Modeling Multi-stage Decision Making under Incomplete and Uncertain Information

Viktor Bindewald[†,*], Fabian Dunke[*], Stefan Nickel[*]

June 12, 2020

We propose a new universal framework for multi-stage decision making under limited information availability. It is developed as part of a larger research project (German Research Foundation (DFG) [2020]) which aims at providing analytical methods to compare and evaluate different models and algorithms for multi-stage decision making. In our setting, we have an open time horizon and limited information about upcoming stages in different granularity (e.g., deterministic information about the near future and/or forecast-based information about the more distant future). The framework is designed to reflect the need of decoupling uncertainty models from algorithms (Bakker et al. [2019]) and to integrate standard disciplines, such as online optimization, stochastic programming and robust optimization, within a unified setting. Hence, the design of the framework provides the modeler with full flexibility to configure data availability, uncertainty representation, and solution methodology according to the problem under consideration. To the best of the authors' knowledge, such a setting was not considered so far. The modeling capabilities of the framework are illustrated through a broad range of examples. In order to assess the quality of solution algorithms, the framework is connected to an evaluation module. The overall architecture consisting of the modeling framework, solution methodology, and evaluation module represents a comprehensive tool for comparing and relating different algorithmic strategies and different uncertainty models in multi-stage decision making problems.

[*]Karlsruhe Institute of Technology, Institute for Operations Research, Discrete Optimization and Logistics, Kaiserstr. 12, 76131 Karlsruhe, Germany, `https://dol.ior.kit.edu`;
e-mail: {viktor.bindewald, fabian.dunke, stefan.nickel}@kit.edu; † Corresponding author

Contents

1 Introduction

The need for sequential decision making arises in many real-life applications where information becomes known gradually over time and decisions are required intermit-

tently in order to proceed to the next system state. In lot sizing, for example, today's production decisions block storage capacity, thus influencing the future scope for decision making. However, decision makers often do not have all information concerning the future. This means that the decision maker has to react to scheduled or random events, but at the same time cannot fully assess all consequences of the decisions made. Subsequently, we outline the process of dynamic decision making in more detail along with its intrinsic difficulties arising from the necessity of decision making under incomplete information on the future.

1.1 Problem Statement

In a dynamic decision making process, required decisions are made based on information about the current and future context of the decision process available at decision time (e.g., demands, prices, travel distances). Data about the future can be obtained in many ways. For instance, data collection may be based on information received from installed sensors, or data may be predicted by forecasting methods including possible future scenarios derived from historical data or expert knowledge. In the first case, we acquire deterministic data which usually is present only for the near future. In the second case, we obtain uncertain data which comes with probabilistic information and can be available also for the more distant future. Nonetheless, in general not all information concerning future events is available to the decision maker. Hence, the decision making process typically operates under incomplete and uncertain data.

The challenge then is to make decisions on an ongoing basis such that their results support and contribute to the overall goal of the decision maker. To tackle this problem with its inherent uncertain character, the decision maker should compare different models for the underlying decision making task, data and uncertainty representations, and solution algorithms. The framework developed in this research project is designed to provide the necessary flexibility for this broad class of decision making problems such that decision makers can deploy various solution strategies and compare their respective outcomes. Since decisions are required recurrently, we will speak of a decision process which can be subdivided into decision stages (cf. Definition 1). In practice, decision stages are induced either naturally whenever data is provided periodically (period-driven decision process) or through the occurrence of an event causing the release of new information (event-driven decision process).

As an illustrative example, consider the following setting: In order to determine a production plan, a decision maker has to specify lot sizes which minimize the overall costs while satisfying the customer demands. Traditionally this task is modeled as a lot sizing problem where complete information about costs and demands for the entire planning horizon is assumed. However, this is a rather artificial situation since in practice full omniscience with respect to the upcoming orders and demands occurs only for very short planning horizons. More realistically, one can consider an open time horizon where demands for the near future are revealed to the decision maker

periodically. The demand information may be available with different degrees of uncertainty. Demands for the imminent future, e.g., the next days, usually represent certain data. Information about the demands in the more distant future is based on preorders and sometimes enriched by forecasts using historical information or expert knowledge and hence is naturally subject to uncertainty. Consequently, the goal is to balance production, storage, setup and penalty costs without knowing all forthcoming demands exactly. Depending on the available information and risk attitude, the decision maker may aim to find production plans in favor of the average or the worst case employing techniques from stochastic and robust optimization, respectively. However, as these techniques are only applied to the information at hand, the overall decision making process about lot sizes takes place in an overarching rolling horizon scheme as an online decision process. The proposed modeling framework and solution methods allow to integrate different mathematical optimization approaches into such an online decision process.

1.2 Research Gaps

Decision problems as described above are typically modeled as multi-stage optimization problems with uncertain parameters. Well-known solution approaches comprise online optimization, stochastic programming or robust optimization. However, as shown in Dunke et al. [2016], these methods model and use information about the future in fundamentally different ways. For instance, online optimization with lookahead resolves uncertainty by providing small snippets of deterministic information about the near future at each stage, whereas stochastic programming uses probabilistic information about parameter realizations concerning the entire time horizon. Additionally, multi-stage approaches in robust optimization and stochastic programming are often restricted to two stages and suffer from excessive computational effort for more than two stages.

At first glance the approaches mentioned above look very different and are considered separately by most researchers (see Section 2 for a literature overview). However, motivated by similar procedures concerning data retrieval and related decision making as observed in numerous real-life applications, there is a need for unifying these disciplines. Such a unified framework allows for an integrated perspective on multi-stage optimization under uncertainty coalescing different mathematical approaches to optimization under uncertainty.

1.3 In a Nutshell

The proposed framework models decision processes where several interrelated decisions have to be made over an open time horizon while information is available only for a small portion of the time horizon. Information about the uncertain future can be represented depending on the considered decision problem and available data as encountered in a specific problem setting. From a methodological point of view, the

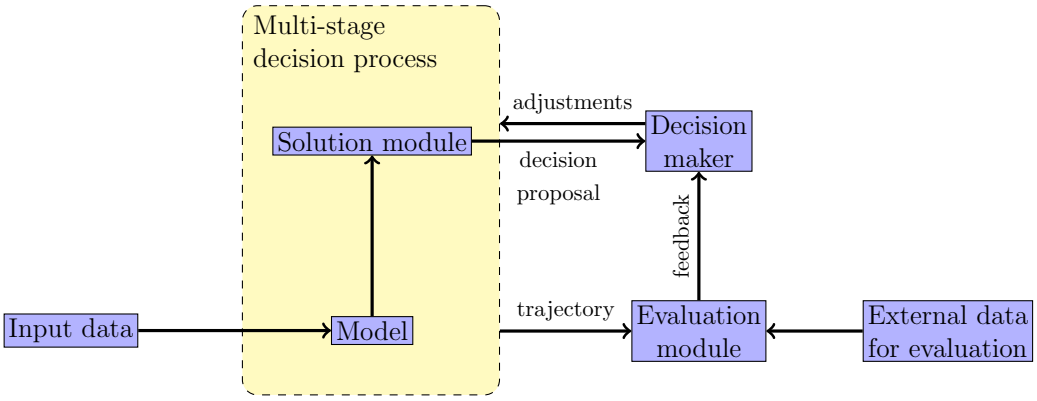

Figure 1: Relations between model components.

availability of incomplete information at each decision step implies that no solution methodology can be optimal, i.e., no natural concept of optimality exists. Rather, as the system state depends on previous decisions and current data, solutions as well as policies for the entire time horizon cannot be precomputed upfront, but have to be computed in an overarching online decision process. The framework facilitates the integration of techniques from online optimization, stochastic programming and robust optimization into this online decision process. Hence, the framework gives the modeler the possibility to adaptively employ a solution method and uncertainty model best suited to the current situation. With the evaluation module, the decision maker is supplied with a methodological tool that can be used to assess the quality of algorithms. In particular, the evaluation module can be used already throughout the solution process in order to make present decisions dependent on past decisions and current data projections. Figure 1 illustrates the architecture of the overall framework.

As part of the research project "Sequential decision making under system inherent uncertainty" (German Research Foundation (DFG) [2020]), the modeling framework, the integration of different solution methods and the evaluation module represent the basis for further application-oriented steps which comprise the formation of an information pool for basic problems, as well as the application to real-world problem settings in the context of simulation models.

1.4 Contribution and Outline

The contribution of this paper is twofold. First, we introduce a new flexible modeling framework for multi-stage decision making that provides multiple possibilities to model uncertainty in problem parameters and covers a large class of real-life problems. It unifies different approaches to optimization under uncertainty that include, among others, online optimization, robust optimization and stochastic programming. Second, this framework is designed to enable its users to study time-dynamic decision

making processes in a holistic way. Therefore we present custom-tailored methods to analyze and evaluate different models, uncertainty representations, solution algorithms and combinations thereof. Furthermore these tools facilitate measuring the value of additional information, overall performance, and solution method sensitivity with respect to changing parameters.

The document is organized as follows. In Section 2 we give an overview of related publications. It covers modeling frameworks, specific approaches to optimization under uncertainty, rolling horizon decision making as well as performance measures. Section 3 introduces our new framework for time-dynamic optimization problems in its full generality and describes the solution process in the rolling horizon setting in detail. In Section 4 we exemplify our framework using eight decision making problems from different fields of application. Section 5 deals with performance evaluations methods adapted to our setting and provides a methodology for a sensitivity analysis of algorithms applied in the framework. Section 6 concludes by presenting possible subsequent lines of research.

2 Related Work

According to the architecture of the overall framework in Figure 1, we discuss existing research literature concerning the overall decision process, specific solution methods, and performance measurement for algorithms. The review also indicates which parts of the available research differ from the approach taken in this research project.

2.1 General Frameworks

Two types of general frameworks can be distinguished: First, mathematical frameworks explicitly aim at providing a formal environment for solving multi-stage decision making problems by mathematical methods. Secondly, multi-stage decision making problems can also be related to event-driven models emphasizing the role of events as triggers for the advancement of the overarching online decision process.

Mathematical Approaches

The most intuitive framework for modeling multi-stage decision making problems is given by Markov decision processes (Puterman [2005]) where decision makers carry out actions in order to advance from system state to system state via a probabilistic state transition. However, the problem is modeled over a-priori information about transition probabilities and expects fully specified definitions of a state space and an action space. The model further assumes a Markov structure for the transitions from one state to another in such a way that transition probabilities depend on the current state and the selected action only. Problems formulated as Markov decision processes can be solved using the dynamic programming paradigm (Bellman [1957]).

Reinforcement learning (Sutton and Barto [2018]) can be seen as a sampling-based solution methodology for Markov decision processes and has experienced increasing popularity as it represents a model-free approach in terms of not requiring any probabilistic information.

