Multi-Stage Stochastic Programming Models for Provisioning Cloud Computing Resources

Hazal Erol, Kerem Bülbül, Nilay Noyan
Sabancı University, Industrial Engineering, Orhanlı-Tuzla, 34956, Istanbul, Turkey
herol@sabanciuniv.edu, bulbul@sabanciuniv.edu, nnoyan@sabanciuniv.edu

Abstract: We focus on the resource provisioning problem of a cloud consumer from an Infrastructure-as-a-Service type of cloud. The cloud provider offers two deployment options, which can be mixed and matched as appropriate. Cloud instances may be reserved for a fixed time period in advance at a smaller usage cost per hour but require a full commitment and payment for the entire contract duration. In contrast, on-demand instances reflect a pay-as-you-go policy at a premium. The trade-off between these two options is rooted in the inherent uncertainty in demand and price and makes it attractive to complement a base reserved capacity with on-demand capacity to hedge against the spikes in demand. This paper provides several novel multi-stage stochastic programming formulations to enable a cloud consumer to handle the cloud resource provisioning problem at a tactical level. We first formulate the cloud resource provisioning problem as a risk-neutral multi-stage stochastic program, which serves as the base model for further modeling variants. In our second set of models, we also incorporate a certain concept of system reliability. In particular, chance constraints integrated into the base formulation require a minimum service level met from reserved capacity, provide more visibility into the future available capacity, and smooth out expensive on-demand usage by hedging against possible demand fluctuations. Two alternate modeling paradigms—node-based versus scenario-based—are applied to all formulation types, and the corresponding computational efficiency is explored in experiments. Furthermore, some practical managerial insights are gleaned from the analysis of the solutions of the proposed models.

Keywords: OR in service industries; stochastic programming; cloud computing; resource provisioning; chance constraints; IaaS; virtual machine; on-demand instance; reserved instance; multi-stage

1. Introduction The term "cloud computing" is officially described by the National Institute of Standards and Technology (Erl et al., 2013) as a "model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." The cloud computing technology, which emerged in 1999 with the arrival of `Salesforce.com`, has been experiencing a very strong growth over the last decade. In 2006, `Amazon` launched `Elastic Compute Cloud (EC2)` enabling small companies to lease computational resources, store their own data, and run their own computer applications remotely. Subsequently, `Google App Engine` and `Microsoft Azure` were introduced into the cloud technology market in 2008 and 2010, respectively (see, e.g., Prakash, 2012). In the following five years, high-profile companies and institutions (e.g., `FBI, Netflix`) moved their services to the cloud (Microsoft Corporation, 2015; Matillion, 2016) and notable companies such as `Apple` started their own cloud services.

Cloud computing as a technology is currently mature and secure enough to enjoy a rapidly growing widespread use—a claim also supported by well-respected market research firms. For instance, `International Data Corporation (IDC)` asserts in *Worldwide Semiannual Public Cloud Services Spending Guide* of 2015 that worldwide revenues generated by public cloud services would be around \$96.5 billion in 2016 and will exceed \$195 billion in 2020 with an annual growth rate of 20.4% over the forecast period 2015-2020. These numbers will only be further buoyed up by the ever-increasing interest in machine learning (ML) and artificial intelligence (AI). The release of the ML&AI platforms such as `TensorFlow` and `Microsoft Azure Machine Learning Studio` allow ML and AI to go mainstream (Butler, 2016) and will only lead to further increases in demand for high performance computing on the cloud. The business benefits of moving resources and services from internal servers to the cloud—a.k.a. migration to the cloud—are numerous and include access to the state-of-the-art software and hardware technology, resource scaling with respect to demand, flexible payment options, and highly reliable service. As may be expected, however, this new way of doing business gives rise to novel decision-making problems as laid out in more detail in Section 2. In this context, the goal of this paper is to state, formalize, and solve a very fundamental decision-making problem from the perspective of the cloud customer as discussed next.

The cloud computing industry features three types of players: *cloud services providers (CSPs)*, *cloud customers*, and *cloud brokers*, who act as third-party facilitators by finding and provisioning resources to satisfy the demands of the cloud customers from a CSP or a set of CSPs. In this paper, we assume the role of a cloud broker and focus on the fundamental tactical problem of determining the optimal provisioning plan for meeting the computing needs of a cloud customer over a planning horizon with several provisioning stages or periods—e.g., eight quarterly stages in a bi-annual planning cycle. The two basic options available to the cloud customer are virtual machine (VM) instances leased for a fixed duration through reservation contracts/plans or VM instances procured as required on demand. In the current study, we ignore the additional option of using *spot instances*, where the cloud customers bid for unused capacity, because we tackle the cloud resource provisioning problem at a tactical level and the spot market is more pertinent to the decisions at the operational level. While the customer pays a fixed hourly rate for the *on-demand instances*, the usage of a *reserved instance* requires a fixed contractual fee covering the entire contact duration at a smaller hourly rate compared to that of a corresponding on-demand instance. A major complicating factor in this problem is that the demands of the customer and the prices of the cloud resources as set by the CSPs are subject to uncertainty over time. Clearly, in the absence of uncertainty, an optimal provisioning plan would exclusively rely on reserved instances, except when end-of-horizon effects are better mitigated by on-demand usage. However, the dynamic and uncertain nature of the problem requires a careful planning that determines a mix of reserved and on-demand cloud resources for each of the provisioning stages. We refer to this problem as the *stochastic resource provisioning problem for the cloud customer* (SRPP–CC). Despite the obvious significance of SRPP–CC due to its potential cost implications affecting the bottom line of the cloud customer, the literature on cloud resource provisioning from the viewpoint of cost minimization for the cloud customer is very scarce. Our work intends to partially fill this gap in the literature by developing a set of formulations for SRPP–CC, where each formulation underscores a different decision-making context. The setting of our initial base formulation is similar to that in Chaisiri et al. (2012); however, we subsequently extend this base formulation further by also incorporating a certain concept of system reliability into our models. A rich set of novel models applicable to SRPP–CC in practice is a major contribution of this paper. We will take up these issues in more detail in Section 3.

The setting of SRPP–CC described above gives rise to a set of challenging optimization problems because the uncertainty in the prices and demands renders the total cost a random variable, and we get a different random variable representing the total cost for each different set of decisions. Thus, picking the optimal set of decisions is accomplished by comparing these *random outcomes*. The prominent optimization methodology that deals with such decision-making problems under uncertainty is known as stochastic programming. In this paper, we rely on state-of-the-art modeling techniques from *multi-stage* stochastic programming to properly formulate and solve variants of SRPP–CC capturing different decision-making contexts. Constructing multi-stage stochastic programming models is not necessarily straightforward and requires avoiding some common pitfalls. We touch upon this subject in Sections 2-3 when we discuss the formulation in Chaisiri et al. (2012).

In all of our formulations, we employ the traditional risk-neutral preference relation to compare the random outcomes stemming from different provisioning policies and minimize the expected total cost over the planning horizon. A closer look at the structure of the optimal solutions from our base model, which minimizes the expected total cost of meeting the demand over the planning horizon without further considerations, reveals that the optimal provisioning policy is too reliant on on-demand VM instances when short-term demand fluctuations are encountered. This is not a desirable feature for a tactical-level planning problem because it severely restricts the visibility into the future and renders budgeting a challenging issue. Particularly for mission-critical infrastructure, tapping on-demand VM instances may be regarded as an undesirable event to be considered as a last resort. Taking this stance allows us to frame SRPP–CC as an engineering application also concerned with the *reliability* of the system, where restricting the probability of certain undesirable events is a fundamental issue. Formulations of such applications incorporate reliability by imposing a set of *chance (a.k.a. probabilistic) constraints* on the occurrence of the undesirable events. In the context of SRPP–CC, the presence of randomness in demand allows us to portray the constraints related to demand satisfaction as a set of *stochastic goal constraints*. Enforcing the feasibility of these constraints for every possible realization of demand may be too conservative and only be achieved at the expense of a significantly increased expected total cost. Escaping this concern and striking a balance between system reliability and cost is then accomplished by allowing the violation of the stochastic goal constraints related to demand fulfillment with a probability of at most α, where α reflects the risk aversion of the decision maker. This way of thinking is also very much in line with the popular rationale of formulating goals in terms of service level constraints in the IT industry, such as keeping the system up time above 99%. In this study, we embed two versions of *joint* chance-constraints into our base model. Initially, we mandate that all resource requirements are met from the reserved instances with a probability of at least $1 - \alpha$ over the entire planning horizon. In the second version, a similar requirement is applied to each time period separately. Thus, chance constraints integrated into the base formulation require a minimum service level met from reserved capacity and smooth out expensive on-demand usage by hedging against possible demand fluctuations.

The literature on optimization models with chance-constraints applied to static (single-stage) problems is rich—see Prékopa (1995), and Shapiro et al. (2009) for reviews and a comprehensive list of references. In contrast, interest in chance-constrained multi-stage stochastic programming has been picking up only recently (see, e.g., Zhang et al., 2014). In general, solving joint chance-constrained models is notoriously difficult due to their non-convex structure (see, e.g., Prékopa, 2003; Luedtke et al., 2010). However, a fairly recent methodological development in Luedtke et al. (2010)—originally presented for single-stage problems, where the randomness in the stochastic goal constraints has a finite support and is restricted to the right-hand side, has proven instrumental to advance our computational capabilities in solving joint chance-constrained optimization problems. Applying this progress to the multi-stage setting (Zhang et al., 2014) requires extra care in modeling to account for an implicit assumption in Luedtke et al.'s original work—see Section 3. Therefore, we consider it a noteworthy contribution to leverage good modeling practices enabled by recent methodological progress and hope to inspire and spur more practice-oriented research in a multi-stage setting with joint chance constraints. We stress that our joint chance-constrained formulations are solved through a state-of-the-art off-the-shelf solver in our numerical study in Section 4 without resorting to custom-tailored solution algorithms. Thus, our modeling and solution framework is easily accessible to practitioners. We also demonstrate the value of the proposed models on numerical instances generated in line with the recent practices at Amazon Web Services (AWS). In particular, we rely on actual AWS EC2 instances for the physical attributes of the VMs, and cost/price and capacity parameters (Amazon Web Services, Inc., 2018c,b)—see the data generation phase of our computational study in Section 4.1. The provided managerial takeaways highlight the benefits of applying operations research methods to a critical business problem posed by recent advances and trends in the information technology industry.

The next section reviews the relevant literature on decision-making problems in a cloud computing environment and positions our work. This is followed by a formal description of SRPP–CC and the related multi-stage stochastic programming formulations. The numerical study in Section 4, where the instances are created based on Amazon's EC2 offering, investigates some computational aspects of our formulations and also analyzes the structure of the optimal solutions obtained from different formulations. We provide concluding remarks in Section 5.

2. Literature Review There is a growing body of literature devoted to the development of optimization models for the decision-making problems encountered in cloud resource planning and management; we refer to Iyoob et al. (2013) for a thorough overview. The vast majority of these studies is dedicated to the problems from the CSP's perspective. In contrast, in our problem of interest, we take the viewpoint of a cloud broker. Accordingly, we focus on research studying cloud resource provisioning optimization, where the decision maker acts with the goal of meeting the computing needs of a cloud customer by procuring resources from one or several CSPs. We highlight the differences in the available problem descriptions and modeling approaches to position our work with respect to the most relevant existing studies.

Among the limited number of studies focusing on the decision-making problems from the perspective of a cloud broker, we first mention the most relevant ones which assume a deterministic setting. Nesmachnow et al. (2015) and Wang et al. (2015) introduce a new type of cloud brokerage system, where a cloud broker reserves a set of VM instances from CSPs such as Google or AWS, and then resells those reserved instances via on-demand at prices lower than those of the original CSPs. In such a setting, the broker exploits the price differential between the reserved and on-demand instances and the aggregated demand from its customer base, also helping its customers lower their cloud bill. Nesmachnow et al. (2015) define a deterministic problem to maximize the broker's profit by optimizing the allocation of the customer requests to the VMs procured from the CSPs and propose heuristic solution methods. Wang et al. (2015), on the other hand, exclude the broker's prices from consideration and formulate the problem of reserving instances from CSPs over time with the objective of minimizing the total service cost via dynamic programming. The lack of scalability of this exact approach prompts the authors to devise approximation algorithms. Note that from a modeling point of view, these two studies can be considered to take a cloud provider's perspective. In contrast, the deterministic setting of Subramanian and Savarimuthu (2016)—similar to the majority of the other studies—assumes the traditional role of a cloud broker, and solves the resource provisioning problem to satisfy the demand of a cloud customer from multiple CSPs. In their modeling framework, the broker evaluates the CSPs utilizing a service measurement index (SMI) based on critical attributes such as security level, performance, and accountability. The proposed mixed-integer programming formulation ensures that the customer demand is fulfilled by CSPs with an acceptable SMI score, while taking into consideration the additional constraints regarding the specific consumer and applications requirements such as location and legal regulations. The authors rely on the classical Benders decomposition approach to engineer a solution algorithm.

In many practical applications—including SRPP–CC, the customer demands and resource prices are subject to uncertainty over time. As variability increases, accounting for the inherent randomness explicitly in the modeling and solution framework becomes essential. In this spirit, Chaisiri et al. (2009) construct a two-stage stochastic integer programming model under price and demand uncertainty to determine an optimal provisioning plan for a customer. The objective is to minimize the expected total cost for procuring multiple VM classes from possibly multiple Infrastructure-as-a-Service (IaaS) cloud providers through reservation and on-demand plans, while balancing the trade-off between the cost of over-provisioning (cost of idle reserved resources) and under-provisioning (resulting in on-demand usage). The authors basically consider a single provisioning stage, which naturally has three phases—reservation, expending (utilization of reserved resources), and on-demand, and model the decisions for these three phases by employing a two-stage stochastic programming framework. The first-stage problem determines the number of VMs provisioned in the reservation phase, while the second-stage problem sets the number of VMs allocated in the latter two phases given the realizations of the uncertain parameters. This modeling approach has later been extended in Chaisiri et al. (2011) by also incorporating the spot option, which may present an attractive less expensive choice to the customers compared to a reservation plan. On the flip side, the spot market prices are highly volatile, and availability may be restricted at the time of peak demand in a bidding system. As another extension of Chaisiri et al. (2009), Chase et al. (2014) additionally account for the randomness in the demand for the external network bandwidth acquired from the internet service providers. These studies employ the classical scenario-based (sampling-based) methods to obtain the deterministic equivalent formulations of the proposed stochastic programming models, and solve those via an off-the-shelf solver.

