The Dissertation Committee for Ali Koc
certifies that this is the approved version of the following dissertation:

Prioritization via Stochastic Optimization

Committee:

David P. Morton, Supervisor

Elmira Popova, Supervisor

Jonathan F. Bard

Constantine Caramanis

Stephen Hess

Erhan Kutanoglu

Prioritization via Stochastic Optimization

by

Ali Koc, B.Sc., M.Sc.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2010

In the memory of my mother...

Acknowledgments

I would like to express my deepest gratitude to my advisors Dr. David Morton and Dr. Elmira Popova, whose encouragement, guidance, and support from the very early stage to the final level made this dissertation possible. Their encouragement and supervision inspired and enriched my growth as a student, a researcher, and a scientist. I am indebted to them more than they know. It is also a pleasure to convey my gratitude to my committee members Dr. Bard, Dr. Caramanis, Dr. Hess, and Dr. Kutanoglu for their guidance during my graduate studies. I gratefully acknowledge the support by Dr. Hess of EPRI for my dissertation. I also would like to thank Dr. Ihsan Sabuncuoglu of Bilkent University for encouraging me to pursue an academic career.

I take this opportunity to thank my wonderful friends in Austin. I am heartily thankful to Burak for supporting me throughout my graduate life. I also would like to thank Fehmi, Ferhat, Emrah Tanriverdi, Emrah Zarifoglu, Murat, Kursad, Bulent, Firat, Adem, Serdal, Gokhan, Tayfun, Tara and numerous others. I thank everyone who has been a part of my life in Austin. I also would like to make a special reference to JP's and Mozart's, where I spent a significant portion of my time in Austin working on my dissertation.

Last, but the most important, I would like to thank all the people in my family and in my life for their endless support, without which it would be impossible to come up with this work.

Prioritization via Stochastic Optimization

Publication No. _____

Ali Koc, Ph.D.
The University of Texas at Austin, 2010

Supervisors: David P. Morton
Elmira Popova

We take a novel perspective on real-life decision making problems involving binary activity-selection decisions that compete for scarce resources. The current literature in operations research approaches these problems by forming an optimal portfolio of activities that meets the specified resource constraints. However, often practitioners in industry and government do not take the optimal-portfolio approach. Instead, they form a rank-ordered list of activities and select those that have the highest priority.

The academic literature tends to discredit such ranking schemes because they ignore dependencies among the activities. Practitioners, on the other hand, sometimes discredit the optimal-portfolio approach because if the problem parameters change, the set of activities that was once optimal no longer remains optimal. Even worse, the new optimal set of activities may exclude some of the previously optimal activities, which they may have already selected. Our approach takes both viewpoints into account. We rank activities

considering both the uncertainty in the problem parameters and the optimal portfolio that will be obtained once the uncertainty is revealed.

We use stochastic integer programming as a modeling framework. We develop several mathematical formulations and discuss their relative merits, comparing them theoretically and computationally. We also develop cutting planes for these formulations to improve computation times. To be able to handle larger real-life problem instances, we develop parallel branch-and-price algorithms for a capital budgeting application. Specifically, we construct a column-based reformulation, develop two branching strategies and a tabu-search-based primal heuristic, propose two parallelization schemes, and compare these schemes on parallel computing environments using commercial and open-source software.

We give applications of prioritization in facility location and capital budgeting problems. In the latter application, we rank maintenance and capital-improvement projects at the South Texas Project Nuclear Operating Company, a two-unit nuclear power plant in Wadsworth, Texas. We compare our approach with several *ad hoc* ranking schemes similar to those used in practice.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Resource-constrained activity selection problems (RCASPs) involve binary activity-selection decisions that compete for scarce resources. Much of the current literature in operations research approaches RCASPs by forming an optimal portfolio of activities that meets the resource constraints. However, practitioners in industry and government often avoid the optimal-portfolio approach, instead forming a rank-ordered list of activities and funding those that have the highest priority. We propose a novel prioritization approach that takes both viewpoints into account.

Capital budgeting practice at South Texas Project Nuclear Operating Company (STPNOC) [43] illustrates an approach typical in industry. As the operator of a large commercial nuclear power-generation station, STPNOC annually develops a rank-ordered list that specifies the highest priority project, the second highest priority project, and so forth. The current budget and project-cost forecasts yield what STPNOC calls the "blue line." Projects above the blue line are to be funded and those below it are not. Over the course of the year, the blue line can shift if the available budget changes, cost overruns materialize, etc.

In government, practitioners often approach such problems similarly. The following quote from Brown et al. [5] characterizes how the military would approach a problem of selecting projects to harden critical infrastructure against terrorist attack, and how the operations research community would view that approach:

> First, it would assume that our infrastructure will be attacked and would take steps to protect it, i.e., harden the infrastructure... The budget for this purpose will always be limited, but often not prespecified. The military typically draws up a prioritized list of "defended assets" in need of protection, along with a list of potential protective measures, and presents these to policy makers. The latter parties make the final decisions after balancing costs, effectiveness, and intangibles, and after determining the budget... However, a prioritized list of defended assets has a serious flaw for our applications. Such a list creates a "preferred set" of $n + 1$ assets by adding one asset to the preferred set of size n. But, we know that an optimal set of size n and an optimal set of size $n + 1$ may have nothing in common.

In discussing typical approaches used in industry to select projects in the context of capital budgeting, Savage et al. [39] say the following:

> It is common when choosing a portfolio of capital investment projects to rank them from best to worst, then start at the top of the list and go down until the budget has been exhausted. This flies in the face

of modern portfolio theory, which is based on the interdependence of investments.

We agree with Brown et al. [5] and Savage et al. [39] that simplistic ranking schemes that form a priority list by individually scoring each candidate activity can be inferior. For example, in a capital budgeting problem, ranking candidate projects based on their profit or benefit-investment ratio ignores structural and stochastic dependencies that may exist among the projects. On the other hand, forming an optimal portfolio assuming problem parameters are known may yield a portfolio that is fragile with respect to changes in these parameters. That is, if the problem's parameters—the resource availability, the activities' resource consumption, or the activities' contribution to the objective function—change, the set of activities that was once optimal may no longer remain optimal.

We take both viewpoints into account. Our approach prioritizes the activities recognizing structural and stochastic dependencies among them and recognizing that the activities ultimately implemented, after the stochastic data are realized, will act as a portfolio. Prioritization is of interest when some problem parameters are random and we must commit to a ranking of the activities before these parameters are realized. Prioritization involves optimally placing activities into a priority list before the uncertainty is revealed, and after realizing the uncertainty, making an optimal prioritized activity selection observing resource constraints.

Figure 1.1 gives an illustrative example of our approach to prioritizing using the k-median problem [e.g., 30]. Customers are at the corners of the grids, and facilities may be located at the centers, amounting to 64 customers and 49 potential facility locations. The objective is to choose k locations that minimize the sum of Euclidean distances that the customers must travel in order to reach their closest facility. Figure 1.1(a) gives optimal solutions for deterministic values of $k = 1, 2, 3$, and 4.

Figures 1.1(b)–(d) concern a version in which k is stochastic, and we incrementally install facilities until we learn that k has been exhausted. Suppose k takes values 1 or 2 with equal probability. Imagine that we first solve this problem for $k = 1$ and greedily locate a facility at the center grid, as indicated by the "O" in Figure 1.1(a). Under the realization of $k = 2$, we find ourselves in an awkward position given that we cannot relocate the initial facility. Figure 1.1(b) shows an optimal solution to the problem where there is 50% chance we will locate an additional facility, after locating this initial facility. Hence, we locate the first facility at the location indicated by the first "X" in Figure 1.1(b). If $k = 1$ is realized, we are close to the optimal solution of the 1-median problem. If $k = 2$ we install the facility at the second "X", obtaining a solution close to that of the 2-median problem. Figures 1.1(c) and (d) illustrate the same idea when k is equally likely to be $1, 2, 3$ and $1, 2, 3, 4$, respectively.

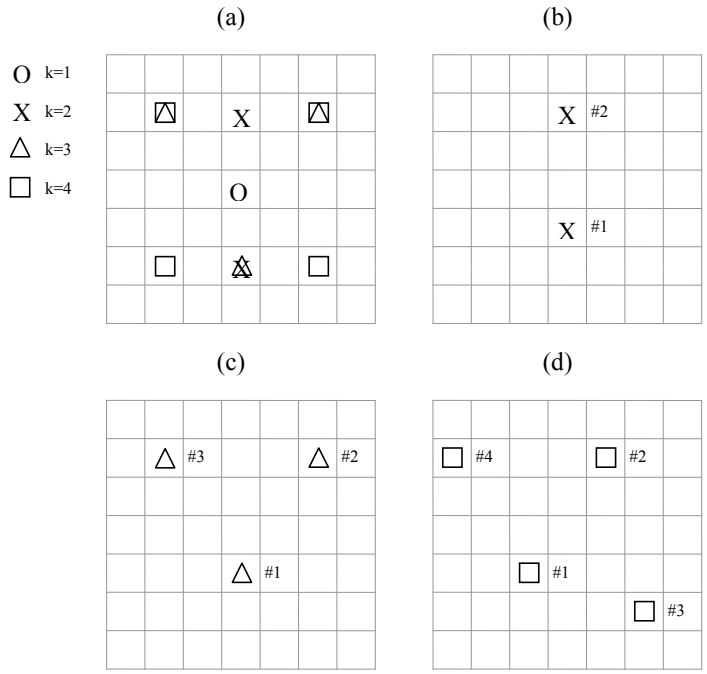

Figure 1.1: Part (a) shows optimal solutions to four k-median problems with k = 1-4. In parts (b)-(d), k is uncertain and we prioritize. In part (b), we have a 50% chance of having either one or two facilities. In parts (c) and (d), we are equally likely to have k = 1-3 and k = 1-4, respectively. Of course, symmetry allows multiple optimal solutions. For example, in part (a) the vertical locations of the $k = 2$ solution could instead be the symmetric horizontal locations.

1.2 Literature Survey

In spite of its common use in practice, prioritization has received little attention in the academic literature. Mettu and Plaxton [29] and Plaxton [33] develop constant factor approximation algorithms for a variant of the k-median problem that we sketched above. Instead of assuming a known probability

distribution that governs k, the goal is to form a priority list that minimizes the competitive ratio, i.e., the worst-case ratio, over all k, of the cost of the priority list's solution to the optimal cost when k is known. See Lin et al. [25] for further work that includes the k-median, k-minimum spanning tree, k-vertex cover, k-set cover and hierarchical clustering problems, again from the perspective of developing constant-factor approximation algorithms.

Dean et al. [10] consider a prioritized knapsack problem in which the items' values and the knapsack size are deterministic but the items' sizes are independent random variables. An item's size is realized only when it is to be placed in the knapsack, and Dean et al. seek a priority list that maximizes the expected value of items successfully inserted. They develop constant-factor approximation algorithms, and further study the benefit of adaptivity, i.e., the benefit from being able to revise the remaining priority list based on the residual knapsack capacity. Dean et al. [9] analyze the benefit of adaptivity in a more general class of stochastic packing problems.

Hochbaum [16] considers a knapsack-constrained version of the so-called selection problem of Balinski [1] and Rhys [36]. Constructing the entire efficient frontier that trades off benefit and cost for this problem is NP-hard. However, Hochbaum shows that the concave envelope of the efficient frontier can be formed in strongly polynomial time. And, from the perspective of prioritization, she shows that solutions at kink points of that concave envelope are nested, as the budget grows. See Nehme and Morton [31] for related work.

1.3 Outline

The dissertation is organized as follows: In Chapter 2, we formalize our prioritization approach and give several mathematical formulations. We compare these formulations theoretically and computationally on a set of multidimensional knapsack problem instances. To improve computational performance of the formulations, we develop two sets of cutting planes and test them on the same problem instances.

In Chapter 3, we develop two parallel branch-and-price algorithms for an application of prioritization to the multidimensional knapsack problem. Specifically, we construct a column-based reformulation, develop two branching strategies, and propose a tabu-search-based primal heuristic. We also propose two parallelization schemes for our branch-and-price algorithm, and compare these schemes on parallel computing environments using commercial and open-source software.

In Chapter 4, we give a real-life application of prioritization in a capital budgeting problem faced by STPNOC. We rank STPNOC's nuclear-maintenance and capital-improvement projects, considering uncertainties in their profits and cost flows and in the annual budgets. We compare our approach with several heuristic ranking schemes. Finally, we summarize our contribution in Chapter 5, and discuss further research directions.

Chapter 2

Prioritization: Mathematical Modeling

Although we allude to the notion of prioritization in Chapter 1, we did not give a formal definition. In this chapter, we formalize the approach, illustrate it by furthering the discussion surrounding Figure 1.1, give several mathematical formulations, and compare the formulations theoretically and computationally. To improve computational performance of the formulations, we develop two sets of cutting planes and show their computational use on a set of problem instances.

2.1 Activity Prioritization

We begin with the statement of the RCASP and proceed with its stochastic version and our prioritization approach, what we call "the prioritized RCASP." We then give an integer programming (IP) formulation for the prioritized RCASP. Consider the following notation and the associated IP formulation:

Indices and sets:

$i \in I$	activities

Data:

$\mathbf{A}$	matrix of resource-consumption coefficients
$\mathbf{b}$	vector of resources
$\mathbf{c}_x = (c_{xi})_{i \in I}$	cost coefficients for binary activity-selection variables
$\mathbf{c}_y$	cost coefficients for remaining variables
C	constraint set that links the activity-selection decisions with the remaining decisions

Decision variables:

$\mathbf{x} = (x_i)_{i \in I}$	1 if activity i is selected; 0 otherwise
$\mathbf{y}$	remaining decision variables

Formulation:

$$\min_{\mathbf{x},\mathbf{y}} \quad \mathbf{c}_x\mathbf{x} + \mathbf{c}_y\mathbf{y} \tag{2.1a}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b}, \tag{2.1b}$$

$$\mathbf{x} \in \{0,1\}^{|I|}, \tag{2.1c}$$

$$(\mathbf{x},\mathbf{y}) \in C. \tag{2.1d}$$

The goal is to minimize the objective function in (2.1a), consisting of the costs associated with decisions $\mathbf{x}$ and $\mathbf{y}$. Constraint (2.1b) models resource constraints, where we assume all entries of the resource-consumption matrix $\mathbf{A}$ and the resource vector $\mathbf{b}$ are nonnegative. Constraint (2.1c) restricts $\mathbf{x}$ to be a binary vector. The set C is an arbitrary constraint linking decisions

$\mathbf{x}$ and $\mathbf{y}$. The problem data as given in the RCASP model (2.1) is deterministic. Its optimal solution, in terms of the $\mathbf{x}$ variables, is a portfolio of activities. For reasons we have motivated in the introduction and motivate further in this chapter, our prioritization approach is of interest when the data $(\mathbf{A}, \mathbf{b}, \mathbf{c}_x, \mathbf{c}_y, C)$ are random and we must commit to a ranking of the binary activity-selection decisions before the data realizations are known.

Let $(\mathbf{A}^\omega, \mathbf{b}^\omega, \mathbf{c}_x^\omega, \mathbf{c}_y^\omega, C^\omega),\ \omega \in \Omega$, denote the data realizations with the probability mass function q^ω, where we assume $|\Omega|$ is finite. A *priority list* is a many-to-one assignment of activities to priority levels such that each priority level is assigned at least one activity. *Prioritized activity selection* has two requirements: Under any scenario $\omega \in \Omega$, a lower-priority activity cannot be selected unless all higher-priority activities are selected, and either all or none of the activities on the same priority level are selected. In view of these definitions, the prioritized RCASP optimally places activities into a priority list before the uncertainty is revealed, and, after realizing the uncertainty, makes an optimal prioritized activity selection observing the RCASP's constraints.

In model (2.1), we have formulated the deterministic RCASP to minimize the cost of the portfolio of activities we select. In formulating the prioritized RCASP, we form a priority list and that priority list determines the portfolio of activities selected under each scenario $\omega \in \Omega$. In the prioritized RCASP, the goal is to minimize the expected value of the cost we incur, i.e., the cost we incur under each scenario $\omega \in \Omega$ is weighted by its probability mass, q^ω. The model we formulate is a two-stage stochastic integer program.

The first stage decision forms the priority list and the second stage decision uses that list to form a portfolio of activities, and determines other decision variables, under each scenario $\omega \in \Omega$.

We now formalize this verbal description of the prioritized RCASP. Let $\mathcal{L}_l \subseteq I,\ l = 1, \ldots, L$, denote the priority levels, where $\mathcal{L}_l \neq \emptyset$, $l = 1, \ldots, L$, $\bigcup_{l=1}^{L} \mathcal{L}_l = I$, and $\mathcal{L}_l \cap \mathcal{L}_{l'} = \emptyset,\ l \neq l',\ l, l' = 1, \ldots, L$. A priority list is denoted as $\mathcal{L} = [\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_L]$, where activities in $\mathcal{L}_1$ have higher priority than those in $\mathcal{L}_2$, activities in $\mathcal{L}_2$ have higher priority than those in $\mathcal{L}_3$, and so forth. Let $F^{\omega}(\mathcal{L})$ denote the optimal value of the RCASP under scenario $\omega \in \Omega$, subject to the prioritized activity–selection restrictions imposed by $\mathcal{L}$. That is,

$$
\begin{aligned}
F^{\omega}(\mathcal{L}) = \quad & \min_{\mathbf{x}^{\omega}, \mathbf{y}^{\omega}} \quad \mathbf{c}_x^{\omega}\mathbf{x}^{\omega} + \mathbf{c}_y^{\omega}\mathbf{y}^{\omega} && \text{(2.2a)}\\
& \text{s.t.} \quad \mathbf{A}^{\omega}\mathbf{x}^{\omega} \leq \mathbf{b}^{\omega}, && \text{(2.2b)}\\
& \qquad \mathbf{x}^{\omega} \in \{0,1\}^{|I|}, && \text{(2.2c)}\\
& \qquad (\mathbf{x}^{\omega}, \mathbf{y}^{\omega}) \in C^{\omega}, && \text{(2.2d)}\\
& \qquad x_i^{\omega} \geq x_{i'}^{\omega}, \quad i \in \mathcal{L}_l,\ i' \in \mathcal{L}_{l'},\ \ l < l',\ l, l' = 1, \ldots, L, && \text{(2.2e)}\\
& \qquad x_i^{\omega} = x_{i'}^{\omega}, \quad i, i' \in \mathcal{L}_l,\ l = 1, \ldots, L. && \text{(2.2f)}
\end{aligned}
$$

The objective function in (2.2a) and constraints (2.2b)–(2.2d) are the same as those in (2.1a)–(2.1d), except they now depend on $\omega \in \Omega$. The last two constraints enforce the two requirements of the prioritized activity selection. Constraint (2.2e) requires that under any scenario, a lower-priority activity cannot

be selected unless all higher-priority activities are selected; constraint (2.2f) requires that either all or none of the activities on the same priority level are selected. Based on the function $F^{\omega}(\mathcal{L})$ defined by model (2.2), we define the prioritized RCASP as,

$$\min_{\mathcal{L},\, L} \quad \left[F(\mathcal{L}) \equiv \sum_{\omega \in \Omega} q^{\omega} F^{\omega}(\mathcal{L}) \right] \tag{2.3a}$$

$$\text{s.t.} \quad \mathcal{L} = [\mathcal{L}_1, \ldots, \mathcal{L}_L], \tag{2.3b}$$

$$\mathcal{L}_l \subseteq I, \qquad l = 1, \ldots, L, \tag{2.3c}$$

$$\mathcal{L}_l \neq \emptyset, \qquad l = 1, \ldots, L, \tag{2.3d}$$

$$\mathcal{L}_l \cap \mathcal{L}_{l'} = \emptyset, \qquad l \neq l',\ l, l' = 1, \ldots, L, \tag{2.3e}$$

$$\bigcup_{l=1,\ldots,L} \mathcal{L}_l = I. \tag{2.3f}$$

Model (2.3) minimizes the expected cost of prioritized activity selection, i.e., the sum of weighted costs of RCASP across scenarios $\omega \in \Omega$. The solution to model (2.3) is an optimal priority list, and the corresponding solution to model (2.2) is the prioritized selection of activities under scenario $\omega \in \Omega$. The timing of decisions and observations of uncertainty is key to understanding the prioritization model (2.3). First, the priority list is formed via $\mathcal{L}$. Next, the values of the problem parameters, $(\mathbf{A}^{\omega}, \mathbf{b}^{\omega}, \mathbf{c}_x^{\omega}, \mathbf{c}_y^{\omega}, C^{\omega}),\ \omega \in \Omega$, are realized. We then effectively work down the priority list selecting the activities via $\mathbf{x}^{\omega}$, until a point where either the budget is exhausted or the cost no longer decreases by selecting more activities. These dynamics are illustrated

in Figure 2.1.

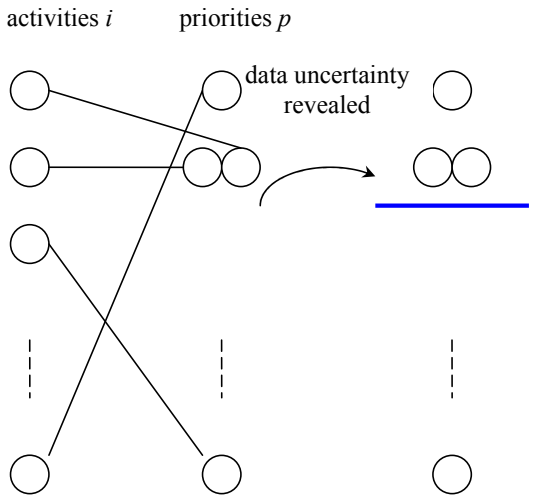

Figure 2.1: Illustration of prioritization.

The minimization in model (2.3) is taken over both the priority list, $\mathcal{L}$, and the number of priority levels, L. In what follows, we use $\mathcal{L}$, but occasionally omit L, in referring to a priority list. Also, it is convenient to have the set of activities, I, to be an ordered set so that it makes sense to write restrictions on it, such as $i < i' \in I$. With an eye towards computation, we next give an IP formulation for the prioritized RCASP of model (2.3). Consider the additional and extended notation over model (2.1), and the associated IP formulation:

Indices and sets:

$i,\, i' \in I$	activities
$\omega \in \Omega$	scenarios

Data:

$(\mathbf{A}^\omega, \mathbf{b}^\omega, \mathbf{c}_x^\omega, \mathbf{c}_y^\omega, C^\omega)$	the problem parameters under scenario ω
q^ω	probability of scenario ω

Decision variables:

$s_{ii'}$	1 if activity i has no lower priority than i'; 0 otherwise
$\mathbf{x}^\omega$, $\mathbf{y}^\omega$	the decision vectors under scenario ω

Formulation:

$$\min_{\mathbf{s},\mathbf{x},\mathbf{y}} \quad \sum_{\omega\in\Omega} q^\omega(\mathbf{c}_x^\omega \mathbf{x}^\omega + \mathbf{c}_y^\omega \mathbf{y}^\omega) \tag{2.4a}$$

$$\text{s.t.} \quad s_{ii'} + s_{i'i} \geq 1, \qquad i < i',\ \ i, i' \in I, \tag{2.4b}$$

$$s_{ii'} \in \{0,1\}, \qquad i \neq i', i, i' \in I, \tag{2.4c}$$

$$x_i^\omega \geq x_{i'}^\omega + s_{ii'} - 1, \qquad i \neq i', i, i' \in I, \omega \in \Omega, \tag{2.4d}$$

$$\mathbf{A}^\omega \mathbf{x}^\omega \leq \mathbf{b}^\omega, \qquad \omega \in \Omega, \tag{2.4e}$$

$$\mathbf{x}^\omega \in \{0,1\}^{|I|}, \qquad \omega \in \Omega, \tag{2.4f}$$

$$(\mathbf{x}^\omega, \mathbf{y}^\omega) \in C^\omega, \qquad \omega \in \Omega. \tag{2.4g}$$

Model (2.4) is a two-stage stochastic integer program. Its first stage variables $\mathbf{s}$ form the priority list, its second stage variables $\mathbf{x}^\omega$ select the portfolio of activities to implement under each scenario $\omega \in \Omega$, and its second stage variables $\mathbf{y}^\omega$ handle the remaining decisions.

The objective function in (2.4a) captures expected cost, formed as the weighted sum over all scenarios. Constraints (2.4b) and (2.4c) formulate the priority list defined by (2.3b)–(2.3f). Given a pair of activities i, $i' \in I$, constraints (2.4b) and (2.4c) make sure that at least one of the activities has no lower priority than the other. In other words, either they have the same priority, i.e., $s_{ii'} = s_{i'i} = 1$, or one has higher priority than the other, e.g., $s_{ii'} = 1$ and $s_{i'i} = 0$. Constraint (2.4d) formulates the two requirements of the prioritized activity selection, i.e., constraints (2.2e) and (2.2f) of model (2.2). Given a pair of activities i, $i' \in I$, if i has higher priority than i', we have $s_{ii'} = 1$ and $s_{i'i} = 0$. Constraint (2.4d) for pair (i, i') then reads $x_i^\omega \geq x_{i'}^\omega$ and $x_{i'}^\omega \geq x_i^\omega - 1$. Since the latter one is redundant, this amounts to constraint (2.2e). If i and i' have the same priority, we have $s_{ii'} = s_{i'i} = 1$. Constraint (2.4d) then reads $x_i^\omega \geq x_{i'}^\omega$ and $x_{i'}^\omega \geq x_i^\omega$. This amounts to $x_i^\omega = x_{i'}^\omega$, same as constraint (2.2f). The last three sets of constraints replicate (2.2b), (2.2c) and (2.2d).

2.2 An Application to Facility Location

We describe facility prioritization briefly in the discussion surrounding Figure 1.1. In this section we elaborate and present some variations. Recalling Figure 1.1, we have a square area consisting of n_G grids on each side. Customers are located at the corners of the grids, and facilities may be located at the centers, which amounts to a total of n_G^2 locations and $(n_G + 1)(n_G + 1)$ customers. There is no limit on the number of customers a facility can handle, nor is there a cost for opening a facility. The only restriction is the budget,

k, i.e., the total number of facilities that may be located. The objective is to locate k facilities such that the sum of Euclidean distances of customers to their closest facility is minimized. Consider the following notation and the corresponding formulation.

Indices and sets:

$i \in I$ candidate facility locations

$j \in J$ customers

Data:

d_{ij} Euclidean distance from customer j to facility i

k budget in terms of total number of facilities

Decision variables:

x_i 1 if location i is selected; 0 otherwise

y_{ij} 1 if customer j is assigned to facility at location i; 0 otherwise

Formulation:

$$\min_{\mathbf{x},\mathbf{y}} \quad \sum_{j\in J}\sum_{i\in I} d_{ij}y_{ij} \tag{2.5a}$$

$$\text{s.t.} \quad \sum_{i\in I} x_i \leq k, \tag{2.5b}$$

$$x_i \geq y_{ij}, \qquad i \in I,\ j \in J, \tag{2.5c}$$

$$\sum_{i\in I} y_{ij} = 1, \qquad j \in J, \tag{2.5d}$$

$$x_i \in \{0,1\}, \qquad i \in I, \tag{2.5e}$$

$$y_{ij} \in [0,1], \qquad i \in I,\ j \in J. \tag{2.5f}$$

Minimizing the objective function in (2.5a) amounts to minimizing the total Euclidean distance that customers must travel. Constraint (2.5b) makes sure we locate no more than k facilities. Constraint (2.5c) allows assigning customers only to the facilities that are opened. Constraint (2.5d) makes sure each customer is assigned to a facility. The last two sets of constraints are binary requirements. We need not force $\mathbf{y}$ to be an integer variable, it is automatically so. Optimal solutions of this problem for $n_G = 7$ and $k = 1, 2, 3, 4$ are given in Figure 1.1(a). To form the priority lists of Figures 1.1(b)–(d), we apply the generic prioritization model (2.4) to this k-median problem (2.5). Consider the following additional and extended notation and the corresponding formulation:

Indices and sets:

$i, i' \in I$	candidate facility locations
$j \in J$	customers
$\omega \in \Omega$	scenarios

Data:

d_{ij}	Euclidean distance from customer j to facility i
k^ω	budget in terms of total number of facilities under scenario ω
q^ω	probability of scenario ω

Decision variables:

$s_{ii'}$	1 if location i has no lower priority than i'; 0 otherwise
x_i^ω	1 if location i is selected under scenario ω; 0 otherwise
y_{ij}^ω	1 if customer j is assigned to facility at location i under scenario ω; 0 otherwise

Formulation:

$$\min_{\mathbf{s},\mathbf{x},\mathbf{y}} \quad \sum_{\omega\in\Omega} q^{\omega} \sum_{j\in J} \sum_{i\in I} d_{ij} y_{ij}^{\omega} \tag{2.6a}$$

$$\text{s.t.} \quad s_{ii'} + s_{i'i} \geq 1, \qquad i < i',\ i, i' \in I, \tag{2.6b}$$

$$s_{ii'} \in \{0,1\}, \qquad i \neq i', i, i' \in I, \tag{2.6c}$$

$$x_i^{\omega} \geq x_{i'}^{\omega} + s_{ii'} - 1, \qquad i \neq i', i, i' \in I, \omega \in \Omega, \tag{2.6d}$$

$$\sum_{i\in I} x_i^{\omega} \leq k^{\omega}, \qquad \omega \in \Omega, \tag{2.6e}$$

$$x_i^{\omega} \geq y_{ij}^{\omega}, \qquad i \in I,\ j \in J,\ \omega \in \Omega, \tag{2.6f}$$

$$\sum_{i\in I} y_{ij}^{\omega} = 1, \qquad j \in J,\ \omega \in \Omega, \tag{2.6g}$$

$$x_i^{\omega} \in \{0,1\}, \qquad i \in I,\ \omega \in \Omega, \tag{2.6h}$$

$$y_{ij}^{\omega} \in [0,1], \qquad i \in I,\ j \in J,\ \omega \in \Omega. \tag{2.6i}$$

Model (2.6) applies the generic prioritization model (2.4) to the k-median problem (2.5). The notation, variables, and constraints read similarly, and hence we do not repeat that discussion here. Optimal solutions for $n_G = 7$ and $k = 1, 2$, $k = 1, 2, 3$, and $k = 1, 2, 3, 4$ with equal probabilities are given in Figures 1.1(b)–(d). Corresponding optimal values are 178.99, 162.79, and 148.55.

One approach to build a heuristic priority list is to use the following greedy heuristic. Suppose the budget, k, has realizations 1 and 2 with equal probability. We may think of this as first realizing a budget to open one facility, and later realizing, with 50% chance, additional budget to open another

facility. As soon as we receive a budget of one, we commit to open a facility at a specific location. The optimal solution of the 1-median problem gives the location marked by 1 in Figure 2.2(a). If we receive additional budget, we solve the 2-median problem fixing one of the facilities to the previously selected location, i.e., the location marked by 1 in Figure 2.2(a). The optimal solution gives the location marked by 2 as the location of the second facility. If, instead, $k = 1, 2, 3$ with equal probability, the heuristic solution gives the same first two locations since it does not make use of the respective probabilities of each budget scenario. The third location is marked by 3. By the same argument, Figure 2.2(a) gives the greedy heuristic's solution when the budget is probabilistic and takes values $k = 1, \ldots, h$, $h \leq 9$, with any nonzero probabilities. The objective function value of any such solution is the weighted sum of total Euclidean distances under each budget scenario. If the budget takes equally likely values of 1–8, the weight for each scenario is 0.125. This gives a solution value of $0.125\,(194.78+167.4+140.01+115.1+90.19+85.62+81.04+76.46) = 118.82$ for the solution in Figure 2.2(a). If we instead use model (2.6), we obtain a solution with optimal value 115.47.