A multi-stage framework for decision problems in general and stochastic optimization problems in particular is presented in Pflug and Pichler [2012]. The constituting elements of a stochastic optimization problem are identified as a stage structure, an information process (stochastic process), a filtration, a value process, the decision space, non-anticipativity, and a loss function. Moreover, the authors suggest that classical dynamic programming approaches differ from the presented multi-stage framework by a stronger emphasis of the notion of a state which evolves as a controlled Markov process. Contrarily, in the framework proposed by the authors, randomness is represented through the information process which is described as a stochastic process. In contrast to our framework, the setting is not considered within an online decision process, i.e., a solution to a problem is determined only once and then contains solution information for the entire planning horizon.

Without focusing on multi-stage settings, Klamroth et al. [2017] introduce three unifying approaches on optimization under uncertainty (vector optimization, set-valued optimization, and nonlinear scalarization) which allow to interpret solutions of uncertain optimization problems from different angles. These different perspectives represent the basis for devising optimization algorithms based on available methods for vector optimization, set-valued optimization, and nonlinear scalarization, respectively. Moreover, it is shown that different concepts from robust optimization (including adjustable robustness) and stochastic programming notions (including two-stage problems and stochastic dominance) can be characterized using the presented unifying approaches.

A framework for online stochastic combinatorial optimization is suggested in Van Hentenryck and Bent [2006]. It follows an approach rather similar to our setting by integrating stochastic programming into an online algorithm to develop so-called online anticipatory algorithms. The authors assume that if the distribution of the input sequence is known, then an online algorithm may use the probabilistic information at each stage. In particular, the algorithm may sample future developments from this known distribution in order to derive a partial decision. The basic form of these anticipatory algorithms (online expectation) amounts to evaluating all possible decisions with respect to all possible or sampled scenarios. Technically, at each stage the algorithm solves a multi-stage stochastic program, but only implements the first stage decisions. More recently, Van Hentenryck et al. [2010] propose a number of approximation algorithms, each one following a specific "local" idea, e.g., minimizing the regret in terms of immediate loss accrued through a decision.

The problem class of multi-stage stochastic and adaptive optimization is considered in Bertsimas and Caramanis [2010] under the concept of finite adaptability. This concept allows the decision maker to compute for each stage a small set of possible partial solutions which are called contingency plans. Each plan corresponds to a

specific subset of all uncertain parameters and yields a prescription of how to react when a certain parameter realization occurs. Hence, a partition of the parameter set is created to derive dynamic policies which are neither as extreme as entirely static models nor as fully adaptive models. Bertsimas et al. [2011b] then shows that finite adaptability is sufficient to lead to good approximations of the fully adaptive solution. In contrast to our setting, finite adaptability is only considered in an offline perspective with fixed horizon and known possible data parameters.

Despite its name suggesting a focused discussion of online optimization methods, the collection Grötschel et al. [2001a] covers a rather wide array of topics related to online decision making under incomplete information. Discussed topics comprise, among others, stochastic optimization, mathematical programming, and moving horizon methods. Hence, several settings connected to multi-stage decision making are presented and classified, but not further unified.

A framework for data-driven optimization in a multi-stage setting is presented in Bertsimas and McCord [2019]. The idea of the approach consists of extending a stochastic dynamic optimization problem by integrating additional external data (auxiliary covariates) into the model in order to make it more precise and applicable to real world situations. The authors suggest solution methods derived from the sample average approximation method and provide results on asymptotic optimality of this numerical outline. Numerical results are presented for multi-stage inventory and lot sizing models.

A declarative framework for online optimization problems is presented in Ek et al. [2020]. The approach makes use of an offline formulation of the problem and allows for annotations to the offline model in order to automatically generate an online model which can then be used in an online process via a so-called sliding window which is similar to the idea of a rolling horizon. In contrast to our framework, the presented outline is strongly based on the capability of a software tool to provide the mentioned annotation mechanism. Applications in job shop scheduling and cargo assembly planning illustrate the handiness of the approach in terms of obtaining an online algorithm in an automated fashion.

The most recent attempt to present a general framework for optimization under uncertainty can be found in Powell [2018]. The author finds a common language for fifteen scientific communities ranging from optimal stopping to approximate dynamic programming. However the assumption therein is different to our setting: all information, deterministic and uncertain alike, is available upfront, i.e., no information disclosure occurs over time. A latest review on uncertainty representations and solution approaches in multi-stage optimization under uncertainty can be found in Bakker et al. [2019]. The authors provide numerous applications for each concept and conclude a need to decouple (uncertainty) models and algorithms.

Event-driven Approaches

From a control theory point of view, the proposed framework exhibits similarities to discrete event systems (Cassandras and Lafortune [2010]). Due to the analogy between event occurrences and information disclosure at discrete time instants, multi-stage optimization under uncertainty can be cast as a discrete event system with the additional element of intentionally controlled actions. A summarizing article of discrete event systems and control mechanisms is given by Lafortune [2019]. Gosavi [2015] addresses the problem of determining actions in order to steer a discrete event system through a favorable state trajectory under the notion of control optimization. To this end, methods of dynamic stochastic programming, reinforcement learning, and stochastic search are suggested. Moreover, a connection is established to discrete event systems underlying stochastic influences which makes the entire setting suitable to be implemented in discrete event simulation models. In contrast to the mathematical optimization community, discrete event system modelers do not focus on decision making, but rather on a formal representation of the controlled discrete event system itself.

A general framework for rolling horizon decision making with a possibility to integrate forecasting is presented in Sethi and Sorger [1991]. The paper also contains references to related surveys and emphasizes that the application of production planning has largely contributed to the popularity of the approach. Therefore, a generic dynamic programming formulation is devised which is then instantiated for production planning. Bes and Sethi [1988] can be seen as a preparatory work introducing formal definitions of forecast and decision horizons which are then used in the theory of rolling horizon decision making. A further focal point of applications is found in dynamic network and routing problems. Powell et al. [1995] surveys a taxonomy of dynamic models including rolling horizon models. As an extension to common rolling horizon decision making in dynamic vehicle routing (Psaraftis [1988]), a double horizon consisting of a short-term and a long-term horizon is considered in Mitrović-Minić et al. [2004]. The additional consideration of a long-term horizon (preserving flexibility for additional requests) allows for significant improvements as compared to the case with a short-term horizon (ensuring short travel distances for upcoming requests) only. Rolling horizon methods are further employed in a multitude of contexts, e.g., lot sizing (Brandimarte [2006]) or portfolio optimization (Bertsimas and Pachamanova [2008]). Moreover, de Ruiter et al. [2016] connects the rolling horizon setting to adjustable robust optimization and finds that in case of multiple solutions the rolling horizon approach helps in eliminating differences between average behavior of different robust solutions. The collection of Fleischmann et al. [2015] on advanced planning systems illustrates how rolling horizons are used nowadays as regular parts of computerized advanced planning systems supporting supply chain planning and execution tasks.

2.2 Solution Methods

Stochastic programming and robust optimization represent the two most prominent mathematical approaches to solving optimization problems under uncertainty. We discuss these disciplines in the light of the multi-stage setting. Moreover, we present online optimization as another discipline for multi-stage problems which differs from the previously mentioned disciplines in its basic assumption that no information about the future is available.

Stochastic Programming

The multi-stage setting of stochastic programming differs from our approach in its closed horizon. Nonetheless, in the form of subroutines the approaches can be utilized within our framework which exhibits an open and moving horizon (see Section 3.4 for details). Birge and Louveaux [2011] and Kall and Mayer [2011] provide an introduction to stochastic programming and also discuss the multi-stage setting – both from the modeling and the solution methodology perspective (exact and approximation/sampling methods). In addition, Shapiro et al. [2014] introduces several alternative risk measures beyond the expected objective value for the goal setting to be used in stochastic programming. To the best of the authors' knowledge, with Birge [1997] there is only one survey paper available on stochastic programming which offers a more comprehensive perspective including a discussion of computational issues and applications.

Robust Optimization

A foundational monograph on robust optimization is given by Ben-Tal et al. [2009]. For the multi-stage setting the authors propose two different notions. First, in the form of adjustable robust optimization, a concept allowing to react upon uncertainty realizations rather than to stick to a best worst-case solution computed upfront. The second suggestion are robust Markov decision processes which expose decision makers to stochastic influences on state transitions. A comprehensive survey on robust optimization can be found in Bertsimas et al. [2011a]. Besides different uncertainty set representations and tractability considerations, the multi-stage setting and its requirement of adaptability is discussed. Further references are provided to related concepts such as receding horizons or dynamic programming. Iyengar [2005] and Nilim and El Ghaoui [2005] extend dynamic programming with robustness in the objective through uncertain state transition probabilities and uncertain objective coefficients. With respect to algorithmic approaches, the survey Goerigk and Schöbel [2016] presents an overview on different robustness concepts, uncertainty sets and solution methods. Recoverable robustness and its extension to the multi-stage setting is discussed in a series of papers in Ahuja et al. [2009]. Online algorithms are shown to be useful instruments in providing recovering capabilities once deviations from robust plans have to be handled. For a comprehensive overview on adjustable

robust optimization we refer to the survey Yanıkoğlu et al. [2018]. Likewise, Bertsimas and Thiele [2006] yields an overview both on static and dynamic decision making in robust optimization. Practical guidance for dealing with robust optimization, also in the multi-stage setting, is provided in Gorissen et al. [2015]. Several approaches leveraging clustering algorithms and neural networks have been used to enable the data-driven construction of uncertainty sets, one approach using support vector clustering is presented in Shang et al. [2017].

Online Optimization

Online optimization problems are characterized in Grötschel et al. [2001b] as problems where decisions have to be made before all data of the problem are known. The authors introduce a distinction between a sequential model (data released in an ordered sequence) and a real time model (data released at specific release dates). Additionally, real time issues are discussed which impact practical implementations, and a connection is established to discrete event simulation as a helpful analytic tool in real world settings. Further, in Fiat and Woeginger [1998] a collection of papers on online optimization capturing the state of the art at the end of the 1990s is presented. The articles both address general topics of online optimization (such as limitations of performance measurement techniques, see Section 2.3) as well as application-specific results on a large number of well-known online optimization problems ranging from paging to routing to search and navigation. For a more recent and more introductory paper we refer the reader to Jaillet and Wagner [2010].

