers a dynamic decision environment where the decision maker has flexibility to respond to information reveals gradually over time.

There are a number of solution algorithms proposed for special classes of SPs (Mulvey and Vladimirou 1991; Laporte and Louveaux 1993; Gade et al. 2014). A subset of these algorithms require the second-stage decision variables to be continuous (Sen 2005), preventing their application to a broad set of problems. The problem size grows exponentially as the number of decision stages increase in multistage models. As a result, exact algorithms for MSPs typically resort to decomposition techniques (Carøe and Schultz 1999; Sen and Sherali 2006; Ahmed 2013; Deng et al. 2020). As an alternative, some studies focused on generating statistical bounds. Examples of bound-generating methods include sample average approximation (Kleywegt et al. 2002) and scenario decomposition (SD) (Higle and Sen 1991; Sen and Zhou 2014; Sandikçi and Özaltın 2017).

This chapter presents a bounding method that systematically generates lower and upper bounds for convex multistage stochastic mixed-integer programs (SMIPs). We reformulate MSPs so they can be decomposed into smaller independent subproblems.

Our approach aligns with a recent stream of studies that generate upper and lower bounds for SMIPs using scenario decomposition. For two-stage SMIP Sandıkçı et al. (2013) proposed lower and upper bounds by solving *group subproblems* whose variables and constraints are associated with a subset of scenarios of size b . They use a *reference scenario* to balance the effect of extreme scenarios and provide a hierarchy of lower bounds that improve monotonically b . An upper bound is calculated by solving the problem for a subset of scenarios and embedding the partial solution in the original model to obtain a feasible solution. This study has been extended to multistage settings in several directions. In particular, Zenarosa et al. (2014) considered a different class of group subproblems derived from *vertex cuts* which are used to partition the scenario nodes by separating the root node from the scenario nodes. For a given partition, they formulate subproblems by choosing a pre-determined number of scenarios from each member of the partition. This method provides a lower bound by taking an expectation over all such group subproblems, and an upper bound by completing the solution of subproblems to construct a feasible solution. Maggioni et al. (2016) generalized the bounds in (Sandıkçı et al. 2013) to multistage SMIPs and showed that the monotonicity of bounds in (Sandıkçı et al. 2013) remains valid in both group and reference scenario set size. Another extension of (Sandıkçı et al. 2013) for multistage case is proposed in (Sandıkçı and Özaltın 2017) which develops a generic bounding method that does not require convexity assumptions. The authors apply the scenario decomposition on a *blockset*, which is a collection of scenario node subsets whose union make up the entire scenario node set. The blockset is used to create group subproblems whose objective values are appropriately aggregated to generate a lower bound. This method can efficiently calculate bounds for enormous instances with up to 100 million scenarios and 1.5 billion decision variables. Building upon (Sandıkçı et al. 2013; Zenarosa et al. 2014; Sandıkçı and Özaltın 2017), Bakir et al. (2020) introduced the concept of *b -partition* which partitions the scenario nodes into subsets of nearly equal cardinality of either b or $b - 1$ and show that the expected values of the partitions form a hierarchy in b . The authors propose several sampling approaches to overcome the difficulty of enumerating all b -partitions. However, their methodology is not tested on large instances.

2.2 Methodology

2.2.1 Multistage Stochastic Programming

Multistage stochastic programming (MSP) is a framework to model finite-horizon optimization problems with uncertainty in which sequential decisions are made at discrete stages $\mathcal{T} = \{1, 2, \dots, T\}$. Let ω_t be a finite number of random events realized in stage t with sample space Ω_t (i.e. $\omega_t \in \Omega_t$) and let $\xi_t(\omega_t) \in \mathbb{R}^{d_t}$ be the associated random variables. For a sequential decision making process, the set $\xi(\omega) = \{\xi_1(\omega_1), \dots, \xi_T(\omega_T)\}$ forms a stochastic process (i.e. a collection of time-indexed random variables) and can be interpreted as a set of uncertain parameters revealed over T stages (Shapiro et al. 2014).

The decision making process is initiated by making decisions in the first stage and continues by observing the outcome of a random event ω_2 at stage 2, and then taking a recourse action, and so on. As evident by the process, the decision variables at stage t are *nonanticipative* meaning that they are functions of both random events and decisions made up to time t , but are not affected by the information revealed or decisions made in the subsequent stages. However, to keep the notation from getting too onerous, we drop these explicit dependencies from actions and random variables, thus stating them simply as z_t and ξ_t . Let the closed and continuous function $f_t(z_t, \xi_t)$ be the objective function contribution of action z_t under the random variable ξ_t at stage t . Then a generic MSP aims to minimize the total expected contributions by the following nested formulation (Sandikçi and Özaltın 2017):

$$\min_{z_1 \in \mathcal{Z}_1} f(z_1) + \mathbb{E} \left[\min_{z_2 \in \mathcal{Z}_2} f_2(z_2, \xi_2) + \dots + \mathbb{E} \left[\min_{z_T \in \mathcal{Z}_T} f_T(z_T, \xi_T) \right] \right] \quad (2.1)$$

where $\mathbb{E}[\cdot]$ is the expectation over a random variable, and $z_t \in \mathbb{R}^{n_t}$ the set of decision variables in stage $t \in \mathcal{T}$. The set of feasible solutions in stage $t \in \mathcal{T}$ is denoted by $\mathcal{Z}_t = \{z_t \in \mathbb{R}^{n_t} | g_t(z_{1:t}, \xi_{1:t}) = h_t(\xi_{1:t})\}$ where $z_{1:t} = (z_1, \dots, z_t)$, $\xi_{1:t} = (\xi_1, \dots, \xi_t)$, $g_t(\cdot) : \mathbb{R}^{n_1} \times \dots \times \mathbb{R}^{n_t} \times \mathbb{R}^{d_1} \times \dots \times \mathbb{R}^{d_t} \rightarrow \mathbb{R}^{m_t}$ is the constraint function, and $h_t(\cdot) : \mathbb{R}^{d_2} \times \dots \times \mathbb{R}^{d_t} \rightarrow \mathbb{R}^{m_t}$ is the right hand side function for positive integer values of n_t, d_t and m_t .

We assume that the stochastic process $\xi = \{\xi_1, \dots, \xi_T\}$ has a finite support $\Xi = \{\xi^1, \dots, \xi^S\}$ where $\xi^s = (\xi_2^s, \dots, \xi_T^s)$ for $s \in \mathcal{S} := \{1, 2, \dots, S\}$ is called a *scenario* with associated probability mass of p_s and represents the history of realizations up to time

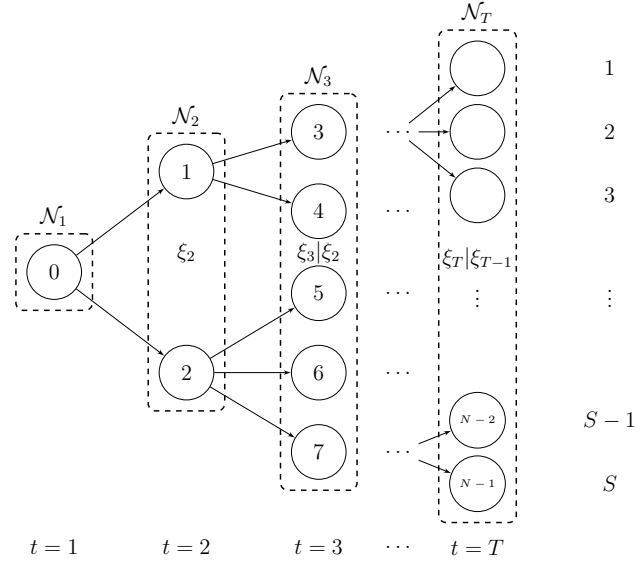


Figure 2.1: Scenario tree representation of a MSP where the stochastic process $\xi = \{\xi_1, \dots, \xi_T\}$ form a Markov chain. The scenario tree has N nodes starting from the root node 0 ending at node $N - 1$. Time marches ahead from the current time $t = 1$ until the end of the planning horizon $t = T$ in discrete steps. \mathcal{N}_t is associated with each time stage and contains the set of corresponding nodes.

T . A MSP can be modeled on a *scenario tree* if $\xi = \{\xi_1, \dots, \xi_T\}$ form a Markov chain; that is, if ξ_t stochastically depends only on its immediate preceding random variable ξ_{t-1} . As shown in Figure 2.1, the scenario tree is a rooted tree with T levels in which level 0 contains only the root node, with index 0, and is associated with ξ_1 . The root node is connected to the nodes in level $t = 2$ that has as many nodes as the number of realizations in ξ_2 ; each node in level $t = 2$ is connected to as many nodes as the number of realization of ξ_3 . Accordingly, each node at level t corresponds to a particular realization of $\xi_{2:t}$ representing the state of the system at stage t (Sandikçi and Özaltın 2017). Let \mathcal{N} be the set of all nodes in the scenario tree and \mathcal{N}_t the set of nodes in stage t . We define *children* as the set of nodes emanating from node n and denote as \mathcal{C}_n . *Parent* is the predecessor of node n and is denoted by $a(n)$. Clearly, $\mathcal{N} := \cup_{t \in \mathcal{T}} \mathcal{N}_t$, $a(0) = \emptyset$. By the same token, $\mathcal{N}_{t+1} := \cup_{n \in \mathcal{N}_t} \mathcal{C}_n$ for $t \in \mathcal{T} \setminus T$ and $\mathcal{C}_{n \in \mathcal{N}_T} = \emptyset$. The scenario tree offers a convenient way of defining a scenario. Let $\mathcal{A}_{m,n}$ be the set of nodes in the path from node m to n where $n, m \in \mathcal{N}$, and S the number of scenarios. A scenario, is then a unique path from root node at stage $t = 1$ to a node in the last stage $t = T$. In other words, a scenario is a realization of the process $\xi_1, \xi_2, \dots, \xi_T$, thus the number of nodes in the last stage of the scenario tree equals S (Shapiro et al. 2014).

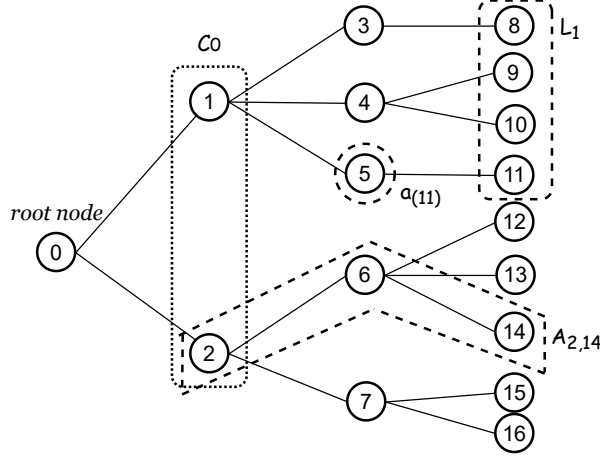


Figure 2.2: A sample notation on a scenario tree with $T = 4$ and $N = 17$

The probability p_s of a scenario node $s \in \mathcal{N}_T$ is derived by the product of the conditional probabilities of all nodes in the path from the root node to s . Figure 2.2 illustrates a sample notation associated with a scenario tree. We use \mathcal{L}_n to denote the set of scenario nodes that have node n in their path to the root node (i.e. $\mathcal{L}_n = \{s | n \in \mathcal{A}_{0,s}, \forall s \in \mathcal{N}_T\}$). Then p_n , the marginal probability of node $n \in \mathcal{N}$, is simply $p_n = \sum_{s \in \mathcal{L}_n} p_s$.

We classify decision variables at each node into two groups: *lasting variables* x_n represent the variables that link decisions across time stages, and hence x_n is a variable that appear in the formulation of nodes in subsequent time stages. *Local variable* y_n , on the other hand, corresponds to a variable that only appears in the formulation of node n . Our definitions of lasting and local variables are similar to the definition of *state* and *stage* variables in (Basciftci et al. 2019), respectively. We assume that the objective function and constraints are convex and additively separable in the lasting and local variables. Let (x_n, y_n) denote the decisions made at node $n \in \mathcal{N}_t$ of stage t ; define $f^n(x_n, y_n) \equiv f_t(x_t, y_t, \xi_t^l)$, $g^n(\{x_m\}_{m \in \mathcal{A}_{0,n}}, y_t) \equiv g_t(x_{1:t}, y_t, \xi_{2:t}^l)$ and $h^n \equiv h_t(\xi_{1:t}^l)$ for $l \in \mathcal{L}_n$. The so-called extensive nodal formulation of a MSP on a scenario tree is given by:

$$\min \phi = \sum_{n \in \mathcal{N}} p_n [f_1^n(x_n) + f_2^n(y_n)] \quad (2.2a)$$

$$s.t. \ g_1^n(\{x_m\}_{m \in \mathcal{A}_{0,n}}) + g_2^n(y_n) \leq h^n, \quad n \in \mathcal{N} \quad (2.2b)$$

$$x_n \in \mathbb{X}, y_n \in \mathbb{Y}, \quad n \in \mathcal{N}_t, t \in \mathcal{T} \quad (2.2c)$$

where $f_1^n(x_n) + f_2^n(y_n) = f^n(x_n, y_n)$ and $g_1^n(\{x_n^m\}_{m \in \mathcal{A}_{0,n}}) + g_2^n(y_n) = g^n(\{x_n^m\}_{m \in \mathcal{A}_{0,n}}, y_n)$ reflecting the additive separable nature of these functions. The notation \mathbb{X} and \mathbb{Y} denote the set of lasting and local variables that can be continuous or integer. Notice that the scenario tree is structured in a way that the nodal formulation naturally ensures nonanticipativity. Despite this convenience, the size of the scenario tree grows exponentially with the number of stages, making it impractical to solve realistic size MSPs directly by commercial solvers.

2.3 Subtree Reformulation

We propose a reformulation of problem (2.2) that, when relaxed, is decomposed into subproblems. This reformulation is based on dividing the scenario trees into subtrees (i.e. subsets of scenario tree nodes which form a tree). We first introduce the notation required to present the reformulation, then continue with an illustrative example, and finally establish a theoretical equivalence result.

2.3.1 Notation and Parameters

We start by defining a collection of subtrees whose union makes up the entire scenario tree. We term this collection as a *bundle set* \mathcal{Q} and each of its members as *bundle* \mathcal{B}_i . A bundle is characterized by its root node $r(\mathcal{B})$ and set of leaf nodes $\mathcal{L}^{\mathcal{B}}$ which are nodes in the last stage of the bundle. We assume that two bundles can share a node only if they span the same number of periods. We define \mathcal{S}_l as the set of bundles sharing node l , and $m_l = |\mathcal{S}_l|$ as the *multiplicity* of node l ; and define \mathcal{R}_l as the set of root nodes potentially affected by the decisions made in node l . Define the *weight* of bundle \mathcal{B} as:

$$w^{\mathcal{B}} = \sum_{l \in \mathcal{L}^{\mathcal{B}}} \frac{p^l}{m_l}.$$

and the adjusted probability of node $n \in \mathcal{B}$ as:

$$\hat{p}_{\mathcal{B}}^n = \frac{1}{w^{\mathcal{B}}} \sum_{l \in \mathcal{L}_n^{\mathcal{B}}} \frac{p^l}{m_l},$$

where $\mathcal{L}_n^{\mathcal{B}}$ is the set of leaf nodes in bundle \mathcal{B} emanating from node $n \in \mathcal{B}$, i.e., $\mathcal{L}_n^{\mathcal{B}} = \{l | l \in \mathcal{L}^{\mathcal{B}}, n \in \mathcal{A}_{r(\mathcal{B}),l}\}$. Note that $\mathcal{L}_{r(\mathcal{B})}^{\mathcal{B}} = \mathcal{L}^{\mathcal{B}}$.

2.3.2 An Illustrative Example

We use the Figure 2.3 to explain the notation defined so far.

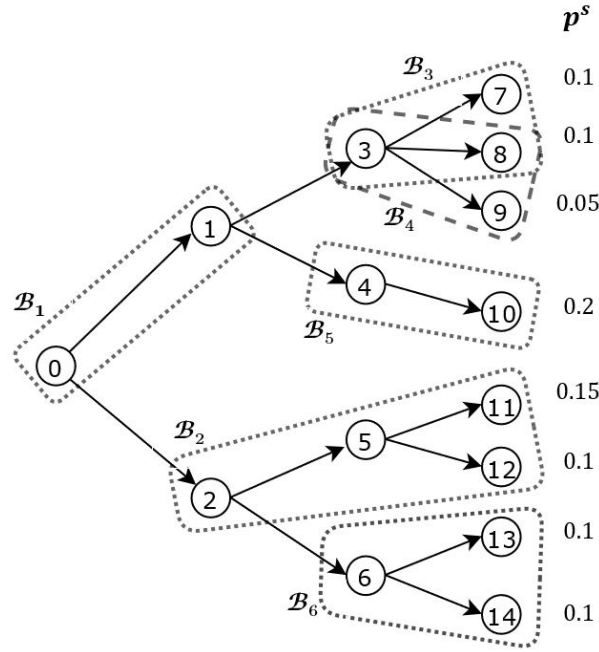


Figure 2.3: An example of a subtree decomposition on a scenario tree with $T = 4$ and $N = 15$

This decomposition's bundle set is $\mathcal{Q} = \{\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4, \mathcal{B}_5, \mathcal{B}_6\}$. The multiplicity of node 3 and node 8 is 2 as they are included in two bundles. The multiplicity of all other nodes is 1. The weights of the bundles are:

$$\begin{aligned}
 w^{\mathcal{B}_1} &= 0.45 & w^{\mathcal{B}_2} &= 0.15 + 0.1 = 0.25 & w^{\mathcal{B}_3} &= 0.1 + \frac{0.1}{2} = 0.15 \\
 w^{\mathcal{B}_4} &= \frac{0.1}{2} + 0.05 = 0.1 & w^{\mathcal{B}_5} &= 0.2 & w^{\mathcal{B}_6} &= 0.1 + 0.2 = 0.3
 \end{aligned}$$

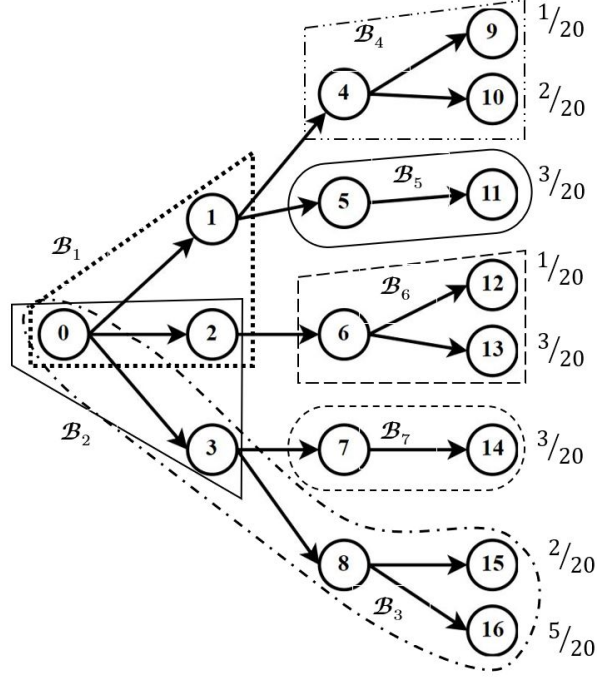


Figure 2.4: Example of an invalid subtree decomposition

The set of affected root nodes for node 0 is $\mathcal{R}_0 = \{2, 3, 4, 6\}$, for node 1 is $\mathcal{R}_1 = \{3, 4\}$, for node 2 is $\mathcal{R}_2 = \{6\}$, and \emptyset for all other nodes.

The set of bundles that share node 3 and 8 is $\mathcal{S}_3 = \mathcal{S}_8 = \{\mathcal{B}_3, \mathcal{B}_4\}$ and \emptyset for all other nodes.

Also notice that $\mathcal{L}_3^{\mathcal{B}_3} = \{7, 8\}$ and $\mathcal{L}_3^{\mathcal{B}_4} = \{8, 9\}$, but $\mathcal{L}_3 = \{7, 8, 9\}$.

Figure 2.4 shows an invalid subtree decomposition. For bundle \mathcal{B}_3 , $m_3 = 2$ but the multiplicity of its leaf nodes are 1. This leads to complications as the following proposition 2.3.1 does not hold for node 3; according to definitions, $\mathcal{S}_3 = \{\mathcal{B}_2, \mathcal{B}_3\}$, therefore:

$$w^{\mathcal{B}_2} = \frac{14}{20}, w^{\mathcal{B}_3} = \frac{7}{20}, \hat{p}_3^{\mathcal{B}_3} = 1, \hat{p}_3^{\mathcal{B}_2} = \frac{10}{14}$$

However: $\sum_{m \in \mathcal{R}_n \cup \{n\}} p_m v_{nm} = \frac{7}{20} \times 1 + \frac{14}{20} \times \frac{10}{14} \neq 0.5$

2.3.3 Subtree Reformulation

We explicitly impose the effect of a lasting variable on the subsequent bundles by creating copy variables. Let \mathcal{R}_n be the set of bundle root nodes that are impacted by the lasting variables at node n . We then define the copy variables as $x_n^m, m \in \mathcal{R}_n \cup \{n\}$. As such, we consider the effect of each x_n^m separately and define *coefficient distribution* v_{nm} to account for the contribution of a lasting variable on the impacted root nodes. If $\mathcal{R}_n = \emptyset$, then $v_{nn} = 1$, otherwise we assume that v_{nn} is a parameter set by the user, and then define the v_{nm} as:

$$v_{nm} = \frac{p_n(1 - v_{nn})}{p_m |\mathcal{R}_n|} \quad m \in \mathcal{R}_n$$

Proposition 2.3.1. *For any node $n \in \mathcal{N}$, the following holds*

$$\sum_{m \in \mathcal{R}_n \cup \{n\}} p_m v_{nm} = p_n$$

Proof.

$$\begin{aligned} \sum_{m \in \{\mathcal{R}_n \cup n\}} p_m v_{nm} &= p_n v_{nn} + \sum_{m \in \mathcal{R}_n} p_m v_{nm} = p_n v_{nn} + \sum_{m \in \mathcal{R}_n} p_m \frac{p_n(1 - v_{nn})}{p_m |\mathcal{R}_n|} \\ &= p_n v_{nn} + p_n(1 - v_{nn}) \sum_{m \in \mathcal{R}_n} \frac{1}{|\mathcal{R}_n|} = p_n v_{nn} + p_n(1 - v_{nn}) = p_n \end{aligned}$$

□

Proposition 2.3.2. *For any node $n \in \mathcal{N}$, the following holds:*

$$\sum_{\mathcal{B} \in \mathcal{S}_n} w^{\mathcal{B}} \hat{p}_{\mathcal{B}}^n = p_n$$

Proof.

$$\sum_{\mathcal{B} \in \mathcal{S}_n} w^{\mathcal{B}} \hat{p}_{\mathcal{B}}^n = \sum_{\mathcal{B} \in \mathcal{S}_n} w^{\mathcal{B}} \frac{1}{w^{\mathcal{B}}} \sum_{l \in \mathcal{L}_{\mathcal{B}}^n} \frac{p^l}{m_l} = \sum_{\mathcal{B} \in \mathcal{S}_n} \sum_{l \in \mathcal{L}_{\mathcal{B}}^n} \frac{p^l}{m_l} = p_n$$

□

We now present the reformulation for problem (2.2). Consider the following reformulation for any bundleset \mathcal{Q} :

$$\min \phi(\mathcal{Q}) = \sum_{\mathcal{B} \in \mathcal{Q}} w^{\mathcal{B}} \left[\sum_{b \in \mathcal{B}} \hat{p}_b^{\mathcal{B}} [v_{bb} f_1^b(x_b^{\mathcal{B}}) + f_2^b(y_b^{\mathcal{B}})] + \sum_{j \in \mathcal{A}_{0,a(r(\mathcal{B}))}} v_{jr(\mathcal{B})} f_1^j(x'_j) \right] \quad (2.3a)$$

$$s.t. \ g_1^{1i}(\{x'_i\}_{i \in \mathcal{A}_{0,a(r(\mathcal{B}))}}) + g_1^{2j}(\{x_j^{\mathcal{B}}\}_{j \in \mathcal{A}_{r(\mathcal{B}),b}}) + g_2^b(y_b^{\mathcal{B}}) \leq h^b \quad \mathcal{B} \in \mathcal{Q}, b \in \mathcal{B} \quad (2.3b)$$

$$x'_n = x_n^{\mathcal{B}} \quad n \in \mathcal{N}, r \in \mathcal{R}_n \{\cup n\}, \mathcal{B} \in \mathcal{S}_r \quad (2.3c)$$

$$y'_n = y_n^{\mathcal{B}} \quad n \in \mathcal{N}, \mathcal{B} \in \mathcal{S}_n \quad (2.3d)$$

$$x_b^{\mathcal{B}}, x'_n \in \mathbb{X}, y_b^{\mathcal{B}}, y'_n \in \mathbb{Y} \quad n \in \mathcal{N}, \mathcal{B} \in \mathcal{Q}, b \in \mathcal{B} \quad (2.3e)$$

where $\phi(\mathcal{Q}, \mathcal{B}, x', y')$ is the bundle subproblem and $(x^{\mathcal{B}}, y^{\mathcal{B}}) \in \operatorname{argmin} \phi(\mathcal{Q}, \mathcal{B}, x', y')$:

$$\phi(\mathcal{Q}, \mathcal{B}, x', y') = \min \sum_{b \in \mathcal{B}} \hat{p}_b^{\mathcal{B}} [v_{bb} f_1^b(x_b) + f_2^b(y_b)] + \sum_{j \in \mathcal{A}_{0,a(r(\mathcal{B}))}} v_{jr(\mathcal{B})} f_1^j(x'_j) \quad (2.4a)$$

$$s.t. \ g_1^{1i}(\{x'_i\}_{i \in \mathcal{A}_{0,a(b)}}) + g_1^{2j}(\{x_j\}_{j \in \mathcal{A}_{r(\mathcal{B}),b}}) + g_2^b(y_b) \leq h^b, \quad \mathcal{B} \in \mathcal{Q}, b \in \mathcal{B} \quad (2.4b)$$

$$x_b, \in \mathbb{X}, y_b, \in \mathbb{Y}, \quad n \in \mathcal{N}, \mathcal{B} \in \mathcal{Q}, b \in \mathcal{B} \quad (2.4c)$$

The objective function (2.3a) is the summation of individual bundles of the bundleset \mathcal{Q} . In constraint (2.3b), $g_1^1(\cdot)$ and $g_1^2(\cdot)$ are constraint functions associated with lasting variables that precede bundle \mathcal{B} and those belong to bundle \mathcal{B} , respectively. Constraints (2.3c) and (2.3d) are binding constraints which requires the lasting and local variables of a node to take the same values.

Theorem 1. *Problem (2.2) and (2.3) are equivalent.*

Proof. We develop the proof in two parts. In the first part, we show that the feasible regions of the two formulations are the same. We first consider a feasible solution (x', y') for problem (2.2) and show that there exist $(x^{\mathcal{B}}, y^{\mathcal{B}})$ such that $(x', y', x^{\mathcal{B}}, y^{\mathcal{B}})$ is feasible to the (2.3). For every node $n \in \mathcal{N}$ create $x^{\mathcal{B}}$, i.e., copy variables of x'_n , for every bundle \mathcal{B} that it impacts. Similarly, create $y^{\mathcal{B}}$, i.e., copy variables of y'_n , for every bundle \mathcal{B} that shares node n . Clearly, this solution satisfies (2.3c) and (2.3d). To see that it also satisfies (2.3b), notice that $\{b | b \in \mathcal{B}, \mathcal{B} \in \mathcal{Q}\} \equiv \mathcal{N}$.

Now consider a feasible solution $(x', y', x^{\mathcal{B}}, y^{\mathcal{B}})$ for (2.3). Clearly, the solution (x', y') satisfies (2.2b) simply by replacing $x_b^{\mathcal{B}}$ by x'_b and $y_b^{\mathcal{B}}$ by y'_b . Then we have:

$$\begin{aligned} g_1^{1i}(\{x'_i\}_{i \in \mathcal{A}_{0,a(r(\mathcal{B}))}}) + g_1^{2j}(\{x_j^{\mathcal{B}}\}_{j \in \mathcal{A}_{r(\mathcal{B}),b}}) + g_2^b(y_b^{\mathcal{B}}) &\leq h^b \quad \mathcal{B} \in \mathcal{Q}, b \in \mathcal{B} \\ g_1^n(\{x'_m\}_{m \in \mathcal{A}_{0,n}}) + g_2^n(y'_n) &\leq h^n, \quad n \in \mathcal{N} \end{aligned}$$

Next we show that the objective functions for both problems are equal for each feasible solution.

$$\sum_{\mathcal{B} \in \mathcal{Q}} w^{\mathcal{B}} \left[\sum_{b \in \mathcal{B}} \hat{p}_b^{\mathcal{B}} [v_{bb} f_1(x_b^{\mathcal{B}}) + f_2(y_b^{\mathcal{B}})] + \sum_{h \in \mathcal{A}_{0,a(r(\mathcal{B}))}} v_{h,r(\mathcal{B})} f(x'_h) \right] \quad (2.5a)$$

$$= \sum_{\mathcal{B} \in \mathcal{Q}} w^{\mathcal{B}} \sum_{b \in \mathcal{B}} \hat{p}_b^{\mathcal{B}} [v_{bb} f_1(x_b^{\mathcal{B}}) + f_2(y_b^{\mathcal{B}})] + \sum_{h \in \mathcal{A}_{0,a(r(\mathcal{B}))}} w^{\mathcal{B}} v_{h,r(\mathcal{B})} f_1(x'_h) \quad (2.5b)$$

$$= \sum_{n \in \mathcal{N}} \sum_{\mathcal{B} \in \mathcal{S}_n} w^{\mathcal{B}} \hat{p}_n^{\mathcal{B}} [v_{nn} f_1(x'_n) + f_2(y'_n)] + \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{R}_n} \sum_{\mathcal{B} \in \mathcal{S}_m} w^{\mathcal{B}} v_{nm} f_1(x'_n) \quad (2.5c)$$

$$= \sum_{n \in \mathcal{N}} p_n [v_{nn} f_1(x'_n) + f_2(y'_n)] + \sum_{n \in \mathcal{N}} \sum_{m \in \mathcal{R}_n} p_m v_{nm} f_1(x'_n) \quad (2.5d)$$

$$= \sum_{n \in \mathcal{N}} \left[p_n v_{nn} f_1(x'_n) + \sum_{m \in \mathcal{R}_n} p_m v_{nm} f_1(x'_n) + p_n f_2(y'_n) \right] \quad (2.5e)$$

$$= \sum_{n \in \mathcal{N}} \left[\sum_{r \in \mathcal{R}_n \cup \{n\}} p_r v_{nr} f_1(x'_n) + p_n f_2(y'_n) \right] \quad (2.5f)$$

$$= \sum_{n \in \mathcal{N}} p_n [f_1(x'_n) + g(y'_n)] \quad (2.5g)$$

Note that $x_b^{\mathcal{B}}$ is replaced by x'_n and $y_b^{\mathcal{B}}$ by y'_n according to constraint (2.3d). (2.5d) holds from Proposition 2.3.2, and (2.5g) from Proposition 2.3.1. □

The unique feature of this reformulation is that by relaxing constraints (2.3c) and (2.3d) the subproblems for each bundle can be solved independently in parallel to get a lower bound.