Due to symmetry, the greedy heuristic gives alternative solutions: There are alternative locations for the second facility including the ones marked by 2 in Figures 2.2(a) and (b). This symmetry holds when locating the third facility as well. Figures 2.2(a) and (b) give two of these solutions for $k = 1$–8. For the objective of minimizing the expected total distance, these alternative solutions happen to have the same objective function value. If we, instead,

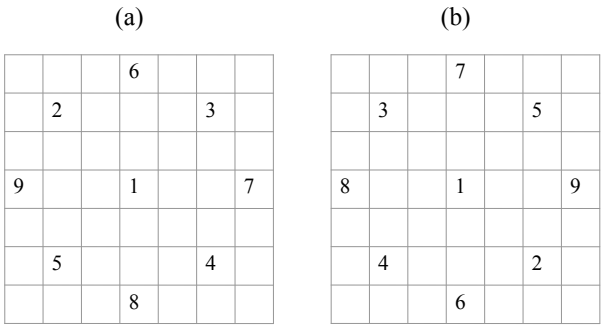

Figure 2.2: Greedy solutions for the stochastic k-median problem. Parts (a) and (b) show two alternative solutions. Each solution is obtained by first finding an optimal solution to the 1-median problem; then to the 2-median problem with one of the locations fixed to the solution obtained from the 1-median problem; then to the 3-median problem with two of the locations fixed to the solution obtained from the 2-median problem; and so forth.

consider the problem of minimizing the expected maximum distance, we see that some alternative solutions have far better objective function values than others. For reference, model (2.7) gives the deterministic problem that minimizes the maximum cost.

$$\min_{\mathbf{x},\mathbf{y}} \quad \max_{j \in J} \sum_{i \in I} d_{ij} y_{ij} \tag{2.7a}$$

$$\text{s.t.} \quad (2.5b),\ (2.5c),\ (2.5d),\ (2.5e),\ (2.5f). \tag{2.7b}$$

Figures 2.3(a)–(c) give three different alternative greedy solutions for the stochastic version of model (2.7). The first location is the same across all alternative solutions. When it comes to select the second location, the greedy heuristic does not differentiate among any of the locations marked by

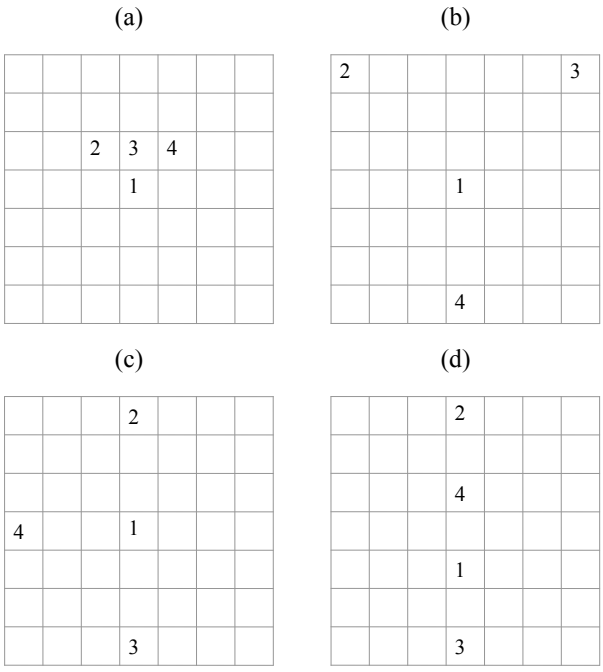

Figure 2.3: Parts (a)–(c) show three alternative greedy solutions for model (2.7) where k takes equally-likely values of 1-4. Part (d) gives the corresponding optimal prioritization solution.

2 in Figures 2.3(a)–(c). In fact, any location in the entire square region is an optimal solution to model (2.7) with $k = 2$ and with the first facility fixed to the location marked by 1. Suppose the heuristic selects the one marked by 2 in Figure 2.3(a). We proceed with solving model (2.7) sequentially for $k = 3$ and then $k = 4$ with the first two and then three facilities fixed to the previously selected locations. We can obtain the locations marked by 3 and 4 in Figure 2.3(a), for again any location is optimal. Alternatively, the heuristic can select the locations marked by 2, 3, and 4 in Figures 2.3(b) and

Table 2.1: Optimal and greedy solutions for prioritized versions of models (2.5) and (2.7). Each cell contains two values. The upper value is the optimal value to model (2.5) or (2.7). The lower one is the optimality gap of a corresponding greedy solution, in percentage terms. We cannot solve the optimal prioritization problem for the instances corresponding to the empty cells; so we leave them empty.

	n_G	h						
		2	3	4	5	6	7	8
model (2.5)	3	22.38	20.70	18.79	17.30	16.30	15.59	15.05
		0%	0.47%	2.27%	1.97%	1.74%	1.56%	1.41%
	5	75.88	68.85	63.08	58.35	54.76	51.73	49.12
		0%	1.58%	3.24%	4.13%	4.28%	3.32%	2.81%
	7	178.99	162.79	148.55	137.15	128.34	121.28	115.47
		1.17%	2.83%	3.88%	3.17%	2.99%	2.97%	2.90%
	9	348.03	317.99	290.66	268.10	250.68	236.85	225.26
		1.76%	2.86%	3.83%	3.37%	3.31%	3.25%	2.34%
model (2.7)	3	2.12	1.94	1.85	1.80	1.65	1.52	1.41
		0%	9.28%	7.29%	6.01%	12.20%	19.63%	26.13%
	5	3.54	3.21	3.04	2.94	2.80	2.63	2.50
		0%	10.25%	16.20%	13.40%	14.45%	18.47%	21.86%
	7	4.95	4.57	4.34	4.18	3.98	3.71	3.45
		0%	8.33%	7.56%	8.64%	3.01%	7.63%	11.59%
	9	6.36	5.82	5.55	5.35	5.03		
		0%	0%	0%	0.81%	4.35%		

(c). The objective function values of prioritization models for these alternative solutions are 4.95, 4.66, and 4.38, respectively. It is no surprise that the unintelligent solution of Figure 2.3(a) has the worst objective function value, followed by better solutions of Figures 2.3(b) and (c). Figure 2.3(d) gives the corresponding optimal solution with the optimal value of 4.18. The optimality gaps of the alternative solutions are 18.42%, 11.48%, and 4.78%.

Table 2.1 summarizes how this greedy heuristic performs. We consider prioritization of both models (2.5) and (2.7), which is indicated by the first column. The table's second column indicates the problem size in terms of the number of grids (n_G). For each problem size, we consider seven different instances, represented by columns 3-9. In each instance, all budget scenarios are equally likely, with the h^{th} instance having $k = 1, 2, \ldots, h$. For each instance Table 2.1 lists two values: The upper value gives the optimal value of the corresponding prioritization model. The lower value gives the percentage optimality gap of a solution from the greedy heuristic. As we indicate above, the greedy heuristic gives alternative solutions with different solution values. Table 2.1 selects one such solution randomly.

One prominent pattern in Table 2.1 is the different performance of the greedy heuristic for priority models (2.5) and (2.7). When minimizing the expected total distance, the heuristic produces optimality gaps of less then 5% on these instances. But it is not as successful in minimizing the expected maximum distance. The primary reason for this seems to be the greedy nature of the heuristic, coupled with the extreme nature of the maximum-distance objective. After centrally locating the first facility, the objective function cannot be reduced when placing a single second facility at any of the $n_G^2 - 1$ potential facility locations. The optimality gap grows as n_G grows because the suboptimality of the greedy heuristic can be amplified as n_G grows.

2.3 Prioritization with Total Order Restriction

The concept of a priority list gives the impression that there is a one-to-one matching between priorities and activities with each activity receiving a unique ranking. The formulation given by constraints (2.3b)–(2.3f), however, allows many-to-one assignment of activities to priorities. In this section, we give an equivalent definition and a corresponding IP formulation for the prioritized RCASP that requires each activity receive a unique ranking. We also compare the two formulations both theoretically and computationally.

Given a priority list, $\mathcal{L}$, we define its *refinement*, $\bar{\mathcal{L}}$, as a priority list that satisfies the following two conditions:

- Each priority level in $\bar{\mathcal{L}}$ is a subset of some priority level in $\mathcal{L}$; and,
- For any two activities i, $i' \in I$ with $i \in \mathcal{L}_l$, $i' \in \mathcal{L}_{l'}$ and $l < l'$, $\bar{\mathcal{L}}$ has the property that $i \in \bar{\mathcal{L}}_{\bar{l}}$, $i' \in \bar{\mathcal{L}}_{\bar{l}'}$ with $\bar{l} < \bar{l}'$.

As an example, $[\{1,2\},\{3\}]$, $[\{1\},\{2\},\{3\}]$, $[\{2\},\{1\},\{3\}]$ are all refinements of $[\{1,2\},\{3\}]$, whereas $[\{3\},\{2,1\}]$ and $[\{3\},\{2\},\{1\}]$ are not since the latter two do not satisfy the second condition for activity pairs (1,3) and (2,3). The following proposition considers the relative objective function values of a priority list and its refinement.

Proposition 2.3.1. *Let $\mathcal{L}$ and $\bar{\mathcal{L}}$ be two feasible priority lists for the prioritized RCASP (2.3), and let $\bar{\mathcal{L}}$ be a refinement of $\mathcal{L}$. Then, $F(\bar{\mathcal{L}}) \leq F(\mathcal{L})$.*

Proof. It is sufficient to show $F^{\omega}(\bar{\mathcal{L}}) \leq F^{\omega}(\mathcal{L}),\ \omega \in \Omega$. Consider the definition of $F^{\omega}(\cdot)$ as the optimal value of model (2.2). For a pair of activities i, i', we have either constraint (2.2e) when they are not on the same priority level, or constraint (2.2f) when they are so. Thus, given a pair of activities i, $i' \in I$, we consider two cases:

(i) They are on the same priority level of $\bar{\mathcal{L}}$. This implies the activities being on the same priority level of $\mathcal{L}$ as well, due to the second condition in the definition of refinement. For these activity pairs we have constraint (2.2f) in the definitions of both $F^{\omega}(\mathcal{L})$ and $F^{\omega}(\bar{\mathcal{L}})$.

(ii) They are on different priority levels of $\bar{\mathcal{L}}$, and assume without loss of generality, i is on a higher priority level than i'. This means we have constraint (2.2e) in the definition of $F^{\omega}(\bar{\mathcal{L}})$. Due to the second condition, we cannot have i on a lower priority level of $\mathcal{L}$ than i'. Hence, either i is on a higher priority level of $\mathcal{L}$ than i', or they are on the same priority level. In the former case, for this pair of activities we have constraint (2.2e) in the definition of $F^{\omega}(\mathcal{L})$; in the latter case, we have constraint (2.2f). That is, in the former, we have the same constraint as in the definition of $F^{\omega}(\bar{\mathcal{L}})$; in the latter, we have a more restrictive constraint since we replace the greater-than type constraint with the equal-to type. Considering the two possibilities sketched above, $F^{\omega}(\mathcal{L})$ is at least as large as $F^{\omega}(\bar{\mathcal{L}})$, for model (2.2) under $\bar{\mathcal{L}}$ is a relaxation of model (2.2) under $\mathcal{L}$. □

A "total order" [37, chap 6] on a set of activities, I, is a permutation

of the elements of I. So, a total order is a priority list, $\mathcal{L} = [\mathcal{L}_1, \ldots, \mathcal{L}_L]$, with $|\mathcal{L}_l| = 1$, $l = 1, \ldots, L$. Thus, a total order can be viewed as the "most-refined" priority list. That is, if $\mathcal{L}$ is a total order, and $\bar{\mathcal{L}}$ is a refinement of $\mathcal{L}$, then $\mathcal{L} = \bar{\mathcal{L}}$. This observation, together with Proposition 2.3.1, implies that the prioritized RCASP with a total order restriction has the same optimal value as that without this restriction. Consider the next model, the following corollary, and the corresponding IP formulation:

$$\min_{\mathcal{L}, L} \quad \left[F(\mathcal{L}) \equiv \sum_{\omega \in \Omega} q^\omega F^\omega(\mathcal{L}) \right] \tag{2.8a}$$

$$\text{s.t.} \quad (2.3b),\ (2.3c),\ (2.3d),\ (2.3e),\ (2.3f), \tag{2.8b}$$

$$|\mathcal{L}_l| = 1, \qquad l = 1, \ldots, L. \tag{2.8c}$$

Corollary 2.3.2. *Models (2.3) and (2.8) have the same optimal value.*

Indices and sets:

$i,\ i' \in I$	activities
$p \in P = \{1, \ldots, \lvert I \rvert\}$	priorities
$\omega \in \Omega$	scenarios

Data:

$(\mathbf{A}^\omega, \mathbf{b}^\omega, \mathbf{c}_x^\omega, \mathbf{c}_y^\omega, C^\omega)$	the problem parameters under scenario ω
q^ω	probability of scenario ω

Decision variables:

z_{ip}	1 if activity i is assigned to priority level p; 0 otherwise
$s_{ii'}$	1 if activity i has higher priority than i'; 0 otherwise
$\mathbf{x}^\omega$, $\mathbf{y}^\omega$	the decision vectors under scenario ω

Formulation:

$$\min_{\mathbf{s},\mathbf{z},\mathbf{x},\mathbf{y}} \quad \sum_{\omega\in\Omega} q^\omega(\mathbf{c}_x^\omega \mathbf{x}^\omega + \mathbf{c}_y^\omega \mathbf{y}^\omega) \tag{2.9a}$$

$$\text{s.t.} \quad \sum_{p\in P} z_{ip} = 1, \qquad i \in I, \tag{2.9b}$$

$$\sum_{i\in I} z_{ip} = 1, \qquad p \in P, \tag{2.9c}$$

$$z_{ip} \in \{0,1\}, \qquad i \in I,\ p \in P, \tag{2.9d}$$

$$|P| s_{ii'} \geq \sum_{p\in P} (|P| - p)(z_{ip} - z_{i'p}), \qquad i \neq i',\ i,\ i' \in I, \tag{2.9e}$$

$$s_{ii'} + s_{i'i} = 1, \qquad i < i',\ \ i, i' \in I, \tag{2.9f}$$

$$(2.4c),\ (2.4d),\ (2.4e),\ (2.4f),\ (2.4g). \tag{2.9g}$$

It is useful to compare model (2.9) with model (2.4). We have additional $\mathbf{z}$ variables in model (2.9) and a modification in the definition of $\mathbf{s}$ variables. Since model (2.9) requires activities be fully-ordered, $s_{ii'}$ is 1 if activity i has 'higher' priority than i', whereas in model (2.4) this variable indicates whether activity i has 'no lower' priority than i'. Hence, constraint (2.4b) is of greater-than type, as opposed to constraint (2.9f), which is of equal-to type. The objective function in (2.9a) is the same as that of model (2.4). Constraints (2.9b), (2.9c), (2.9d) and (2.9e) are new. The first three of these

form a one-to-one matching between activities and priorities. The last one coupled with (2.4c) and (2.9f) defines the $\mathbf{s}$ variables based on this matching.

Proposition 2.3.1 allows us to iteratively refine the optimal priority list of model (2.3), without loss of optimality, until we obtain a total order. Similarly, given an optimal total order, $\mathcal{L}^*$, and a corresponding solution vector, $(\mathbf{x}^*, \mathbf{y}^*)$ with $\mathbf{x}^* = (x_i^{*\omega})_{i \in I, \omega \in \Omega}$, to model (2.8), we can aggregate any two of its priority levels, l, l', if there exist two activities, i, i', such that $i \in \mathcal{L}_l^*$, $i' \in \mathcal{L}_{l'}^*$ and $x_i^{*\omega} = x_{i'}^{*\omega}$ for all $\omega \in \Omega$. This can be done without loss of optimality since the aggregated priority list, together with $(\mathbf{x}^*, \mathbf{y}^*)$, still satisfies constraints (2.2e) and (2.2f) and has the same objective function value. Thus, a solution to either model (2.3) or model (2.8) can be transformed to a solution to the other model with the same objective function value.

The above observation allows us to use either of the two IP formulations, i.e., model (2.4) or model (2.9), in solving the prioritized RCASP. A natural question is then "which one is more efficient?" The first feature to compare is their sizes in terms of the number of constraints and variables. Model (2.9) has obviously $2|I| + |I|(|I| - 1)$ additional structural constraints and $|I|^2$ additional binary variables. Typically, an IP formulation with more constraints and variables is tolerated, even preferred, if it has a tighter linear programming (LP) relaxation. Hence, the next feature to compare is their LP relaxations. To compare the LP relaxations of two different formulations, we usually compare the feasible region over which the linear optimization is performed. This comparison does not lead to sufficiently strong results in our case.

Thus, we follow a different path. We first give two propositions characterizing the LP relaxations of models (2.4) and (2.9), and then give a corollary comparing their optimal values. The proofs of the propositions are almost identical, and so we state the second one without proof. Propositions 2.3.3 and 2.3.4 and Corollary 2.3.5 speak of the LP relaxation of models (2.4) and (2.9). In the former case this amounts to continuous relaxations of constraints (2.4c) and (2.4f) and in the latter, continuous relaxations of (2.4c), (2.4f) and (2.9d), i.e., we assume constraints (2.4g) hold as stated.

Proposition 2.3.3. *If a solution vector* $(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{s}}, \bar{\mathbf{z}})$ *with* $\bar{\mathbf{x}} = (\bar{x}_i^\omega)_{i\in I,\omega\in\Omega}$*, is feasible to the LP relaxation of model (2.9), then*

$$\alpha_{ii'} + \alpha_{i'i} \leq 1, \quad i \neq i',\ i, i' \in I, \tag{2.10}$$

holds, where $\alpha_{ii'} = \max_{\omega\in\Omega}\{\bar{x}_i^\omega - \bar{x}_{i'}^\omega\}$*. Conversely, if a solution vector* $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ *with* $0 \leq \bar{x}_i^\omega \leq 1,\ i \in I,\ \omega \in \Omega$*, is feasible to constraints (2.4e) and (2.4g), and inequality (2.10) holds for the* $\bar{\mathbf{x}}$ *vector, there exist solution vectors* $(\bar{\mathbf{s}}, \bar{\mathbf{z}})$ *such that* $(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{s}}, \bar{\mathbf{z}})$ *is feasible to the LP relaxation of model (2.9).*

Proof. Suppose $(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{s}}, \bar{\mathbf{z}})$ is feasible to the LP relaxation of model (2.9). Constraint (2.4d) reads,

$$\bar{x}_i^\omega \geq \bar{x}_{i'}^\omega + \bar{s}_{ii'} - 1, \qquad i \neq i', i, i' \in I,\ \omega \in \Omega.$$

Rearranging and maximizing over the scenarios, we obtain

$$\max_{\omega\in\Omega}\{\bar{x}_{i'}^{\omega}-\bar{x}_{i}^{\omega}\}\leq 1-\bar{s}_{ii'}\Rightarrow\alpha_{i'i}\leq 1-\bar{s}_{ii'},\quad i\neq i'.$$

We similarly obtain $\alpha_{ii'}\leq 1-\bar{s}_{i'i},\ i'\neq i$. Summing these two inequalities yields $\alpha_{ii'}+\alpha_{i'i}\leq 2-\{\bar{s}_{ii'}+\bar{s}_{i'i}\}=1$, where the equality follows from constraint (2.9f).

We first show that at least one of $\alpha_{ii'}$ and $\alpha_{i'i}$ is nonnegative: Suppose $\alpha_{i'i}=\max_{\omega\in\Omega}\{\bar{x}_{i'}^{\omega}-\bar{x}_{i}^{\omega}\}<0$, and ω^* is the maximizing scenario. Then, $x_i^{\omega^*}-x_{i'}^{\omega^*}>0$. This implies $\alpha_{ii'}=\max_{\omega\in\Omega}\{\bar{x}_{i}^{\omega}-\bar{x}_{i'}^{\omega}\}\geq x_i^{\omega^*}-x_{i'}^{\omega^*}>0$.

Having shown at least one of $\alpha_{ii'}$ and $\alpha_{i'i}$ is nonnegative, we assume, without loss of generality, $\alpha_{ii'}\geq 0$. We construct the $\bar{\mathbf{s}}$ vector as follows: Set $\bar{s}_{ii'}=\alpha_{ii'},\ \bar{s}_{i'i}=1-\alpha_{ii'},\ i\neq i', i,i'\in I$. Then, by inequality (2.10) $\bar{s}_{ii'}\leq 1-\alpha_{i'i}=1-\max_{\omega\in\Omega}\{\bar{x}_{i'}^{\omega}-\bar{x}_{i}^{\omega}\}\leq 1-\{\bar{x}_{i'}^{\omega}-\bar{x}_{i}^{\omega}\},\ \omega\in\Omega\Rightarrow\bar{x}_{i}^{\omega}\geq\bar{x}_{i'}^{\omega}+\bar{s}_{ii'}-1,\quad\omega\in\Omega$. Similarly, $\bar{s}_{i'i}=1-\alpha_{ii'}=1-\max_{\omega\in\Omega}\{\bar{x}_{i}^{\omega}-\bar{x}_{i'}^{\omega}\}\leq 1-\{\bar{x}_{i}^{\omega}-\bar{x}_{i'}^{\omega}\},\ \omega\in\Omega\Rightarrow\bar{x}_{i'}^{\omega}\geq\bar{x}_{i}^{\omega}+\bar{s}_{i'i}-1,\ \omega\in\Omega$. Therefore, constraint (2.4d) is satisfied. By construction, we have $0\leq\bar{s}_{ii'}\leq 1$, and $\bar{s}_{ii'}+\bar{s}_{i'i}=1$. Thus, constraint (2.9f) and the LP relaxation of constraint (2.4c) are satisfied.

We construct the $\bar{\mathbf{z}}$ vector as follows: Set $\bar{z}_{ip}=1/|I|,\ i\in I,\ p\in P$. It is clear that constraints (2.9b), (2.9c) and the LP relaxation of constraint (2.9d) are satisfied. Constraint (2.9e) places no restriction on $\bar{\mathbf{s}}$ variable and hence is satisfied. Also, by hypothesis constraints (2.4e) and (2.4g) and the LP relaxation of constraint (2.4f) are satisfied. □

Proposition 2.3.4. *If a solution vector* $(\bar{\mathbf{x}},\bar{\mathbf{y}},\bar{\mathbf{s}})$ *with* $\bar{\mathbf{x}}=(\bar{x}_i^{\omega})_{i\in I,\omega\in\Omega}$ *is feasible to the LP relaxation of model (2.4), then inequality (2.10) holds. Con-*

versely, if a solution vector $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ *with* $0 \leq \bar{x}_i^\omega \leq 1,\ i \in I,\ \omega \in \Omega$, *is feasible to constraints (2.4e) and (2.4g), and inequality (2.10) holds for the* $\bar{\mathbf{x}}$ *vector, there exists a solution vector* $\bar{\mathbf{s}}$ *such that* $(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{s}})$ *is feasible to the LP relaxation of model (2.4).*

Corollary 2.3.5. *The LP relaxations of models (2.4) and (2.9) have the same optimal value.*

Proof. Suppose $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{s}^*, \mathbf{z}^*)$ is an optimal solution to the LP relaxation of model (2.9). Then, $(\mathbf{x}^*, \mathbf{y}^*)$ is feasible to constraints (2.4e) and (2.4g), and we have $0 \leq x_i^{*\omega} \leq 1,\ i \in I,\ \omega \in \Omega$. By the first part of Proposition 2.3.3, $\mathbf{x}^*$ satisfies inequality (2.10). But, by the second part of Proposition 2.3.4, we can find a solution vector $\hat{\mathbf{s}}^*$ such that $(\mathbf{x}^*, \mathbf{y}^*, \hat{\mathbf{s}}^*)$ is feasible to the LP relaxation of model (2.4). Thus, the optimal value of LP relaxation of model (2.4) is no worse than that of model (2.9). We can carry out the same argument in the opposite direction, proving the corollary. □

Now that we have shown that the optimal values of the LP relaxations of models (2.4) and (2.9) are equal, we expect model (2.4) to be more efficient since it is smaller in size. It is natural to ask whether this theoretical result holds in practice. To this end, we compare models (2.4) and (2.9) on a set of multidimensional knapsack problem instances. The notation and formulation of the multidimensional knapsack model follow:

Indices and sets:

$i \in I$ items

$t \in T$ knapsack dimensions

Data:

a_i profit of item i

b_t capacity of dimension t

c_{it} resource consumption of item i for dimension t

Decision variables:

x_i 1 if item i is selected; 0 otherwise

Formulation:

$$\max_{\mathbf{x}} \quad \sum_{i \in I} a_i x_i \tag{2.11a}$$

$$\text{s.t.} \quad \sum_{i \in I} c_{it} x_i \leq b_t, \qquad t \in T, \tag{2.11b}$$

$$x_i \in \{0, 1\}, \qquad i \in I. \tag{2.11c}$$

The objective function in (2.11a) sums the profit contributions of the selected items. Constraint (2.11b) ensures that the selected items fit within the capacity of the knapsack's dimension, b_t, $t \in T$. Constraint (2.11c) handles the binary restrictions. The optimal solution to the multidimensional knapsack model (2.11) gives the set of items which maximizes total profit and meets the knapsack's capacities.

We give the mathematical formulation for the application of model (2.4) to the multidimensional knapsack problem. Application of model (2.9) can be

performed similarly. The notation and formulation of the prioritization model follow:

Indices and sets:

$i, i' \in I$	items
$t \in T$	knapsack dimensions
$\omega \in \Omega$	scenarios

Data:

a_i^ω	profit of item i under scenario ω
b_t^ω	capacity of dimension t under scenario ω
c_{it}^ω	resource consumption of item i for dimension t under scenario ω
q^ω	probability of scenario ω

Decision variables:

$s_{ii'}$	1 if item i has no lower priority than i'; 0 otherwise
x_i^ω	1 if item i is selected under scenario ω; 0 otherwise

Formulation:

$$\max_{\mathbf{s},\mathbf{x}} \quad \sum_{\omega\in\Omega} q^\omega \sum_{i\in I} a_i^\omega x_i^\omega \tag{2.12a}$$

$$\text{s.t.} \quad s_{ii'} + s_{i'i} \geq 1, \qquad i < i', \;\; i, i' \in I, \tag{2.12b}$$

$$s_{ii'} \in \{0,1\}, \qquad i \neq i', i, i' \in I, \tag{2.12c}$$

$$x_i^\omega \geq x_{i'}^\omega + s_{ii'} - 1, \qquad i \neq i', i, i' \in I, \, \omega \in \Omega, \tag{2.12d}$$

$$\sum_{i\in I} c_{it}^\omega x_i^\omega \leq b_t^\omega, \qquad t \in T, \;\; \omega \in \Omega, \tag{2.12e}$$

$$x_i^\omega \in \{0,1\}, \qquad i \in I, \omega \in \Omega. \tag{2.12f}$$

The test problems we use for the underlying model (2.11) are taken from the OR Library [3], which hosts instances of the multidimensional knapsack problem. We use the two collections of instances, labeled "mknap1" and "mknap2." We use problem instances (10,10), (15,10), (20,10), (28,10), (39,5) and (50,5) from "mknap1" and instances (60,5), (70,5), (80,5), (90,5) from "mknap2" where the first number represents the number of items and the second represents the number of knapsack dimensions.

In the prioritization setting, there is a third parameter: number of scenarios. One application of the multidimensional knapsack model is in capital budgeting. There, the items are projects, and the dimensions of the knapsack are time periods, e.g., years. So, b_t is the available budget in year t, c_{it} is the cost in year t of selecting item i, and a_i is the net present value of item i, i.e., its profit. We use this terminology in describing how we model the uncertainty in cost, profit, and budget, which determines the number of scenarios. In our case, there are three equally-likely scenarios for the cost of an item: optimistic, pessimistic, and most likely. In the optimistic scenario, we multiply the cost of each item in each year by 0.8; in the pessimistic, by 1.2; and in the most likely, by 1. Budget uncertainty is similar to the cost uncertainty, except it has two scenarios with equal probabilities. In the first scenario, we multiply each year's budget by 0.8; in the second, by 1.2. We have 7 different settings for the number of scenarios: 1, 3, 9, 18 27, 54, and 81. We now describe how we generate the scenarios in all these settings.

- If the number of scenarios is 1, there is no uncertainty. We solve the

deterministic multidimensional knapsack problem.

- If the number of scenarios is 3, there is only cost uncertainty. All items move together. That is, all items realize one of the pessimistic, most-likely, and optimistic scenarios.

- If the number of scenarios is 9, again there is only cost uncertainty. The items fall into two groups. Each group moves independent of the other. In the first scenario, items in groups 1 and 2 realize pessimistic forecast; in scenario 2, group 1 realizes pessimistic and group 2 realizes optimistic; in scenario 3, group 1 realizes pessimistic and group 2 realizes most likely, and so on. We put items $1, \ldots, \lceil \frac{n}{2} \rceil$ into the first group, and items $\lceil \frac{n}{2} \rceil + 1, \ldots, n$ into the second, where $n = |I|$ is the number of items.

- In general, if the number of scenarios is not 1, 18 or 54, we have only cost uncertainty and items fall into $\beta = \log_3(|\Omega|)$ groups, where $|\Omega|$ is the number of scenarios. Items in the same group move together, and the ones in different groups move independently, which yields $3^\beta = |\Omega|$ scenarios. Items $1, \ldots, \lceil \frac{n}{\beta} \rceil$ fall into the first group, items $\lceil \frac{n}{\beta} \rceil + 1, \ldots, \lceil \frac{n}{\beta/2} \rceil$ fall into the second, items $\lceil \frac{n}{\beta/2} \rceil + 1, \ldots, \lceil \frac{n}{\beta/3} \rceil$ fall into the third, and so on, until items $\lceil \frac{n}{\beta/(\beta-1)} \rceil + 1, \ldots, n$ fall into the last group.

- If the number of items is 18, the cost uncertainty is the same as the 9-scenario case. We additionally include budget uncertainty. Budget and cost uncertainty are independent, which yields $9 \times 2 = 18$ scenarios.

- If the number of items is 54, it is similar to the 18-scenario case, except we add two budget scenarios on top of the 27-scenario cost uncertainty.

- Finally, in order to have two problem instances for each scenario, we introduce another budget-cost multiplier. In this setting, we have a multiplier pair of (0.4,1.6) instead of (0.8,1.2). That is, we multiply the costs by 0.4 in the optimistic case and by 1.6 in the pessimistic case. Similarly, the budget is either multiplied by 0.4 or by 1.6.

We label a problem instance with multiplier pair (0.8,1.2) with a 0, and an instance with pair (0.4,1.6) with a 1. So, "20-10-9-0" means the multidimensional knapsack problem (20,10) from "mknap1" with 9 scenarios and multiplier pair (0.8,1.2). We use Cplex version 11.1 on a Dell Poweredge 2950 computer with Intel (Xenon) 3.73 GHz processor and with 8 GB of RAM, and we report the results in Table 2.2.

The first column in Table 2.2 gives the number of scenarios; the first row gives the number of items-dimensions. In each cell there are two values. The upper value corresponds to model (2.4); the lower one, to model (2.9). Each value is the average over two problem instances. The values with the percentage sign give the remaining optimality gaps after we spend an hour on the problem. These percentage values are the *geometric* averages over the two instances. The values without the percentage sign give the running times in terms of CPU seconds to solve the instances to 0.01% optimality. These values are the *arithmetic* averages over the two instances. Some of the cells

are empty. It means after spending an hour the corresponding model cannot obtain an optimality gap for at least one of the instances. We mostly solve the smaller instances to 0.01% optimality, and spend only an hour on the larger ones. This is not quite strict, however. Some of the instances we can solve to 0.01% optimality in a couple of hours using one model, but cannot solve using the other. For these instances we report only one-hour-optimality-gaps under both models, for a fair comparison of the two models.

Interpreting the results in Table 2.2, the problem instances with 20 or fewer items do not strictly favor either of the models. Those with 28 or more items, however, favor model (2.4) over (2.9). This observation holds in both the larger-scenario instances and the smaller-scenario instances, i.e., those we solve to 0.01% optimality and those we do not reach a solution to 0.01% optimality in one hour. This conclusion gives evidence to what we have inferred from the LP relaxations and the sizes of the two formulations. Model (2.4) has $2|I| + |I|(|I| - 1)$ fewer constraints and $|I|^2$ fewer variables than model (2.9). Thus, it is expected that the difference become more visible as the number of activities, i.e., items, increases.

2.4 Scenario Prioritization

In the prioritized RCASP model (2.3), we place activities on a priority list and let each scenario select its optimal set of activities, making sure that the activity selection decision is prioritized. We now consider a complementary formulation, where we place the scenarios on a priority list and let each activity

Table 2.2: Computational comparison of models (2.4) and (2.9). The first column is the number of scenarios, the first row is the number of items-dimensions. The percentage values are the remaining optimality gaps after solving the instances for an hour of CPU time. The plain values are the CPU times in seconds to solve the instances to 0.01% optimality. The upper value in the cell corresponds to model (2.4); the lower, to (2.9).

# scenarios	# items - # dimensions									
	10-10	15-10	20-10	28-10	39-5	50-5	60-5	70-5	80-5	90-5
3	0	0	0	1	29	34	9	10	24	17
	0	0	1	4	188	1173	180	758	1430	2382
9	1	8	48	150	0%	0.95%	0.43%	0%	0%	0%
	1	8	46	235	1.42%	2.70%	1.77%	0.59%	0.28%	0.21%
18	4	27	717	1293	1.49%	3.41%	2.02%	1.17%	2.14%	
	4	32	697	2979	2.52%	5.05%				
27	17	5778	1.06%	1.42%			2.70%	3.64%		
	12	5956	1.12%	1.42%						
54	174	0.87%	3.20%	1.93%						
	114	1.51%	2.63%	2.20%						
81	1.92%	5.66%	6.10%							
	0%	6.29%	5.96%							

select its optimal set of scenarios, making sure that the scenario selection decision is prioritized. What it means for a scenario selection decision to be prioritized is similar to the prioritized activity-selection rule: An activity cannot select a lower-priority scenario unless all higher-priority scenarios are selected, and that activity must select either all or none of the scenarios on each priority level.