Online optimization with lookahead (Dunke and Nickel [2016]) is an extension of the pure online setting, moving it slightly closer to the offline case. Lookahead describes a limited amount of information about input elements in the near future. Hence, these data elements are made available earlier in comparison to the pure online setting and enhance the decision maker's informational level at each point in time. Lookahead information can be made available by tracking and tracing or other sensor data (Ghiani et al. [2004], Schuh et al. [2011]). We emphasize that in contrast to uncertain information used in stochastic programming and robust optimization, lookahead refers to deterministic information about the near future. Most recently, Dunke and Nickel [2019] introduced the notion of gradual lookahead as a generalization applicable to uncertainty. Gradual lookahead allows for an uncertain outlook on future data which becomes more and more precise as time advances.

2.3 Performance Measurement

The idea of having a value or a price of information is found in many optimization contexts. In stochastic programming, the value of the stochastic solution gives the improvement in the objective when using stochastic information as opposed to expectated parameter values (Birge and Louveaux [2011]). Stochastic programming related measures like value of stochastic solution, expected value of perfect information are ex-

tended to the multi-stage setting in Escudero et al. [2007]. In robust optimization, the price of incorporating uncertainty, the so-called price of robustness, has been defined several times: In Bertsimas and Sim [2004], the price of robustness is the trade-off between feasibility violation probability and impact on the objective value. In Ben-Tal et al. [2005], it describes the difference in the objective when an approximate adjustable robust optimization approach is carried out as opposed to an algorithm operating under perfect hindsight with all data known at the outset. Competitive analysis – a widely recognized worst-case performance measure for online algorithms – is extensively examined in Borodin and El-Yaniv [2005]. It considers the worst case in two fashions: Quality guarantees have to hold for all instances, and quality is measured against a hypothesized omniscient and optimal offline algorithm. As a consequence of different points of criticism (e.g., overly pessimistic results, neglect of overall algorithm performance), numerous enhancements of competitive analysis and alternatives were devised (e.g., Koutsoupias and Papadimitriou [2000], Ben-David and Borodin [1994], Dorrigiv et al. [2009], Hiller and Vredeveld [2012], Boyar et al. [2009]). A similar, but less strict metric origins from Mitrović-Minić et al. [2004]. The authors evaluate the performance of an algorithm by comparing the objective values of the same algorithm when applied to solving both the online and the offline instance. Dunke and Nickel [2016] introduces counting distribution functions for the objective value and for performance ratios in order to assess algorithm performance more comprehensively. Counting distribution functions lend themselves to a sampling-based analysis in the case of a large number of possible input instances as encountered in multi-stage problems under uncertainty. Sampling approaches are an agreed form of analysis in stochastic programming (Birge and Louveaux [2011]) and have also been transferred to other related fields such as distributionally robust optimization (Bertsimas et al. [2018]). In Powell et al. [1995], measures for evaluating outcomes of rolling horizon procedures are presented in terms of average sample costs and error bounds which are calculated against the case where all information would have been available.

3 The Modeling Framework and its Peculiarities

In this section we provide the formal definition of our framework. Since our goal is to model time-dynamic decision making, we first explain when decisions are made and how information is disclosed.

3.1 Preliminaries

In our framework we assume that whenever some new information is available, the decision maker has to react by making and implementing a decision. To formally describe our model for time-dynamic decision making, we will use the term *stage* which we define next.

Definition 1 (stage)**.** *A stage is a period of time where a decision has to be made. Two consequent stages are divided by the disclosure of new information. Revelation of new information initiates a new stage.*[1]

Depending on the nature of the information disclosure process, we distinguish between two types of decision processes.

In period-driven decision processes, information is released periodically. In lot sizing, for instance, customer demands are assumed to be specified on a daily, weekly, or monthly basis (see Section 4.1.1 for modeling details). Thus, the time horizon is divided into stages upfront and this division is defined by the modeler. Usually the stages correspond to work days or weeks and hence are of equal length. At the beginning of every stage new information is revealed. Hence, from a modeling point of view only the ordering of the stages is important because it determines which information is available to the decision maker at which point in time. Because of this, we refer to the stages by an index $t \in \mathcal{T} := \{1, \dots, T\}$ with $T \in \mathbb{N} \cup \{\infty\}$ and neglect the exact temporal structure. Note that we explicitly allow an open time horizon.

In event-driven decision processes, the occurrence of an external event leads to the release of new information. In contrast to the period-driven case, the time instants where new information appears are not known in advance but depend on an external random process. A typical example for this case is paging in computer memory management where user requests cannot be seen in advance (see Section 4.2.1 for modeling details). From a modeling perspective, we assume that the decision maker reacts to events that occur at arbitrary points in time and disclose new information. Therefore, every event triggers a new stage. As before, only the ordering of the events or equivalently of the induced stages is of importance. This allows us to use t as the stage index also for event-driven decision processes.

3.2 Formal Definition

Before presenting the formal description of our framework we want to specify our design goals. The framework has to be Markovian, i.e., the current decision depends on the current system state only. This simplifies the subsequent modeling but does not impose any limitations on the modeling capabilities of the framework [Powell, 2014, p. 11]. The next requirement is that the modeler has to be able to choose different uncertainty representations for different data, e.g., production costs and customer demands. Furthermore, we request the possibility to switch the representation for the future information over time. For instance, it may be reasonable to start with a deterministic lookahead amenable to online optimization and after some time to progress to scenarios with probabilities amenable to stochastic programming.

[1] This is the implicit definition from Birge and Louveaux [2011] (cf. p. 58). A different notion is used in [King and Wallace, 2012, p. 6]: "A stage is a point in time where decisions are made within a model."

We are now ready to define a multi-stage decision process formally. The overall goal of the decision maker is captured in the notion of a decision criterion. We use the term decision criterion and not objective function intentionally, since it is not possible to evaluate the decision criterion because the required information is released over the (possibly open) time horizon gradually. The decision criterion represents the function to be optimized over the entire planning horizon through implementing stage-wise decisions as beneficial as possible for the overall outcome. For the sake of clarity we will describe minimization processes only. For maximization processes the definition can be adjusted by exchanging costs for rewards. After the definition we will explain the single elements of a multi-stage decision process in more details.

Definition 2 (Multi-stage decision process)**.** *A multi-stage decision process $\mathcal{P}$ with time horizon $T \in \mathbb{N} \cup \{\infty\}$ is specified by six components.*

1. *Initial input data I_0*
2. *Time-dynamic input data I_t, $t \in \mathcal{T} = \{1, \ldots, T\}$*
3. *Data about the future $\mathbb{F}_t$*
   a) *$\mathbb{L}_t$: deterministic lookahead for the next $\tau_t \in \mathbb{N}_0$ stages*
   b) *$\mathbb{U}_t$: uncertain data for the subsequent $\sigma_t \in \mathbb{N}_0$ stages*
4. *Stage task at stage $t \in \mathcal{T}$*
   a) *$\texttt{paradigm}_t$: prescribes how $\mathbb{L}_t$ and $\mathbb{U}_t$ must be considered in order to make a decision, e.g., online with lookahead*
   b) *decision x_t that must be made*
   c) *cost-per-stage: $c_t(x_t) := c_t(I_0, I_t, x_t)$ is immediately incurred after implementing a decision x_t*
5. *Stage transition information*
   - `event` *that triggers the transition (only in the event-driven case[2])*
   - *set of rules specifying the transition $I_t \to I_{t+1}$*
6. *Decision criterion*

The set $\mathbb{F}_t$ contains deterministic and uncertain information about the upcoming $\tau_t + \sigma_t$ stages and is referred to as the *information horizon* at stage t. The transition from a stage t to its successor is illustrated in Figure 2. The new input data I_{t+1} implicitly reflects that all previously made decisions $x_{t'}$ for $t' < t$ are also considered in I_{t+1}. In Section 3.4, we will present the transition in more details after having explained how decision x_t in 4b) can be obtained in the solution process. The framework

[2]In the period-driven case, we can consider "lapse of a time unit" as the triggering event. But this event is rather artificial and does not provide proper information.

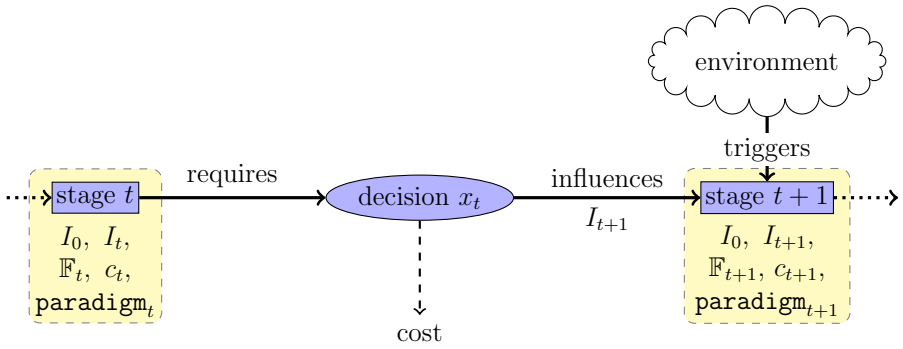

Figure 2: Illustration of the simplified transition process from stage t to stage $t+1$.

defines a closed-loop control system because the decision x_t depends on the feedback from the environment. The latter suggests that its value can be computed upfront for the entire time horizon which is impossible due to absence of information about all upcoming stages at any point in time. The decision criterion is usually evaluated for the past stages, e.g., via $C_t := \sum_{s=1}^{t} c_s(x_s)$.

For concrete modeling examples from different fields of application, please consult Section 4.

Remark 3. *There are many different terms in the literature for identical or similar objects. The data I_0 and I_t describe the resources that are managed by the decision maker and (in combination) evolve over time. Often, I_0 and I_t are summarized under the term state (e.g., in literature on Markov Decision Processes (MDP)). Data $\mathbb{F}_t$ is often called exogenous information (e.g., in Powell [2018]). Note that data I_t contains both endogenous and exogenous information.*

After having introduced the formal framework, we briefly discuss its inherent underlying assumptions. First, we neglect the time needed to compute the stage decision x_t. This means we assume that the selected algorithm is fast enough to determine a solution before a new stage commences. Second, we assume that information is disclosed at a stroke and the decision making process can be started immediately thereafter. Finally, once implemented a decision cannot be corrected later.

Despite its generality and flexibility, several extensions of the framework are possible and we will present two of them here: In many applications it makes sense to allow accumulation, i.e., the decision maker does not have to respond to each disclosed information immediately, instead it is possible to wait and gather information. This option can be integrated into the framework by formally allowing in Item 4 of Definition 2 the decision *"do nothing"* which carries a penalty. A second extension can be that the uncertainty depends on the decision x_t which is referred to as endogenous uncertainty in the literature.

3.3 Multi-stage Process Instances

In this section we formalize the term *instance* in the context of our open-horizon modeling framework. Recall that an instance is the collection of all parameter values needed to describe a concrete version of a decision or optimization problem (Garey and Johnson [1979]). We will emphasize the peculiarities resulting from the fact that the decision making processes we are interested in have inherent online characteristics. First, we start with the most basic definition.