2.3.4 Diameter Expansion

The reformulation proposed in the previous section can accommodate any decomposition that satisfy the definition of a bundleset. In this section we propose a *diameter expansion procedure (DEP)* to systematically construct bundlesets based on only a few user-defined parameters. DEP generates a *pattern* \mathcal{P} which is characterized by a tuple: *diameter* length d and scenario node *cover* \mathcal{C} , hence summarized as (d, \mathcal{C}) . The parameter d specifies the number of periods for the bundle containing the root node. The cover \mathcal{C} , on the other hand, groups scenario nodes into non-empty subsets whose union comprises the set of scenario nodes. Each cover has the form of $(keyword, params)$ where *keyword* shows the method by which a cover is generated and *params* contains the required parameters for the *keyword*. DEP starts by creating the *root bundle* that contains all the nodes until stage d . All the remaining bundles are formed with their root at stage $d + 1$. In other words, each node at stage $d + 1$ is the root for at least one bundle. We consider the following methods to group the scenario nodes in $\mathcal{L}_n, n \in \mathcal{N}_{d+1}$:

- *Single Bundle per Node (sn)*: assign all scenario nodes in \mathcal{L}_n to a single bundle.
- *Sequential Partition (sp)*: given a fixed value $s' \leq |\mathcal{L}_n|$, construct bundles by grouping nodes in order of their indices in the scenario tree. This method does not allow sharing a node among bundles.
- *Sequential Cover (sc)*: given a fixed value $s' \leq |\mathcal{L}_n|$, construct bundles by grouping nodes in order of their indices in the scenario tree such that each bundle has exactly s' leaf nodes. Shared nodes are possible in this method.
- *Random Assignment (ra)*: given a fixed value $s' \leq |\mathcal{L}_n|$, construct bundles by randomly sampling s' leaf nodes without replacement. The process continues until all nodes are assigned.

We assume that *sp* method is the default so that a cover generated with this method is only identified by one parameter s' ; for other cases both the keyword and the parameter should be stated. Note that DEP is a generalization of the scenario decomposition (SD) framework proposed in (Sandikçi and Özaltın 2017). That is, when $d = 0$, all the bundles share the same node which is the root node, hence DEP becomes equivalent to SD.

Figure 2.5 illustrates different grouping methods on a sample tree with $T = 4$ and $N = 16$.

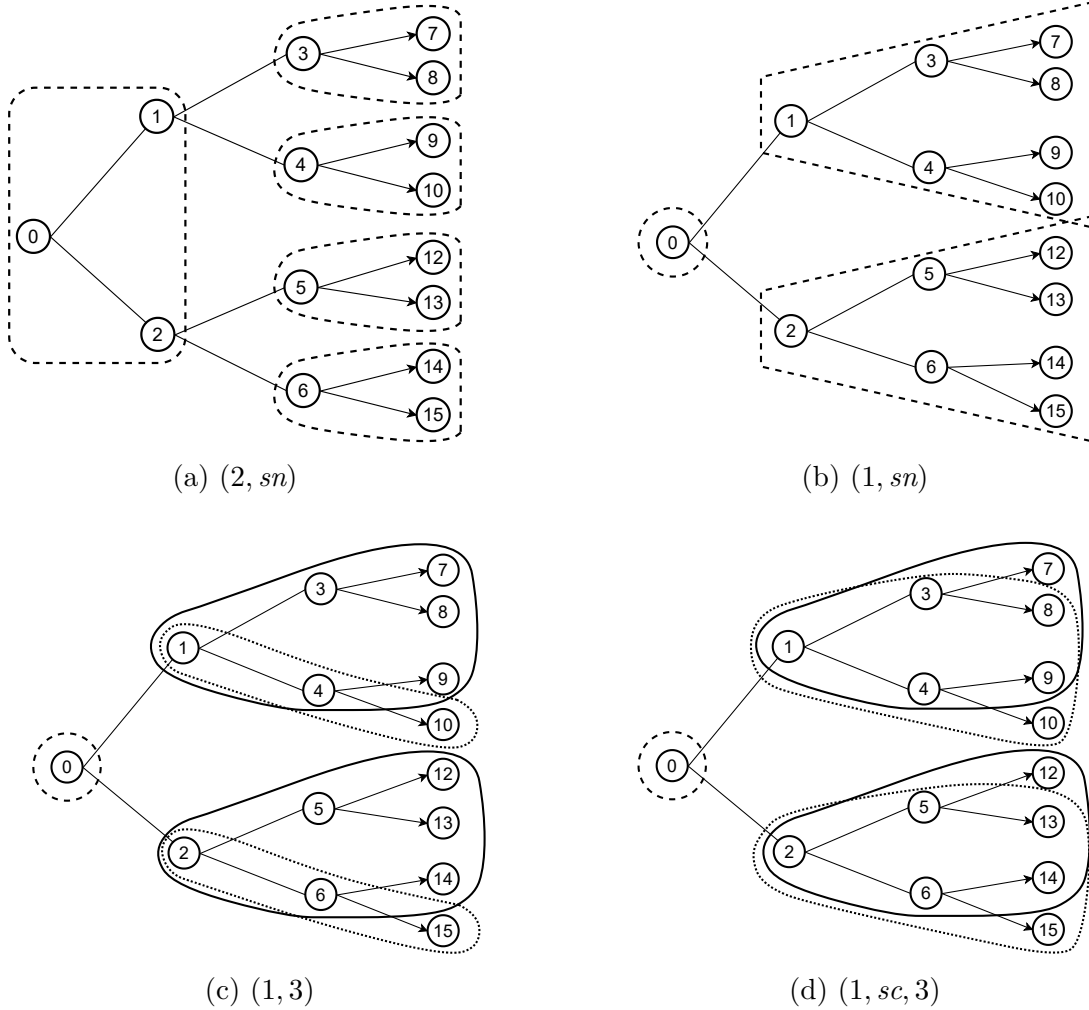


Figure 2.5: Examples of DEP with different diameters and covers

2.3.5 The Upper Bound Procedure

The size of the scenario tree grows exponentially with the number of stages. This is the main hurdle to solving large-scale instances. We propose a method to generate feasible solutions and an upper bound, by which we can assess the quality of the lower bound. The DEP decomposes the tree in a way that only the nodes of the root bundle have

nonempty \mathcal{R} (i.e. set of root nodes impacted). Therefore, fixing the values of decision variables in the root bundle allows the remaining bundles to be solved independently in parallel. Let (\hat{x}, \hat{y}) denote the optimal solution to the root bundle. If the residual solution for all remaining bundles is feasible, then a full feasible solution, hence an upper bound is obtained.

2.4 Computational Study

We carry out numerical analysis on two well-known problems from the literature to test the performance of the bounding methods. Both algorithms are coded in `C++` with `CPLEX 12.9` solver in single thread mode and `OpenMP 4.5` for parallelization. We run instances on a computer with a six-core `Intel Xeon 2.99 GHz` processor and 32 GB RAM. The relative optimality gap in all instances is set to 10e-3.

2.4.1 Multistage Capacity Acquisition Problem (MCAP)

The multistage Capacity Acquisition Problem (MCAP) under uncertainty is a well-studied multistage stochastic program in the literature to determine the minimum cost acquisition and capacity allocation decisions of \mathcal{R} resources to satisfy the uncertain processing requirements of \mathcal{T} tasks over a planning horizon. Various versions of MCAP have been studied with different constraints and objectives (Sandikçi and Özaltın 2017; Basçiftci et al. 2019; Ahmed and Garcia 2003; Huang and Ahmed 2009). We use the MCAP version studied in (Sandikçi and Özaltın 2017):

$$[MCAP] \quad \min \sum_{n \in \mathcal{N}} p_n \left(\sum_{i \in \mathcal{R}} [\nu_n^i x_n^i + \phi_n^i u_n^i] + \sum_{j \in \mathcal{T}} \left[s_n^j z_n^j + \sum_{i \in \mathcal{R}} c_n^{ij} y_n^{ij} \right] \right) \quad (2.6a)$$

$$\text{subject to } x_n^i \leq \kappa_n^i u_n^i \quad n \in \mathcal{N}, i \in \mathcal{R} \quad (2.6b)$$

$$\sum_{j \in \mathcal{T}} \delta_n^j y_n^{ij} \leq \sum_{m \in \mathcal{A}_{0,n}} x_m^i \quad n \in \mathcal{N}, i \in \mathcal{R} \quad (2.6c)$$

$$\sum_{i \in \mathcal{R}} y_n^{ij} + z_n^j = 1 \quad n \in \mathcal{N}, j \in \mathcal{T} \quad (2.6d)$$

$$x_n^i \geq 0, u_n^i, z_n^j, y_n^{ij} \in \{0, 1\} \quad n \in \mathcal{N}, i \in \mathcal{R}, j \in \mathcal{T} \quad (2.6e)$$

Table 2.1: Parameters in MCAP

Parameters	
\mathcal{R}	Set of resources
\mathcal{T}	Set of tasks
ϕ_n^i	Fixed cost of expanding capacity resource i
ν_n^i	Variable cost of expanding capacity resource i
c_n^{ij}	Cost of assigning resource i for task j at node n
s_n^j	Penalty cost of not completing task j
δ_n^j	Processing requirement of task j
κ_n^i	Upper limit on the capacity expansion of resource i

Table 2.1 summarizes the problem parameters. Decision variables include (x, u, z, y) ; x_n^i is the amount of capacity expansion for resource i at node n , u_n^i is an indicator variable for expanding capacity of resource i at node n , y_n^{ij} takes value 1 if resource i is assigned to task j at node n , and z_n^j is an indicator variable for not satisfying the processing requirement for task j at node n . The objective function (2.6a) aims to minimize the total expected cost of capacity acquisition and resource assignment to tasks. The upper limit for capacity acquisition is enforced in constraints (2.6b). Constraints (2.6c) limit the total assigned capacity of a resource to its available amount, and (2.6d) determines whether the process requirement for task j is satisfied or not. Given a solution $(x_n, u_n, y_n, z_n)_{n \in \mathcal{A}_{0,a}(r(\mathcal{B}))}$, the subproblem for bundle \mathcal{B} is given by:

$$\min \sum_{b \in \mathcal{B}} \hat{p}_b^{\mathcal{B}} \left(v_{bb} \sum_{i \in \mathcal{R}} [\phi_b^i u_b^i + \nu_b^i x_b^i] + \sum_{j \in \mathcal{T}} \left[s_b^j z_b^j + \sum_{i \in \mathcal{R}} c_b^{ij} y_b^{ij} \right] \right) \quad (2.7a)$$

$$\text{subject to } x_b^i \leq \kappa_b^i u_b^i \quad b \in \mathcal{B}, i \in \mathcal{R} \quad (2.7b)$$

$$\sum_{j \in \mathcal{T}} \delta_b^j y_b^{ij} \leq \sum_{h \in \mathcal{A}_r(\mathcal{B}), b} x_h^i + \sum_{h \in \mathcal{A}_{0,a}(r(\mathcal{B}))} x_h^i \quad b \in \mathcal{B}, i \in \mathcal{R} \quad (2.7c)$$

$$\sum_{i \in \mathcal{R}} y_b^{ij} + z_b^j = 1 \quad b \in \mathcal{B}, j \in \mathcal{T} \quad (2.7d)$$

$$x_b^i \geq 0, u_b^i, z_b^j, y_n^{ij} \in \{0, 1\} \quad b \in \mathcal{B}, i \in \mathcal{R}, j \in \mathcal{T} \quad (2.7e)$$

2.4.2 Multistage Asset Management Problem (MSAM)

Our second test problem is a multistage financial planning problem, which seeks to optimize investment strategies for a set of assets $\mathcal{A} = \{1, \dots, I\}$. The objective is to maximize expected wealth by trading assets with random returns over a finite discrete planning horizon. We extend the formulation presented in (Mulvey and Shetty 2004) by including borrowing limits and investment diversification constraints. Furthermore, the value of the asset return is available at the time of the investment decision in (Mulvey and Shetty 2004), whereas in our model this value is only observed in the subsequent stage when portfolio re-balancing decisions are made.

Investment decisions are shaped by the risk attitude of the decision maker (Infanger 2006). Several approaches including the classical mean-variance model have been proposed in the literature. We employ the additive quadratic downside utility function to penalize the objective function when wealth falls below a prescribed target (Infanger 2006). A multistage stochastic programming formulation is given by:

$$[\text{MSAM}] \quad \max \quad \sum_{n \neq 0, n \in \mathcal{N}} q_n \delta_n [W_n - \alpha_1 \max\{0, \Psi_n - W_n\}] - \quad (2.8a)$$

$$\sum_{n \neq 0, n \in \mathcal{N}} \alpha_2 (\max\{0, \Psi_n - W_n\})^2$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{A}} x_0^i = B \quad (2.8b)$$

$$x_n^i = r_n^i x_{a(n)}^i + y_n^i (1 - \sigma_n^i) - z_n^i \quad n \in \mathcal{N} \setminus \{0\}, i \in \mathcal{A} \setminus \{1\}, \quad (2.8c)$$

$$x_n^1 = r_n^1 x_{a(n)}^1 + \sum_{i \in \mathcal{A} \setminus 1} [z_n^i (1 - \sigma_n^i) - y_n^i] - b_{a(n)} (1 + \beta_{a(n)}) + b_n, \quad n \in \mathcal{N} \setminus \{0\}, \quad (2.8d)$$

$$b_n \leq \kappa_2 \left[\sum_{i \in \mathcal{A}} r_n^i x_{an}^i - b_{a(n)} (1 + \beta_{a(n)}) \right] \quad n \in \mathcal{N} \setminus \{0\}, \quad (2.8e)$$

$$W_n = \sum_{i \in \mathcal{A}} x_n^i - b_n \quad n \in \mathcal{N} \setminus \{0\}, \quad (2.8f)$$

$$x_n^i \leq \kappa_1 \sum_{j \in \mathcal{A}} x_n^j \quad n \in \mathcal{N}, i \in \mathcal{A}, \quad (2.8g)$$

$$W_n, x_n^i, y_n^i, z_n^i, b_n \geq 0 \quad n \in \mathcal{N}, i \in \mathcal{A} \quad (2.8h)$$

The first asset with index $i = 1$ represents cash. The variables W_n and b_n denote the wealth and the amount of cash borrowed at node n , respectively. x_n^i , y_n^i , and z_n^i denote the total value of asset i , the amount of cash invested in asset i , and the amount of

cash earned by selling asset i at scenario tree node $n \in \mathcal{N}$, respectively. The parameters α_1 and α_2 denote the linear and quadratic downside risk coefficients, κ_1 the maximum fraction of total assets that can be assigned to any single asset, κ_2 the maximum fraction of wealth that can be borrowed, W_0 denotes the initial wealth, and for scenario tree node $n \in \mathcal{N}$, Ψ_n , δ_n , and β_n denote the target wealth, discount rate, and uncertain borrowing rate at n , respectively, r_n^i and σ_n^i denote the uncertain return and uncertain transaction cost, respectively, of asset i at node n . Note that we assume symmetric transaction costs (i.e., cost of selling equals cost of buying).

The Objective function (2.8a) maximizes the expected discounted wealth minus the penalty for wealth falling below a target wealth. Constraints (2.8b), (2.8g), and (2.8e) enforce the initial budget, investment diversification, and borrowing limit restrictions, respectively. In particular, Constraints (2.8g) state that the holding of an asset at any node cannot be more than a certain fraction of total assets. Finally, constraints (2.8e) limit borrowing to a certain fraction of the wealth at the beginning of the period. Constraints (2.8c) and (2.8d) are flow balance equations, and (2.8f) determine the wealth at each stage of the planning horizon.

The subproblem formulation for a bundle is given below, with b and x as lasting variables. B' is the upper bound for the wealth at the parent node of the root node which is obtained by solving a subtree problem on nodes $\mathcal{A}_{0,a(r(\mathcal{B}))}$ with the objective of maximizing the wealth at the leaf node.

$$\max \sum_{b \neq 0, b \in \mathcal{B}} q_b \delta_b \left[W_b - \alpha_1 \max\{0, \Psi_b - W_b\} - \alpha_2 (\max\{0, \Psi_b - W_b\})^2 \right] \quad (2.9a)$$

$$\text{s.t. } \sum_{i \in \mathcal{A}} x_{a(r(\mathcal{B}))}^i = B' \quad (2.9b)$$

$$x_b^i = r_b^i x_{a(b)}^i + y_b^i (1 - \sigma_b^i) - z_b^i \quad b \neq 0 \in \mathcal{B} \setminus r(\mathcal{B}), \quad i \in \mathcal{A} \setminus \{1\}, \quad (2.9c)$$

$$x_{r(\mathcal{B})}^i = r_{r(\mathcal{B})}^i x_{a(r(\mathcal{B}))}^i + y_{r(\mathcal{B})}^i (1 - \sigma_{r(\mathcal{B})}^i) - z_{r(\mathcal{B})}^i \quad i \in \mathcal{A} \setminus \{1\}, \quad (2.9d)$$

$$x_b^1 = r_b^1 x_{a(b)}^1 + \sum_{i \neq 1} \left[z_b^i (1 - \sigma_b^i) - y_b^i \right] - b_{a(b)} (1 + \beta_{a(b)}) + b_b, \quad b \in \mathcal{B} \setminus \{0\}, \quad (2.9e)$$

$$x_{r(\mathcal{B})}^1 = r_{r(\mathcal{B})}^1 x_{a(r(\mathcal{B}))}^1 + \sum_{i \neq 1} \left[z_{r(\mathcal{B})}^i (1 - \sigma_{r(\mathcal{B})}^i) - y_{r(\mathcal{B})}^i \right] - b_{a(r(\mathcal{B}))} (1 + \beta_{a(r(\mathcal{B}))}) + b_{r(\mathcal{B})} \quad (2.9f)$$

$$b_b \leq \kappa_2 \left[\sum_{i \in \mathcal{A}} r_b^i x_{a(b)}^i - b_{a(b)} (1 + \beta_{a(b)}) \right] \quad b \in \mathcal{B} \setminus \{0\} \quad (2.9g)$$

$$b_{r(\mathcal{B})} \leq \kappa_2 \left[\sum_{i \in \mathcal{A}} r_{r(\mathcal{B})}^i x_{a(r(\mathcal{B}))}^i - b_{a(r(\mathcal{B}))} \left(1 + \beta_{a(r(\mathcal{B}))} \right) \right] \quad (2.9h)$$

$$W_b = \sum_{i \in \mathcal{A}} x_b^i - b_b \quad b \in \mathcal{B} \setminus \{0\}, \quad (2.9i)$$

$$x_b^i \leq \kappa_1 \sum_{j \in \mathcal{A}} x_b^j \quad b \in \mathcal{B}, i \in \mathcal{A}, \quad (2.9j)$$

$$x_b^i, y_b^i, z_b^i, b_b \geq 0 \quad b \in \mathcal{B}, i \in \mathcal{A} \quad (2.9k)$$

2.4.3 Instance Generation

We randomly generate instances in various sizes. We consider symmetric scenario trees where the number of child nodes emanating from each non-leaf node is the same for all time stages. These two parameters determine the size of a scenario tree and its number of scenarios; a (T, \mathbf{r}) instance would have $(\mathbf{r}^T - 1)/(\mathbf{r} - 1)$ nodes and \mathbf{r}^{T-1} scenarios. We refer to an instance as *small* when the number of nodes in its corresponding scenario tree is $N < 10e4$ and large otherwise.

We populate data for MCAP instances as in (Sandikçi and Özaltın 2017) which solves instances with up to 100 million scenarios. We set $\kappa_n^i = 1$, ν_n^i is uniformly distributed between 5 and 10, denoted by $\nu_n^i \sim U[5, 10]$, $\phi_n^i \sim U[25, 50]$, $s_n^j \sim U[500, 1000]$, $c_n^{ij} \sim U[5, 10]$ and demand $\delta_n^j \sim U[0.5, 1.5]$ for every $i \in \mathcal{R}, j \in \mathcal{T}, n \in \mathcal{N}$.

In the MSAM problem, we consider the annual investment decisions for five asset classes: cash, US stocks, international stocks, corporate bonds, and government bonds (Infanger 2006). We model the asset returns using a multivariate normal distribution with means, standard deviations, and the correlation matrix provided in (Infanger 2006). We set the risk aversion coefficients $\alpha_1 = 1,000$ and $\alpha_2 = 2,000$, the discount factor $\delta_n = (0.99)^{t_n}$ corresponding to 1% per year, where t_n denotes the time stage of scenario tree node $n \in \mathcal{N}$, and the target wealth in each stage $\Psi_n = W_0(1.02)^{t_n}$, reflecting the desired minimum return of 2% per year. Borrowing is not considered in (Infanger 2006), so we assume that the borrowing rate β_n follows the same distribution as cash, and it is perfectly correlated with cash. We also set $W_0 = 100$, $\sigma_n \sim U[0.02, 0.05]$, $\kappa_1 = 0.3$, and $\kappa_2 = 0.2$.

2.4.4 Analysis of the Results

We analyze the performance of our reformulation on two sets of small and large instances with different number of periods and realizations of random variables. For small-size instances, we set the time limit to 3,600 seconds for the extensive form (EF), and 1,800 seconds for lower and upper bounds. Our preliminary results showed that solving the extensive form for large-size instances usually requires a memory far exceeding that of our machine, so we only focus on lower and upper bound calculations for which we set the time limit to 3,600 seconds.

Table 2.2 and Table 2.4 present the numerical results for 25 instances of MCAP and MSAM problems, respectively. The first column shows the number of periods and number of branches per each non-scenario node in the tree. The extensive form (EF) solution, upper bound (UB), and four lower bounds (LB) with different patterns form the columns of the table. For all solutions, the CPU time is given inside parenthesis if the solution obtained within the time limit, otherwise it is indicated by time-out (*TO*). To gain insight into the bounds' quality, we give the best found EF solution when the CPU time exceeds the time limit. Number in all tables are rounded down.

To gain an insight into the performance of algorithms with regard to different patterns, we calculate gaps for both problems. MCAP is a minimization problem so each pattern potentially generates a lower bound, whereas MSAM is a maximization problem where a pattern produces an upper bound. We calculated the lower and upper bound optimality gap as:

$$\text{LB gap: } 100 \times \frac{(EF - LB)}{EF} \qquad \text{UB gap: } 100 \times \frac{(UB - EF)}{EF}$$

Table 2.3 and Table 2.6 compare the gaps for different patterns presented in Table 2.2, respectively and Table 2.4. The *inf* notation is used to indicate that the gap is not available. Figure 2.6 and Figure 2.7 illustrate the optimality gap for lower bounds for both MCAP and MSAM instances. As evident by these figures and tables, the two problems behave differently as the diameter of patterns increase. MCAP, generally produces the best LB when $d = 0$, and worst one when $d = 1$. However, the best solution time is attained when $d = 3$, and the worst when $d = 1$. The potential of our reformulation can be seen in LB4 where it establishes quality bounds without spending more than 60 seconds on any instance. The inconsistent behavior of bounds with regard to diameter size can be traced back to MCAP's formulation. Having a diameter $d = 0$ allows bundles

Table 2.2: Numerical results on small-size instances for MCAP

$T \times r$	EF (CPU)	UB (CPU)	LB1 Pattern	LB1 (CPU)	LB2 Pattern	LB2 (CPU)	LB3 Pattern	LB3 (CPU)	LB4 Pattern	LB4 (CPU)
8-2	2678(0)	2708(0)	(0,256)	2677(0)	(1, 256)	2590(1)	(2, 128)	2644(0)	(3, 128)	2661(0)
10-2	2685(7)	2717(0)	(0,512)	2685(6)	(1,512)	2598(41)	(2,256)	2654(3)	(2,256)	2661(0)
12-2	2741(303)	2776(1)	(0,512)	2740(90)	(1,512)	TO	(2,512)	2709(46)	(3,512)	2716(0)
14-2	2786(TO)	2824(8)	(0,512)	2785(157)	(1,512)	2699(1492)	(2,512)	2754(142)	(3,512)	2761(4)
16-2	2824(TO)	2862(55)	(0,512)	2821(684)	(1,512)	TO	(2,512)	2791(1329)	(3,512)	2796(13)
6-3	2514(1)	2523(0)	(0,81)	2512(1)	(1,81)	2427(2)	(2,27)	2486(0)	(3,9)	2476(0)
7-3	2523(20)	2544(0)	(0,243)	2521(25)	(1,243)	2436(52)	(2,81)	2495(0)	(3,27)	2483(0)
8-3	2553(195)	2563(1)	(0,729)	2551(510)	(1,729)	TO	(2,243)	2523(4)	(3,81)	2508(0)
9-3	2578(1017)	2588(6)	(0,729)	2576(866)	(1,729)	TO	(2,729)	2548(195)	(3,243)	2531(0)
5-4	2439(1)	2469(0)	(0,64)	2436(0)	(1,64)	2352(0)	(2,16)	2411(0)	(3,16)	2415(0)
6-4	2449(26)	2465(0)	(0,256)	2446(13)	(1,256)	2362(14)	(2,64)	2419(0)	(3,16)	2418(0)
7-4	2480(523)	2496(1)	(0,256)	2477(142)	(1,256)	2392(156)	(2,256)	2449(6)	(3,64)	2445(0)
8-4	2522(TO)	2524(8)	(0,256)	2504(364)	(1,256)	2419(384)	(2,256)	2475(35)	(3,256)	2468(1)
9-4	2544(TO)	2545(42)	(0,256)	2524(962)	(1,256)	2440(1476)	(2,256)	2496(115)	(3,256)	2487(7)
5-5	2522(6)	2547(0)	(0,125)	2520(2)	(1,125)	2435(3)	(2,25)	2498(0)	(3,5)	2498(0)
6-5	2540(109)	2552(1)	(0,625)	2538(218)	(1,625)	2453(254)	(2,125)	2514(3)	(3,25)	2510(0)
7-5	2564(3386)	2578(5)	(0,625)	2561(667)	(1,625)	TO	(2,625)	2536(50)	(3,125)	2529(0)
8-5	2641(TO)	2599(27)	(0,625)	TO	(1,625)	TO	(2,625)	2558(1580)	(3,125)	2548(10)
5-6	2556(40)	2569(0)	(0,216)	2555(12)	(1,216)	2469(19)	(2,36)	2532(0)	(3,6)	2533(0)
6-6	2589(2177)	2604(1)	(0,216)	2586(61)	(1,216)	2501(83)	(2,216)	2563(5)	(3,36)	2558(0)
7-6	2622(TO)	2629(9)	(0,216)	2611(464)	(1,216)	2526(728)	(2,216)	2588(63)	(3,36)	2579(7)
5-7	2525(106)	2543(1)	(0,343)	2523(35)	(1,343)	2438(110)	(2,49)	2501(0)	(3,7)	2502(0)
6-7	2553(3468)	2571(7)	(0,343)	2551(260)	(1,343)	2465(414)	(2,49)	2527(6)	(3,7)	2524(14)
5-8	2505(299)	2520(1)	(0,512)	2503(93)	(1,512)	2418(414)	(2,64)	2482(1)	(3,8)	2481(0)
6-8	2534(TO)	2544(2)	(0,512)	2525(1344)	(1,512)	TO	(2,64)	2503(11)	(3,8)	2500(60)

to incorporate all periods. This is especially useful when there is no substantial variance between weights of bundles. As diameter increases to $d = 3$, the root bundle controls most of the contribution toward the objective function. This leads to quality bounds as the problem structure captures importance of earlier nodes. Note that increasing the diameter to the extreme of $d = T$ yields the original problem. Diameters such as $d = 1$ reflect the trade-off in bundle solutions between spanning all periods or incorporating all nodes in earlier stages. As such, the cost of losing control over all periods is higher than having all nodes in the root bundle. The upper bound for the problem generates tight bounds as all, but four, of the instances have gap less than a percent.

MSAM reacts quite differently to diameter increase. Although the bound quality is not greatly affected as the diameter increases from 0 to 3, the solution time consistently grows to a point that the bound for some instances such as $9 - 4, 8 - 5, 5 - 7, 6 - 8$, is not attained in the time limit. The performance of the method is generally worse than for MCAP; most instance have more than 10% gap whereas the gap for MCAP is under 4% for all instances. This is because the initial budget plays a vital role in the problem formulation. The root node, which is the only node directly impacted by the budget, becomes crucial for getting a tight bound. While patterns with $d = 0$ do not consider all the contribution of the root node in the solution, the commercial solver is unable to solve larger root bundles. Accordingly, the generated bounds lack the quality of bounds of MCAP. This also applies for the LB quality where its gap is usually above %10.

Table 2.3: Upper and lower bound gap for small-size MCAP instances. Rows corresponds to the actual gap for each class of instances. LB1 and LB2 results in the best bounds with less than 2% for all the instances.