Let $\mathcal{K}_k \subseteq \Omega,\ k = 1, \ldots, K$, denote the priority levels, where $\mathcal{K}_k \neq \emptyset$, $k = 1, \ldots, K$, $\bigcup_{k=1}^{K} \mathcal{K}_k = \Omega$, and $\mathcal{K}_k \cap \mathcal{K}_{k'} = \emptyset,\ k \neq k',\ k, k' = 1, \ldots, K$. Then $\mathcal{K} = [\mathcal{K}_1, \mathcal{K}_2, \ldots, \mathcal{K}_K]$ denotes a priority list for scenarios, where scenarios in $\mathcal{K}_1$ have higher priority than those in $\mathcal{K}_2$, scenarios in $\mathcal{K}_2$ have higher priority than those in $\mathcal{K}_3$, and so forth. Let $G(\mathcal{K})$ denote the optimal value of the expected cost of RCASP, subject to the prioritized scenario selection restriction imposed by $\mathcal{K}$. That is,

$$
\begin{aligned}
G(\mathcal{K}) = \quad & \min_{\mathbf{x},\mathbf{y}} \quad \sum_{\omega \in \Omega} q^{\omega} (\mathbf{c}_x^{\omega} \mathbf{x}^{\omega} + \mathbf{c}_y^{\omega} \mathbf{y}^{\omega}) && && \text{(2.13a)} \\
& \text{s.t.} \quad \mathbf{A}^{\omega} \mathbf{x}^{\omega} \leq \mathbf{b}^{\omega}, && \omega \in \Omega, && \text{(2.13b)} \\
& \qquad \mathbf{x}^{\omega} \in \{0,1\}^{|I|}, && \omega \in \Omega, && \text{(2.13c)} \\
& \qquad (\mathbf{x}^{\omega}, \mathbf{y}^{\omega}) \in C^{\omega}, && \omega \in \Omega, && \text{(2.13d)} \\
& \qquad x_i^{\omega} \geq x_i^{\omega'}, && i \in I,\ \omega \in \mathcal{K}_k,\ \omega' \in \mathcal{K}_{k'}, && \\
& && k < k',\ k, k' = 1, \ldots, K, && \text{(2.13e)} \\
& \qquad x_i^{\omega} = x_i^{\omega'}, && i \in I,\ \omega, \omega' \in \mathcal{K}_k,\ k = 1, \ldots, K. && \text{(2.13f)}
\end{aligned}
$$

Minimizing the objective function in (2.13a) minimizes the expected cost of the decisions. Constraints (2.13b), (2.13c) and (2.13d) are the same as those in (2.1b), (2.1c) and (2.1d), except they now depend on $\omega \in \Omega$. Constraint (2.13e) imposes the first condition of the prioritized scenario selection: If an activity is selected under a lower-priority scenario, it must also be selected under a higher-priority scenario. Constraint (2.13f) imposes the second condition: An activity is either selected, or not, under both scenarios on the same priority level. Based on $G(\mathcal{K})$ defined by model (2.13), we define the scenario prioritization model as:

$$\min_{\mathcal{K},K} \quad G(\mathcal{K}) \tag{2.14a}$$

$$\text{s.t.} \quad \mathcal{K} = [\mathcal{K}_1, \ldots, \mathcal{K}_K], \tag{2.14b}$$

$$\mathcal{K}_k \subseteq \Omega, \qquad k = 1, \ldots, K, \tag{2.14c}$$

$$\mathcal{K}_k \neq \emptyset, \qquad k = 1, \ldots, K, \tag{2.14d}$$

$$\mathcal{K}_k \cap \mathcal{K}_{k'} = \emptyset, \qquad k \neq k', k, k' = 1, \ldots, K, \tag{2.14e}$$

$$\bigcup_{k=1,\ldots,K} \mathcal{K}_k = \Omega. \tag{2.14f}$$

Model (2.14) selects the optimal priority list for scenarios so that the expected cost of the RCASP subject to the prioritized scenario-selection constraints is minimized. Similar to the discussion surrounding model (2.3), the minimization in model (2.14) is taken over both $\mathcal{K}$ and K. But, in referring to a priority list for the scenarios, we omit K occasionally, when it is not relevant. In these situations, we assume the number of priority levels is K.

We now prove a result that models (2.3) and (2.14) are equivalent, i.e., a solution to one can be transformed to a solution to the other with the same objective function value. This result enables us to solve either model, whichever is simpler, and obtain a solution to the other.

Theorem 2.4.1. *Consider model (2.3) with $F^{\omega}(\cdot)$ defined in (2.2) and model (2.14) with $G(\cdot)$ defined in (2.13). Let $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) = (\bar{\mathbf{x}}^{\omega}, \bar{\mathbf{y}}^{\omega})_{\omega \in \Omega}$, where $(\bar{\mathbf{x}}^{\omega}, \bar{\mathbf{y}}^{\omega})$ satisfies (2.2b)–(2.2d), $\omega \in \Omega$. There exists a priority list $\bar{\mathcal{K}}$ for the set of scenarios, Ω, such that $(\bar{\mathcal{K}}, \bar{\mathbf{x}})$ satisfies (2.13e)–(2.13f) and $\bar{\mathcal{K}}$ satisfies (2.14b)–(2.14f) if and only if there exists a priority list $\bar{\mathcal{L}}$ for the set of activities, I, such that $(\bar{\mathcal{L}}, \bar{\mathbf{x}})$ satisfies (2.2e)–(2.2f), $\omega \in \Omega$, and $\bar{\mathcal{L}}$ satisfies (2.3b)–(2.3f). Furthermore, constructing $\bar{\mathcal{L}}$ from $(\bar{\mathcal{K}}, \bar{\mathbf{x}})$ and constructing $\bar{\mathcal{K}}$ from $(\bar{\mathcal{L}}, \bar{\mathbf{x}})$ can be performed in $O(|I||\Omega|)$ time.*

Proof. Suppose the hypothesis for $(\bar{\mathcal{K}}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$ holds. It suffices to find a priority list $\bar{\mathcal{L}}$ for the set of activities, satisfying constraints (2.3b)–(2.3f), such that $(\bar{\mathcal{L}}, \bar{\mathbf{x}})$ is feasible to constraints (2.2e)–(2.2f).

Let $\bar{K}$ denote the number of priority levels in $\bar{\mathcal{K}}$. Let $S_k = \{i \in I \,|\, \bar{x}_i^{\omega} = 1, \text{ for some } \omega \in \bar{\mathcal{K}}_k\}$, $k = 1, \ldots, \bar{K}$, and let $S_0 = I$, $S_{\bar{K}+1} = \emptyset$. By constraints (2.13e) and (2.13f), we have $S_0 \supseteq S_1 \supseteq \cdots \supseteq S_{\bar{K}} \supseteq S_{\bar{K}+1}$. Consider Algorithm 1 to construct $\bar{\mathcal{L}}$. By construction, $\bar{\mathcal{L}}$ satisfies constraints (2.3b)–(2.3f). Consider two activities i, i', where $i \in \bar{\mathcal{L}}_l$, $i' \in \bar{\mathcal{L}}_{l'}$. We must show that if $l = l'$, constraint (2.2f) is satisfied, and otherwise constraint (2.2e) is satisfied.

Algorithm 1 Construct $\bar{\mathcal{L}}$ from $\bar{\mathcal{K}}$

Input: $\bar{\mathcal{K}} = [\bar{\mathcal{K}}_1, \ldots, \bar{\mathcal{K}}_K]$, where K is the number of priority levels.
$\bar{\mathbf{x}} = (\bar{x}_i^\omega)_{i \in I, \omega \in \Omega}$.
Output: $\bar{\mathcal{L}} = [\bar{\mathcal{L}}_1, \ldots, \bar{\mathcal{L}}_L]$, where L is the number of priority levels.

$S_k \leftarrow \{i \in I \mid \bar{x}_i^\omega = 1, \text{ for some } \omega \in \bar{\mathcal{K}}_k\}$, $k = 1, \ldots, \bar{K}$. $S_0 \leftarrow I$. $S_{\bar{K}+1} \leftarrow \emptyset$.
$t \leftarrow \bar{K}, j \leftarrow 1$.
repeat
  if $S_t \setminus S_{t+1} \neq \emptyset$ **then**
    $\bar{\mathcal{L}}_j \leftarrow S_t \setminus S_{t+1}$.
    $j \leftarrow j + 1$.
  end if
  $t \leftarrow t - 1$.
until $t = -1$
$L \leftarrow j$, and $\bar{\mathcal{L}} \leftarrow [\bar{\mathcal{L}}_1, \ldots, \bar{\mathcal{L}}_L]$.

- If $l = l'$, by the construction of $\bar{\mathcal{L}}$, $\exists t \in \{0, \ldots, \bar{K}\}$ such that $i, i' \in S_k$ for $k = 0, \ldots, t$, and $i, i' \notin S_k$ for $k = t+1, \ldots, \bar{K}+1$. That is, $\bar{x}_i^\omega = \bar{x}_{i'}^\omega = 1$ for $\omega \in \bigcup_{k=1}^{t} \bar{\mathcal{K}}_k$, and $\bar{x}_i^\omega = \bar{x}_{i'}^\omega = 0$ for $\omega \in \bigcup_{k=t+1}^{\bar{K}} \bar{\mathcal{K}}_k$. Hence, $\bar{x}_i^\omega = \bar{x}_{i'}^\omega$, $\forall \omega \in \Omega$.

- Suppose, without loss of generality, that $l < l'$. Then by the construction of $\bar{\mathcal{L}}$, $\exists t > t' \in \{0, \ldots, \bar{K}\}$ such that $i, i' \in S_k$ for $k = 0, \ldots, t'$, $i \in S_k$, $i' \notin S_k$ for $k = t'+1, \ldots, t$, and $i, i' \notin S_k$ for $k = t+1, \ldots, \bar{K}+1$. That is, $\bar{x}_i^\omega = \bar{x}_{i'}^\omega = 1$ for $\omega \in \bigcup_{k=1}^{t'} \bar{\mathcal{K}}_k$, $\bar{x}_i^\omega = 1$, $\bar{x}_{i'}^\omega = 0$ for $\omega \in \bigcup_{k=t'+1,\ldots,t} \bar{\mathcal{K}}_k$, and $\bar{x}_i^\omega = \bar{x}_{i'}^\omega = 0$ for $\omega \in \bigcup_{k=t+1,\ldots,\bar{K}} \bar{\mathcal{K}}_k$. Hence, $\bar{x}_i^\omega \geq \bar{x}_{i'}^\omega$, $\forall \omega \in \Omega$.

Suppose the hypothesis for $(\bar{\mathcal{L}}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$ holds. It suffices to find a priority list $\bar{\mathcal{K}}$ for the set of scenarios, satisfying constraints (2.14b)–(2.14f), such that $(\bar{\mathcal{K}}, \bar{\mathbf{x}})$ is feasible to constraints (2.13e)–(2.13f).

Let $\bar{L}$ denote the number of priority levels in $\bar{\mathcal{L}}$. Let $S_l = \{\omega \in \Omega \,|\, \bar{x}_i^\omega = 1, \text{ for some } i \in \bar{\mathcal{L}}_l\}$, $l = 1, \ldots, \bar{L}$, and $S_0 = \Omega$, $S_{\bar{L}+1} = \emptyset$. By constraints (2.2e) and (2.2f), we have $S_0 \supseteq S_1 \supseteq \cdots \supseteq S_{\bar{L}} \supseteq S_{\bar{L}+1}$. Consider Algorithm 2 to construct $\bar{\mathcal{K}}$:

Algorithm 2 Construct $\bar{\mathcal{K}}$ from $\bar{\mathcal{L}}$

Input: $\bar{\mathcal{L}} = [\bar{\mathcal{L}}_1, \ldots, \bar{\mathcal{L}}_L]$, where L is the number of priority levels.
$\bar{\mathbf{x}} = (\bar{x}_i^\omega)_{i \in I, \omega \in \Omega}$.
Output: $\bar{\mathcal{K}} = [\bar{\mathcal{K}}_1, \ldots, \bar{\mathcal{K}}_K]$, where K is the number of priority levels.

$S_l \leftarrow \{\omega \in \Omega \,|\, \bar{x}_i^\omega = 1, \text{ for some } i \in \bar{\mathcal{L}}_l\}$, $l = 1, \ldots, \bar{L}$. $S_0 \leftarrow \Omega$. $S_{\bar{L}+1} \leftarrow \emptyset$.
$t \leftarrow \bar{L}, j \leftarrow 1$.
repeat
  if $S_t \setminus S_{t+1} \neq \emptyset$ **then**
    $\bar{\mathcal{K}}_j \leftarrow S_t \setminus S_{t+1}$.
    $j \leftarrow j + 1$.
  end if
  $t \leftarrow t - 1$.
until $t = -1$
$K \leftarrow j$, and $\bar{\mathcal{K}} \leftarrow [\bar{\mathcal{K}}_1, \ldots, \bar{\mathcal{K}}_K]$.

By construction, $\bar{\mathcal{K}}$ satisfies constraints (2.14b)–(2.14f). Consider two scenarios ω, ω', where $\omega \in \bar{\mathcal{K}}_k$, $\omega' \in \bar{\mathcal{K}}_{k'}$. We must show that if $k = k'$, constraint (2.13f) is satisfied, and otherwise, constraint (2.13e) is satisfied.

- If $k = k'$, by the construction of $\bar{\mathcal{K}}$, $\exists\, t \in \{0, \ldots, \bar{L}\}$ such that $\omega, \omega' \in S_l$ for $l = 0, \ldots, t$, and $\omega, \omega' \notin S_l$ for $l = t+1, \ldots, \bar{L}+1$. That is, $\bar{x}_i^\omega = \bar{x}_i^{\omega'} = 1$ for $i \in \bigcup_{l=1}^{t} \bar{\mathcal{L}}_l$, and $\bar{x}_i^\omega = \bar{x}_i^{\omega'} = 0$ for $i \in \bigcup_{l=t+1}^{\bar{L}} \bar{\mathcal{L}}_l$. Hence, $\bar{x}_i^\omega = \bar{x}_i^{\omega'}$, $\forall i \in I$.

- Suppose, without loss of generality, that $k < k'$. Then by the construction of $\bar{\mathcal{K}}$, $\exists\, t > t' \in \{0, \ldots, \bar{L}\}$ such that $\omega, \omega' \in S_l$ for $l = 0, \ldots, t'$, $\omega \in S_l$, $\omega' \notin S_l$ for $l = t'+1, \ldots, t$, and $\omega, \omega' \notin S_l$ for $l = t+1, \ldots, \bar{L}+1$. That is, $\bar{x}_i^{\omega} = \bar{x}_i^{\omega'} = 1$ for $i \in \bigcup_{l=1}^{t'} \bar{\mathcal{L}}_l$, $\bar{x}_i^{\omega} = 1$, $\bar{x}_i^{\omega'} = 0$ for $i \in \bigcup_{l=t'+1}^{t} \bar{\mathcal{L}}_l$, and $\bar{x}_i^{\omega} = \bar{x}_i^{\omega'} = 0$ for $i \in \bigcup_{l=t+1}^{\bar{L}} \bar{\mathcal{L}}_l$. Hence, $\bar{x}_i^{\omega} \geq \bar{x}_i^{\omega'},\ \forall i \in I$.

Constructing the sets $S_k,\ k = 1, \ldots, \bar{K}$, at the beginning of Algorithm 1 takes $O(|I|\bar{K})$ time and so does the rest of the algorithm. Similarly, Algorithm 2 takes $O(|\Omega|\bar{L})$ time. Since $\bar{K} \leq |\Omega|$ and $\bar{L} \leq |I|$, both algorithms can be performed in $O(|I||\Omega|)$ time.

□

We develop an IP formulation for model (2.14), analogous to the development of model (2.4) for model (2.3). This model is very similar to model (2.4), except we now prioritize scenarios instead of activities. The notation and formulation read similarly; hence, we do not give details. For this formulation it is convenient to have the set of scenarios, Ω, to be an ordered set so that it makes sense to write restrictions on it, such as $\omega < \omega' \in \Omega$. Consider the following notation and the associated formulation:

Indices and sets:

$i \in I$	activities
$\omega,\, \omega' \in \Omega$	scenarios

Data:

$(\mathbf{A}^\omega, \mathbf{b}^\omega, \mathbf{c}_x^\omega, \mathbf{c}_y^\omega, C^\omega)$	the problem parameters under scenario ω
q^ω	probability of scenario ω

Decision variables:

$s_{\omega\omega'}$	1 if scenario ω has no lower priority than ω'; 0 otherwise
$\mathbf{x}^\omega,\, \mathbf{y}^\omega$	the decision vectors under scenario ω

Formulation:

$$\min_{\mathbf{s},\,\mathbf{x},\,\mathbf{y}} \quad \sum_{\omega\in\Omega} q^\omega(\mathbf{c}_x^\omega \mathbf{x}^\omega + \mathbf{c}_y^\omega \mathbf{y}^\omega) \tag{2.15a}$$

$$\text{s.t.} \quad s_{\omega\omega'} + s_{\omega'\omega} \geq 1, \qquad \omega < \omega',\ \ \omega,\, \omega' \in \Omega, \tag{2.15b}$$

$$s_{\omega\omega'} \in \{0,1\}, \qquad \omega \neq \omega', \omega,\, \omega' \in \Omega, \tag{2.15c}$$

$$x_i^\omega \geq x_i^{\omega'} + s_{\omega\omega'} - 1, \qquad i \in I,\, \omega \neq \omega',\, \omega,\, \omega' \in \Omega, \tag{2.15d}$$

$$(2.4e),\ (2.4f),\ (2.4g). \tag{2.15e}$$

It is possible to develop an IP formulation for the scenario-prioritization models (2.14) and (2.15) with a total order restriction, as we have done for activity-prioritization models (2.3) and (2.4), by models (2.8) and (2.9). Doing so would lead to corollaries similar to Corollaries 2.3.2 and 2.3.5, stating that the optimal values of models (2.14) and (2.15) with and without a total order restriction are equal and the optimal values of the LP relaxation of

model (2.15) with and without a total order restriction are the same. We do not pursue this path as the development above already shows that enforcing a total order restriction only increases the problem size, without tightening the formulation. We instead compare the optimal values of the LP relaxations of models (2.4) and (2.15). This is useful as we already have algorithms to transform a solution to one model to a solution to the other model in polynomial time. Hence, we have the freedom of solving the more efficient formulation and obtaining solutions to either problem. We first replicate Proposition 2.3.3 for model (2.15) without proof, and then compare the LP relaxations of the two models.

Proposition 2.4.2. *If a solution vector* $(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{s}})$ *with* $\bar{\mathbf{x}} = (\bar{x}_i^{\omega})_{i \in I, \omega \in \Omega}$ *is feasible to the LP relaxation of model (2.15), then*

$$\alpha_{\omega\omega'} + \alpha_{\omega'\omega} \leq 1, \quad \omega \neq \omega',\ \omega,\ \omega' \in \Omega, \tag{2.16}$$

holds, where $\alpha_{\omega\omega'} = \max_{i \in I} \{\bar{x}_i^{\omega} - \bar{x}_i^{\omega'}\}$. *Conversely, if a solution vector* $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ *with* $0 \leq \bar{x}_i^{\omega} \leq 1,\ i \in I,\ \omega \in \Omega$, *is feasible to constraints (2.4e) and (2.4g), and inequality (2.16) holds for the* $\bar{\mathbf{x}}$ *vector, there exists a vector* $\bar{\mathbf{s}}$ *such that* $(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{s}})$ *is feasible to the LP relaxation of model (2.15).*

Theorem 2.4.3. *The LP relaxations of models (2.4) and (2.15) have the same optimal value.*

Proof. Suppose $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{s}^*)$ is an optimal solution to the LP relaxation of model (2.4). Then, $(\mathbf{x}^*, \mathbf{y}^*)$ is feasible to constraints (2.4e) and (2.4g), and

$0 \leq x_i^{*\omega} \leq 1,\ i \in I,\ \omega \in \Omega$. And, by the first part of Proposition 2.3.4, $\mathbf{x}^*$ satisfies inequality (2.10). That is,

$$
\begin{aligned}
& \alpha_{ii'} + \alpha_{i'i} \leq 1, \quad i \neq i',\ i, i' \in I \\
\Leftrightarrow \quad & \max_{\omega \in \Omega} \{x_i^{*\omega} - x_{i'}^{*\omega}\} + \max_{\omega \in \Omega} \{x_{i'}^{*\omega} - x_i^{*\omega}\} \leq 1, \quad i \neq i',\ i, i' \in I \\
\Leftrightarrow \quad & x_i^{*\omega} - x_{i'}^{*\omega} + x_{i'}^{*\omega'} - x_i^{*\omega'} \leq 1, \quad \omega, \omega' \in \Omega,\ i \neq i',\ i, i' \in I \\
\Leftrightarrow \quad & x_i^{*\omega} - x_{i'}^{*\omega} + x_{i'}^{*\omega'} - x_i^{*\omega'} \leq 1, \quad \omega \neq \omega',\ \omega, \omega' \in \Omega,\ i, i' \in I \\
\Leftrightarrow \quad & x_i^{*\omega} - x_i^{*\omega'} + x_{i'}^{*\omega'} - x_{i'}^{*\omega} \leq 1, \quad \omega \neq \omega',\ \omega, \omega' \in \Omega,\ i, i' \in I \\
\Leftrightarrow \quad & \max_{i \in I} \{x_i^{*\omega} - x_i^{*\omega'}\} + \max_{i \in I} \{x_i^{*\omega'} - x_i^{*\omega}\} \leq 1, \quad \omega \neq \omega',\ \omega, \omega' \in \Omega \\
\Leftrightarrow \quad & \alpha_{\omega\omega'} + \alpha_{\omega'\omega} \leq 1, \quad \omega \neq \omega',\ \omega, \omega' \in \Omega.
\end{aligned}
$$

The first equivalence is the definition of $\alpha_{ii'}, i \neq i',\ i, i' \in I$. The second one comes from the definition of the *max* operator. In the third one, we exclude or include the scenario indices such that $\omega = \omega'$ and the activity indices such that $i = i'$. The exclusion and inclusion can be done with equivalence since for indices $i = i'$ or $\omega = \omega'$, the left-hand side of the inequality is zero. The fourth equivalence is just rearranging the terms. The fifth one again comes from the definition of the *max* operator. The final equivalence is the definition of $\alpha_{\omega\omega'}, \omega \neq \omega',\ \omega, \omega' \in \Omega$. Now, by the second part of Proposition 2.4.2, we can find a solution vector $\hat{\mathbf{s}}^*$ such that $(\mathbf{x}^*, \mathbf{y}^*, \hat{\mathbf{s}}^*)$ is feasible to model (2.4). Hence, the optimal value of the LP relaxation of model (2.15) is no worse than that of model (2.4). We can carry out the same argument in the opposite direction, proving the theorem. □

The constraint sets that links prioritization and activity selection in models (2.4) and (2.15) are (2.4d) and (2.15d), respectively. The number of constraints in these sets are $|I|(|I|-1)|\Omega|$ and $|\Omega|(|\Omega|-1)|I|$, respectively. And, the number of respective $\mathbf{s}$ variables in these two models is $|I|(|I|-1)$ and $|\Omega|(|\Omega|-1)$. For $|I| < |\Omega|$ model (2.4) is smaller than model (2.15), and vice versa. Having also proven that optimal values of their LP relaxations are equal, we expect the smaller model to be more efficient. To see if this theoretical conclusion applies in practice, we test both models on the set of problem instances from Table 2.2. The experimental setting is the same as in Table 2.2, and we report the results in Table 2.3. The results are interpreted as in Table 2.2 except now the upper and the lower values in a cell correspond to models (2.4) and (2.15), respectively. We indeed observe the expected behavior. For $|I|$ smaller than $|\Omega|$, model (2.4) performs better, while for $|\Omega|$ smaller than $|I|$, model (2.15) performs better. In Table 2.2 we terminate all runs after one hour, but here we show some longer running times to illustrate the relative merits of models (2.4) and (2.15).

2.5 Cutting Planes

To improve solution times further, we develop two sets of cutting planes for models (2.4) and (2.15). We first set notation and introduce terminology, referring to data $(\mathbf{A}^\omega, \mathbf{b}^\omega, \mathbf{c}_x^\omega, \mathbf{c}_y^\omega, C^\omega)$, $\omega \in \Omega$, used in models (2.4) and (2.15). Let $Y^\omega = \{(\mathbf{x}, \mathbf{y}) \,|\, \mathbf{x} \in \{0,1\}^{|I|}, \mathbf{A}^\omega \mathbf{x} \leq \mathbf{b}^\omega, (\mathbf{x}, \mathbf{y}) \in C^\omega\}$, and $X^\omega = \{\mathbf{x} \,|\, \exists \mathbf{y} \text{ such that } (\mathbf{x}, \mathbf{y}) \in Y^\omega\}$. For $\mathbf{x} \in X^\omega$, we let

Table 2.3: Comparing activity-prioritization model (2.4) with scenario-prioritization model (2.15). Everything is interpreted as in Table 2.2 except now upper and upper lower in a cell correspond to models (2.4) and (2.15), respectively.

# scenarios	# items - # dimensions									
	10-10	15-10	20-10	28-10	39-5	50-5	60-5	70-5	80-5	90-5
3	0 0	0 0	0 0	1 0	29 1	34 1	9 0	10 0	24 0	17 0
9	1 1	8 6	48 15	150 19	14430 220	203405 3895	10654 63	81082 203	22663 46	2578 23
18	4 22	27 107	717 1015	1293 860	1.49% 0.75%	3.41% 0.74%	2.02% 1.22%	1.17% 0.50%	2.14% 0.47%	 0.10%
27	17 638	5778 184952	8558 29633	1.42% 1.51%			2.70% 3.44%	3.64% 1.12%	 6.34%	 0.57%
54	174 181143	71873 246185	3.20% 6.25%	1.93% 2.81%						
81	1.92% 12.17%	5.66%	6.10%							

$$h^\omega(\mathbf{x}) \equiv \mathbf{c}_x^\omega \mathbf{x} + \min_{\mathbf{y}} \ \mathbf{c}_y^\omega \mathbf{y} \tag{2.17a}$$

$$\text{s.t. } (\mathbf{x}, \mathbf{y}) \in C^\omega. \tag{2.17b}$$

Model (2.17) is feasible by construction since $\mathbf{x} \in X^\omega$, and thus has a minimizer, which we denote by $\mathbf{y}^\omega(\mathbf{x})$. We note that given a solution, $\mathcal{K}$, to model (2.14) and a corresponding optimal solution, $(\mathbf{x}^\omega, \mathbf{y}^\omega)_{\omega\in\Omega}$, to model (2.13), we have $G(\mathcal{K}) = \sum_{\omega\in\Omega} q^\omega h^\omega(\mathbf{x}^\omega)$, where $G(\cdot)$ is defined as in model (2.14). Finally, for $\mathbf{x} \in \{0,1\}^{|I|}$, we let $S(\mathbf{x})$ denote the set of selected activities, i.e., $S(\mathbf{x}) = \{i \in I \,|\, x_i = 1\}$.

We say scenario ω is *improving* if for any two vectors $\mathbf{x}, \bar{\mathbf{x}} \in X^\omega$ with $S(\mathbf{x}) \supseteq S(\bar{\mathbf{x}})$, we have $h^\omega(\mathbf{x}) \leq h^\omega(\bar{\mathbf{x}})$. We let $\Omega_I \subseteq \Omega$ denote the set of improving scenarios. Given two scenarios ω, $\omega' \in \Omega$, we say ω *dominates* ω' if $\mathbf{x} \in X^{\omega'}$ implies $\mathbf{x} \in X^\omega$. We let Ω_D denote the set of dominating scenario pairs, i.e., $\Omega_D = \{(\omega, \omega') \in \Omega \times \Omega \mid \omega \text{ dominates } \omega'\}$.

For a vector $\mathbf{x} \in \{0,1\}^{|I|}$, we let $\mathbf{x}_{(i,i')}$ denote a new vector $\bar{\mathbf{x}} \in \{0,1\}^{|I|}$ which is the same as $\mathbf{x}$, except that its i^{th} element is set to 1, and i'^{th} element is set to 0, i.e., $\bar{x}_j = x_j$ for $j \notin \{i, i'\}$, $\bar{x}_i = 1$, $\bar{x}_{i'} = 0$. We say activity i *dominates* activity i' if, for any vector $\mathbf{x} \in X^\omega$ with $x_{i'} = 1$, $x_i = 0$, we have $\mathbf{x}_{(i,i')} \in X^\omega$ and $h^\omega(\mathbf{x}_{(i,i')}) \leq h^\omega(\mathbf{x})$, for all $\omega \in \Omega$. We let I_D denote the set of dominating activity pairs, i.e., $I_D = \{(i, i') \in I \times I \,|\, i \text{ dominates } i'\}$.

We illustrate this notation on the facility prioritization problem (2.6) and the prioritized multidimensional knapsack problem (2.12). In the multidi-

mensional knapsack problem, given two scenarios, ω, ω', scenario ω dominates scenario ω' if the costs under scenario ω are at most those under scenario ω', and if the budgets under scenario ω are at least those under scenario ω', i.e., $c_{it}^{\omega} \leq c_{it}^{\omega'}$, $i \in I,\ t \in T$, and $b_t^{\omega} \geq b_t^{\omega'}$, $t \in T$. Scenario ω is improving if the profits under scenario ω are nonnegative, i.e., $a_i^{\omega} \geq 0$, $i \in I$. In the facility prioritization problem, scenarios with larger budgets dominate scenarios with smaller budgets. That is, given two scenarios, ω, ω', scenario ω dominates scenario ω' if $k^{\omega} \geq k^{\omega'}$. All scenarios in the facility prioritization problem are improving since opening more facilities does not increase the total distance of the customers to their closest facilities. In the multidimensional knapsack prioritization problem, given two activities, i, i', if the profit of activity i is at least that of i' under all scenarios, and the cost of activity i is no more than that of i' under all scenarios, then i dominates i', i.e., $a_i^{\omega} \geq a_{i'}^{\omega}$, and $c_{it}^{\omega} \leq c_{i't}^{\omega}$, $\omega \in \Omega$. In the instances of facility prioritization problem that we describe in Section 2.2, we do not have any dominating activity pairs.

Given a vector $\mathbf{x}$ with $\mathbf{x} = (\mathbf{x}^{\omega})_{\omega \in \Omega}$ and $\mathbf{x}^{\omega} \in \{0,1\}^{|I|}$, if there exists an ordering, $\omega_1, \ldots, \omega_{|\Omega|}$, on the set of scenarios such that $S(\mathbf{x}^{\omega_1}) \supseteq \cdots \supseteq S(\mathbf{x}^{\omega_{|\Omega|}})$, Algorithm 3 constructs a priority list, $\mathcal{K}$, for the set of scenarios so that $\mathcal{K}$ satisfies constraints (2.14b)–(2.14f) and $(\mathcal{K}, \mathbf{x})$ satisfies constraints (2.13e)–(2.13f). By construction, $\mathcal{K}$ satisfies constraints (2.14b)–(2.14f). It remains to show that $(\mathcal{K}, \mathbf{x})$ satisfies constraints (2.13e)–(2.13f). Consider two scenarios ω, ω', where $\omega \in \mathcal{K}_k,\ \omega' \in \mathcal{K}_{k'},\ k, k' = 1, \ldots, K$. We must show that if $k = k'$, constraint (2.13f) is satisfied, and otherwise, constraint (2.13e) is

Algorithm 3 Construct $\mathcal{K}$ from the nested sets, $S(\mathbf{x}^{\omega_1}) \supseteq \cdots \supseteq S(\mathbf{x}^{\omega_{|\Omega|}})$.

Input: $S(\mathbf{x}^{\omega_1}) \supseteq \cdots \supseteq S(\mathbf{x}^{\omega_{|\Omega|}})$.
Output: $\mathcal{K} = [\mathcal{K}_1, \ldots, \mathcal{K}_K]$, where K is the number of priority levels.