The studies based on a two-stage decision-making framework above fail to capture the multi-period dynamic nature of the cloud resource provisioning problem. In this regard, Chaisiri et al. (2012) and Chase and Niyato (2017) build upon Chaisiri et al. (2009) and Chase et al. (2014), respectively, and introduce multi-stage stochastic programming versions of the earlier corresponding research in analogous modeling settings. Differently from the other studies, Chaisiri et al. (2012) devise a custom solution algorithm based on Benders decomposition. This method is only valid if the integrality restrictions on the decision variables—such as the number of provisioned instances—are relaxed. Moreover, the Benders master problem and one of the subproblems are still large-scale optimization problems due to the presence of scenario-dependent variables and constraints. In other words, the authors do not rely on a scenario decomposition, which is a commonly used effective approach in decomposing scenario-based stochastic programming models into smaller (single-scenario) subproblems (see, e.g., Shapiro et al., 2009). Consequently, the scalability of the proposed Benders algorithm with respect to the number of scenarios becomes a significant issue, and therefore, Chaisiri et al. (2012) resort to sample average approximation to reduce the number of scenarios and solve the resulting deterministic equivalent formulation via an off-the-shelf solver. We next contrast our approach of modeling the multi-period cloud resource provisioning problem with those of Chaisiri et al. (2012) and Chase and Niyato (2017).

Our initial base problem, mainly built upon Chaisiri et al. (2012), minimizes the expected total cost of reservation and on-demand plans for meeting the demand of a cloud customer over a planning horizon with multiple provisioning stages. Similarly to Chaisiri et al. (2012)—and differently from Chase and Niyato (2017)—we only focus on the VM management. As we stated in Section 1, establishing a multi-stage stochastic programming model may seem straightforward but it contains subtle details that should not be overlooked. Chaisiri et al. (2012) and Chase and Niyato (2017) both miss an important constraint type, known as *non-anticipativity constraints*, in their scenario-based formulations. These constraints are essential for the validity of the decision-making models, since we cannot anticipate the future while making decisions in the presence of uncertainty. We properly remedy this shortcoming in our formulations, and additionally make some modifications in the problem definition following the recent changes in practice (see, e.g., Zolman, 2014; Amazon Web Services, Inc., 2018b). We defer a detailed discussion on these different features to Section 3. More importantly, we develop two new risk-averse multi-stage stochastic programming problems by integrating chance constraints into the base formulation. These chance constraints require a minimum service level met from reserved capacity, provide more visibility into the future available capacity, and smooth out expensive on-demand usage by hedging against possible demand fluctuations. To the best of our knowledge, using chance constraints is a first in the context of SRPP–CC. Independent of the field, the literature on chance-constrained multi-stage stochastic programming is very limited; however, interest has been picking up recently (see, e.g., Zhang et al., 2014; Shen and Wang, 2014).

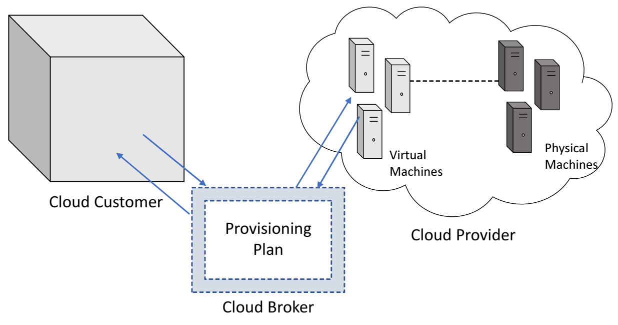
To conclude this section, we point out some references to other major classes of decision-making problems from the cloud computing industry, even if these are not directly related to our work. Some of the problems of interest to a CSP include but are not limited to resource allocation/load balancing (see, e.g., Maguluri et al., 2012), VM placement (see, e.g., Bin et al., 2011), application installation (see, e.g., Amiri, 2016), computational resource scheduling (see, e.g., Shen and Wang, 2014), and workflow/task scheduling (see, e.g., Arabnejad et al., 2016; Shi et al., 2016; Tang et al., 2016). The cloud migration (see, e.g., Avram, 2014; Bibi et al., 2012) and cloud product selection problems (see, e.g., Godse and Mulik, 2009; Boussoualim and Aklouf, 2014) are, on the other hand, among the further common decision-making contexts facing a cloud customer. Finally, a recent study by Li et al. (2017) is worth mentioning here. These authors present a system-wide modeling approach to solve a resource allocation problem in software-as-a-service (SaaS) clouds and incorporate multiple types of decision makers—SaaS users, SaaS providers, and cloud resource providers.

3. Problem Description and Multi-Stage Optimization Models

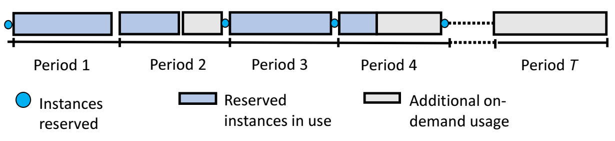
In this section, we describe the stochastic resource provisioning problem SRPP–CC of interest and present our multi-stage modeling approaches along with the corresponding mathematical programming formulations.

In a cloud computing environment, the customers can reduce the cost of executing certain jobs by provisioning computing resources—such as processing power, storage, software, and network bandwidth—from third-party cloud providers. Two primary resource provisioning options offered to the cloud customers are reservation and on-demand plans/contracts—also see Section 1. More specifically, a customer leases VM instances for a fixed duration through reservation contracts or procures VM instances as required on demand. A cloud provider typically hosts different types of VMs supplying varying amounts of resources such as computing power (in some standardized unit of CPU power), storage capacity (in Gigabytes—GBs), and network bandwidth (in Megabits/second—Mbps). Following the practice at AWS, we assume that VMs are grouped into families based on different performance and use cases—such as general purpose, compute optimized, memory optimized, and storage optimized. According to their operations and required applications, the customers specify their demand for each resource type over the duration of each provisioning stage; e.g., expressed in CPU-hours for compute power and GB-hours for storage. This demand structure provides a practical benefit of not forcing the customers to define their resource requirements in terms of the number of the specific VM instances a priori. The burden of trading off the performance versus cost and identifying the optimal mix of VMs to be provisioned is transferred to the mathematical model.

In this study, we take the role of a cloud broker—see Figure 1a—and tackle the problem of computing an optimal provisioning plan for meeting the demand of a cloud customer over a planning horizon with several provisioning stages—each represented by a time period as depicted in Figure 1b. The dynamics of provisioning and utilizing different types of VM instances at each stage can be captured in three phases: reservation, expending (utilization), and on-demand. In the reservation phase, the broker decides on the number and type of the reservation plans to purchase without knowing the actual prices of the CSPs and the customer demand in the future. These reserved instances can be put to use for the duration and the fixed price stated on the associated contracts. In the expending phase, the actual price and demand values are observed, and the broker determines the number and type of the reserved VM instances to be



(a) Players in a cloud environment.



(b) Provisioning stages and phases.

Figure 1: Illustrations of the problem elements in SRPP–CC.

employed from the currently available pool. If the total capacity of the reserved instances falls short of the total demand realized, this capacity is augmented by leasing additional VM instances in the on-demand phase. We further elaborate on this dynamic and uncertainty-dependent structure in Section 3.1.

For reservation contracts, different types of pricing schemes appear in the literature. For instance, Chaisiri et al. (2012) assume that the total cost of a reserved VM instance depends on the usage amount in addition to a fixed one-time fee. In line with the revamped AWS reserved instance pricing still in use (see, e.g., Zolman, 2014), we drop the variable cost associated with a reserved instance and assume that the cloud consumers pay for the whole reservation duration regardless of the utilized amount. In addition, differently from Chaisiri et al. (2012), limits are enforced on the number of instances in use at any given stage for each type of individual reserved and on-demand instance, and across each instance family. These capacity restrictions are aligned with the current practice at AWS (Amazon Web Services, Inc., 2018b).

In our stochastic optimization models, we capture the inherent uncertainty via a finite probability space $(\Omega, 2^{\Omega}, \mathbb{P})$ with $\Omega = \{\omega^1, \ldots, \omega^n\}$ and $\mathbb{P}(\omega^s) = \pi^s > 0,\ s \in S = \{1, \ldots, n\} = [n]$, where $[z]$ stands for the set of the first z positive integers in the rest of the paper. In other words, we discretize the stochastic input process associated with the uncertain parameters over the planning horizon, and consider a finite set of scenarios capturing the potential future outcomes. We represent the input process over T time stages by $\boldsymbol{\xi} = (\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_T)$, where the random vector $\boldsymbol{\xi}_t$ is comprised of the vectors $\mathbf{D}_t$ and $\mathbf{P}_t$ associated with the random demand and cost/price values in time period $t \in \mathcal{T} = [T]$, respectively. The history of the input process up to period t is designated by $\boldsymbol{\xi}_{[t]} = (\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_t)$. We assume that the random vector $\boldsymbol{\xi} = (\mathbf{D}, \mathbf{P})$ has finitely many realizations given by $\boldsymbol{\xi}(\omega^1) = (\mathbf{D}^1, \mathbf{P}^1), \ldots, \boldsymbol{\xi}(\omega^n) = (\mathbf{D}^n, \mathbf{P}^n)$ with corresponding probabilities $\pi^1, \pi^2, \ldots, \pi^n$, respectively. More specifically, $\boldsymbol{\xi}_t(\omega^s) = \boldsymbol{\xi}_t^s = (\mathbf{D}_t^s, \mathbf{P}_t^s)$ expresses the joint realization of the random demand and cost parameters at stage t under scenario s. It is natural to represent this uncertainty characterization in the form of a scenario tree as illustrated in Figure 2. The root note represents the current time, and all data at this node is known with certainty. A node at level $t \in \mathcal{T}$ referred to as a stage-t node corresponds to a stage-t scenario denoting a specific realization of the random vector $\boldsymbol{\xi}_{[t]}$. Each stage-t node with $t > 1$ has a unique *ancestor* at stage $t-1$, and a node at stage $t < T$ has a set of *descendants (child nodes)* at stage $t+1$. Each path from the root node associated with $t = 0$ to a leaf node at the final stage T is unique and corresponds to a scenario, i.e., there is a one-to-one correspondence between the set of scenarios and the set of leaf nodes. The underlying probability measure specifies the conditional probability distribution of $\boldsymbol{\xi}_{t+1}$ given $\boldsymbol{\xi}_{[t]}$ for $t < T$, and the stage-t marginal probability distribution associated with $\boldsymbol{\xi}_{[t]},\ t \in \mathcal{T}$, which yields the probability of passing through a node $l \in \mathcal{N}$ at stage t. The list of sets and parameters below summarize the notation required to formalize the scenario tree structure and incorporate it into our models in the rest of this section:

$S = [n]$: set of scenarios,

π^s: probability of scenario $s \in S$,

S_s^t: set of scenarios that share a common history with scenario s at stage $t \in \mathcal{T}$,

$\mathcal{N}$: set of nodes in the scenario tree, where the root node is indexed by 1,

L_t: set of nodes at level $t \in \{0\} \cup \mathcal{T}$,

ρ^l: the probability of passing through node $l \in \mathcal{N}$,

$A(l, m)$: the mth ancestor of node $l \in \mathcal{N}$,

$t(l)$: stage associated with $l \in \mathcal{N}$.

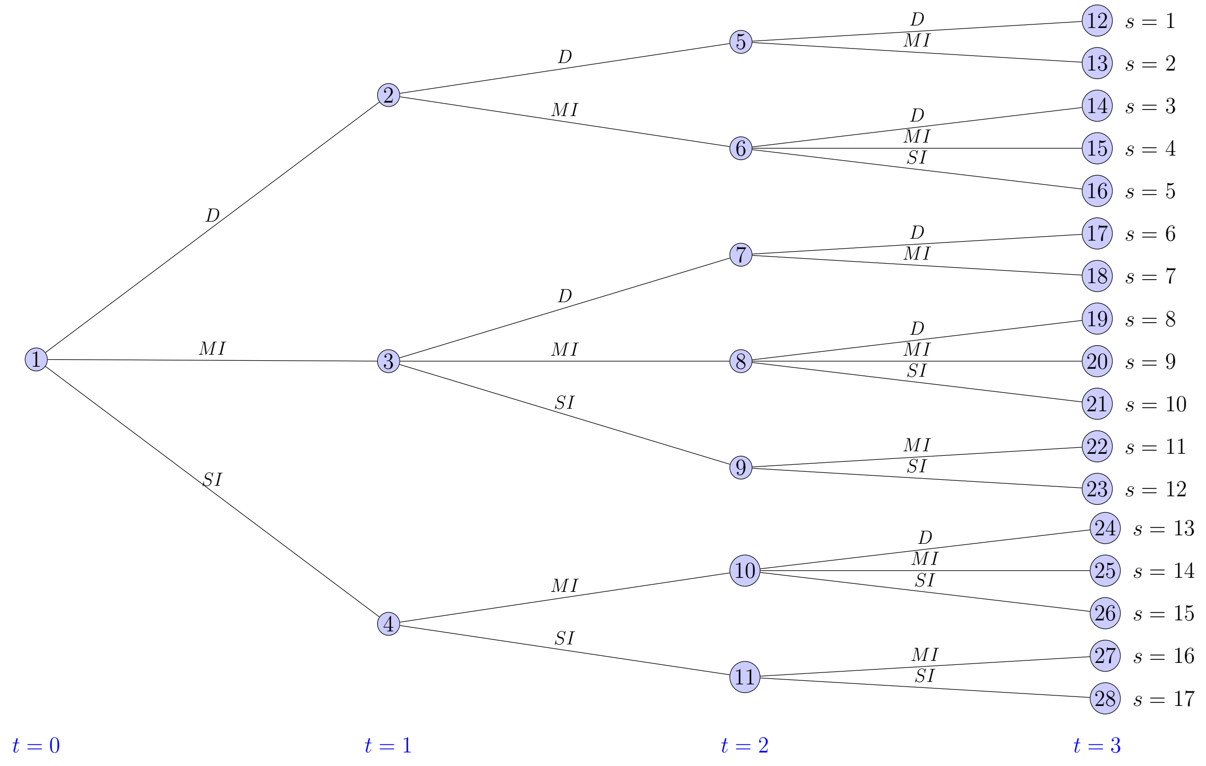


Figure 2: A sample scenario tree representation.