$T \times r$	UB	LB1	LB2	LB3	LB4
8x2	1.11	0.02	3.26	1.25	0.62
10x2	1.18	0	3.25	1.13	0.87
12x2	1.27	0.04	inf	1.15	0.92
6x3	0.34	0.08	3.47	1.12	1.51
7x3	0.8	0.09	3.46	1.14	1.59
8x3	0.37	0.1	inf	1.16	1.78
9x3	0.37	0.1	inf	1.17	1.83
5x4	1.2	0.12	3.58	1.17	0.99
6x4	0.62	0.12	3.56	1.23	1.26
7x4	0.62	0.11	3.53	1.25	1.43
5x5	0.98	0.07	3.46	0.93	0.94
6x5	0.46	0.08	3.44	1	1.19
7x5	0.52	0.12	inf	1.09	1.38
5x6	0.47	0.07	3.4	0.97	0.93
6x6	0.55	0.11	3.39	1	1.2
5x7	0.68	0.08	3.47	0.95	0.93
6x7	0.68	0.09	3.44	1.01	1.14
5x8	0.57	0.09	3.48	0.91	0.96

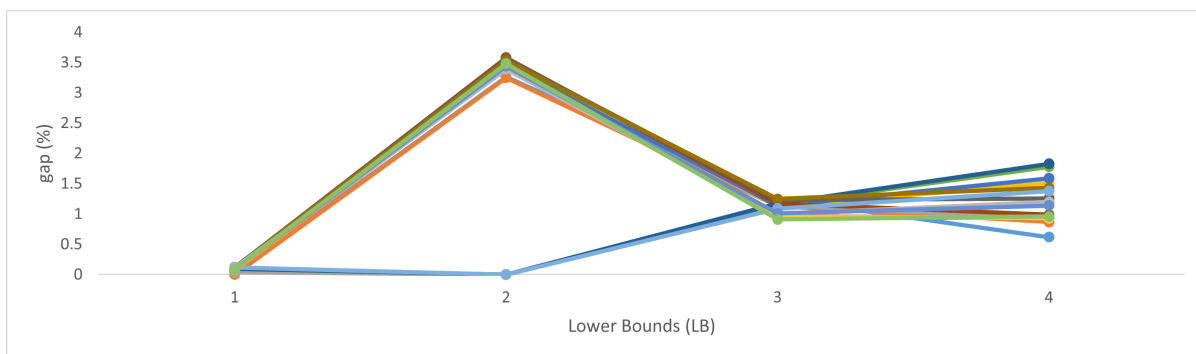


Figure 2.6: Lower bound gap for small-size MCAP instances

Table 2.4: Numerical results for small-size instances of MSAM

$T \times r$	EF(CPU)	LB(CPU)	UB1 Pattern	UB1 (CPU)	UB2 Pattern	UB2 (CPU)	UB3 Pattern	UB3 (CPU)	UB4 Pattern	UB4 (CPU)
8-2	1058(0)	881(0)	(0,256)	1058(0)	(1,256)	1139(0)	(2,128)	1207(0)	(3,128)	1231(50)
10-2	1523(0)	1177(0)	(0,512)	1523(0)	(1,512)	1647(0)	(2,256)	1754(1)	(3,256)	1802(49)
12-2	2100(3)	1534(0)	(0,512)	2383(0)	(1,512)	2383(0)	(2,512)	2431(1)	(3,512)	2511(40)
14-2	2807(23)	1972(2)	(0,512)	3487(1)	(1,512)	3487(0)	(2,512)	3560(12)	(3,512)	3533(101)
16-2	3670(185)	2646(7)	(0,512)	4985(4)	(1,512)	4985(3)	(2,512)	5091(49)	(3,512)	5057(421)
6-3	669(0)	591(0)	(0,81)	716(0)	(1,81)	716(0)	(2,27)	744(12)	(3,9)	755(165)
7-3	846(0)	744(0)	(0,243)	910(0)	(1,243)	910(0)	(2,81)	951(12)	(3,27)	974(154)
8-3	1048(2)	873(0)	(0,729)	112(0)	(1,729)	1127(0)	(2,243)	1181(5)	(3,81)	1218(165)
9-3	1268(10)	1031)	(0,729)	142(0)	(0,729)	1424(0)	(0,729)	1435(15)	(3,243)	1487(153)
5-4	503(0)	454(0)	(0,64)	530(0)	(1,64)	530(0)	(2,16)	553(37)	(3,4)	552(473)
6-4	662(0)	588(0)	(0,256)	701(0)	(1,256)	701(0)	(2,64)	736(39)	(3,16)	744(96)
7-4	840(4)	739(0)	(0,256)	929(0)	(1,256)	929(0)	(2,256)	940(51)	(3,64)	959(486)
8-4	1036(25)	870(2)	(0,256)	1190(1)	(1,256)	1194(1)	(2,256)	1209(208)	(3,256)	1196(487)
9-4	1253(151)	104(11)	(0,256)	150(4)	(1,256)	1507(4)	(2,256)	1527(891)	(3,256)	TO
5-5	505(0)	455(0)	(0,125)	531(0)	(1,125)	531(0)	(2,25)	554(79)	(3,5)	553(959)
6-5	665(2)	590(0)	(0,625)	701(0)	(1,625)	701(0)	(2,125)	737(105)	(3,25)	745(944)
7-5	844(19)	7402)	(0,625)	930(1)	(1,625)	930(1)	(2,625)	941(106)	(3,125)	961(969)
8-5	1040(159)	872(10)	(0,625)	119(6)	(1,625)	1197(5)	(2,625)	1211(489)	(3,125)	TO
5-6	502(1)	454(0)	(0,216)	531(0)	(1,216)	531(0)	(2,36)	552(133)	(3,6)	551(1709)
6-6	662(7)	589(1)	(0,216)	729(0)	(1,216)	729(0)	(2,216)	736(138)	(3,36)	745(1750)
7-6	839(68)	743(7)	(0,216)	963(3)	(1,216)	963(3)	(2,216)	972(938)	(3,36)	TO
5-7	501(1)	454(0)	(0,343)	528(0)	(1,343)	528(0)	(2,49)	550(263)	(3,7)	TO
6-7	662(17)	591(2)	(0,343)	727(1)	(1,343)	727(1)	(2,49)	TO	(3,7)	TO
5-8	503(3)	455(0)	(0,512)	530(0)	(1,512)	530(0)	(2,64)	552(315)	(3,8)	TO
6-8	663(44)	591(4)	(0,512)	729(2)	(1,512)	729(2)	(2,64)	TO	(3,8)	TO

Table 2.5: Gap between LB3 and UB for large-size instances of MCAP

Instance	18-2	20-2	12-2	13-3	10-4	9-5	8-6	7-7	8-7	7-8
LB3-gap	1.53	1.57	2.18	2.2	1.83	1.6	1.21	1.5	1.26	1.44

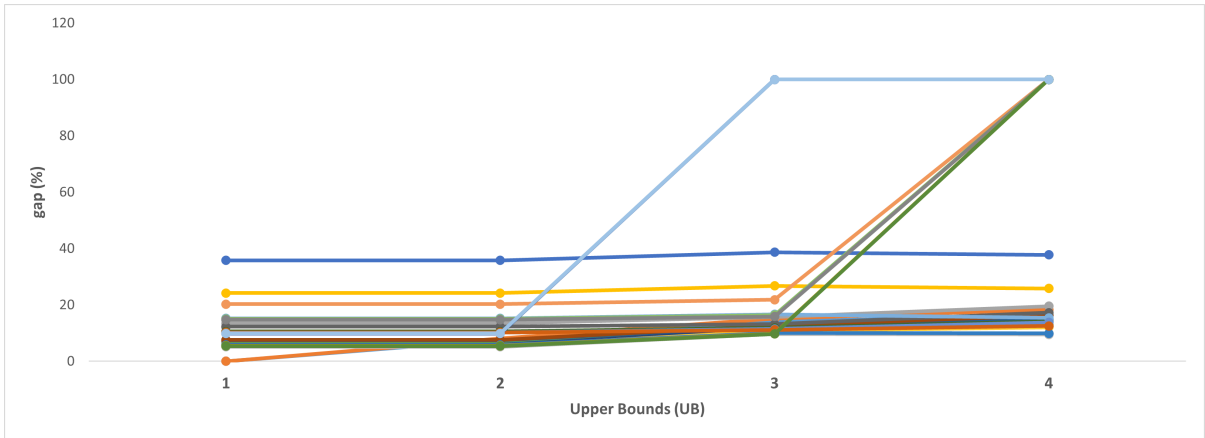


Figure 2.7: Lower bound gap for small-size MSAM instances

Table 2.6: Percentage upper and lower bound gap for small-size MSAM instances.

$T \times r$	LB	UB1	UB2	UB3	UB4
8-2	16.74	0	7.66	14.09	16.38
10-2	22.72	0	8.11	15.13	18.29
12-2	26.94	13.46	13.46	15.72	19.55
14-2	29.76	24.19	24.19	26.8	25.86
16-2	27.89	35.84	35.84	38.73	37.8
6-3	11.57	7	7	11.19	12.96
7-3	12.1	7.49	7.49	12.3	15
8-3	16.71	7.59	7.59	12.74	16.28
9-3	18.06	12.28	12.28	13.18	17.28
5-4	9.81	5.44	5.44	10.02	9.74
6-4	11.11	5.89	5.89	11.28	12.37
7-4	12.04	10.59	10.59	11.89	14.1
8-4	15.98	15.26	15.26	16.7	15.41
9-4	16.48	20.26	20.26	21.82	Inf
5-5	9.94	5.13	5.13	9.76	9.54
6-5	11.3	5.43	5.43	10.74	12.08
7-5	12.29	10.22	10.22	11.47	13.82
8-5	16.13	15.02	15.02	16.41	Inf
5-6	9.56	5.79	5.79	10.08	9.92
6-6	10.94	10.23	10.23	11.17	12.56
7-6	11.42	14.77	14.77	15.87	Inf
5-7	9.28	5.52	5.52	9.77	Inf
6-7	10.7	9.86	9.86	Inf	Inf
5-8	9.47	5.38	5.38	9.73	Inf
6-8	10.86	9.91	9.91	Inf	Inf

Table 2.7: Upper and lower bound values for large-size instances of MCAP

$T \times r$	UB (CPU)	LB1 Pattern	LB1 (CPU)	LB2 Pattern	LB2 (CPU)	LB3 Pattern	LB3 (CPU)
18-2	2881(87)	(0,512)	2863(2993)	(2,512)	2833(1281)	(4,512)	2837(43)
20-2	2923(750)	(0,512)	TO	(2,512)	TO	(4,512)	2877(223)
12-2	2663(47)	(0,729)	TO	(2,729)	2608(3712)	(4,729)	2605(44)
13-3	2684(439)	(0,729)	TO	(2,729)	TO	(4,729)	2625(146)
10-4	2569(109)	(0,256)	TO	(2,256)	2517(584)	(4,256)	2522(25)
9-5	2625(487)	(0,629)	TO	(2,629)	TO	(4,629)	2583(62)
8-6	2647(150)	(0,216)	TO	(2,216)	2610(547)	(4,216)	2615(24)
7-7	2596(74)	(0,343)	2572(1599)	(2,343)	2550(228)	(4,49)	2557(12)
8-7	2620(840)	(0,343)	TO	(2,343)	TO	(4,343)	2587(111)
7-8	2569(298)	(0,512)	TO	(2,512)	2527(2749)	(4,64)	2532(40)

We also test the performance of DEP on 10 large-size instances of MCAP. As Table 2.7 shows, these instances are created based on different time periods and number of realizations. As anticipated, the LB1 with $d = 0$ is unable to establish a bound for 8 instances. This number is 4 for LB2 with $d = 2$ and 0 for LB3 with $d = 4$. Although, LB3 produces slightly worse lower bounds than LB1, its solution time is significantly less. Table 2.5 provides the percentage of between LB3 and UB, and show that the potential of DEP as the gap is generally less than %2. The solution time for LB1 is significant, leading to time-out in most cases. However, Table 2.5 shows that LB3 not only is very fast, but efficient as well.

2.5 Summary

This chapter presents a reformulation for a general multistage stochastic program where the objective and constraints are additive separable functions. The main idea is to divide the scenario tree into subtrees in order to establish lower and upper bounds. Our reformulation is particularly amenable to parallel implementation as it allows the scenario tree to decomposed into multiple independent subproblems, each of which solvable by commercial solvers. The proposed method is broadly applicable to any decomposition methods subject of mild assumptions. To evaluate the performance of different patterns of decomposition, we propose a systematic way to decompose the tree. The method only

requires a keyword, indicating the general way of grouping scenario nodes, and set of parameters for the keyword to generate a decomposition. We test the bounding method on two well known problems from the literature. Computational analysis shows that the method can obtain bounds for very large instances and outperform the best algorithm in the literature (i.e. scenario decomposition.)

Chapter 3

Managing Product Transitions in Semiconductor Manufacturing: A Bilevel Programming Approach

Abstract:

We study the problem of managing product transitions in semiconductor manufacturing firms, which requires the coordination of product development and manufacturing units. We model the hierarchical and decentralized nature of the product transition process using a mixed-integer bilevel program. Acting as the leader, corporate management maximizes profit over a finite planning horizon. There are two interdependent followers, manufacturing and engineering units, who must share factory capacity. In addition, the manufacturing unit cannot fulfill the demand for new products until the engineering unit completes all development activities. We model this interdependency as a generalized Nash equilibrium at the lower level of the proposed bilevel model. We then present a reformulation where the interdependency between the engineering and manufacturing units is resolved through the coordination of the leader, and derive solution methods using the constraint and column generation approach. Our computational experiments show that the proposed method can solve realistic instances to optimality in a reasonable time. We assess the impact of key problem parameters on the decision units considered in the model, and provide managerial insights into how the allocation of decision making authority between corporate leadership and functional units affects solution structure and performance. This chapter presents the first exact solution algorithm to mixed-integer bilevel programs with interdependent followers. The proposed bilevel model and solution approach provide a flexible framework

to study decentralized, hierarchical decision making problems in organizational design.

3.1 Introduction

Product transitions involve the introduction of new products to replace those that are currently being sold to customers in response to ever-changing customer preferences (Klastorin and Tsai 2004), shorter product life cycles (Wu et al. 2009) and increasing global competition (Wu et al. 2005). Effective management of product transitions can bring significant competitive advantage to a firm, while poor decisions can have serious adverse business consequences. A central challenge during product transitions is the coordination of the product development process, which creates new products that can be manufactured efficiently, and the supply chain that manufactures and distributes them. The product development process often requires access to manufacturing capacity to fabricate prototypes to assess their manufacturability and functionality, while the supply chain requires timely delivery of new products with enhanced features and reduced costs to stay competitive in the market. Although the challenges faced by firms in managing product transitions have been addressed in the literature (Bilg ner and Erhun 2010; Lim and Tang 2006), most previous studies share several shortcomings:

- They treat the firm as a single decision entity with complete information about all aspects of its operations. However, as firms grow in size and diversify their product portfolio, operational control decisions become distributed among different functional groups, rendering detailed centralized planning impractical. The critical resource allocation decisions and domain knowledge are distributed among different functional groups within the firm such as product divisions and manufacturing units who are trying to reconcile their local, potentially conflicting, objectives with those of the corporate management (Bansal et al. 2020a).
- They ignore the hierarchy of decision makers involved in managing product transitions. In practice, upper-level management addresses strategic\tactical planning, resulting in directives to the functional groups who must make and implement operational decisions. This hierarchy is especially important for firms operating in multiple markets such as large semiconductor manufacturing companies.
- They treat product transitions in isolation from routine operations, generally neglecting their impact on products that are not directly involved in the transition. However, high-technology firms with broad product portfolios, such as large semi-

conductor manufacturers, must often engage in simultaneous product transitions in several of the markets they serve, while the highly capital-intensive nature of high-technology production requires them to share expensive manufacturing facilities. Hence new product introductions can adversely affect other products with whom they share manufacturing, development or marketing resources (Ulrich and Eppinger 2016).

Understanding the general information flow and hierarchical relation between the decision-making units involved in production transitions is essential to developing more realistic models that can support this highly important but also highly complex decision process. In this chapter, we focus on semiconductor manufacturing firms, and present bilevel models that represent the decentralized, hierarchical environment in which product transitions take place. Although semiconductor manufacturing firms differ in the specifics of their high-volume manufacturing processes and product development activities, the knowledge and data required for successful introduction of new products and retirement of older ones are usually distributed across three principal units: 1) corporate management (CORP), whose leadership role is to direct the company toward achieving its strategic business goals; 2) one or more product engineering (ENG) units each of which is responsible for all activities related to new product development for a specific market segment, including market research, design, implementation, and prototype testing; and 3) a manufacturing (MFG) organization responsible for the factories that manufacture existing products for sale and prototypes for products under development by the ENG units.

CORP communicates desired production quantities to MFG based on demand. The ENG units determine specifications for new products based on market research, and then develop these product specifications into a detailed design. Once a product design has been tested as far as possible in software, the prototypes are fabricated to fully debug the new designs and assess their manufacturability. Hence, the ENG units must have access to factory capacity to ensure timely fabrication of prototypes. The factory capacity used by the ENG units reduces the capacity available for revenue-generating products that meet current customer demand, but is essential to maintain a profitable product pipeline in the future.

Optimizing corporate objectives by effectively coordinating the functional units in-

volved in product transitions is extremely challenging. The individual decision problems of the MFG and ENG units are computationally difficult to solve even in a completely deterministic environment. We propose a bilevel model where CORP, acting as a leader, seeks to maximize its profit over a finite planning horizon subject to the decisions of the MFG and ENG units, who act as followers. Unlike traditional bilevel programs, the decisions of the two followers in the proposed model are interdependent; the ENG and MFG units share factory capacity, and MFG is dependent on ENG for the development of new products to meet future demand. The contributions of this chapter are as follows:

- We propose a bilevel integer programming model to effectively coordinate product transitions in a large decentralized semiconductor manufacturing firm by considering the technological constraints and objectives of the functional units involved in product transitions. To the best of our knowledge, this is the first bilevel model developed for product transitions in the literature.
- The proposed model is also novel from the bilevel programming perspective as it considers two interdependent followers. To the best of our knowledge, this chapter presents the first solution algorithm for mixed-integer bilevel programs with interdependent followers.
- We present two single-level reformulations of the bilevel model, and develop an efficient solution approach based on these reformulations. We evaluate the performance of the solution approach through extensive computational experiments, and provide policy insights by analyzing the implications of decision hierarchy and key model parameters.

The remainder of this chapter is organized as follows. We review the literature on product transitions and production planning in semiconductor manufacturing, as well as the state of the art in bilevel programming and generalized Nash equilibrium problem, in Section 3.2. We formulate a bilevel mixed-integer model for product transitions in Section 3.3. We develop a solution approach in Section 3.4, and present the results of computational experiments in Section 3.5. We conclude the chapter and discuss future research directions in Section 3.6.

3.2 Literature Review

3.2.1 Product Transitions

A growing body of researchers have examined product transitions (Ferrer and Swaminathan 2006; Bİlgíner and Erhun 2010). Li et al. (2013) and Bhaskaran et al. (2011) study capacity planning for product transitions considering demand uncertainty, the effect of competition, and service-level requirements. The impact of initial investment on quality improvement and time-to-market are examined in Wu et al. (2009). A rich body of work treats customer behavior as one of the main factors affecting product transition decisions such as the optimal timing of product introduction (Druehl et al. 2009; Liao and Seifert 2015) and product rollover strategies (Liang et al. 2014; Lobel et al. 2015). Klastorin and Tsai (2004) develop a game-theoretical model to capture the interactions among pricing, timing and product design when entering a new market. Koca et al. (2010) analyze the impact of pre-announcement and inventory decisions on the demand of new products.

While this extensive body of work addresses many aspects of the product transition problem and provides useful insights, it fails to consider several important aspects of the problem addressed in this chapter, particularly the hierarchy of decision makers, the interactions between them and the complex technological constraints that an implementable solution must satisfy. It also fails to consider the impact of product transitions on other products that share capacity with the new products but are not in transition themselves. Thus our chapter presents a novel direction for studying this complex and important business problem.

3.2.2 Production Planning in Semiconductor Manufacturing

A number of authors have addressed decentralized production planning in semiconductor manufacturing. Karabuk and Wu (2003) propose a multi-stage stochastic programming model under demand and yield uncertainty with a manufacturing unit and several product managers. In a subsequent chapter (Karabuk and Wu 2005) consider corporate headquarters and product managers as the units involved in the capacity allocation process, and propose a game-theoretic approach to elicit private information from the

product managers to maximize expected corporate profit. More recently, Bansal et al. (2020a) consider a simplified version of the product transition problem considered in this chapter. They develop two iterative combinatorial auction schemes based on Lagrangian Relaxation and Column Generation that seek to coordinate negotiations over factory capacity between MFG, acting as the auctioneer, and ENG units who bid for factory capacity. Bansal et al. (2020a) do not enforce a strict decision hierarchy, but instead employ a decentralized iterative combinatorial auction to coordinate the participants towards an implementable solution.

3.2.3 Generalized Nash Equilibrium Problem

The Generalized Nash Equilibrium Problem (GNEP) is a non cooperative game in which each player's strategy set depends on the strategies of other players (Facchinei and Kanzow 2010). From an optimization perspective, GNEP is a decentralized model where decision makers share a set of constraints referred to as *coupling constraints*. The outcome of a GNEP is a set of *equilibria*, a collection of strategies from which no decision maker has incentive to deviate unilaterally. GNEP has many applications ranging from pricing in telecommunication networks (Altman and Wynter 2004a) to electricity market analysis (Le Cadre et al. 2020) and transportation problems (Zhou et al. 2005; Stein and Sudermann-Merx 2018; Sagratella et al. 2020). Despite its diverse applications, solution methods for GNEP are generally confined to cases where players control a set of continuous variables (Dreves et al. 2011; Facchinei et al. 2014; Aussel and Sagratella 2017).

In recent years there have been some attempts to study GNEP with mixed-integer variables. Sagratella (2017) consider generalized potential Nash games, a special case of GNEP where players (unknowingly) optimize the same objective function over the aggregated feasible strategy set of all players. Sagratella (2019) show that, under mild assumptions, the set of equilibrium points of a GNEP with mixed-integer variables and linear coupling constraints is finite and propose algorithms to generate all possible equilibria. More recently, Sagratella et al. (2020) propose a mixed-integer GNEP to address the noncooperative fixed charge transportation problem with different cost structures.

3.2.4 Bilevel Programming

Bilevel programming (BP) provides a powerful tool for modeling hierarchical decision making problems in which the outcome of decisions by an upper-level authority (the leader) is affected by the response from a lower-level entity (follower) who seeks to optimize its own objective function (Bard 1998). From the perspective of game theory, BP models a static Stackelberg game between a leader and a follower (Bard 1998; Dempe 2002). An important feature of BP is that the feasible region the decision problem in each level may be impacted by variables controlled in the other level. This embedded hierarchy renders BP a suitable approach to address the problem studied in our chapter.

BPs can be generally classified as bilevel linear programs (BLP) that have no integer variables, and mixed-integer bilevel linear programs (MIBLP) that have integer variables in at least one decision level. BLPs are non-convex and strongly NP-hard (Colson et al. 2007), although they can be reformulated as a single level model by enforcing the optimality of the follower through Karush-Kuhn-Tucker (KKT) conditions (Bertsimas and Tsitsiklis 1997). The resulting single-level problem, which is usually nonlinear, can be solved using methods such as branch-and-bound (Bard and Moore 1990; Hansen et al. 1992), branch-and-cut (Audet et al. 2007), Bender’s decomposition (Nishi et al. 2011; Saharidis and Ierapetritou 2009), and penalty function methods (Campelo et al. 2000).

Bard and Moore (1990, 1992), the first to consider discrete variables in the lower-level problem, apply a branch-and-bound algorithm and establish several properties for pure integer BPs. DeNegre and Ralphs (2009) extend this work to develop a branch-and-cut algorithm. Wang and Xu (2017) propose an algorithm for pure integer BPs that removes bilevel infeasible solutions by disjunctive cuts.

Xu and Wang (2014) consider a BP with a mixed-integer lower level and a pure integer upper level with bounded variables. They propose a branch-and-bound algorithm where branching occurs on the slack variables of the follower’s constraints. Lozano and Smith (2017) extend this work by employing a single-level value function reformulation. Fischetti et al. (2017, 2018) propose a branch-and-cut algorithm for MIBLPs by generating intersection cuts. Yue et al. (2019) develop a projection-based approach for MIBLPs that first reformulates the problem into a single-level equivalent, and then decomposes it by implicitly enumerating the follower’s integer variables. The applications of MIBLPs arise in transportation (Arslan et al. 2018), production planning (Lukač et al. 2008),

defense and homeland security (Aksen and Aras 2012), electricity markets (Lavigne et al. 2000), product introduction (Hemmati and Smith 2016), healthcare (Özaltın et al. 2018), and supply chain (Yue and You 2017).

The limited research on BPs with multiple followers mostly considers BLPs. Shi et al. (2007) study a BLP with multiple followers who simultaneously determine a set of common variables. Calvete and Galé (2007) propose a method to transform a problem with multiple independent followers to a BLP with a single follower whose objective function is the sum of the objective functions of all followers. Calvete et al. (2019) consider a nonlinear BP with multiple followers in a rank pricing problem, and present a nonlinear single level equivalent formulation. The model is then solved with a branch-and-cut algorithm using linearization techniques and valid inequalities. Tavashioğlu et al. (2019) develop a value function-based approach for MIBLPs with multiple independent followers. Their approach assumes that all followers have the same constraint matrix, and the leader’s variables impacting the feasible regions of the followers can only take integer values. To the best of our knowledge, BPs with interdependent followers and discrete variables in both levels have not been considered in the literature.

3.3 Model Formulation

We present a bilevel model of the product transition management problem to capture its hierarchical and decentralized nature. CORP acts as a leader, seeking to maximize its total profit over a finite planning horizon $\mathcal{T} = \{1, \dots, T\}$. Profit is earned by selling the products manufactured by the MFG unit. The MFG and ENG units are two interdependent followers, responding to the decisions of the CORP in a manner that optimizes their local objective functions. They play a generalized Nash equilibrium game over the usage of factory capacity. Although we consider one MFG and one ENG unit in the lower level of the proposed model, our approach remains valid for more than two followers, however the computational burden is expected to increase with the number of followers.

Figure 3.1 depicts the information exchange between the CORP, MFG and ENG units. CORP determines when to release new products to MFG for sale in the market. New products cannot be released to MFG unless ENG completes their developments, rendering MFG dependent on the decisions of ENG. On the other hand, ENG cannot

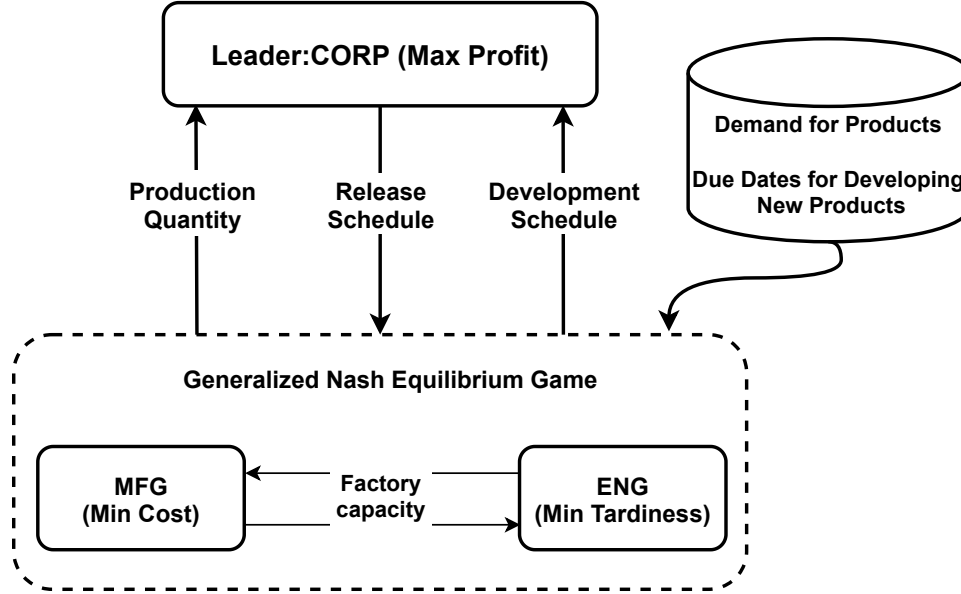


Figure 3.1: Schematic view of the proposed bilevel model with CORP as the leader and MFG and ENG as interdependent followers who play a generalized Nash equilibrium game over factory capacity. Arrows represent “decision variables” communicated between CORP and the followers. The demand estimates and due date for development of products are parameters made available for the followers from outside sources.

complete the development of new products without timely access to sufficient factory capacity for prototype fabrication. This situation leads to competition between MFG and ENG over factory capacity. As indicated by the solid black arrows in Figure 3.1, once MFG and ENG reach an equilibrium over the usage of factory capacity, MFG communicates the production quantities to the CORP. We refer to this model as the Bilevel Product Transition Model with interdependent followers (BPTM-Nash).

Let N denote the set of all products, and $P \subset N$ the set of new products currently under development. Table 3.1 summarizes the model parameters, and Table 3.2 presents the upper- and lower-level decision variables. CORP specifies the value of the upper-level variable $Y_{pt} \in \{0, 1\}$ to release new product $p \in P$ to MFG at the beginning of period t . Thus $Y \equiv \{Y_{p \in P, t \in T}\}$ represents the new product *release schedule*. Given a product release schedule Y , we denote the generalized Nash equilibrium game between MFG and ENG in the lower-level, as well as its solution set by $\text{GNEP}(Y)$. Let \mathcal{X} denote the tuple

Table 3.1: Parameters of the bilevel product transition model

Parameter	Description
D_{nt}	Demand for product n in period t
π_{nt}	Marginal profit for product n in period t
r_{nt}	Marginal production cost of product n in period t
b_{nt}	Marginal backorder cost of product n in period t
H_{pt}	Capacity required for prototype fabrication of product $p \in P$ in period t
δ_p	Due date for completing the development of new product $p \in P$
w_p	Per period tardiness cost for product $p \in P$
h_{nt}	Unit inventory holding cost for product n in period t
C_t	Factory capacity in period t

Table 3.2: Decision variables of the bilevel product transition model

(leader)	Y_{pt}	1 if product p is released to MFG at period t , 0 otherwise (Y is referred to as “product <i>release</i> schedule”)
MFG (follower)	B_{nt} X_{nt} I_{nt}	Backorder of product n in period t Fraction of factory capacity assigned to product n in period t Inventory level of product n in period t
ENG (follower)	F_t V_p Z_{pt}	Fraction of factory capacity allocated to ENG in period t Tardiness in the development of product p 1 if product p is developed in period t , 0 otherwise (Z is referred to as “product <i>development</i> schedule”)

(X, B, I, F, Z, V) . We formulate the bilevel BPTM-Nash model as follows:

$$\text{[BPTM-Nash]} \quad \max \sum_{n \in N} \sum_{t \in \mathcal{T}} \pi_{nt} (D_{nt} + B_{n,t-1} - B_{nt}) \quad (3.1a)$$

$$\text{subject to } Y_{p,t-1} \leq Y_{pt} \quad t \in \mathcal{T} \setminus \{1\} \quad (3.1b)$$

$$Y_{pt} \leq \sum_{\tau \leq t} Z_{p\tau} \quad p \in P, t \in \mathcal{T} \quad (3.1c)$$

$$\mathcal{X} \in \text{GNEP}(Y) \quad (3.1d)$$

The upper-level objective function (3.1a) maximizes the total profit of the CORP. The demand D_{nt} is a constant that can be removed from (3.1a). We keep it however to ensure that the objective function denotes the profit earned by satisfying demand. Constraints (3.1b) ensure that once a new product is released to MFG in period t , it can be manufactured in all subsequent periods. Constraints (3.1c) ensure that the development of new product $p \in P$ is completed in period $\tau \leq t$, i.e. $Z_{p\tau} = 1$ before its release to MFG in period t . Constraint (3.1d) ensures that \mathcal{X} is a solution to the generalized Nash equilibrium game taking place between MFG and ENG in the lower level of the bilevel model. BPTM-Nash follows an optimistic approach, allowing CORP to select the most advantageous equilibrium of $\text{GNEP}(Y)$ when there is more than one.

We formulate the MFG's problem in $\text{GNEP}(Y)$ as the following linear program:

$$\text{[MFG}(F, Y)] \quad \min \sum_{n \in N} \sum_{t \in \mathcal{T}} (h_{nt} I_{nt} + r_{nt} X_{nt} C_t + b_{nt} B_{nt}) \quad (3.2a)$$

$$\text{subject to } F_t + \sum_{n \in N} X_{nt} = 1 \quad t \in \mathcal{T} \quad (3.2b)$$

$$I_{nt} = I_{n,t-1} + X_{nt} C_t - (D_{nt} + B_{n,t-1} - B_{nt}) \quad n \in N, t \in \mathcal{T} \quad (3.2c)$$

$$X_{pt} \leq Y_{pt} \quad p \in P, t \in \mathcal{T} \quad (3.2d)$$

$$\sum_{n \in N} X_{nt} \leq 1 \quad t \in \mathcal{T} \quad (3.2e)$$

$$X_{nt}, I_{nt}, B_{nt} \geq 0 \quad n \in N, t \in \mathcal{T} \quad (3.2f)$$

The MFG's objective function (3.2a) minimizes the sum of production, inventory holding and backorder costs over the planning horizon. Coupling constraints (3.2b) ensure that the factory capacity allocated to the ENG for prototype fabrication, i.e., F_t , and that used by MFG to meet demand, i.e., $C_t \sum_{n \in N} X_{nt}$, is equal to the available capacity in period t .

Note that the F_t variables in (3.2b) are controlled by the ENG. Thus the feasible region of the MFG problem as a follower in the lower level of BPTM-Nash depends on the decisions of both the leader CORP, through the Y variables, and those of the other follower ENG. Constraints (3.2c) enforce inventory balance across time periods, while (3.2d), through the upper-level variable Y , ensure that new products can be manufactured if they are released to MFG. Without loss of generality, we assume that $N \supset P$, i.e., there is at least one current product, in order to ensure that the MFG(F, Y) is feasible for any given (F, Y) tuple. If there is no such current product, we can create a dummy product with zero demand. Any factory capacity assigned to the dummy product would then represent the unused factory capacity allocation of the MFG. Finally, constraints (3.2e) impose the factory capacity limit in each period. For a given Y , we will denote constraints (3.2c) - (3.2f) as **MFG-primal-feasibility** (Y).

We formulate ENG's problem in GNEP(Y) as the following mixed-integer program:

$$[\text{ENG}(X)] \quad \min \sum_{p \in P} w_p V_p \quad (3.3a)$$

$$\text{subject to } F_t + \sum_{n \in N} X_{nt} = 1 \quad t \in \mathcal{T} \quad (3.3b)$$

$$V_p \geq t(1 - \sum_{\tau < t} Z_{p\tau}) - \delta_p \quad p \in P, t \in \mathcal{T} \quad (3.3c)$$

$$\sum_{p \in P} H_{pt} Z_{pt} \leq C_t F_t \quad t \in \mathcal{T} \quad (3.3d)$$

$$\sum_{t \in \mathcal{T}} Z_{pt} \leq 1 \quad p \in P \quad (3.3e)$$

$$V_p \geq 0, Z_{pt} \in \{0, 1\} \quad p \in P, t \in \mathcal{T} \quad (3.3f)$$

The ENG's objective function (3.3a) minimizes the total weighted tardiness of new products in development as calculated in (3.3c). The coupling constraints (3.3b) also appear in MFG's problem. Constraints (3.3d) ensure that the factory capacity used for product development does not exceed F_t in each period t . Note that this constraints allows idle factory capacity as the ENG does not have to use all of the allocated capacity. Finally, constraints (3.3e) ensure that a new product is developed in a single period. This assumption can be relaxed by considering multi-period product development; the proposed bilevel framework would still be applicable. For a given F , we refer to constraints (3.3c) - (3.3f) as **ENG-feasibility** (F).