Definition 4 (Multi-stage process instance)**.** *A multi-stage process instance is a collection of concrete values for all parameters of a multi-stage process and can be described as a tuple* $(T, I_0, D_{\leq T}, c_{\leq T})$*, where*

- T *is the time horizon,*
- I_0 *is the initial input data (see Item 1 in Def. 2),*
- $D_{\leq T} := (D_1, D_2, \ldots, D_T)$ *where* $D_t := (I_t, \mathbb{F}_t, \texttt{paradigm}_t)$ *is the time-dynamic data available in stage* $t \in \mathcal{T}$*, and*
- $c_{\leq T} = (c_1, \ldots, c_T)$ *is the cost-per-stage vector.*

We denote a set of instances of a multi-stage process by $\mathbb{I}$. Depending on the context, $\mathbb{I}$ can refer to all instances of the process under consideration or to a set of considered instances, e.g., as part of a sampling-oriented analysis. Because of their temporal structure, we refer to $D_{\leq T}$ and $c_{\leq T}$ as data and cost *trajectories* respectively.

Definition 5 (Solution, solution set)**.** *For a given instance* $\mathcal{I} = (T, I_0, D_{\leq T}, c_{\leq T})$ *of a multi-stage process, a feasible solution to* $\mathcal{I}$ *is a sequence of decisions* $x = (x_1, \ldots, x_T)$ *for every stage* $t \in \mathcal{T}$*. For a given instance* $\mathcal{I}$ *we define by* $\mathrm{Sol}(\mathcal{I})$ *the set of all feasible solutions.*

Details on the solution process and algorithms will be presented in Section 3.4.

As a special type of a multi-stage process instance, we define the so-called *underlying instance* which is free of uncertainty and any information about the future. The following definition also implies that the stage data I_t is the most important element distinguishing instances from each other apart from modeling choices with respect to uncertainty representation and solution methodology. Thus, lacking any information about the future stages, such an instance is only amenable to pure online optimization approaches.

Definition 6 (Underlying multi-stage process instance)**.** *The underlying multi-stage process instance of a multi-stage process is the future-infomation-free collection of concrete values for the parameters of the multi-stage process and can be described as a tuple* $(T, I_0, D_{\leq T}, c_{\leq T})$ *with* $D_{\leq T} = (D_1, \ldots, D_T)$ *where* $D_t = (I_t, \emptyset, \emptyset)$ *for* $t \in \mathcal{T}$*.*

In order to generate multi-stage process instances for computational testing and evaluation (see Section 5), we suggest to impose the application of modification mechanisms upon a set of underlying multi-stage process instances $\mathbb{I}$. In this way, different data trajectories can be created systematically and compared later on. As an example for a modification mechanism one can think of first examining the data points recorded in $D_{\leq T}$ or more precisely in I_t, $t \leq T$, of each underlying multi-stage process instance from $\mathbb{I}$. Then, in a second step, information in $\mathbb{F}_t$ can be created based on data collected in the first step – according to the specifics prescribed in requested $\texttt{paradigm}_t$ – through providing additional, uncertainty-related information such as uncertainty sets or probabilistic information. In this way, $(I_t, \mathbb{F}_t)$ from the multi-stage process instance can be interpreted as an extension of I_t from the corresponding underlying multi-stage process instance.

3.4 Solution Process

In this section, we want to explain the solution process for a given multi-stage process instance $\mathcal{I}$ in more details. In each stage $t \in \mathcal{T}$, a decision x_t is required in order to allow the system to proceed to the next stage (see Item 4b in Def. 2). At the same time, cost-per-stage $c_t(x_t)$ is incurred as a result of this decision which contributes to the overall costs. To this end, in compliance with $\texttt{paradigm}_t$ a decision making method has to be invoked in order to obtain x_t. This is done in two steps. First, we formulate a stage-related optimization problem based on the currently available data $I_0, I_t, \mathbb{F}_t$ and $\texttt{paradigm}_t$ which we call *snapshot problem* at stage t. Second, a computational method from rational decision making which solves the snapshot problem is selected. We present a formal definition for the snapshot problem next.

Definition 7 (Snapshot problem)**.** *A snapshot problem P_t at stage t is an optimization problem that can be formulated using data $I_0, I_t, \mathbb{F}_t$ and $\texttt{paradigm}_t$ and contains in its output the required stage decision x_t.*

Observe that for a given decision making process modeled as a multi-stage process according to Def. 2, the modeler does not have to specify the snapshot problem P_t or even an objective function for P_t. In fact, the specification of a snapshot problem represents an additional degree of freedom for the modeler which is related to the concrete solution process. In general, there exist several different snapshot problems varying for example in the choice of objectives functions. Irrespective of the choice, the snapshot problem shall ensure that it contributes to the overall decision criterion. Frequently used snapshot problems are mixed integer programming formulations which consider both the constraints and the decision criterion objective function for the available data. Likewise, a snapshot problem could only ask for selecting x_t according to some decision rule. For many multi-stage processes, the objective function for P_t and the decision criterion may be very different in nature (see e.g., [Mitrović-Minić et al., 2004, p. 6]).

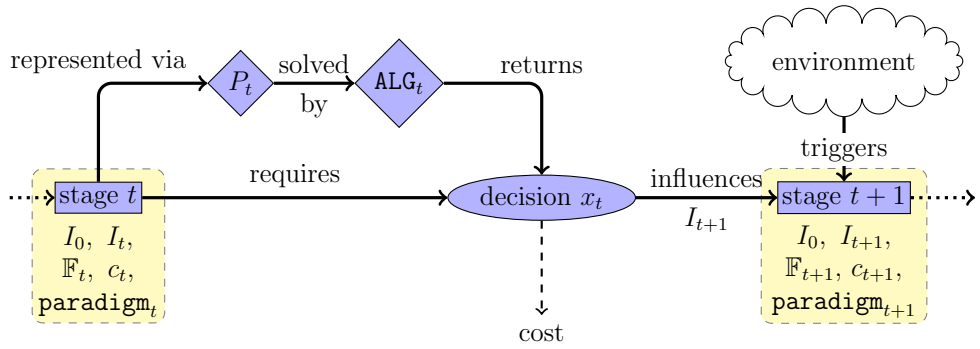

Figure 3: Illustration of the solution process from stage t to stage $t+1$.

Next, we want to formalize the term *algorithm* in order to talk about solving a given snapshot problem P_t.

Definition 8 (Snapshot algorithm)**.** *A snapshot algorithm* $\texttt{ALG}_t$ *is a solution method for a snapshot problem* P_t *which takes the data* $I_0, I_t, \mathbb{F}_t, \texttt{paradigm}_t$ *as input and computes an output for* P_t *which provides the required stage decision* x_t*.*

The entire solution procedure as part of the transition process is depicted in Figure 3.

Remark 9. *In addition to a current-stage decision* x_t *in Step 4, a snapshot algorithm* $\texttt{ALG}_t$ *can also return policies* x_{t+1}, x_{t+2} *etc. that may be implemented in the future. To keep the framework's description compact, we do not specify this explicitly because such functionality can be obtained by a suitable algorithm (which saves the policies and returns one of them in the subsequent stage without any new computations).*

From a system-wide point of view, a decision x_t is implemented in practice contributing stage cost $c_t(x_t)$ to the decision criterion. Using the snapshot-oriented approach, we introduce the following definition addressing the solution method of the overall multi-stage process.

Definition 10 (Algorithm)**.** *An algorithm* $\texttt{ALG}$ *for a multi-stage process is a sequence of snapshot algorithms* $\texttt{ALG}_1, \texttt{ALG}_2, \ldots, \texttt{ALG}_T$*, i.e.,* $\texttt{ALG} = (\texttt{ALG}_1, \texttt{ALG}_2, \ldots, \texttt{ALG}_T)$*.*

This means that for a given instance $\mathcal{I}$, $\texttt{ALG}$ returns a solution $\texttt{ALG}(\mathcal{I}) \in \mathrm{Sol}(\mathcal{I})$ and the components $\texttt{ALG}_t(\mathcal{I})$ constitute a decision x_t for stage $t \in \mathcal{T}$. According to its sequential structure, an algorithm is also referred to as an algorithm trajectory. Depending on a series of snapshot problems, an algorithm can only be specified in retrospective due to the snapshot problems' dependence on $I_0, I_t, \mathbb{F}_t$ and $\texttt{paradigm}_t$ for every stage $t \in \mathcal{T}$.

An easily accessible possibility of a snapshot algorithm $\texttt{ALG}_t$ for a given snapshot problem P_t is represented by a mathematical programming (MP) solver which solves a

mixed integer programming formulation of P_t. An MP-based example snapshot problem for lot sizing with uncertain demand is presented in Section 4.1.1. Depending on the amount of information contained in $\mathbb{F}_t$, a mathematical programming model may be solvable in reasonable time, especially in case of small lookahead information or a limited amount of uncertainty information. Likewise, problem-specific heuristics, metaheuristics, or even simple rule-based heuristics can be used to solve the snapshot problems. Using heuristics instead of exact algorithms for snapshot problems is also justified by the fact that there is no uniquely determined or best snapshot problem. The selection of the snapshot problem is a modeling choice itself which may be addressed with machine learning methods as proposed in the outlook in Section 6.

We refer to the tuple $(P_t, \texttt{ALG}_t)$ as the *solution method* S_t of stage t. In retrospective, i.e., at stage T, we also have full information about the solution method trajectory $S_{\leq T}$, i.e., for each stage $t \in \mathcal{T}$ we know $S_t = (P_t, \texttt{ALG}_t)$ which brings us to the notion of the so-called *extended (or retrospective) instance* $(T, I_0, D_{\leq T}, c_{\leq T}, S_{\leq T})$ which can be seen as an augmentation of an instance $(T, I_0, D_{\leq T}, c_{\leq T})$ by the solution method trajectory $S_{\leq T}$ which becomes known in retrospective. In the context of evaluation, we will mostly deal with extended instances (see Section 5). Since solution methods and data $D_{\leq T}$ have to be compatible with each other, different data options, in particular with respect to modeling future data $\mathbb{F}_t$, lead to different solution methods S_t resulting in a variety of different extended multi-stage process instances which are all based on the same underlying multi-stage process instance.

4 Examples

We present several example problems to demonstrate the modeling capacities of our framework. Four examples are period-driven and four are event-driven. Naturally, events depend on the application under consideration. For instance, in scheduling the release of a new job is an event that triggers a new stage. In a routing application the arrival of a new customer request defines a triggering event.