Example 3.1 ***Scenario Tree Representation.*** *In the illustrative example given in Figure 2, we assume that only demand is uncertain. The underlying stochastic process evolves such that the relative change in demand from the previous to the current period is either a decrease (D), a moderate increase (MI), or a strong increase (SI). A relative change of type D cannot be followed by SI and vice versa—see Figure 2. The development of this stochastic progress is described through the conditional probabilities* γ_{ij} *attached to the edges of the scenario tree, where* γ_{ij} *specifies the probability of going through node* j *at stage* $t+1$ *given that we are currently at node* i *at stage* t*. The probability* ρ^l *of passing through node* $l \in \mathcal{N}$ *is then computed by multiplying the conditional probabilities along each edge starting from the root node until we reach node* l*. For example, observe that* $\rho^9 = \gamma_{13}\gamma_{39}$*.*

The root node at $t = 0$ marks the beginning of the planning horizon with $T = 3$. The nodes are numbered by traversing the tree in a breadth-first fashion, where $\mathcal{N} = [28]$. Consequently, all leaf nodes, and for that matter all nodes at a particular stage, are numbered consecutively. Each leaf node in the set $L_T = \{12, \ldots, 28\}$ stands for a particular realization of $\boldsymbol{\xi}$ labeled as a scenario, where $S = [17]$ and $\pi^1 = \rho^{12}, \ldots, \pi^{17} = \rho^{28}$. To illustrate the rest of our notation, consider node 8 at stage 2 so that $t(8) = 2$. This node descends from node 3 at stage 1 and node 1 at stage 0, which implies that $A(8,1) = 3$ and $A(8,2) = 1$. Node 8 denotes a specific realization of the random vector $\boldsymbol{\xi}_{[2]}$, and all scenarios that go through node 8—the scenarios 8, 9, 10—share a common history up until stage 2. Thus, we have $S_8^2 = \{9, 10\}$, $S_9^2 = \{8, 10\}$, $S_{10}^2 = \{8, 9\}$.

The described uncertainty characterization can be incorporated into multi-stage stochastic programs via two alternate modeling paradigms: scenario-based versus node-based. We apply them both to our proposed models, leading to alternative mathematical programming formulations. We now proceed to present our initial base multi-stage stochastic optimization model, which will be subsequently extended by incorporating two types of chance constraints.

3.1 Risk-Neutral Multi-Stage Optimization Models The scenario-based modeling paradigm creates decision variables for each scenario at each stage. While this approach has intuitive appeal, it needs to be implemented carefully. The decisions which are based on the same realization of the random vector $\boldsymbol{\xi}_{[t]}$ must be identical because we cannot anticipate the future while making decisions in the presence of uncertain parameters. Expressed alternatively, the decisions corresponding to a node in a scenario tree must be the same for every scenario that goes through that node. In scenario-based multi-stage models, this consideration is taken into account by enforcing a set of additional constraints. These so-called *non-anticipativity* constraints are essential for the validity of the model—recall that they are absent from the formulations in Chaisiri et al. (2012). Missing non-anticipativity constraints in a multi-stage stochastic program may lead to inconsistent solutions with possibly differing values for the decision variables corresponding to the same history of the stochastic input process. Additional notation necessary for our multi-stage formulations of SRPP–CC is listed below:

Sets

I :	set of VM instance types indexed by i,
$\mathcal{T}$:	set of provisioning stages indexed by t, where $\lvert\mathcal{T}\rvert = T$,
R :	set of resources indexed by r,
K :	set of reservation contracts indexed by k,
F :	set of instance type families indexed by f,
I_f :	set of instance types in family $f \in F$,
$\mathcal{T}_r =$	$[T-1]$: set of provisioning stages at the end of which reservation contracts can be purchased.

Parameters

H :	number of hours available in one provisioning stage,
d_{rt}^s :	demand for resource $r \in R$ at provisioning stage $t \in \mathcal{T}$ under scenario $s \in S$ (e.g., in CPU-hours for compute power and GB-hours for storage),
a_{ri} :	amount of resource $r \in R$ in instance type $i \in I$,
C_i^r :	maximum number of reserved instances allowed for instance type $i \in I$,
C_i^o :	maximum number of on-demand instances allowed for instance type $i \in I$,
C_f^r :	maximum number of reserved instances allowed for instance type family $f \in F$,
C_f^o :	maximum number of on-demand instances allowed for instance type family $f \in F$,

p_{ik0} :	hourly cost/price for reserving an instance of type $i \in I$ by contract $k \in K$ at $t = 0$,
p^s_{ikt} :	hourly cost/price for reserving an instance of type $i \in I$ by contract $k \in K$ at the end of the provisioning stage $t \in \mathcal{T}_r$ under scenario $s \in S$,
o^s_{it} :	hourly cost/price for on-demand instance of type $i \in I$ at provisioning stage $t \in \mathcal{T}$ under scenario $s \in S$,
l_k :	length of the reservation contract $k \in K$ expressed in the number of provisioning stages.

Recall that the vector $\mathbf{D}^s_t$ represents the demand for each resource type $r \in R$ at stage $t \in \mathcal{T}$ under scenario $s \in S$, i.e., $\mathbf{D}^s_t = \left(d^s_{1t}, \ldots, d^s_{|R|t}\right)$. Similarly, $\mathbf{P}^s_t$ is the vector of the cost/price parameters $o^s_{it},\ i \in I$, and $p^s_{ikt},\ i \in I,\ k \in K$, at stage $t \in T$ under scenario $s \in S$ that can be represented in a similar manner. Observe that the values of the parameters d^s_{rt} and $d^{s'}_{rt}$ must be identical if two scenarios s and s' share a common history up to stage $t \in \mathcal{T}$. This is also the case for the parameters p^s_{ikt} and o^s_{it}. These requirements are fulfilled during the data generation phase of our computational study in Section 4.1.

Decision Variables

y_{ik0} :	number of instances of type $i \in I$ reserved by contract $k \in K$ at $t = 0$,
y^s_{ikt} :	number of instances of type $i \in I$ reserved by contract $k \in K$ at the end of the provisioning stage $t \in \mathcal{T}_r$ under scenario $s \in S$,
ζ^s_{ikt} :	number of instances of type $i \in I$ reserved by contract k and used at stage $t \in \mathcal{T}$ under scenario $s \in S$,
x^s_{it} :	number of instance-hours of the on-demand instance of type $i \in I$ used at stage $t \in \mathcal{T}$ under scenario $s \in S$.

The dynamic and uncertainty-adaptive structure of the decisions—except those at $t = 0$—deserves some discussion. Most models in the literature developed for planning problems over a time horizon are nonadaptive to uncertainty in nature. All decisions are set at the beginning of the horizon, and from a modeling viewpoint these are not subject to change even if more information about the uncertain time-dependent model elements becomes available over the course of the planning horizon. Clearly, allowing the decisions to adapt to the observed uncertainty over time would pave the way to a more flexible and efficient planning. To this end, in our multi-stage stochastic programming formulations in the following, the decisions at each stage $t \geq 1$ take the history of the input process $\boldsymbol{\xi}_{[t]}$ up to that stage into consideration. For example, the value of ζ^s_{ikt} can be viewed as the realization of the uncertainty-dependent decision $\zeta_{ikt}(\omega)$ under scenario s. In our models, the events unfold and the decision process evolves in the following sequence: at the beginning of the project, the values of the decision variables $y_{ik0},\ i \in I,\ k \in K$, are determined by considering the resulting expected total future cost. The uncertainty at stage 1 is resolved subsequently—say scenario s' is observed. Then, given the realized demand and cost values $\mathbf{D}^{s'}_1$ and $\mathbf{P}^{s'}_1$, the minimum cost decisions on the utilization amounts of the already reserved instances represented by $\zeta^{s'}_{ik1},\ i \in I,\ k \in K$, and the amount of on-demand instances to be procured in the first stage denoted by $x^{s'}_{i1}, i \in I$, are calculated. The decisions $y^{s'}_{ik1},\ i \in I,\ k \in K$, on the new reservation contracts to be purchased succeed and mark the end of stage 1. The process proceeds similarly stage by stage until the end of the horizon—see Figure 1b. We investigate and quantify the value of this uncertainty-adaptive modeling approach in Section 4.2.1 as part of our numerical study. We next present the scenario-based formulation of our risk-neutral multi-stage stochastic programming model:

$$\textbf{(SB}-\textbf{RN)}$$

$$\text{minimize} \qquad H \sum_{i \in I} \sum_{k \in K} l_k p_{ik0} y_{ik0} + \sum_{i \in I} \sum_{t \in \mathcal{T}} \sum_{s \in S} \pi^s \left(o^s_{it} x^s_{it}\right)$$

$$+ H \sum_{i\in I}\sum_{t\in\mathcal{T}_r}\sum_{s\in S}\sum_{k\in K}\pi^s\left(l_k p^s_{ikt} y^s_{ikt}\right) \tag{1a}$$

subject to

$$H\sum_{i\in I}\sum_{k\in K} a_{ri}\zeta^s_{ikt} + \sum_{i\in I} a_{ri} x^s_{it} \geq d^s_{rt}, \qquad \forall r\in R, t\in\mathcal{T}, s\in S, \tag{1b}$$

$$\zeta^s_{ik1} \leq y_{ik0}, \qquad \forall i\in I, k\in K, s\in S, \tag{1c}$$

$$\zeta^s_{ikt} \leq y_{ik0} + \sum_{t'\in[t-1]} y^s_{ikt'}, \qquad \forall i\in I, k\in K, t\in[2,l_k], s\in S, \tag{1d}$$

$$\zeta^s_{ikt} \leq \sum_{t'\in[t-l_k,t-1]} y^s_{ikt'}, \qquad \forall i\in I, k\in K, t\in[l_k+1,T], s\in S, \tag{1e}$$

$$\sum_{k\in K} y_{ik0} \leq C^r_i, \qquad \forall i\in I, \tag{1f}$$

$$\sum_{\{k\in K\mid t\leq l_k\}} y_{ik0} + \sum_{k\in K}\sum_{\substack{t'\in[t-l_k,t-1]\\ t'\geq 1}} y^s_{ikt'} \leq C^r_i, \qquad \forall i\in I, t\in[2,T], s\in S, \tag{1g}$$

$$\sum_{i\in I_f}\sum_{k\in K} y_{ik0} \leq C^r_f, \qquad \forall f\in F, \tag{1h}$$

$$\sum_{i\in I_f}\left(\sum_{\{k\in K\mid t\leq l_k\}} y_{ik0} + \sum_{k\in K}\sum_{\substack{t'\in[t-l_k,t-1]\\ t'\geq 1}} y^s_{ikt'}\right) \leq C^r_f, \qquad \forall f\in F, t\in[2,T], s\in S, \tag{1i}$$

$$x^s_{it} \leq HC^o_i, \qquad \forall i\in I, t\in\mathcal{T}, s\in S, \tag{1j}$$

$$\sum_{i\in I_f} x^s_{it} \leq HC^o_f, \qquad \forall f\in F, t\in\mathcal{T}, s\in S, \tag{1k}$$

$$y^s_{ikt} = y^{s'}_{ikt}, \qquad \forall i\in I, k\in K, t\in\mathcal{T}_r, s\in S, s'\in S^t_s, \tag{1l}$$

$$x^s_{it} = x^{s'}_{it}, \qquad \forall i\in I, t\in\mathcal{T}, s\in S, s'\in S^t_s, \tag{1m}$$

$$\zeta^s_{ikt} = \zeta^{s'}_{ikt}, \qquad \forall i\in I, k\in K, t\in\mathcal{T}, s\in S, s'\in S^t_s, \tag{1n}$$

$$y_{ik0} \in \mathbb{Z}_+, \qquad \forall i\in I, k\in K, \tag{1o}$$

$$y^s_{ikt} \in \mathbb{Z}_+, \quad \zeta^s_{ikt} \in \mathbb{Z}_+, \qquad \forall i\in I, k\in K, t\in\mathcal{T}_r, s\in S, \tag{1p}$$

$$x^s_{it} \in \mathbb{Z}_+, \qquad \forall i\in I, t\in\mathcal{T}, s\in S. \tag{1q}$$

The first term in the objective (1a) expresses the cost of the instances reserved at $t = 0$. The expected total costs of the on-demand and reserved instances over T provisioning stages are quantified by the second and third terms, respectively. The constraints (1b) guarantee that the demand for each resource type at each provisioning stage is satisfied by a combination of reserved and on-demand instances under each scenario. The amount of reserved instances used in the current provisioning stage cannot exceed the availability in the outstanding reservation contracts. This is enforced for each instance type, contract, and scenario over $\mathcal{T}$ through the constraints (1c)-(1e). The constraints (1f)-(1g) limit the number of instances that can be reserved simultaneously for each instance type over $\mathcal{T}$ under each scenario. Analogous requirements are imposed on the reserved instance families through the constraints (1h)-(1i). Similar capacity restrictions on individual on-demand instance types and associated instance families are mandated via the constraints (1j)-(1k). Finally, the non-anticipativity constraints (1l)-(1n) ensure that the decisions at stage $t \in T$ under scenario $s \in S$ are identical for the set of scenarios that share a common history with scenario s up to that stage—recall the discussion at the start of Section 3.1.

Alternative Forms of the Non-Anticipativity Constraints. There exist alternative ways of expressing the non-anticipativity constraints, which differ in terms of computational performance (Shapiro et al., 2009). We present here two alternative forms, which are computationally more effective than that in (1l)-(1n) stipulated for each relevant scenario pair (s, s') and is quadratic in cardinality.