3.4 Solution Approach

We first show that there are multiple equilibrium solutions to the generalized Nash equilibrium game between the MFG and ENG in the lower level for any given product release schedule by the CORP.

Lemma 3.4.1. *For any \hat{Y} , there is an equilibrium of $GNEP(\hat{Y})$ corresponding to any nonnegative factory capacity allocation to ENG, i.e., $\hat{F}_t \geq 0, \forall t \in \mathcal{T}$.*

Proof. Let $(\hat{X}, \hat{B}, \hat{I})$ be the optimal response of the MFG(\hat{F}, \hat{Y}). Note that $F_t = 1 - \sum_{n \in N} \hat{X}_{nt} = \hat{F}_t, \forall t \in \mathcal{T}$ in any feasible strategy of the ENG(\hat{X}). Let $(\hat{F}, \hat{Z}, \hat{V})$ be the optimal response of the ENG(\hat{X}). Then, $\hat{\mathcal{X}} = (\hat{X}, \hat{B}, \hat{I}, \hat{F}, \hat{Z}, \hat{V})$ is an equilibrium of $GNEP(\hat{Y})$. \square

Given a product release schedule \hat{Y} as an upper-level decision, we can merge the decision problems of the MFG and ENG into a single problem by taking a linear combination of their respective objective functions and enforcing all of their constraints together. The optimal solution of this problem would be an equilibrium of $GNEP(\hat{Y})$ that does not optimize the profit of the leader. However, as BPTM-Nash takes an optimistic approach, after deciding on the product release schedule Y , the CORP should be able to select an equilibrium of the lower-level game to maximize its profit. From Lemma 3.4.1, CORP can choose an equilibrium by setting the factory capacity allocation for product development, i.e., F_t variable, in each time period $t \in \mathcal{T}$. We use this result to reformulate BPTM-Nash as a bilevel program with independent followers. Consider the following bilevel program:

$$[\text{BPTM}^*] \quad \max \sum_{n \in N} \sum_{t \in \mathcal{T}} \pi_{nt} (D_{nt} + B_{n,t-1} - B_{nt}) \quad (3.4a)$$

$$\text{subject to } Y_{pt} \leq \sum_{\tau \leq t} Z_{p\tau} \quad p \in P, t \in \mathcal{T}, \quad (3.4b)$$

$$Y_{p,t-1} \leq Y_{pt} \quad t \in \mathcal{T} \setminus \{1\}, \quad (3.4c)$$

$$(X, B, I) \in \text{argmin MFG}(F, Y), \quad (3.4d)$$

$$(Z, V) \in \text{argmin ENG}(F), \quad (3.4e)$$

where $\text{ENG}(F)$ is formulated as $\phi_E(F) = \min\{\sum_{p \in P} w_p V_P : \text{ENG-feasibility}(F)\}$. In this model the factory capacity F_t allocated to product development in each period t is

decided in the upper level and passed to the ENG problem in the lower level, decoupling the MFG(F, Y) and ENG(F) subproblems in the lower-level of BPTM*.

Proposition 3.4.1. *BPTM* is equivalent to BPTM-Nash*

Proof. We show that there is a feasible solution to BPTM* corresponding to any feasible solution to BPTM-Nash and vice versa. Consider a bilevel feasible solution $(\hat{Y}, \hat{\mathcal{X}})$ to BPTM* satisfying the upper-level constraints (3.1b)-(3.1c) in BPTM-Nash. By (3.4d), it also satisfies the optimality conditions of the MFG problem. Note that $F_t = 1 - \sum_{n \in N} \hat{X}_{nt} = \hat{F}_t$ for $t \in \mathcal{T}$ in ENG(\hat{X}). Therefore, if $(\hat{Z}, \hat{V}) \in \text{argmin ENG}(\hat{F})$, then $(\hat{Z}, \hat{V}, \hat{F}) \in \text{argmin ENG}(\hat{X})$. As a result, $\hat{\mathcal{X}}$ is an equilibrium of GNEP(\hat{Y}), and $(\hat{Y}, \hat{\mathcal{X}})$ is a bilevel feasible solution to BPTM-Nash.

Now consider a bilevel feasible solution (Y', \mathcal{X}') to BPTM-Nash. This solution satisfies the upper-level constraints (3.4b)-(3.4c) in BPTM*. It also satisfies constraints (3.4d) because MFG's response has to be optimal in an equilibrium of GNEP(Y'). Note that $F'_t = 1 - \sum_{n \in N} X'_{nt}$ for $t \in \mathcal{T}$ in ENG(F'). Therefore, if $(Z', V', F') \in \text{argmin ENG}(X')$, then $(Z', V') \in \text{argmin ENG}(F')$. Thus, (Y', \mathcal{X}') is a bilevel feasible solution to BPTM*.

The fact that BPTM* and BPTM-Nash share the same upper-level objective function completes the proof. \square

The argument in the proof of Proposition 3.4.1 can be extended to any bilevel problem with multiple followers who play a GNEP with exclusively equality coupling constraints. This property was also noted and utilized in Sagratella et al. (2020) for a noncooperative fixed charge transportation problem. They showed that any feasible solution to the coupling equality constraints can be used to find an equilibrium of the GNEP by solving each player's problem independently after fixing the variables in the coupling constraints to their values in that feasible solution.

Although Proposition 3.4.1 separates the lower level problem into two independent follower's problem, solving BPTM* is still difficult because the continuous variables F_t are passed to both lower-level subproblems in each time stage. To alleviate this difficulty, we present a modified version of BPTM* where the factory capacity constraint (3.2b) in

the MFG(F, Y) problem is enforced as an upper level constraint.

$$[\text{BPTM}] \quad \max \sum_{n \in N} \sum_{t \in \mathcal{T}} \pi_{nt} (D_{nt} + B_{n,t-1} - B_{nt}) \quad (3.5a)$$

$$\text{subject to } Y_{pt} \leq \sum_{\tau \leq t} Z_{p\tau} \quad p \in P, t \in \mathcal{T}, \quad (3.5b)$$

$$Y_{p,t-1} \leq Y_{pt} \quad t \in \mathcal{T} \setminus \{1\}, \quad (3.5c)$$

$$F_t + \sum_{n \in N} X_{nt} = 1 \quad t \in \mathcal{T}, \quad (3.5d)$$

$$(X, B, I) \in \text{argmin MFG}(Y), \quad (3.5e)$$

$$(Z, V) \in \text{argmin ENG}(F), \quad (3.5f)$$

where MFG(Y) is formulated as:

$$\min \left\{ \sum_{n \in N} \sum_{t \in \mathcal{T}} h_{nt} I_{nt} + r_{nt} X_{nt} C_t + b_{nt} B_{nt} : \text{MFG-primal-feasibility}(Y) \right\}.$$

Proposition 3.4.2. *If there exists an optimal solution to BPTM, it is also optimal to BPTM*.*

Proof. Consider an optimal solution (Y^*, \mathcal{X}^*) to BPTM. This solution satisfies upper-level constraints (3.4b)-(3.4c) in BPTM*. From (3.5f), it also satisfies the optimality of the ENG(F^*). The feasible region of MFG(Y^*) contains that of MFG(F^*, Y^*). As (X^*, B^*, I^*) is optimal to MFG(Y^*) and satisfies constraint (3.2b), it is also optimal to MFG(F^*, Y^*). The fact that BPTM* and BPTM share the same upper-level objective function completes the proof. \square

In BPTM, factory capacity allocation to ENG, F is a continuous upper-level variable passed to the lower-level ENG problem which is a mixed-integer model. In the bilevel programming literature, researchers have allowed only integer upper-level variables to appear in a mixed-integer lower-level problem in order to guarantee the existence of an optimal solution (Vicente et al. 1996; Köppe et al. 2010). In BPTM, however, the continuous F variables do not appear in the CORP's objective function. They only impact the model through the Z variables of the ENG problem, whose values are constrained to be binary for any value of the F variables. Thus, an optimal solution is always attained when BPTM is feasible.

We propose a solution approach that generates upper and lower bounds in each iteration based on a single-level value function reformulation of BPTM. The main idea of this reformulation is to enforce the optimality of $\text{MFG}(Y)$ and $\text{ENG}(F)$ using constraints in the upper level problem. Let ζ_{nt} , θ_{pt} , and ψ_t be dual variables associated with constraints (3.2c), (3.2d), and (3.2e), respectively. The dual of the $\text{MFG}(Y)$ problem is then given by:

$$\max - \sum_{n \in N} \sum_{t \in \mathcal{T}} \zeta_{nt} D_{nt} + \sum_{p \in P} \sum_{t \in \mathcal{T}} \theta_{pt} Y_{pt} + \sum_{t \in \mathcal{T}} \psi_t \quad (3.6a)$$

$$\text{subject to } \zeta_{nt} - \zeta_{n,t+1} \leq h_{nt} \quad n \in N, t \in \mathcal{T} \quad (3.6b)$$

$$- C_t \zeta_{pt} + \psi_t + \theta_{pt} \leq r_{pt} C_t \quad p \in P, t \in \mathcal{T} \quad (3.6c)$$

$$- C_t \zeta_{nt} + \psi_t \leq r_{nt} C_t \quad n \in N \setminus P, t \in \mathcal{T} \quad (3.6d)$$

$$- \zeta_{nt} + \zeta_{n,t+1} \leq b_{nt} \quad n \in N, t \in \mathcal{T} \quad (3.6e)$$

$$\zeta_{nt} \in \mathbb{R}, \psi_t, \theta_{pt} \leq 0 \quad p \in P, t \in \mathcal{T} \quad (3.6f)$$

We refer to constraints (3.6b)-(3.6f) as **MFG-dual-feasibility**. Since $\text{MFG}(Y)$ is a linear program, we can impose its optimality through the primal feasibility, dual feasibility and strong duality conditions (Labbé et al. 1998; Zeng and An 2014). The strong duality condition requires the equality of primal and dual objectives at optimality, and can be stated as:

$$\sum_{n \in N} \sum_{t \in \mathcal{T}} h_{nt} I_{nt} + r_{nt} X_{nt} C_t + b_{nt} B_{nt} = - \sum_{n \in N} \sum_{t \in \mathcal{T}} \zeta_{nt} D_{nt} + \sum_{p \in P} \sum_{t \in \mathcal{T}} \theta_{pt} Y_{pt} + \sum_{t \in \mathcal{T}} \psi_t. \quad (3.7)$$

We define an auxiliary variable ω_{pt} to reformulate the bilinear term $\theta_{pt} Y_{pt}$ through McCormick (1976) inequalities: $0 \leq \omega_{pt} \leq M Y_{pt}$, $\omega_{pt} \leq \theta_{pt}$, $\omega_{pt} \geq \theta_{pt} - M(1 - Y_{pt})$, where M is a large positive constant. We collectively refer to **MFG-primal-feasibility** (Y), **MFG-dual-feasibility** and the strong duality condition (3.7) as **MFG-optimality** (Y), and to constraints (3.5b) - (3.5d) as **Upper-level-feasibility**. We can now restate

BPTM as:

$$\begin{aligned}
\text{[EPTM]} \quad & \max \sum_{n \in N} \sum_{t \in \mathcal{T}} \pi_{nt} (D_{nt} + B_{n,t-1} - B_{nt}) \\
& \text{subject to } \text{Upper-level-feasibility} \\
& \text{MFG-optimality } (Y) \\
& (Z, V) \in \text{argmin ENG}(F)
\end{aligned}$$

Although **MFG-optimality** (Y) is a set of linear constraints, the Extended Product Transition Model (EPTM) is still a bilevel program due to the constraint $(Z, V) \in \text{argmin ENG}(F)$. We now present two exact single-level reformulations of EPTM that enforce the optimality of ENG with a finite set of constraints and variables.

3.4.1 Single-level Reformulations of the EPTM

Let \mathcal{Z} denote the set of all possible distinct product development schedules of ENG, and J be the index set of \mathcal{Z} . The size of \mathcal{Z} increases exponentially with the number of time periods T and the number of new products $|P|$, but it is finite and bounded above by $2^{T \times |P|}$. Given a product development schedule $j \in J$, we can calculate its total weighted tardiness $\sum_{p \in P} w_p \hat{V}_p^j$, and factory capacity requirement $\hat{H}_t^j = \sum_{p \in P} H_{pt} \hat{Z}_{pt}^j$ in each time period $t \in \mathcal{T}$. We use this fact to enforce the optimality of ENG using a finite set of constraints and variables.

Exact Reformulation 1. Without loss of generality, we assume that the factory capacity requirements for developing new products are positive integers, i.e., $H_{pt} \in \mathbb{Z}_+$. Therefore, the factory capacity requirement \hat{H}_t^j for product development in period t under development plan j is also a positive integer. We define binary variable $\gamma_t^j, j \in J, t \in \mathcal{T}$ which takes a value of 0 if the ENG has sufficient factory capacity for product development plan j in period t , and 1 otherwise. Consider the following single-level reformulation of

the EPTM:

$$\begin{aligned}
[\text{MP}_1(J)] \quad \phi^* &= \max \sum_{n \in N} \sum_{t \in \mathcal{T}} \pi_{nt} (D_{nt} + B_{n,t-1} - B_{nt}) \\
&\text{subject to } \text{Upper-level-feasibility} \\
&\quad \text{MFG-optimality (Y)} \\
&\quad \text{ENG-feasibility (F)} \\
C_t F_t &\geq \hat{H}_t^j - \hat{H}_t^j \gamma_t^j & t \in \mathcal{T}, j \in J & \quad (3.9a) \\
C_t F_t &\leq \hat{H}_t^j - \gamma_t^j + C_t(1 - \gamma_t^j) & t \in \mathcal{T}, j \in J & \quad (3.9b) \\
\sum_{p \in P} w_p V_P &\leq M \sum_{t \in \mathcal{T}} \gamma_t^j + \sum_{p \in P} w_p \hat{V}_P^j & j \in J & \quad (3.9c) \\
\gamma_t^j &\in \{0, 1\} & t \in \mathcal{T}, j \in J & \quad (3.9d)
\end{aligned}$$

The mixed-integer program $\text{MP}_1(J)$ determines the values of the ENG variables Z and V . Their feasibility to the ENG problem is enforced by the **ENG-feasibility (F)** constraints, and their optimality by (3.9a)-(3.9d). Constraints (3.9a) and (3.9b) ensure that sufficient factory capacity is available to ENG to implement product development plan j in period t , i.e., $C_t F_t \geq \hat{H}_t^j$, if $\gamma_t^j = 0$. Otherwise, if $\gamma_t^j = 1$, (3.9a) is not binding, and (3.9b) ensures that the factory capacity allocated for product development in period t is strictly less than \hat{H}_t^j , i.e., $C_t F_t \leq \hat{H}_t^j - 1$. Hence it is not feasible to implement the product development plan j in period t due to insufficient factory capacity. Note that if product development plan j is not feasible, it might still be the case that $C_t F_t \geq \hat{H}_t^j$ for some periods, i.e., constraints (3.9a) are binding with $\gamma_t^j = 0$, while $C_{t'} F_{t'} \leq \hat{H}_{t'}^j - 1$ for some other periods, i.e., constraints (3.9b) are binding with $\gamma_{t'}^j = 1$.

Finally, constraints (3.9c) ensure that if sufficient factory capacity is available in every period for product development plan j , i.e., $\sum_{t \in \mathcal{T}} \gamma_t^j = 0$, then the weighted tardiness of the new products, i.e., the ENG objective function, should not exceed that under product development plan j , i.e., $\sum_{p \in P} w_p V_P \leq \sum_{p \in P} w_p \hat{V}_P^j$. Note that if $\sum_{t \in \mathcal{T}} \gamma_t^j = 0$, product development plan j is feasible, however it may not be selected by ENG for implementation if there are other feasible product development plans that are more favorable for ENG. The largest possible value for the left-hand side of constraints (3.9c) is attained when no new products are developed until the last time period, so we can set $M = \sum_{p \in P} (T - \theta_p) w_p$.

Proposition 3.4.3. *$\text{MP}_1(J)$ is equivalent to the EPTM.*

Proof. We show that there is a feasible solution to $\text{MP}_1(J)$ corresponding to any feasible solution to EPTM, and vice versa. Consider a bilevel feasible solution $(\bar{Y}, \bar{\mathcal{X}})$ to EPTM, and for every $j \in J$ and $t \in \mathcal{T}$ define

$$\bar{\gamma}_t^j = \begin{cases} 1 & \text{if } C_t \bar{F}_t < \hat{H}_t^j \\ 0 & \text{if } C_t \bar{F}_t \geq \hat{H}_t^j. \end{cases}$$

$(\bar{\mathcal{X}}, \bar{\gamma})$ is feasible to constraints (3.9a) and (3.9b) by construction. It is also feasible to (3.9c) because $(\bar{Z}, \bar{V}) \in \text{argmin ENG}(\bar{F})$. Thus, $(\bar{Y}, \bar{\mathcal{X}}, \bar{\gamma})$ is a feasible solution to $\text{MP}_1(J)$. We now consider a feasible solution $(Y^*, \mathcal{X}^*, \gamma^*)$ to $\text{MP}_1(J)$. This solution satisfies **Upper-level-feasibility**, **MFG-optimality**(Y^*), and **ENG-feasibility**(F^*). The index set of all feasible $\text{ENG}(F^*)$ solutions is given by $J^* = \{j \in J \mid \hat{H}_t^j \leq C_t F_t^*, t \in \mathcal{T}\}$. Note that $\gamma_t^{*j} = 0$ for every $j \in J^*$ and $t \in \mathcal{T}$ due to constraints (3.9a) and (3.9b). Thus, $\sum_{t \in \mathcal{T}} \gamma_t^{*j} = 0$ and constraints (3.9c) ensure that

$$\sum_{p \in P} w_p V_P^* \leq \sum_{p \in P} w_p \hat{V}_P^j, \quad j \in J^*$$

As a result, (Z^*, V^*) is optimal to the $\text{ENG}(F^*)$ problem, and (Y^*, \mathcal{X}^*) is feasible to EPTM. The fact that $\text{MP}_1(J)$ and EPTM have the same objective completes the proof.

□

Exact Reformulation 2. To obtain an alternative single-level reformulation of EPTM, we define $\mathcal{B}_t(j) = \{k \in J \mid \hat{H}_t^k \leq \hat{H}_t^j\}$ as the set of all product development plans k whose factory capacity requirement does not exceed that of development plan j at time t . Thus, if the factory capacity allocated to product development in period t is not sufficient to execute plan k , i.e., strictly less than \hat{H}_t^k , it will not be sufficient for plan j either. We define auxiliary binary variable $v_t^j \in \{0, 1\}$. If $v_t^j = 1$, the factory capacity allocation for product development in period t , i.e., $C_t F_t$ is enforced to be less than \hat{H}_t^j . If $v_t^j = 0$, on the other hand, no requirement is imposed on $C_t F_t$. The main difference between the v_t^j and γ_t^j variables is that if $\gamma_t^j = 0$ sufficient factory capacity is allocated to execute plan j in period t , whereas if $v_t^j = 0$, the product development plan j is feasible in period t if $\sum_{k \in \mathcal{B}_t(j)} v_t^k = 0$, and infeasible if $\sum_{k \in \mathcal{B}_t(j)} v_t^k = 1$. We can now reformulate EPTM as

follows:

$$\begin{aligned}
[\text{MP}_2(J)] \quad \phi^* &= \max \sum_{n \in N} \sum_{t \in \mathcal{T}} \pi_{nt} (D_{nt} + B_{n,t-1} - B_{nt}) \\
&\text{subject to Upper-level-feasibility} \\
&\text{MFG-optimality (Y)} \\
&\text{ENG-feasibility (F)} \\
C_t F_t &\leq C_t - \sum_{j \in J} (C_t + 1 - \hat{H}_t^j) v_t^j & t \in \mathcal{T} \quad (3.10a) \\
\sum_{p \in P} w_p V_P &\leq M \sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{B}_t(j)} v_t^k + \sum_{p \in P} w_p \hat{V}_P^j & j \in J \quad (3.10b) \\
v_t^j &\in \{0, 1\} & t \in \mathcal{T}, j \in J \quad (3.10c)
\end{aligned}$$

Constraints (3.10a)-(3.10c) enforce the optimality of the ENG(F) problem. In particular, constraints (3.10a) state that v_t^j can be equal to 1 for at most one $j \in J$ such that $C_t F_t \leq \hat{H}_t^j - 1$ in each time period t . Constraints (3.10b) ensure that if there is no infeasible development plan in $\mathcal{B}_t(j)$ in any time period, the weighted tardiness of the ENG problem cannot exceed that of product development plan j . Lozano and Smith (2017) presented a similar single-level reformulation for general bilevel mixed-integer programs. As in the first reformulation, we set $M = \sum_{p \in P} (T - \theta_p) w_p$ in constraints (3.10b).

Proposition 3.4.4. *If $\text{MP}_2(J)$ is feasible, there exists an optimal solution $(Y^*, \mathcal{X}^*, v^*)$ to $\text{MP}_2(J)$ such that in each period $t \in \mathcal{T}$, $v_t^{*j_t} = 1$ for $j_t \in \text{argmin}\{\hat{H}_t^k \mid C_t F_t^* \leq \hat{H}_t^k - 1, k \in J\}$ and $v_t^{*j} = 0$ for all $j \in J \setminus \{j_t\}$.*

Proof. Let $(Y^*, \mathcal{X}^*, v^*)$ be an optimal solution to $\text{MP}_2(J)$. Constraints (3.10a) ensure that $v_t^{*j} = 1$ for at most one $j \in \hat{J}_t = \{k \mid C_t F_t^* \leq \hat{H}_t^k - 1, k \in J\}$ and $v_t^{*j} = 0$ for $j \in J \setminus \hat{J}_t$ in each $t \in \mathcal{T}$.

Case 1. There exists a $t \in \mathcal{T}$ such that $v_t^{*j} = 0$ for all $j \in \hat{J}_t$. In this case, we can construct $(Y^*, \mathcal{X}^*, \bar{v})$ such that $\bar{v} = v^*$ except $\bar{v}_t^{j_t} = 1$. Then, \bar{v}_t satisfies (3.10a) because $C_t F_t^* \leq \hat{H}_t^{j_t} - 1$. This solution also satisfies constraints (3.10b) because $\bar{v}_t^j \geq v_t^{*j}$ for all $j \in J$ and $t \in \mathcal{T}$. As a result, $(Y^*, \mathcal{X}^*, \bar{v})$ is a feasible solution.

Case 2. There exists a $t \in \mathcal{T}$ such that $v_t^{*\ell} = 1$ for $\ell \in \hat{J}_t$, but $\ell \neq j_t$. In this case, we can construct $(Y^*, \mathcal{X}^*, \bar{v})$ such that $\bar{v} = v^*$ except $\bar{v}_t^{j_t} = 1$ and $\bar{v}_t^\ell = 0$. Then,

\bar{v}_t satisfies (3.10a) because $C_t F_t^* \leq \hat{H}^{j_t} - 1$. Note that $\hat{H}_t^{j_t} \leq \hat{H}_t^\ell$ by definition of j_t . This solution also satisfies constraints (3.10b) because for any $j \in J$ such that $\hat{H}_t^{j_t} \leq \hat{H}_t^j < \hat{H}_t^\ell$, we have that $\sum_{k \in \mathcal{B}_t(j)} \bar{v}_t^k = 1 \geq \sum_{k \in \mathcal{B}_t(j)} v_t^{*k} = 0$. As a result, $(Y^*, \mathcal{X}^*, \bar{v})$ is a feasible solution.

In either case, the constructed feasible solution has the same objective function, and thus is also optimal. \square

Proposition 3.4.5. *MP₂(J) is equivalent to EPTM.*

Proof. We show that there is a feasible solution to MP₂(J) corresponding to any feasible solution to EPTM, and vice versa. Consider a bilevel feasible solution $(\bar{Y}, \bar{\mathcal{X}})$ to EPTM. Based on Proposition 3.4.4, identify a solution $j_t \in J$ such that $j_t \in \operatorname{argmin}\{\hat{H}_t^k \mid C_t \bar{F}_t \leq \hat{H}_t^k - 1, k \in J\}$ for every $t \in \mathcal{T}$. Now set $\bar{v}_t^{j_t} = 1$ and $\bar{v}_t^j = 0$ for all $j \in J \setminus \{j_t\}$. $(\bar{Y}, \bar{\mathcal{X}}, \bar{v})$ is feasible to constraints (3.10a) by construction. Suppose by contradiction that $(\bar{Y}, \bar{\mathcal{X}}, \bar{v})$ violates at least one constraint (3.10b). Then there is a solution $j' \in J$ such that $\sum_{p \in P} w_p \bar{V}_P > \sum_{p \in P} w_p \hat{V}_P^{j'}$ and $\bar{v}_t^k = 0$ for every $t \in \mathcal{T}$ and $k \in \mathcal{B}_t(j')$. This means that $(\hat{Z}^{j'}, \hat{V}^{j'})$ is a feasible solution to ENG(\bar{F}) and its corresponding objective value is less than $\sum_{p \in P} w_p \bar{V}_P$. This contradicts the fact that $(\bar{Z}, \bar{V}) \in \operatorname{argmin} \operatorname{ENG}(\bar{F})$. Thus, $(\bar{Y}, \bar{\mathcal{X}}, \bar{v})$ is a feasible solution to MP₂(J).

Now consider a feasible solution $(Y^*, \mathcal{X}^*, v^*)$ to MP₂(J). This solution satisfies **Upper-level-feasibility**, **MFG-optimality**(Y^*), and **ENG-feasibility**(F^*). The index set of all feasible ENG(F^*) solutions is given by $J^* = \{j \in J \mid C_t F_t^* \geq \hat{H}_t^j, t \in \mathcal{T}\}$. Note that $v_t^{*k} = 0$ for every $k \in \mathcal{B}_t(j)$, $j \in J^*$ and $t \in \mathcal{T}$ because otherwise $C_t F_t^* \leq \hat{H}_t^k - 1$ due to constraints (3.10a), contradicting $C_t F_t^* \geq \hat{H}_t^j$ as $\hat{H}_t^j \geq \hat{H}_t^k$ by definition of $\mathcal{B}_t(j)$. Thus, $\sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{B}_t(j)} v_t^{*k} = 0$ for $j \in J^*$, and constraints (3.10b) ensure that

$$\sum_{p \in P} w_p V_P^* \leq \sum_{p \in P} w_p \hat{V}_P^j, \quad j \in J^*$$

As a result, (Z^*, V^*) is optimal to the ENG(F^*) problem, and (Y^*, \mathcal{X}^*) is feasible to EPTM. The fact that MP₂(J) and EPTM have the same objective completes the proof.

\square

3.4.2 Constraint and Column Generation Algorithm

The number of product development schedules in set J grows exponentially with the number of new products $|P|$ and the number of time periods T , rendering the solution time of $MP_1(J)$ or $MP_2(J)$ using off-the-shelf algorithms prohibitively time consuming even for moderate size instances. We thus propose a constraint and column generation (CCG) method where constraints (3.9a)-(3.9d) (in the first reformulation $MP_1(J)$) or constraints (3.10a)-(3.10c) (in the second reformulation $MP_2(J)$) are relaxed and then enforced through an iterative process. Since the main steps of the CCG method are similar for $MP_1(J)$ and $MP_2(J)$, we refer to both reformulations simply as $MP(J)$ in the rest of this section. A restricted master problem $MP(J^k)$ is solved in each iteration k where $J^k \subseteq J$ and $J^0 = \emptyset$. The optimal objective function value ϕ^k of $MP(J^k)$ is an upper bound on the optimal objective value ϕ^* of the EPTM, i.e., $\phi^k \geq \phi^*$, because the feasible region of $MP(J^k)$ contains that of $MP(J)$.

Let (Y^k, \mathcal{X}^k) be an optimal solution of $MP(J^k)$ at iteration k . We first solve $ENG(F^k)$ to determine $\phi_E(F^k)$, the optimal objective value of ENG under the factory capacity allocation F^k . We then solve the following subproblem to check the feasibility of ENG 's product development schedule for the product release schedule Y^k :

$$\begin{aligned}
 [ENG'(F^k, Y^k)] \quad \phi'_E(F^k, Y^k) = \min \quad & \sum_{p \in P} w_p V_p \\
 \text{subject to} \quad & \text{ENG-feasibility}(F^k) \\
 & \sum_{\tau \leq t} Z_{p\tau} \geq Y_{pt}^k \quad p \in P, t \in T \quad (3.11)
 \end{aligned}$$

The only difference between subproblems $ENG(F^k)$ and $ENG'(F^k, Y^k)$ is constraint (3.11), which forces ENG to meet the product release schedule Y^k . Let (Z', V') be the optimal solution to $ENG'(F^k, Y^k)$ and $\phi'_E(F^k, Y^k)$ the optimal objective value. If $\phi'_E(F^k, Y^k) = \phi_E(F^k)$, then Z' is an optimal product development schedule for ENG in response to the factory capacity allocation F^k , and it is feasible to the product release schedule Y^k . Thus, $(Y^k, \mathcal{X}') = (Y^k, F^k, X^k, B^k, I^k, Z', V')$ is bilevel feasible. Note that (Y^k, \mathcal{X}') is also bilevel optimal because the feasible region of the restricted master problem $MP(J^k)$ contains the feasible region of $MP(J)$. If, on the other hand, $\phi'_E(F^k, Y^k) > \phi_E(F^k)$, we generate new columns and constraints to add to $MP(J^k)$. Algorithm 1 presents the CCG algorithm.

Proposition 3.4.6. *Algorithm 1 terminates in finite number of iterations.*

Algorithm 1: Constraint and column generation (CCG) algorithm

```
1 Initialization: set  $k = 0$ ,  $J^k = \emptyset$ 
2 while  $J^k \subsetneq J$  do
3   Solve  $\text{MP}(J^k)$ .
4   if  $\text{MP}(J^k)$  is infeasible or  $J^k = J$  then
5     |  $\text{MP}(J)$  is infeasible, STOP.
6   end
7   Let  $(Y^k, \mathcal{X}^k) = (Y^k, F^k, X^k, B^k, I^k, Z^k, V^k)$  be the optimal solution to
    $\text{MP}(J^k)$ .
8   Solve  $\text{ENG}(F^k)$  to get the optimal solution  $Z^*$  and the objective value
    $\phi_E(F^k)$ .
9   if  $\sum_{p \in P} w_p V_P^k \leq \phi_E(F^k)$  then
10    | Return the optimal bilevel solution  $(Y^k, \mathcal{X}^k)$ .
11  end
12  Solve  $\text{ENG}'(F^k, Y^k)$  to get the optimal solution  $(Z', V')$  and the objective
   value  $\phi'_E(F^k, Y^k)$ .
13  if  $\phi_E(F^k) = \phi'_E(F^k, Y^k)$  then
14    | Return the bilevel optimal solution,
    $(Y^k, \mathcal{X}') = (Y^k, F^k, X^k, B^k, I^k, Z', V')$ .
15  end
16  Add constraints (3.9a)-(3.9d) to  $\text{MP}_1(J^k)$  (or constraints (3.10a)-(3.10c) to
    $\text{MP}_2(J^k)$ ).
17  Set  $J^k = J^k \cup \{Z^*\}$ ,  $k = k + 1$ .
18 end
```

Proof. A solution to ENG is generated in line 8. If the same solution has been generated in previous iterations, the stopping condition in line 9 will be satisfied. Thus, in each iteration, either a new product development schedule is added to J^k , or the algorithm terminates. The result follows since the set of distinct feasible product development schedules J is finite. \square

Proposition 3.4.7. *Algorithm 1 either returns the optimal solution to $MP(J)$ or identifies infeasibility.*

Proof. In any iteration k , the feasible region of $MP(J^k)$ contains the feasible region of $MP(J)$. Therefore, if $MP(J^k)$ is infeasible, $MP(J)$ is also infeasible, and the algorithm will terminate in line 5. If the condition in line 9 is satisfied, the current $MP(J^k)$ solution satisfies the optimality conditions for ENG, and is thus bilevel optimal. If the condition in line 13 is satisfied, (Z', V') is optimal to $ENG(F^k)$ and feasible to the product release schedule Y^k . Thus, $(Y^k, \mathcal{X}') = (Y^k, F^k, X^k, B^k, I^k, Z', V')$ is a bilevel feasible solution. It is also bilevel optimal because $MP(J^k)$ is a relaxation of $MP(J)$ whose objective does not include variables of the ENG. If none of the three stopping conditions within the while loop are satisfied, the algorithm will stop when $J^k = J$, that is when all ENG solutions have been generated. At that point, it can be concluded that there is no bilevel feasible solution to $MP(J)$. \square

The CCG algorithm solves the restricted master problem $MP(J^k)$ in each iteration. Our computational experiments showed that $MP(J^k)$ is a difficult mixed-integer program, and a high quality initial feasible solution can speed up its solution considerably. We develop a greedy heuristic to generate a feasible solution to $MP(J^k)$ in Algorithm 2.