$\mathcal{K}_j \leftarrow \emptyset$, $j = 1, \ldots, |\Omega|$.
$t \leftarrow 1, j \leftarrow 1$.
repeat
  $\mathcal{K}_j \leftarrow \mathcal{K}_j \cup \{\omega_t\}$.
  if $S(\mathbf{x}^{\omega_t}) \setminus S(\mathbf{x}^{\omega_{t+1}}) \neq \emptyset$ **then**
    $j \leftarrow j + 1$.
  end if
  $t \leftarrow t + 1$.
until $t = |\Omega|$
$\mathcal{K}_j \leftarrow \mathcal{K}_j \cup \{\omega_{|\Omega|}\}$. $K \leftarrow j$. $\mathcal{K} \leftarrow [\mathcal{K}_1, \ldots, \mathcal{K}_K]$.

satisfied. If $k = k'$, by construction we must have $S(\mathbf{x}^{\omega}) = S(\mathbf{x}^{\omega'})$ and thus $x_i^{\omega} = x_i^{\omega'},\ i \in I$. If, without loss of generality, $k < k'$, by construction we must have $S(\mathbf{x}^{\omega}) \supset S(\mathbf{x}^{\omega'})$ and thus $x_i^{\omega} \geq x_i^{\omega'},\ i \in I$. Consider the following two propositions.

Proposition 2.5.1. *There exists an optimal solution, $\bar{\mathcal{K}}$, to model (2.14) and a corresponding optimal solution, $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, to model (2.13) with $\bar{\mathbf{x}} = (x_i^{\omega})_{i \in I, \omega \in \Omega}$ such that $\bar{\mathbf{x}}$ satisfies the set of inequalities*

$$\bar{x}_i^{\omega'} \geq \bar{x}_i^{\omega''}, \qquad i \in I, \tag{2.18}$$

for all $(\omega', \omega'') \in \Omega_D$ such that $\omega' \in \Omega_I$.

Proof. Let $\mathcal{K}^*$ be an optimal solution to model (2.14), and $(\mathbf{x}^*, \mathbf{y}^*) = (\mathbf{x}^{*\omega}, \mathbf{y}^{*\omega})_{\omega \in \Omega}$ be the corresponding optimal solution to model (2.13). Given two scenarios, $\omega', \omega'' \in \Omega$, having $S(\mathbf{x}^{*\omega'}) \supseteq S(\mathbf{x}^{*\omega''})$ implies having $x_i^{*\omega'} \geq x_i^{*\omega''},\ i \in I$.

Hence, if for all $(\omega', \omega'') \in \Omega_D$ with $\omega' \in \Omega_I$ we have $S(\mathbf{x}^{*\omega'}) \supseteq S(\mathbf{x}^{*\omega''})$, the proof is complete. Suppose for a pair $(\omega', \omega'') \in \Omega_D$ with $\omega' \in \Omega_I$ we have $S(\mathbf{x}^{*\omega'}) \subset S(\mathbf{x}^{*\omega''})$. Then, we construct a new solution, $\bar{\mathcal{K}}$, to model (2.14), and a corresponding solution, $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) = (\bar{\mathbf{x}}^{\omega}, \bar{\mathbf{y}}^{\omega})_{\omega \in \Omega}$, to model (2.13) such that

(i) $S(\bar{\mathbf{x}}^{\omega}) = S(\mathbf{x}^{*\omega})$, $\omega \in \Omega \setminus \{\omega'\}$, and $S(\bar{\mathbf{x}}^{\omega'}) = S(\mathbf{x}^{*\omega''})$,

(ii) $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ satisfies constraints (2.13b)–(2.13d), $(\bar{\mathcal{K}}, \bar{\mathbf{x}})$ satisfies constraints (2.13e)–(2.13f), and $\bar{\mathcal{K}}$ satisfies constraints (2.14b)–(2.14f), and

(iii) $\bar{\mathcal{K}}$ has no worse objective function value than $\mathcal{K}^*$, i.e., $G(\bar{\mathcal{K}}) \leq G(\mathcal{K}^*)$.

Repeating this argument for all $(\omega', \omega'') \in \Omega_D$ with $\omega' \in \Omega_I$ such that $S(x^{*\omega'}) \subset S(x^{*\omega''})$ proves the proposition. Consider a new solution $(\bar{\mathcal{K}}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$ with $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) = (\bar{\mathbf{x}}^{\omega}, \bar{\mathbf{y}}^{\omega})_{\omega \in \Omega}$ formed as follows. Set $(\bar{\mathbf{x}}^{\omega}, \bar{\mathbf{y}}^{\omega}) = (\mathbf{x}^{*\omega}, \mathbf{y}^{*\omega})$ for $\omega \in \Omega \setminus \{\omega'\}$, and $(\bar{\mathbf{x}}^{\omega'}, \bar{\mathbf{y}}^{\omega'}) = (\mathbf{x}^{*\omega''}, \mathbf{y}^{\omega'}(\mathbf{x}^{*\omega''}))$.

(i) This holds by construction.

(ii) Proving that $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ satisfies constraints (2.13b)–(2.13d) is the same as showing that $(\bar{\mathbf{x}}^{\omega}, \bar{\mathbf{y}}^{\omega}) \in Y^{\omega}$, which holds for scenarios $\omega \in \Omega \setminus \{\omega'\}$ since for these scenarios $(\bar{\mathbf{x}}^{\omega}, \bar{\mathbf{y}}^{\omega}) = (\mathbf{x}^{*\omega}, \mathbf{y}^{*\omega})$. Also, $(\bar{\mathbf{x}}^{\omega'}, \bar{\mathbf{y}}^{\omega'}) \in Y^{\omega'}$, where $(\bar{\mathbf{x}}^{\omega'}, \bar{\mathbf{y}}^{\omega'}) = (\mathbf{x}^{*\omega''}, \mathbf{y}^{\omega'}(\mathbf{x}^{*\omega''}))$, by the following observations:

- $\mathbf{x}^{*\omega''} \in X^{\omega''}$ and scenario ω' dominates scenario ω''. Thus, $\mathbf{x}^{*\omega''} \in X^{\omega'}$.

- $\mathbf{y}^{\omega'}(\mathbf{x}^{*\omega''})$ is the optimizer of model (2.17) for scenario ω' and for $\mathbf{x} = \mathbf{x}^{*\omega''}$, and thus $(\mathbf{x}^{\omega''}, \mathbf{y}^{\omega'}(\mathbf{x}^{*\omega''})) \in Y^{\omega''}$.

By constraints (2.2e)–(2.2f), there exists an ordering, $\omega_1, \ldots, \omega_{|\Omega|}$, on the set of scenarios so that $S(\mathbf{x}^{*\omega_1}) \supseteq S(\mathbf{x}^{*\omega_2}) \supseteq \cdots \supseteq S(\mathbf{x}^{*\omega_{|\Omega|}})$. And, by item (i) above we have this nested structure for $S(\bar{\mathbf{x}}^{\omega})_{\omega\in\Omega}$ as well, possibly after reordering the scenarios. Algorithm 3 constructs $\bar{\mathcal{K}}$ so that $(\bar{\mathcal{K}}, \bar{\mathbf{x}})$ satisfies constraints (2.13e)–(2.13f), and $\bar{\mathcal{K}}$ satisfies constraints (2.14b)–(2.14f).

(iii) $G(\mathcal{K}^*) = \sum_{\omega\in\Omega} q^{\omega} h^{\omega}(\mathbf{x}^{*\omega})$, and $G(\bar{\mathcal{K}}) = \sum_{\omega\in\Omega} q^{\omega} h^{\omega}(\bar{\mathbf{x}}^{\omega})$. Since $(\bar{\mathbf{x}}^{\omega}, \bar{\mathbf{y}}^{\omega}) = (\mathbf{x}^{*\omega}, \mathbf{y}^{*\omega})$, $\omega \in \Omega \setminus \{\omega'\}$, we have $G(\bar{\mathcal{K}}) - G(\mathcal{K}^*) = q^{\omega'}[h^{\omega'}(\bar{\mathbf{x}}^{\omega'}) - h^{\omega'}(\mathbf{x}^{*\omega'})] = q^{\omega'}[h^{\omega'}(\mathbf{x}^{*\omega''}) - h^{\omega'}(\mathbf{x}^{*\omega'})]$. Hence, it suffices to show that $h^{\omega'}(\mathbf{x}^{*\omega''}) \leq h^{\omega'}(\mathbf{x}^{*\omega'})$. Since $\mathbf{x}^{*\omega''} \in X^{\omega''}$ and ω' dominates ω'', we have that $\mathbf{x}^{*\omega''} \in X^{\omega'}$. We also have $S(\mathbf{x}^{*\omega''}) \supset S(\mathbf{x}^{*\omega'})$. Finally, since ω' is improving, we must have $h^{\omega'}(\mathbf{x}^{*\omega''}) \leq h^{\omega'}(\mathbf{x}^{*\omega'})$.

□

Proposition 2.5.2. *There exists an optimal solution, $\bar{\mathcal{K}}$, to model (2.14) and a corresponding optimal solution, $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, to model (2.13) with $\bar{\mathbf{x}} = (x_i^{\omega})_{i\in I, \omega\in\Omega}$ such that $\bar{\mathbf{x}}$ satisfies the set of inequalities*

$$\bar{x}_i^{\omega} \geq \bar{x}_{i'}^{*\omega}, \qquad \omega \in \Omega, \tag{2.19}$$

for all $(i, i') \in I_D$.

Proof. Let $\mathcal{K}^*$ be an optimal solution to model (2.14), and $(\mathbf{x}^*, \mathbf{y}^*) = (\mathbf{x}^{*\omega}, \mathbf{y}^{*\omega})_{\omega\in\Omega}$ be the corresponding optimal solution to model (2.13). Suppose there exists a pair $(i, i') \in I_D$ for which $\mathbf{x}^*$ does not satisfy the set of inequalities in (2.19). Then, we construct a new solution, $\bar{\mathcal{K}}$, to model (2.14), and a corresponding solution, $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) = (\bar{\mathbf{x}}^{\omega}, \bar{\mathbf{y}}^{\omega})_{\omega\in\Omega}$, to model (2.13) such that

(i) $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ satisfies constraints (2.13b)–(2.13d), $(\bar{\mathcal{K}}, \bar{\mathbf{x}})$ satisfies constraints (2.13e)–(2.13f), and $\bar{\mathcal{K}}$ satisfies constraints (2.14b)–(2.14f),

(ii) $\bar{\mathcal{K}}$ has no worse objective function value than $\mathcal{K}^*$, i.e., $G(\bar{\mathcal{K}}) \leq G(\mathcal{K}^*)$, and

(iii) $\bar{\mathbf{x}}$ satisfies the set of inequalities in (2.19) for pair (i, i').

Repeating this argument for all pairs $(i, i') \in I_D$ for which $\mathbf{x}^*$ does not satisfy the set of inequalities in (2.19) proves the proposition.

Given two scenarios, $\omega', \omega'' \in \Omega$, by constraints (2.13e)–(2.13f) we have either $S(\mathbf{x}^{*\omega'}) \supseteq S(\mathbf{x}^{*\omega''})$ or $S(\mathbf{x}^{*\omega'}) \subset S(\mathbf{x}^{*\omega''})$. Thus, there exists an ordering, $\omega_1, \ldots, \omega_{|\Omega|}$, on the set of scenarios so that $S(\mathbf{x}^{*\omega_1}) \supseteq S(\mathbf{x}^{*\omega_2}) \supseteq \cdots \supseteq S(\mathbf{x}^{*\omega_{|\Omega|}})$. Since the set of inequalities in (2.19) are not satisfied for pair (i, i'), there must exist a nonempty subset of scenarios under which activity i is not selected, but activity i' is selected. Due to the nested nature of the sets $S(\mathbf{x}^{*\omega_j})$, $j = 1, \ldots, |\Omega|$, this nonconforming subset of scenarios must constitute a subset of the ordering $\omega_1, \omega_2, \ldots, \omega_{|\Omega|}$, without an intermediary conforming scenario that satisfies the set of inequalities in (2.19).

In other words, there must exist t, t' with $0 \leq t < t' \leq |\Omega|$ such that $i, i' \in S(\mathbf{x}^{*\omega_j})$ for $j = 1, \ldots, t$, $i \notin S(\mathbf{x}^{*\omega_j})$, $i' \in S(\mathbf{x}^{*\omega_j})$ for $j = t+1, \ldots, t'$, and $i, i' \notin S(\mathbf{x}^{*\omega_j})$ for $j = t'+1, \ldots, |\Omega|$. Consider a new solution $(\bar{\mathcal{K}}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$ with $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) = (\bar{\mathbf{x}}^{\omega}, \bar{\mathbf{y}}^{\omega})_{\omega \in \Omega}$ constructed as follows.

$$(\bar{\mathbf{x}}^{\omega}, \bar{\mathbf{y}}^{\omega}) = \begin{cases} (\mathbf{x}^{*\omega}, \mathbf{y}^{*\omega}) & \text{for } \omega \in \Omega \setminus \{\omega_{t+1}, \ldots, \omega_{t'}\}, \\ (\mathbf{x}^{*\omega}_{(i,i')}, \mathbf{y}^{\omega}(\mathbf{x}^{*\omega}_{(i,i')})) & \text{for } \omega \in \{\omega_{t+1}, \ldots, \omega_{t'}\}. \end{cases}$$

(i) Proving that $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ satisfies constraints (2.13b)–(2.13d), is the same as showing that $(\bar{\mathbf{x}}^{\omega}, \bar{\mathbf{y}}^{\omega}) \in Y^{\omega}$, which holds for scenarios $\omega \in \Omega \setminus \{\omega_{t+1}, \ldots, \omega_{t'}\}$ since for these scenarios $(\bar{\mathbf{x}}^{\omega}, \bar{\mathbf{y}}^{\omega}) = (\mathbf{x}^{*\omega}, \mathbf{y}^{*\omega})$. For scenarios $\omega \in \{\omega_{t+1}, \ldots, \omega_{t'}\}$, $(\bar{\mathbf{x}}^{\omega}, \bar{\mathbf{y}}^{\omega}) \in Y^{\omega}$, where $(\bar{\mathbf{x}}^{\omega}, \bar{\mathbf{y}}^{\omega}) = (\mathbf{x}^{*\omega}_{(i,i')}, \mathbf{y}^{\omega}(\mathbf{x}^{*\omega}_{(i,i')}))$, by the following observations:

- $\mathbf{x}^{*\omega} \in X^{\omega}$, $x_i^{*\omega} = 0$, $x_{i'}^{*\omega} = 1$, and i dominates i'. Thus, $\mathbf{x}^{*\omega}_{(i,i')} \in X^{\omega}$.
- $\mathbf{y}^{\omega}(\mathbf{x}^{*\omega}_{(i,i')}))$ is the optimizer of model (2.17) for scenario ω and for $\mathbf{x} = \mathbf{x}^{*\omega}_{(i,i')}$, and thus $(\mathbf{x}^{*\omega}_{(i,i')}, \mathbf{y}^{\omega}(\mathbf{x}^{*\omega}_{(i,i')})) \in Y^{\omega}$.

The new solution, $(\bar{\mathbf{x}}^{\omega})_{\omega \in \Omega}$, differs from the old one, $(\mathbf{x}^{*\omega})_{\omega \in \Omega}$, only in scenarios $\omega \in \{\omega_{t+1}, \ldots, \omega_{t'}\}$. In these scenarios, we set $\bar{x}_i^{\omega} = 1$ and $\bar{x}_{i'}^{\omega} = 0$. This preserves the nestedness of the sets $S(\bar{\mathbf{x}}^{\omega})_{\omega \in \Omega}$. In fact, this preserves the ordering of the scenarios as well. Algorithm 3 constructs $\bar{\mathcal{K}}$ so that $(\bar{\mathcal{K}}, \bar{\mathbf{x}})$ satisfies constraints (2.13e)–(2.13f), and $\bar{\mathcal{K}}$ satisfies constraints (2.14b)–(2.14f).

(ii) $G(\mathcal{K}^*) = \sum_{\omega\in\Omega} q^\omega h^\omega(\mathbf{x}^{*\omega})$, and $G(\bar{\mathcal{K}}) = \sum_{\omega\in\Omega} q^\omega h^\omega(\bar{\mathbf{x}}^\omega)$. Since $(\bar{\mathbf{x}}^\omega, \bar{\mathbf{y}}^\omega)$ $= (\mathbf{x}^{*\omega}, \mathbf{y}^{*\omega})$, $\omega \in \Omega \setminus \{\omega_{t+1}, \ldots, \omega_{t'}\}$, we have $G(\bar{\mathcal{K}}) - G(\mathcal{K}^*) =$ $\sum_{\omega\in\{\omega_{t+1},\ldots,\omega_{t'}\}} q^\omega[h^\omega(\bar{\mathbf{x}}^\omega) - h^\omega(\mathbf{x}^{*\omega})] = \sum_{\omega\in\{\omega_{t+1},\ldots,\omega_{t'}\}} q^\omega[h^\omega(\mathbf{x}^{*\omega}_{(i,i')}) - h^\omega(\mathbf{x}^{*\omega})]$. Hence, it suffices to show that $h^\omega(\mathbf{x}^{*\omega}_{(i,i')}) \leq h^\omega(\mathbf{x}^{*\omega})$ for $\omega \in \{\omega_{t+1}, \ldots, \omega_{t'}\}$. Since $\mathbf{x}^{*\omega} \in X^\omega$, $x^{*\omega}_{i'} = 1$, $x^{*\omega}_i = 0$, and i dominates i', we must have $h^\omega(\mathbf{x}^{*\omega}_{(i,i')}) \leq h^\omega(\mathbf{x}^{*\omega})$.

(iii) By construction, we have $i, i' \in S(\bar{\mathbf{x}}^{\omega_j})$ for $j = 1, \ldots, t$, $i \in S(\bar{\mathbf{x}}^{\omega_j})$, $i' \notin S(\bar{\mathbf{x}}^{\omega_j})$ for $j = t+1, \ldots, t'$, and $i, i' \notin S(\bar{\mathbf{x}}^{\omega_j})$ for $j = t'+1, \ldots, |\Omega|$. Hence, $x^\omega_i \geq x^\omega_{i'}$, $\omega \in \Omega$.

□

Due to Theorem 2.4.1, an immediate corollary to both propositions is that the inequalities in (2.18) and (2.19) also hold for models (2.3) and (2.2).

Corollary 2.5.3. *There exists an optimal solution, $\bar{\mathcal{L}}$, to model (2.3) and a corresponding optimal solution, $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, to model (2.2) with $\bar{\mathbf{x}} = (x^\omega_i)_{i\in I, \omega\in\Omega}$ such that $\bar{\mathbf{x}}$ satisfies the set of inequalities*

$$\bar{x}^{\omega'}_i \geq \bar{x}^{\omega''}_i, \qquad i \in I, \tag{2.20}$$

for all $(\omega', \omega'') \in \Omega_D$ such that $\omega' \in \Omega_I$. Further, there exists an optimal solution, $\bar{\mathcal{L}}$, to model (2.3) and a corresponding optimal solution, $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, to model (2.2) with $\bar{\mathbf{x}} = (x^\omega_i)_{i\in I, \omega\in\Omega}$ such that $\bar{\mathbf{x}}$ satisfies the set of inequalities

$$\bar{x}^\omega_i \geq \bar{x}^{*\omega}_{i'}, \qquad \omega \in \Omega, \tag{2.21}$$

for all $(i, i') \in I_D$.

We note that we do not use the term *valid inequality* for the cutting planes (2.18) and (2.19). The reason is that the cutting planes (2.18) and (2.19) may rule out some of the feasible region, even some of the optimal solutions. But, what we guarantee is that there remains at least one optimal solution which is feasible. These cutting planes are more powerful than valid inequalities in helping improve the solution times because we allow them to get rid of not only integer infeasible solutions, but also integer feasible solutions, even some optimal solutions. Israeli and Wood [17] refer to such inequalities as super-valid inequalities.

Another good point about the cutting planes (2.18) and (2.19) is that we can extract them from the problem data before we solve the problem. They are globally valid, i.e., valid for all nodes of the branch-and-bound tree. A naive approach is to add them to the problem definition initially. However, typically most of these cutting planes do not become tight in the LP-relaxation solution. In order to avoid increasing the problem size unnecessarily, we place them in a cut pool and add them only as necessary, i.e., we add only those that are infeasible to the LP-relaxation solution. This is the approach we use employing cutting planes (2.18) for model (2.4) and cutting planes (2.19) for model (2.15). There is even a simpler way to use cutting planes (2.18) for model (2.15) and cutting planes (2.19) for model (2.4). If we have cutting planes of the form $x_i^{\omega'} \geq x_i^{\omega''},\ i \in I$, we can fix variable $s_{\omega'\omega''} = 1$ in model (2.15) . Then, constraint (2.15d) for pair (ω', ω'') reads $x_i^{\omega'} \geq x_i^{\omega''},\ i \in I$. Similarly, if we have cutting planes of the form $x_i^{\omega} \geq x_{i'}^{\omega},\ \omega \in \Omega$, we can fix variable $s_{ii'} = 1$

in model (2.4) . Then, constraint (2.4d) for pair (i, i') reads $x_i^\omega \geq x_{i'}^\omega,\ \omega \in \Omega$. Hence, we can both decrease the solution time and obtain a tighter LP relaxation using these ideas.

Table 2.4 shows the effect of cutting planes (2.18) and (2.19). The table is to be interpreted as Table 2.3. It is appropriate to compare the two tables cell by cell. It is expected that using cutting planes in small problem instances increases solution times, the reason being that small problems are already simple to solve. Thus, there is no need to make them larger with the hope of improving their LP-relaxation values. This notion is consistent with the results for 3 scenarios. As the number of scenarios grows, the benefit of the cutting planes is amplified yielding ratios of CPU times of 5, 10, 20, even 160. For example, in the instances with 10 items and 54 scenarios, and 15 items and 27 scenarios, using cutting planes improves the solution time of model (2.15) almost by a factor of 160. We observe the same improving effect in instances in which we only spend an hour. We can solve some of the instances to optimality in an hour that we could not solve in this time without using the cutting planes. Similarly we can obtain one-hour-optimality-gaps for several instances that we could not without using the cutting planes.

Table 2.4: The effect of cutting planes. Values are to be interpreted as in Table 2.3.

# scenarios	# items - # dimensions									
	10-10	15-10	20-10	28-10	39-5	50-5	60-5	70-5	80-5	90-5
3	0	0	0	2	55	120	29	23	66	38
	0	0	0	0	0	1	0	0	0	0
9	2	6	32	144	6238	185885	7418	32344	4080	2371
	1	2	4	6	82	2446	22	73	14	11
18	5	27	341	2306	0.85%	1.89%	1.70%	0.77%	0.66%	0.36%
	7	22	140	273	0.11 %	0.18%	0.23%	0%	0.14%	0%
27	12	482	2894	143257	3.77%	3.10%	2.20%	0.78%	0.92%	0.54%
	23	1144	3135	47103	2.36%	1.76%	1.79%	0.65%	0.73%	0.44%
54	60	2629	1.33%	1.43%	4.77%	2.97%	2.17%	1.35%	1.19%	0.82%
	1105	45291	2.37%	1.25%	4.72%	2.83%	2.17%	1.36%	1.18%	0.81%
81	243	3.00%	4.02%	2.15%	4.85%		2.56%	1.16%	1.18%	0.66%
	14037	4.60%	4.35%	2.15%			2.64%	1.16%		0.66%

Chapter 3

Branch-and-Price Approach for the Prioritized Multidimensional Knapsack Problem

A branch-and-price algorithm relies on performing a branch-and-bound algorithm on what is typically a column-based reformulation of an original problem. A column-based formulation often has a tighter LP-relaxation, hence yielding in a smaller branch-and-bound tree. On the negative side, a column-based formulation has an exponential number of columns that are impractical to handle simultaneously. Delayed column generation is an effective method to handle this drawback. We start with a small number of columns that are sufficient to form a feasible formulation and add additional columns as necessary, until no more columns that can improve the LP-relaxation remain. For more background on branch-and-price algorithms see the surveys by Barnhart et al. [2] and Lübbecke and Desrosiers [26], and the book by Desaulniers et al. [11].

3.1 A Serial Algorithm

In this section, we develop a branch-and-price decomposition algorithm for model (2.12). Specifically, we construct a column-based reformulation of

model (2.12), develop two branching strategies, and introduce a tabu-search heuristic in order to speed up the overall algorithm. The key elements in a branch-and-price algorithm are the reformulation process and the branching rules since both affect the number of nodes in the branch-and-bound tree. The former is important as the tighter the LP relaxation of the column-based reformulation, the fewer the number of nodes in the tree. There are various reformulation strategies for stochastic integer programs, which can be classified as scenario decomposition [e.g., 27], nodal decomposition [e.g., 42], complete decomposition, and geographical decomposition [e.g., 40, 41]. The first three strategies use the fact that the problem decomposes into scenario subproblems, while the last uses a loosely coupled structure of an original deterministic problem. We prefer complete decomposition, since the resulting pricing problem becomes the same as the original deterministic problem before applying prioritization. Hence, any specific algorithm developed for the original problem can be used as the pricing-problem-solver in our branch-and-price implementation.

Branching rules are developed to guarantee integrality of the original decision variables. The *de facto* strategy in branching rules is to preserve the pricing problem structure even after branching. We develop two branching strategies; one preserves the pricing problem structure, the other does not, and we show that the latter outperforms the former significantly in terms of the overall running time.

3.1.1 Column-based Reformulation

In complete decomposition for a two-stage stochastic integer program, the second-stage constraints are handled by the pricing problem, and all other constraints are kept in the master problem. Specifically, given $\omega \in \Omega$, let $\mathbf{x}^{\omega j} = (x_1^{\omega j}, \ldots, x_{|I|}^{\omega j}),\ j \in K_\omega$, enumerate feasible solutions of the constraint set

$$\left\{ (x_i)_{i \in I} : \sum_{i \in I} c_{it}^{\omega} x_i \leq b_t^{\omega},\ t \in T, x_i \in \{0,1\}, i \in I \right\}.$$

In formulation (2.12), setting $x_i^{\omega} = \sum_{j \in K_\omega} \lambda^{\omega j} x_i^{\omega j}$, where $\lambda^{\omega j} \in \{0,1\}$ and $\sum_{j \in K_\omega} \lambda^{\omega j} = 1,\ \omega \in \Omega$, we obtain the following column-based reformulation:

$$\max_{\mathbf{s},\,\lambda} \quad \sum_{\omega \in \Omega} q^{\omega} \sum_{i \in I} a_i^{\omega} \sum_{j \in K_\omega} x_i^{\omega j} \lambda^{\omega j} \tag{3.1a}$$

$$\text{s.t.} \quad \sum_{j \in K_\omega} x_i^{\omega j} \lambda^{\omega j} \geq \sum_{j \in K_\omega} x_{i'}^{\omega j} \lambda^{\omega j} + s_{ii'} - 1, \quad i \neq i',\ i, i' \in I,\ \omega \in \Omega, \tag{3.1b}$$

$$\sum_{j \in K_\omega} \lambda^{\omega j} = 1, \qquad \omega \in \Omega, \tag{3.1c}$$

$$\lambda^{\omega j} \in \{0,1\}, \qquad j \in K_\omega,\ \omega \in \Omega, \tag{3.1d}$$

$$(2.12b),\ (2.12c). \tag{3.1e}$$

Model (3.1) is referred to as the master program (MP) and its LP relaxation as the master linear program (MLP). To solve the MP, we apply a branch-and-bound algorithm, using the MLP. Including all elements of the set

$K_\omega,\ \omega \in \Omega$, in the MLP usually leads to a very large problem. We start with a partial set $K'_\omega \subset K_\omega,\ \omega \in \Omega$, making sure that the new restricted master linear program (RMLP) is feasible. Given a solution to the RMLP, along with its dual variables, we then, search through each set K_ω to find the element with the largest positive reduced cost, by solving a pricing problem. If the largest reduced cost is positive, we update K'_ω, adding the associated column, and we resolve the RMLP. We repeat this process until the largest reduced cost over all $K_\omega, \omega \in \Omega$, is non-positive. The termination criterion means that adding further columns does not improve the optimal value of RMLP. We now define both RMLP and the pricing problems:

$$\max_{\mathbf{s},\lambda} \quad \sum_{\omega\in\Omega}\sum_{j\in K'_\omega}\sum_{i\in I} q^\omega a_i^\omega x_i^{\omega j}\lambda^{\omega j} \tag{3.2a}$$

$$\text{s.t.} \quad \sum_{j\in K'_\omega}(x_i^{\omega j} - x_{i'}^{\omega j})\lambda^{\omega j} \geq s_{ii'} - 1, \qquad i \neq i',\ \ i, i' \in I,\ \omega \in \Omega, \tag{3.2b}$$

$$\sum_{j\in K'_\omega}\lambda^{\omega j} = 1, \qquad \omega \in \Omega, \tag{3.2c}$$

$$0 \leq \lambda^{\omega j} \leq 1, \qquad j \in K'_\omega,\ \omega \in \Omega, \tag{3.2d}$$

$$0 \leq s_{ii'} \leq 1, \qquad i \neq i',\ \ i, i' \in I, \tag{3.2e}$$

$$(2.12b). \tag{3.2f}$$

Let $\pi_{ii'}^\omega$ and α^ω be the dual variables corresponding to the constraints (3.2b) and (3.2c), respectively. The pricing problem for scenario ω (i.e., set K_ω) is:

$$\max_{\mathbf{x}} \quad \sum_{i \in I} \left(q^{\omega} a_i^{\omega} + \sum_{i' \in I, i' \neq i} (\pi_{i'i}^{\omega} - \pi_{ii'}^{\omega}) \right) x_i \tag{3.3a}$$

$$\text{s.t.} \quad \sum_{i \in I} c_{it}^{\omega} x_i \leq b_t^{\omega}, \qquad t \in T, \tag{3.3b}$$

$$x_i \in \{0, 1\}, \qquad i \in I. \tag{3.3c}$$

After the column generation process has terminated, i.e., no more columns with positive reduced cost remain, the solution to the MLP may not be integer. An immediate idea is to solve the problem at hand to IP-optimality with the existing columns. But, this gives only a heuristic solution to the MP. The reason is that whenever we branch we need to check whether some of the excluded columns in the parent node can now enter the basis as their reduced cost may now be positive due to new branching restrictions. In the next section, we develop two branching rules that guarantee we find an IP-optimal solution.

3.1.2 Branching

Branching on the convexity variables, i.e., the λ variables in model (3.1), may not be a good practice. It may not be obvious how to fix the columns corresponding to a component of λ to zero or one, in the pricing problem. To avoid this complication we branch on the original variables of model (2.12). Typically, in branch-and-price applications, branching on the original variables is done in such a way that the structure of the pricing problem is not destroyed. In our application we consider two branching rules; one destroys

the pricing problem structure, the other does not; and we show that the former outperforms the latter remarkably in terms of running time. A simple consequence is that trading a nicely structured pricing problem for a smaller branch-and-bound tree can sometimes be desirable. For both branching rules, we need the following proposition.

Proposition 3.1.1. *Relaxing constraint (2.12c) to be continuous in model (2.12) does not change its optimal value. Furthermore, given any feasible solution* $(\mathbf{x}, \mathbf{s})$ *to model (2.12) with the continuous relaxation of constraint (2.12c), we can find integral* $(\mathbf{x}, \bar{\mathbf{s}})$ *that are feasible and have the same objective function value.*

Proof. We first prove the second part. We have integer $\mathbf{x}$ and (possibly) non-integer $\mathbf{s}$ that satisfy the constraints of model (2.12), except that constraint (2.12c) is relaxed. We take $\mathbf{x}$ as given, and find integer $\bar{\mathbf{s}}$'s that form a feasible solution to model (2.12). Since the latter variables do not appear in the objective function, doing so is sufficient to prove the result. Given $\mathbf{x}$ and a pair of items, $(i,\, i')$, there are two possibilities:

- $\exists$ at least one $\omega \in \Omega$ such that $x_i^\omega < x_{i'}^\omega$,
- $\forall \omega \in \Omega,\ x_i^\omega \geq x_{i'}^\omega$.

We further analyze the first case together with constraints (2.12b), (2.12d) and the continuous relaxation of (2.12c). Noting that $\mathbf{x}$ is binary, whenever $x_i^\omega < x_{i'}^\omega$ for some $\omega \in \Omega$, by constraint (2.12d) for pair $(i,\, i')$ we

must have $s_{ii'} = 0$. Then, by constraints (2.12b) and the relaxation of (2.12c), we have $s_{i'i} = 1$. Finally, considering constraint (2.12d) for pair (i', i), we have that $x_i^\omega \leq x_{i'}^\omega,\ \forall \omega \in \Omega$. Hence, given a pair of items (i, i') we have two possibilities:

- $\forall \omega \in \Omega,\ x_i^\omega \leq x_{i'}^\omega$,
- $\forall \omega \in \Omega,\ x_i^\omega \geq x_{i'}^\omega$.