Each example is presented in a verbal and formal description. We will present lot sizing as our first example in a more verbose way and also include example snapshot problems. The reader should read this example first. For the sake of clarity we assume in this section that the information horizon $\tau_t + \sigma_t$ is fixed, i.e., the lookahead $\mathbb{L}_t$ and the uncertain data $\mathbb{U}_t$ both have the same size in every stage t and hence omit the subscript t for τ and σ.

4.1 Period-driven Decision Processes

4.1.1 Lot Sizing with Uncertain Demand

Unlike in the classical, deterministic Lot Sizing Problem, in the multi-stage process version of lot sizing we have to decide on the lot sizes without knowing the complete future. For this reason we are not obliged to meet the demands, but not meeting

them incurs penalty costs. The goal is to balance the storage, setup costs and penalty costs with the additional difficulty of not knowing the upcoming demands exactly.

Formal description

1. Initial input data I_0:
   - production costs c
   - setup costs f
   - storage costs h
   - penalty costs p
2. Time-dynamic input data I_t:
   - stock s_t at the beginning of stage t
   - latest demand d_t
3. Future data $\mathbb{F}_t$ about the upcoming demands
4. Stage task:
   a) $\texttt{paradigm}_t$ (see below for concrete examples)
   b) decision: decide on the production output x_t
   c) cost-per-stage: $c_t(x_t) = cx_t + f \text{ (if } x_t > 0) + p \max\{d_t - x_t - s_t, 0\} + hs_t$
5. Stage transition description:
   - $s_{t+1} = \max\{x_t + s_t - d_t, 0\}$
6. Decision criterion: minimize overall costs

The future data $\mathbb{F}_t$ contains information about the upcoming demands. It can be modeled using different uncertainty representations that we call *flavors*. These are selected to be amenable to techniques from online optimization, stochastic programming and robust optimization, respectively. Concrete examples are given in the following table.

Flavor	Lookahead $\mathbb{L}_t$	Uncertain data $\mathbb{U}_t$
OO	$d_{t+1}, \ldots, d_{t+\tau}$	$\emptyset$, i.e., $\sigma = 0$
SP	$\emptyset$, i.e., $\tau = 0$	distributions $\mathbb{P}_{t+1}, \ldots, \mathbb{P}_{t+\sigma}$
RO	$d_{t+1}, \ldots, d_{t+\tau}$	scenario sets $\mathcal{U}_{t+\tau+1}, \ldots, \mathcal{U}_{t+\tau+\sigma}$

As can be seen from the table, pure online or stochastic representations of $\mathbb{F}_t$ as well as hybrids are possible. For the sake of completeness, we also have to define $\texttt{paradigm}_t$. Obviously it has to be defined to reflect the nature of the requested flavor. Hence, we set $\texttt{paradigm}_t$ to `online with lookahead`, `expected`

`costs orientation` and `worst case costs orientation` respectively. More precise specifications like `multi-stage stochastic programming` or `adjustable robustness` are also admissible. We will omit the specifics about $\texttt{paradigm}_t$ for the following examples.

Snapshot Problems for the Online Optimization Paradigm The snapshot problem $P_{t'}$ at stage t' is constructed from the initial input data $I_0 = (c, f, h, p)$, the time-dynamic input data $I_t = (s_{t'}, d_{t'})$ and the deterministic information about the upcoming demands stored in the lookahead $\mathbb{L}_{t'} = \{d_{t'+1}, \ldots, d_{t'+\tau}\}$ for the upcoming τ stages (we assume $\tau \geq 1$). Because all the information is deterministic, we enforce that the demands are met to the full extent and no penalties occur. The corresponding Lot Sizing LP with periods $H := \{t', t'+1, \ldots, t'+\tau\}$ can be formulated as follows

$$
\begin{aligned}
\min \quad & \sum_{t=t'}^{t'+\tau} c \cdot x_t + f \cdot y_t + h \cdot s_t \\
\text{s.t.} \quad & x_t + s_t - s_{t+1} = d_t && \forall t \in H \\
& x_t \leq M y_t && \forall t \in H \\
& s_0 = s_{t'} \\
& s_{t'+\tau+1} = \kappa \\
& y_t \in \{0,1\} && \forall t \in H \\
& x_t \in \mathbb{R}_{\geq 0} && \forall t \in H \\
& s_t \in \mathbb{R}_{\geq 0} && \forall t \in H \cup \{t'+\tau+1\}.
\end{aligned}
\tag{LS-OO-t'}
$$

The parameter κ governs the ending condition and is one of the degrees of freedom for selecting a suitable snapshot problem. In the rolling horizon approach, it may not be the best choice to enforce an empty inventory at the end of the information horizon. Possible values for κ are 0, $\frac{1}{\tau+1}\sum_{t=t'+1}^{t'+\tau} d_t$ or a specific threshold value. Note that from the solution vector $(x_{t'}, \ldots, x_{t'+\tau+1})$ to LS-OO-t' we only implement $x_{t'}$ in the current stage t'. The remainder of the solution can be used for example to speed up the solution process.

A different approach would be to use a snapshot problem amenable for heuristics, e.g., from Silver and Meal [1973]. An interesting question would be how a heuristic performs in comparison with the LP-based approach presented above.

4.1.2 Network Flows with Uncertain Demands

In each stage, the goal is to provide a flow in a network satisfying prescribed demands and supplies that are uncertain and are revealed only for the current stage. In order to do so, the decision maker has to design a subnetwork by allocating capacities to the arcs in the network incurring allocation costs. Additionally maintenance costs are incurred for each already allocated edge in each stage which can be interpreted as renting costs. We assume that the maintenance costs are much smaller than the

allocation costs, but it may be cheaper in the long run to reconsider prior network design decisions and reallocate edge capacities.

Formal description

1. Initial input data I_0:
   - directed graph $G = (V, A)$
   - cost vectors $c^{alloc}, c^{maint} \in \mathbb{Z}_+^A$
2. Time-dynamic input data I_t:
   - arc capacities $C_t \in \mathbb{Z}_+^A$
   - demands and supplies $d_t \in \mathbb{R}^V$
3. Future data $\mathbb{F}_t$: upcoming demands and supplies d_t
4. Stage task:
   a) $\texttt{paradigm}_t$
   b) decision: if necessary, change arc capacities and find a d_t-flow
   c) cost-per-stage: c_a^{alloc} for a capacity unit newly allocated to arc $a \in A$, c_a^{maint} for a capacity unit already allocated to arc a in the previous stage
5. Stage transition description:
   - set C_{t+1} to all currently allocated capacities
6. Decision criterion: minimize the overall allocation and maintenance costs

The future data $\mathbb{F}_t$ contains information about the upcoming demands, supplies, and capacities. Possible flavors are shown in the following table.

Flavor	Lookahead $\mathbb{L}_t$	Uncertain data $\mathbb{U}_t$
OO	$d_{t+1}, \ldots, d_{t+\tau}$	$\emptyset$
SP	$\emptyset$	distributions $\mathbb{P}_{t+1}, \ldots, \mathbb{P}_{t+\sigma}$
RO	$\emptyset$	scenario sets $\mathcal{U}_{t+1}, \ldots, \mathcal{U}_{t+\sigma}$

4.1.3 Portfolio Selection with Uncertain Prices

We are given a set of n assets which have to be rebalanced in a portfolio over time. Asset prices are uncertain and realized in each stage. We assumed that investors have a finite planning horizon $T < \infty$ which is not known a priori. The price vector contains one component for the reference currency. Prices are measured relative to the reference currency and also the return of the portfolio is measured against the reference currency.

Formal description

1. Initial input data I_0:
   - none
2. Time-dynamic input data I_t:
   - portfolio composition x_t at the beginning of stage t
   - price vector p_t for the n assets, i.e., $p_t = (p_{t,1}, p_{t,2}, \ldots, p_{t,n})$
3. Future data $\mathbb{F}_t$: data about future price vectors
4. Stage task:
   a) $\texttt{paradigm}_t$
   b) decision: $x_t = (x_{t,1}, x_{t,2}, \ldots, x_{t,n})$ where $x_{t,i}$ gives the reallocated share of asset i in the beginning of stage t
   c) reward-per-stage: portfolio return at the end of stage t resulting from x_t and p_t; observe that also negative portfolio returns are possible
5. Stage transition description:
   - $s_{t+1} = x_t$: state is described by the percentage asset allocation, i.e., by the portfolio itself
6. Decision criterion: maximize the return of the portfolio, i.e., the sum of the returns of the assets

Minimizing the value at risk or minimizing the portfolio variance can also be considered as decision criteria; stage-wise costs would have to be re-defined in these cases The future data $\mathbb{F}_t$ contains information about the upcoming asset prices. Possible flavors are shown in the following table.

Flavor	Lookahead $\mathbb{L}_t$	Uncertain data $\mathbb{U}_t$
OO	$p_{t+1}, \ldots, p_{t+\tau}$	
SP	$\emptyset$	distributions $\mathbb{P}_{t+1}, \ldots, \mathbb{P}_{t+\sigma}$
RO	$\emptyset$	scenario sets $\{p_{t'}^{lower}, p_{t'}^{upper}, corr_{t'}\}$, $t' \in \{t+1, \ldots, t+\sigma\}$, where $corr_{t'}$ is the set of price correlations

4.1.4 Facility Location and Relocation under Uncertainty

In multi-period facility location and relocation, problem parameters such as customer demands, opening and closing costs of facilities, or costs for relocating facilities are time-dependent. The task then consists of deciding upon the locations of the facilities

to be operated in each time period such that all constraints (e.g., adhering to capacity constraints) are respected and the overall costs are minimized. Here we allow a decision to be made in each period, turning the setting into a multi-stage problem which is more suitable for a rapidly changing environment. A typical example for multi-stage facility location and relocation is given by a food truck service with a limited fleet of vehicles facing the repetitive task to decide where to locate the available food trucks for the next day based upon customer demand information (e.g., based on public event locations).

1. Initial input data I_0:
   - metric space (X, d) with set X of potential locations and existing customers, metric d on X, maximum number of facilities K
2. Time-dynamic input data I_t:
   - set of open facilities at the end of stage $t-1$, given by $F_t = \{f_1, f_2, \ldots, f_k\}$ where $f_1, f_2, \ldots, f_k \in X$, $k \leq K$ and $k \in \mathbb{N}$
   - latest customer demands D_t, cost parameter values C_t, facility capacities cap_t
3. Future data $\mathbb{F}_t$: data about demands, cost parameter values, facility capacities in upcoming stages
4. Stage task:
   a) $\texttt{paradigm}_t$
   b) decision: select the open facilities in stage t, i.e., select $F' = \{f'_1, f'_2, \ldots, f'_{k'}\}$ where $f'_1, f'_2, \ldots, f'_{k'} \in X$, $k' \leq K$ and $k' \in \mathbb{N}$
   c) cost-per-stage: costs incurred by transportation, facility opening, closing, and relocation
5. Stage transition description:
   - update current set of open facilities $F_{t+1} := F'$
6. Decision criterion: minimize overall costs incurred by transportation, facility opening, closing, and relocation

The future data $\mathbb{F}_t$ contains information about the upcoming demands $D_{t'}$, vector of cost parameter values $C_{t'}$, and vector of facility capacities $cap_{t'}$. Possible flavors are shown in the following table.