\mathcal{P} denotes the set of products that have not been developed, and \mathcal{U}_t is the set of products developed in period t . This greedy heuristic iterates over time periods until all new products are developed or infeasibility is detected. In each iteration, `nSol` feasible subsets of the new products that have not been developed are randomly chosen to be developed, and the time period is incremented by one. We initialize $\mathcal{P} \leftarrow P$, $\mathcal{U}_t \leftarrow \emptyset$ for all $t \in \mathcal{T}$, $LB = -\infty$, and $\tau = T_0$, where T_0 is the earliest time period a new product can be developed. After running Greedy-MP, we solve $MP(J^k)$ with the fixed product development schedule Z_{greedy} to get a feasible solution. This feasible solution is then fed into the solver when solving the $MP(J^k)$.

Algorithm 2: Greedy heuristic for $\text{MP}(J^k)$

```
1 Greedy-MP( $\mathcal{P}, \mathcal{U}, \tau, LB, Z_{\text{greedy}}$ )
2 if  $\mathcal{P} = \emptyset$  ;                               /* all new products are developed */
3 then
4   Let  $\hat{Z}_{pt} = 1$  for  $p \in \mathcal{U}_t$ ,  $t \in \mathcal{T}$ , and  $\hat{Z}_{pt} = 0$  otherwise
5   Solve  $\text{MP}(J^k)$  with  $Z = \hat{Z}$ , and let  $\phi^*(\hat{Z})$  be the optimal objective value
6   if  $\phi^*(\hat{Z}) > LB$  then
7     |  $LB = \phi^*(\hat{Z})$  and  $Z_{\text{greedy}} = \hat{Z}$ 
8   end
9 else
10  if  $\tau = T$  then
11    | if  $\sum_{p \in \mathcal{P}} H_{p\tau} \leq C_T$  then
12      | Set  $\mathcal{U}_T \leftarrow \mathcal{P}$ ,  $\mathcal{P} \leftarrow \emptyset$ , and call Greedy-MP( $\mathcal{P}, \mathcal{U}, \tau, LB, Z_{\text{greedy}}$ )
13    | end
14  else
15    | for  $\ell = 1, \dots, nSol$  do
16      | Select new product subsets  $\mathcal{U}_\tau^\ell \subseteq \mathcal{P}$  such that  $\sum_{p \in \mathcal{U}_\tau^\ell} H_{p\tau} \leq C_\tau$ 
17      | Set  $\mathcal{U}_\tau \leftarrow \mathcal{U}_\tau^\ell$ , and call Greedy-MP( $\mathcal{P} \setminus \mathcal{U}_t, \mathcal{U}, \tau + 1, LB, Z_{\text{greedy}}$ )
18    | end
19  end
20 end
```

3.4.3 Lower Bound for BPTM

We also propose a heuristic to generate a bilevel feasible solution to BPTM that will provide a lower bound. Consider the following mixed-integer program which we refer to as the Extended ENG (EENG) problem:

$$\begin{aligned}
 \text{[EENG]} \quad \eta^* &= \min \sum_{p \in P} w_p V_p \\
 &\text{subject to Upper-level-feasibility} \\
 &\quad \text{MFG-optimality } (Y) \\
 &\quad \text{ENG-feasibility } (F)
 \end{aligned}$$

Let η^* denote the optimal objective value of the EENG, and F^* the factory capacity allocation to ENG in the corresponding optimal solution. Note that $\eta^* \geq \phi_E(F^*)$ because each (Z^*, V^*) solution with $F = F^*$ in the feasible solution set of EENG is feasible to $\text{ENG}(F^*)$, and these two problems have the same objective function. Therefore, if $\eta^* = \phi_E(F^*)$, an optimal solution to EENG is a bilevel feasible solution to BPTM because it simultaneously satisfies the optimality conditions for both MFG and ENG.

Our preliminary computational experiments, however, showed that off-the-shelf solvers struggle to solve even moderate-size instances of EENG in a reasonable time. Therefore, we have developed a warm-start method to generate an initial feasible solution to EENG in which we solve a multiple knapsack problem (MKP) to generate product development schedules yielding solutions for the Z variables. Each time period is treated as a knapsack with a certain amount of factory capacity allocated to product development. New products under development represent items to be packed into these knapsacks.

The “weight” of each new product (item) $p \in P$ is set to the factory capacity H_{pt} required for its development. Recall that δ_p denotes the due date for developing product $p \in P$. In our MKP we define the “value” of developing product p in time period $t \in \mathcal{T}$ as:

$$S_{pt} = \begin{cases} -\infty & \text{if } t < T^0, \\ T - (\delta_p - t) & \text{if } t \leq \delta_p, \\ (\delta_p - t)w_p & \text{if } t > \delta_p, \end{cases}$$

where T^0 is the earliest time period a new product can be developed. The definition of S_{pt} favors developing new products close to their due dates. This property allows the warm-start method to generate product development schedules that are not too focused on minimizing the weighted tardiness objective of the ENG.

We define a coefficient f to control the factory capacity allocated to product development in each time period $t \in \mathcal{T}$ as a fraction of the total demand for the current products, i.e., $\sum_{n \in N \setminus P} D_{nt}$. This is equivalent to specifying a minimum fill rate for current demand which MFG must meet. The formulation of the MKP for a given f is as follows:

$$\text{MKP}(f) \quad \max \sum_{p \in P} \sum_{t \in \mathcal{T}} S_{pt} Z_{pt} \quad (3.13a)$$

$$\sum_{p \in P} H_{pt} Z_{pt} \leq C_t - f \sum_{n \in N \setminus P} D_{nt} \quad t \in \mathcal{T} \quad (3.13b)$$

$$Z_{pt} \in \{0, 1\} \quad p \in P, t \in \mathcal{T} \quad (3.13c)$$

We solve $\text{MKP}(f)$ with $f \in \{1.0, 1.2, 1.4\}$ in the numerical experiments. The resulting product development schedules, i.e., Z variables, are fixed in EENG to generate feasible solutions. These solutions are then provided to solver as initial feasible solutions when optimizing the EENG. We henceforth refer to this algorithm as the Lower Bounding Algorithm (LBA). Note that if EENG is infeasible, then BPTM is also infeasible because the feasible region of BPTM is a subset of the feasible region of EENG. Thus the LBA can identify the infeasibility of a BPTM instance.

3.5 Numerical Experiments

We run computational experiments to evaluate the performance of the proposed solution algorithm and model formulations. We also analyze the impact of product mix on the optimal objective values of the CORP, MFG and ENG problems. Finally, we provide managerial insights about the value of CORP's leadership.

Table 3.3: Parameter values

Demand in Market 1	U[600, 1000]	h (holding cost)	5
Demand in Market 2	U[800, 1200]	r (production cost)	$10h$
Demand in Market 3	U[1200, 1600]	b (backorder cost)	$20h$
Demand in Market 4	U[1400, 1800]	$w \in \mathbb{Z}^P$ (tardiness cost)	U[5, 200]
T^0	$0.4T$	$\pi \in \mathbb{Z}^N$ (revenue)	U[25, 30]

3.5.1 Instance Generation

Our test instances represent a realistic demand pattern observed during product transition periods in the semiconductor industry: demand for current products diminishes to zero over time while the demand for new products rises over time and then stabilizes. Specifically, in our test instances the current products have positive demand in the first 60% of the planning horizon, and new products have positive demand in the last 60%. Thus there is a time interval with positive demand for both current and new products. The demand for current products diminishes in the last quarter of their life time, while demand for new products increases in the first quarter of their life time. Since semiconductor companies operate in different markets such as mobile devices, memory and servers (Rash and Kempf 2012), we consider four discrete uniform distributions to model demand. Each product is randomly assigned to a market.

The due date for the development of each new product is selected randomly with equal probability from the periods with positive demand for that product. The factory capacity C_t in period t is randomly generated from a discrete uniform distribution between $\lceil 0.7 \sum_{n \in N} D_{nt} \rceil$ and $\lceil 1.2 \sum_{n \in N} D_{nt} \rceil$. Once the factory capacity is determined, the factory capacity H_{pt} required for the development of new product p in period t is generated randomly from a discrete uniform distribution between $\lceil 0.2C_t \rceil$ and $\lceil 0.6C_t \rceil$. The values of other model parameters are given in Table 3.3.

3.5.2 Computational Performance

We analyze the performance of our algorithm using 32 problem classes with number of periods $T \in \{12, 18, 24, 30\}$, number of products $|N| \in \{12, 14\}$, and number of new products $|P| \in \{4, 6, 8, 10\}$. We report the size of each problem class in Table 3.4. We

generate and solve six instances per problem class for a total of $32 \times 6 = 192$ instances. The LBA identified three infeasible instances which are removed from the analysis. We set the time limit to 7,200 seconds for the overall solution algorithm, 3,600 seconds for solving the master problem in each iteration of the CCG algorithm, and 300 seconds for the LBA in the first iteration. We do not interrupt an iteration if it is in progress when the time limit is exceeded. We carry out computations on a computer with an Intel Xeon 2.99 GHz with 2 processors and 32 GB RAM. We code our algorithms in C++, and solve the mixed-integer programs using CPLEX 12.9.

Table 3.5 reports the performance of the solution method for both exact reformulations. The **t-max(min)** column reports the maximum (minimum) solution time in seconds, and **t-avg** column is the average solution time for optimally solved instances. The **iter** column reports the maximum (minimum) number of iterations for optimally solved instances, and the **us** column reports the number of instances (out of 6 instances in each problem class) which can not be solved optimally within the time limit. Neither of the exact reformulations has consistently better computational performance than the other. The **f-FLBA** (failed LBA) column gives the number of instances for which the LBA failed to generate a lower bound within the time limit. The solution times of some instances are significantly shorter with Exact Reformulation 1, e.g., the C22 problem class, while the solution times of some other problem instances are significantly shorter with Exact Reformulation 2, e.g., the C14 problem class. It is interesting that some instances are solved optimally even when the LBA fails to generate an initial feasible solution, e.g., all six instances in C9 are solved optimally although the LBA failed to generate an initial feasible solution for three of them.

The number of new products has the greatest impact on the solution time. Solving the bilevel model becomes more difficult as the number of new products increases because MFG and ENG have to be coordinated on the development of more products. The number of periods has mixed effect on the solution time. The problem size increases with T , but a longer T provides scheduling flexibility for product development. In Table 3.5, some of the longest solution times occur in problem classes with $T = 12$ periods, e.g., C7 for Exact Reformulation 1, and C8 for Exact Reformulation 2.

Table 3.4: Number of constraints and variables in each instance. The number of binary variables is given in parenthesis. CORP is the leader in BPTM. MFG and ENG are followers.

				Number of Constraints			Number of Variables		
Class	T	N	P	CORP	MFG	ENG	CORP	MFG	ENG
C1	12	12	4	71	720	64	60 (48)	432	52 (48)
C2	12	12	6	95	1,008	90	84 (72)	432	78 (72)
C3	12	12	8	119	1,296	116	108 (96)	432	104 (96)
C4	12	12	10	143	1,584	142	132 (120)	432	130 (120)
C5	12	14	4	71	744	64	60 (48)	504	52 (48)
C6	12	14	6	95	1,032	90	84 (72)	504	78 (72)
C7	12	14	8	119	1,320	116	108 (96)	504	104 (96)
C8	12	14	10	143	1,608	142	132 (120)	504	130 (120)
C9	18	12	4	107	1,512	94	90 (72)	648	76 (72)
C10	18	12	6	143	2,160	132	126 (108)	648	114 (108)
C11	18	12	8	179	2,808	170	162 (144)	648	152 (144)
C12	18	12	10	215	3,456	208	198 (180)	648	190 (180)
C13	18	14	4	107	1,548	94	90 (72)	756	76 (72)
C14	18	14	6	143	2,196	132	126 (108)	756	114 (108)
C15	18	14	8	179	2,844	170	162 (144)	756	152 (144)
C16	18	14	10	215	3,492	208	198 (180)	756	190 (180)
C17	24	12	4	143	2,592	124	120 (96)	864	100 (96)
C18	24	12	6	191	3,744	174	168 (144)	864	150 (144)
C19	24	12	8	239	4,896	224	216 (192)	864	200 (192)
C20	24	12	10	287	6,048	274	264 (240)	864	250 (240)
C21	24	14	4	143	2,640	124	120 (96)	1,008	100 (96)
C22	24	14	6	191	3,792	174	168 (144)	1,008	150 (144)
C23	24	14	8	239	4,944	224	216 (192)	1,008	200 (192)
C24	24	14	10	287	6,096	274	264 (240)	1,008	250 (240)
C25	30	12	4	179	3,960	154	150 (120)	1,080	124 (120)
C26	30	12	6	239	5,760	216	210 (180)	1,080	186 (180)
C27	30	12	8	299	7,560	278	270 (240)	1,080	248 (240)
C28	30	12	10	359	9,360	340	330 (300)	1,080	310 (300)
C29	30	14	4	179	4,020	154	150 (120)	1,260	124 (120)
C30	30	14	6	239	5,820	216	210 (180)	1,260	186 (180)
C31	30	14	8	299	7,620	278	270 (240)	1,260	248 (240)
C32	30	14	10	359	9,420	340	330 (300)	1,260	310 (300)

Table 3.5: The performance of the CCG algorithm with Exact Reformulations 1 and 2.

	Exact Reformulation 1				Exact Reformulation 2				f-LBA
	t-max(min)	t-avg	iter	us	t-max(min)	t-avg	iter	us	
C1	7 (1)	3	4 (1)	-	6 (1)	3	4 (1)	-	-
C2	30 (1)	10	4 (1)	-	23 (1)	9	4 (1)	-	-
C3	4,576 (1)	1,711	16 (1)	-	4,365 (2)	1,642	17 (1)	-	-
C4	2 (1)	2	1 (1)	4	3 (2)	3	1 (1)	4	1
C5	6 (1)	3	2 (1)	-	5 (1)	3	4 (1)	-	2
C6	209 (4)	44	8 (1)	-	144 (3)	34	8 (1)	-	-
C7	7,045 (4)	2,101	20 (1)	1	7,291 (4)	2,417	24 (1)	-	-
C8	300 (4)	152	1 (1)	4	8,553 (3)	3,838	9 (1)	2	1
C9	16 (2)	6	3 (1)	-	8 (2)	4	3 (1)	-	3
C10	2,064 (303)	663	11 (1)	-	1,944 (304)	839	11 (1)	-	-
C11	53 (3)	28	1 (1)	4	148 (3)	76	1 (1)	4	5
C12	752 (2)	200	1 (1)	2	75 (2)	29	1 (1)	2	3
C13	33 (3)	12	3 (1)	-	24 (1)	9	3 (1)	-	-
C14	3,859 (2)	774	9 (1)	-	1,939 (2)	404	8 (1)	-	2
C15	305 (2)	203	1 (1)	3	304 (2)	202	1 (1)	3	4
C16	302 (2)	152	1 (1)	2	301 (2)	152	1 (1)	2	2
C17	19 (2)	7	2 (1)	-	10 (2)	5	2 (1)	-	-
C18	7,518 (2)	1,255	4 (1)	-	6,992 (2)	1,452	10 (1)	-	1
C19	451 (313)	382	1 (1)	4	451 (310)	381	1 (1)	4	4
C20	306 (302)	304	1 (1)	4	302 (302)	302	1 (1)	4	2
C21	94 (15)	42	7 (1)	-	57 (6)	21	5 (1)	-	-
C22	1,233 (5)	494	4 (1)	-	9,841 (3)	1,905	4 (1)	-	2
C23	1,655 (10)	567	1 (1)	2	1,657 (7)	567	1 (1)	2	4
C24	605 (605)	605	1 (1)	5	605 (605)	605	1 (1)	5	4
C25	303 (3)	98	5 (1)	-	367 (4)	155	5 (1)	-	-
C26	4,346 (3)	1,035	8 (1)	1	5,019 (3)	1,180	9 (1)	1	2
C27	657 (1)	344	1 (1)	2	1,653 (6)	593	1 (1)	1	4
C28	459 (5)	162	1 (1)	1	455 (4)	154	1 (1)	1	4
C29	301 (4)	150	3 (1)	-	302 (4)	152	3 (1)	-	1
C30	2,405 (5)	806	6 (1)	2	2,690 (4)	985	5 (1)	1	4
C31	600 (5)	274	1 (1)	1	603 (5)	274	1 (1)	1	4
C32	1,205 (6)	331	1 (1)	-	1,204 (4)	329	1 (1)	-	3

Table 3.6: Average change in the optimal objective values of CORP, MFG, and ENG as the number of new products $|P|$ increases.

	$ P = 4$ to 6	$ P = 6$ to 8	$ P = 8$ to 10
CORP (max profit)	85.5%	19.8%	55.3%
MFG (min cost)	21.2%	6.3%	19.8%
ENG (min tardiness)	-37.1%	63.8%	-80.0%

3.5.3 The Impact of Product Mix

In this section we explore the impact of product mix, i.e., the number of current products and new products in set N , on the optimal objective values of the CORP, MFG, and ENG problems. We consider instances with $T = 18$ periods and $|N| = 14$ products. We analyze the change in the optimal objective value of each decision entity by increasing the number of new products $|P|$ from 4 to 6, 6 to 8, and 8 to 10 while keeping the total number of products $|N| = 14$ in all instances. For example, consider an instance with $|P| = 4, |N| = 14, T = 18$ and another with $|P| = 6, |N| = 14, T = 18$. The number of current products is $(14 - 4) = 10$ in the first instance, and $(14 - 6) = 8$ in the second. We ensure that the data for the first 4 new products and 8 current products stays the same as we increase the number of new products. We generate 6 instances with $|P| = 4$, 6 with $|P| = 6$, and 6 with $|P| = 10$.

Table 3.6 shows that CORP's profit and MFG's cost increase with the number of new products, while the tardiness of ENG shows no consistent pattern. Before producing a new product, MFG has to wait for its development by the ENG, a constraint that does not apply to current products. Therefore, increasing the number of new products forces MFG to use less factory capacity for the current products in order to meet the demand for new products. As a result, the MFG objective deteriorates when there are more new products in the product mix. We also analyze the amount of change in each component of the MFG objective in Table 3.7. Consistent with the discussion above, Table 3.7 shows that backorder cost increases significantly with the number of new products.

Having more new products in the product mix gives CORP more control over MFG through the product release schedule, i.e., Y variables. Furthermore, CORP's objective does not directly incorporate the backorder cost. Thus, the effect of this increased control

Table 3.7: Change in each component of the MFG objective with number of new products.

Cost component	$ P = 4$ to 6	$ P = 6$ to 8	$ P = 8$ to 10
Inventory	-21,766	111,372	264,023
Backorder	424,091	144,796	263,492
Production	-26,146	-21,396	-30,900

power is higher profit for CORP. As for the ENG objective, tardiness may increase with the number of new products simply because more products must be developed within a fixed planning horizon of T periods. On the other hand, MFG has more incentive to provide factory capacity to help ENG to develop products in a timely manner to meet future demand. This explains the mixed effect of increasing the number of new products on the ENG objective in Table 3.6.

3.5.4 The Value of CORP's Leadership

The aim of this section is to assess the value of CORP's leadership. To this end, we consider an alternative bilevel model in which MFG manages the factory capacity allocation acting as the leader. ENG in the lower-level is the only follower. The formulation of this model, referred to as MFG-Leader, is given by:

$$\text{MFG-Leader} \quad \min \sum_{n \in N} \sum_{t \in T} h_{nt} I_{nt} + r_{nt} X_{nt} C_t + b_{nt} B_{nt} \quad (3.14a)$$

$$\text{subject to } Y_{pt} \leq \sum_{\tau \leq t} Z_{p\tau} \quad p \in P, t \in \mathcal{T}, \quad (3.14b)$$

$$F_t + \sum_{n \in N} X_{nt} = 1 \quad t \in \mathcal{T}, \quad (3.14c)$$

$$\text{MFG-primal-feasibility } (Y)$$

$$(Z, V) \in \text{argmin ENG}(F)$$

The last constraint in MFG-Leader formulates the decision problem of ENG as a follower. Our proposed solution approach can be adapted to solve this model by modifying the restricted master problem. Specifically, we replace CORP's objective with the MFG objective, and change the MFG-optimality (Y) with MFG-primal-feasibility (Y) con-

Table 3.8: The difference in the optimal objective values of the CORP, MFG, and ENG between MFG-Leader and BPTM-Nash. The reported instances are in problem class C2.

	Ins. 1	Ins. 2	Ins. 3	Ins. 4	Ins. 5	Ins. 6
CORP (max profit)	-12%	-8%	-11%	-5 %	-6%	-9%
MFG (min cost)	-25%	-15%	-32%	-11 %	-5%	-22%
ENG (min tardiness)	105%	325%	-14%	0 %	21%	0%

Table 3.9: The difference in the optimal objective values of the CORP, MFG, and ENG between CORP-Release and BPTM-Nash. The reported instances are in problem class C2.

	Ins. 1	Ins. 2	Ins. 3	Ins. 4	Ins. 5	Ins. 6
CORP (max profit)	0%	-14%	0%	-12%	-23%	0%
MFG (min cost)	0%	2%	0%	1%	26%	0%
ENG (min tardiness)	0%	-100%	0%	29%	-100%	0%

straints in the restricted master problem $MP(J^k)$.

Table 3.8 shows the difference in the optimal objective values of the CORP, MFG, and ENG between BPTM-Nash and MFG-Leader for the six instances in problem class C2. As a leader, MFG can start producing new products as soon as they are ready without waiting CORP’s decision. Thus, the MFG objective function improves, i.e., manufacturing cost decreases, compared to the BPTM. CORP’s profit, on the other hand, decreases as expected because CORP has no control over the decisions of MFG and ENG in MFG-Leader. The impact on ENG is mixed. MFG’s leadership can help ENG by reducing tardiness in some cases, e.g., instances 3 and 4. It might also result in a worse situation for ENG where the tardiness increases significantly, e.g., instances 1 and 2. Another interesting observation pertains to the “stability” of the bilevel solution obtained through the leadership of CORP. If MFG and ENG realize that they can both improve their objective functions by bypassing the CORP’s control as in instances 3 and 4 in Table 3.8, they might collude to implement a solution that is not always in the best interest of CORP.

In BPTM-Nash, CORP releases new products to manufacturing. Specifically, CORP

chooses the values of the Y variables; if the right-hand-side of constraint (3.1c) is 1 for a new product p in period t , i.e., the product development has been completed, then CORP can release this product to manufacturing by setting the Y_{pt} variable to 1, or it can delay its manufacturing by setting the Y_{pt} variable to 0. An interesting model between BPTM-Nash and MFG-Leader arises when CORP has to release new products to manufacturing as soon as their developments are completed. This model, referred to as CORP-Release, can be formulated by adding the upper-level constraints $Y_{pt} \geq \sum_{\tau \leq t} Z_{p\tau}$ to BPTM-Nash for $p \in P$, $t \in \mathcal{T}$.

We numerically examine the impact of this new setting on the optimal objective values of the CORP, MFG, and ENG using six instances in problem class C2. As seen in Table 3.9, CORP's profit either stays the same or decreases in CORP-Release. However, this reduction is less than MFG-Leader because CORP can still try to optimize profit by deciding on the release of new products to manufacturing in CORP-Release. These results illustrate the impact of allocating decision authority differently among the decision makers.

3.6 Summary

We formulate a mixed-integer bilevel model with interdependent followers for capacity coordination during product transitions in a semiconductor manufacturing firm. Corporate management acts as a leader for MFG and ENG units who must coordinate production, prototype fabrication and new product development schedules. We propose two single-level reformulations that differ in how they enforce the ENG optimality. Our algorithm provides exact solutions to the single-level reformulation in which the optimality of the MFG solution is imposed by direct inclusion of the KKT optimality conditions, and that of the ENG solution by considering all possible product development schedules. and derive a solution algorithm based on constraint and column generation.

Chapter 4

Coordinating Resource Allocation and Product Transitions in Semiconductor Manufacturing Using Bilevel Programming Model

Abstract:

This chapter studies the product transition problem in semiconductor manufacturing firm which requires the coordination of multiple product divisions. We propose a multi-follower bilevel model to capture the hierarchical and decentralized nature of the product transition process. The corporate management is the leader who seeks to minimize the system cost and total allocated budget. The follower's problem consist of multiple product divisions who share factory and engineering resources to carry out their operations. Each product division needs engineering capacity to develop new products and requires factory capacity to fulfill the demand for current and new products. We model this interdependency as a generalized Nash equilibrium problem in the lower level and propose a reformulation where the leader coordinates the resource allocation decisions. We then derive an equivalent single level reformulation and develop a cut-and-column generation algorithm. We conduct extensive computational experiments to evaluate the performance of the algorithm, and to provide managerial insights on how the key parameters and competition affects the system cost.

4.1 Introduction

Product transition is the process of introducing new products to replace older generations. It involves the transformation of a market opportunity and product characteristics into a new product available for sale (Krishnan and Ulrich 2001). Product transition is increasingly gaining attention as the driving force for companies, especially in high-tech market, to maintain market share, sustain organic growth, and generate profit (Koca et al. 2010).

Efficient management of product transitions offers a significant competitive edge for companies to survive under global competition, keep up with the technology advancement, respond to dynamic marketplaces, and satisfy the changing customer demand. On the contrary, poor decisions can result in a failed product transition and pose serious business consequences, hence the product transition process involves highly complex decision making. Product transition is perhaps most challenging for high-tech companies who serve multiple markets each potentially with different product lines. A product transition in these companies involves the flow of information between units, timely access to material, and human resources to carry out the process. The process often requires factory capacity, skilled workers, resource allocation, and solid operations management to avoid failures and maintain the financial viability. The importance of product transition necessitates models that properly reflect the organizational structure in which the problem arises.

We focus on large semiconductor manufacturing companies in this study and try to present a model that better reflects the product transition. A key characteristics of such companies is the decentralized nature of the decision making process. Semiconductor companies consist of multiple functional groups, each with its own private information, objective, and operational constraints. Each functional group enjoys a level of autonomy to address its decision problems (Bansal et al. 2020b; Mönch et al. 2018). Therefore, treating the firm as a monolithic entity with complete information that takes centralized decision is neither realistic, nor desirable. A growing body of research has addressed aspects of product transition in a decentralized environment, including capacity planning under uncertainty (Karabuk and Wu 2002, 2003), supply chain coordination (Mallik and Harker 2004), capacity coordination and allocation through mechanism design (Karabuk and Wu 2005; Bansal et al. 2020b; Mallik 2007), the role of capacity conversion on the

equipment renewal and upgrade schedule (Li et al. 2014), and the impact of lead time on production (Kacar et al. 2016). This body of research, however, ignores the hierarchical nature of decision making environment in which product transition is managed (Ho et al. 2002); in large semiconductor manufacturing companies, corporate management lies at the top of this hierarchy and takes strategic/tactical decisions. The upper-level decisions then results in set of directives by which the lower-level functional groups take operational decisions. The hierarchical chain of decisions mandates a clear line of authority and control, ignoring which can render the model impractical and misleading.

The distribution of domain knowledge among different functional groups and line of authority governing the decision making environment requires extensive coordination across the firm (Bansal et al. 2020b). Although the specifics of product development and high volume production can differ from one semiconductor firm to another, there are two principal units posses the required resources and knowledge to fulfill a successful introduction of new products and phasing out the current ones: 1) corporate management (CORP) who leads the firms toward its strategic business goals; 2) Product Divisions (PDs) who manage all activities related to new product development and manufacturing. Each PD is responsible for a specific market segments each of which potentially with multiple product lines.

CORP distributes the available total operating budget and capacity of resources among PDs, who then use it to obtain resources required for their operations. PDs are organized as a revenue center for the firm. Given capacity release amount from the leader, PDs are responsible to use the allocated budget to obtain shared factory and engineering resources available across the firm to effectively conduct their operations. Each PD determines specifications of new products through market research and comes up with a detailed design for each product that outlines its features, required engineering resources, timeline for introduction, projected demand, and estimated costs (Rash and Kempf 2012). The design step is followed by prototype fabrication to identify potential failures and test the manufacturability of the product. Hence, each PD must secure sufficient factory capacity overtime to produce both its prototypes and current products. The factory capacity used for prototype fabrication limits the capacity available for producing current products to generate revenue and meet the demand, but is essential to survive in the market. The overarching rationale behind CORP's decisions is to maintain the long term profitability of the firm. However, CORP can only impact PDs decisions through

budget allocation. The substantial decision autonomy of PDs creates an environment in which the CORP does not have direct knowledge of operational constraints of each PD or the manner in which PDs compete for factory and engineering resources.

The combination of complex decision environment and autonomous units involved in product transition creates an extremely challenging problem. A simpler problem faced by PDs, even in its complete deterministic form, results in a generalized Nash equilibrium problem (GNEP) which is a highly challenging problem to solve (Facchinei and Kanzow 2010). In this chapter, we propose a bilevel model that captures the decentralized and hierarchical environment in which product transition take place in large semiconductor companies. The leader is CORP who minimizes the aggregated cost of PDs and allocated budget subject to the decision of the follower's problem. The follower's problem is a GNEP where PDs compete over factory and engineering resources to minimize their individual total cost over a finite planning horizon. Hence, the decisions of each PD depend on those of all other PDs which, unlike traditional bilevel models, yields a bilevel program with interdependent followers. The contribution of this chapter over previous work is as follows:

- Our model incorporates the main technological and operational constraints in the upper and lower level problems. It offers a framework to consider products that are not directly part of the production transition, but share factory facilities with new products.
- We formulate the problem as a multi-follower bilevel program where followers exchange information, thus making the lower-level problem a GNEP.
- We present a single-level equivalent reformulation of the bilevel model based on which we use to develop an efficient exact algorithm.
- We test the performance of the proposed algorithm through extensive numerical experiments and provide managerial insights.

The remainder of the chapter is organized as follows. Section 4.2 briefly reviews the relevant literature on product planning and transition in semiconductor manufacturing, GNEP, and bilevel programming. We formulate the problem in Section 4.3, and develop the solution approach in Section 4.4. Section 4.5 describes the numerical experiment

and presents numerical analysis and managerial insights. We conclude the chapter in Section 4.6 and discuss some research directions for future studies.

4.2 Literature Review

As the main motivation for this study, we start by reviewing the relevant work in production transition literature in Section 4.2.1 and narrow our focus to product planning in semiconductor firms in Section 4.2.2. Our model and solution approach relates to GNEP and bilevel programming whose literature are covered in Section 4.2.3 and 4.2.4, respectively.

4.2.1 Production Transition

Aspects of product transition has garnered attention of researchers in recent decades (Krishnan and Ulrich 2001; Ferrer and Swaminathan 2006; Gokpinar et al. 2010; León and Farris 2011). Ho et al. (2002) study supply-related decisions. (Wu et al. 2009) examine the impact of initial investment on the speed-to-market, quality improvement, and pricing of new products. The inventory planning decisions without replenishment is studied in Li et al. (2010) considering demand uncertainty and product substitution. Li et al. (2014) develop multi-period capacity planning model with uncertain demand and target service level. Druehl et al. (2009) study the timing of new product introduction and propose a model to identify factors that impact the pace of product transition. Some studies analyze the role of customers in the process. Liang et al. (2014) and Lobel et al. (2015) analyze the optimal product launch policies under different operational settings in the presence of strategic customers. Cui and Wu (2017) examine the impact of customer involvement in the success of new product development. Several address the pricing aspect of the process including interaction among timing, pricing, and product design when entering a new market (Klastorin and Tsai 2004), dynamic pricing (Koca et al. 2010), and price manipulation (Li et al. 2010).