Setting $\bar{s}_{ii'} = 0,\ \bar{s}_{i'i} = 1$ in the first case, and $\bar{s}_{ii'} = 1,\ \bar{s}_{i'i} = 0$ otherwise, gives us a feasible and integer $\bar{\mathbf{s}}$. We note that a tie can occur, i.e., both of the cases may be satisfied. In this case, although either of the settings work, without loss of generality, we set the lower indexed item to precede the upper indexed one, i.e., $\bar{s}_{ii'} = 1,\ \bar{s}_{i'i} = 0$ for $i < i'$.

By the second part of the proposition, the relaxed model cannot have a better optimal objective function value than the original model. Thus, both the relaxed and the non-relaxed models have the same optimal value.

□

Selection Branching

Following the result of Proposition 3.1.1 at any node of the branch-and-bound tree, after the column generation process has terminated, we check whether the resulting $\mathbf{x}$ solution is integral. If so, we fathom the node deeming it feasible. If not, we choose a pair (i^*, ω^*) such that $0 < x_{i^*}^{\omega^*} < 1$. On one branch, we fix $x_{i^*}^{\omega^*}$

to 0, on the other, to 1. To do so, we partition the columns corresponding to set K'_{ω^*} in model (3.2) into $\underline{K'}_{\omega^*}$ and $\overline{K'}_{\omega^*}$ such that $\underline{K'}_{\omega^*} = \{j \in K'_{\omega^*} |\ x_{i^*}^{\omega^* j} = 0\}$, and $\overline{K'}_{\omega^*} = \{j \in K'_{\omega^*} |\ x_{i^*}^{\omega j} = 1\}$. On one branch, we fix $\lambda^{\omega^* j} = 0$, for $j \in \underline{K'}_{\omega^*}$, and on the other branch we fix $\lambda^{\omega^* j} = 0$, for $j \in \overline{K'}_{\omega^*}$. To make sure that pricing problems do not produce such columns, we add constraints $x_{i^*} = 1$ and $x_{i^*} = 0$, correspondingly, to the pricing problem indexed by ω^*.

The above-detailed branching, what we call *selection branching*, is valid as it satisfies the following two criteria: First, there are finitely many branching objects, i.e., at most $|I| \times |\Omega|$. Second, given any fractional solution $\mathbf{x}$, we can find a corresponding branching object, i.e., a pair (i^*, ω^*) with $0 < x_{i^*}^{\omega^*} < 1$, such that the fractional solution $\mathbf{x}$ is not feasible to either of the two sibling branches.

Selection branching does not destroy the structure of the pricing problems. Fixing some variables in RCASP just decrements the number of items in the problem by one.

Precedence Branching

Our second branching rule branches on the precedence, i.e., $\mathbf{s}$, variables. We first note that this branching is not valid in model (2.12). That is, if we relax the $\mathbf{x}$ variables to be continuous, and require only $\mathbf{s}$ variables to be integer, we may obtain a non-integral optimal $\mathbf{x}$ solution. This is mainly due to the underlying deterministic problem. Consider, for instance, a knapsack problem with two identical items and two scenarios. In both scenarios, the

items have identical benefits and weights, and we have budget to accommodate only one and a half items. Then, the optimal solution to model (2.12) with $\mathbf{x}$ variables relaxed is to have one and a half items in both scenarios, e.g., $x_1 = 1,\ x_2 = 1/2,\ s_{12} = 1,\ s_{21} = 0$. One explanation as to why this branching rule works in our branch-and-price algorithm is that we already have integral $\mathbf{x}$ variables (columns) in the RMLP. What remains to be shown for precedence branching is that the convex combination of these columns is also integral.

The details of the precedence branching scheme follow: Upon convergence of column generation, we check the columns with positive solution value. If we can find a pair of items $(i,\ i')$, $i,\ i' \in I$, a pair of (possibly identical) scenarios $(\omega,\ \omega')$, $\omega_1,\ \omega_2 \in \Omega$, and a pair of columns $(j,\ j')$, $j \in K_\omega,\ j' \in K_{\omega'}$, such that $x_i^{\omega\, j} = 1,\ x_i^{\omega'\, j'} = 0$ and $x_{i'}^{\omega\, j} = 0,\ x_{i'}^{\omega'\, j'} = 1$, we branch on $s_{ii'}$. As in selection branching, branching on the original variables require translation of ideas between the original formulation and the column-based re-formulation. On one branch, we set $s_{ii'} = 1,\ s_{i'i} = 0$, fix to 0 all columns such that item i has coefficient 0 and item i' has coefficient 1, and add the constraint $x_i \geq x_{i'}$ to all pricing problems; and we do just the opposite on the other branch. If we cannot find such pairs of items and scenarios, we already have an integral $\mathbf{x}$ solution, and based on Proposition 3.1.1 we fathom the node.

Although it may seem complicated to implement precedence branching, it is quite simple. We pool all nonzero-solution columns, disregarding to which scenario they belong. We then check all possible pairs of columns in the pool. Given a pair of columns, if the difference of the set of items having coefficient 1

in the first column from the set of items having coefficient 1 in the second, and the same difference for the second column from the first are both nonempty, we have such a pair. That is, if: (a) item i has coefficient 1 in the first column and coefficient 0 in the second, and (b) item i' has coefficient 1 in the second column and coefficient 0 in the first, then, (i, i') constitutes such pair. In our implementation, for each distinct item pair we count the number of times these two items are involved in such occurrences, and choose the pair having the highest frequency as a branching object.

Before proving the validity of precedence branching we, set notation and state an assumption. Given a vector $\mathbf{x} \in \{0,1\}^{|I|}$, let $S(\mathbf{x})$ denote the set of selected items, i.e., $S(\mathbf{x}) = \{i \in I \,|\, x_i = 1\}$; and, let $P^\omega(\mathbf{x})$ denote the total profit of these items under scenario $\omega \in \Omega$, i.e., $P^\omega(\mathbf{x}) = \sum_{i \in S(\mathbf{x})} a_i^\omega$. With this notation, $S(\mathbf{x}^{\omega j})$ denotes the set of items having coefficient 1 in column j of scenario ω, and $P(\mathbf{x}^{\omega j})$ denotes the total profit of these items.

A1: Given two vectors $\mathbf{x}, \bar{\mathbf{x}} \in \{0,1\}^{|I|}$ such that $S(\mathbf{x}) \subset S(\bar{\mathbf{x}})$, we have $P^\omega(\mathbf{x}) \neq P^\omega(\bar{\mathbf{x}}),\ \forall \omega \in \Omega$.

Theorem 3.1.2. *If A1 holds, precedence branching is a valid branching rule, i.e.,*

- *the branch-and-bound algorithm terminates in a finite number of steps;*
- *a current solution is not feasible to any of the sibling nodes; and,*
- *the branching rule finds a branching object if and only if the current* $\mathbf{x}$ *solution is fractional.*

Proof.

- There are finitely many branching objects, i.e., $|I| \cdot (|I| - 1)/2$.
- The second condition holds by the following observation. Suppose we have branched on pair $(i,\, i')$. In one of the branches, we add constraint $x_i \geq x_{i'}$ to all of the pricing problems, and fix to 0 all columns in the RMLP that have coefficient 0 for item i and 1 for item i'. Thus, after the column generation is terminated it is not possible to find a column j' such that $x_i^{\omega\, j'} = 0,\ x_{i'}^{\omega\, j'} = 1$ for some $w \in \Omega$. Hence, it is not possible to find a pair of columns $(j,\, j')$ such that (i, i') qualifies as a branching object again. We can repeat this argument in the other branch since we add the constraint $x_{i'} \geq x_i$ to the pricing problem.
- If the current $\mathbf{x}$ solution is not fractional, we must have only one column per scenario that has nonzero solution value. That is, only one of the columns must have solution value 1, and the others must have 0. This holds because of constraint (3.2c). Having more than one nonzero column makes one of the x_i^ω (where $x_i^\omega = \sum_{j \in K'_\omega} \lambda^{\omega\, j} x_i^{\omega\, j}$) fractional, as each column is distinct and each element of the columns is zero or one. Let $\omega_1 j_1, \ldots, \omega_{|\Omega|} j_{|\Omega|}$ index these non-zero columns. By constraint (3.2b) we must have $S(\mathbf{x}^{\omega_1 j_1}) \subseteq \cdots \subseteq S(\mathbf{x}^{\omega_{|\Omega|} j_{|\Omega|}})$, possibly after reordering the scenarios. To prove this claim, suppose the opposite. Then, there must exist two columns $\omega_m j_m$ and $\omega_n j_n$ such that $S(\mathbf{x}^{\omega_m j_m}) \setminus S(\mathbf{x}^{\omega_n j_n}) \neq \emptyset$ and

$S(\mathbf{x}^{\omega_n j_n}) \setminus S(\mathbf{x}^{\omega_m j_m}) \neq \emptyset$. But then, for item pair $(i,\, i')$, $i \in S(\mathbf{x}^{\omega_m j_m}) \setminus S(\mathbf{x}^{\omega_n j_n})$, $i' \in S(\mathbf{x}^{\omega_n j_n}) \setminus S(\mathbf{x}^{\omega_m j_m})$, constraint (3.2b) is violated. This nestedness of the sets implies that for each pair of scenarios, the set of items selected in one of the scenarios is a subset of the other, leaving no possibility for the branching rule to find a branching object.

We now prove the second part. If we cannot find a branching object, we can order nonzero-solution columns such that $S(\mathbf{x}^{\omega_1 j_1}) \subseteq \cdots \subseteq S(\mathbf{x}^{\omega_k j_k})$. As there is at least one nonzero-solution column per scenario, we have $k \geq |\Omega|$. If $k = |\Omega|$, the proof is complete since the trivial convex combination of only one integral column is integral. We show that $k > |\Omega|$ leads to a contradiction. If $k > |\Omega|$, we must have two nonzero-solution columns ωj_m and ωj_n for some scenario $\omega \in \Omega$ such that, without loss of generality, $S(\mathbf{x}^{\omega j_m}) \subset S(\mathbf{x}^{\omega j_n})$. And, A1 ensures, without loss of generality, that $P(\mathbf{x}^{\omega j_m}) < P(\mathbf{x}^{\omega j_n})$. In this case, we form a new solution by changing only $\lambda^{\omega j_n}$ and $\lambda^{\omega j_m}$ variables. We set the value of $\lambda^{\omega j_n}$ to $\lambda^{\omega j_n} + \lambda^{\omega j_m}$, and set the value of $\lambda^{\omega j_m}$ to 0. This new solution is feasible to model (3.2) and has better objective function value, contradicting the optimality of the current solution.

□

3.1.3 Primal Heuristic

We develop a primal heuristic based on a tabu-search approach [e.g., 14]. Efficient heuristic methods help terminate branch-and-bound based al-

gorithms earlier by improving the upper bound directly, improving the lower bound indirectly when logical fixing, variable fixing, and/or reduced-cost fixing is applied, and pruning more nodes based on bound-checking.

Heuristic and meta-heuristic algorithms based on local search start with a given solution and move to its neighbors, hoping to visit solutions with better objective function values. The initial solution can be chosen among one of the good solutions, or it can be random. The latter is often preferred especially if the heuristic may become stuck in a local optima. Sometimes, a simple local search is performed instead of more complicated ones; but the search is repeated many times starting from randomly constructed initial solutions [e.g., 12].

Two main elements in local-search-based heuristics are coding a feasible solution of the problem as a string of values, and defining a move function from one solution to another. To code a feasible solution, we use the permutation of items. As the move function, we use a swap function: At any iteration, given the current solution and two items, the swap function swaps the items to move to a new solution. Selecting which items to swap is important, and strategies involving a *candidate neighbor list* play an important role in tabu-search. To illustrate, we give two extreme alternatives: One is to consider swapping all $\binom{|I|}{2}$ item pairs, and to move to the best one. The other is to choose a random pair from the $\binom{|I|}{2}$ possibilities, and to move to it. The first option leads to a locally good choice, but is more prone to becoming stuck in a local optima. Additionally, it takes more computational effort than the second option. We

use a candidate list strategy that is between these two extremes. We define the candidate neighbors as follows: Choose a random item, consider swapping all other $|I|-1$ items with it, and move to the solution with the best objective function value.

Tabu search, as the name suggests, puts restrictions on the search. It deems some moves as *tabu*, and forbids them for some number of iterations, to diversify the search. We label both of the items chosen for swap as tabu, eliminating them from the random list of candidate items for some number of iterations. After selecting the random item, there is no tabu on selecting the second item to swap. We still use the entire list, try each of the $|I|-1$ items as paired with the randomly chosen item, and choose the one leading to the best objective function value. The number of iterations items are banned is called the *tabu tenure*, and is a parameter defining the search.

Tabu search is well known for its diversification and intensification strategies. The above strategy of making some items tabu is a means for diversification. For intensification, we save the top quality solutions we encounter throughout the search. After the search terminates, we use these *elite* solutions as a starting solution and do a separate search for each of them. The aim is to explore the good solutions further, hoping to visit better neighboring solutions. Two parameters are important here: One is the size of the list of elite solutions. The larger the size, the greater the chance of visiting the optimal solution. On the other hand, the larger the size, the greater the search time. The second parameter uses the notion of a distance between two

solutions, and ensures the list of elite solutions contains only those that exceed a pairwise minimum distance threshold. The point is that, we do not want to fill the list with multiple similar solutions or solutions near the same local optimum. Combining the two parameters, we update the list of elite solutions as follows: At each iteration we consider the current permutation as a candidate for the list. If the list contains a permutation that has a better objective function value and that is close to the current permutation according to the *distance parameter*, then current permutation is not inserted into the list. Otherwise, the current permutation is inserted into the list. All of the permutations that have both: (a) a worse objective function value than the inserted permutation and (b) a distance less than the distance parameter are removed from the list. Finally, if the list's size becomes larger than the size parameter, the permutation with the worst objective function value is removed. There is always the possibility that the list shrinks from iteration to iteration, but its size should be at least one, and at most the size parameter.

A permutation can be defined by the $\mathbf{s}$ vector used in models (2.12) and (3.2). The distance function we use is the Hamming distance between the two solutions, $\mathbf{s}$ and $\mathbf{s}'$. Hence, it can be at least 0 and most $\binom{|I|}{2}$.

Making a move to a solution tabu is based on a short term memory structure, whereas keeping the list of elite solutions is based on a long-term memory structure. One more diversification routine is implemented based on a long-term memory structure. A search might become stuck in a local optimum in spite of our process of making solutions tabu. To help prevent this, we can

store properties of the mostly-visited solutions, and start the search again from a permutation with differing properties. We implement this as follows: Permutations are used to define a score for each item. The first item in the sequence receives the lowest score and the last one receives the highest score. At each iteration, we sum the scores of the items. After the search is complete, we order the items based on descending score, and do another search starting from this sequence.

Compiling all of the ideas above, we implement the heuristic in two passes. The first pass performs $n + 1$ searches, where n is a parameter of heuristic. The first search starts from a prespecified permutation and the remaining n searches start from random permutations. The number of iterations in each search is defined by another parameter, m. During each search of the first pass, we keep a separate scoring list for items and a common list of elite solutions. In the second pass, we do a separate search starting from the scoring lists formed during the first pass. We also do a search for each of the permutations in the list of elite solutions. Hence, in the second pass, there are as many searches as the size of the resulting list of elite solutions plus $n + 1$. After finishing the second pass, the heuristic reports the permutation with the best objective function value.

In our implementation, the parameters are as follows: The number of initial random-start searches, n, is equal to the number of items in the problem, i.e., $|I|$. The number of iterations per search, m, is equal to 100. The tabu tenure is equal to $1/3|I|$. The size of the elite solution list is $1/2|I|$. Finally,

the distance parameter is $|I|$.

3.1.4 Additional Implementation Details

Let z^*_{MLP} denote the optimal value of the LP relaxation of model (3.1), $z^*_{RMLP,k}$ denote the the optimal value of model (3.2) at iteration k of the column-generation algorithm, and $z^{*\omega}_k$ denote the optimal value of model (3.3) for scenario ω at iteration k. At every iteration k, $z^*_{RMLP,k}$ forms a lower bound for z^*_{MLP}, and $z^*_{RMLP,k}$ is nondecreasing. Hence, at each iteration we obtain an improving lower bound for the optimal MLP. As an upper bound, we use the quantity [e.g., 26]:

$$\overline{z}_{MLP,k} = z^*_{RMLP,k} + \sum_{\omega \in \Omega} z^{*\omega}_k. \tag{3.4}$$

Unfortunately, $\overline{z}_{MLP,k}$ is not necessarily nonincreasing, and so we keep track of the best upper bound. If the upper bound is smaller than the objective function value of the incumbent solution, the corresponding branch-and-bound node can be pruned. Also, when these upper and lower bounds are equal, or deemed sufficiently close, we can terminate the column-generation process.

One other implementation issue concerns the column elimination. As column generation continues, some columns generated at the early iterations may never enter the basis again. Hence, we might want to delete columns that stay out of the basis for some pre-specified number of iterations. But, we need to be wary of cycling that can occur if column elimination is performed too aggressively. Another form of column elimination is based on the

number of branch-and-bound nodes for which that column never enters the basis. Columns generated at a node are inherited by the descendants of that node. It may happen that, some of the columns never prove to be useful in the descendants, and hence, we can eliminate those columns from further consideration [e.g., 24]. In our implementation, neither form of column elimination significantly improves the algorithm's performance.

A third issue involves feasibility restoration. Both of our branching rules force elimination of some columns. In the corresponding child nodes, the RMLP with the remaining columns may be infeasible. In this case, we must initialize with new columns so that column generation can restart. The same issue also exists at the root node. There is a simple way to handle this. Whatever branching rules we use, regardless of the form of the RMLP, we simply add columns corresponding to the null solution so that the RMLP becomes feasible. That is, adding $|\Omega|$ columns consisting of all 0s except a single 1 at the index corresponding to the ω^{th} position yields a feasible RMLP.

3.2 Computational Results

Our branch-and-price algorithm is implemented using COIN-BCP, a serial and parallel branch-cut-price framework developed and maintained by the COIN-OR initiative [7]. BCP is a software system that solves mixed-integer linear programming problems by the branch-and-bound algorithm with linear programming relaxations. It maintains the branch-and-bound tree, a pool of cuts and columns, and offers the user a wide variety of IP solution

strategies, such as, strong branching, variable selection strategies, tree search strategies, etc. In branch-and-price, the user needs an LP solver to solve RMLP and an IP solver to solve the pricing problems. For this purpose, BCP uses COIN-OSI, another COIN-OR project, as an interface to various commercial or open-source solvers, such as Cplex, Xpress, COIN-CBC, etc.

When running the branch-and-price algorithm on a single processor, we use Cplex version 10.1 to solve both the master and pricing problems. We use the problem instances described in Section 2.3. We run our branch-and-price algorithm on a Dell Precision 530 Workstation with an Intel Xenon 1.8 GHz dual processor with 1 GB RAM. We solve the problems to a 0.01% optimality gap, unless otherwise stated. While presenting the results, whenever we say "average", we mean the geometric average. We use geometric average as a summary statistic, since we typically consider the ratio of running times when comparing two algorithms.

3.2.1 Branching Rules

We first give computational results comparing the precedence branching with and the selection branching strategies described in Section 3.1.2. The latter strategy proved to be highly inefficient, only able to solve small-sized problems. Hence, to compare both branching rules, we use small problem instances, (6-10) and (10-10). In Table 3.1, we compare two branching rules on 26 problem instances using two tree search strategies: breadth-first search (BFS) and best-bound search (BBS). Depth-first search (DFS) is not included because

Table 3.1: Comparing branching rules. BFS stands for a breadth-first tree search strategy; BBS for a best-bound first search strategy. Speedup columns give the ratio of running times under selection branching to that under precedence branching. Hence, values greater than one favor precedence branching.

	BFS					BBS				
	prec. bran.		selec. branc.			prec. bran.		selec. bran.		
Prob. Inst.	# node	run time	# node	run time	s-up	# node	run time	# node	run time	s-up
6-10-1-1	1	0	1	0	3.00	1	0	1	0	0.33
6-10-3-0	1	0	3	0	1.25	1	0	1	0	1.67
6-10-3-1	1	0	1	0	1.67	1	0	1	0	0.83
6-10-9-0	7	0	15	0	2.14	5	0	9	0	1.56
6-10-9-1	1	0	1	0	1.00	1	0	1	0	0.85
6-10-18-0	3	0	15	1	3.06	3	0	7	0	1.70
6-10-18-1	25	1	33	1	0.98	17	1	17	1	1.26
6-10-27-0	3	0	65	7	16.06	3	0	13	1	2.53
6-10-27-1	15	1	115	8	5.62	9	1	117	8	9.62
6-10-54-0	7	2	4685	828	402	7	2	179	23	11.92
6-10-54-1	51	7	30401	3400	494	23	4	2629	362	95.59
6-10-81-0	7	3	1.93M	360K*	104.2K	5	3	1.26M	379.8K*	117.8K
6-10-81-1	5	3	1.29M	378K*	116.1K	5	3	15521	2068	630
Average					**28.64**					**9.04**
10-10-1-1	1	0	1	0	1.25	1	0	1	0	0.75
10-10-3-0	171	5	209	6	1.29	171	5	187	6	1.17
10-10-3-1	3	0	7	0	2.27	3	0	7	0	2.32
10-10-9-0	143	15	5309	661	45.49	95	11	4665	617	57.66
10-10-9-1	131	12	771	81	6.77	73	7	521	65	9.16
10-10-18-0	117	21	241523	72.9K	3528	89	14	144.7K	34.3K	2389
10-10-18-1	183	21	1131	283	13.56	51	7	787	147	20.05
10-10-27-0	145	48	990345	360K*	7494	89	31	1.02M	360K*	11434
10-10-27-1	75	27	264714	557K*	20.5K	45	18	44043	19933	1133
10-10-54-0	151	114	73490	360K*	3153	89	69	70435	360K*	5224
10-10-54-1	373	177	68234	360K*	2031	155	66	80177	360K*	5447
10-10-81-0	125	185	29872	360K*	1947	51	94	30234	360K*	3824
10-10-81-1	251	396	23421	360K*	908	159	246	28765	360K*	1464
Average					**169.32**					**170.89**

of memory limitations. In most of these problem instances, the combination of selection branching and depth-first search results in an out-of-memory error. Table 3.1 provides two columns to summarize the results. The second column gives the running time of the algorithm in terms of seconds. Some values for selection branching are given with asterisk sign, denoting lower bound estimates for the running time. That is, after the given time we terminate the algorithm without waiting for the algorithm to solve optimally. The first column represents the total number of branch-and-bound nodes evaluated before terminations.

Precedence branching solves all problem instances within ten minutes, and largely outperforms selection branching. The last column in each search strategy, speedup, gives the ratio of the running time of selection branching to that of precedence branching so that values greater than 1 favor precedence branching. On average, precedence branching is 28.64 times better in the (6,10) data set, if breadth-first search is used. The same number is 169.32 for the (10,10) data set. If, instead we use best-bound search, the speedup is smaller for the (6,10) data set, but about the same for the (10,10) data set. The benefit of precedence branching becomes amplified as the problem size increases, either in terms of the number of scenarios or the number of items.

Based on the superior performance of precedence branching, we use this branching rule for all other remaining computational experiments. We also select BBS as the search strategy in the computational studies that follow.

Table 3.2: The contribution of the primal heuristic. Speedup columns specify the ratio of the algorithm's running time not using the primal heuristic to that using the primal heuristic. Hence, values greater than 1 favor the heuristic.

	Tolerance: 0.01%					Tolerance: 5%					
	w. heur.		w.o. heur.			w. heur.		w.o. heur.			
Prob. Inst.	# node	run time	# node	run time	s-up	# node	run time	# node	run time	s-up	Prob. Inst.
6-10-9-0	1	0	7	0	1.06	5	2	141	13	8.60	10-10-9-0
6-10-9-1	1	0	1	0	0.40	1	1	53	5	5.91	10-10-9-1
6-10-18-0	3	0	5	0	0.62	3	3	37	8	3.19	10-10-18-0
6-10-18-1	17	1	21	1	0.92	1	2	53	7	4.51	10-10-18-1
6-10-27-0	1	1	3	0	0.58	5	7	107	35	5.15	10-10-27-0
6-10-27-1	9	1	15	1	1.03	1	3	55	19	5.63	10-10-27-1
6-10-54-0	3	1	7	2	1.30	7	17	115	81	4.73	10-10-54-0
6-10-54-1	23	5	39	5	1.19	1	5	123	58	10.79	10-10-54-1
6-10-81-0	5	5	7	3	0.76	3	21	95	138	6.71	10-10-81-0
6-10-81-1	5	5	7	3	0.81	9	42	183	263	6.27	10-10-81-1
Average					**0.82**					**5.83**	
10-10-9-0	95	11	159	15	1.32	1	2	127	38	17.93	15-10-9-0
10-10-9-1	73	8	81	8	1.01	1	3	369	108	41.21	15-10-9-1
10-10-18-0	45	11	69	13	1.16	1	4	903	524	121.1	15-10-18-0
10-10-18-1	51	9	69	9	1.06	1	4	51	27	6.81	15-10-18-1
10-10-27-0	55	25	107	35	1.38	1	6	1241	1288	212.6	15-10-27-0
10-10-27-1	29	15	55	19	1.27	51	178	4929	4658	26.24	15-10-27-1
10-10-54-0	89	72	147	99	1.37	1	26	4317	1035	387	15-10-54-0
10-10-54-1	73	48	129	62	1.28	1	27	183	816	30.78	15-10-54-1
10-10-81-0	47	95	95	140	1.48	1	19	651	3327	173.2	15-10-81-0
10-10-81-1	113	210	199	289	1.37	35	1047	5949	25551	24.4	15-10-81-1
Average					**1.26**					**52.89**	
15-10-9-0	83	37	137	44	1.21	1	4	281	203	51.7	20-10-9-0
15-10-9-1	213	89	375	115	1.30	1	4	1177	832	190.4	20-10-9-1
15-10-18-0	549	428	917	576	1.35	1	8	3495	5431	677.1	20-10-18-0
15-10-18-1	29	26	51	28	1.11	1	15	1759	3475	227.7	20-10-18-1
15-10-27-0	749	1018	1431	1490	1.46	1	11	799	2344	212.8	20-10-27-0
15-10-27-1	7655	7222	11925	9638	1.33	1	50	31151	110.5K	2192	20-10-27-1
15-10-54-0	3227	9298	5357	12487	1.34	1	27	34259	249.1K	9212	20-10-54-0
15-10-54-1	4227	11298	5789	14573	1.29	1	81	44556	289.4K	1601	20-10-54-1
15-10-81-0	365	2767	677	3511	1.27	1	40	19249	234.8K	5873	20-10-81-0
15-10-81-1	5219	27798	8963	35648	1.28	61	10097	10923	360K*	35.65	20-10-81-1
Average					**1.29**					**541.95**	

3.2.2 The Primal Heuristic

We carry out computational tests to assess the potential benefit of the primal heuristic described in Section 3.1.3. We test for two different settings: In the first, the test problems are solved up to a 0.01% optimality gap; in the second, up to 5%. For the first setting, we use the problem instances (6,10), (10,10) and (15,10). For the second setting, we use instances (10,10), (15,10), and (20,10). The other instances are omitted since solving them without the heuristic takes too much time, even up to 5% optimality.

The results are displayed in two portions in Table 3.2. The left portion, along with the problem instances in the left-most column, gives the results when the problems are solved to within a 0.01% tolerance. The right portion with problem instances in the right-most column, gives the same results when the problems are solved to within a 5% tolerance. The speedup column in each portion is the ratio of the running time of the algorithm when the heuristic is not applied to that of the case when the heuristic is applied. Hence, values greater than 1 favor the heuristic.

In the case of solving problems to within a 0.01% tolerance, heuristic seems to have little impact on running time. In our smallest data set, (6,10), the computational effort associated with the heuristic is more than its benefit in terms of reducing the search tree. In the case of solving with a 5% tolerance, using the heuristic significantly reduces the running time. In both cases, the effect of the heuristic is greater as the problem size increases in terms of the number of items.

Table 3.3: Cplex results. The speedup columns give the ratio of Cplex's running time to that of our branch-and-price algorithm. Hence, values greater than 1 favor branch-and-price algorithm.

	Tolerance: 0.01%			Tolerance: 5%			
Prob. Inst.	# node	run time	s-up	# node	run time	s-up	Prob. Inst.
6-10-9-0	1	0	0.66	60	3	1.79	10-10-9-0
6-10-9-1	10	0	0.84	60	2	2.54	10-10-9-1
6-10-18-0	5	0	0.24	200	9	3.56	10-10-18-0
6-10-18-1	18	0	0.23	50	4	2.70	10-10-18-1
6-10-27-0	10	1	0.99	500	24	3.54	10-10-27-0
6-10-27-1	15	1	1.01	200	11	3.27	10-10-27-1
6-10-54-0	1	2	1.11	3300	260	15.08	10-10-54-0
6-10-54-1	100	9	2.09	990	82	15.17	10-10-54-1
6-10-81-0	1	1	0.20	23400	3020	146.4	10-10-81-0
6-10-81-1	12	1	0.18	27700	4375	104.1	10-10-81-1
Average			**0.55**			**8.41**	
10-10-9-0	348	4	0.38	1930	134	54.14	15-10-9-0
10-10-9-1	136	2	0.32	1400	67	23.45	15-10-9-1
10-10-18-0	683	15	1.35	500	66	15.25	15-10-18-0
10-10-18-1	174	6	0.74	1190	68	16.98	15-10-18-1
10-10-27-0	3800	108	4.25	13317	3459	570.6	15-10-27-0
10-10-27-1	583	31	2.11	5600	2113	11.9	15-10-27-1
10-10-54-0	13800	980	13.56	12140	12070	465.5	15-10-54-0
10-10-54-1	2466	294	6.08	923	484	18.23	15-10-54-1
10-10-81-0	185.2K	360K*	3803	33468	50081	2607	15-10-81-0
10-10-81-1	3.12M	336.1K	1601	41000	360K*	344	15-10-81-1
Average			**7.47**			**83.76**	
15-10-9-0	7000	459	12.52	7523	2289	581.6	20-10-9-0
15-10-9-1	2515	62	0.70	230	45	10.39	20-10-9-1
15-10-18-0	3740	640	1.49	11400	4898	610.6	20-10-18-0
15-10-18-1	2137	147	5.73	1600	565	37.05	20-10-18-1
15-10-27-0	350K	415.6K*	408.3	4400	2348	213.1	20-10-27-0
15-10-27-1	91700	200K	27.69	70300	360K*	7142	20-10-27-1
15-10-54-0	186.6K	360K*	38.72	40500	228.8K	8460	20-10-54-0
15-10-54-1	78592	110.9K	9.81	62300	360K*	1992	20-10-54-1
15-10-81-0	449.3K	1.06M*	384.6	32300	360K*	9003	20-10-81-0
15-10-81-1	192K	360K*	12.95	26500	360K*	35.65	20-10-81-1
Average			**16.61**			**507.15**	

3.2.3 Comparing with Cplex

In this section, we solve model (2.12) with Cplex version 10.1 and compare the results with those of Table 3.2 which uses our branch-and-price algorithm with precedence branching and the primal heuristic on model (3.1). Table 3.3 gives the Cplex results. The table reads as the previous ones, except for the speedup column, which now gives the ratio of the running time of Cplex to the running times of Table 3.2 using the heuristic. For instance, the speedup of 12.95 given at the last row and fourth column of Table 3.3 is the ratio of 360K to 27798. The results, except for the smallest data set (6,10), show a clear superiority of our branch-and-price algorithm over Cplex. This superiority is greater when we solve the problem instances to a 5% tolerance. We use Cplex to solve the RMLP and the pricing problems in our branch-and-price implementation.

3.3 Parallelization Approaches

Parallelizing branch-and-bound algorithms has received considerable attention in the literature. There are three main approaches [e.g., 13]: First, the nodes of the branch-and-bound tree are distributed to the processors. Namely, one processor, called the master, maintains the branch-and-bound tree, specifies the search strategy, and sends the nodes to the other processors, called slaves. These processors solve LP relaxations of the nodes, perform bounding, create child nodes, and send the child nodes back to the master processor. In the second approach, the workload performed at each node

of the branch-and-bound tree is shared among multiple processors. That is, processors work on the same node of the tree, and when finished, move to the next node. In these two approaches, parallelization does not change the serial branch-and-bound algorithm. It merely distributes the workload to multiple processors. In the third approach, processors work on their own branch-and-bound algorithm, each using (possibly) different strategies, and the processors share information, such as upper bound, lower bound, etc.