Flavor	Lookahead $\mathbb{L}_t$	Uncertain data $\mathbb{U}_t$
OO	$(D_{t'}, C_{t'}, cap_{t'}), t' \in \{t+1, \ldots, t+\tau\}$	$\emptyset$
SP	$\emptyset$	distributions for $(D_{t'}, C_{t'}, cap_{t'})$, $t' \in \{t+\tau+1, \ldots, t+\sigma\}$
RO	$\emptyset$	scenario sets for $(D_{t'}, C_{t'}, cap_{t'})$, $t' \in \{t+\tau+1, \ldots, t+\sigma\}$

4.2 Event-driven Decision Processes

4.2.1 Paging with Uncertain Release Ordering

Pages from a pre-specified page alphabet are requested over time and they have to be accessed in their order of release. The storage system consists of two memories, i.e., every page of the page alphabet is either stored in the small and fast cache memory or in the large but slow main memory. To serve a page request, the page needs to be loaded in the cache. Thus, if the requested page is not yet in the cache and the cache is full, one page from the cache has to be evicted first (and be brought into the main memory) in order to make space for the requested page. Whenever this situation occurs, a so-called page fault has been incurred. The overall goal is to achieve as few page faults as possible.

1. Initial input data I_0:
   - page alphabet $\mathcal{A} = \{a, b, c, \ldots\}$
   - cache size $k \in \mathbb{N}$
2. Time-dynamic input data I_t:
   - current cache configuration $\{x_1, x_2, \ldots, x_{k'}\}$ with $x_i \in \mathcal{A}$, $i = 1, \ldots, k' \leq k$
   - page request y_t with $y_t \in \mathcal{A}$
3. Future data $\mathbb{F}_t$: data about the upcoming page requests $y_{t'}$ with $t' > t$
4. Stage task:
   a) $\texttt{paradigm}_t$
   b) decision: in case of a page fault, decide which current cache page to evict in order to make space for the requested page in the cache
   c) cost-per-stage: 1 in case of page fault, 0 otherwise
5. Stage transition:
   - $\texttt{event}_t$: arrival of a new page request
   - update cache configuration according to decision

6. Decision criterion: minimize the number of page faults

The future data $\mathbb{F}_t$ contains information about the upcoming page requests. We are using the concept of a strong request lookahead proposed in Albers [1993], i.e., the lookahead consists of l pairwise distinct additional upcoming pages. To this end, we define $\tau(l) := \max\{t' \colon |\{y_{t+1}, \ldots, y_{t'}\}| \leq l\}$. Possible flavors are shown in the following table.

Flavor	Lookahead $\mathbb{L}_t$	Uncertain data $\mathbb{U}_t$
OO	$\{y_{t'} : t' \in \{t+1, t+2, \ldots, \tau(l)\}\}$	
SP	$\emptyset$	distributions $\mathbb{P}_{t+1}, \ldots, \mathbb{P}_{t+\sigma}$ on the page alphabet $\mathcal{A}$
RO	$\emptyset$	scenario sets $\mathcal{U}_{t+1}, \ldots, \mathcal{U}_{t+\sigma} \subseteq \mathcal{A}$

4.2.2 Scheduling with Uncertain Processing Times

We are given a single machine with a fixed speed. Jobs arrive randomly and need to be processed immediately and the decisions made cannot be adjusted later, i.e., rejected jobs are lost. Jobs can be interrupted without the possibility of resumption and only completed jobs generate a reward. We consider the case where the reward of a job is equal to its processing time. The setting can easily be extended to arbitrary rewards.

Formal description

1. Initial input data I_0:
   - none
2. Time-dynamic input data I_t:
   - job J_t with processing time p_t
3. Future data $\mathbb{F}_t$: data about the upcoming jobs and their processing times
4. Stage task:
   a) $\texttt{paradigm}_t$
   b) decision: accept or reject job J_t
   c) reward-per-stage: processing time $p_{t'}$ if the job $J_{t'}$ was completed in the current stage, i.e., between the arrival of job J_{t-1} and J_t
5. Stage transition description:
   - $\texttt{event}_t$: arrival of a new job
   - keeping track of completed jobs

6. Decision criterion: maximize the sum of the rewards of completed jobs

In the three-field notation, the pure online version (i.e., without lookahead) is $1|\mathit{online\text{-}r_j}, \mathit{cancellation}|\sum p_i$. The future data $\mathbb{F}_t$ contains information about the upcoming processing times. Possible flavors are shown in the following table.

Flavor	Lookahead $\mathbb{L}_t$	Uncertain data $\mathbb{U}_t$
OO	$p_{t+1}, \ldots, p_{t+\tau}$	$\emptyset$
SP	$\emptyset$	distributions $\mathbb{P}_{t+1}, \ldots, \mathbb{P}_{t+\sigma}$
RO	$\emptyset$	scenario sets $[l_{t'}, u_{t'}]$, $t' \in \{t+1, \ldots, t+\sigma\}$

4.2.3 Vehicle Routing with Uncertain Demands

A service provider has to meet customer demands using a fleet of homogeneous vehicles with a specified capacity operated from a single depot. Each customer request contains a preferable delivery time and the overall goal is to minimize the customers' waiting times. A customer cannot be visited before his or her chosen delivery time. In context of food delivery, a lookahead can be implemented by prescribing the customers to preorder within a specified time window.

Formal description

1. Initial input data I_0:
   - directed graph $G = (V, A)$ with $V = \{0, \ldots, n\}$ and 0 being the depot
   - homogeneous fleet of k vehicles with capacity C
2. Time-dynamic input data I_t:
   - regular disclosure time s_t
   - vehicle positions $pos_t^i = (a, \lambda) \in A \times [0, 1)$, $i \in [k]$
   - customer requests $r_t \in \mathbb{Z}_+^n$ and preferable delivery times d_t
3. Future data $\mathbb{F}_t$: upcoming requests and delivery times
4. Stage task:
   a) $\texttt{paradigm}_t$
   b) decision: find a set of routes complying with capacity restriction C
   c) cost-per-stage: waiting times at customers served
5. Stage transition:
   - $\texttt{event}_t$: arrival of a new request vector and delivery times; the regular disclosure time of these customers equals s_t

- update positions according to time passed since predecessor event (deterministic drivers can be assumed, i.e., their positions change proportional to the time passed since event $t-1$ or latest position of each driver is asked by the operator)

6. Decision criterion: minimize the sum of waiting times for all customers

The future data $\mathbb{F}_t$ consists of a time lookahead of duration Δ, i.e., all upcoming customer requests and corresponding delivery times up to the regular disclosure time $s_t + \Delta$ are known. Hence, we have $\tau(\Delta) := \max\{t' \in \mathbb{N} \mid t' > t, s_{t'} \leq s_t + \Delta\}$. In other words, the lookahead means that all requests are revealed Δ time steps earlier than in a pure online setting. For the uncertain data $\mathbb{U}_t$ the duration Ω is defined similarly, leading to $\sigma(\Omega) := \max\{t' \in \mathbb{N} \mid t' > t+\tau(\Delta), s_{t'} \leq s_t+\tau(\Delta)+\sigma(\Omega)\}$. Possible flavors are shown in the following table.

Flavor	Lookahead $\mathbb{L}_t$	Uncertain data $\mathbb{U}_t$
OO	$\{(r_{t'}, d_{t'}) : t' \in \{t+1, t+2, \ldots, \tau(\Delta)\}\}$	
SP	$\emptyset$	probability distribution D_t on customer requests $r_{t'}$ and delivery times $d_{t'}$ with $t + \tau(\Delta) < t' \leq t + \sigma(\Omega)$
RO	$\emptyset$	scenario sets $\mathcal{U}_{t+1}, \ldots, \mathcal{U}_{t+\sigma(\Omega)}$ for customer requests and delivery times

4.2.4 Bin Packing with Uncertain Item Sizes

A set of items has to be packed into bins. Each item has to be packed immediately after its arrival and this decision is irrevocable. Any data on the future does not provide any ordering information.

Formal description

1. Initial input data I_0:
   - bin capacity C
2. Time-dynamic input data I_t:
   - current item i_t with size $s_t \leq C$
3. Future data $\mathbb{F}_t$: upcoming item sizes
4. Stage task:
   a) `paradigm`$_t$

b) decision: assign item i_t to an opened bin or open a new bin

c) cost-per-stage: 1 if a new bin has been opened, 0 otherwise

5. Stage transition:
   - $\texttt{event}_t$: arrival of a new item
6. Decision criterion: minimize the number of opened bins

The future data $\mathbb{F}_t$ contains information about the upcoming item sizes. Possible flavors are shown in the following table.

Flavor	Lookahead $\mathbb{L}_t$	Uncertain data $\mathbb{U}_t$
OO	$\{s_{t+1}, \ldots, s_{t+\tau}\}$	$\emptyset$
SP	$\emptyset$	probability distribution on the interval $[0, C]$ for the size of the next σ items
RO	$\emptyset$	scenario sets $\mathcal{U}_{t'} = [l_{t'}, u_{t'}] \subseteq [0, C]$, $t' \in \{t+1, \ldots, t+\sigma\}$

5 Evaluation Module

Recap from Section 3.3 that a solution x to a a multi-stage process instance $\mathcal{I} = (T, I_0, D_{\leq T}, c_{\leq T})$ consists of a sequence of decisions

$$x = (x_1, x_2, \ldots, x_T)$$

and allows for the computation of the related decision criterion value $C_t = \sum_{s=1}^{t} c_s(x_s)$ throughout the solution process at arbitrary stages $t \in \{1, 2, \ldots, T\}$. The evaluation at stage t can be both of retrospective character assessing what has been obtained so far, and – as long as the end of the planning horizon T has not been reached – it can also be of prospective character taking into account a number of future stages that are based on predictions. Prospective evaluations throughout the solution process can be utilized to steer the system under consideration according to decision maker's current perception of the system, while retrospective evaluations focus on providing a performance summary related to the overall decision criterion.