We define $S(l)$ to denote the set of scenarios going through node $l \in \mathcal{N}$, and without loss of generality, assume that these scenarios are ordered such that $S(l) = \{l^v, l^v+1, \ldots, l^v + n_l - 1\}$, where $n_l = |S(l)|$. It is easy to see that such a particular ordering can be obtained for each node using a proper scenario indexing as in Figure 2. For example, for node 3 in Figure 2, we have $l^v = 6$ and $S(l) = \{6, 7, \ldots, 12\}$ with $n_l = 7$. The two alternative forms are illustrated for the variables y^s_{ikt} below:

$$y^{l^v}_{ikt(l)} = y^s_{ikt(l)}, \qquad \forall i \in I,\ k \in K,\ l \in \mathcal{N} \setminus \{1\} : t(l) < T,\ s = l^v+1, l^v+2, \ldots, l^v + n_l - 1, \tag{2a}$$
$$y^s_{ikt(l)} = y^{s+1}_{ikt(l)}, \qquad \forall i \in I,\ k \in K,\ l \in \mathcal{N} \setminus \{1\} : t(l) < T,\ s = l^v, l^v+1, \ldots, l^v + n_l - 2. \tag{2b}$$

The first case (2a) prescribes that the values of all variables y^s_{ikt} for $s \in S(l)$ must be identical to that of $y^{l^v}_{ikt}$, and in the latter case (2b), the values of two successive y^s_{ikt}-variables based on the ordering in $S(l)$ must match. For a detailed discussion and further options, we refer to Shapiro et al. (2009). In Section 4, we present results on the alternative forms above to provide insights about their comparative computational performance.

3.1.1 Node-Based Formulation of the Risk-Neutral Multi-Stage Stochastic Programming Model

In contrast to the scenario-based modeling framework, the node-based modeling approach indexes the uncertain parameters and the relevant decision variables by the nodes of the scenario tree. In line with this rationale, we define:

d_{rl} :	demand for resource $r \in R$ associated with node $l \in \mathcal{N} \setminus \{1\}$,
p_{ikl} :	hourly cost/price for reserving an instance of type $i \in I$ by contract $k \in K$ at node $l \in \mathcal{N} : t(l) < T$,
o_{il} :	hourly cost/price for on-demand instance of type $i \in I$ at node $l \in \mathcal{N} \setminus \{1\}$,
y_{ikl} :	number of instances of type $i \in I$ reserved by contract $k \in K$ at node $l \in \mathcal{N} : t(l) < T$,
ζ_{ikl} :	number of instances of type $i \in I$ reserved by contract k and used at node $l \in \mathcal{N} \setminus \{1\}$,
x_{il} :	number of instance-hours of the on-demand instance of type $i \in I$ used at node $l \in \mathcal{N} \setminus \{1\}$.

The following relations highlight the correspondence between the scenario- and node-based representations of the uncertain parameters: $d^s_{rt(l)} = d_{rl}$, $p^s_{ikt(l)} = p_{ikl}$, and $o^s_{it(l)} = o_{il}$ hold for all relevant values of the appearing indices $l \in \mathcal{N},\ s \in S(l),\ i \in I,\ r \in R,\ k \in K$. The node-based formulation of our risk-neutral multi-stage stochastic programming model is then stated as:

(NB – RN)

$$\begin{aligned}
\text{minimize} \quad & H \sum_{i \in I} \sum_{k \in K} l_k p_{ik1} y_{ik1} + \sum_{i \in I} \sum_{l \in \mathcal{N} \setminus \{1\}} \rho^l \left(o_{il} x_{il}\right) \\
& + H \sum_{i \in I} \sum_{k \in K} \sum_{\substack{l \in \mathcal{N} \setminus \{1\} \\ t(l) < T}} \rho^l \left(l_k p_{ikl} y_{ikl}\right) && (3a) \\
\text{subject to} \quad & H \sum_{i \in I} \sum_{k \in K} a_{ri} \zeta_{ikl} + \sum_{i \in I} a_{ri} x_{il} \geq d_{rl}, && \forall r \in R, l \in \mathcal{N} \setminus \{1\}, \quad (3b) \\
& \zeta_{ikl} \leq y_{ik1}, && \forall i \in I, k \in K, l \in L_1, \quad (3c) \\
& \zeta_{ikl} \leq y_{ik1} + \sum_{m=1}^{j-1} y_{ikA(l,m)}, && \forall i \in I, k \in K, l \in L_j : j \in [2, l_k], \quad (3d) \\
& \zeta_{ikl} \leq \sum_{m=1}^{l_k} y_{ikA(l,m)}, && \forall i \in I, k \in K, l \in L_j : j \in [l_k+1, T], \quad (3e) \\
& \sum_{k \in K} y_{ik1} \leq C^r_i, && \forall i \in I, \quad (3f)
\end{aligned}$$

$$\sum_{\substack{k\in K\\ j\leq l_k}} y_{ik1} + \sum_{k\in K}\sum_{m=1}^{\min\{l_k,j-1\}} y_{ikA(l,m)} \leq C_i^r, \qquad \forall i\in I, l\in L_j : j\in[2,T], \tag{3g}$$

$$\sum_{i\in I_f}\sum_{k\in K} y_{ik1} \leq C_f^r, \qquad \forall f\in F, \tag{3h}$$

$$\sum_{i\in I_f}\left(\sum_{\substack{k\in K\\ j\leq l_k}} y_{ik1} + \sum_{k\in K}\sum_{m=1}^{\min\{l_k,j-1\}} y_{ikA(l,m)}\right) \leq C_f^r, \qquad \forall f\in F, l\in L_j : j\in[2,T], \tag{3i}$$

$$x_{il} \leq HC_i^o, \qquad \forall i\in I, l\in\mathcal{N}\setminus\{1\}, \tag{3j}$$

$$\sum_{i\in I_f} x_{il} \leq HC_f^o, \qquad \forall f\in F, l\in\mathcal{N}\setminus\{1\}, \tag{3k}$$

$$y_{ikl}\in\mathbb{Z}_+, \qquad \forall i\in I, k\in K, l\in\mathcal{N} : t(l)<T, \tag{3l}$$

$$\zeta_{ikl}\in\mathbb{Z}_+, \qquad \forall i\in I, k\in K, l\in\mathcal{N}\setminus\{1\}, \tag{3m}$$

$$x_{il}\in\mathbb{Z}_+, \qquad \forall i\in I, l\in\mathcal{N}\setminus\{1\}. \tag{3n}$$

For the sake of brevity, we skip a detailed description of (3) but only point out the correspondence of the model elements to those in ($\mathbf{SB-RN}$). The expressions in the objective function (3a)—referred to as $f(\mathbf{y},\mathbf{x})$ in the rest of the paper—match those in (1a) one by one. The sets of constraints (1b), (1c)-(1e), (1f)-(1i), (1j)-(1k) are the counterparts of (3b), (3c)-(3e), (3f)-(3i), and (3j)-(3k), respectively. The defining attribute of ($\mathbf{NB-RN}$) is that no copies of the same decision are created, rendering the non-anticipativity constraints redundant. The potential computational benefits will be explored in our numerical study in Section 4.

3.2 Chance-Constrained Multi-Stage Optimization Models The dynamic structure of our risk-neutral base model allows for fluctuations in the base capacity acquired from the reserved VM instances, where the ill effects of the spikes in demand are mitigated through more expensive on-demand usage. In the context provided in Section 1, overreliance on on-demand instances may be less than desirable for a tactical level planning problem. This prompts us to incorporate a notion of reliability into our models through (joint) chance constraints, which mandate a minimum service level met from reserved capacity. The delicate balance between better visibility into the future available capacity and the cost of reliability due to over-provisioning from reserved instances is controlled through the enforced probability levels of the chance constraints. The first type of joint chance constraint we offer in this work prescribes that all resource requirements are met from the reserved instances with a probability of at least $1-\alpha$ over the entire planning horizon. In the second type, a similar restriction is imposed on each individual time period. Applying both the scenario- and node-based modeling approaches, we obtain the chance-constrained variants of the base formulations ($\mathbf{SB-RN}$) and ($\mathbf{NB-RN}$), respectively. However, in the interest of brevity, the scenario-based formulations are omitted here. The computational efficiency of our alternate chance-constrained formulations is explored in Section 4.3.

3.2.1 Multi-Stage Stochastic Model with a Horizon-wise Joint Chance Constraint The following joint chance constraint stipulates that the probability of fulfilling the demand of all resources completely through reserved instances at all stages is at least $1-\alpha$ and enforces a *horizon-wise* minimum service level met from reserved capacity:

$$\mathbb{P}\left(H\sum_{i\in I}\left(\sum_{k\in K} a_{ri}\zeta_{ikt}(\omega)\right) \geq D_{rt}(\omega), \quad \forall r\in R,\ t\in\mathcal{T}\right) \geq 1-\alpha. \tag{4}$$

Here, $D_{rt}(\omega)$ designates the random demand for resource type $r \in R$ at provisioning stage $t \in \mathcal{T}$, and $1-\alpha$ specifies the horizon-wise reliability level. This constraint is integrated into $(\mathbf{NB-RN})$ by appending a set of linear constraints (5c)-(5d) featuring the binary decision vector $\boldsymbol{\lambda}$. These additional constraints correspond to a mathematical programming reformulation of the probabilistic statement (4) and lead to the following mixed-integer linear programming (MIP) formulation:

$$
\begin{aligned}
&(\mathbf{NB-JCCI}) && && \\
&\text{minimize} && f(\mathbf{x},\mathbf{y}) && && \text{(5a)}\\
&\text{subject to} && (3b)-(3n), && && \text{(5b)}\\
& && H\sum_{i\in I}\sum_{k\in K} a_{ri}\zeta_{ikl} \ge d_{rl}\left(1-\lambda^s\right), && \forall r \in R, l \in \mathcal{N}\setminus\{1\}, s \in S(l), && \text{(5c)}\\
& && \sum_{s\in S}\pi^s\lambda^s \le \alpha, && && \text{(5d)}\\
& && \lambda^s \in \{0,1\}, && \forall s \in S. && \text{(5e)}
\end{aligned}
$$

The stochastic goal constraints related to the satisfaction of demand are specified over the entire planning horizon and require us to check whether all resource demands are completed via reservation contracts at all stages along each root-to-leaf path of the scenario tree. Accordingly, the binary decision variables $\boldsymbol{\lambda}$ are indexed by scenario. The variable $\lambda^s,\ s \in S$, takes the value 1 if the entire demand for each resource is *not* met from reserved capacity at least once on some node along the root-to-leaf path corresponding to scenario s. In this case, the constraints (5c) become redundant for scenario s because the left-hand sides of these constraints are guaranteed to be non-negative. The constraints (5c)-(5e) then collectively ensure that the probability of not satisfying the set of goal constraints in (4) does not exceed the risk-tolerance level α. Incorporating similar constraints into $(\mathbf{SB-RN})$ provides the scenario-based version of (5), which we refer to as $(\mathbf{SB-JCCI})$. In this case, λ^s is set to 1 if the stochastic goal constraint $H\sum_{i\in I}\sum_{k\in K} a_{ri}\zeta^s_{ikt} \ge d^s_{rt}$ is violated for at least one pair of $r \in R,\ t \in \mathcal{T}$ under scenario s.

3.2.2 Multi-Stage Stochastic Model with Stage-wise Joint Chance Constraints The horizon-wise joint chance constraint (4) reflects a stringent restriction and may lead to conservative solutions which almost rule out on-demand cloud usage depending on the specified reliability level. Moreover, from a managerial point of view, it may make more sense to consider separate goal constraints for each provisioning stage. Thus, alternatively, we may ask for a *stage-wise* minimum service level met from reserved capacity through the following *multiple* joint chance constraints:

$$
\mathbb{P}\left(H\sum_{i\in I}\left(\sum_{k\in K} a_{ri}\zeta_{ikt}(\omega)\right) \ge D_{rt}(\omega), \quad \forall r \in R\right) \ge 1-\alpha_t, \quad \forall t \in \mathcal{T}. \tag{6}
$$

Making the risk-tolerance parameters $\alpha_t,\ t \in \mathcal{T}$, stage-dependent provides a mechanism of differentiating between the stages in the planning horizon and results in more flexibility from the standpoint of the decision maker. Integrating (6) into our base model leads to a chance-constrained variant, which can be reformulated as a MIP akin to the horizon-wise case:

$$
\begin{aligned}
&(\mathbf{NB-JCCII}) && && \\
&\text{minimize} && f(\mathbf{x},\mathbf{y}) && && \text{(7a)}\\
&\text{subject to} && (3b)-(3n), && && \text{(7b)}\\
& && H\sum_{i\in I}\sum_{k\in K} a_{ri}\zeta_{ikl} \ge d_{rl}\left(1-\lambda^l\right), && \forall r \in R,\ l \in \mathcal{N}\setminus\{1\}, && \text{(7c)}
\end{aligned}
$$

$$\sum_{l \in L_t} \rho^l \lambda^l \leq \alpha_t, \qquad \forall t \in \mathcal{T}, \tag{7d}$$

$$\lambda^l \in \{0,1\}, \qquad \forall l \in \mathcal{N} \setminus \{1\}\,. \tag{7e}$$

In this case, the stochastic goal constraints related to the satisfaction of demand are specified for each stage separately and require us to check whether all resource demands are covered by reservation contracts at each node at a particular stage. Therefore, differently from the horizon-wise case, the binary decision variables $\boldsymbol{\lambda}$ are indexed by node. The variable λ^l, $l \in \mathcal{N} \setminus \{1\}$, takes the value of 1 if the entire demand is *not* met from reserved capacity for at least one resource $r \in R$ at node l. The constraints (7c)-(7e) then collectively guarantee that the probability of violating the set of goal constraints in (6) is no more than the risk-tolerance level α_t at stage $t \in \mathcal{T}$. The corresponding scenario-based formulation is obtained by following a similar logic and designated as $(\mathbf{SB-JCCII})$.