Most of the literature on parallelizing the branch-and-bound algorithm is devoted to the first approach. Specifically, COIN-BCP implements this approach, although there is some flexibility towards implementing the second one [35]. In this section, we apply the first two approaches to our branch-and-price algorithm. For the first approach, we use COIN-BCP's implementation; for the second, we develop our own. For reasons we discuss later, if the number of nodes in the tree is sufficiently large, the first approach proves more efficient. The reason we develop the second approach is for small-sized branch-bound trees. Some problems are so large that that even solving the root node of the tree is challenging. In this case, it might be desirable to solve only the root node, or at most, a few other nodes. Hence, we develop the second approach to obtain better efficiencies in these circumstances. In what follows, we call the first parallelization approach *inter-node parallelization*, and the second approach *intra-node parallelization*.

We give more detailed explanations about the two approaches in the corresponding sections, but we give a broad outline of the intra-node paral-

lelization approach here. In this approach, at a given node of the tree, we mainly perform two tasks: solving the RMLP and solving the pricing problems. There are as many pricing problems as the number of scenarios, $|\Omega|$. These pricing problems are independent of each other, which lends themselves to parallelization. Hence, we implement the intra-node approach as follows: The master processor solves the RMLP, and sends out dual solutions to the slaves. These slaves solve the pricing problems, and send resulting solutions back to the master.

We test both inter-node and intra-node parallel implementations on the data sets (10,10) and (15,10). These two data sets are chosen to represent the problem instances that require small and large numbers of nodes in the branch-and-bound tree. We run the implementations on the Lonestar Dell Dual-Core Linux Cluster at the Texas Advanced Computing Center [44]. Compute nodes have two processors, each a Xeon 5100 series 2.66GHz dual-core processor with a 4MB unified (Smart) L2 cache. The memory system uses Fully Buffered DIMMS (FB-DIMMS) and a 1333 MHz (10.7 GB/sec) front side bus. An InfiniBand switch fabric, employing PCI Express interfaces, interconnects the nodes (I/O and compute) through a fat-tree topology, with a point-to-point bandwidth of 1GB/sec (unidirectional speed). We use COIN-CLP, COIN-OR's LP solver, to solve the master problem, and COIN-CBC, COIN-OR's mixed integer programming solver, to solve the pricing problems. Before describing the implementations in greater detail, we briefly discuss how to measure the efficiency of a parallel algorithm.

3.3.1 Efficiency of Parallel Algorithms

One of the commonly used efficiency criterion involves measuring the savings in completion time of the algorithm by running on multiple processors, as compared to running it on a single processor. Restating in ratio terms, the *speedup* criterion is the ratio of the CPU time on one processor to that on multiple processors. Ideally, one can obtain a linear speedup, equal to the number of processors involved in parallelization. The efficiency is considered good if the speedup is close to linear.

What makes the speedup less than the number of processors is the serial fraction of the algorithm that cannot be run in parallel. Let f_n be the fraction of time the algorithm that runs in serial when it is run on n processors. Then, the speedup, s_n, on n processors is

$$s_n = \frac{\text{serial time} + \text{parallel time}}{\text{serial time} + \frac{\text{parallel time}}{n}} = \frac{1}{f_n + \frac{1-f_n}{n}}. \tag{3.5}$$

If f_n is bounded below by f, the speedup can at most be $1/f$. Taking, for instance, f to be 0.05, speedup can at most be 20, even if we use thousands of processors.

In some parallel algorithms, it is easy to estimate f_n beforehand. Consider, for instance, our implementation of the intra-node approach. If the number of scenarios is evenly divisible by the number of processors, and if the running times of pricing problems are identical, then the serial fraction of the algorithm is the ratio of the time to (repeatedly) solve the RMLP to the overall time, at least under a synchronous implementation. Even in this

simple case, to obtain f_n, we had to make some assumptions on the number of scenarios, on the number of processors, and on the running times of pricing problems. On more intricate cases, it is more difficult. Furthermore, in most of the parallel algorithms, some non-negligible amount of time is spent for message passing. This time depends on the communication requirement of the algorithm as well as the specific hardware on which the algorithm is running. Hence, it is usually a good idea to obtain the observed serial fraction, $\hat{f}_n$, and use it to estimate the performance of the algorithm on more processors [e.g., 18]. Isolating f_n in equation (3.5), we obtain

$$f_n = \frac{1/s_n - 1/n}{1 - 1/n}. \tag{3.6}$$

It is reasonable to assume that f_n increases with n, for at least two reasons: First, the granularity of the algorithm might be limited. Considering our implementation of the intra-node parallelization approach, we can use at most as many processors as the number of scenarios. After this number, any additional processor waits idle. The second reason is the message passing overhead. Even if the granularity of the algorithm is not limited, after some number of processors, the message passing time dominates the time spent carrying out computations at the processors. Hence, assuming f_n increases with n, the observed serial fraction $\hat{f}_n$ gives us a good deal of information about the speedup. That is, we can deduce that the best speedup, having the flexibility of using arbitrarily many processers, is limited by $1/\hat{f}_n$ for each n.

Lastly, consider two parallel implementations of the same algorithm. Suppose one has a serial fraction sequence, indexed by the number of processors

as, $f_n = \frac{n}{2(n+10)}\ \forall n$. The other has $f_n = 1/3\ \forall n$. For values of n less than 20, the first implementation is better than the second, but eventually the second algorithm has the best speedup of 3, while the first drops towards 2. So, we use speedup as the primary efficiency criterion, but we also report the observed serial fractions to point out the potential best speedup that the algorithm can achieve, if we were unconstrained in the number of processors that could be used.

3.3.2 Inter-Node Parallelization

There are multiple implementations of an inter-node parallelization scheme. We sketch the basic algorithm that COIN-BCP implements [35]. It uses one master processor to organize and maintain the branch-and-bound tree, and lead the search. Namely, the master processor sends current nodes in the pool to the slave processors, receives the nodes processed by the slaves, and repeats this process until all of the nodes in the tree are pruned. Slave processors solve LP relaxations and perform bounding. During the search, as soon as a slave finds an incumbent solution, it sends it to the master processor, which communicates the solution to all the slaves.

BCP uses message passing as its parallel programming paradigm [34]. Specifically, it can be run either using PVM, or MPI. We use the MPI parallel programming library, which is the most modern form of libraries implementing the message passing paradigm. For more on different parallel programming paradigms, on the message passing paradigm, and specifically on MPI see

Table 3.4: Speedup results for inter-node parallelization. The instances are solved to within a 0.01% tolerance. The third column shows the serial running time in seconds. The remaining columns show the speedup.

Prob. Inst.	# nodes	# processors								
		1	4	8	12	16	20	24	28	32
10-10-9-0	57	20	2.19	2.32	2.29	2.29	2.29	2.26	2.27	2.27
10-10-9-1	59	19	2.92	3.43	3.50	3.46	3.49	3.47	3.44	3.38
10-10-18-0	41	36	2.61	2.78	2.76	2.76	2.76	2.77	2.75	2.70
10-10-18-1	43	20	1.79	1.88	1.87	1.86	1.89	1.88	1.88	1.87
10-10-27-0	47	72	1.81	1.82	1.82	1.82	1.82	1.81	1.81	1.80
10-10-27-1	33	55	1.90	1.91	1.90	1.90	1.91	1.91	1.90	1.90
10-10-54-0	93	252	3.10	3.55	3.55	3.55	3.56	3.56	3.55	3.55
10-10-54-1	79	110	2.09	2.43	2.47	2.46	2.47	2.46	2.46	2.48
10-10-81-0	65	421	2.24	2.24	2.23	2.23	2.24	2.23	2.23	2.24
10-10-81-1	121	508	2.92	3.79	3.83	3.82	3.79	3.83	3.83	3.81
Average			**2.31**	**2.53**	**2.53**	**2.52**	**2.53**	**2.52**	**2.52**	**2.51**
15-10-9-0	91	499	3.50	4.05	4.02	4.05	4.03	4.12	4.11	4.11
15-10-9-1	143	652	3.62	6.26	7.57	8.21	8.17	8.30	8.31	8.35
15-10-18-0	541	4820	3.83	7.24	10.04	12.29	14.22	16.51	17.69	18.70
15-10-18-1	39	280	2.78	2.81	2.81	2.81	2.81	2.87	2.88	2.87
15-10-27-0	717	13264	3.87	7.26	10.77	13.55	15.10	18.23	19.69	21.11
15-10-27-1	6881	49665	3.84	7.70	11.28	14.94	18.26	22.53	25.85	29.18
15-10-54-0	3223	75514	3.80	7.53	11.02	14.34	17.52	21.99	25.26	28.24
15-10-54-1	3537	47025	3.73	7.47	11.01	14.57	17.70	22.19	25.27	28.41
15-10-81-0	321	23838	3.66	6.79	9.28	9.93	10.24	11.01	11.12	11.12
15-10-81-1	5249	192208	3.91	7.68	11.51	14.81	18.27	23.16	26.47	29.40
Average			**3.64**	**6.21**	**8.18**	**9.67**	**10.77**	**12.35**	**13.24**	**14.04**

Pacheco [32].

Table 3.4 shows results of our inter-node parallelization on various numbers of processors. The values in column 3 are the CPU times spent on a single processor to run the algorithm. The values in other columns are the speedups. Hence, time required to run the algorithm on twelve processors, for instance,

Table 3.5: Observed serial fractions for the results in Table 3.4.

Prob. Inst.	# processors							
	4	8	12	16	20	24	28	32
10-10-9-0	0.28	0.35	0.38	0.40	0.41	0.42	0.42	0.42
10-10-9-1	0.12	0.19	0.22	0.24	0.25	0.26	0.26	0.27
10-10-18-0	0.18	0.27	0.30	0.32	0.33	0.33	0.34	0.35
10-10-18-1	0.41	0.46	0.49	0.51	0.51	0.51	0.51	0.52
10-10-27-0	0.40	0.49	0.51	0.52	0.53	0.53	0.53	0.54
10-10-27-1	0.37	0.46	0.48	0.49	0.50	0.50	0.51	0.51
10-10-54-0	0.10	0.18	0.22	0.23	0.24	0.25	0.25	0.26
10-10-54-1	0.30	0.33	0.35	0.37	0.37	0.38	0.38	0.38
10-10-81-0	0.26	0.37	0.40	0.41	0.42	0.43	0.43	0.43
10-10-81-1	0.12	0.16	0.19	0.21	0.23	0.23	0.23	0.21
Average	**0.23**	**0.30**	**0.34**	**0.35**	**0.36**	**0.37**	**0.37**	**0.38**
15-10-9-0	0.05	0.14	0.18	0.20	0.21	0.21	0.22	0.22
15-10-9-1	0.03	0.04	0.05	0.06	0.08	0.08	0.09	0.09
15-10-18-0	0.01	0.02	0.02	0.02	0.02	0.02	0.02	0.02
15-10-18-1	0.15	0.26	0.30	0.31	0.32	0.32	0.32	0.33
15-10-27-0	0.01	0.01	0.01	0.01	0.02	0.01	0.02	0.02
15-10-27-1	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
15-10-54-0	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
15-10-54-1	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
15-10-81-0	0.03	0.03	0.03	0.04	0.05	0.05	0.06	0.06
15-10-81-1	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Average	**0.02**	**0.02**	**0.02**	**0.02**	**0.03**	**0.02**	**0.02**	**0.02**

can be obtained by dividing the number in column 1 to that in column 12. We also give the number of branch-and-bound nodes generated throughout the algorithm. After each data set, there is a row giving the geometric averages of the speedups.

An immediate observation is the correlation between the number of branch-and-bound nodes and the speedup: the more nodes, the greater the

speedup. This can be observed better by comparing the average speedups of the data sets (10,10) and (15,10). In some problem instances, there is no improvement in speedup, after some number of processors. This is due to the limited granularity of the algorithm in small branch-and-bound trees. That is, throughout the search, the number of nodes that are run in parallel is less than the number of processors. However, if there are more than a few branch-and-bound nodes in the tree, which is the case for many linear integer programming problems, the speedup is close to ideal, as in the cases of 15-10-27-1 and 15-10-81-1. One caveat about the number of nodes is the following. If there were no pruning based on the upper bound information, the number of nodes in the tree would be identical no matter how many processors are used. But, in the case of pruning by an upper bound, using different numbers of processors can lead to different number of nodes, due to different timings of obtaining new incumbent solutions. To prevent this from happening we used a best-bound-search strategy. Furthermore, the effectiveness of our primal heuristic leads to near-optimal solutions early in the tree most of the time. Hence, in our results, the number of nodes in the tree is always the same. This is important, especially when comparing the speedups, because otherwise we would be comparing running times of algorithms that search different branch-and-bound trees.

Inspecting Table 3.4 for the data set (15,10) leads to the question of whether we can obtain increasing speedup as we increase the number of processors. This can be estimated, to some extent, by looking at the observed

serial fractions in Table 3.5. These observed serial fractions show that the data set (15,10) has potential to give increasing speedup as we increase the number of processors. In fact, some of the instances can have a speedup close to 100. On the other hand, data set (10,10) does not have promising speedups. Some of the instances, 10-10-18-1 and 10-10-27-0, are limited to a speedup of less than 2.

3.3.3 Intra-Node Parallelization

In our intra-node parallelization scheme, we distribute the pricing problems to slave processors. Specifically, master processor solves the RMLP, sends dual solutions to the slaves, which solve the pricing problems, and return solutions to the master. Clearly, the serial fraction here is the ratio of the solution times for the RMLP to that time plus the computational time for the pricing problems. In our branch-and-price implementation, average serial fractions for the data sets (10,10) and (15,10) are 0.2 and 0.09, respectively. This prevents us from obtaining better speedups than inter-node parallelization, at least in the case of (15,10). As we mention above, it is best to apply this parallelization to the case of small branch-and-bound trees, where inter-node parallelization performs poorly.

To illustrate this, we run the branch-and-price algorithm until we obtain ten nodes in the branch-and-bound tree. As soon as the tenth node is solved, the algorithm terminates. We take the view that we may be required to terminate with small branch-and-bound trees like this when solving the prob-

Table 3.6: Speedup results for inter-node parallelization. Only ten branch-and-bound nodes are solved. The third column shows the serial running time in seconds. The remaining columns show the speedup.

Prob. Inst.	Rem. Gap (%)	# processors								
		1	4	8	12	16	20	24	28	32
10-10-9-0	2.3	9	1.61	1.64	1.63	1.62	1.63	1.61	1.62	1.62
10-10-9-1	3.1	7	1.89	1.88	1.85	1.87	1.87	1.87	1.87	1.86
10-10-18-0	2.4	15	1.44	1.67	1.66	1.65	1.65	1.66	1.65	1.65
10-10-18-1	1.2	10	1.44	1.45	1.45	1.43	1.43	1.42	1.46	1.45
10-10-27-0	2.1	41	1.80	1.79	1.80	1.80	1.80	1.80	1.80	1.79
10-10-27-1	2.4	36	1.63	1.64	1.63	1.63	1.63	1.64	1.62	1.63
10-10-54-0	3.7	79	2.02	2.02	2.02	2.03	2.02	2.02	2.02	2.02
10-10-54-1	1.4	41	1.17	1.17	1.17	1.17	1.17	1.17	1.17	1.17
10-10-81-0	3.6	188	2.13	2.13	2.13	2.13	2.13	2.13	2.13	2.12
10-10-81-1	4.4	156	1.87	1.92	1.92	1.92	1.92	1.92	1.92	1.91
Average			**1.67**	**1.71**	**1.70**	**1.70**	**1.70**	**1.70**	**1.71**	**1.70**
15-10-9-0	0.9	80	1.85	1.64	1.63	1.64	1.64	1.64	1.64	1.64
15-10-9-1	2.0	74	1.70	1.72	1.72	1.72	1.72	1.72	1.72	1.71
15-10-18-0	2.6	187	1.79	1.79	1.80	1.79	1.79	1.79	1.79	1.78
15-10-18-1	0.3	119	1.84	1.84	1.84	1.83	1.83	1.84	1.84	1.84
15-10-27-0	2.0	328	1.85	1.78	1.79	1.78	1.78	1.78	1.78	1.78
15-10-27-1	6.1	412	2.00	2.22	2.22	2.21	2.20	2.21	2.22	2.21
15-10-54-0	4.3	654	1.76	1.79	1.78	1.78	1.80	1.79	1.78	1.78
15-10-54-1	3.1	481	1.79	1.82	1.83	1.82	1.83	1.83	1.83	1.83
15-10-81-0	1.6	1253	1.97	1.97	1.97	2.14	2.12	1.95	2.14	2.14
15-10-81-1	5.8	2367	2.03	2.11	2.09	2.10	2.10	2.11	2.10	2.09
Average			**1.85**	**1.86**	**1.86**	**1.87**	**1.87**	**1.86**	**1.87**	**1.87**

lem to near optimality is excessively expensive. We give the associated results of the inter-node parallelization in Tables 3.6 and 3.7, and of the intra-node parallelization in Tables 3.8 and 3.9.

The second column of Table 3.6 gives the remaining optimality gaps after solving ten nodes. Comparing the speedups in Table 3.6 with those of

Table 3.7: Observed serial fractions for the results in Table 3.6.

Prob. Inst.	# processors							
	4	8	12	16	20	24	28	32
10-10-9-0	0.50	0.55	0.58	0.59	0.59	0.60	0.60	0.60
10-10-9-1	0.37	0.47	0.50	0.50	0.51	0.51	0.52	0.52
10-10-18-0	0.59	0.54	0.57	0.58	0.59	0.59	0.59	0.59
10-10-18-1	0.59	0.65	0.66	0.68	0.68	0.69	0.67	0.68
10-10-27-0	0.41	0.50	0.52	0.53	0.53	0.54	0.54	0.54
10-10-27-1	0.49	0.56	0.58	0.59	0.59	0.59	0.60	0.60
10-10-54-0	0.33	0.42	0.45	0.46	0.47	0.47	0.48	0.48
10-10-54-1	0.80	0.84	0.84	0.84	0.85	0.85	0.85	0.85
10-10-81-0	0.29	0.39	0.42	0.43	0.44	0.45	0.45	0.45
10-10-81-1	0.38	0.45	0.48	0.49	0.50	0.50	0.50	0.51
Average	**0.45**	**0.52**	**0.55**	**0.56**	**0.56**	**0.57**	**0.57**	**0.57**
15-10-9-0	0.39	0.55	0.58	0.58	0.59	0.59	0.60	0.60
15-10-9-1	0.45	0.52	0.54	0.55	0.56	0.56	0.57	0.57
15-10-18-0	0.41	0.50	0.52	0.53	0.54	0.54	0.54	0.55
15-10-18-1	0.39	0.48	0.50	0.52	0.52	0.52	0.53	0.53
15-10-27-0	0.39	0.50	0.52	0.53	0.54	0.54	0.55	0.55
15-10-27-1	0.33	0.37	0.40	0.42	0.42	0.43	0.43	0.43
15-10-54-0	0.42	0.49	0.52	0.53	0.53	0.54	0.54	0.55
15-10-54-1	0.41	0.49	0.50	0.52	0.52	0.53	0.53	0.53
15-10-81-0	0.34	0.44	0.46	0.43	0.44	0.49	0.45	0.45
15-10-81-1	0.32	0.40	0.43	0.44	0.45	0.45	0.46	0.46
Average	**0.38**	**0.47**	**0.50**	**0.50**	**0.51**	**0.52**	**0.52**	**0.52**

Table 3.4, it is clear that, inter-node parallelization is less efficient than inter-node parallelization. This can also be observed from the serial fractions in Table 3.7. For both data sets, (10,10) and (15,10), the serial fractions are on average greater than 1/2, meaning that we cannot obtain a speedup greater than 2. On the other hand, looking at Table 3.8, we see some speedups close to 5 for data set (15,10). Intra-node parallelization gives better speedups with

smaller number of nodes.

Table 3.8 gives two results: one immediate, and one more subtle. As the number of items increases, intra-node parallelization tends to give better speedups. Secondly, speedups corresponding to problem instances with suffix "-0" are better than the ones with suffix "-1." This is due to greater asymmetry in the pricing problems in the case of "-1." In the case of "-1" we multiply the costs by 0.6, 1, and 1.4 for pessimistic, most likely and optimistic estimates, respectively. Similarly, we multiply the budget with 1.4 and 0.6. Hence, it is expected that scenarios (pricing problems) corresponding to items and budget taking pessimistic multipliers are more difficult to solve than scenarios in which they take optimistic multipliers, as the knapsack constraints are tighter in the first case. The differences between the difficulty of pricing problems are more emphasized when the multipliers are 1.4, 1, and 0.6 as opposed to that of 1.2, 1, and 0.8. This asymmetry affects intra-node parallelization, as some of the processors solve easy pricing problems and wait idle for those that solve harder ones. To observe this difference, we also compute the average speedups and serial fractions for problem instances with suffix "-0" and "-1." The results are given in the last two lines of Tables 3.8 and 3.9.

To overcome the negative effect of the asymmetry in pricing problems, one can modify the implementation so that the master processor does not wait for all the slave processors. After receiving some percentage of pricing problem solutions, the master performs another iteration, computes new dual solutions, and sends those to slaves. The slaves check periodically for dual

solutions from the master. If they receive a new dual solution, they preempt their current task and start working on new pricing problems, using new dual solutions. This strategy comes with some caveats, though. First, the master processor needs to keep track of the pricing problems that are not solved at the current iteration, so that it gives higher priority to those in the next iteration. This way, the algorithm becomes more balanced. Second, after some number of iterations, the master should wait for all the pricing problems to be solved so that it computes the upper bound.

Before finishing the discussion we note the relation between the column based reformulation and intra-node parallelization. The serial fraction in intra-node parallelization depends on how we reformulate the problem. If we had formulated it in such a way that the pricing problems are larger and the RMLP is smaller than their current forms, as could be the case if we had implemented scenario decomposition instead of complete decomposition, then the serial fraction would be much smaller, and we could obtain better speedups.

Table 3.8: Speedup results for intra-node parallelization. Only ten nodes are solved. The second column shows the serial running time in seconds. The remaining columns show the speedup.

Prob. Inst.	# processors								
	1	4	8	12	16	20	24	28	32
10-10-9-0	9	1.82	2.25	2.42	2.44	2.44	2.43	2.41	2.40
10-10-9-1	7	1.60	1.71	1.84	1.82	1.82	1.84	1.84	1.81
10-10-18-0	15	1.80	2.11	2.23	2.34	2.37	2.44	2.43	2.43
10-10-18-1	10	1.60	1.74	1.71	1.86	1.82	1.87	1.87	1.86
10-10-27-0	41	2.15	2.77	3.05	3.25	3.37	3.55	3.71	3.75
10-10-27-1	36	1.96	2.39	2.66	2.79	2.87	3.00	3.07	3.10
10-10-54-0	80	2.01	2.45	2.72	2.83	2.96	3.04	3.08	3.10
10-10-54-1	42	1.78	2.08	2.16	2.26	2.33	2.36	2.41	2.43
10-10-81-0	188	2.32	3.07	3.54	3.75	3.99	4.09	4.15	4.23
10-10-81-1	157	2.00	2.44	2.64	2.78	2.87	2.92	2.99	2.98
Average		**1.89**	**2.27**	**2.44**	**2.55**	**2.61**	**2.67**	**2.71**	**2.71**
15-10-9-0	80	2.62	3.56	4.89	4.97	4.94	4.93	4.95	4.93
15-10-9-1	74	1.98	2.70	2.96	3.00	3.01	3.00	3.01	2.99
15-10-18-0	186	2.61	3.65	4.33	4.58	5.84	5.89	5.86	5.87
15-10-18-1	119	2.28	2.99	3.36	3.74	4.39	4.38	4.36	4.40
15-10-27-0	327	2.99	4.41	5.30	6.38	6.44	6.61	8.32	8.32
15-10-27-1	410	2.41	3.26	3.77	4.14	4.21	4.47	5.11	5.04
15-10-54-0	650	2.54	3.44	3.95	4.28	4.57	4.65	5.03	5.01
15-10-54-1	478	2.17	2.75	3.04	3.26	3.37	3.46	3.63	3.63
15-10-81-0	1235	2.65	3.67	4.30	4.62	4.86	5.02	5.31	5.32
15-10-81-1	2346	2.28	2.96	3.32	3.55	3.65	3.79	3.95	3.96
Average		**2.44**	**3.30**	**3.85**	**4.16**	**4.42**	**4.51**	**4.78**	**4.77**
Average -0		**2.32**	**3.06**	**3.53**	**3.76**	**3.96**	**4.04**	**4.22**	**4.23**
Average -1		**1.99**	**2.45**	**2.67**	**2.82**	**2.91**	**2.98**	**3.06**	**3.06**

Table 3.9: Observed serial fractions for the results in Table 3.8.

Prob. Inst.	# processors							
	4	8	12	16	20	24	28	32
10-10-9-0	0.40	0.37	0.36	0.37	0.38	0.39	0.39	0.40
10-10-9-1	0.50	0.52	0.50	0.52	0.53	0.52	0.53	0.54
10-10-18-0	0.41	0.40	0.40	0.39	0.39	0.38	0.39	0.39
10-10-18-1	0.50	0.52	0.55	0.51	0.53	0.52	0.52	0.52
10-10-27-0	0.29	0.27	0.27	0.26	0.26	0.25	0.24	0.24
10-10-27-1	0.35	0.33	0.32	0.32	0.31	0.30	0.30	0.30
10-10-54-0	0.33	0.32	0.31	0.31	0.30	0.30	0.30	0.30
10-10-54-1	0.42	0.41	0.41	0.41	0.40	0.40	0.39	0.39
10-10-81-0	0.24	0.23	0.22	0.22	0.21	0.21	0.21	0.21
10-10-81-1	0.33	0.33	0.32	0.32	0.31	0.31	0.31	0.31
Average	**0.37**	**0.36**	**0.35**	**0.35**	**0.35**	**0.35**	**0.34**	**0.35**
15-10-9-0	0.18	0.18	0.13	0.15	0.16	0.17	0.17	0.18
15-10-9-1	0.34	0.28	0.28	0.29	0.30	0.30	0.31	0.31
15-10-18-0	0.18	0.17	0.16	0.17	0.13	0.13	0.14	0.14
15-10-18-1	0.25	0.24	0.23	0.22	0.19	0.19	0.20	0.20
15-10-27-0	0.11	0.12	0.11	0.10	0.11	0.11	0.09	0.09
15-10-27-1	0.22	0.21	0.20	0.19	0.20	0.19	0.17	0.17
15-10-54-0	0.19	0.19	0.19	0.18	0.18	0.18	0.17	0.17
15-10-54-1	0.28	0.27	0.27	0.26	0.26	0.26	0.25	0.25
15-10-81-0	0.17	0.17	0.16	0.16	0.16	0.16	0.16	0.16
15-10-81-1	0.25	0.24	0.24	0.23	0.24	0.23	0.23	0.23
Average	**0.21**	**0.20**	**0.19**	**0.19**	**0.18**	**0.19**	**0.18**	**0.18**
Average -0	**0.23**	**0.22**	**0.21**	**0.21**	**0.21**	**0.21**	**0.21**	**0.21**
Average -1	**0.40**	**0.43**	**0.44**	**0.44**	**0.45**	**0.45**	**0.45**	**0.45**

Chapter 4

An Application in the Nuclear Power Industry

4.1 Introduction

When practitioners plan for capital budgeting they often form a priority list of candidate projects, by scoring the projects individually, using economic measures like net present value, benefit-investment ratio, payback period, internal rate of return, etc. The academic literature frequently points out [e.g., 5, 39] that priority lists built on such simple ranking measures are inferior to allocating funds to capital projects using variants of a multidimensional knapsack model. The multidimensional knapsack approach to capital budgeting [e.g., 4, 20, 45] takes as input a budget forecast, along with the stream of liabilities and the profit of each project. The multidimensional knapsack model is an integer program that has: a binary decision variable for each project to indicate whether it is selected; a budget constraint for each time period (e.g., year); and, the objective of maximizing total profit. The output of the multidimensional knapsack model is a collection of projects to be performed, assuming the point forecasts for these input parameters are correct. We refer to this selected collection of projects as a *project portfolio*.

If the costs and profits of the candidate projects as well as the bud-

gets in coming years are known with certainty, the multidimensional knapsack model provides an attractive tool for selecting a project portfolio. However, how should we approach capital budgeting when we have uncertain forecasts for these parameters? One approach is to re-solve a multidimensional knapsack model when refined forecasts for costs, profits and budgets become available. Unfortunately, this is not always viable. Capital projects typically are implemented in phases over time and usually, some irreversible decisions have been made. Thus, it is not always practical to fully revise a project portfolio whenever better forecasts become available. Additionally, the process of obtaining and analyzing the data, performing required reviews and obtaining necessary approvals typically is very time consuming and resource intensive. As a result, practitioners use either simplistic approaches or intuition and experience to address the impact of emerging events and conditions.

To illustrate how uncertainty can cause a problem, we discuss the following common scenario. Imagine that a multidimensional knapsack model has been used in the capital budgeting process to form a project portfolio. Then, over the course of the year, the available budget decreases due to some reason such as an external event or because some projects experience cost overruns. As a practical matter, some other projects are forced out of the portfolio, i.e., they are not performed. Unfortunately, multidimensional knapsack model is not designed to address such a scenario.

Experience over many years of capital budgeting practice indicates that decision-makers often have the right intuition in seeking a priority list that

is robust with respect to changes in budget values as well as project costs and profits. However, it is well known that priority lists formed by scoring projects individually fail to capture dependencies between projects, and we demonstrate this by considering heuristics based on scoring projects using net present value (NPV) and benefit- investment ratio (BIR). (For more on these and other commonly-used investment criteria see, e.g., Chapter 9 of Ross et al. [38].) Thus, it would be beneficial to have an approach to prioritizing that does capture dependencies between projects. Associated analysis then could be used to provide better priority lists to decision-makers to support better risk-informed decisions. The path of investigation described in this chapter is as follows:

- We first investigate whether the optimal solution to a multidimensional knapsack model naturally yields a prioritized list that is robust to the uncertainties described above. We show it does not.

- Next, we heuristically alter the multidimensional knapsack approach, and force it to produce a prioritized list. We describe a class of ways to do so depending on the initial budget value, and we call these *greedy-heuristic priority lists*. We also build heuristic priority lists using BIR and NPV, as is commonly done in practice.

- Then, we ask whether we can build a priority list that outperforms the heuristic priority list, at least when we assume a probabilistic forecast for the uncertain parameters. For the two sets of candidate projects we

examine, this question is answered affirmatively. We formulate a model that explicitly incorporates multiple budget, cost and profit scenarios and forms an *optimal priority list*, which maximizes the expected NPV of the project portfolio ultimately implemented.

We detail briefly our approach of prioritizing projects. Our model is a two-stage stochastic integer program. It builds a priority list and forms a corresponding optimal project portfolio as its first- and second-stage decisions. We assume that the uncertainties regarding the budget, projects' cost and profit reveal during the first year after we commit to the priority list. When forming the optimal project portfolio for a specific scenario, our prioritization model makes sure that no project is in the portfolio unless all projects with higher priority are also in the portfolio. Thus, portfolio of projects corresponding to different scenarios are subsets of each other. This type of approach eliminates the fragility of the multidimensional knapsack model. Namely, projects are not removed from the portfolio, but only added as uncertainty reveals.

The next section provides further background specific to our motivating capital budgeting problem using data from the South Texas Project Nuclear Operating Company (STPNOC), a two-unit nuclear power plant in Wadsworth, Texas. Then, we formulate a multidimensional knapsack model for capital budgeting under a point forecast (i.e., a deterministic forecast) of the problem's input parameters. The solution yields an optimal set of projects to select, and we show these portfolios fail to yield a prioritized list of projects.

That is, we address the first of the three items listed above. The subsequent two sections then address the second and third bulleted items in turn, assuming a distributional forecast for the uncertain parameters. The chapter's penultimate section then applies the optimal prioritization scheme to two larger problem instances involving 41 projects; it also investigates the performance of the heuristics on these problem instances. We note that abbreviated versions of this chapter have appeared in Koç et al. [21, 23], and the full version have appeared in Koç et al. [22].

We consider two real-life problem instances from STPNOC. The first one has 9 projects, the second one has 41 projects. Each project can last up to 5 years. The first instance considers only budget uncertainty. It has 10 budget scenarios. The second one considers also cost uncertainty. It has 10 budget scenarios and 9 cost scenarios, amounting to 90 scenarios. We also construct stylized examples to assess the performance of our heuristics with an eye towards showing that they can perform poorly, i.e., that performance guarantees for the heuristics are not assured. As a result, we recommend using the optimal prioritization scheme, instead of employing an approximation using a heuristic.