5.1 Terminology and Notation

In order to evaluate the decision criterion value $C_t = \sum_{s=1}^{t} c_s(x_s)$ at stage t for a solution x to a decision process instance $\mathcal{I} = (T, I_0, D_{\leq T}, c_{\leq T})$ we observe that – apart from the stage-independent data I_0 – each decision x_s and hereby also its costs $c_s(x_s)$ depend on the following stage-dependent arguments:

- input data I_s
- data about the future $\mathbb{F}_s$
- solution paradigm $\texttt{paradigm}_s$
- snapshot problem P_s
- snapshot algorithm $\texttt{ALG}_s$

As introduced in Section 3.3 and Section 3.4, respectively, $D_s = (I_s, \mathbb{F}_s, \texttt{paradigm}_s)$ denotes the data of stage s, and $S_s = (P_s, \texttt{ALG}_s)$ denotes the solution method of stage s. Moreover, we introduce the following notation to simplify the subsequent presentation. With respect to previous stages and including the current stage t, we define:

- input data trajectory $I_{\leq t} := (I_1, I_2, \ldots, I_t)$
- trajectory of data about the future $\mathbb{F}_{\leq t} := (\mathbb{F}_1, \mathbb{F}_2, \ldots, \mathbb{F}_t)$
- data trajectory $D_{\leq t} := (D_1, D_2, \ldots, D_t)$, where $F_1 = ((I_1, \mathbb{F}_1, \texttt{paradigm}_1)$ etc.
- snapshot problem trajectory $P_{\leq t} := (P_1, P_2, \ldots, P_t)$
- snapshot algorithm trajectory $\texttt{ALG}_{\leq t} := (\texttt{ALG}_1, \texttt{ALG}_2, \ldots, \texttt{ALG}_t)$
- solution method trajectory $S_{\leq t} := (S_1, S_2, \ldots, S_t)$

If at stage t the decision maker intends to evaluate system performance until an arbitrary future stage $t_{fut} > t$, then it may occur that for some of the future stages $t+1, t+2, \ldots, t_{fut}$ no data is available, i.e., it may happen that $\mathbb{F}_t$ does not hold any information on data in these stages. Yet, in order to make an assessment still possible, the decision maker has to introduce additional predictions about data concerning stages $t+1, t+2, \ldots, t_{fut}$ for the sake of assessment only. Therefore, we define $\tilde{D}_s$ as the data assumptions for stage s which are used for evaluation purposes only, i.e., $\tilde{D}_s$ is not part of the modeling framework.

Let t_{fut} be the stage until which data is predicted in stage t. We summarize data as well as solution method trajectory and predictions made in stage t and concerning data until stage $t_{fut} > t$ in

$$D_{\leq t}^{\leq t_{fut}} := (D_1, D_2, \ldots, D_t, \tilde{D}_{t+1}, \tilde{D}_{t+2}, \ldots, \tilde{D}_{t_{fut}}) \text{ and}$$
$$S_{\leq t}^{\leq t_{fut}} := (S_1, S_2, \ldots, S_t, \tilde{S}_{t+1}, \tilde{S}_{t+2}, \ldots, \tilde{S}_{t_{fut}}).$$

Because of the prospective character and in alignment with Section 3.4, we call $D_{\leq t}^{\leq t_{fut}}$ and $S_{\leq t}^{\leq t_{fut}}$ data and solution method trajectories, respectively.

Based on this notation, the evaluation module offers possibilities for measuring the realized performance of trajectory tuples

$$\mathcal{H}_{\leq t} := (I_0, D_{\leq t}, S_{\leq t})$$

for $t = 1, 2, \ldots, T$. Retrospective evaluation at stage t can hence be carried out immediately at stage t such that the evaluation of trajectory tuples happens during the course of solving the multi-stage process and gives the decision maker a continuous feedback on the currently achieved performance. Likewise, the evaluation module can be used to predict the anticipated performance of trajectory tuples

$$\tilde{\mathcal{H}}_{\leq t}^{\leq t_{fut}} := (I_0, D_{\leq t}^{\leq t_{fut}}, S_{\leq t}^{\leq t_{fut}})$$

until stage t_{fut} with $t_{fut} > t$.

According to the decision criterion $C_t = \sum_{s=1}^{t} c_s(x_s)$, evaluations are based upon the costs per stage that have been incurred already or are predicted to be realized in the future, respectively. Section 5.2 introduces basic performance evaluation measures. At stage t with $t_{fut} > t$, we obtain prospective performance measures, while for $t = t_{fut}$ the measures reduce to a retrospective performance analysis.

5.2 Evaluation Measures

Absolute Cost Values

For given data trajectory $D_{\leq t}^{\leq t_{fut}}$ and solution method trajectory $S_{\leq t}^{\leq t_{fut}}$, the following measures related to the decision criterion are computed in order to evaluate the resulting quality of the pair $(D_{\leq t}^{\leq t_{fut}}, S_{\leq t}^{\leq t_{fut}})$ in absolute terms.

- Accumulated costs per stage until stage t_{fut} at stage t:

$$C(t, t_{fut}, D_{\leq t}^{\leq t_{fut}}, S_{\leq t}^{\leq t_{fut}}) := \sum_{s=1}^{t} c_s(x_s(D_s, S_s)) + \sum_{s=t+1}^{t_{fut}} c_s(x_s(\tilde{D}_s, S_s)).$$

- Break-down of costs per stage in a vector at stage t where each component gives the contribution of the respective stages:

$$\begin{aligned} C_{stage}(t, t_{fut}, D_{\leq t}^{\leq t_{fut}}, S_{\leq t}^{\leq t_{fut}}) \quad := \quad & (c_1(x_1(D_1, S_1)), c_2(x_2(D_2, S_2)), \ldots, \\ & c_t(x_t(D_t, S_t)), c_{t+1}(x_{t+1}(\tilde{D}_{t+1}, S_{t+1})), \ldots, \\ & c_{t_{fut}}(x_{t_{fut}}(\tilde{D}_{t_{fut}}, S_{t_{fut}}))). \end{aligned}$$

The elements of C_{stage} can be subjected to further statistical processing such as a deviation analysis of the single components from the mean value or a variance analysis of the single components.

Relative Cost Values

For two given pairs of data and solution method trajectories $(D'^{\leq t_{fut}}_{\leq t}, S'^{\leq t_{fut}}_{\leq t})$ and $(D''^{\leq t_{fut}}_{\leq t}, S''^{\leq t_{fut}}_{\leq t})$, we can provide a relative performance evaluation of these two pairs using the following notation.

- Ratio of realized accumulated costs per stage:

$$R(t, t_{fut}, (D'^{\leq t_{fut}}_{\leq t}, S'^{\leq t_{fut}}_{\leq t}), (D''^{\leq t_{fut}}_{\leq t}, S''^{\leq t_{fut}}_{\leq t})) := \frac{C(t, t_{fut}, D'^{\leq t_{fut}}_{\leq t}, S'^{\leq t_{fut}}_{\leq t})}{C(t, t_{fut}, D''^{\leq t_{fut}}_{\leq t}, S''^{\leq t_{fut}}_{\leq t})}$$

- Break-down of ratios of costs per stage in a vector at stage t with

$$R_{stage,s}((D'^{\leq t_{fut}}_{\leq t}, S'^{\leq t_{fut}}_{\leq t}), (D''^{\leq t_{fut}}_{\leq t}, S''^{\leq t_{fut}}_{\leq t})) := \frac{c_s(x_s(D'_s, S'_s))}{c_s(x_s(D''_s, S''_s,))} \quad \text{for } s = 1, 2, \ldots, t,$$

$$R_{stage,s}((D'^{\leq t_{fut}}_{\leq t}, S'^{\leq t_{fut}}_{\leq t}), (D''^{\leq t_{fut}}_{\leq t}, S''^{\leq t_{fut}}_{\leq t})) := \frac{c_s(x_s(\tilde{D}'_s, S'_s))}{c_s(x_s(, \tilde{D}''_s, S''_s,))} \quad \text{for } s = t+1, \ldots, t_{fut}.$$

By appropriately specifying the four elements $D'^{\leq t_{fut}}_{\leq t}$, $S'^{\leq t_{fut}}_{\leq t}$, $D''^{\leq t_{fut}}_{\leq t}$, $S''^{\leq t_{fut}}_{\leq t}$, these ratios contain several interesting cases depending on the specification of the set of future data $\mathbb{F}_s$ (containing deterministic lookahead data $\mathbb{L}_s$ and uncertain data $\mathbb{U}_s$) for different stages $s \in \{1, 2, \ldots, T\}$: The competitive ratio from online optimization is captured by comparing a pure online setting ($\mathbb{L}_s = \emptyset$ for $s = 1, 2, \ldots, T$) with an offline setting ($\mathbb{L}_s$ contains all information about stages $s+1, s+2, \ldots, T$ for $s \in \{1, 2, \ldots, T\}$); the value of lookahead in online optimization is elicited through comparing an online setting with lookahead ($\mathbb{L}_s \neq \emptyset$ for $s \in \{1, 2, \ldots, T\}$) to an online setting without lookahead ($\mathbb{L}_s = \emptyset$ for $s = 1, 2, \ldots, T$); the price of robustness is found by comparing a robust optimization setting ($\mathbb{U}_s \neq \emptyset$ for $s \in \{1, 2, \ldots, T\}$) with an offline setting ($\mathbb{L}_1$ contains all information about stages $1, 2, \ldots, T$ and $\mathbb{L}_s = \emptyset$ for $s = 2, 3, \ldots, T$). Likewise, both for stochastic programming and robust optimization it becomes possible to analyze the effect of providing different granularity of scenario information in the data histories.

5.3 Sensitivity Analysis

The decision criterion value $C_t = \sum_{s=1}^{t} c_s(x_s)$ attained at the end of stage t of a multi-stage optimization process depends on the decisions x_s taken in the course of the solution process. The decisions x_s, $s \in \{1, \ldots, t\}$, in turn depend on the data $D_s = (I_s, \mathbb{F}_s, \texttt{paradigm}_s)$ and the solution methods $S_s = (P_s, \texttt{ALG}_s)$ of the corresponding stages. Moreover, if prospective performance measurement until stage

$t_{fut} > t$ is conducted in stage t, then anticipated future decisions depend on predicted data $\tilde{D}_s$ with $s = t+1, \ldots, t_{fut}$. Recall that the sequences of data and solution methods are subsumed in the trajectories $D_{\leq t}$, $D_{\leq t}^{\leq t_{fut}}$, $S_{\leq t}$, and $S_{\leq t}^{\leq t_{fut}}$. Sensitivity analysis examines the effects on the decision criterion value when one or more of the components leading to the decision trajectory are changed. Thus, we consider the influence of changes in any parameter of an extended multi-stage process instance $(T, I_0, D_{\leq T}, c_{\leq T}, S_{\leq T})$, i.e., data components in $(I_s, \mathbb{F}_s, \texttt{paradigm}_s)$ from $D_{\leq T}$ and solution method components in $S_s = (P_s, \texttt{ALG}_s)$ from $S_{\leq T}$.