4.2.2 Capacity Planning in Semiconductor Firms

Decentralized capacity planning in semiconductor manufacturing has been studied in a number of chapters. Mallik and Harker (2004) and Mallik (2007) examine capacity allocation among retailers under a competitive setting where multiple product lines request production capacity from manufacturing units. Incorporating demand and yield uncertainties via a discrete scenario scheme, Karabuk and Wu (2003) independently model the divisional decision problems of the marketing and production units. They force the capacity allocation and expansion decision to be consistent for both models and apply different recourse approximation schemes to decentralize the capacity planning process. In the subsequent chapter, they consider corporate management and production unit and develop a game-theoretical model to elicit private information from production unit to maximize the corporate's expected profit (Karabuk and Wu 2005). Bansal et al. (2020b) propose two iterative combinatorial auction schemes for a simpler version of the problem is considered in this chapter to coordinate the negotiation over factory capacity between manufacturing, acting as the auctioneer, and product development units who act as bidders. Although previous work address many challenges of product transition in the context of semiconductor manufacturing and provide useful insight, none of them enforce strict hierarchy nor consider product mix or interaction between decision maker.

4.2.3 Generalized Nash Equilibrium Problem

In the context of economic sciences, noncooperative games refer to a game where players engage in competition with each other. The Generalized Nash Equilibrium Problem (GNEP) is a noncooperative game in which the strategy of an individual player may depend on strategies of the rival players. A collection of strategies is called an *equilibrium* is attained when no player gains incremental benefit by making unilateral change of strategy. A collection of equilibrium points are called *equilibria*. As a mathematical optimization model, GNEP is a decentralized model whose decision makers share a set of constraints known as *coupling constraints*. It has been fruitfully applied to many optimization problems (Facchinei and Kanzow 2010), including transportation (Zhou et al. 2005; Stein and Sudermann-Merx 2018; Sagratella et al. 2020), electricity markets (Wang et al. 2021; Oggioni et al. 2012), and telecommunication networks (Altman and Wynter 2004b).

Solution methods for GNEP are predominantly limited to problems with players who control only continuous decision variables (Facchinei and Kanzow 2010; Facchinei et al. 2009, 2014; Aussel and Sagratella 2017). Although GNEP with integer variables naturally arise in many applications, its literature has remained limited because integer variables make the problem non-convex and non-differentiable for which no general algorithmic approach is developed yet (Sagratella et al. 2020). Sagratella (2017) offers the first treatment of GNEP with integer variables. They consider a special case of GNEP, generalized potential games, where each player unconsciously optimizes the same (potential) objective function over the aggregated strategy sets of all players. Under some assumptions, Sagratella (2019) establish the attainability of equilibria in a GNEP with mixed-integer variables when the coupling constraints are linear, and develop an algorithm that computes the approximate equilibrium points. Their algorithm is used in (Fabiani and Grammatico 2019) to develop a solution approach for a transportation problem with mixed-integer variables. More recently, Sagratella et al. (2020) introduce a noncooperative fixed charge transportation problem with different cost structures and constraints. They formulate the problem as a mixed-integer GNEP and establish some structural properties which allow them to apply the algorithms developed for pseudo Nash equilibrium problems, a concept that we will explain with more details later in this chapter.

4.2.4 Bilevel Programming

Bilevel programming (BP) is a nested optimization problem over a feasible region where the optimization problem of the lower level model (follower) defines a subset of the upper-level's (leader's) constraints (Bard 1998; Dempe 2002). BP can be optimistic or pessimistic. In the former case the leader select the most advantageous of follower's responses whereas in the latter the leader has to choose the least favorable follower's responses. Bilevel programs provides a powerful tool to model hierarchical decision process with two decision makers who share a set of common resources but pursue different objectives; the leader first makes a decision on its anticipation of the follower's response, then the follower responds by making a decision (Colson et al. 2007). It is this inherent hierarchy that opens BP to wide range of applications from network design (Dewez et al. 2008) to facility location (Caramia and Mari 2016), machine learning (Bennett et al. 2008) and healthcare (Özaltın et al. 2018).

Bilevel linear programs (BLP) where both leader's and follower's problems are linear programs (LPs) are so far the most studied problem in the literature of the BP (Hemmati and Smith 2016). Although BLPs are non-convex and strongly NP-hard, they enjoy properties that allows the optimality of the follower to be enforced through the Karush-Kuhn-Tucker (KKT) or strong duality conditions of the follower's problem (Bolusani and Ralphs 2021). The resulting nonlinear single-level problem can be solved by combination of branch-and-bound and implicit enumeration of extreme points (Kleinert et al. 2021).

Bard and Moore (1990) offer the first algorithmic approach to the mixed-integer bilevel linear programs (MIBLP) which is a BP that has integer variables in at least one decision level. They identify some node fathoming rules and develop a simple branch-and-bound method. Algorithmic development for the general MIBLP ceased for nearly two decades until DeNegre and Ralphs (2009) extended this work to propose a branch-and-cut algorithm and sparked a flurry of recent activity in the literature (Bolusani and Ralphs 2021). Wang and Xu (2017) propose a branch-and-bound algorithm with multi-way branching carried out on the slack variables. In subsequent work, the authors present a similar approach that remove bilevel infeasible solutions by multi-way branching disjunctions (Wang and Xu 2017). Lozano and Smith (2017) propose an algorithm based on the single-level value function reformulation. The use of intersection cuts in branch-and-cut algorithm is suggested in (Fischetti et al. 2018). Yue et al. (2019) propose a projection-based single-level reformulation that implicitly enumerates the follower's integer variables. Finally, Bolusani et al. (2020) utilize duality theory for MILP and develop a Benders-like decomposition algorithm that approximates the value function of the follower. MIBLP has successfully applied to several optimization problems (Kleinert et al. 2021; Dempe 2018). Hemmati and Smith (2016) consider a competitive prioritized set covering problem that arises in new product development and introduction in a competitive market; they formulate the problem as a MIBLP and develop a cutting-plane algorithm. Other applications of MIBLP include facility location (Beresnev and Melnikov 2018), transportation (Arslan et al. 2018), healthcare (Özaltın et al. 2018), supply chain design (Yue and You 2017), and electricity markets (Lavigne et al. 2000).

While the majority the of BP literature is devoted to the case with single follower, some studies address the lower level problem with more than one follower. Shi et al. (2007) study multi-follower BLP with partially shared variables among followers. Calvete et al. (2019) formulate a rank pricing problem as a nonlinear bilevel program with multiple

independent followers and develop a tailored branch-and-cut algorithm. Tavashoğlu et al. (2019) apply a generalized value function, which takes objective function gradient and right-hand-side of constraints as argument, to a multiple follower MIBLP where followers share the same constraints matrix. Basilico et al. (2020) focus on a BLP whose multiple followers play a Nash game. They established several structural properties and propose branch-and-bound method. Coniglio et al. (2020) investigate the pessimistic case of the same problem and offer a branch-and-bound scheme. To the best of our knowledge, MIBLP with multiple followers playing a GNEP has not been studied in the literature.

4.3 Model formulation

We formulate the product transition problem in semiconductor manufacturing as a bilevel program to capture its hierarchical and decentralized structure. The upper-level problem consists of the CORP problem whose objective is to minimize weighted cost of PDs as well as total allocated budget. As a leader, the CORP controls the budget allocation among PDs and capacity release for the usage of PDs. The allocated budget allows PDs to procure the factory and engineering capacity required for product development and manufacturing operations. The lower-level problem is a generalized Nash game played between multiple PDs. Constraints concerning the usage of factory and engineering capacity are the coupling constraints of the GNEP. Each PD seeks to minimize its total cost which includes lost revenue and the costs of production, backorder, and inventory. The dynamics of the proposed bilevel model are illustrated in Figure 4.1.

Let $\mathcal{T} = \{1, \dots, T\}$ be the set of planning periods. We use \mathfrak{B} to denote the total budget of the firm over the planning horizon. We denote the set of PDs by $\mathcal{J} = \{1, \dots, J\}$, set of all current and new products by N , and the set of new products that are currently under development by $P \subset N$. We define N_j and P_j to denote the set of all products and new products of PD $_j$, respectively. Other parameters are defined in Table 4.1. We define the integer variable \mathcal{B}_j to model the CORP's budget allocation to PD $_j$. Furthermore, we define integer variables S_t^f and S_t^e to represent the factory capacity and engineering capacity released by CORP for the usage of PDs in each period t . We use bold face symbols to refer to vectors of variables, for example \mathbf{B} is the vector of \mathcal{B}_j variables.

Given the upper-level budget allocation \mathbf{B} and capacity release decisions $(\mathbf{S}^f, \mathbf{S}^e)$,

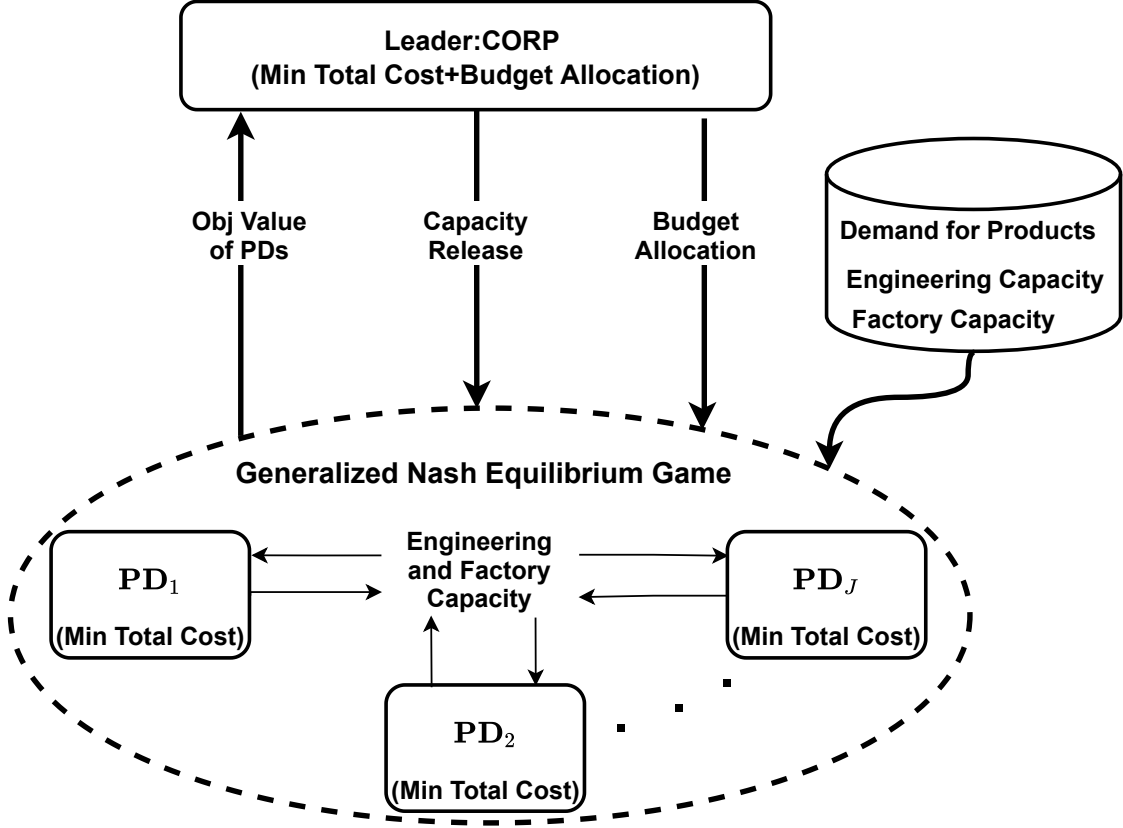


Figure 4.1: General flow of resources and information among decision units in the proposed bilevel model. Given the upper-level budget allocation and capacity release decisions, PDs compete among themselves over the factory and engineering resources as depicted by solid arrows in the lower level problem. In each period, the availability of engineering and factory capacity, and demand for each products are given as parameters. Once PDs reach an equilibrium, they communicate their objective values to the leader.

Table 4.1: Parameters of the bilevel product transition model for each PD_j

w_j	Weight of total cost of PD_j in the objective function of the leader
D_{jnt}	Demand for product n in period t
π_{jnt}	Per unit revenue of product n in period t
r_{jnt}	Per unit production cost of product n in period t
H_{jp}^f	Factory capacity required for prototype testing of product p
H_{jp}^e	Engineering capacity required for developing product p
h_{jnt}	Holding cost of product n in period t
R_t^f	Total factory capacity available in period t
R_t^e	Total engineering capacity available in period t
c_j^f	Unit factory capacity cost
c_j^e	Unit engineering capacity cost

we use $\text{GNEP}(\mathcal{B}, \mathcal{S}^f, \mathcal{S}^e)$ to denote the generalized Nash equilibrium game as well as its equilibria in the lower-level. The outcome of $\text{GNEP}(\mathcal{B}, \mathcal{S}^f, \mathcal{S}^e)$ is an equilibrium in which each PD has an objective value of ϕ_j^* . We refer to this product transition model as PTM-Nash, and present its formulation as follows:

$$[\text{PTM-Nash}] \quad \min \theta = \sum_{j \in \mathcal{J}} w_j \phi_j + \eta \sum_{j \in \mathcal{J}} \mathcal{B}_j \quad (4.1a)$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{J}} \mathcal{B}_j \leq \mathfrak{B} \quad (4.1b)$$

$$S_t^f \leq R_t^f \quad t \in \mathcal{T} \quad (4.1c)$$

$$S_t^e \leq R_t^e \quad t \in \mathcal{T} \quad (4.1d)$$

$$\phi \in \text{GNEP}(\mathcal{B}, \mathcal{S}^f, \mathcal{S}^e) \quad (4.1e)$$

$$\mathfrak{B}_j, S_t^f, S_t^e \in \mathbb{Z}^+ \quad j \in \mathcal{J}, t \in \mathcal{T} \quad (4.1f)$$

The objective function (4.1a) minimizes the weighted total cost of all PDs, and the total budget allocation weighted by η . Constraint (4.1b) ensures that the total available budget is not exceeded. Constraints (4.1c) and (4.1c) restrict the factory and engineering capacity release to the available amount. The generalized Nash game between PDs in the lower level is represented by (4.1e). PTM-Nash is an optimistic bilevel model because it allows the leader, CORP, to select the most favorable lower-level equilibrium when there are multiple ones (Kleinert et al. 2021).

$\text{GNEP}(\mathcal{B}, \mathcal{S}^f, \mathcal{S}^e)$ involves a set of PD problems with the same constraint structure and objective function but with different parameters. For each PD_j and every product n in period t , let X_{jnt} , I_{jnt} and B_{jnt} to denote the factory capacity usage, inventory level, and backorder level, respectively. Furthermore, let R_{jt}^f and R_{jt}^e denote the the factory and engineering capacities obtained by PD_j in period t . We define the binary variable Z_{jpt} to be 1 if the new product p is developed in period t by PD_j , and 0 otherwise. We then formulate each PD_j problem in $\text{GNEP}(\mathcal{B}, \mathcal{S}^f, \mathcal{S}^e)$ as follows:

$$[\text{PD}_j(\mathcal{B}, \mathcal{S}^f, \mathcal{S}^e)] \quad \min \phi_j = \sum_{n \in N^j} \sum_{t \in \mathcal{T}} [\pi_{jnt}(B_{jnt} - B_{jn,t-1}) + h_{jnt}I_{jnt} + r_{jnt}X_{jnt} + b_{jnt}B_{jnt}] \quad (4.2a)$$

$$\text{s.t.} \quad \sum_{i \in \mathcal{J}} R_{it}^f = S_t^f \quad t \in \mathcal{T} \quad (4.2b)$$

$$\sum_{i \in \mathcal{J}} R_{it}^e = S_t^e \quad t \in \mathcal{T} \quad (4.2c)$$

$$\sum_{t \in \mathcal{T}} R_{jt}^f c_{jt}^f + \sum_{t \in \mathcal{T}} R_{jt}^e c_{jt}^e \leq \mathcal{B}_j \quad (4.2d)$$

$$\sum_{p \in P^j} H_{jp}^f Z_{jpt} + \sum_{n \in N^j} X_{jnt} \leq R_{jt}^f \quad t \in \mathcal{T} \quad (4.2e)$$

$$\sum_{p \in P^j} H_{jp}^e Z_{jpt} \leq R_{jt}^e \quad t \in \mathcal{T} \quad (4.2f)$$

$$I_{jnt} = I_{jn,t-1} + X_{jnt} - (D_{jnt} + B_{jn,t-1} - B_{jnt}) \quad n \in N_j, t \in \mathcal{T} \quad (4.2g)$$

$$\sum_{t \in \mathcal{T}} Z_{jpt} \leq 1 \quad p \in P_j \quad (4.2h)$$

$$X_{jpt} \leq R_t^f \sum_{\tau \leq t} Z_{jp\tau} \quad p \in P^j, t \in \mathcal{T} \quad (4.2i)$$

$$I_{jnt}, X_{jnt}, B_{jnt}, R_{jt}^e, R_{jt}^f \in \mathbb{Z}^+, Z_{jpt} \in \{0, 1\} \quad \forall t \in \mathcal{T}, n \in N^j, p \in P^j \quad (4.2j)$$

The objective function (4.2a) minimizes the sum of lost revenue, inventory, production, and backorder costs over the planning horizon. Constraints (4.2b) and (4.2c) are the coupling constraints. They ensure that the factory and engineering usage by PDs do not exceed the capacity made available by the CORP in each period. Note that these constraints include factory and engineering capacity usage of all PDs. Therefore, the feasible region of each PD problem is dependent on both the upper-level decisions and the decisions of other PDs. Constraint (4.2d) limits the cost of procuring factory and engineering capacities over the planning horizon by \mathcal{B}_j , the budget allocation from CORP. Constraints (4.2e) ensure that the capacity used for prototype fabrication, i.e. $\sum_{p \in P^j} H_{jp}^f Z_{jpt}$, plus the factory capacity used for manufacturing, i.e. $\sum_{n \in N^j} X_{jnt}$, is less than the procured factory capacity R_{jt}^f . Similarly, constraints (4.2f) restrict the engineering capacity usage for new product development in each time period. Constraints (4.2g) are the inventory balance equations across the time periods. Constraints (4.2h) ensure that each new product is only developed once in the planning horizon. This assumption can be relaxed by considering multi-period product development; the proposed bilevel framework would still be applicable. Finally, constraints (4.2i) ensures that new products can only be manufactured after development. We refer to constraints (4.2b)-(4.2j) as $\text{PD}_j\text{-feasibility}(\mathcal{B}, \mathcal{S}^f, \mathcal{S}^e)$. We also define $\text{PD}_j\text{-consts}(\mathcal{B}, \mathcal{R}^f, \mathcal{R}^e)$ to refer to constraints(4.2d)-(4.2j) in which budget, factory capacity, and engineering capacity allocated

to PD_j are parameters.

4.4 Solution Approach

Our solution approach is based on the concept of pseudo-Nash equilibrium described in Section 4.4.1. We propose a single-level reformulation of PTM-Nash in Section 4.4.2. Finally, we present the steps of the proposed solution algorithm in Section 4.4.3.

4.4.1 Pseudo-Nash Equilibrium Games

Let \mathcal{P} be the set of players in a generalized Nash equilibrium game. Let x^ν and $x^{-\nu}$ be the set of variables controlled by player $\nu \in \mathcal{P}$ and other players in $\mathcal{P} \setminus \{\nu\}$, respectively. Furthermore, let $K_\nu(x^{-\nu})$ and $S_\nu(x^{-\nu})$ be the strategy set and the response set of player ν , respectively. A solution $\hat{x} = (\hat{x}^\nu, \hat{x}^{-\nu})$ is a generalized Nash equilibrium if and only if it satisfies $\hat{x}^\nu \in S_\nu(\hat{x}^{-\nu})$ for all $\nu \in \mathcal{P}$. In other words, $\hat{x} \in S(\hat{x}) \equiv \Pi_{\nu \in \mathcal{P}} S_\nu(\hat{x}^{-\nu})$. Any generalized Nash equilibrium $\hat{x} = (\hat{x}^\nu, \hat{x}^{-\nu})$ satisfies $\hat{x}^\nu \in K_\nu(\hat{x}^{-\nu})$ for all $\nu \in \mathcal{P}$ since $S_\nu(x^{-\nu}) \subseteq K_\nu(x^{-\nu})$. That is, $\hat{x} \in K(\hat{x}) \equiv \Pi_{\nu \in \mathcal{P}} K_\nu(\hat{x}^{-\nu})$. We denote by $FP(S) \equiv \{x : x \in S(x)\}$ the set of fixed points of the point-to-set mapping S . Note that $FP(S)$ corresponds to the set of all generalized Nash equilibrium solutions. Similarly, we denote by $FP(K) \equiv \{x : x \in K(x)\}$ the set of fixed points of the point-to-set mapping K .

Definition 4.4.1. (*Aussel and Sagratella 2017*) *A generalized Nash equilibrium game is a pseudo-Nash equilibrium game if its set of equilibrium solutions $FP(S)$ is equivalent to $\cup_{x \in K(x)} S(x)$.*

Based on Definition 4.4.1, an equilibrium of a pseudo-Nash game can be computed by identifying a solution $\hat{x} \in K(\hat{x})$ and then finding the response set $S_\nu(\hat{x}^{-\nu})$ of each player $\nu \in \mathcal{P}$. Note that $S_\nu(\hat{x}^{-\nu})$ does not depend on the decisions of other players once $\hat{x}^{-\nu}$ is fixed. Aussel and Sagratella (2017) showed that any generalized Nash equilibrium game with linear objective functions and linear constraints whose coupling constraints are, exclusively, equality constraints is a pseudo-Nash equilibrium game.

Proposition 4.4.1. *$GNEP(\mathcal{B}, \mathbf{S}^f, \mathbf{S}^e)$ is a pseudo-Nash equilibrium game.*

Proof. Each PD in $\text{GNEP}(\mathcal{B}, \mathcal{S}^f, \mathcal{S}^e)$ has a linear objective function and linear constraints. Moreover, coupling constraints are exclusively equality constraints. \square

This proposition establishes the attainability of an equilibrium from a feasible solution to the lower-level problem. Given budget allocation and capacity release, consider the following *equilibrium problem (EP)* whose objective function is the linear combination of objective functions of PD models, and whose constraints consist of all constraints in the lower level problem.

$$\begin{aligned}
 [\text{EP}(\mathcal{B}, \mathcal{S}^f, \mathcal{S}^e)] \min & \sum_{j \in \mathcal{J}} \sum_{n \in N^j} \sum_{t \in \mathcal{T}} w_j [\pi_{jnt}(B_{jnt} - B_{jn,t-1}) + h_{jnt}I_{jnt} + r_{jnt}X_{jnt} + b_{jnt}B_{jnt}] \\
 \text{s.t. PD}_j\text{-feasibility} & (\mathcal{B}, \mathcal{S}^f, \mathcal{S}^e) \qquad \qquad \qquad j \in \mathcal{J}
 \end{aligned}$$

The solution of $\text{EP}(\mathcal{B}, \mathcal{S}^f, \mathcal{S}^e)$ gives an equilibrium of $\text{GNEP}(\mathcal{B}, \mathcal{S}^f, \mathcal{S}^e)$. However, this solution may not be an optimistic response to the leader because it does not optimize any component of the upper-level's objective function. In an optimistic response, CORP should be able to select the most advantageous equilibrium of the lower-level problem.

Proposition 4.4.2. *Given a budget allocation and capacity release, i.e. $(\mathcal{B}, \mathcal{S}^f, \mathcal{S}^e)$, there is an equilibrium of $\text{GNEP}(\mathcal{B}, \mathcal{S}^f, \mathcal{S}^e)$ corresponding to any nonnegative $(\hat{\mathcal{R}}^f, \hat{\mathcal{R}}^e)$ such that:*

$$\begin{aligned}
 \sum_{i \in \mathcal{J}} \hat{R}_{it}^f &= S_t^f & t \in \mathcal{T} \\
 \sum_{i \in \mathcal{J}} \hat{R}_{it}^e &= S_t^e & t \in \mathcal{T}
 \end{aligned}$$

Proof. The proof directly follows from Theorem 4 of Aussel and Sagratella (2017) and Proposition 4.4.1.

As a result of Proposition 4.4.2, the leader can choose an equilibrium by setting the factory and engineering allocation for each PD_j at each time period t . We use this fact to formulate the following problem:

$$[\text{PTM}] \quad \min \theta = \sum_{j \in \mathcal{J}} w_j \phi_j + \eta \sum_{j \in \mathcal{J}} \mathcal{B}_j \quad (4.5a)$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{J}} \mathcal{B}_j \leq \mathfrak{B} \quad (4.5b)$$

$$\sum_{i \in \mathcal{J}} R_{it}^f \leq R_t^f \quad t \in \mathcal{T} \quad (4.5c)$$

$$\sum_{i \in \mathcal{J}} R_{it}^e \leq R_t^e \quad t \in \mathcal{T} \quad (4.5d)$$

$$\phi_j \in \text{argmin PD}_j\text{-optimality}(\mathcal{B}, \mathbf{R}^f, \mathbf{R}^e) \quad j \in \mathcal{J} \quad (4.5e)$$

where R_{it}^f and R_{it}^e are now upper-level variables and $\text{PD}_j\text{-optimality}(\mathcal{B}, \mathbf{R}^f, \mathbf{R}^e)$ is formulated as $\min\{\phi_j : \text{PD}_j\text{-consts}(\mathcal{B}, \mathbf{R}^f, \mathbf{R}^e)\}$.

Proposition 4.4.3. *PTM is equivalent to PTM-Nash*

Proof. We establish the proof by showing that any feasible solution of PTM corresponds to a feasible solution of PTM-Nash and vice versa. Let $(\hat{\mathcal{B}}, \hat{\mathbf{R}}^f, \hat{\mathbf{R}}^e)$ be a feasible solution to PTM. Let $(\hat{\mathcal{S}}^f, \hat{\mathcal{S}}^e)$ be

$$\hat{\mathcal{S}}_t^f = \sum_{j \in \mathcal{J}} \hat{R}_{it}^f \quad t \in \mathcal{T}$$

$$\hat{\mathcal{S}}_t^e = \sum_{j \in \mathcal{J}} \hat{R}_{it}^e \quad t \in \mathcal{T}$$

$(\hat{\mathcal{B}}, \hat{\mathcal{S}}^f, \hat{\mathcal{S}}^e)$ satisfy constraints (4.1b)-(4.1d). It also satisfies the optimality condition of each PD problem by Constraint (4.5e). Note that

$$\hat{R}_{it}^f = \hat{\mathcal{S}}_t^f - \sum_{j \in \mathcal{J} \setminus i} \hat{R}_{jt}^f \quad t \in \mathcal{T}$$

$$\hat{R}_{it}^e = \hat{\mathcal{S}}_t^e - \sum_{j \in \mathcal{J} \setminus i} \hat{R}_{jt}^e \quad t \in \mathcal{T}$$

Therefore, if $\hat{\phi}_j \in \text{argmin PD}_j\text{-optimality}(\hat{\mathcal{B}}, \hat{\mathbf{R}}^f, \hat{\mathbf{R}}^e)$, then $\hat{\phi}_j \in \text{argmin PD}_j(\hat{\mathcal{B}}, \hat{\mathcal{S}}^f, \hat{\mathcal{S}}^e)$, hence $(\hat{\mathcal{B}}, \hat{\mathcal{S}}^f, \hat{\mathcal{S}}^e)$ is an equilibrium of $\text{GNEP}(\hat{\mathcal{B}}, \hat{\mathcal{S}}^f, \hat{\mathcal{S}}^e)$ and a feasible solution to PTM-Nash.

Now let $(\mathbf{B}', \mathbf{R}'^f, \mathbf{R}'^e, \mathbf{S}'^f, \mathbf{S}'^e)$ be a feasible solution to PTM-Nash. This solution satisfies the upper level constraints (4.5b)-(4.5d). It also satisfies constraints (4.5e) since each PD's response is optimal in an equilibrium of $\text{GNEP}(\mathbf{B}', \mathbf{S}'^f, \mathbf{S}'^e)$. Note that, for every $\text{PD}_j(\mathbf{B}', \mathbf{S}'^f, \mathbf{S}'^e)$ we have

$$\begin{aligned} S_t'^f &= \sum_{j \in \mathcal{J}} R_{jt}'^f & t \in \mathcal{T} \\ S_t'^e &= \sum_{j \in \mathcal{J}} R_{jt}'^e & t \in \mathcal{T} \end{aligned}$$

Therefore, if $\phi'_j \in \text{argmin PD}_j(\mathbf{B}', \mathbf{S}'^f, \mathbf{S}'^e)$, then

$$\phi'_j \in \text{argmin PD}_j\text{-optimality}(\mathbf{B}', \mathbf{R}'^f, \mathbf{R}'^e)$$

hence $(\mathbf{B}', \mathbf{R}'^f, \mathbf{R}'^e)$ is a feasible solution to PTM. We use the equivalency established in Proposition 4.4.3 to develop a single-level reformulation of the problem.

4.4.2 Single-level Reformulation

We present a single-level equivalent reformulation of PTM-Nash through the enumeration of all possible of budget allocations. Let \mathcal{G} denote the set of all possible distinct budget allocations by the CORP to J number of PDs and G be its index set. Then, \mathcal{G} is equivalent to weak composition of \mathfrak{B} into $J + 1$ nonnegative integers (Sagan 2020), and its size is given by

$$\binom{\mathfrak{B} + J}{J} = \frac{(\mathfrak{B} + J)!}{(\mathfrak{B})!J!}.$$

The size of \mathcal{G} depends both on the number of PDs and the total available budget \mathfrak{B} , but it is finite. CORP has at least one optimal factory and engineering capacity allocation schedule denoted by $\hat{\mathbf{R}}_g = (\hat{\mathbf{R}}_g^f, \hat{\mathbf{R}}_g^e)$ corresponding to each budget allocation $\hat{\mathbf{B}}_g$ in $g \in \mathcal{G}$. Given $(\mathcal{B}_g, \hat{\mathbf{R}}_g^f, \hat{\mathbf{R}}_g^e)$, we can solve the $\text{PD}_j\text{-optimality}(\mathcal{B}_g, \hat{\mathbf{R}}_g^f, \hat{\mathbf{R}}_g^e)$ to obtain the corresponding optimal objective function of the PD, ϕ_{gj}^* . Note that for any given budget, and resource allocation, the lower level problem is feasible.