4.2 Background and Motivation

As the operator of a large commercial nuclear power generating station, STPNOC evaluates investment in numerous projects and aims to select projects that achieve the organization's objectives. To do so, STPNOC an-

nually develops a priority list of projects. This rank-ordered list specifies the highest priority project, the second highest priority project and so forth. The current budget and project-cost forecasts yield what STPNOC calls the "blue line." Projects above the blue line are to be funded and those below it are not. Thus, the blue line serves as the demarcation where the available budget is exhausted. Over the course of the year, the blue line can shift due to a variety of reasons such as an external event or because a high-priority project experiences cost over-runs. We note that this paradigm is not unique to STPNOC, nor to the nuclear power industry. Rather, similar capital budgeting practices are employed across a wide range of industrial and government applications. The optimization model we describe recognizes the fact that prioritizing is common practice and aims to build priority lists that are financially robust to the types of uncertainties described above. More specifically, we seek a priority list that maximizes the expected NPV of the project portfolio ultimately implemented.

Our approach to forming an optimal priority list focuses on financial performance measures. However, it is recognized that financial goals alone do not drive capital planning decisions. The need to ensure regulatory compliance enters heavily into decision-making at STPNOC, and throughout the commercial nuclear power industry. So, as in other optimization problems, forming an optimal priority list generally requires addressing multiple criteria including both financial and non-financial issues. At STPNOC, and many commercial nuclear plants, this results in application of a multi-attribute utility theoretic

approach to performing this integration [e.g., 19]. In demonstrating the approach we propose, we first consider a small set of example projects from STPNOC in which some projects have negative NPV estimates and hence would be rejected from a purely financial perspective. However, these projects are forced into the project portfolio by plant management because they are deemed necessary to address a safety or regulatory issue. In this example we show how this affects our approach and we further discuss how regulatory and safety issues often can be well-aligned with financial goals.

A typical project at STPNOC is implemented over 1-5 years, and we may prioritize a project now even though its first costs are not incurred until a future year. Furthermore, STPNOC carries out project prioritization annually, and in this sense capital budgeting decisions are implemented using a rolling horizon. Current STPNOC practice in estimating the cost streams associated with the candidate projects is as follows: Optimistic, pessimistic and most-likely cost streams are estimated for each project. Then, each project is categorized as being low-risk, medium-risk or high-risk. This categorization is based on answering multiple questions within each of 17 categories, which range from the projects engineering-design complexity to STPNOCs level of experience with the proposed contractor to the nature of the radiation fields in the installation environment (i.e., anticipated personnel radiation exposure during installation), etc. The higher the risk of the candidate project, the more this weighted sum is skewed toward the pessimistic cost forecast. A point estimate for the projects costs is formed by assigning normalized weights to each

of the optimistic, pessimistic and most-likely cost streams, and calculating the weighted sum. The manner in which the prioritization model we propose explicitly captures multiple budget, cost and profit scenarios was motivated by this type of cost-stream analysis at STPNOC.

A simplistic ranking scheme scores projects individually, e.g., using their NPV, BIR, internal return rate, etc., and then forms a priority list by sorting the projects based on their individual scores. Such an approach fails to recognize the structural and stochastic dependencies among the projects. While our approach forms a priority list, it recognizes that the projects ultimately implemented, after the stochastic budgets, costs and profits are realized, act as a portfolio. That is, the model captures dependencies among the projects. We emphasize that our model is appropriate only when irreversible decisions regarding project selection must be made *before* knowing budget, cost and profit values *with certainty*. If we can wait until these quantities become known before committing to project selection decisions, we should do so and solve what is then a deterministic multidimensional knapsack model.

4.3 Optimal Project Portfolio

As indicated above, capital budgeting classically is formulated using variants of a multidimensional knapsack problem [e.g., 4, 6, 28, 45]. Specifically, given a set of candidate projects, and given (point estimates of) the NPV of each project, the cost of each project in each year and the available yearly budgets, the goal is to find the subset of projects which maximizes the

total NPV while staying within the budget in each year. When considering a single year, the problem can be visualized as packing a knapsack with items of different volume and utility such that the selected items fit in the knapsack and maximize total utility. When the time horizon includes multiple years and selecting a project can obligate funds in more than one year, there are multiple knapsack constraints, i.e., budget constraints, to satisfy; hence the name multidimensional knapsack. The knapsack problem, and its variations, such as multidimensional knapsack, have a rich history, and have received significant attention in the literature [e.g., 20].

In this section, we first set notation and briefly describe a multidimensional knapsack formulation for the deterministic capital-budgeting problem, and then discuss the implications of instead having stochastic budget levels. For simplicity, we begin by only considering stochastic budget levels, but later we handle uncertain project costs and profits. The notation and formulation of the multidimensional knapsack model are as follows:

Indices and sets:

$i \in I$ candidate projects

$t \in T$ time periods (years)

Data:

a_i net present value of project i

b_t available budget in year t

c_{it} cost of project i in year t

Decision variables:

x_i 1 if project i is selected; 0 otherwise

Formulation:

$$\max_{\mathbf{x}} \quad \sum_{i \in I} a_i x_i \tag{4.1a}$$

$$\text{s.t.} \quad \sum_{i \in I} c_{it} x_i \leq b_t, \qquad t \in T, \tag{4.1b}$$

$$x_i \in \{0, 1\}, \qquad i \in I. \tag{4.1c}$$

Constraint (4.1b) ensures the yearly cost of selected projects is within the budget for each year b_t, $t \in T$. Yes/No restrictions on selecting projects are enforced by constraint (4.1c). The objective function (4.1a) sums the NPV contributions of all selected projects. The optimal solution to the multidimensional knapsack model (4.1) gives the portfolio of projects to select which maximizes total NPV while staying within the yearly budgets.

To understand the nature of solutions to model (4.1), we consider a numerical example with 16 projects (see Table 4.1) each having liabilities in

Table 4.1: Problem data (cost and NPV values are in $M).

	yearly costs c_{it}						
Project i	1	2	3	4	5	NPV (a_i)	BIR
1	0.219	0.257	0.085			2.315	4.405
2			0.122	0.103	0.013	0.824	4.328
3	5.044	1.839				22.459	3.338
4	6.740	6.134	10.442			60.589	2.871
5	0.425					0.667	1.569
6	2.125	2.122				5.173	1.272
7	2.387	0.190	0.012	2.383	0.192	4.003	0.883
8		0.950				0.582	0.669
9	0.030	0.030	0.688			0.122	0.192
10		0.2	0.763	0.739	2.539	-2.870	-0.905
11	0.081	0.032				-0.102	-0.925
12	0.300					-0.278	-0.927
13	0.347					-0.322	-0.928
14	4.025	0.297				-3.996	-0.930
15	0.095	0.095	0.095			-0.246	-0.940
16	5.487	5.664	0.500	6.803	6.778	-20.155	-0.957

some or all of the next 5 years. These projects are from STPNOC and were selected because they constituted a set of projects that were close to the budget cutoff point, with some being funded and others not. Thus, the subset of projects identified in Table 4.1 was selected to provide a useful validation of the applicability of the methods discussed in this chapter.

Table 4.1 shows the project cost (c_{it}) and NPV (a_i) values for each of the projects, and the table orders projects by their BIR, i.e., by the ratio of the NPV of a project to its net present cost. Projects 10-16 have negative NPVs,

i.e., $a_i < 0$, and thus also have negative BIRs. We note that the optimization model (4.1) is driven by a purely financial goal, and hence if this were the sole basis for the investment decision, we would not choose any of these projects. However, projects 10-16 have been managerially mandated for inclusion in the portfolio for reasons beyond the scope of our analysis. Obviously, regulatory and safety goals are of foremost concern in the nuclear-power industry. In some industries, inaction on regulatory or safety requirements may result in a fine or other regulatory consequence which is deemed to be "acceptable" to the organization, i.e., one that could be accepted as a profitable business decision. However, failure to meet regulatory or safety goals in commercial nuclear power can very easily result in significant revenue loss. For example, a year-long regulatory mandated shutdown could lead to revenue losses in the range of hundreds of millions of dollars in some cases, and more than a billion dollars at a multi-plant site. We contend that if a more detailed financial analysis of projects 10-16 were performed that included the potential impacts of unfavorable regulatory actions, this analysis could lead to their NPVs being positive. However, for the purposes of this study, since these projects are managerially mandated, we assume there is little reason to justify them financially. Thus, in our example, projects 10-16 are not included when solving model (4.1), except that they reduce the budget available for choosing among projects 1-9, and they do decrease overall NPV of the portfolio by almost \$28M.

We solve 10 instances of model (4.1) with $b_t = \$11\text{M}, \ldots, \$20\text{M}$ for each of the five years in these respective instances, and display the solutions

Table 4.2: Solutions to 10 instances of model (4.1) with $b_t = \$11\text{M}, \ldots, \$20\text{M}$.

Budget, b_t	x_i									NPV
($M)	1	2	3	4	5	6	7	8	9	($M)
11	1	1	0	0	1	0	0	1	0	-23.58
12	1	1	0	0	1	0	0	1	1	-23.46
13	1	1	0	0	0	1	0	1	1	-18.95
14	1	1	0	0	1	1	0	1	1	-18.29
15	0	1	0	0	0	1	1	1	1	-17.27
16	1	1	1	0	0	0	0	1	1	-1.67
17	1	1	1	0	1	0	0	1	1	-1.00
18	1	1	0	1	1	0	0	1	1	37.13
19	1	1	0	1	1	0	0	1	1	37.13
20	1	1	0	1	1	1	0	1	1	42.30

in Table 4.2. For each budget level, the 1s and 0s indicate whether the corresponding project was selected (1) or not (0), and the final column gives the optimal NPV. For example, when $b_t = \$16\text{M}$ we do not select projects 4, 5, 6, 7 but do select the others. The corresponding NPV for this budget is negative \$1.67M.

We solve model (4.1) for a range of budget values because we recognize that the budget is uncertain. Moreover, for reasons explained in the first two sections, we know STPNOC management seeks a priority list. If the sets of projects selected as we increase the budget from \$11M to \$20M are *nested*, i.e., the project portfolio at each budget level is a superset of all those at lower budget levels, then the multidimensional knapsack model yields a prioritized solution. However, as we notice from Table 4.2, some of the projects are part of the portfolio for a particular budget level but are absent from the portfolio

at higher budget levels. For instance, as we parametrically range the budget level from \$11M to \$20M, projects 1, 3, 5, 6 and 7 alternate in and out of the portfolio. This is a typical situation in knapsack problems, and more generally, in resource-constrained combinatorial optimization problems. That is, when the problem data are slightly perturbed, the new optimal solution can be far from the original optimal solution. This phenomenon represents a significant issue to decision-makers in our setting because it can result in decreased confidence in the project- selection decisions recommended by the multidimensional knapsack model.

To understand why the solutions behave in this manner, it is instructive to compare projects 4 and 7. Project 4 has a large NPV compared to project 7 (\$60M and \$4M, respectively), but it is costly, at \$23M (nominal) over three years (see Table 4.1). Hence, we would like to include project 4 in the portfolio if it fits within the available budget. This is exactly what happens for budget levels of \$18M and higher. Project 7 enters the portfolio only when its relatively low cost allows it to "just fit" within the residual budget when other more profitable projects are too costly to do so, and this is what occurs at the \$15M budget level. So, we can view project 7 as a "filler" project, funded when its cost profile happens to align well with the residual planned budget.

4.4 Heuristic Project Prioritization

Managers, including those at STPNOC, often seek a priority list as the solution to a capital budgeting problem. The "volatility" of the optimal port-

folios obtained from model (4.1), with respect to budget changes, complicates our ability to extract a priority list from the portfolios in Table 4.2. More generally, this volatility may be disconcerting to decision makers, and so we investigate an alternative that lends itself to building a priority list: As an initial simplistic approach, we begin by solving model (4.1) with $b_t = \$11\text{M}$. Then, we solve model (4.1) with $b_t = \$12\text{M}$ under the additional requirement that all projects selected at the \$11M-budget level remain in the portfolio. We continue in this way to larger budget levels. The result is a nested collection of portfolios, shown in Table 4.3, from which we easily can extract a priority list. The associated heuristic priority list consists of the following. Projects in the group {1,2,5,8} all receive top priority because they are funded for all budget levels we consider. Projects 9, 6 and 7 follow, prioritized in that order. Project 9 is funded if the budget is \$12M or above, project 6 if the budget is \$14M or above, and project 7 if the budget is \$16M or above. Finally, projects 3 and 4 received lowest priority because they are not funded even with the highest budget level of \$20M. We denote the resulting priority list $\mathcal{L} = [\{1, 2, 5, 8\}, \{9\}, \{6\}, \{7\}, \{3, 4\}]$.

The heuristic approach has placed additional restrictions on the multi-dimensional knapsack model, which were not present in the solutions obtained via the analysis presented in Table 4.2. Thus it is natural that the NPV values obtained for portfolios from the heuristic approach are not as large. The magnitude of the difference between the NPVs can be significant, particularly at the larger budget values. The intuition behind this result should be clear: As

Table 4.3: Solutions to the restricted problem that form the heuristic priority list, H-11.

Budget, b_t ($M)	1	2	3	4	5	6	7	8	9	NPV ($M)
11	1	1	0	0	1	0	0	1	0	-23.58
12	1	1	0	0	1	0	0	1	1	-23.46
13	1	1	0	0	1	0	0	1	1	-23.46
14	1	1	0	0	1	1	0	1	1	-18.29
15	1	1	0	0	1	1	0	1	1	-18.29
16	1	1	0	0	1	1	1	1	1	-14.28
17	1	1	0	0	1	1	1	1	1	-14.28
18	1	1	0	0	1	1	1	1	1	-14.28
19	1	1	0	0	1	1	1	1	1	-14.28
20	1	1	0	0	1	1	1	1	1	-14.28

we incrementally raise the budget level we continue to add projects which fit within the new budget increment. While these projects increase NPV, this incremental strategy never allows us to select the higher cost (but higher value) project 4. In practice, project 4 would likely be funded by management because its benefits are so clear. However, less extreme instances of this issue often arise for projects, and collections of projects, that fall near the cut-off point. Without the type of tool we describe, the benefits of such projects easily may be missed.

We can initialize the heuristic scheme just described at levels other than the lowest budget. Instead, we could begin by solving model (4.1) with the largest budget level, b_t = \$20M. Then, decrement the budget to b_t = \$19M, and resolve model (4.1), subject to the restriction that we can only

Algorithm 4 Greedy heuristic starting at budget scenario ω': H-ω'.

Input: $(a_i, c_{it}, b_t^\omega)$, $i \in I$, $t \in T$, $\omega \in \Omega = \{\omega_{\min}, \omega_{\min}+1, \ldots, \omega_{\max}\}$, and ω', the heuristic's initial budget scenario.
Output: Priority list for greedy heuristic H-ω', $\mathcal{L} = [\mathcal{L}_1, \ldots, \mathcal{L}_L]$. L is the number of priority levels, $\mathcal{L}_1$ denotes the highest priority projects, $\mathcal{L}_2$ denotes the second highest priority projects, etc.

Solve model (4.2) with parameters $(a_i, c_{it}, b_t = b_t^{\omega'})$, $i \in I$, $t \in T$, to obtain solution x^*.
$S^{\omega'} \leftarrow \{i \,|\, x_i^* = 1\}$.

for $\omega = \omega' + 1$ incremented to $\omega_{\max}$ **do**
  Solve model (4.2) with parameters $(a_i, c_{it}, b_t = b_t^\omega)$, $i \in I$, $t \in T$, and with additional constraint set $\{x \,|\, x_i = 1,\ i \in S^{\omega-1}\}$, to obtain solution x^*.
  $S^\omega \leftarrow \{i \,|\, x_i^* = 1\}$.
end for

for $\omega = \omega' - 1$ decremented to $\omega_{\min}$ **do**
  Solve model (4.2) with parameters $(a_i, c_{it}, b_t = b_t^\omega)$, $i \in I$, $t \in T$, and with additional constraint set $\{x \,|\, x_i = 0,\ i \notin S^{\omega+1}\}$, to obtain solution x^*.
  $S^\omega \leftarrow \{i \,|\, x_i^* = 1\}$.
end for

$j \leftarrow 0$, $S^{\omega_{\min}-1} \leftarrow \emptyset$
for $\omega = \omega_{\min}$ incremented to $\omega_{\max}$ **do**
  if $S^\omega \setminus S^{\omega-1} \neq \emptyset$ **then**
    $j \leftarrow j + 1$
    $\mathcal{L}_j \leftarrow S^\omega \setminus S^{\omega-1}$
  end if
end for
if $\cup_{\omega\in\Omega} S^\omega = I$ **then**
  $L \leftarrow j$ and $\mathcal{L} = [\mathcal{L}_1, \ldots, \mathcal{L}_L]$.
else
  $L \leftarrow j + 1$, $\mathcal{L}_L = I \setminus \cup_{\omega\in\Omega} S^\omega$, and $\mathcal{L} = [\mathcal{L}_1, \ldots, \mathcal{L}_L]$.
end if

choose projects that were present in the portfolio for b_t = \$20M. (From the last row of Table 4.2, we see this excludes from consideration projects 3 and

7.) We then decrement the budget, i.e., repeat this with b_t = \$18M, and so on. This again leads to a nesting of project portfolios at different budget levels from which we can extract a priority list. Given the incremental and decremental techniques for forcing nested portfolios, we can therefore start by solving model (4.1) at any intermediate budget level, e.g., b_t = \$17M, and then decrement to b_t = \$16M, ..., \$11M, and increment from b_t = \$17M to b_t = \$18M, ..., \$20M and finally extract a priority list. We call these *greedy* heuristics. Algorithm 4 formalizes this class of heuristics, where H-ω' corresponds to initializing the heuristic with budget scenario ω'. Algorithm 4 assumes that $\Omega = \{\omega_{\min}, \omega_{\min} + 1, ..., \omega_{\max}\}$ is a set of consecutive integers, and in our numerical example $\Omega = \{11, 12, ..., 20\}$. Of course, constructing heuristic priority lists need not be rooted in model (4.1). We could instead score the projects individually using, e.g., BIR or NPV, to form a priority list, and we label these *ranking* heuristics. We use the designations H-BIR and H-NPV to denote the ranking heuristic lists based on the projects' benefit-investment ratios and net present values, respectively.

In order to assess the quality of a priority list, we require a model of uncertainty governing the realizations of the projects' profits, costs and the yearly budgets. Our uncertainty model places a probability distribution on the realizations of these model parameters, and for the moment, we focus on budget uncertainty. The range of possible budgets used in the analyses provided in Tables 4.2 and 4.3 is large and management probably only approves capital budgets relatively close to some predefined target. To address this, we assign

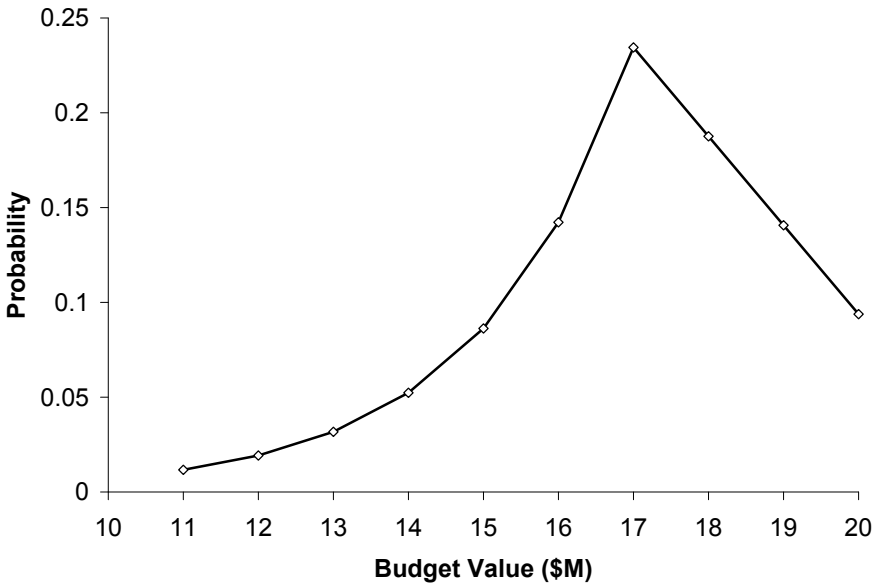

Figure 4.1: Discrete probability mass function for budget values. The line only serves to illustrate exponential and linear drops in probability mass.

relatively low probabilities to budget realizations far from the target budget. For this example, these probabilities were assigned in an *ad hoc* manner but were selected so that they represented plausible values. For our example application, we assume the most likely (target) budget value is \$17M. There is some chance that the budget is larger and the probability it is \$18M, \$19M or \$20M is assumed to drop off linearly. Conversely, the actual budget may be smaller than \$17M, and the individual probability masses are assumed to drop exponentially from \$17M to \$11M with the specific values shown in Figure 4.1

Table 4.4: Budget realizations and probabilities.

Budget, b_t^ω (\$M)	Weight	Probability, q^ω
11	$5e^{-3}$	0.012
12	$5e^{-5/2}$	0.019
13	$5e^{-2}$	0.032
14	$5e^{-3/2}$	0.052
15	$5e^{-1}$	0.086
16	$5e^{-1/2}$	0.142
17	5	0.235
18	4	0.188
19	3	0.141
20	2	0.094

Table 4.5: Heuristic priority lists and their expected NPVs.

Heuristic	Priority List	NPV (\$M)
H-11	{1, 2, 5, 8}, {9}, {6}, {7}, {3, 4}	-15.42
H-12	{1, 2, 5, 8}, {9}, {6}, {7}, {3, 4}	-15.42
H-13	{1, 2, 8, 9}, {6}, {5}, {7}, {3, 4}	-15.29
H-14	{1, 2, 8, 9}, {6}, {5}, {7}, {3, 4}	-15.29
H-15	{2, 8, 9}, {6}, {7}, {1, 5}, {3, 4}	-15.50
H-16	{1, 2, 8, 9}, {3}, {5}, {6}, {4, 7}	-4.54
H-17	{1, 2, 8, 9}, {3}, {5}, {6}, {4, 7}	-4.54
H-18	{1, 2, 5, 8}, {9}, {4}, {6}, {3, 7}	2.59
H-19	{1, 2, 5, 8}, {9}, {4}, {6}, {3, 7}	2.59
H-20	{1, 2, 5, 8}, {9}, {4}, {6}, {3, 7}	2.59
H-BIR	{1, 2}, {3}, {4, 5, 6, 7, 8, 9}	-6.89
H-NPV	{4}, {1, 2, 3, 5, 6, 7, 8, 9}	-2.40

and Table 4.4. The weights in the second column of Table 4.4 are normalized to yield the probabilities in the third column. And, the 10 budget realizations

of Table 4.4 are perfectly correlated over time, i.e., if the budget realization in year 1 is \$13M then it also takes that same value in the next four years.

Under this probability distribution we obtain an *expected* NPV of -\$15.42M by implementing the heuristic priority list, H-11. This expectation is computed by forming the weighted sum of the 10 NPV realizations in Table 4.3, using the probability mass function, q^{ω}, from Table 4.4. We compute the expected NPV under the other heuristics in a similar manner. The resulting heuristic priority lists and their NPVs are given in Table 4.5.

Examining the results in Table 4.5, we see that initializing the greedy heuristic at a larger budget value in the range \$11M to \$20M allows selection of higher-cost projects that are also of higher NPV.As indicated above, project 4 is a project that provides a large NPV but also incurs high costs. Heuristics H-11 through H-17 give project 4 the lowest possible priority because they cannot feasibly choose that project with their initial budget, and when the budget grows their existing commitments leave no room for project 4. Nominally, the ranking heuristics produce a fully-ordered list. For example, H-BIR's ordered list is simply projects 1-9, in that order. The H-BIR list given in Table 4.5 indicates that projects 1 and 2 can both be performed at the lowest budget level (\$11M), that project 3 is included at a higher budget level (which is \$16M), and that the remaining projects are unfunded, even at the highest budget level. The H-NPV prioritization is obtained by simply sorting the projects from largest to smallest NPV. From Table 4.1 we see that under this scheme project 4 has the highest priority, project 3 is ranked second, etc. Table 4.5

indicates that project 4 is funded (eventually, under the \$18M budget scenario) but that even under the highest budget realization, we cannot fund project 3.

We also compute the expected NPV under *perfect information* by weighting the NPVs in Table 4.2 to obtain \$11.90M. The expected NPV under perfect information is the value we would obtain if we could wait until the budget is realized before selecting our portfolio of projects. Comparing \$11.90M with the values obtained using the heuristics in Table 4.5 indicates that the "value of information" is significant in this problem. If we could improve the budget forecast, e.g., by further data collection, obtaining better estimates of project cost, revenue and NPV, or performing a more detailed forecasting analysis, then the ability to prioritize the projects could be improved substantially.

Some additional remarks regarding the greedy and ranking heuristics are in order. From Table 4.5, one may be tempted to conclude that greedy heuristics perform better if initialized at larger budget values. Comparing H-14 and H-15 already shows this temptation is vain, but the following example gives further insight.

Example 4.4.1. Consider an instance of a single-knapsack model with eleven projects. Ten projects have profit and cost equal to 1, and the other project has a profit of 11, and a cost of 10. The budget takes values $1, \ldots, 10$ with probabilities $q^1, \ldots, q^{10}$, which sum to one. For this problem, H-1 to H-9 all yield the same expected NPV of $\sum_{\omega=1}^{10} \omega q^\omega$, while H-10 has an expected NPV of $11q^{10}$. If the budget scenarios are equally likely, the H-10 heuristic is worse than the other heuristics by a factor of 5, and as the probability mass on the

budget realization of 10 shrinks to zero the factor by which H-10 is worse grows without bound, regardless of the specific values of $q^1, \ldots, q^9$. □

As Example 4.4.1 indicates, the fact that the greedy heuristics ignore the probability distribution means we should not anticipate, in general, one heuristic to dominate the others with respect to expected NPV. As the example also suggests, the worst-case performance of the greedy heuristics can be arbitrarily poor.

The ranking heuristics H-BIR and H-NPV ignore the budget, which points to a potential pitfall. Foremost, the highest priority project may fail to satisfy the budget under one or more (even all!) of the scenarios. In this case, until the budget scenario climbs to a level where that project can be funded, we cannot perform any other project. As described above, this is exactly what occurred for H-NPV in Table 4.5 where no project was funded for budget realizations \$11-17M. Even when this feasibility issue does not arise, a potential pitfall remains as the following example illustrates.

Example 4.4.2. Consider a single-knapsack problem instance with two projects. The first project's cost and profit are 2 and 4, respectively. The second project's cost and profit are M, a large number. There are two budget scenarios: M and $M + 1$. The expected NPV obtained under H-BIR is 4, whereas the optimal solution has an expected NPV of M. As M grows, the ratio of the optimal priority list's expected NPV to that of H-BIR grows without bound. □

These simple examples show that commonly-employed heuristics that ignore either the budgets, or their likelihood of occurrence, can lead to arbitrarily poor priority lists. The next section formulates a new model which yields an optimal priority list by incorporating a probability distribution governing the budget, cost and profit realizations.

4.5 Optimal Project Prioritization

The deterministic capital budgeting model (4.1) assumes that we know the problem data (i.e., project costs and revenues and the budget) in advance with certainty. And, as was demonstrated, the model does not naturally produce a priority list. In the previous section, we used the deterministic model to deal with uncertain budgets, but that analysis was admittedly *ad hoc* and is why we referred to the results as heuristic priority lists. In this section we build a model that explicitly incorporates multiple budget, cost and revenue scenarios.

The need to deal with these uncertain parameters motivates extending model (4.1) to form a priority list with the goal of maximizing the expected NPV of the projects we can implement after the uncertain parameters are revealed. The notation and formulation of the optimal prioritization model

are as follows:

Indices and sets:

$i, i' \in I$	candidate projects
$p \in P$	priorities; $P = \{1, 2, \ldots, \lvert I \rvert\}$
$t \in T$	time periods (years)
$\omega \in \Omega$	scenarios

Data:

a_i^ω	net present value of project i under scenario ω
b_t^ω	available budget in period t under scenario ω
c_{it}^ω	cost of project i in period t under scenario ω
q^ω	probability of scenario ω

Decision variables:

$s_{ii'}$	1 if project i has higher priority than i'; 0 otherwise
z_{ip}	1 if project i is assigned priority level p; 0 otherwise
x_i^ω	1 if project i is selected under scenario ω; 0 otherwise

Formulation:

$$\max_{\mathbf{s},\mathbf{z},\mathbf{x}} \sum_{\omega\in\Omega} q^{\omega} \sum_{i\in I} a_i^{\omega} x_i^{\omega} \tag{4.2a}$$

$$\text{s.t.} \sum_{i\in I} c_{it}^{\omega} x_i^{\omega} \le b_t^{\omega}, \qquad t\in T,\ \ \omega\in\Omega, \tag{4.2b}$$

$$\sum_{i\in I} z_{ip} = 1, \qquad p\in P, \tag{4.2c}$$

$$\sum_{p\in P} z_{ip} = 1, \qquad i\in I, \tag{4.2d}$$

$$|P| s_{ii'} \ge \sum_{p\in P} (|P|-p)(z_{ip} - z_{i'p}), \qquad i\neq i', i, i'\in I, \tag{4.2e}$$

$$s_{ii'} + s_{i'i} = 1, \qquad i < i',\ \ i, i'\in I, \tag{4.2f}$$

$$x_i^{\omega} \ge x_{i'}^{\omega} + s_{ii'} - 1, \qquad i\neq i', i, i'\in I, \omega\in\Omega, \tag{4.2g}$$

$$x_i^{\omega} \in \{0,1\}, \qquad i\in I, \omega\in\Omega, \tag{4.2h}$$

$$s_{ii'} \in \{0,1\}, \qquad i\neq i', i, i'\in I, \tag{4.2i}$$

$$z_{ip} \in \{0,1\}, \qquad i\in I, p\in P. \tag{4.2j}$$

Model (4.2) is an application of model (2.9) to multidimensional knapsack problem (4.1). All constraints and variables read similarly, and hence we do not give details. Although model (4.2) prioritizes a relatively simple multidimensional knapsack model, we can similarly prioritize any resource-constrained combinatorial optimization problem with binary activity selection decisions (See models (2.3) and (2.14)). In the context of capital budgeting, a more detailed model would capture important issues such as: Selecting one project can yield a synergistic opportunity for other projects; selection of a project may require selection of one or more prerequisite projects; a collec-

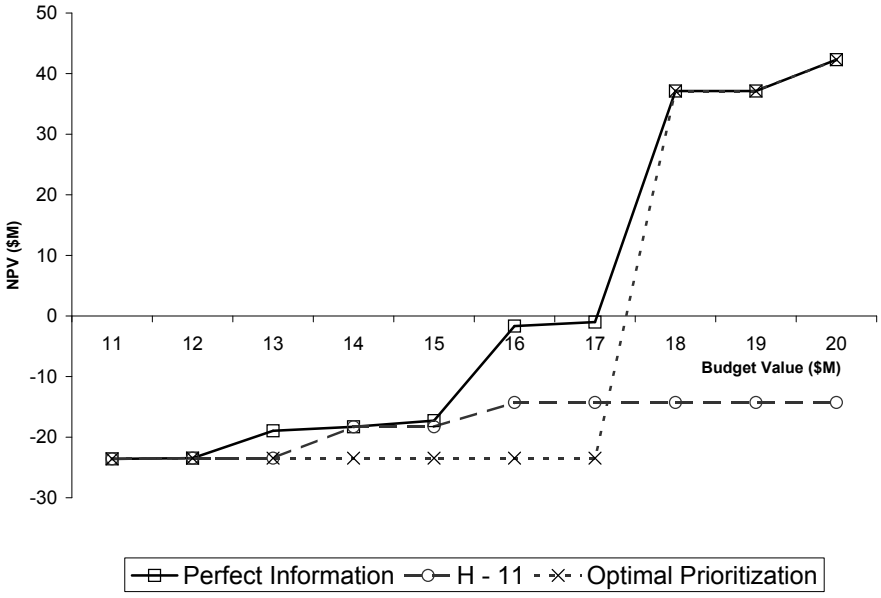

Figure 4.2: Obtained NPVs for each budget realization under perfect information, the H-11 heuristic priority list and the optimal priority list. The y-axis is NPV ($M) and x-axis is budget level ($M).

tion of projects may represent mutually exclusive alternatives; some projects are implemented in phases, with opportunities for acceleration or delay; colors, beyond yearly availability, can distinguish types of money; both fixed and variable costs play a role in asset replacement models; resources in addition to monetary budgets can limit project selection; and, demand constraints can drive project selection. See, for instance, Brown et al. [6] and Hartman [15] for a discussion of these and related issues. For capital budgeting under uncer-

tainty, Meier et al. [28] couple a knapsack model with contingent-claim analysis, in place of traditional cash-flow analysis. Through a proper partitioning of the uncertainties, their model is also amenable to prioritization.