REMARK 3.1 *Let A_t denote the event that $H\sum_{i\in I}\left(\sum_{k\in K} a_{ri}\zeta_{ikt}(\omega)\right) \geq D_{rt}(\omega)$ holds for all $r \in R$. Then, (4) and (6) can be respectively expressed as $\mathbb{P}(\bigcap_{t\in T} A_t) \geq 1-\alpha$ and $\mathbb{P}(A_t) \geq 1-\alpha_t,\ \forall t \in \mathcal{T}$. It is easy to see that a necessary—but not sufficient—condition for $\mathbb{P}(\bigcap_{t\in T} A_t) \geq 1-\alpha$ to hold is that $\mathbb{P}(A_t) \geq 1-\alpha,\ \forall t \in \mathcal{T}$. In other words, (4) implies (6) for $\alpha_t = \alpha,\ \forall t \in \mathcal{T}$. However, enforcing $\mathbb{P}(A_t) \geq 1-\alpha,\ \forall t \in \mathcal{T}$, would generally result in a horizon-wise reliability lower than $1-\alpha$. Put alternatively, ensuring that (6) implies (4) requires the individual reliability levels $1-\alpha_t,\ t \in \mathcal{T}$, to be sufficiently higher than $1-\alpha$. For instance, applying the well-known Boole inequality $\mathbb{P}(\bigcup_{t\in T} A_t) \leq \sum_{t\in T} \mathbb{P}(A_t)$, it is easy to show that (6) implies (4) if the risk-tolerance parameters satisfy the relation $\sum_{t\in\mathcal{T}} \alpha_t \leq \alpha$ (see, e.g., Prékopa, 2003; Elçi et al., 2018). Thus, under this setting of the risk-tolerance parameters $\alpha_t,\ t \in \mathcal{T}$, our optimization model with the stage-wise joint chance constraints (6) can be employed as an approximation of the corresponding optimization model with a single joint chance constraint (4). This approximation is a natural extension of the classical Bonferroni approximation, which is based on the particular choice of equal risk tolerances, i.e., $\alpha_t = \alpha/T,\ \forall t \in \mathcal{T}$. A part of our numerical investigations in Section 4.3 sheds more light into the relationship of the reliability levels attained through horizon-wise versus stage-wise chance-constrained formulations of SRPP–CC.*

3.2.3 Enhanced MIP Reformulations of the Joint Chance-Constrained Models The formulations $(\mathbf{NB-JCCI})$ and $(\mathbf{NB-JCCII})$ rely on the classical approach of reformulating the chance constraints via the big-M paradigm. The parameters d_{rl} in these models act as big-M coefficients because the left-hand sides of (5c) and (7c) are always non-negative. A major drawback of this reformulation approach is that it does not scale to large instances due to the well-known weakness of the linear programming (LP) relaxations of the big-M formulations. Our primary objective in this section is therefore to offer stronger and computationally effective reformulations of $(\mathbf{NB-JCCI})$ and $(\mathbf{NB-JCCII})$. To this end, we leverage recent methodological advances in chance-constrained optimization, which exploit the connection between the big-M constraints of the form (5c), (7c) and the mixing set—a well-known concept in integer programming. We first review the key results for the more commonly studied single-stage (static) models for completeness and ease of exposition.

Consider a single-stage joint chance-constrained optimization model of the general compact form

$$\text{minimize} \quad \{\mathbf{c}^T\mathbf{x} \ : \mathbb{P}(A_q\mathbf{x} \geq \bar{\xi}_q,\ q \in Q) \geq 1-\alpha,\ \mathbf{x} \in \mathcal{X}\}. \tag{8}$$

Here, only the right-hand side vector $\bar{\boldsymbol{\xi}} \in \mathbb{R}^{|Q|}$ is random, and without loss of generality we assume that $\bar{\boldsymbol{\xi}} \geq 0$. Also suppose that all vectors and the matrix A have the appropriate dimensions. Note that our dynamic model with the horizon-wise joint chance constraint would also fit the form in (8) if the decisions at each stage were not allowed to depend on the realizations of the uncertain parameters. Denoting the realizations of the random variable $\bar{\xi}_q$ by $\bar{d}_q^s,\ s \in S = [n]$, with the corresponding probabilities $\pi^s > 0$, the joint chance constraint in (8) can be reformulated by analyzing the following system (Luedtke et al., 2010):

$$\mathbf{y} = A\mathbf{x}, \quad y_q \geq \bar{d}_q^s(1-\lambda^s)\ \forall q \in Q,\ s \in S, \quad \sum_{s\in S} \pi^s\lambda^s \leq \alpha,\ \boldsymbol{\lambda} \in \{0,1\}^n. \tag{9}$$

Observe that setting the value of the binary variable λ^s to 0 guarantees that $y_q = A_q\mathbf{x} \geq \bar{d}_q^s$ for all $q \in Q$. Obviously, the decision variables $\mathbf{y}$ are not necessary but their use in (9) exposes the connection with the mixing set. More specifically, the system defined by the big-M type inequalities $y_q \geq \bar{d}_q^s(1-\lambda^s),\ \forall s \in S$, for a fixed $q \in Q$ is known as the mixing set. Luedtke et al. (2010) first leverage the knapsack inequality $\sum_{s\in S}\pi^s\lambda^s \leq \alpha$ to obtain a strengthening of the set $G_q = \{(y_q,\boldsymbol{\lambda}) \in \mathbb{R}_+ \times \{0,1\}^n : \sum_{s\in S}\pi^s\lambda^s \leq \alpha,\ y_q + \bar{d}_q^s\lambda^s \geq \bar{d}_q^s,\ \forall s \in S\}$, and then apply the mixing inequalities of Günlük and Pochet (2001) to obtain the strengthened star (mixing) inequalities presented below, which are facet-defining for G_q.

We first focus on a mixing set associated with the goal constraint indexed by q as a means of devising a strong MIP formulation for the joint chance-constrained problem (8). The pillar idea that enables the development of the strengthened star inequalities is that the realizations of $\bar{\boldsymbol{\xi}}$ can be sorted independently for each component without loss of generality. The sorted values of $\bar{\xi}_q$ are labeled as $\tilde{d}_q^1 \geq \tilde{d}_q^2 \geq \ldots \tilde{d}_q^n \geq 0 = \tilde{d}_q^{n+1}$, and we keep track of the original scenario index associated with the ith largest realization of $\bar{\xi}_q$ by defining the notation $J_q(i)$, i.e., $\tilde{d}_q^i = \bar{d}_q^{J_q(i)}$. The fundamental rationale exploited is that if $A_q\mathbf{x} \geq \tilde{d}_q^\ell$ does *not* hold, then $A_q\mathbf{x} \geq \tilde{d}_q^i$ is violated for all $i \in [\ell]$. Detecting the threshold index i'_q for any fixed set of decisions $\bar{\mathbf{x}}$ such that $A_q\bar{\mathbf{x}} < \tilde{d}_q^i$ for $i \in [i'_q - 1]$, and $A_q\bar{\mathbf{x}} \geq \tilde{d}_q^i$ for $i = i'_q, \ldots n$, is the central theme in the development of the strengthened mixing inequalities. To this end, we introduce an additional vector of binary variables $\boldsymbol{\eta}$ such that $\eta_q^i = 1$ for $i \in [i'_q - 1]$, and $\eta_q^i = 0$ for $i = i'_q, \ldots n$. The natural structure of the permissible values for the η-variables implies that $\eta_q^i \geq \eta_q^{i+1},\ i \in [n-1]$. Moreover, any feasible solution $\bar{\mathbf{x}} \in \mathcal{X}$ of (8) must ensure that $A_q\bar{\mathbf{x}} \geq \tilde{d}_q^{i_q^*+1}$, where $i_q^* + 1$ is the scenario index corresponding to the $(1-\alpha)$-quantile of $\bar{\xi}_q$; otherwise, there is no way of satisfying the joint chance constraint in (8). Consequently, $\eta_q^i = 0$ for all $i \in \{i_q^* + 1, \ldots, n\}$ in any feasible solution of (8), and these variables can be omitted from the formulation. Ultimately, the big-M constraints in (9) can be substituted by the following strengthened mixing inequalities (Luedtke et al., 2010):

$$y_q \geq \tilde{d}_q^1 - \sum_{i=1}^{i_q^*}(\tilde{d}_q^i - \tilde{d}_q^{i+1})\eta_q^i, \quad \forall\, q \in Q. \tag{10}$$

Finally, the relationship between the binary variables $\boldsymbol{\lambda}$ and $\boldsymbol{\eta}$ is captured via constraints of the form $\lambda^{J_q(i)} \geq \eta_q^i$, which follow from the observation that λ^s must take the value 1—indicating a violation of scenario s—if $s = J_q(i)$ for at least one variable η_q^i with the value of 1. Embedding the facet-defining inequalities (10) into a MIP reformulation of (8) in lieu of the big-M inequalities in (9) results in a tighter LP relaxation and faster solution times—see Section 4.3. We designate such enhanced formulations as "strong" and mark them with an "St" appended to the head of the formulation titles.

The idea of using the mixing inequalities is extended from the single- to the multi-stage case in Zhang et al. (2014). A simple observation to this end is that each mixing inequality is constructed for a single stochastic goal constraint—indexed by q in the single-stage case. In the setting with multiple decision stages, each goal constraint is concerned with whether the demand for a particular resource is fulfilled at a particular node of the scenario tree. Therefore, in the multi-stage context the index pair $r \in R,\ l' \in \mathcal{N}$ corresponds to q and indicates a single goal constraint. Moreover, it's essential to note that there is an implicit non-anticipativity property of the here-and-now type of decisions in the single-stage case—represented by the **x**-variables in (8). Zhang et al.'s extension of the mixing inequalities to the multi-stage framework accounts for this non-anticipativity based on the realization that all demands for r at stage t associated with the scenarios that go through the parent node of l' are pertinent while constructing a mixing inequality for resource r at node l' located at stage t. These scenarios share a common history up until stage $t-1$ and lend the non-anticipativity property to the resulting strong formulations. The notation we introduce to formalize this discussion follows closely that in the single-stage case. In particular, we focus on a node l' at stage $t \in \mathcal{T}$, and let l be its immediate ancestor; i.e., $A(l',1) = l$. Relying on the fundamental rationale of the mixing inequalities, we obtain a permutation $\mathbf{J}_{rl}$ of $S(l)$ for $r \in R$ describing a non-increasing order of the demand realizations $d^s_{rt},\ s \in S(l)$, i.e., $d^{J_{rl}(1)}_{rt} \geq d^{J_{rl}(2)}_{rt} \geq \cdots \geq d^{J_{rl}(n_l)}_{rt}$. For brevity of notation, these ordered demand realizations are represented by $\tilde{d}^1_{rl} \geq \tilde{d}^2_{rl} \geq \cdots \geq \tilde{d}^{n_l}_{rl} \geq 0 = \tilde{d}^{n_l+1}_{rl}$, where $\tilde{d}^i_{rl} = d^{J_{rl}(i)}_{rt}$ holds by definition. We then compute the index $i^*_{rl} = \max\{i \in [n_l] : \sum_{j=1}^{i} \pi^{J_{rl}(j)} \leq \alpha\}$ for the goal constraint associated with $r \in R,\ l' \in \mathcal{N}$, which identifies the maximal set of scenarios that can be violated under the horizon-wise joint chance constraint (4). This is analogous to the calculation of i^*_q in the single-stage setting. The binary decision variables $\eta^i_{rl}, i \in [i^*_{rl}+1]$, are defined in a similar vein such that $\eta^i_{rl} = 1$ indicates that the utilized capacity from the reserved instances for resource r falls short of the demand $\tilde{d}^i_{rl}$ at some child node of l. The mixing inequalities (10) are easily adapted to the multi-stage framework with this notation and provide the *strong* version of the node-based formulation (**NB − JCCI**), obtained by replacing the big-M type constraints (5c) by (11c)-(11g) based on the mixing idea:

(**StNB − JCCI**)

$$\begin{aligned}
&\text{minimize} && f(\mathbf{x},\mathbf{y}) && && \text{(11a)}\\
&\text{subject to} && (3b)-(3n), (5d)-(5e) && && \text{(11b)}\\
& && \eta^i_{rl} - \eta^{i+1}_{rl} \geq 0, && \forall r \in R, l \in \mathcal{N}, i \in [i^*_{rl}] : t(l) < T, && \text{(11c)}\\
& && \eta^{i^*_{rl}+1}_{rl} = 0, && \forall r \in R, l \in \mathcal{N} : t(l) < T, && \text{(11d)}\\
& && \lambda^{J_{rl}(i)} - \eta^i_{rl} \geq 0, && \forall r \in R, l \in \mathcal{N}, i \in [i^*_{rl}] : t(l) < T, && \text{(11e)}\\
& && H \sum_{i\in I}\sum_{k\in K} a_{ri}\zeta_{ikl'} + \sum_{i=1}^{i^*_{rl}} (\tilde{d}^i_{rl} - \tilde{d}^{i+1}_{rl})\eta^i_{rl} \geq \tilde{d}^1_{rl}, && \forall r \in R, l' \in \mathcal{N}\setminus\{1\} : l = A(l',1), && \text{(11f)}\\
& && \eta^i_{rl} \in \{0,1\}, && \forall r \in R, l \in \mathcal{N}, i \in [i^*_{rl}+1] : t(l) < T. && \text{(11g)}
\end{aligned}$$

If the ith largest realization of demand for $r \in R$ cannot be met at a child node of l, then none of the larger demand realizations is satisfied either. This relation is established in (11c). However, as a necessary condition for fulfilling the joint chance constraint (4), we cannot fall short of the $(i^*_{rl}+1)$st largest realization of demand for r at a child node of l—see (11d). If η^i_{rl} is set to 1, then the demand for r under scenario $s = J_{rl}(i)$ is not met at some child node of l as stipulated by the constraints (11e) and λ^s is forced to 1. The mixing inequalities (10) take the form in (11f), where the first term expresses the total amount of resource r in use from reserved instances at node l' and corresponds to y_q in (10). Matching modeling constructs based on the mixing set can be employed to transform the scenario-based formulation (**SB − JCCI**) into its strong version (**StSB − JCCI**).

The strengthening of the stage-wise joint chance-constrained formulation (**NB − JCCII**) follows suit with the development above—we only need a few extra provisions to account for the differences in structure between the joint chance constraints (4) and (6). First, the threshold index i^*_{rl} for a pair $r \in R,\ l \in \mathcal{N}$ with $t(l) = t-1$ is calculated with respect to α_t instead of α. We also need to keep track of the node corresponding to scenario $s \in S$ at stage $t \in \mathcal{T}$ through the notation $\hat{l}^t_s$. Then, we replace the big-M type constraints (7c) in ($\mathbf{NB-JCCII}$) by (11c)-(11d), (11f)-(11g), and the following variant of (11e) to arrive at the strengthened formulation ($\mathbf{StNB-JCCII}$):

$$\lambda^{\hat{l}^{t(l)+1}_{J_{rl}(i)}} - \eta^i_{rl} \geq 0, \qquad \forall r \in R,\ l \in \mathcal{N},\ i \in [i^*_{rl}] : t(l) < T. \tag{12}$$

This constraint ensures that if the demand for r under scenario $J_{rl}(i)$ is not met at child node l' of node l—as captured by $\eta^i_{rl} = 1$, then $\lambda^{l'}$ is set to 1. Differently from ($\mathbf{StNB-JCCI}$), here we must identify the precise child node of l, where scenario $J_{rl}(i)$ is violated. The scenario-based counterpart of ($\mathbf{StNB-JCCII}$) is obtained in a parallel way and labeled as ($\mathbf{StSB-JCCII}$).