Thus, the upper-level solutions can be enumerated over budget allocations. The following is a single-level reformulation of PTM where the optimality of every PD for each possible budget allocation is enforced through a set of constraints and variables:

$$[\text{MP}(G)] \quad \min \quad \theta = \sum_{j \in \mathcal{J}} w_j \phi_j + \eta \sum_{j \in \mathcal{J}} \mathcal{B}_j \quad (4.6a)$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{J}} \mathcal{B}_j \leq \mathfrak{B} \quad (4.6b)$$

$$\sum_{i \in \mathcal{J}} R_{it}^f \leq R_t^f \quad t \in \mathcal{T} \quad (4.6c)$$

$$\sum_{i \in \mathcal{J}} R_{it}^e \leq R_t^e \quad t \in \mathcal{T} \quad (4.6d)$$

$$\text{PD}_j\text{-const}(\mathcal{B}, \mathbf{R}^f, \mathbf{R}^e) \quad j \in \mathcal{J} \quad (4.6e)$$

$$\phi_j = \sum_{n \in N^j} \sum_{t \in \mathcal{T}} [\pi_{jnt}(B_{jnt} - B_{jn,t-1}) + \quad (4.6f)$$

$$h_{jnt}I_{jnt} + r_{jnt}X_{jnt} + b_{jnt}B_{jnt}] \quad j \in \mathcal{J}$$

$$R_{jt}^e \geq \hat{R}_{gjt}^e(1 - \alpha_{gj}) \quad g \in G, j \in \mathcal{J}, t \in \mathcal{T} \quad (4.6g)$$

$$R_{jt}^f \geq \hat{R}_{gjt}^f(1 - \alpha_{gj}) \quad g \in G, j \in \mathcal{J}, t \in \mathcal{T} \quad (4.6h)$$

$$\sum_{j \in \mathcal{J}} (|\mathcal{B}_j - \hat{\mathcal{B}}_{gj}|) \geq \frac{\sum_{j \in \mathcal{J}} \alpha_{gj}}{J} \quad g \in G \quad (4.6i)$$

$$\phi_j \leq \phi_{gj}^* + M\alpha_{gj} \quad g \in G, j \in \mathcal{J} \quad (4.6j)$$

$$\alpha_{gj} \in \{0, 1\}, \mathfrak{B}_j, R_t^f, R_t^e \in \mathbb{Z}^+ \quad g \in G, j \in \mathcal{J}, t \in \mathcal{T} \quad (4.6k)$$

The objective value of PD_j in the equilibrium of $\text{GNEP}(\hat{\mathcal{B}}_g, \mathbf{S}^f, \mathbf{S}^e)$ is denoted by ϕ_{gj}^* for budget allocation $g \in G$. The mixed-integer program $\text{MP}(G)$ determines the value of upper level and lower level variables. The lower level feasibility is ensured by $\text{PD}_j\text{-const}(\mathcal{B}, \mathbf{R}^f, \mathbf{R}^e)$ for $j \in \mathcal{J}$, and optimality of each PD enforced by constraints (4.6g)-(4.6j). Specifically, binary variable α_{gj} takes the value of 0 if sufficient factory and engineering resources is allocated to allocated to PD_j to pursue the resource purchase plan g . In other words, if $\alpha_{gj} = 0$, both constraints (4.6g) and (4.6h) are binding, and the allocated resources in every time period t must be more than what PD_j requires to achieve the objective value of ϕ_{gj}^* . If enough resources is allocated for every time period, then constraint (4.6j) ensures that ϕ_j , does not exceed the optimal objective value of the PD_j under plan g , i.e. $\phi_j \leq \phi_{gj}^*$, hence the optimality of PD_j is enforced. On the other hand, if $\alpha_{gj} = 1$, then constraints (4.6g) and (4.6h) are not binding, and there must be time

period in which sufficient resources are not allocated to carry out the purchase plan g . In this case constraint (4.6i) secures the generation of new set of budget allocation. More specifically, the right hand side of the constraint (4.6i) becomes more than zero which then enforces the absolute difference of the budget allocation (i.e. \mathcal{B}_j) and allocation plan g (i.e. $\hat{\mathcal{B}}_{gj}$ to be at least one. This results in a new distinct set of budget allocation \mathcal{B} .

Proposition 4.4.4. *MP(G) is equivalent to the PTM-Nash*

Proof. We establish the equivalency by showing that both models have the same constraint spaces and objective functions. Let \mathcal{X} represent the set of all decision variables of PTM, and let $\bar{\mathcal{X}}$ be its feasible solution. Define

$$\bar{\alpha}_{gj} = \begin{cases} 0 & \text{if } R_{jt}^f \geq \hat{R}_{gjt}^f \text{ and } R_{jt}^e \geq \hat{R}_{gjt}^e \\ 1 & \text{otherwise} \end{cases}$$

$(\bar{\mathcal{X}}, \bar{\alpha})$ satisfy constraints (4.6g) and (4.6h) by construction. It also satisfies (4.6j) as $\bar{\phi}_{gj} \in \text{argmin PD}_j\text{-optimality}(\mathcal{B}, \mathbf{R}^f, \mathbf{R}^e)$. Hence $(\bar{\mathcal{X}}, \bar{\alpha})$ is a feasible solution to MP(G). Now consider a feasible solution $(\hat{\mathcal{X}}, \hat{\alpha})$ to MP(G). The solution $\hat{\mathcal{X}}$ is feasible to upper and lower level constraints of PTM. The index set of all feasible resource allocations to PD $_j$ -optimality($\hat{\mathcal{B}}, \hat{\mathbf{R}}^f, \hat{\mathbf{R}}^e$) is given by $\hat{G} = \{g \in G | R_{jt}^f \geq \hat{R}_{gjt}^f \text{ and } R_{jt}^e \geq \hat{R}_{gjt}^e\}$ then $\hat{\alpha} = 0$ for every $k \in \hat{G}$ due to constraints (4.6g) and (4.6h), thus constraints (4.6j) ensures that $\hat{\phi}_j \leq \phi_{gj}^*$ which makes the $\hat{\mathcal{X}}$ a feasible solution to PTM. The proof completes as both models share the same objective function. \square

4.4.3 Cut-and-Column Generation Algorithm

MP(G) offers a reformulation of the problem with an additional set of constraints whose size grow exponentially with the number of time periods and PDs. Consequently, the problem becomes intractable even for small-size instances. We ameliorate this issue by proposing a cut-and-column generation (CCG) algorithm that starts with a subset of constraints (4.6g)-(4.6j) and adds more such constraints in an iterative framework. The algorithm solves restricted master problem MP(G^k) in each iteration k where $G^k \subseteq G$ and $G^0 = \emptyset$. The optimal objective value of the MP(G^k), θ^k , is a lower bound on the optimal objective value θ^* as the feasible region of MP(G^k) contains that of MP(G). Let $\bar{\mathcal{X}}^k$ be the solution of MP(G^k) in iteration k . We first solve PD $_j$ -optimality($\bar{\mathcal{B}}, \bar{\mathbf{R}}^f, \bar{\mathbf{R}}^e$)

to determine $\bar{\phi}_j^*$, the objective function of PD_j under the budget and resource allocation $(\bar{\mathcal{B}}, \bar{\mathcal{R}}^f, \bar{\mathcal{R}}^e)$. If $\bar{\phi}_j = \bar{\phi}_j^*$ for all $j \in \mathcal{J}$, then the allocated budget and resources are optimal for PD_j to achieve its objective value, hence $(\bar{\mathcal{B}}, \bar{\mathcal{R}}^f, \bar{\mathcal{R}}^e)$ is bilevel feasible. This solution is also bilevel optimal because because the feasible region of the restricted master problem $\text{MP}(G^k)$ contains the feasible region of $\text{MP}(G)$. If, on the other hand,

$$\bar{\phi}_j > \bar{\phi}_j^*$$

for any $j \in \mathcal{J}$, we define a new α_{gj} and add set of new constraints in the form of constraints (4.6g)-(4.6j) to the restricted master problem. Algorithm 3 presents the CCG algorithm.

Algorithm 3: Constraint and column generation (CCG) algorithm

```

1 Initialization: set  $k = 0, G^k = \emptyset$ 
2 while  $G^k \subsetneq G$  do
3   Solve  $\text{MP}(G^k)$ .
4   if  $\text{MP}(G^k)$  is infeasible or  $G^k = G$  then
5     |  $\text{MP}(G)$  is infeasible, STOP.
6   end
7   Let  $\bar{\mathcal{X}}^k = (\bar{\mathcal{B}}_j^k, \bar{R}_{jt}^f, \bar{R}_{jt}^e, \bar{Z}_{jpt}, \bar{X}_{jnt}, \bar{I}_{jnt}, \bar{B}_{jnt}, \bar{\phi})$  be the optimal solution to
    $\text{MP}(G^k)$ .
8   For all  $j \in \mathcal{J}$ , solve  $\text{PD}_j$ -optimality $(\bar{\mathcal{B}}, \bar{\mathcal{R}}^f, \bar{\mathcal{R}}^e)$  to get the optimal objective
   value  $\phi_j^k$ .
9   if  $\bar{\phi}_j^k = \phi_j^k$  for all  $j \in \mathcal{J}$  then
10    | Return the optimal bilevel solution  $\bar{\mathcal{X}}^k$ .
11  end
12  else
13    | Add constraint (4.6g) to (4.6j) to  $\text{MP}(G^k)$ .
14  end
15  Set  $G^k = G^k \cup \{\bar{\mathcal{B}}_j, \bar{R}_{jt}^f, \bar{R}_{jt}^e\}$ ,  $k = k + 1$ .
16 end

```

Proposition 4.4.5. *Algorithm 3 terminates finite number of iterations.*

Proof. At each iteration a new set of \mathcal{B} is generated where they form a weak composition of \mathfrak{B} . The finiteness of the algorithm is established as size of all possible distinct budget

allocations (i.e. \mathcal{G}) is finite.

Proposition 4.4.6. *Algorithm 3 either returns the optimal solution to $MP(G)$ or identifies infeasibility.*

Proof. $MP(G^k)$ is a relaxation of $MP(G)$, so if infeasibility of the former establishes the infeasibility of the latter, thus the algorithm would terminate in line 5. If the condition in line 9 is satisfied, the solution $\bar{\mathcal{X}}^k$ is a bilevel feasible because PD_j problems are in optimality, thus $\bar{\phi}_j^k \in \text{argmin } PD_j\text{-optimality}(\bar{\mathcal{B}}, \bar{\mathbf{R}}^f, \bar{\mathbf{R}}^e)$. It is also bilevel optimal as $MP(G^k)$ is a relaxation of $MP(G)$ whose objective function is a lower bound to that of $MP(G)$. If the infeasibility of $MP(G^k)$ or optimality of $\bar{\mathcal{X}}^k$ are not confirmed, the algorithm will continue generating all feasible budget allocations at which point $G^k = G$ and the algorithm will stop at line 5 by announcing the infeasibility of the problem.

4.5 Computational Experiments

This section examines the performance of the proposed approach on randomly generated instances. We start by describing the procedure by which we generate random instances and briefly explain the implementation of the algorithm. We then report the results of our computational study and proving managerial insights by examining the impact of competition.

4.5.1 Instance Generation and Implementation Details

Each product's demand is randomly generated from one of the four discrete uniform distributions presented in Table 4.3 to reflect the fact that semiconductor companies operate in different market segments (Rash and Kempf 2012). The demand pattern of each product resembles the realistic pattern observed in semiconductor companies where demand for current products diminish to zero over time and the demand for new products increase for a few periods and then stabilizes. Table 4.2 present the details of positive demand periods for current product ($dEnd$), the starting time when the first demand for current products starts is observed, positive demand periods for new products ($dStr$), and the periods in which their demand is stabilized.

Table 4.2: Parameter values

Current product demand end period ($dEnd$)	U[0.5T, T]
Current product demand diminish start period	U[0.3T, $dEnd - 1$]
New product demand start period ($dStr$)	U[2, 0.4T]
New product stabilized demand start period	U[$dStr + 1$, 0.7T]

Table 4.3: Demand Distributions

Demand in Market 1	U[10, 25]	π (revenue per unit)	U[1, 3]
Demand in Market 2	U[20, 50]	h (holding cost)	U[0.1, 0.5]
Demand in Market 3	U[50, 150]	r (production cost)	$\pi \times$ U[0.2, 0.8]
Demand in Market 4	U[100, 300]	b (backorder cost)	$\pi \times$ U[7, 15]

The total available factory capacity in each period is generated randomly from a discrete uniform distribution between $[0.7 \sum_{j \in \mathcal{J}} \sum_{n \in N_j} D_{nt}]$ and $[0.9 \sum_{j \in \mathcal{J}} \sum_{n \in N_j} D_{nt}]$ so that, on average, the factory capacity is sufficient for only %80 of total possible capacity requests. Likewise, the total available engineering capacity in each period is generated randomly from a discrete uniform distribution between 30 and 50. Once the factory and engineering capacities are determined, the values of H_{jp}^f and H_{jp}^e , i.e. required factory and engineering capacities for prototype testing of product p in PD $_j$, are generated; to resemble a realistic situation, we first calculate the average factory and engineering resources available for each PD in each time period. This value is obtained by summing over available factory and engineering resources across the planning horizon and then dividing the values by the number of periods and number of PDs. H_{jp}^f and H_{jp}^e are then generated by multiplying the average resources by a random number Hr generated from a normal distribution $\mathcal{N}(\mu, \sigma)$ with mean μ and standard deviation σ . Unit manufacturing and engineering costs (i.e. c_j^f and c_j^e) are set to 1 and 30 respectively. Each w_j is generated from a uniform distribution between 0 and 1, such that $\sum_{j \in \mathcal{J}} W_j = 1$. The weight of the allocated budget in the leader's objective function, η , is set to 10 in all instances. Eventually, the total available budget is generated by multiplying a coefficient Bc to the cost of using all available factory and engineering resources in all periods, i.e. $Bc \times \sum_{j \in \mathcal{J}} \sum_{t \in \mathcal{T}} (R_t^f c_j^f + R_t^e c_j^e)$. The value of other parameters are presented in Table 4.3.

Table 4.4: Set of values and distributions considered to generate instances

Parameter or coefficient	set of values or distributions
T	{8, 12, 16}
J	{4, 8, 12, 16, 20}
$N_j \quad \forall j \in \mathcal{J}$	10
$P_j \quad \forall j \in \mathcal{J}$	{3, 7}
Total budget generation coefficient, Bc	{0.3, 0.4, 0.5}
Hr for $H_{jp}^f \quad \forall j \in \mathcal{J}, p \in P_j$	{ $\mathcal{N}(0.2, 0.2), \mathcal{N}(0.4, 0.2)$ }
Hr for $H_{jp}^e \quad \forall j \in \mathcal{J}, p \in P_j$	{ $\mathcal{N}(0.2, 0.2), \mathcal{N}(0.4, 0.2)$ }

4.5.2 Results and Discussions

We analyze the performance of our algorithms on randomly generated instances and consider 360 instance classes with various problem size and parameter level as presented in Table 4.4. We generate and solve four instances per problem class for a total of $360 \times 4 = 1440$ instances. We set the overall solution algorithm time limit to 1,800 seconds and 900 seconds for solving the master problem in each iteration of the CCG algorithm. We allow the master problem to be solved in the given time limit even if the total solution time limit (i.e. 1800 seconds) is exceeded. The algorithm is implemented in C++ programming language equipped with CPLEX 12.9 solver. All instances are run on a computer with an Intel 3.19 GH with 12 logical processors and 16 GB RAM, and are solved optimally within the time limit.

Table 4.5 reports the performance of the solution method on the randomly generated instances. The first column assigns a number to each instance class. The second to fourth columns represent the planning periods, number of PDs, and number of new products for each PD. The fifth to seventh columns report the mean, maximum, and minimum solution time in seconds. Likewise, eighth to eleventh columns report the mean, minimum, and maximum number of iterations for solving instances. Finally, the last columns report the number of instances that are solved within the time limit. Notice that for each class in a row, 48 instances are solved.

The maximum CPU time is observed for the class in row 22 and the maximum number of iterations to obtain the optimal solution in row 5. Classes in row 20, 22, and 30 have the most number of unsolved instances. There are 41 unsolved instances in total which are

removed from analysis. As expected, the mean and maximum solution time and number of iterations are generally non-decreasing as the number of new products grow from 3 to 7 as evident by comparing the even rows that are associated with $P_j = 7$ with odd rows who are associated with $P_j = 3$. This is because increase the number of new products not only intensifies the competition over resources, it also impacts the number of variables and constraints in each PD problem. The growing competition is the result of growing competition over engineering resource. Since the total number of products is unchanged, the growing competition over engineering resource is more severe because it is the only resource that is specifically used to develop new products. Moreover, this factor is crucial for the tractability of instances within the time limit since most of the unsolved instances (39 out of 41) are those with 7 number of new products.

The number of PDs is another factor significantly impacting the performance of the algorithm. The more PDs involved in the lower level, the greater competition form over distributing shared resources. The higher number of PDs also increases the number of total variables and constraints in the master problem, thus the number of unsolved instances significantly increase with the number of PDs. For a fixed number of new products, the growing number of PDs generally increases the mean CPU time and iterations as evident by comparing, for instance, classes in row 12, 14, 16, 18, and 20 where $T = 12$ and $P_j = 7$.

The number of planning periods is another factor that impact the performance of the solution approach. Unlike two other factors, increasing the number of planning periods, decreases the computational difficulty for the algorithm. In one hand increasing the number of periods not only makes PDs to agree on more periods, but also increase the size of instances. On the other hand, a wider planning horizon allows PDs to have more flexibility in developing new products, leading to less over all competition over resources. However, in the instances that we consider the flexibility that is granted by increasing planning periods is more pronounced than the increased competition burden.

We perform analysis to examine the impact of other parameters. We quantify the impact of factory capacity H^f and engineering requirement H^e to develop new products. Our analysis show that when the mean of normal distribution used to generate H^e changes from 0.2 to 0.4, the objective function of the leader increases %0.496. However, the change is almost negligible when the mean of distribution changes for H^f . In other words, the engineering requirement plays a more significant role in the overall cost of the system compared to the factory capacity requirement for new products. Moreover, the increase

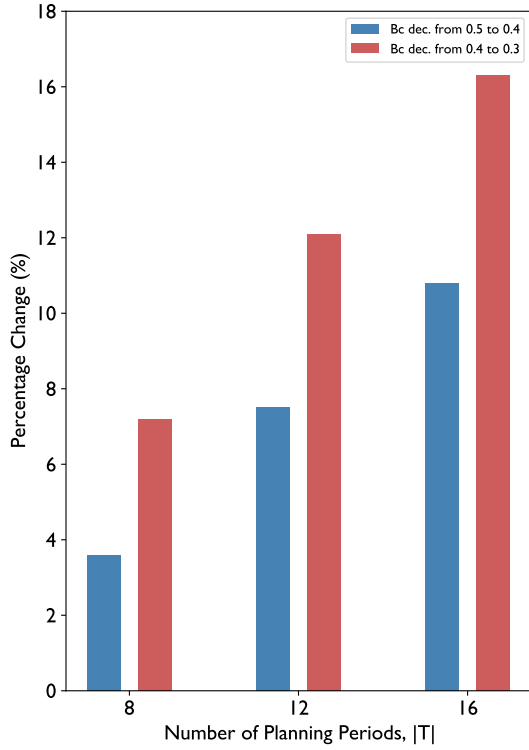
Table 4.5: The performance of the CCG algorithm

row				CPU Time			# Iterations			# Unsolved
	T	J	P_j	mean	min	max	mean	min	max	
1	8	4	3	0	0	2	2	1	5	-
2	8	4	7	1	0	6	2	1	3	-
3	8	8	3	3	0	23	3	1	22	-
4	8	8	7	7	1	18	2	1	4	-
5	8	12	3	55	0	584	13	1	117	-
6	8	12	7	67	3	599	8	1	47	-
7	8	16	3	57	0	565	10	1	74	2
8	8	16	7	176	4	1264	12	1	56	4
9	8	20	3	12	2	161	2	1	13	-
10	8	20	7	133	10	574	5	1	19	-
11	12	4	3	0	0	5	1	1	1	-
12	12	4	7	0	0	1	1	1	1	-
13	12	8	3	0	0	1	1	1	1	-
14	12	8	7	5	1	8	2	1	4	-
15	12	12	3	4	0	27	2	1	8	-
16	12	12	7	20	3	123	3	1	11	-
17	12	16	3	7	1	48	2	1	6	-
18	12	16	7	68	6	436	5	1	19	1
19	12	20	3	10	1	59	1	1	4	-
20	12	20	7	346	16	1506	7	1	21	8
21	16	4	3	9	2	55	1	1	4	-
22	16	4	7	234	12	1836	6	1	23	9
23	16	8	3	0	0	1	1	1	1	-
24	16	8	7	1	0	3	1	1	1	-
25	16	12	3	0	0	1	1	1	2	-
26	16	12	7	10	3	28	2	1	4	-
27	16	16	3	7	1	36	2	1	6	-
28	16	16	7	20	3	54	2	1	5	-
29	16	20	3	9	1	24	1	1	2	-
30	16	20	7	319	17	1216	5	1	13	17

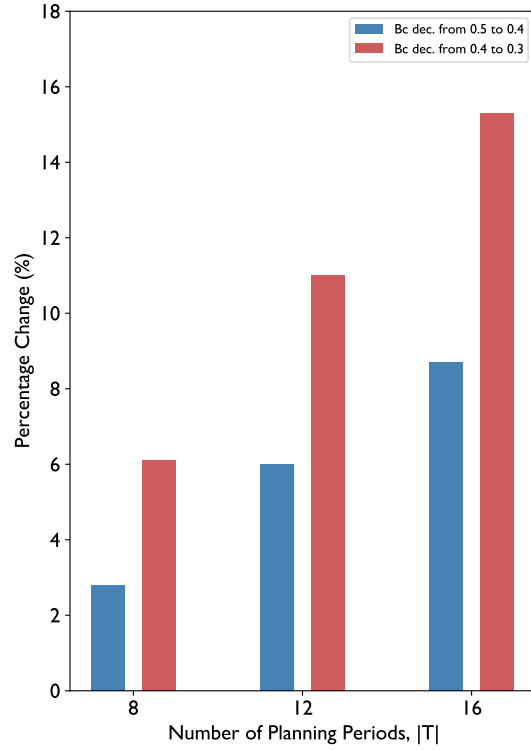
in the CORP's objective value is %0.57 when the mean of normal distribution is 0.4 for both H^f and H^e .

Figure 4.2 illustrates the impact of Bc , the coefficient used to generate total budget. For different time periods, Figure 4.2a depicts the percentage increase in the objective value of the leader when Bc decreases from 0.5 to 0.4 and when it decreases from 0.4 to 0.3. Figure 4.2b demonstrates the same impact on the average individual PD objective values. As evident, the parameter has a more pronounced impact when decreased from 0.4 to 0.3 for both cases. Its impact also increases with the number of planning periods. Although, the parameter's impact is quite similar for both cases, it is slightly more impactful for the total objective value of the leader.

The overall competition which is determined by the number of PDs in the lower-level is another factor that may affect the objective functions of the leader and followers. To quantify the affect of competition we run another set of instances in which the magnitude of each PD's objective function remains the same, i.e. we set $w_j = 1$ for all $j \in \mathcal{J}$ and $\eta = 100$. We first run instances with only one PD in the system and note the entities' costs when there is no competition in the lower level. We then compare the average objective value of PDs for various number of PDs. The percentage of increase in the average objective value of PDs for different problem parameters is illustrated in figure 4.3. As expected, the competition leads to higher cost when $Bs = 0.3$ hence the available budget level is more restrained. The affect of number of planning periods is mixed. When $T = 8$, the increase in the cost is between 25% – 70%, but is drops to –10% – 60% when $T = 12$, and again increases to the highest level at $T = 16$ with the increase between 20% to 125%. This mixed affect can be explained by the fact that the increasing the number of periods has two opposing impacts. In one hand, it gives flexibility to PDs to developed their products in wider range. But on the other hand, PDs should agree on higher number of periods to reach to an equilibrium. The mixed affect can also be attributed to the number of PDs. When $T = 16$, increasing the number of PDs generally leads to higher cost. However, there is a reduction in $J = 12$ when $T = 8$ and $T = 12$.



(a)



(b)

Figure 4.2: The impact of Bc on the total objective value of the leader (a), and average objective value of PDs (b)

T	8	8	8	12	12	12	16	16	16
J \ Bc	0.3	0.4	0.5	0.3	0.4	0.5	0.3	0.4	0.5
2	0.27	0.09	0.09	0.87	0.38	0.37	0.88	0.32	0.31
5	1.14	0.63	0.28	2.00	1.25	0.69	2.12	1.31	0.71
10	1.14	0.71	0.35	2.07	1.44	0.93	2.23	1.55	0.99
15	1.18	0.74	0.38	2.21	1.56	1.06	2.40	1.75	1.17
20	1.32	0.93	0.57	2.31	1.69	1.25	2.61	2.04	1.56

Table 4.6: Effect of competition on the average objective value of PDs

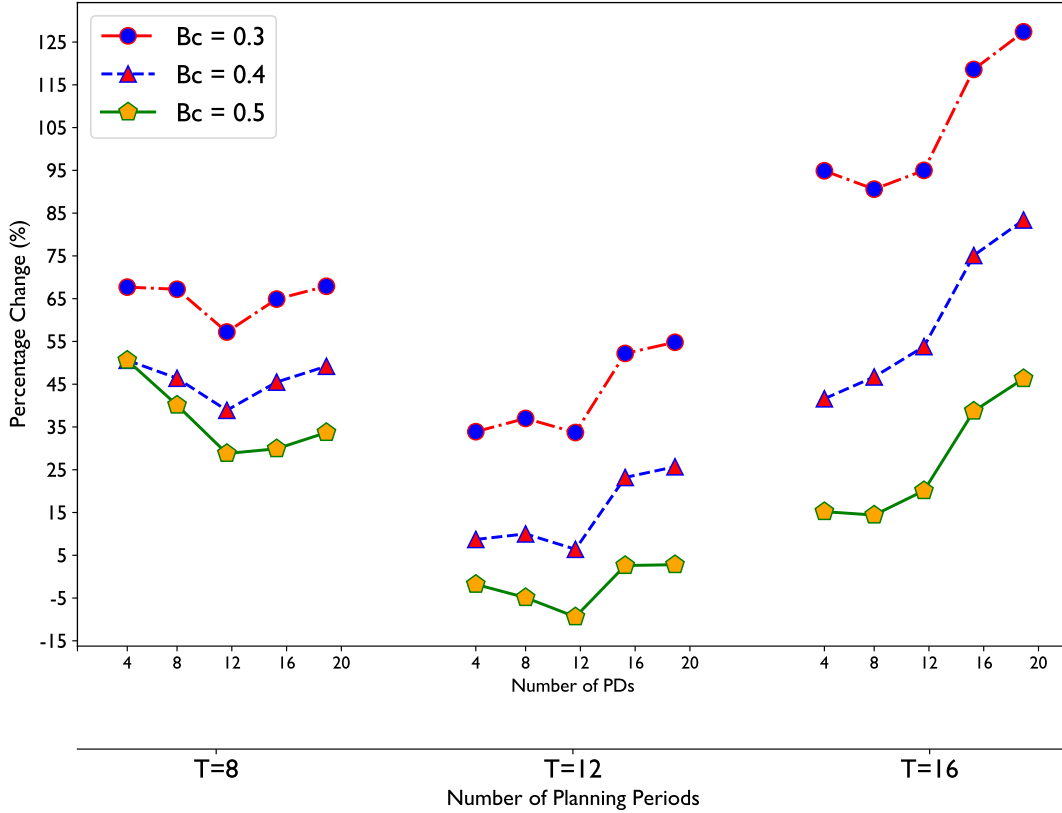


Figure 4.3: Percentage Change in the Average Obj. Value of PDs Under Competition

4.6 Summary

This chapter propose a mixed-integer bilevel model with interdependent followers for the decentralized and hierarchical decision process on which the product transition takes place in a semiconductor manufacturing company. The model coordinates the allocation of budget and distribution of factory and engineering resources among followers. The modeling framework is quite flexible and allows the incorporation of more constraints and distribution of authority. The leader of the model is the corporate management and followers are product divisions who are responsible for manufacturing and development of new products. The leader allocates budget among followers and minimizes the total system as well as total weighted allocated budget. We present a single-level reformulation of the problem and develop a solution algorithm based on cut-and-column paradigm that generates new set of resource allocations for each PD.

Chapter 5

Conclusion

This dissertation considers two well-known problems arising in the context of hierarchical decision making (HDM) processes. HDM is a game-theoretical framework that involves one or more decision makers who solve sequential interdependent problems over the planning horizon. In particular, we consider multistage stochastic programs and bilevel programs. In multistage stochastic programs, decision makers are cooperative, seeking to optimize the same objective function over the planning horizon while considering the expected impact of their future decisions. In bilevel programming, however, decision makers are noncooperative who make decisions in a sequential manner.

In Chapter 2, we present a reformulation for a general multistage stochastic program where the objective and constraints are additive separable functions. The reformulation is carried out on subtrees whose union make up the whole tree. The reformulation allows for the relaxation of certain constraints, hence the scenario tree can be decomposed into independent subtrees. Therefore, it is particularly amenable to multi-threaded environments where subtree problems can be solved in parallel. Moreover, we propose a systematic way to decompose the tree. The method only requires two parameters, one indicating the general way of grouping scenario nodes, and the other generate a decomposition. We test the bounding method on two well known problems from the literature namely multistage capacity acquisition problem (MCAP) and multistage asset management problem (MSAM). Computational analysis shows that the method can obtain bounds for very large instances and outperform the best algorithm in the literature (i.e. scenario decomposition).

In Chapter 3, we formulate a mixed-integer bilevel model with interdependent followers for capacity coordination during product transitions in a semiconductor manufacturing firm. The leader is corporate management and followers are MFG and ENG units who must coordinate production, prototype fabrication and new product development schedules. We propose two single-level reformulations, both of which consist of a master problem which deals with new production plans, and two subproblems to identify optimality or infeasibility. We then propose a solution algorithm based on constraint and column generation that generates a set of new product development plan in each iteration. To speed up the solution algorithm, a lower bounding algorithm and a warm start strategy are developed. The lower bound is obtained from bilevel feasible solutions found by a heuristic. The master problem in each iteration employs a warm start strategy that schedules development of new products in a greedy fashion.

We perform an extensive computational analysis to examine the performance of the proposed solution approach. We also provide managerial insights on the impact of product mix and the value of the CORP's leadership. Intuitively, taking away CORP's control over the information exchange between the MFG and ENG units reduces its objective by allowing the followers to focus on their local objectives.

To the best of our knowledge, this chapter presents the first exact solution algorithm for mixed-integer bilevel programs with interdependent followers. The proposed bilevel model and solution approach are flexible, and allow us to study the impact of different settings for the distribution of decision authority among the decision makers. Therefore, this chapter provides a useful tool to study decentralized, hierarchical decision problems in organizational design. Our computational experiments provide practical insights, but the reported numerical values should be interpreted cautiously because test instances are randomly generated. In practice, some problem parameters may be inherently related to each other. For example, tardiness weight associated with the development of a new product may depend on the demand, backordering cost and unit profit of that product. The decision makers should consider such relations when calibrating the proposed model.

A number of directions for future research emerge from this work. Despite the algorithmic enhancements we have implemented such as the warm start and lower bounding techniques, the computational burden of the solution approach remains substantial, requiring significant improvements to be able to address larger instances. A systematic study of how the performance of the solution method scales with additional followers

and more complex information flows among the followers would provide valuable insights. Exploring the relationship between the number of decision entities, the distribution of decision authority among them, and the number of feasible bilevel solutions is also of great practical interest. It may well be that under certain decision hierarchies there exist very few bilevel feasible solutions, but small adjustments to problem parameters such as capacity levels may make a larger number of such solutions available. Finally, exploring the relationship between iterative combinatorial auction-based approaches to product transitions (Bansal et al. 2020a) and the bilevel approach proposed here is of both theoretical and practical interest.

Chapter 4 generalizes the problem considered in Chapter 3 by allowing product division to handle all activities related to product development and production. We propose a mixed-integer bilevel model with interdependent followers in which the model coordinates the allocation of budget and distribution of factory and engineering resources among followers. Corporate management is the leader of the model who allocates budget among followers and minimizes the total system cost and budget. The followers are product divisions each of them responsible for market analysis, product design, product development including prototype testing, and manufacturing of new and current products. The objective of each product division is to minimize the lost revenue, the cost of inventory, production, and backorder. The proposed modeling framework is flexible as it allows multiple product division, and is capable of incorporating other resources and distribution of authority. We present a single-level reformulation of the problem and develop a solution algorithm based on cut-and-column paradigm that generates new set of resource allocations for each product division in each iteration.

The performance of the algorithm is examined by extensive computational analysis over randomly generated instances that differ in key problem parameters such as the number of planning periods, PDs and new products. We also analyze the impact of various problem parameters on the objective function value of the leader. Moreover, we provide managerial insights by examining the effect of competition on the average individual objective function of PDs and find that, besides the number of PDs, the available budget can significantly impact the system cost of each PD.