To illustrate the optimal prioritization model, we continue our example from the previous two sections in which only the budget is random, and we have nine projects with positive NPV. While our illustrative example only has uncertainty in the budget level, model (4.2) also handles uncertainty in the profits and costs of the projects. The next section considers a larger problem with 41 projects, including an instance in which the budget, cost and profit parameters are modeled as random variables with defined distribution functions. We use the same values for a_i and c_{it} as given in Table 4.1. As in the previous section, we use the budget realizations, b_t^ω, and associated probabilities, q^ω, given in Table 4.4. The priority list solving this instance of model (4.2) is given in Table 4.6. The priority lists for a number of the greedy heuristics are also given for reference. As is shown in the table, the list obtained by the heuristics H-18, H-19 and H-20 find the optimal priority list, confirmed by the solution of model (4.2). The other heuristics yield substantially inferior lists, i.e., they yield a substantially lower expected NPV. It is, perhaps, not surprising that when prioritizing a small set of candidate projects (in this case consisting of only 9 projects) using multiple heuristics (in this case 12 different heuristics; see Table 4.5) that some of those heuristics find the optimal solution. This result is also not surprising given the dominant nature of project 4, and the fact that the heuristics H-18, H-19 and H-20 have sufficient budget to

include that project.

We can obtain further insight by comparing the optimal priority list to that obtained by the heuristics H-11 and H-12. Comparing these two lists in Table 4.6, the only difference is that in the optimal list, the largest NPV project 4 has higher priority. Despite this simple difference, it is interesting to note that priority lists obtained by intermediate heuristics, e.g., H-15 and H-17, have a somewhat different structure. In H-15, projects 1 and 5 move substantially down the list. H-17 is arguably a natural heuristic to run in this setting because it first solves model (4.1) under the most-likely budget scenario. Still, its expected NPV is \$7.13M short of optimal.

The performance of the optimal priority list under each budget scenario is given in Table 4.7. Comparing this with the results of the H-11 heuristic for each scenario in Table 4.3, we see that the optimal priority list underperforms the greedy heuristic (by a relatively small amount) for budget values of \$14M-17M. However, for larger amounts of available budget, the stochastic approach significantly outperforms the heuristic priority list, as seen by the resultant portfolio NPVs obtained for the budget values between \$18M-20M. The optimal expected NPV from the prioritization problem of \$2.59M is (necessarily) at least as large as that of all the heuristics in Table 4.5, and, of course, smaller than that under perfect information (\$11.90M). Figure 4.2 compares the NPVs for each budget scenario for H-11, the optimal priority list, and perfect information. The expected NPV obtained under H-11 (-\$15.42M), the optimal prioritization (\$2.59M) and perfect information (\$11.90M) can

Table 4.6: Priority lists from greedy heuristics and model (4.2), with their expected NPVs.

H-11, H-12		H-15		H-17		H-18—H-20, Optimal	
Priority	Project	Priority	Project	Priority	Project	Priority	Project
1-4	{1,2,5,8}	1-3	{2, 8, 9}	1-4	{1,2,8,9}	1-4	{1,2,5,8}
		4	6				
5	9	5	7	5	3	5	9
6	6	6-7	{1,5}	6	5	6	4
7	7			7	6	7	6
8-9	{3,4}	8-9	{3,4}	8-9	{4,7}	8-9	{3,7}
-$15.42M		-$15.50M		-$4.54M		$2.59M	

be obtained from Figure 4.2 by weighing the respective points for each budget realization by the probabilities from Figure 4.1 and summing. Again, as the figure shows, the heuristic priority list outperforms the optimal list under some budget scenarios ($14M-17M), but not in the overall expected value of the NPV due to its large underperformance for budget values $18M-20M.

With the small number of projects in this example, we likely could have obtained the optimal priority list by trial-and-error, or even brute force, as the total number of possible priority lists is modest (i.e., there are 9!=362,880 possible orderings of the 9 projects). This small-sized example is useful because the behavior of the optimal solution is transparent. However, as we show in the next section, the prioritization model can produce similar results when the number of projects is larger and it is impossible to exhaustively examine all such alternatives.

Before turning to this larger numerical example, we note that from

Table 4.7: Solution to the 9-project prioritization problem.

Budget Level ($M)	1	2	3	4	5	6	7	8	9	NPV ($M)
11	1	1	0	0	1	0	0	1	0	-23.58
12	1	1	0	0	1	0	0	1	1	-23.46
13	1	1	0	0	1	0	0	1	1	-23.46
14	1	1	0	0	1	0	0	1	1	-23.46
15	1	1	0	0	1	0	0	1	1	-23.46
16	1	1	0	0	1	0	0	1	1	-23.46
17	1	1	0	0	1	0	0	1	1	-23.46
18	1	1	0	1	1	0	0	1	1	37.13
19	1	1	0	1	1	0	0	1	1	37.13
20	1	1	0	1	1	1	0	1	1	42.30

Table 4.6 one may be tempted to infer that the best of the greedy heuristics obtains an optimal, or near-optimal solution, as H-18-H-20 do in this example. However, the following example shows that the factor by which the expected NPV of the optimal prioritization outperforms that of the best of the greedy heuristics can be arbitrarily large. In other words, we do not have any such performance guarantee for these greedy heuristics.

Example 4.5.1. Let k be a positive integer and consider an instance of a two-dimensional knapsack problem with projects indexed by $I = \{1, 2, \ldots, 2k\}$, and scenarios indexed by $\Omega = \{1, 2, \ldots, k\}$. Projects $i = 1, \ldots, k$ have profit/cost streams of $M/(1,1), 2M/(2,1), \cdots, kM/(k,1)$, where the pair $(i,1)$ represents the first- and second-year project costs, respectively. The remaining projects $i = k+1, \ldots, 2k$ are all identical and have a profit of $M-1$ and

costs of $(1, 1/k)$. The only uncertainty lies in the budget. In scenario ω there is a budget of $(\omega, 1)$ for $\omega = 1, \ldots, k$. The probabilities of these scenarios are $q^{\omega} = \frac{1}{\omega(\omega+1)}$ for $\omega = 1, \ldots, k-1$, and the last scenario has a probability of $\frac{1}{k}$.

When M is sufficiently large, it is not difficult to see that the H-ω greedy heuristic, for $\omega = 1, \ldots, k$, begins by choosing project $i = \omega$ since the optimal solution of the two-dimensional knapsack problem with budget $(\omega, 1)$ is project $i = \omega$. In all scenarios $\omega' < \omega$, no project is selected under H-ω. And, for scenarios $\omega' > \omega$, project $i = \omega$ remains the only project selected because it exhausts the second-year budget. Hence, the objective function value under the H-ω heuristic is given by:

$$\sum_{\omega'=\omega}^{k} (\omega M) q^{\omega'} = (\omega M) \left(\frac{1}{k} + \sum_{\omega'=\omega}^{k-1} \frac{1}{\omega'(\omega'+1)} \right) = (\omega M) \left(\frac{1}{k} + \frac{1}{\omega} - \frac{1}{k} \right) = M.$$

On the other hand, the solution to the optimal prioritization model (4.2) selects project $k+1$ under the first scenario, selects projects $k+1$ and $k+2$ under the second, and so forth until it selects projects $k+1, \ldots, 2k$ under the last scenario. This optimal solution's objective function value is:

$$\sum_{\omega=1}^{k} \omega(M-1) q^{\omega} = k(M-1)\frac{1}{k} + \sum_{\omega=1}^{k-1} \frac{\omega(M-1)}{\omega(\omega+1)} = (M-1) \sum_{\omega=1}^{k} \frac{1}{\omega}.$$

In this example, the H-ω heuristics all perform identically. Thus the ratio of the expected NPV from the best of the H-ω heuristics to that of the optimal prioritization is $\dfrac{M}{(M-1)\sum_{\omega=1}^{k} \frac{1}{\omega}}$. As M grows large, this ratio converges to

$\left(\sum_{\omega=1}^{k} \frac{1}{\omega}\right)^{-1}$. And finally, as k grows large, this ratio shrinks to zero, meaning that the factor by which the optimal prioritization outperforms the best of the H-ω heuristics grows without bound. □

4.6 A Problem with More Projects

In this section, we consider a larger problem with 41 projects, all of which have positive NPV. As in the smaller problem considered in the previous sections, all of the project data, including profits and cost streams, are from STPNOC, with the cost stream estimates covering the next five years. Point forecasts for the project profits and cost streams (nominal) over the next five years are given in Table 4.8, along with the benefit-investment ratio for each project. The NPV and cost-stream values are of similar magnitude to those reported in Table 4.1, but Table 4.8 reports both the c_{it} and a_i values as percentages of the total NPV summed over all 41 projects due to business sensitivity. So, the sum of the 41 entries in the NPV (a_i) column is 100. As before, a blank entry for a c_{it} value means that project i does not incur a cost in year t.

Table 4.8: Data for 41 projects. Both the cost and NPV values are given as percentages of total NPV summed over all 41 projects.

	yearly costs c_{it}						
Project i	1	2	3	4	5	NPV (a_i)	BIR
1		0.07	0.08	0.01	0.01	12.71	96.612
2		0.09				3.10	38.769
3	0.02					0.40	20.855
4	0.04					0.66	16.009
5	0.21					3.12	14.950
6	0.29					2.86	10.006
7	0.34					3.35	9.973
8		0.33	0.22			4.63	9.499
9	0.08	0.08	0.08			2.11	9.267
10		0.06				0.36	6.973
11	0.04					0.26	6.237
12	0.11	0.09				1.17	6.016
13	0.22	0.21				2.21	5.350
14	0.27	0.53	0.11			4.32	5.090
15	0.18	0.21	0.07			1.94	4.484
16				0.29	0.45	1.71	3.159
17	0.93	1.30	0.35			7.37	3.055
18		0.12	0.31	0.07		1.11	2.661
19		0.08	0.19	0.12		0.77	2.342
20			0.13	0.29	0.15	1.02	2.339
21		0.08	0.19	0.12		0.76	2.315
22			2.24	6.46	4.81	17.72	1.735
23	0.36					0.56	1.568
24	1.78	1.78				4.34	1.272
25			0.17	0.78	0.07	0.98	1.231
26			0.27	4.54	13.96	16.26	1.204
27	0.08	0.17	0.17	0.17	0.17	0.75	1.196
28		0.29	0.29			0.59	1.141
29	0.13	0.13	0.13	0.13	0.13	0.44	0.784
30				0.10	0.29	0.21	0.740
31		0.80				0.49	0.669
32	0.34					0.23	0.662
33	0.13	0.17	0.13	0.13	0.13	0.32	0.568
34				0.52	0.75	0.33	0.359
35	0.20				0.01	0.07	0.339
36	0.13					0.04	0.301
37		0.61	1.08	0.42	0.66	0.50	0.222
38	0.15	0.36				0.10	0.208
39	0.12	0.04				0.03	0.201
40	0.04					0.01	0.158
41				0.45	0.81	0.09	0.101

We analyze two problem instances based on these projects. In the first instance, only the budget is uncertain and we consider 10 budget scenarios over the range of \$2.5M to \$7M in increments of \$0.5M. As in the previous sections, if the budget scenario takes a value, say \$4.5M, in year 1, then it takes that same value in years 2-5. These yearly budget values range from about 2% to 6% of the total NPV. The probability weights we place on these 10 scenarios are those given in the third column of Table 4.4. That is, the probability of having a budget of \$2.5M is 0.012, that of having \$3M is 0.019, and so on.

Solving this instance of model (4.2), we obtain the desired priority list. For comparison we also obtain priority lists using the greedy heuristics H-2.5 through H-7. Finally, we also use the H-BIR and H-NPV heuristics. Figure 4.3 plots the NPV of the selected projects as a function of the budget realization for this problem instance for the prioritization model (4.2), the best (H-4.5) and worst (H-2.5) performing greedy heuristic procedures, the BIR heuristic, and the NPV under perfect information. These NPV results are reported as a percentage of total NPV, summed over all projects.

The expected NPV of each method is given in Table 4.9, again as a percentage of total. In our earlier small test problem the heuristics initialized at the largest budget realizations performed best. Here, however, the heuristic initialized at either extreme (H-2.5 or H-7) performs poorly relative to the optimal list and relative to the heuristic initialized near the most-likely budget scenario (H-5.5). We note that this emphasizes the conclusion presented in the

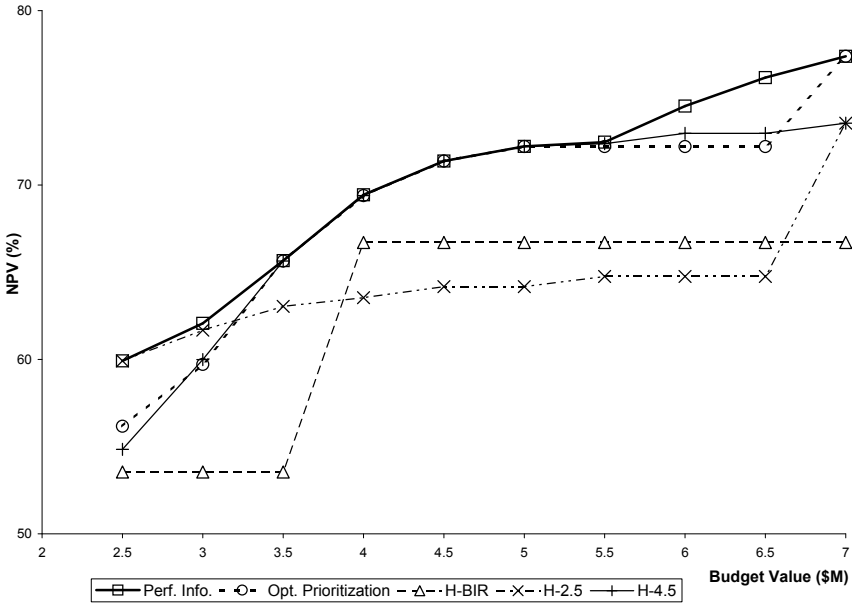

Figure 4.3: Replication of Figure 4.2 for 41-project problem under budget uncertainty.

previous section that use of the greedy heuristics is not guaranteed to obtain an optimal, or near-optimal solution. We also note that the H-NPV has an expected NPV of zero because the highest NPV project is so costly that it cannot be implemented even under the highest budget scenario we consider.

Finally, we turn to a problem instance in which the cost streams also are uncertain. As described in the background section, for each project STP-NOC forecasts a pessimistic, optimistic and most-likely cost stream, and these in turn yield three forecasts for each project's NPV. In the 41 projects we con-

Table 4.9: Procedures for 41-project problem instance and their expected NPVs (%).

Procedure	Perf. Info.	Optimal	H-2.5	H-3	H-3.5	H-4	H-4.5
NPV (%)	61.20	60.25	54.69	54.69	60.16	60.16	60.17
Procedure	H-5	H-5.5	H-6	H-6.5	H-7	H-BIR	H-NPV
NPV (%)	60.17	60.14	58.83	57.77	57.77	55.25	0.00

sider, there are two types of projects, those labeled low-risk and those labeled medium-risk (i.e., none of the projects were classified as high risk). For a low-risk project, its cost and profit are assigned the pessimistic NPV value with probability 1/6, the optimistic value with probability 1/6 and the most-likely value with probability 4/6. For medium-risk projects these three respective probability masses are instead 2/6, 1/6 and 3/6. (For completeness we note that for high-risk projects, these respective probability masses are 3/6, 1/6 and 2/6.) These weights reflect estimates based on STPNOC's experience.

The point forecasts for the cost-streams and the profits are the same as those used above, i.e., those given in Table 4.8. We give the uncertainties in the costs and profits in terms of multipliers. Consider the multipliers in Table 4.10, corresponding to the pessimistic, optimistic and most-likely cost streams and profits. Profit (cost) of a project under the pessimistic, optimistic or most-likely scenario is its point forecast times the corresponding pessimistic multiplier in Table 4.10. That table also contains the low-risk or medium-risk label for each project. In our analysis, we assume that the project's cost and NPV are perfectly correlated, e.g., if the cost stream takes the pessimistic

realization, then so does the NPV. We further assume that all projects with the same risk label are perfectly correlated. So, all medium-risk projects either take the pessimistic, optimistic or most-likely realization. The same holds for projects within the low-risk category. However, the low-risk and medium-risk projects are assumed to behave independently. Each risk group thus has three realizations and the two groups are independent; hence there are a total of 9 scenarios governing the cost-profit uncertainty. In addition to this uncertainty, we also have budgetary uncertainty, which is modeled as unfolding independently of the cost-profit uncertainty. We use the same 10 budget scenarios described previously. This results in a total of $|\Omega| = 90$ scenarios.

The greedy heuristic must be altered slightly to deal with a problem instance that contains cost uncertainty. When there is only budgetary uncertainty, the scenarios can be ordered, and hence we can naturally produce a nested set of portfolios that, in turn, yields a priority list. This is not possible when costs, profits and budgets are all uncertain. So, we instead use the average cost estimate and then form the greedy priority lists. Still, as is shown in Tables 4.5 and 4.6, these heuristics produce a partially-ordered priority list. For example, under a number of the greedy heuristics for the 9-project problem, projects {1,2,5,8} all receive top priority. We break the ties within this partial ordering using BIR, and this allows us to produce the fully-ordered priority lists that are required to compute the expected NPV under cost uncertainty.

Table 4.10: Multiplier factors for 41 projects under pessimistic, optimistic and most-likely scenarios.

		Cost factors			NPV factors		
Project i	Risk Level	Pess.	Opt.	M.L.	Pess.	Opt.	M.L.
1	L	1.131	0.880	0.997	0.999	1.001	1.000
2	M	1.104	0.912	0.960	0.997	1.002	1.001
3	L	1.171	0.927	0.976	0.993	1.004	1.001
4	L	1.100	0.900	1.000	0.994	1.007	1.000
5	L	1.000	1.000	1.000	1.000	1.000	1.000
6	L	1.050	0.950	1.000	0.995	1.005	1.000
7	M	1.200	0.720	0.960	0.980	1.036	1.001
8	L	1.131	0.934	0.984	0.987	1.007	1.001
9	L	1.050	0.950	1.000	0.995	1.006	1.000
10	L	1.100	0.900	1.000	0.986	1.015	1.000
11	L	1.000	1.000	1.000	1.000	1.000	1.000
12	M	1.134	0.898	0.945	0.979	1.017	1.008
13	L	1.046	0.961	0.998	0.992	1.007	1.000
14	L	1.000	1.000	1.000	1.000	1.000	1.000
15	L	1.082	0.984	0.984	0.983	1.003	1.003
16	M	1.104	0.912	0.960	0.968	1.029	1.011
17	L	1.127	0.872	1.000	0.961	1.046	0.998
18	M	1.104	0.912	0.960	0.963	1.034	1.014
19	M	1.104	0.912	0.960	0.958	1.039	1.015
20	M	1.104	0.912	0.960	0.957	1.039	1.015
21	M	1.104	0.912	0.960	0.957	1.039	1.016
22	M	1.104	0.912	0.960	0.943	1.052	1.021
23	L	1.000	1.000	1.000	1.000	1.000	1.000
24	M	1.067	0.932	0.978	0.949	1.055	1.016
25	M	1.104	0.912	0.960	0.919	1.074	1.029
26	M	1.086	0.897	0.977	0.931	1.091	1.016
27	L	1.000	1.000	1.000	1.000	1.000	1.000
28	L	1.138	0.933	0.982	0.891	1.059	1.012
29	L	1.000	1.000	1.000	1.000	1.000	1.000
30	M	1.104	0.912	0.960	0.866	1.122	1.048
31	L	1.050	0.950	1.000	0.928	1.077	0.999
32	L	1.000	1.000	1.000	1.000	1.000	1.000
33	L	1.000	1.000	1.000	1.000	1.000	1.000
34	L	1.131	0.934	0.984	0.670	1.184	1.036
35	L	1.050	0.950	1.000	0.857	1.152	0.998
36	L	1.000	1.000	1.000	1.000	1.000	1.000
37	L	1.140	0.893	0.992	0.435	1.508	1.014
38	M	1.073	0.927	0.976	0.663	1.360	1.105
39	L	1.157	0.801	1.010	0.306	2.112	0.895
40	L	1.000	1.000	1.000	1.000	1.000	1.000
41	M	1.104	0.912	0.960	0.067	1.851	1.338

The problem instance with cost uncertainty is almost an order of magnitude larger than the 41-project problem with just 10 budget scenarios. We solve this stochastic integer program to within 1% of optimality, again using CPLEX version 10.1 [8]. The near-optimal priority list we find has an expected NPV of 60.18%. The respective expected NPV results for the heuristics H-2.5, H-5.5 and H-7 are 54.65%, 60.04% and 57.04%. (The expected NPVs of these priority lists are given as percentages of the total NPVs summed over all 41 candidate projects.) We report H-5.5 because it is the heuristic initialized at the most-likely scenario and because, of all the greedy heuristics, it yielded the highest expected NPV. The H-BIR heuristic yielded an expected NPV of 55.27%, and H-NPV again could not implement any projects because of the top priority being given to an excessively costly project. Finally, the expected NPV under perfect information is 61.19%.

As in our smaller computational example using data from STPNOC, sometimes a heuristic performs well. However, predicting which, if any, heuristic will perform well can be difficult. One can avoid the need for our prioritization model (4.2) if the gap between a heuristics expected NPV and the expected NPV under perfect information is small. This gap is a *posterior* bound in that it can be computed only once the problem data are known. In the 90-scenario instance just considered, the *relative* gap between the expected NPV of the H-5.5 heuristic and that under perfect information is 1.9%, and if that is deemed sufficiently small, we can employ the priority list from the H-5.5 heuristic. However, in other problem instances (e.g., our 9-project in-

stance) this gap is large. In such cases, we recommend applying our optimal prioritization model (4.2).

Chapter 5

Conclusions and Future Work

Resource-constrained activity selection problems have applications in many real-life decision making activities, such as capital budgeting and facility location problems. The current literature in operations research frequently approaches these problems by forming an optimal portfolio of activities. Practitioners in industry and government instead often form a priority list of activities and select those that have the highest priority. Considering both viewpoints, we propose a new prioritization approach that prioritizes the activities recognizing structural and stochastic dependencies among them.

In Chapter 1, we motivate our prioritization approach, and in Chapter 2, we provide several mathematical programming tools to apply this approach to resource-constrained activity selection problems. Namely, we first give a formulation that takes an activity-prioritization perspective on our approach and compare it with a fully-ordered priority list. We then give a formulation that takes a scenario-prioritization perspective and show that each of the two perspectives can be more efficient than the other depending on the problem parameters. We also develop two sets of cutting planes for both formulations and show their computational use. The formulations in this chapter

are generic, i.e., we mean that apply generally to resource-constrained activity selection problems. An important research direction related to this chapter is to develop special purpose prioritization algorithms, i.e., algorithms that exploit special structures in particular problems, providing more efficient solution approaches than our more general formulation. One other research direction relates to prioritizing the k-median problem. The computational instances we discuss in Chapters 1 and 2 assume equal probabilities for each realization of k. It would be insightful to investigate threshold levels for these probabilities at which there are structural changes in the optimal prioritization.

Real-life problems are usually large in scale and the formulations in Chapter 2 sometimes fail to solve such larger problem instances. In Chapter 3, we develop a branch-and-price decomposition algorithm for an application of our prioritization approach to the multidimensional knapsack problem. Specifically, we construct a column-based reformulation, develop two branching strategies, and propose a tabu-search-based primal heuristic. We also propose two parallelization schemes for our branch-and-price algorithm: inter-node and intra-node parallelization. Comparing the two parallelization, we see that inter-node parallelization tends to perform better on large branch-and-bound trees and intra-node parallelization performs better on small branch-and-bound trees. Several research directions may extend the study of this chapter. One is to develop an alternative column-based reformulation that better lends itself to intra-node parallelization. The second is to develop a hybrid parallelization that uses the ideas in both parallelization approaches.

The third one relates to the tabu-search heuristic. It is worth experimenting with the parameters of the heuristic to improve its performance and testing it on a wider variety of problem instances.

In Chapter 4, we discuss the practical advantages of our approach and present a case study on its application to a real-life problem. We rank STPNOC's nuclear-maintenance and capital-improvement projects, considering uncertainties in the profits and cost flows of the projects and in the yearly budgets. We compare our approach with several heuristic ranking schemes. In this chapter, we use STPNOC's optimistic, pessimistic and most-likely scenarios to model the uncertainty in the project costs and profits. One direction of future research is to investigate the effect of various other characterizations of the uncertainty on the relative priorities of items, and to experiment and see if these different characterizations have any impact on the computational time to solve the project prioritization problem.

Bibliography

[1] Balinski, M. L., 1970. Selection problem. Management Science Series A–Theory 17, 230–231.

[2] Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., Vance, P. H., 1998. Branch-and-price: Column generation for solving huge integer programs. Operations Research 46, 316–329.

[3] Beasley, J. E., May 2010. OR-Library. `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`

[4] Bierman, H., Smidt, S., 1980. The Capital Budgeting Decision: Economic Analysis of Investment Projects, 5th Edition. Macmillan Publishing Co., New York.

[5] Brown, G. G., Carlyle, M., Salmerón, J., Wood, R. K., 2006. Defending critical infrastructure. Interfaces 36, 530–544.

[6] Brown, G. G., Dell, R. F., Newman, A. M., 2004. Optimizing military capital budgeting. Interfaces 34, 415–425.

[7] COIN-OR, May 2010. Computational Infrastructure for Operations Research. COIN-OR Foundation, Inc., `http://www.coin-or.org`

[8] CPLEX, May 2010. IBM ILOG. http://www-01.ibm.com/software/integration/optimization/cplex

[9] Dean, B., Goemans, M. X., Vondrák, J., 2005. Adaptivity and approximation for stochastic packing problems. In: Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms. Vancouver, British Columbia, pp. 395–404.

[10] Dean, B., Goemans, M. X., Vondrák, J., 2008. Approximating the stochastic knapsack problem: the benefit of adaptivity. Mathematics of Operations Research 33, 945–964.

[11] Desaulniers, G., Desrosiers, J., Solomon, M. M., 2005. Column Generation. Springer Science, New York.

[12] Feo, T. A., Resende, M. G. C., 1995. Greedy randomized adaptive search procedures. Journal of Global Optimization 6, 109–133.

[13] Gendron, B., Crainic, T. G., 1994. Parallel branch-and-bound algorithms: Survey and synthesis. Operations Research 42, 1042–1066.

[14] Glover, F. W., Laguna, M., 1997. Tabu Search. Kluwer Academic Publishers, Boston.

[15] Hartman, J. C., 2000. The parallel replacement problem with demand and capital budgeting constraints. Naval Research Logistics 47, 40–56.

[16] Hochbaum, D. S., 2009. Dynamic evolution of economically preferred facilities. European Journal of Operational Research 193, 649–659.

[17] Israeli, E., Wood, R. K., 2002. Shortest-path network in interdiction. Networks 40, 97–111.

[18] Karp, A. H., Flatt, H. P., 1990. Measuring parallel processor performance. Communications of the ACM 33, 539–543.

[19] Keeney, R. L., Raiffa, H., 1976. Decisions with Multiple Objectives: Preferences and Value Tradeoffs. John Wiley & Sons, New York.

[20] Kellerer, H., Pferschy, U., Pisinger, D., 2004. Knapsack Problems. Springer-Verlag, Heidelberg.

[21] Koç, A., Morton, D., Popova, E., Hess, S., Kee, E., Richards, D., May 11-15 2008. Optimizing project prioritization under budget uncertainty. In: Proceedings of ICONE 16: ASME 2008 International Conference on Nuclear Engineering. Orlando, Florida USA.

[22] Koç, A., Morton, D., Popova, E., Hess, S. M., Kee, E., Richards, D., 2009. Prioritizing project selection. The Engineering Economist 54, 267–297.

[23] Koç, A., Morton, D., Popova, E., Kee, E., Richards, D., Sun, A., Hess, S., July 22-26 2007. Project prioritization via optimization. In: Proceedings of PVP 2007/CREEP 8 ASME PVP 2007/CREEP 8 Conference. San Antonio, Texas USA.

[24] Larsen, J., 2004. Refinements of the column generation process for the vehicle routing problem with time windows. Journal of Systems Science and Systems Engineering 13, 326–341.

[25] Lin, G., Nagarajan, C., Rajaraman, R., Williamson, D. P., 2006. A general approach for incremental approximation and hierarchical clustering. In: Proceedings of the 17th ACM-SIAM Symposium on Discrete Algorithms. Miami, Florida, pp. 1147–1156.

[26] Lübbecke, M. E., Desrosiers, J., 2005. Selected topics in column generation. Operations Research 53, 1007–1023.

[27] Lulli, G., Sen, S., 2004. A branch-and-price algorithm for multistage stochastic integer programming with application to stochastic batch-sizing problems. Management Science 50, 786–796.

[28] Meier, H., Christofides, N., Salkin, G., 2001. Capital budgeting under uncertainty - an integrated approach using contingent claims analysis and integer programming. Operations Research 49, 196–206.

[29] Mettu, R. R., Plaxton, C. G., 2003. The online median problem. SIAM Journal on Computing 32, 816–832.

[30] Mirchandani, P. B., Francis, R. L., 1990. Discrete Location Theory. John Wiley & Sons, Inc., USA.

[31] Nehme, M. V., Morton, D. P., 2010. Efficient nested solutions of the bipartite network interdiction. In: Proceedings of the IIE Annual Conference. Cancun, Mexico, 2010.

[32] Pacheco, P., 1997. Parallel Programming with MPI. Morgan Kaufmann Publishers, San Francisco.

[33] Plaxton, C. G., 2006. Approximation algorithms for hierarchical location problems. Journal of Computer and System Sciences 72, 425–443.

[34] Ralphs, T. K., Ladányi, L., January 2001. COIN/BCP Users Manual. COIN-OR Foundation, Inc., `http://www.coin-or.org/Presentations/bcp-man.pdf`

[35] Ralphs, T. K., Ladányi, L., Saltzman, M. J., 2003. Parallel branch, cut, and price for large-scale discrete optimization. Mathematical Programming 98, 253–280.

[36] Rhys, J. M. W., 1970. Selection problem of shared fixed costs and network flows. Management Science Series A– Theory 17, 200–207.

[37] Rosen, K. H., 2006. Discrete Mathematics and Its Applications. McGraw-Hill, Inc., New York.

[38] Ross, S. A., Westerfield, R. W., Jordan, B. D., 2008. Fundamentals of Corporate Finance, 8th Edition. McGraw-Hill, Boston.

[39] Savage, S., Scholtes, S., Zweidler, D., 2006. Probability management. ORMS Today 33, 20–28.

[40] Shiina, T., Birge, J. R., 2004. Stochastic unit commitment problem. International Transactions in Operational Research 11, 19–32.

[41] Silva, E. F., Wood, R. K., 2006. Solving a class of stochastic mixed-integer programs with branch and price. Mathematical Programming 108, 395–418.

[42] Singh, K. J., Philpott, A. B., Wood, R. K., 2009. Dantzig-Wolfe decomposition for solving multistage stochastic capacity-planning problems.

[43] STPNOC, May 2010. South Texas Project Nuclear Operating Company. http://www.stpnoc.com

[44] TACC, May 2010. Texas Advanced Computing Center. Dell Duo-Core Linux Cluster User Guide. http://www.tacc.utexas.edu/services/userguides/lonestar/

[45] Weingartner, H. M., 1966. Capital budgeting of interrelated projects: survey and synthesis. Management Science 12, 485–516.

Vita

Ali Koç was born in İstanbul, Turkiye on 15 September 1980, the son of Ayşe and Süleyman Koç. He received his B.Sc. and M.Sc. degrees in Industrial Engineering from Bilkent University, Ankara, Turkiye, in 2002 and 2005. As a part of his undergraduate studies, he worked as a co-op student at Unilever Turkiye in 2001-2002. During his graduate studies at Bilkent, he performed research on assembly and disassembly line balancing problems. In the Fall 2005, he joined the Graduate Program in Operations Research and Industrial Engineering at the University of Texas at Austin as a Ph.D. student. During his doctoral study, he worked on resource-constrained activity selection problems under uncertainty with Dr. David P. Morton and Dr. Elmira Popova. In the summer of 2007, he interned at the San Diego Supercomputing Center, San Diego, California, to use high-performance and parallel computing in his dissertation research. In the summer of 2008, he interned at Freescale Semiconductor Inc. Ali joined the Business Analytics and Mathematical Sciences Department at IBM T.J. Watson Research Center as a postdoctoral researcher in March 2010. He is currently a postdoctoral researcher at IBM T.J. Watson Research Center performing research on optimization of electric power generation and distribution.

Permanent address: Ataturk Mahallesi, Yildiz Sokak, No: 5/1,
Umraniye, Istanbul, TURKİYE

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.