A Further Consideration. Recall that $d^s_{rt(l)} = d_{rl},\ \forall r \in R, l \in \mathcal{N} \setminus \{1\}, s \in S(l)$—the demand realizations for $r \in R$ associated with all scenarios in $S(l)$ at a given node l at stage $t(l)$ are all identical. Unfortunately, this gives rise to repetitions in the ordered demand values $d^{J_{rl}(1)}_{rt} \geq d^{J_{rl}(2)}_{rt} \geq \cdots \geq d^{J_{rl}(n_l)}_{rt}$ used in the mixing set calculations and results in i^*_{rl} values higher than necessary. Consequently, the size of the decision vector $\boldsymbol{\eta}$ and the number of constraints in the strong formulations presented in this section are bloated. We now offer a remedy for this potential drawback and investigate the computational implications in Section 4.3. Previously, $\mathbf{J}_{rl}$ denoted a permutation of $S(l)$ for $r \in R$ describing a non-increasing order of the demand realizations $d^s_{rt},\ s \in S(l)$. The key to eliminate the repeated values in the ordered set of demands is to re-define the permutation $\mathbf{J}_{rl}$ with respect to the d_{ri} values at all child nodes $i \in \mathcal{N}^c_l$ of l. The resulting ordered set of demand values $d_{rJ_{rl}(1)} \geq d_{rJ_{rl}(2)} \geq \cdots \geq d_{rJ_{rl}(|\mathcal{N}^c_l|)}$ are still designated as $\tilde{d}^1_{rl} \geq \tilde{d}^2_{rl} \geq \cdots \geq \tilde{d}^{|\mathcal{N}^c_l|}_{rl}$ for consistency and brevity of notation. Under its updated definition, $J_{rl}(i)$ maps i to the node with the ith largest demand realization in $\mathcal{N}^c_l$ instead of a scenario in $S(l)$ based on its original definition. The astute reader will realize that these changes lead to a minimal set of modifications in the strong formulations, where the index i^*_{rl} is determined as $\max\{i \in [\ |\mathcal{N}^c_l|\ ] : \sum_{j=1}^i \rho^{J_{rl}(j)} \leq \alpha\}$ and $\max\{i \in [\ |\mathcal{N}^c_l|\ ] : \sum_{j=1}^i \rho^{J_{rl}(j)} \leq \alpha_t\}$ for the alternate strong formulations $(\mathbf{StNB-JCCI})^a$ and $(\mathbf{StNB-JCCII})^a$, respectively:

$$\lambda^s - \eta^i_{rl} \geq 0, \qquad \forall r \in R,\ l \in \mathcal{N},\ i \in [i^*_{rl}],\ s \in S(J_{rl}(i)) : t(l) < T, \text{ and}$$
$$\lambda^{J_{rl}(i)} - \eta^i_{rl} \geq 0, \qquad \forall r \in R,\ l \in \mathcal{N},\ i \in [i^*_{rl}] : t(l) < T,$$

replace (11e) in $(\mathbf{StNB-JCCI})^a$ and (12) in $(\mathbf{StNB-JCCII})^a$, respectively.

In the next section, we make good on our promise of providing evidence that the models proposed for SRPP–CC in this study can be solved to optimality by a readily available commercial solver for instances of meaningful size. In particular, keeping up with the good modeling practices relying on recent methodological progress pays off for solving large-scale chance-constrained versions of SRPP–CC. For more speed or better scalability, however, one might resort to a branch-and-cut method which would initialize with a small subset of the mixing inequalities and then add the absent ones as required by solving an appropriate separation subproblem in the search tree. Such a custom-tailored implementation would be time-intensive from a development perspective but may result in potential gains in the solution times (Zhang et al., 2014). For the risk-neutral case with continuous decision variables and no chance constraints, Benders-type decomposition methods are available. The nested Benders decomposition (Louveaux, 1980) and its Monte Carlo sampling-based version—known as the stochastic dual dynamic programming (Pereira and Pinto, 1991)—are the most commonly employed specialized approaches to solve risk-neutral multi-stage stochastic linear programs.

4. Computational Study

Many modern practical applications require quick turnaround times on mathematical modeling, solution, and analysis. In this environment, a critical enabler of success is the ability to model and solve optimization problems via off-the-shelf solvers without going through a development of custom-tailored solution algorithms. A further benefit of this approach is that the speedups in solver technology are exploited immediately without further effort. Against this background, the overarching goal of our computational study is to demonstrate that good modern modeling practices enable the cloud customer to define and solve variants of the cloud resource provisioning problem, depending on the business objectives. This is accomplished in two parts. In the first part, we investigate the computational effectiveness of the various alternate modeling techniques with respect to the solution times. Specifically, we benchmark the scenario-based formulations against their node-based counterparts and explore the impact of the alternate ways of incorporating the non-anticipativity constraints. For our chance-constrained models, we look into the benefits of employing the strong formulations. The second aspect that we focus on in the numerical experiments is the value of dynamic modeling, and in the context of chance-constrained optimization, the trade-off between reliability and the expected total cost. To this end, we examine the expected total cost, its composition, and the attained reliability levels of the stochastic goal constraints in different models. Note that we relax the integrality restrictions on the y-, ζ-, and x-variables in the analyses in this section so that solving instances with meaningful time horizons remains within the reach of the proposed formulations. This simplification does not detract anything significant from the value of the solutions as rounding fractional values obtained for general integer variables in a tactical level planning problem is a plausible option.

All runs were executed on a workstation with two 2.3 GHz Intel® Xeon® E5-2630 processors and 64 GB of memory running on the 64-bit Windows 10 Professional operating system. The formulations were implemented and solved via AMPL 12.2.0 running on CPLEX 12.6.2 with its default settings and a time limit of one hour. We allowed CPLEX to use up to 12 GB of memory for the branch-and-bound tree and at most 4 threads in parallel by setting the parameters *treememory* and *threads* to 12,000 (in megabytes) and 4, respectively. The solution times are elapsed times in seconds. If CPLEX terminates without proving optimality within the time limit, then we retrieve the relative optimality gap (ROG) from the solver calculated as ROG $= (UB - LB)/UB$, where LB and UB refer to the best available lower and upper bounds, respectively. All figures reported in the tables in this section are averaged over five randomly generated instances unless otherwise stated.

4.1 Data Generation

Data for experimentation with the different formulations we developed has been generated in line with the recent practices at AWS. In particular, we relied on actual AWS instances for the physical attributes of the VMs, and cost/price and capacity parameters (Amazon Web Services, Inc., 2018c,b) to better approximate the choices facing a cloud customer. The features of the four families of AWS instances we employ in this study are listed in Table 1. The instance families are categorized with respect to their purpose of usage and release dates—see Columns 1 and 2. The general-purpose instances offer a balance between compute, memory, and network resources (Amazon Web Services, Inc., 2015) while the memory-optimized instances are better suited to memory-intensive, latency-sensitive applications (Amazon Web Services, Inc., 2016). Instance families released later in time obviously benefit from newer technology and provide better performance at lower cost. AWS restricts the number of on-demand and reserved instances that may be provisioned at any given point in time. Such limits are imposed both on instance families—see Columns 4-5—and individual instance types as specified in Columns 8-9. “FCap” and “ICap” respectively stand for family and individual instance type capacities preceded either by “OD-” for on-demand or “R-” for reserved-instance usage. We consider two resource types in all experiments: memory and CPU. For each instance type, the available memory and CPU capacity are indicated in Columns 10-11, respectively, where the EC2 Compute Unit (ECU) is a standardized measure of CPU speed as defined by AWS (Amazon Web Services, Inc., 2018a). The official terms of standard reserved instances are one or three years. In this study, we opt for a shorter denominator of a single quarter with 2-quarter reserved-instance contracts and consider planning horizons up to thirteen quarters. To this end,

Purpose	**Release Date**	**Family #**	**OD–FCap**	**R–FCap**	**Name**	**Inst. ID**	**OD–ICap**	**R–ICap**	**Memory (GB)**	**ECU**	**OD–Co ($/hr)**	**R–Co ($/hr)**
General Purpose	Jun 2015	1	20	20	m4.large	1	20	20	8	6.5	0.246	0.157
					m4.xlarge	2	20	20	16	13	0.491	0.315
					m4.2xlarge	3	20	20	32	26	0.983	0.630
					m4.4xlarge	4	10	20	64	53.5	1.966	1.260
					m4.10xlarge	5	5	20	160	124.5	4.914	3.140
	Oct 2012	2	20	20	m3.medium	6	20	20	3.75	3	0.130	0.084
					m3.large	7	20	20	7.5	6.5	0.259	0.167
					m3.xlarge	8	20	20	15	13	0.518	0.333
					m3.2xlarge	9	20	20	30	26	1.036	0.664
Memory Optimized	Apr 2014	3	20	20	r3.large	10	20	20	15.25	6.5	0.291	0.199
					r3.xlarge	11	20	20	30.5	13	0.583	0.360
					r3.2xlarge	12	20	20	61	26	1.045	0.709
					r3.4xlarge	13	10	20	122	52	1.944	1.248
					r3.8xlarge	14	5	20	244	104	3.500	1.666
	Nov 2016	4	20	20	r4.large	15	20	20	15.25	7	0.225	0.170
					r4.xlarge	16	20	20	30.5	13.5	0.450	0.340
					r4.2xlarge	17	20	20	61	27	0.900	0.681
					r4.4xlarge	18	10	20	122	53	1.800	1.362
					r4.8xlarge	19	5	20	244	99	3.600	2.723
					r4.16xlarge	20	1	20	488	195	7.200	5.447

Table 1: Features of the AWS EC2 instances used for data generation.

we scale the hourly costs of the 1-year standard AWS Windows reserved instances—see the last column of Table 1—by a factor of 1.15, also making sure that the hourly rates of the reserved instances remain below those of their on-demand counterparts in the next-to-last column.

In our computational experiments, we restrict our attention to problems with random demand and keep the hourly VM instance costs constant throughout. The cloud prices have enjoyed a downward trend over the past (Amazon Web Services, Inc., 2017, 2018d) and are arguably more predictable than demand. If desired, some deterministic discount factor estimated from past data may be applied to the hourly costs every so many quarters. The demand generation scheme is best explained in the context of the scenario tree in Figure 2. We follow this tree structure in all instances generated. For both resource types, we assume that the current base demand at the root node is equivalent to a continuous usage of 50 m4.2xlarge instances over 90 days—a single quarter equal to $H = 2,160$ hours. The demands for CPU capacity and storage are expressed in ECU-hours and GB-hours, respectively. Starting with this base demand at $t = 0$, we multiply the current demand by a factor determined by the branch we follow on the scenario tree. If the relative change in the ECU demand from the current to the next period is expected to be a decrease (D), then the multiplicative factor is drawn from a continuous uniform distribution $U(0.90, 0.95)$. The scaling factors on the branches for moderate (MI) and strong increase (SI) follow uniform distributions $U(1.05, 1.10)$ and $U(1.15, 1.20)$, respectively. The corresponding distributions for the memory demand are specified as $U(0.85, 0.90)$ for D, $U(1.00, 1.05)$ for MI, and $U(1.10, 1.15)$ for SI. We note that the scenario-based stochastic programming methodology is very general and allows us to construct data from any distribution and also model the correlations between the random parameters. The choice of uniform distributions throughout just reflects our intention of presenting an intuitive scenario generation procedure to illustrate the value of our models. Moreover, uniform input models are commonly preferred in practice if there is limited information.

Table 2 displays the conditional probabilities γ_{ij} attached to the edges of the scenario tree. For instance, if we arrive at node i over an edge labeled as MI, then $\gamma_{ij} = 0.2$ for an edge (i, j) labeled as SI. The probabilities for the edges emanating from the root node follow those in row MI—equivalent to assuming that the incoming edge of the root is labeled as MI.

	D	MI	SI
D	0.3	0.7	0
MI	0.2	0.6	0.2
SI	0	0.7	0.3

Table 2: Branching probabilities in the scenario tree generation.

4.2 Risk-Neutral Models In this section, we investigate the scalability of the risk-neutral models from Section 3.1. For the scenario-based formulation ($\mathbf{SB-RN}$), we experiment with both variants of the non-anticipativity constraints. ($\mathbf{SB-RN}$) embedded with (2a) or (2b) is designated as ($\mathbf{SB_{NA1}-RN}$) or ($\mathbf{SB_{NA2}-RN}$), respectively. Table 3 attests to the better scalability and dominance of the node-based formulation ($\mathbf{NB-RN}$) over its scenario-based counterpart ($\mathbf{SB-RN}$), regardless of the form of the non-anticipativity constraints. In particular, the figures in the last column indicate that the solution time of the better performing scenario-based formulation is 2 to 6 times higher on average compared to that of ($\mathbf{NB-RN}$). Moreover, the dominance of ($\mathbf{NB-RN}$) becomes more pronounced with an increasing number of provisioning stages, and ($\mathbf{NB-RN}$) is the clear formulation choice for the risk-neutral setting of SRPP–CC. As for the scenario-based formulations, the solution times favor (2a) over (2b), and going forward, we will only report results based on this variant of the non-anticipativity constraints for the scenario-based formulations.

Another clear insight from Table 3 is the detrimental impact of the length of the planning horizon on the solution times of all models. Instances with $T \leq 8$ are solved in less than 10 seconds on average across the board; however, the solution times pick up quickly after that due to the tremendous growth of the number of scenarios with the number of stages—as might be expected. Ultimately, the results demonstrate promising potential for practical use as instances with sufficiently long planning horizons of up to $T = 13$ quarters are handled successfully.

4.2.1 Value of Dynamic & Uncertainty-Adaptive Modeling The dynamic and uncertainty-adaptive structure of the models we propose in this work was emphasized in Section 3.1. An essential aspect of our models is that we can respond to the fluctuations in demand by adjusting the reservation levels over time and / or acquiring on-demand capacity. In this section, we quantify the positive impact of this modeling approach on the bottom line of the cloud customer. To this end, we formulate a *semi-dynamic* version of our risk-neutral multi-stage optimization model by fixing the procurement schedule of all reservation contracts at the start of the planning horizon. That is, we decide ahead of time—before any future uncertainty is resolved—when and which contracts to purchase. However, as before, the decisions related to the utilization of the reserved instances and on-demand usage are deferred until the period of consumption. By definition, the actual utilizations can only be set after observing the uncertainty—this hybrid nature is the reason we refer to this model as semi-dynamic. Through benchmarking the semi-dynamic risk-neutral model specified below against the dynamic and uncertainty-adaptive model of Section 3.1 from the perspectives of cost and solution structure, we hope to demonstrate the value of the latter to the reader.