This work can be extended in multiple directions. Firstly, to deal with larger instances, the algorithm's performance can be enhanced by various speed-up methods including warm starts. Second, the future study can focus on the parameter calibration and ana-

lyzing the impact of other parameters such as the total allocated budget weight in the leader's objective function. Third, this study is conducted under complete information about the value of parameters which could not be the case in many practical applications. Therefore, considering uncertainty in a way that scales up with the tractability of model would be of great practical interest. Finally, this work can be extended by exploring its relationship with other resource coordination frameworks, such as iterative combinatorial auction (Bansal et al. 2020b) and design of mechanism (Mallik and Harker 2004), proposed to this process.

The mathematical framework proposed for the product transition process in Chapter 3 and 4 can be extended and applied to other domains and problems. Although we develop the bilevel model in the context of production planning and capacity allocation in semiconductor manufacturing firms, many real-world problems can be modeled as a bilevel program with multiple interdependent followers. The transportation application can be found, for example, in noncooperative transportation problem in which multiple competing companies have shared resources (i.e. infrastructure such as roads) and are required to carry a certain amount of goods from suppliers to demand points. The overall goal of such problems is to minimize cost and maximize the societal factors, hence a government organization can be considered as the leader who maximizes the social welfare.

There is a fierce competition in the retail industry to serve more customers with minimum cost. Consider competing retail companies that strive to open several branches in a rapidly developing city. The municipality can act as the leader to ensure equity in the distribution of branches. In other words, municipality tries to prevent any concentration of branches in particular regions which could happen due to various demographical and social factors. Therefore, while followers (retail companies) compete over customers as the shared resource, the leader (municipality) seeks the equitable distribution of branches and oversees the allocation of locations. This problem can arise in other public sector services such as fire department, hospitals, and the police.

Numerous problems of healthcare can be modeled as bilevel program with interdependent followers. Vaccine production, for example, is carried out by multiple companies who, with the advent of new diseases such as Covid 19, initially compete over public funding, scientists, and ingredients and later compete over the market share. The government has to coordinate the allocation of resources while minimizing damages inflicted by the

disease. The pharmaceutical companies, on the other hand, compete over resources to maximize their profit. Similar Applications can also be found in distribution of medical supplies by the state and federal government. Future studies can also focus on extending the framework to other domains.

REFERENCES

- Ahmed, S. (2013). A scenario decomposition algorithm for 0–1 stochastic programs. *Operations Research Letters*, 41(6):565–569.
- Ahmed, S., Cabral, F. G., and da Costa, B. F. P. (2020). Stochastic lipschitz dynamic programming. *Mathematical Programming*, pages 1–39.
- Ahmed, S. and Garcia, R. (2003). Dynamic capacity acquisition and assignment under uncertainty. *Annals of Operations Research*, 124(1-4):267–283.
- Ahmed, S., King, A. J., and Parija, G. (2003). A multi-stage stochastic integer programming approach for capacity expansion under uncertainty. *Journal of Global Optimization*, 26(1):3–24.
- Ahmed, S., Tawarmalani, M., and Sahinidis, N. V. (2004). A finite branch-and-bound algorithm for two-stage stochastic integer programs. *Mathematical Programming*, 100(2):355–377.
- Aksen, D. and Aras, N. (2012). A bilevel fixed charge location model for facilities under imminent attack. *Computers & Operations Research*, 39(7):1364–1381.
- Altman, E. and Wynter, L. (2004a). Equilibrium, games, and pricing in transportation and telecommunication networks. *Networks and Spatial Economics*, 4(1):7–21.
- Altman, E. and Wynter, L. (2004b). Equilibrium, games, and pricing in transportation and telecommunication networks. *Networks and Spatial Economics*, 4(1):7–21.
- Arslan, O., Jabali, O., and Laporte, G. (2018). Exact solution of the evasive flow capturing problem. *Operations Research*, 66(6):1625–1640.
- Audet, C., Savard, G., and Zghal, W. (2007). New branch-and-cut algorithm for bilevel linear programming. *Journal of Optimization Theory and Applications*, 134(2):353–370.
- Aussel, D. and Sagratella, S. (2017). Sufficient conditions to compute any solution of a quasivariational inequality via a variational inequality. *Mathematical Methods of Operations Research*, 85(1):3–18.
- Bakir, I., Boland, N., Dandurand, B., and Erera, A. (2020). Sampling scenario set partition dual bounds for multistage stochastic programs. *INFORMS Journal on Computing*, 32(1):145–163.
- Bansal, A., Uzsoy, R., and Kempf, K. (2020a). Iterative combinatorial auctions for managing product transitions in semiconductor manufacturing. *IIEE Transactions*, 52(4):413–431.

- Bansal, A., Uzsoy, R., and Kempf, K. (2020b). Iterative combinatorial auctions for managing product transitions in semiconductor manufacturing. *IIEE Transactions*, 52(4):413–431.
- Barahona, F., Bermon, S., Günlük, O., and Hood, S. (2005). Robust capacity planning in semiconductor manufacturing. *Naval Research Logistics (NRL)*, 52(5):459–468.
- Barbarosoğlu, G. and Arda, Y. (2004). A two-stage stochastic programming framework for transportation planning in disaster response. *Journal of the operational research society*, 55(1):43–53.
- Bard, J. (1998). *Practical bilevel optimization*. Kluwer Academic Publishers.
- Bard, J. F. and Falk, J. E. (1982). An explicit solution to the multi-level programming problem. *Computers & Operations Research*, 9(1):77–100.
- Bard, J. F. and Moore, J. T. (1990). A branch and bound algorithm for the bilevel programming problem. *SIAM Journal on Scientific and Statistical Computing*, 11(2):281–292.
- Bard, J. F. and Moore, J. T. (1992). An algorithm for the discrete bilevel programming problem. *Naval Research Logistics (NRL)*, 39(3):419–435.
- Baringo, L. and Conejo, A. J. (2012). Transmission and wind power investment. *IEEE transactions on power systems*, 27(2):885–893.
- Basciftci, B., Ahmed, S., and Gebrael, N. (2019). Adaptive two-stage stochastic programming with an application to capacity expansion planning. *arXiv preprint arXiv:1906.03513*.
- Basilico, N., Coniglio, S., Gatti, N., and Marchesi, A. (2020). Bilevel programming methods for computing single-leader-multi-follower equilibria in normal-form and polymatrix games. *EURO Journal on Computational Optimization*, 8(1):3–31.
- Bennett, K. P., Kunapuli, G., Hu, J., and Pang, J.-S. (2008). Bilevel optimization and machine learning. In *IEEE World Congress on Computational Intelligence*, pages 25–47. Springer.
- Beresnev, V. and Melnikov, A. (2018). Exact method for the capacitated competitive facility location problem. *Computers & Operations Research*, 95:73–82.
- Bertsimas, D. and Tsitsiklis, J. N. (1997). *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA.
- Bhaskaran, S. R., Goel, A., and Ramachandran, K. (2011). Managing product transitions under technology uncertainty. *Available at SSRN 1775430*.

- Bílgíner, Ö. and Erhun, F. (2010). Managing product introductions and transitions. *Wiley Encyclopedia of Operations Research and Management Science*.
- Birge, J. R. (1985). Decomposition and partitioning methods for multistage stochastic linear programs. *Operations research*, 33(5):989–1007.
- Birge, J. R. and Louveaux, F. (2011). *Introduction to stochastic programming*. Springer Science & Business Media.
- Boland, N., Christiansen, J., Dandurand, B., Eberhard, A., Linderoth, J., Luedtke, J., and Oliveira, F. (2018). Combining progressive hedging with a frank–wolfe method to compute lagrangian dual bounds in stochastic mixed-integer programming. *SIAM Journal on Optimization*, 28(2):1312–1336.
- Boland, N., Dumitrescu, I., and Froyland, G. (2008). A multistage stochastic programming approach to open pit mine production scheduling with uncertain geology. *Optimization online*, pages 1–33.
- Bolusani, S., Coniglio, S., Ralphs, T. K., Tahernejad, S., Dempe, S., and Zemkoho, A. (2020). A unified framework for multistage and multilevel mixed integer linear optimization. *Bilevel Optimization, Adv. Next Challenges*.
- Bolusani, S. and Ralphs, T. K. (2021). A framework for generalized benders’ decomposition and its application to multilevel optimization. *arXiv preprint arXiv:2104.06496*.
- Bracken, J. and McGill, J. T. (1973). Mathematical programs with optimization problems in the constraints. *Operations Research*, 21(1):37–44.
- Calvete, H. I., Domínguez, C., Galé, C., Labbé, M., and Marin, A. (2019). The rank pricing problem: models and branch-and-cut algorithms. *Computers & operations research*, 105:12–31.
- Calvete, H. I. and Galé, C. (2007). Linear bilevel multi-follower programming with independent followers. *Journal of Global Optimization*, 39(3):409–417.
- Campelo, M., Dantas, S., and Scheimberg, S. (2000). A note on a penalty function approach for solving bilevel linear programs. *Journal of global optimization*, 16(3):245.
- Caramia, M. and Mari, R. (2016). A decomposition approach to solve a bilevel capacitated facility location problem with equity constraints. *Optimization Letters*, 10(5):997–1019.
- Carøe, C. C. and Schultz, R. (1999). Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24(1-2):37–45.
- Carøe, C. C. and Tind, J. (1998). L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming*, 83(1):451–464.

- Carrión, M., Philpott, A. B., Conejo, A. J., and Arroyo, J. M. (2007). A stochastic programming approach to electric energy procurement for large consumers. *IEEE Transactions on Power Systems*, 22(2):744–754.
- Castaing, J., Cohn, A., Denton, B. T., and Weizer, A. (2016). A stochastic programming approach to reduce patient wait times and overtime in an outpatient infusion center. *IIE Transactions on Healthcare Systems Engineering*, 6(3):111–125.
- Chen, Z.-L. and Powell, W. B. (1999). Convergent cutting-plane and partial-sampling algorithm for multistage stochastic linear programs with recourse. *Journal of Optimization Theory and Applications*, 102(3):497–524.
- Colson, B., Marcotte, P., and Savard, G. (2007). An overview of bilevel optimization. *Annals of Operations Research*, 153(1):235–256.
- Coniglio, S., Gatti, N., and Marchesi, A. (2020). Computing a pessimistic stackelberg equilibrium with multiple followers: The mixed-pure case. *Algorithmica*, 82(5):1189–1238.
- Cui, A. S. and Wu, F. (2017). The impact of customer involvement on new product development: Contingent and substitutive effects. *Journal of Product Innovation Management*, 34(1):60–80.
- Dan, T., Lodi, A., and Marcotte, P. (2020). Joint location and pricing within a user-optimized environment. *EURO Journal on Computational Optimization*, 8(1):61–84.
- Daryalal, M., Bodur, M., and Luedtke, J. R. (2020). Lagrangian dual decision rules for multistage stochastic mixed integer programming. *arXiv preprint arXiv:2001.00761*.
- de Matos, V. L. and Finardi, E. C. (2012). A computational study of a stochastic optimization model for long term hydrothermal scheduling. *International Journal of Electrical Power & Energy Systems*, 43(1):1443–1452.
- Dempe, S. (2002). *Foundations of Bilevel Programming*. Kluwer Academic Publishers, Dordrecht.
- Dempe, S. (2018). *Bilevel optimization: theory, algorithms and applications*, volume 3. TU Bergakademie Freiberg, Fakultät für Mathematik und Informatik.
- DeNegre, S. T. and Ralphs, T. K. (2009). A branch-and-cut algorithm for integer bilevel linear programs. In *Operations research and cyber-infrastructure*, pages 65–78. Springer.
- Deng, Y., Jia, H., Ahmed, S., Lee, J., and Shen, S. (2020). Scenario grouping and decomposition algorithms for chance-constrained programs. *INFORMS Journal on Computing*.

- Dewez, S., Labbé, M., Marcotte, P., and Savard, G. (2008). New formulations and valid inequalities for a bilevel pricing problem. *Operations research letters*, 36(2):141–149.
- Dowson, O., Morton, D. P., and Pagnoncelli, B. K. (2020). Partially observable multistage stochastic programming. *Operations Research Letters*, 48(4):505–512.
- Dreves, A., Facchinei, F., Kanzow, C., and Sagratella, S. (2011). On the solution of the kkt conditions of generalized nash equilibrium problems. *SIAM Journal on Optimization*, 21(3):1082–1108.
- Druehl, C. T., Schmidt, G. M., and Souza, G. C. (2009). The optimal pace of product updates. *European Journal of Operational Research*, 192(2):621–633.
- Fabiani, F. and Grammatico, S. (2019). Multi-vehicle automated driving as a generalized mixed-integer potential game. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):1064–1073.
- Facchinei, F., Fischer, A., and Piccialli, V. (2009). Generalized nash equilibrium problems and newton methods. *Mathematical Programming*, 117(1):163–194.
- Facchinei, F. and Kanzow, C. (2010). Generalized nash equilibrium problems. *Annals of Operations Research*, 175(1):177–211.
- Facchinei, F., Kanzow, C., and Sagratella, S. (2014). Solving quasi-variational inequalities via their kkt conditions. *Mathematical Programming*, 144(1):369–412.
- Ferrer, G. and Swaminathan, J. M. (2006). Managing new and remanufactured products. *Management science*, 52(1):15–26.
- Fischetti, M., Ljubić, I., Monaci, M., and Sinnl, M. (2017). A new general-purpose algorithm for mixed-integer bilevel linear programs. *Operations Research*, 65(6):1615–1637.
- Fischetti, M., Ljubić, I., Monaci, M., and Sinnl, M. (2018). On the use of intersection cuts for bilevel optimization. *Mathematical Programming*, 172(1-2):77–103.
- Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., and Pontil, M. (2018). Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, pages 1568–1577. PMLR.
- Gabriel, S. A., Conejo, A. J., Fuller, J. D., Hobbs, B. F., and Ruiz, C. (2012). *Complementarity modeling in energy markets*, volume 180. Springer Science & Business Media.
- Gade, D., Küçükyavuz, S., and Sen, S. (2014). Decomposition algorithms with parametric gomory cuts for two-stage stochastic integer programs. *Mathematical Programming*, 144(1-2):39–64.

- Garcés, L. P., Conejo, A. J., García-Bertrand, R., and Romero, R. (2009). A bilevel approach to transmission expansion planning within a market environment. *IEEE Transactions on Power Systems*, 24(3):1513–1522.
- Garcia-Herreros, P., Zhang, L., Misra, P., Arslan, E., Mehta, S., and Grossmann, I. E. (2016). Mixed-integer bilevel optimization for capacity planning with rational markets. *Computers & Chemical Engineering*, 86:33–47.
- Gokpinar, B., Hopp, W. J., and Iravani, S. M. (2010). The impact of misalignment of organizational structure and product architecture on quality in complex product development. *Management science*, 56(3):468–484.
- Grass, E. and Fischer, K. (2016). Two-stage stochastic programming in disaster management: A literature survey. *Surveys in Operations Research and Management Science*, 21(2):85–100.
- Gul, S. (2018). A stochastic programming approach for appointment scheduling under limited availability of surgery turnover teams. *Service Science*, 10(3):277–288.
- Gul, S., Denton, B. T., and Fowler, J. W. (2012). A multi-stage stochastic integer programming model for surgery planning. *Michigan Engineering*.
- Gumus, Z. H. and Ciric, A. R. (1997). Reactive distillation column design with vapor/liquid/liquid equilibria. *Computers & chemical engineering*, 21:S983–S988.
- Hansen, P., Jaumard, B., and Savard, G. (1992). New branch-and-bound rules for linear bilevel programming. *SIAM Journal on scientific and Statistical Computing*, 13(5):1194–1217.
- Hemmati, M. and Smith, J. C. (2016). A mixed-integer bilevel programming approach for a competitive prioritized set covering problem. *Discrete Optimization*, 20:105–134.
- Higle, J. L. and Sen, S. (1991). Stochastic decomposition: An algorithm for two-stage linear programs with recourse. *Mathematics of operations research*, 16(3):650–669.
- Ho, T.-H., Savin, S., and Terwiesch, C. (2002). Managing demand and sales dynamics in new product diffusion under supply constraint. *Management Science*, 48(2):187–206.
- Hong, M., Wai, H.-T., Wang, Z., and Yang, Z. (2020). A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic. *arXiv preprint arXiv:2007.05170*.
- Huang, K. and Ahmed, S. (2009). The value of multistage stochastic programming in capacity planning under uncertainty. *Operations Research*, 57(4):893–904.
- Infanger, G. (2006). Dynamic asset allocation strategies using a stochastic dynamic programming approach. In Zenios, S. A. and Ziemba, W. T., editors, *Handbook of Asset and Liability Management*, pages 200–251. Springer, Berlin.

- Infanger, G. (2008). Dynamic asset allocation strategies using a stochastic dynamic programming approach. In *Handbook of asset and liability management*, pages 199–251. Elsevier.
- Kacar, N. B., Mönch, L., and Uzsoy, R. (2016). Modeling cycle times in production planning models for wafer fabrication. *IEEE Transactions on Semiconductor Manufacturing*, 29(2):153–167.
- Kall, P., Wallace, S. W., and Kall, P. (1994). *Stochastic programming*. Springer.
- Karabuk, S. and Wu, S. D. (2002). Decentralizing semiconductor capacity planning via internal market coordination. *IIE transactions*, 34(9):743–759.
- Karabuk, S. and Wu, S. D. (2003). Coordinating strategic capacity planning in the semiconductor industry. *Operations Research*, 51(6):839–849.
- Karabuk, S. and Wu, S. D. (2005). Incentive schemes for semiconductor capacity allocation: A game theoretic analysis. *Production and Operations Management*, 14(2):175–188.
- Kırbış, E. Y., Büyüktaktakın, İ. E., Haight, R. G., Akhundov, N., Knight, K., and Flower, C. E. (2021). A multistage stochastic programming approach to the optimal surveillance and control of the emerald ash borer in cities. *INFORMS Journal on Computing*, 33(2):808–834.
- Klastorin, T. and Tsai, W. (2004). New product introduction: Timing, design, and pricing. *Manufacturing & Service Operations Management*, 6(4):302–320.
- Kleinert, T., Labbé, M., Ljubić, I., and Schmidt, M. (2021). A survey on mixed-integer programming techniques in bilevel optimization. *preprint: <https://hal.inria.fr/hal-03095139>*.
- Kleywegt, A. J., Shapiro, A., and Homem-de Mello, T. (2002). The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502.
- Koca, E., Souza, G. C., and Druehl, C. T. (2010). Managing product rollovers. *Decision Sciences*, 41(2):403–423.
- Köppe, M., Queyranne, M., and Ryan, C. T. (2010). Parametric integer programming algorithm for bilevel mixed integer programs. *Journal of optimization theory and applications*, 146(1):137–150.
- Krishnan, V. and Ulrich, K. T. (2001). Product development decisions: A review of the literature. *Management science*, 47(1):1–21.
- Labbé, M., Marcotte, P., and Savard, G. (1998). A bilevel model of taxation and its application to optimal highway pricing. *Management science*, 44(12-part-1):1608–1622.

- Laporte, G. and Louveaux, F. V. (1993). The integer l-shaped method for stochastic integer programs with complete recourse. *Operations research letters*, 13(3):133–142.
- Lavigne, D., Loulou, R., and Savard, G. (2000). Pure competition, regulated and stackelberg equilibria: Application to the energy system of quebec. *European Journal of Operational Research*, 125(1):1–17.
- Le Cadre, H., Jacquot, P., Wan, C., and Alasseur, C. (2020). Peer-to-peer electricity market analysis: From variational to generalized nash equilibrium. *European Journal of Operational Research*, 282(2):753–771.
- León, H. C. M. and Farris, J. A. (2011). Lean product development research: Current state and future directions. *Engineering Management Journal*, 23(1):29–51.
- Li, H., Graves, S. C., and Huh, W. T. (2013). Optimal capacity conversion for product transitions under high service requirements. *Manufacturing & Service Operations Management*, 16(1):46–60.
- Li, H., Graves, S. C., and Huh, W. T. (2014). Optimal capacity conversion for product transitions under high service requirements. *Manufacturing & Service Operations Management*, 16(1):46–60.
- Li, H., Graves, S. C., and Rosenfield, D. B. (2010). Optimal planning quantities for product transition. *Production and Operations Management*, 19(2):142–155.
- Li, X., Armagan, E., Tomasgard, A., and Barton, P. I. (2011). Stochastic pooling problem for natural gas production network design and operation under uncertainty. *AIChE Journal*, 57(8):2120–2135.
- Liang, C., Çakanyıldırım, M., and Sethi, S. P. (2014). Analysis of product rollover strategies in the presence of strategic customers. *Management Science*, 60(4):1033–1056.
- Liao, S. and Seifert, R. W. (2015). On the optimal frequency of multiple generation product introductions. *European Journal of Operational Research*, 245(3):805–814.
- Lim, W. S. and Tang, C. S. (2006). Optimal product rollover strategies. *European Journal of Operational Research*, 174:905–922.
- Linderoth, J. and Wright, S. J. (2005). Computational grids for stochastic programming. In *Applications of stochastic programming*, pages 61–77. SIAM.
- Lobel, I., Patel, J., Vulcano, G., and Zhang, J. (2015). Optimizing product launches in the presence of strategic consumers. *Management Science*, 62(6):1778–1799.
- Lozano, L. and Smith, J. C. (2017). A value-function-based exact approach for the bilevel mixed-integer programming problem. *Operations Research*, 65(3):768–786.

- Lukač, Z., Šorić, K., and Rosenzweig, V. V. (2008). Production planning problem with sequence dependent setups as a bilevel programming problem. *European Journal of Operational Research*, 187(3):1504–1512.
- MacKay, M., Vicol, P., Lorraine, J., Duvenaud, D., and Grosse, R. (2019). Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. *arXiv preprint arXiv:1903.03088*.
- Maggioni, F., Allevi, E., and Bertocchi, M. (2016). Monotonic bounds in multi-stage mixed-integer stochastic programming. *Computational Management Science*, 13(3):423–457.
- Mallik, S. (2007). Contracting over multiple parameters: Capacity allocation in semiconductor manufacturing. *European Journal of Operational Research*, 182(1):174–193.
- Mallik, S. and Harker, P. T. (2004). Coordinating supply chains with competition: Capacity allocation in semiconductor manufacturing. *European Journal of Operational Research*, 159(2):330–347.
- McCormick, G. P. (1976). Computability of global solutions to factorable nonconvex programs: Part i—convex underestimating problems. *Mathematical programming*, 10(1):147–175.
- Mönch, L., Uzsoy, R., and Fowler, J. W. (2018). A survey of semiconductor supply chain models part iii: master planning, production planning, and demand fulfilment. *International Journal of Production Research*, 56(13):4565–4584.
- Motto, A. L., Arroyo, J. M., and Galiana, F. D. (2005). A mixed-integer lp procedure for the analysis of electric grid security under disruptive threat. *IEEE Transactions on Power Systems*, 20(3):1357–1365.
- Mulvey, J. M. and Shetty, B. (2004). Financial planning via multi-stage stochastic optimization. *Computers and Operations Research*, 31(1):1–20.
- Mulvey, J. M. and Vladimirov, H. (1991). Applying the progressive hedging algorithm to stochastic generalized networks. *Annals of Operations Research*, 31(1):399–424.
- Nishi, T., Hiranaka, Y., and Grossmann, I. E. (2011). A bilevel decomposition algorithm for simultaneous production scheduling and conflict-free routing for automated guided vehicles. *Computers & Operations Research*, 38(5):876–888.
- Nürnberg, R. and Römisch, W. (2002). A two-stage planning model for power scheduling in a hydro-thermal system under uncertainty. *Optimization and Engineering*, 3(4):355–378.
- Oggioni, G., Smeers, Y., Allevi, E., and Schaible, S. (2012). A generalized nash equilibrium model of market coupling in the european power system. *Networks and Spatial Economics*, 12(4):503–560.

- Özaltın, O. Y., Prokopyev, O. A., and Schaefer, A. J. (2018). Optimal design of the seasonal influenza vaccine with manufacturing autonomy. *INFORMS Journal on Computing*, 30(2):371–387.
- Özaltın, O. Y., Prokopyev, O. A., Schaefer, A. J., and Roberts, M. S. (2011). Optimizing the societal benefits of the annual influenza vaccine: A stochastic programming approach. *Operations research*, 59(5):1131–1143.
- Pereira, M. V. and Pinto, L. M. (1991). Multi-stage stochastic optimization applied to energy planning. *Mathematical programming*, 52(1):359–375.
- Powell, W. B. and Topaloglu, H. (2003). Stochastic programming in transportation and logistics. *Handbooks in operations research and management science*, 10:555–635.
- Rash, E. and Kempf, K. (2012). Product line design and scheduling at intel. *Interfaces*, 42(5):425–436.
- Rebennack, S. (2016). Combining sampling-based and scenario-based nested benders decomposition methods: application to stochastic dual dynamic programming. *Mathematical Programming*, 156(1-2):343–389.
- Robbins, M. J. and Lunday, B. J. (2016). A bilevel formulation of the pediatric vaccine pricing problem. *European Journal of Operational Research*, 248(2):634–645.
- Ryan, K., Ahmed, S., Dey, S. S., Rajan, D., Musselman, A., and Watson, J.-P. (2020). Optimization-driven scenario grouping. *INFORMS Journal on Computing*, 32(3):805–821.
- Ryu, J.-H., Dua, V., and Pistikopoulos, E. N. (2004). A bilevel programming framework for enterprise-wide process networks under uncertainty. *Computers & Chemical Engineering*, 28(6-7):1121–1129.
- Sagan, B. E. (2020). *Combinatorics: The art of counting*, volume 210. American Mathematical Soc.
- Sagratella, S. (2017). Algorithms for generalized potential games with mixed-integer variables. *Computational Optimization and Applications*, 68(3):689–717.
- Sagratella, S. (2019). On generalized nash equilibrium problems with linear coupling constraints and mixed-integer variables. *Optimization*, 68(1):197–226.
- Sagratella, S., Schmidt, M., and Sudermann-Merx, N. (2020). The noncooperative fixed charge transportation problem. *European Journal of Operational Research*, 284(1):373–382.
- Saharidis, G. K. and Ierapetritou, M. G. (2009). Resolution method for mixed integer bi-level linear problems based on decomposition technique. *Journal of Global Optimization*, 44(1):29–51.

- Sahinidis, N. V. (2004). Optimization under uncertainty: state-of-the-art and opportunities. *Computers & Chemical Engineering*, 28(6-7):971–983.
- Sandıkçı, B., Kong, N., and Schaefer, A. J. (2013). A hierarchy of bounds for stochastic mixed-integer programs. *Mathematical Programming*, 138(1-2):253–272.
- Sandıkçı, B. and Özaltın, O. Y. (2017). A scalable bounding method for multistage stochastic programs. *SIAM Journal on Optimization*, 27(3):1772–1800.
- Santoso, T., Ahmed, S., Goetschalckx, M., and Shapiro, A. (2005). A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research*, 167(1):96–115.
- Schultz, R., Stougie, L., and Van Der Vlerk, M. H. (1996). Two-stage stochastic integer programming: a survey. *Statistica Neerlandica*, 50(3):404–416.
- Sen, S. (2005). Algorithms for stochastic mixed-integer programming models. *Handbooks in operations research and management science*, 12:515–558.
- Sen, S. and Sherali, H. D. (2006). Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, 106(2):203–223.
- Sen, S., Yu, L., and Genc, T. (2006). A stochastic programming approach to power portfolio optimization. *Operations Research*, 54(1):55–72.
- Sen, S. and Zhou, Z. (2014). Multistage stochastic decomposition: a bridge between stochastic programming and approximate dynamic programming. *SIAM Journal on Optimization*, 24(1):127–153.
- Shapiro, A. (2006). On complexity of multistage stochastic programs. *Operations Research Letters*, 34(1):1–8.
- Shapiro, A. and Ahmed, S. (2004). On a class of minimax stochastic programs. *SIAM Journal on Optimization*, 14(4):1237–1249.
- Shapiro, A., Dentcheva, D., and Ruszczyński, A. (2014). *Lectures on stochastic programming: modeling and theory*. SIAM.
- Shapiro, A. and Ding, L. (2020). Periodical multistage stochastic programs. *SIAM Journal on Optimization*, 30(3):2083–2102.
- Shelar, D., Amin, S., and Hiskens, I. (2021). Evaluating resilience of electricity distribution networks via a modification of generalized benders decomposition method. *IEEE Transactions on Control of Network Systems*.

- Sherali, H. D. and Zhu, X. (2006). On solving discrete two-stage stochastic programs having mixed-integer first-and second-stage variables. *Mathematical programming*, 108(2):597–616.
- Shi, C., Zhou, H., Lu, J., Zhang, G., and Zhang, Z. (2007). The kth-best approach for linear bilevel multifollower programming with partial shared variables among followers. *Applied Mathematics and Computation*, 188(2):1686–1698.
- Shi, J. and Oren, S. S. (2016). Wind power integration through stochastic unit commitment with topology control recourse. In *2016 Power systems computation conference (PSCC)*, pages 1–7. IEEE.
- Stein, O. and Sudermann-Merx, N. (2018). The noncooperative transportation problem and linear generalized nash games. *European Journal of Operational Research*, 266(2):543–553.
- Tavashoğlu, O., Prokopyev, O. A., and Schaefer, A. J. (2019). Solving stochastic and bilevel mixed-integer programs via a generalized value function. *Operations Research*, 67(6):1659–1677.
- Ulrich, K. T. and Eppinger, S. D. (2016). *Product Design and Development*. McGraw-Hill, New York, NY.
- Vicente, L., Savard, G., and Judice, J. (1996). Discrete linear bilevel programming problem. *Journal of optimization theory and applications*, 89(3):597–614.
- Wallace, S. W. and Ziemba, W. T. (2005). *Applications of stochastic programming*. SIAM.
- Wang, L. and Xu, P. (2017). The watermelon algorithm for the bilevel integer linear programming problem. *SIAM Journal on Optimization*, 27(3):1403–1430.
- Wang, Z., Liu, F., Ma, Z., Chen, Y., Jia, M., Wei, W., and Wu, Q. (2021). Distributed generalized nash equilibrium seeking for energy sharing games in prosumers. *IEEE Transactions on Power Systems*.
- Wu, L., De Matta, R., and Lowe, T. J. (2009). Updating a modular product: How to set time to market and component quality. *IEEE Transactions on Engineering Management*, 56(2):298–311.
- Wu, S. D., Erkoç, M., and Karabuk, S. (2005). Managing capacity in the high-tech industry: A review of literature. *The engineering economist*, 50(2):125–158.
- Xu, P. and Wang, L. (2014). An exact algorithm for the bilevel mixed integer linear programming problem under three simplifying assumptions. *Computers & operations research*, 41:309–318.

- Yue, D., Gao, J., Zeng, B., and You, F. (2019). A projection-based reformulation and decomposition algorithm for global optimization of a class of mixed integer bilevel linear programs. *Journal of Global Optimization*, 73(1):27–57.
- Yue, D. and You, F. (2017). Stackelberg-game-based modeling and optimization for supply chain design and operations: A mixed integer bilevel programming framework. *Computers & Chemical Engineering*, 102:81–95.
- Zenarosa, G. L., Prokopyev, O. A., and Schaefer, A. J. (2014). Scenario-tree decomposition: bounds for multistage stochastic mixed-integer programs. *Optimization Online*.
- Zeng, B. and An, Y. (2014). Solving bilevel mixed integer program by reformulations and decomposition. *Optimization online*, pages 1–34.
- Zhan, Y. and Zheng, Q. P. (2018). A multistage decision-dependent stochastic bilevel programming approach for power generation investment expansion planning. *IIEE Transactions*, 50(8):720–734.
- Zhou, J., Lam, W. H., and Heydecker, B. G. (2005). The generalized nash equilibrium model for oligopolistic transit market with elastic demand. *Transportation Research Part B: Methodological*, 39(6):519–544.
- Zou, J., Ahmed, S., and Sun, X. A. (2018). Multistage stochastic unit commitment using stochastic dual dynamic integer programming. *IEEE Transactions on Power Systems*, 34(3):1814–1823.
- Zou, J., Ahmed, S., and Sun, X. A. (2019). Stochastic dual dynamic integer programming. *Mathematical Programming*, 175(1):461–502.