		Solution Time			
$\lvert S\rvert$	$\lvert\mathcal{T}\rvert$	$(\mathbf{NB}-\mathbf{RN})$	$(\mathbf{SB_{NA1}}-\mathbf{RN})$	$(\mathbf{SB_{NA2}}-\mathbf{RN})$	$\mathbf{SB_{Best}/NB}$
41	4	0.05	0.10	0.10	1.9
99	5	0.10	0.28	0.27	2.7
239	6	0.23	0.78	0.81	3.4
577	7	1.05	2.72	2.94	2.6
1393	8	2.53	7.69	8.93	3.0
3363	9	8.12	26.29	336.96	3.2
8119	10	25.66	103.71	500.58	4.0
19601	11	101.24	404.89	455.82	3.9
47321	12	264.07	1632.10	1803.89	6.1
114243	13	1621.39	†	†	–
275807	14	$\geq$ 3600‡	†	†	–

† : `CPLEX` crashes with no progress.

‡ : `CPLEX` terminates with no integer feasible solution.

Table 3: Benchmarking the computational performance of the risk-neutral formulations.

To obtain the scenario-based risk-neutral semi-dynamic formulation $(\mathbf{SB_{SDyn}}-\mathbf{RN})$ from its dynamic and uncertainty-adaptive counterpart $(\mathbf{SB}-\mathbf{RN})$, the non-anticipativity constraints (11) for the y-variables are dropped, and the scenario indices are eliminated from the variables y^s_{ikt}. A few other obvious implied changes are applied to the objective and the constraints as well. On the other hand, constructing the node-based semi-dynamic formulation $(\mathbf{NB_{SDyn}}-\mathbf{RN})$ out of $(\mathbf{NB}-\mathbf{RN})$ only involves imposing the additional set of constraints (13) below in order to remove the uncertainty-adaptiveness from the y-variables. These constraints ensure that the y-variables associated with the same stage take the same value, where we assume that the nodes at a given stage are numbered sequentially:

$$y_{ikl} = y_{ik(l+1)}, \qquad \forall i \in I,\ k \in K,\ l \in \mathcal{N} \setminus \{1, |\mathcal{N}|\} : t(l) = t(l+1). \tag{13}$$

In Table 4, "TC" and "ODC" respectively stand for "total cost" and "on-demand cost." The underlying set of instances are identical to those in Table 3. The absolute total cost figures in Columns 3 and 5 along with the percentage increase in total cost in the semi-dynamic model over that in the original risk-neutral model computed in the last column affirm the expected. The ability to adjust the reserved instance quantities in reaction to the revealed uncertainty is absolutely crucial as T increases. Decisions fixed at the outset account quite poorly for the distant future. A rational characteristic of the solution structure in the semi-dynamic model is that it avoids overcommitting to reserved instances. Consequently, the on-demand cost in the semi-dynamic model makes up a larger portion of the total cost both percentage-wise—as evident from Columns 4 and 6—and in absolute numbers. A curious observation in Table 3 is that the ratio of the on-demand cost to the total cost keeps oscillating between lower and higher numbers for even and odd T, respectively. This is a consequence of the exclusive use of 2-quarter reservation contracts, where it makes more sense to satisfy the demand in the final period directly through on-demand usage rather than paying for reserved capacity that would remain underutilized. Overall, the results in Table 4 clearly substantiate the benefits of adapting the decisions to uncertainty that is resolved over time.

4.3 Chance-Constrained Formulations Three aspects are investigated in the context of the computational performance of the proposed chance-constrained formulations: solution time, the cost of relia-

		Dynamic		**Semi-Dynamic**		
$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{T}\rvert$	**TC (\$)**	**ODC / TC (%)**	**TC (\$)**	**ODC / TC (%)**	**% Increase in TC**
41	4	255,134	6.9	263,168	10.5	3.1
99	5	379,423	26.8	383,344	22.9	1.0
239	6	415,428	7.4	436,373	12.9	5.0
577	7	534,445	18.7	551,754	19.4	3.2
1393	8	565,997	7.2	601,952	14.8	6.4
3363	9	746,895	15.6	780,707	21.2	4.5
8119	10	816,292	8.6	894,335	16.5	9.5
19601	11	980,224	13.7	1,173,715	15.7	19.7
47321	12	1,056,323	9.3	1,482,860	10.4	40.7
114243	13	1,221,147	13.3	1,800,969	10.9	47.1

Table 4: Demonstrating the value of dynamic and uncertainty-adaptive decision making.

bility, and the attained reliability levels of the stochastic goal constraints. Table 5 depicts the solution time and optimality gap performance of the various formulations for the chance-constrained versions of SRPP–CC. In order to get a sense of the relative level of difficulty of the horizon-wise joint chance-constrained formulations versus their stage-wise counterparts, the horizon-wise reliability level $1-\alpha$ and all stage-wise reliability levels $1-\alpha_t,\ t \in \mathcal{T}$, are fixed to 0.90 in all instances underlying Table 5. In the "Time" column, if only a subset of the five instances associated with a cell hits the time limit without a proven optimal solution, then the number of such instances is indicated in parentheses, and the corresponding ROG value is computed over these instances only. On the other hand, all instances that terminate due to the time limit contribute 3600 seconds to the calculation of the average solution time. An immediate observation from Table 5 is that the decoupling across time in the stage-wise formulations gives rise to a less challenging structure and results in quicker solution times. Nevertheless, this benefit is more than offset by the fast growth in the formulation size with T, and our ability to solve either chance-constrained version of SRPP–CC to proven optimality within the allotted time maxes out at $T = 7$—even though the ROG values at termination for $T = 8$ are quite small. A direct comparison with Table 3 exposes the well-documented computational difficulties of solving chance-constrained formulations. While instances with up to $T = 13$ quarters can be solved to proven optimality in one hour in Table 3, this number drops quickly to $T = 7$ in Table 5. The figures in this table also underline the superior performance of the strong formulations. Replacing the big-M constraints in the initial chance-constrained formulations by the strengthened mixing inequalities extends the maximum horizon length that can be tackled by two quarters, which may be critical in practice. However, the further potential enhancement pointed out for the strong node-based formulations at the end of Section 3.2.3 offers no additional benefits—compare (**StNB – JCCII**) and (**StNB – JCCII**)^a for illustrative purposes. To conclude the analysis of Table 5, we emphasize that the major speedup provided by the strong formulations enables the decision maker to experiment with different parameter settings in substantially less time and helps shorten the decision-making cycle.

Table 6 delves into the structure of the optimal provisioning plan as a function of the reliability level under both types of chance constraints. A conspicuous fact supported by the numbers in Column "TC" is that requiring extra reliability is costly—TC increases with decreasing α or α_t. The additional burden in total cost over that in the risk-neutral model ranges from 3% to 13% as reported in Column "% Increase

		Horizon-wise			Stage-wise		
$\lvert S\rvert$	$\lvert\mathcal{T}\rvert$	Formulation	Time	ROG (%)	Formulation	Time	ROG (%)
41	4	**(StNB – JCCI)**	0.35		**(StNB – JCCII)**	0.29	
					(StNB – JCCII)^a	0.32	
		(StSB – JCCI)	0.39		**(StSB – JCCII)**	0.35	
		(NB – JCCI)	0.77		**(NB – JCCII)**	0.56	
		(SB – JCCI)	0.98		**(SB – JCCII)**	1.47	
99	5	**(StNB – JCCI)**	2.45		**(StNB – JCCII)**	1.34	
					(StNB – JCCII)^a	1.22	
		(StSB – JCCI)	2.87		**(StSB – JCCII)**	1.55	
		(NB – JCCI)	29.31		**(NB – JCCII)**	10.43	
		(SB – JCCI)	37.73		**(SB – JCCII)**	36.55	
239	6	**(StNB – JCCI)**	12.15		**(StNB – JCCII)**	8.12	
					(StNB – JCCII)^a	7.82	
		(StSB – JCCI)	12.99		**(StSB – JCCII)**	11.41	
		(NB – JCCI)	≥ 3600	0.258	**(NB – JCCII)**	2427.34(3)	0.222
		(SB – JCCI)	≥ 3600	0.241	**(SB – JCCII)**	≥ 3600	0.718
577	7	**(StNB – JCCI)**	1556.94		**(StNB – JCCII)**	125.14	
					(StNB – JCCII)^a	129.07	
		(StSB – JCCI)	2348.29(1)	0.044	**(StSB – JCCII)**	316.29	
1393	8	**(StNB – JCCI)**	≥ 3600	0.450	**(StNB – JCCII)**	≥ 3600	0.026
					(StNB – JCCII)^a	≥ 3600	0.023
		(StSB – JCCI)	≥ 3600	0.476	**(StSB – JCCII)**	≥ 3600	0.047

Table 5: Benchmarking the computational performance of the joint chance-constrained formulations.

in TC." A less visible but expected observation is that the cost of reliability is higher for the horizon-wise chance-constraints because these reflect a more stringent restriction on the procurement of reservation contracts as discussed at the start of Section 3.2.2. This observation is revealed when the TC figures for the horizon- and stage-wise chance-constrained formulations corresponding to the same $|S|$, $|\mathcal{T}|$, and reliability level values are compared. The main managerial takeaway from Table 6 is probably that the chance-constrained versions of SRPP–CC equip the cloud broker and customer with a powerful tool that helps control the on-demand component of the optimal provisioning plan. The decrease in the on-demand cost over that in the risk-neutral model is no less than 78% anywhere in the last column.

Figure 3 displays the results of our effort to portray the interplay between the total cost / the cost structure of the cloud resource provisioning plan and the attained reliability levels in the chance-constrained models by solving an instance with $T = 4$ repeatedly for different α and $\alpha_1 = \alpha_2 = \ldots = \alpha_T$ values. The attained horizon-wise and stage-wise reliability levels are then calculated for each optimal solution obtained. The data labels in the plots indicate the α and α_t values associated with the corresponding optimal solution in the horizon-wise and stage-wise chance-constrained models, respectively, and the abbreviation "JCC" in the legends stands for joint chance constraint. For starters, the trends in total cost and cost structure evident in Table 6 are also clearly visible in Figure 3. Both types of chance-constrained formulations realize gains in reliability and dips in the percentage of on-demand cost at the expense of elevated total costs. Figures 3a and 3c confirm that the premium the cloud customer pays for a minimum guaranteed level of reliability is always at least as high in the case of horizon-wise chance constraints as long as all α and α_t values are identical—as also discussed in the context of Table 6 above. On the

	$\|\mathcal{S}\|$	$\|\mathcal{T}\|$	$1-\alpha$ / $1-\alpha_t,\ \forall t \in \mathcal{T}$	**TC ($)**	**% Increase in TC**	**% Decrease in ODC**
Risk-Neutral	41	4	-	255,134	-	-
	99	5	-	379,423	-	-
	239	6	-	415,428	-	-
	577	7	-	534,445	-	-
Horizon-wise	41	4	0.95	272,947	7.0	96.0
			0.90	270,119	5.9	92.7
			0.85	267,814	5.0	89.1
	99	5	0.95	425,616	12.2	98.6
			0.90	421,357	11.1	93.3
			0.85	417,179	10.0	84.5
	239	6	0.95	445,580	7.3	95.6
			0.90	441,423	6.3	90.0
			0.85	438,056	5.5	84.0
	577	7	0.95	603,576	12.9	93.6
			0.90	596,671	11.6	87.6
			0.85	590,555	10.5	80.9
Stage-wise	41	4	0.95	272,902	7.0	94.4
			0.90	268,366	5.2	82.1
			0.85	264,603	3.7	77.8
	99	5	0.95	423,229	11.6	98.6
			0.90	416,852	9.9	97.4
			0.85	412,133	8.6	96.3
	239	6	0.95	442,637	6.6	93.5
			0.90	435,436	4.8	84.1
			0.85	429,502	3.4	78.1
	577	7	0.95	596,525	11.6	96.8
			0.90	584,552	9.4	94.3
			0.85	575,839	7.7	92.1

Table 6: Demonstrating the impact of chance constraints on the cost structure of the cloud customer.

flip side, the horizon-wise chance-constrained model ensures a larger decrease in the on-demand cost as a percentage of TC for corresponding α and α_t values—see Figures 3b and 3d.

The graphs in Figure 3 deliver a good illustration of the points in Remark 3.1 as well. In particular, mandating the stage-wise chance constraints $\mathbb{P}(A_t) \geq 1-\alpha,\ \forall t \in \mathcal{T}$, may result in a substantially lower horizon-wise reliability than $1-\alpha$—see Figures 3a and 3b, where $\alpha_t = \alpha = 0.1,\ \forall t \in \mathcal{T}$, achieves a horizon-wise reliability of less than 0.80. As expected, the horizon-wise chance-constrained model attains a reliability level at least as large as 0.90 for $\alpha = 0.1$. This difference would clearly be more pronounced for instances with more provisioning stages as the probability of a larger number of intersecting events would be smaller. A similar reasoning concludes that the solution from the horizon-wise chance-constraint model for a given α leads to higher stage-wise reliability levels compared to those obtained though the stage-wise chance-constrained model for $\alpha_t = \alpha,\ \forall t \in \mathcal{T}$. For instance, in Figures 3c and 3d the *minimum* reliability level over all stages attained by the optimal solution of the horizon-wise chance-constrained model for $\alpha = 0.3$ is rather close to 0.8—a clear indication of the conservatism of the horizon-wise chance constraint. A final insight gathered from studying Figures 3a and 3b is that employing the stage-wise chance-constrained model as an approximation of its horizon-wise counterpart by relying on the Boole inequality as mentioned in Remark 3.1 may be more demanding than intended. Note that the theoretically guaranteed horizon-wise reliability levels associated with $\alpha_1 = \ldots = \alpha_4 = 0.05$ and 0.10 in the solution of

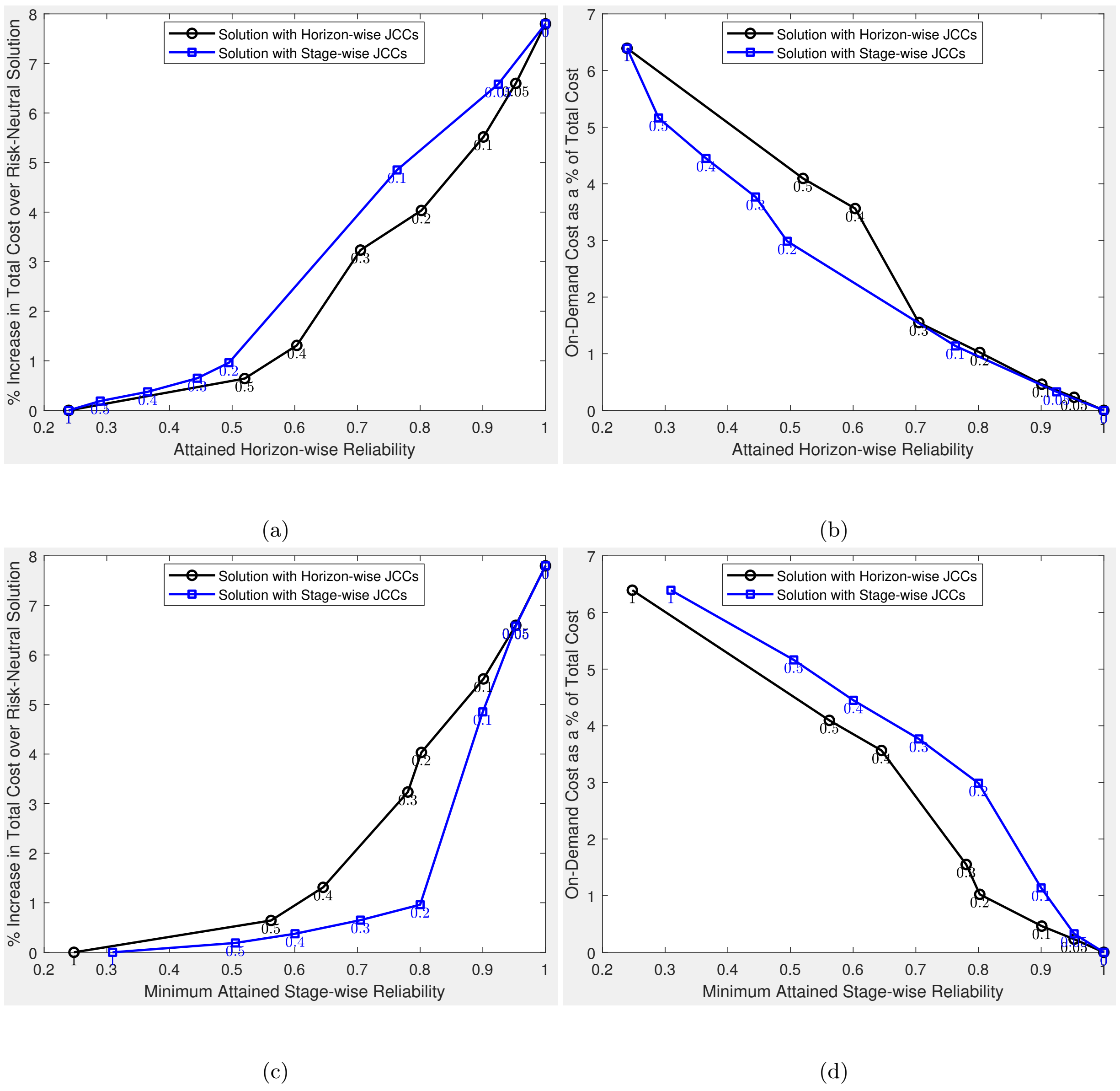


Figure 3: Detailed analysis of cost versus attained reliability in the chance-constrained formulations.

the stage-wise chance-constrained model are 0.80 and 0.60, respectively. The actual attained horizon-wise reliability levels, however, exceed 0.90 and 0.70, respectively.

Overall, many considerations factor into the use of chance-constrained models and their associated formulations. A couple of critical observations stand out based on the analyses in this section and would impact the success of the modeling and implementation. First and foremost, the decision maker must be acutely aware that the horizon-wise and stage-wise chance constraints can behave quite differently in terms of their effect on the total cost, its composition, and the reliability they provide. Second, employing the strengthened mixing inequalities instead of the naive big-M inequalities pays off and enables a faster experimentation cycle for decision making. Third, it is an insightful exercise to calculate the attained reliability levels for a given solution—as is done for Figure 3—in order to produce an appropriate cloud resource provisioning plan that satisfies the business needs with an acceptable cost structure.

5. Conclusions. This study describes a new and critical business problem posed by recent advances and trends in the information technology industry—the resource provisioning problem of a cloud consumer from an IaaS cloud in an uncertain environment. Demonstrating that the essence of this real problem can be captured by multi-stage stochastic programming—an advanced modeling framework from operations research—is the dominating contribution of our work. Moreover, we illustrate that the iterative decision-making cycle of mathematical modeling, solution, analysis, and the re-adjustment of model parameters can be performed quickly by readily available commercial software. We hope that practitioners will appreciate the value of and apply these models. A further point not to be overlooked is that recent theoretical advances in the literature should be followed closely as they may have very tangible benefits for the practice of operations research in the industry.

In our computational study, we set out to demonstrate the use and performance of the various modeling options and their associated formulations for different settings that may be considered in the context of SRPP–CC. Modeling insights are gleaned by analyzing several trade-offs carefully, and some managerial takeaways are provided. We hope that the presented results will help the reader and the practitioners choose the right modeling framework and a corresponding formulation depending on the business objectives and the computational resources available.

A possible avenue of future research is to incorporate risk measures into the objective function, following the increasing interest in risk-averse multi-stage stochastic programs (see, e.g., Philpott and de Matos, 2012). For example, the risk-neutral objective function can be replaced by a risk-averse variant featuring the widely-applied risk measure conditional value-at-risk (Rockafellar and Uryasev, 2000), and the presented models can be extended to this case.

References

Amazon Web Services, Inc. (2015). Introducing M4 Instances and Lower Amazon EC2 Instance Prices. Retrieved from `https://amzn.to/2vCF9JT`. Accessed on: 2018-08-23.

Amazon Web Services, Inc. (2016). Introducing Amazon EC2 R4 Instances, the Next Generation of Memory-Optimized Instances. Retrieved from `https://amzn.to/2vK0NMr`. Accessed on: 2018-08-23.

Amazon Web Services, Inc. (2017). AWS News Blog—EC2 Price Reductions – Reserved Instances & M4 Instances. Retrieved from `https://amzn.to/2NprMDq`. Accessed on: 2018-08-26.

Amazon Web Services, Inc. (2018a). Amazon EC2 FAQs—Hardware Information. Retrieved from `https://amzn.to/2uBD9AT`. Accessed on: 2018-08-23.

Amazon Web Services, Inc. (2018b). Amazon EC2 FAQs—Instance Capacities. Retrieved from `https://goo.gl/B9rQXH`. Accessed on: 2018-08-23.

Amazon Web Services, Inc. (2018c). Amazon EC2 Instance Types. Retrieved from `https://goo.gl/FboHap`. Accessed on: 2018-08-23.

Amazon Web Services, Inc. (2018d). AWS News Blog—Category: Price Reduction. Retrieved from `https://amzn.to/2LsOB7u`. Accessed on: 2018-08-26.

Amiri, A. (2016). Application placement and backup service in computer clustering in Software as a Service (SaaS) networks. *Computers & Operations Research*, 69:48–55.

Arabnejad, H., Barbosa, J. G., and Prodan, R. (2016). Low-time complexity budget–deadline constrained workflow scheduling on heterogeneous resources. *Future Generation Computer Systems*, 55:29–40.

Avram, M.-G. (2014). Advantages and challenges of adopting cloud computing from an enterprise perspective. *Procedia Technology*, 12:529–534.

Bibi, S., Katsaros, D., and Bozanis, P. (2012). Business application acquisition: on-premise or SaaS-based solutions? *IEEE Software*, 29(3):86–93.

Bin, E., Biran, O., Boni, O., Hadad, E., Kolodner, E. K., Moatti, Y., and Lorenz, D. H. (2011). Guaranteeing high availability goals for virtual machine placement. In *2011 31st International Conference on Distributed Computing Systems (ICDCS)*, pages 700–709.

Boussoualim, N. and Aklouf, Y. (2014). An Approach based on user preferences for selecting SaaS product. In *2014 International Conference on Multimedia Computing and Systems (ICMCS)*, pages 1182–1188.

Butler, B. (2016). 10 must-watch IaaS cloud trends for 2017. Retrieved from http://bit.ly/2msHsKr. Accessed on: 2018-08-23.

Chaisiri, S., Kaewpuang, R., Lee, B.-S., and Niyato, D. (2011). Cost minimization for provisioning virtual servers in Amazon Elastic Compute Cloud. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), IEEE 19th International Symposium on*, pages 85–95.

Chaisiri, S., Lee, B.-S., and Niyato, D. (2009). Optimal virtual machine placement across multiple cloud providers. In *Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific*, pages 103–110.

Chaisiri, S., Lee, B.-S., and Niyato, D. (2012). Optimization of resource provisioning cost in cloud computing. *IEEE Transactions on Services Computing*, 5(2):164–177.

Chase, J., Kaewpuang, R., Yonggang, W., and Niyato, D. (2014). Joint virtual machine and bandwidth allocation in software defined network (SDN) and cloud computing environments. In *Communications (ICC), 2014 IEEE International Conference on*, pages 2969–2974.

Chase, J. and Niyato, D. (2017). Joint optimization of resource provisioning in cloud computing. *IEEE Transactions on Services Computing*, 10(3):396–409.

Elçi, O., Noyan, N., and Bülbül, K. (2018). Chance-constrained stochastic programming under variable reliability levels with an application to humanitarian relief network design. *Computers and Operations Research*, 96:91–107.

Erl, T., Puttini, R., and Mahmood, Z. (2013). *Cloud Computing: Concepts, Technology & Architecture*. Prentice Hall.

Godse, M. and Mulik, S. (2009). An approach for selecting Software-as-a-Service (SaaS) product. In *CLOUD'09. IEEE International Conference on Cloud Computing*, pages 155–158.

Günlük, O. and Pochet, Y. (2001). Mixing mixed-integer inequalities. *Mathematical Programming*, 90(3):429–457.

Iyoob, I., Zarifoglu, E., and Dieker, A. (2013). Cloud computing operations research. *Service Science*, 5(2):88–101.

Li, C., Liu, Y. C., and Yan, X. (2017). Optimization-based resource allocation for software as a service application in cloud computing. *Journal of Scheduling*, 20(1):103–113.

Louveaux, F. V. (1980). A solution method for multistage stochastic programs with recourse with application to an energy investment problem. *Operations Research*, 28(4):889–902.

Luedtke, J., Shabbir, A., and Nemhauser, G. (2010). An integer programming approach for linear programs with probabilistic constraints. *Mathematical Programming*, 122(2):247–272.

Maguluri, S. T., Srikant, R., and Ying, L. (2012). Stochastic models of load balancing and scheduling in cloud computing clusters. In *2012 Proceedings IEEE INFOCOM*, pages 702–710.

Matillion (2016). Why did Netflix migrate to the AWS Cloud? Retrieved from http://bit.ly/2wqA2vU. Accessed on: 2018-08-26.

Microsoft Corporation (2015). California DOJ: Microsoft Azure Government is compliant with FBI's CJIS standards for criminal justice info in the cloud. Retrieved from http://bit.ly/2BIXiet. Accessed on: 2018-08-26.

Nesmachnow, S., Iturriaga, S., and Dorronsoro, B. (2015). Efficient heuristics for profit optimization of virtual cloud brokers. *IEEE Computational Intelligence Magazine*, 10(1):33–43.

Pereira, M. V. and Pinto, L. M. (1991). Multi-stage stochastic optimization applied to energy planning. *Mathematical programming*, 52(1-3):359–375.

Philpott, A. and de Matos, V. (2012). Dynamic sampling algorithms for multi-stage stochastic programs with risk aversion. *European Journal of Operational Research*, 218(2):470 – 483.

Prakash, N. (2012). Did You Know Cloud Computing Has Been Around Since the '50s? Retrieved from https://goo.gl/jvJhQn. Accessed on: 2018-08-23.

Prékopa, A. (1995). *Stochastic Programming*. Kluwer Academic, Dordrecht, Boston.

Prékopa, A. (2003). *Handbooks in Operations Research and Management Science, vol.10*, chapter Probabilistic Programming, pages 267–351. Elsevier, Amsterdam. Editors: A. Ruszczyński and A. Shapiro.

Rockafellar, R. and Uryasev, S. (2000). Optimization of conditional value-at-risk. *The Journal of Risk*, 2(3):21–41.

Shapiro, A., Dentcheva, D., and Ruszczyński, A. (2009). *Lectures on stochastic programming: modeling and theory*. Society for Industrial and Applied Mathematics, Philadelphia, USA.

Shapiro, A., Tekaya, W., da Costa, J. P., and Soares, M. P. (2013). Risk neutral and risk averse stochastic dual dynamic programming method. *European Journal of Operational Research*, 224(2):375–391.

Shen, S. and Wang, J. (2014). Stochastic modeling and approaches for managing energy footprints in cloud computing service. *Service Science*, 6(1):15–33.

Shi, J., Luo, J., Dong, F., Zhang, J., and Zhang, J. (2016). Elastic resource provisioning for scientific workflow scheduling in cloud under budget and deadline constraints. *Cluster Computing*, 19(1):167–182.

Subramanian, T. and Savarimuthu, N. (2016). Application based brokering algorithm for optimal resource provisioning in multiple heterogeneous clouds. *Vietnam Journal of Computer Science*, 3(1):57–70.

Tang, Z., Qi, L., Cheng, Z., Li, K., Khan, S. U., and Li, K. (2016). An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment. *Journal of Grid Computing*, 14(1):55–74.

Wang, W., Niu, D., Liang, B., and Li, B. (2015). Dynamic cloud instance acquisition via IaaS cloud brokerage. *IEEE Transactions on Parallel and Distributed Systems*, 26(6):1580–1593.

Zhang, M., Küçükyavuz, S., and Goel, S. (2014). A branch-and-cut method for dynamic decision making under joint chance constraints. *Management Science*, 60(5):1317–1333.

Zolman, T. (2014). Understanding the New AWS Reserved Instance Model. Retrieved from http://bit.ly/2O2gk1j. Accessed on: 2018-08-23.