In contrast to the evaluation of absolute and relative cost values, sensitivity analysis is concerned with solutions and outcomes gathered from a large number of uncertainty realizations. Hence, various data and solution method trajectories have to be sampled to obtain substantial results. Methodologically, the necessity of sampling several such trajectories places sensitivity analysis in a position to be utilized both in retrospective and prospective fashion ("what would have been if" vs. "what is going to be if").

Scope of Sensitivity Analysis

Sensitivity analysis yields quantitative conclusions about the effect of changing specific parts of an underlying multi-stage process instance. To this end, the following components of an underlying instance $(T, I_0, D_{\leq T}, c_{\leq T}, S_{\leq T})$ can define a typical scope for sensitivity analysis. In each case, the varying component requires a specification of the range of values which are to be considered. Recall that sensitivity analysis asks to carry out the analysis over a sufficiently large set of underlying multi-stage process instances.

- Varying $\mathbb{F}_s$ in $D_s = (I_s, \mathbb{F}_s, \texttt{paradigm}_s)$ and constant $S_s = (P_s, \texttt{ALG}_s)$: This scope focuses on different levels of future information $\mathbb{F}_s$ while the solution method is kept constant. Typical examples would be to compare different lookahead sizes in online optimization or varying scenario granularity in stochastic programming or robust optimization. Hence, this analysis can be used to assess the value of data (e.g., additional lookahead or finer probabilistic information).

- Varying $\mathbb{F}_s$ as well as $\texttt{paradigm}_s$ in $D_s = (I_s, \mathbb{F}_s, \texttt{paradigm}_s)$ and varying $S_s = (P_s, \texttt{ALG}_s)$: When both the type of future data and the solution methods are varied, comparisons between different solution paradigms and related snapshot problems and algorithms are facilitated. In particular, different paradigms (online optimization, robust optimization, stochastic programming) can be tested against each other and a cross-comparison between different optimization paradigms becomes possible. Moreover, offline algorithms or stochastic programming methods with non-anticipativity relaxations (see Santos et al. [2018]) can be considered as hypothetical omniscient adversaries (perfect information relaxations without uncertainty).

- Constant $D_s = (I_s, \mathbb{F}_s, \texttt{paradigm}_s)$ and varying $S_s = (P_s, \texttt{ALG}_s)$: Under the same data trajectory, different snapshot problem definitions and snapshot algorithms can be tested to analyze sensitivity with respect to the character of the snapshot approach. As an interesting special case, one can consider a comparison between $\texttt{ALG}_s$ as an exact snapshot algorithm versus a heuristic snapshot algorithm. This yields the value of exact reoptimization.

Distributional Presentation of Sensitivity Analysis Results

Based on the definition of multi-stage process instances in Section 3.3 and the different scopes for sensitivity analysis, we suggest a distributional presentation of the results obtained upon varying some instance elements. Observe that varying one or more elements of a multi-stage process instance leads to a set of considered instances $\mathbb{I}$. Moreover, it is possible to define some nominal instance $\mathcal{I}_{nom} \in \mathbb{I}$ which can be understood as the reference point to which other instances from $\mathbb{I}$ are compared. In the general case, there are no probabilistic information available concerning the likelihood for some instance $\mathcal{I} \in \mathbb{I}$ to occur. Therefore, we recur to the concept of counting distribution functions (see Dunke and Nickel [2016]) which imputes a uniform distribution over all elements from $\mathcal{I}$ leading to counting results. For $v \in \mathbb{R}$, the counting distribution function value $F_{\mathbb{I}}(v)$ gives the number of instances in $\mathbb{I}$ leading to a decision criterion value at most v. For the sake of a simple presentation, we restrict ourselves to the case where $\mathbb{I}$ is a finite discrete set.

Definition 11 (Counting distribution function of decision criterion value)**.** *Let $\mathbb{I}$ be the set of instances for a multi-stage decision process $\mathcal{P}$ to be considered in the sensitivity analysis and $\texttt{ALG}$ an algorithm for $\mathcal{P}$. Also let $\texttt{ALG}(\mathcal{I})$ be the solution to an instance $\mathcal{I} \in \mathbb{I}$ determined by the algorithm $\texttt{ALG}$ and $C(\texttt{ALG}(\mathcal{I}))$ its decision criterion value. Then the function $F_{\mathbb{I}} : \mathbb{R} \to [0, 1]$ with*

$$F_{\mathbb{I}}(v) := \frac{1}{|\mathbb{I}|} \sum_{\mathcal{I} \in \mathbb{I}} \mathbf{1}_{[-\infty, v]}(C(\texttt{ALG}(\mathcal{I})))$$

is called the counting distribution function of the decision criterion value over $\mathbb{I}$[3]*.*

For a comparison of decision criterion values relative to the nominal instance $\mathcal{I}_{nom} \in \mathbb{I}$, the following definition can be utilized.

Definition 12 (Counting distribution function of decision criterion value relative to nominal instance)**.** *Let $\mathbb{I}$ be the set of instances for a multi-stage decision process $\mathcal{P}$ to be considered in the sensitivity analysis, let $\texttt{ALG}$ be an algorithm for $\mathcal{P}$ and let $\mathcal{I}_{nom} \in \mathbb{I}$ be a nominal instance. Also let $\texttt{ALG}(\mathcal{I})$ be the solution to an instance $\mathcal{I} \in \mathbb{I}$*

[3]Let A be a set, then the indicator function $\mathbf{1}_A(x)$ is 1 if $x \in A$ and 0 otherwise.

determined by the algorithm ALG *and* $C(\texttt{ALG}(\mathcal{I}))$ *its decision criterion value. Then the function* $F_{\mathbb{I},\mathcal{I}_{nom}} : \mathbb{R} \to [0,1]$ *with*

$$F_{\mathbb{I},\mathcal{I}_{nom}}(v) := \frac{1}{|\mathbb{I}|} \sum_{\mathcal{I} \in \mathbb{I}} \mathbf{1}_{[-\infty,v]} \left(\frac{C(\texttt{ALG}(\mathcal{I}))}{C(\texttt{ALG}(\mathcal{I}_{nom}))} \right)$$

is called the counting distribution function of the decision criterion value over $\mathbb{I}$ *relative to the nominal instance* $\mathcal{I}_{nom}$*.*

Finally, we integrate the possibility to display distributional results in case of two instance sets that are to be compared to each other. Therefore, consider now two sets of instances $\mathbb{I}'$ and $\mathbb{I}''$ as well as a bijection $b : \mathbb{I}' \to \mathbb{I}''$. The bijection b can be thought of as a mechanism which prescribes how instances from $\mathbb{I}'$ are changed systematically, e.g., through exchanging data and solution method trajectories amenable to online optimization with data and solution method trajectories amenable to stochastic programming. Such a bijection can be used to analyze the impact of a specific change in underlying multi-stage process instances $(T, I_0, D_{\leq T}, c_{\leq T}, S_{\leq T})$.

Definition 13 (Counting distribution function of relative decision criterion values)**.** *Let* $\mathbb{I}', \mathbb{I}''$ *be two sets of process instances for a mulit-stage decision process* $\mathcal{P}$ *to be considered and compared in the sensitivity analysis, let* b *be a bijection between* $\mathbb{I}'$ *and* $\mathbb{I}''$*, and let* ALG *be an algorithm for* $\mathcal{P}$*. Also let* $\texttt{ALG}(\mathcal{I}'))$ *be the solution for an instance* $\mathcal{I}' \in \mathbb{I}$ *determined by the algorithm* ALG *and* $C(\texttt{ALG}(\mathcal{I}'))$ *its decision criterion value. Then the function* $F_{\mathbb{I}',\mathbb{I}''} : \mathbb{R} \to [0,1]$ *with*

$$F_{\mathbb{I}',\mathbb{I}''}(v) := \frac{1}{|\mathbb{I}'|} \sum_{\mathcal{I}' \in \mathbb{I}'} \mathbf{1}_{[-\infty,v]} \left(\frac{C(\texttt{ALG}(\mathcal{I}'))}{C(\texttt{ALG}(b(\mathcal{I}')))} \right)$$

is called counting distribution function of relative decision criterion values over $\mathbb{I}'$ *and* $\mathbb{I}''$*.*

We believe that the distributional presentation of sensitivity analysis results yields the largest benefit to a decision maker since it allows for a holistic examination including worst case, best case and average case analysis. Additionally, ranges of attainable performance measures become apparent immediately.

As a major advantage of the distributional analysis we point out the possibility that in case of available stochastic information about occurrence probabilities, the above definitions can easily be adapted to the general case where instances are weighted with their occurrence probability. On the other hand, sensitivity analysis can be also studied on the snapshot level, when stochastic programming methods are applied by the decision maker and the utilized stochastic information differs from the true occurrence probabilities. In this sense, the decision maker can check the sensitivity of the stochastic programming results under varying probabilistic scenarios.

5.4 Robustness Analysis

While sensitivity analysis considers the effect of parameter changes on solution quality (in terms of the decision criterion value), robustness analysis additionally asks for the effect of parameter changes on the decision (or solution) space. To this end, consider two underlying multi-stage process instances $\mathcal{I}' = (T, I_0, D'_{\leq T}, c'_{\leq T}, S'_{\leq T})$ and $\mathcal{I}'' = (T, I_0, D''_{\leq T}, c''_{\leq T}, S''_{\leq T})$, with $D'_t = (I_t, \mathbb{F}'_t, \texttt{paradigm}'_t), D''_t = (I_t, \mathbb{F}''_t, \texttt{paradigm}''_t)$. Let $x' \in \text{Sol}(\mathcal{I}')$ and $x'' \in \text{Sol}(\mathcal{I}'')$ be the decision (or solution) trajectories and let $distance(x', x'')$ be a measure for the distance between x' and x'', e.g., a metric on a suitable vector space. Then we make the following definitions for robustness between two instances:

- $\mathcal{I}'$ and $\mathcal{I}''$ are called objective-robust relative to each other when $|C_T(x') - C_T(x'')| < \epsilon_{objective}$ for some prespecified value $\epsilon_{objective} > 0$ giving a maximum allowable distance between the decision criterion values attained by x' and x'', respectively.

- $\mathcal{I}'$ and $\mathcal{I}''$ are called decision-robust (or solution-robust) relative to each other when $distance(x', x'') < \epsilon_{decision}$ for some prespecified value $\epsilon_{decision} > 0$ giving a maximum allowable distance between x' and x''.

Consider now two sets of instances $\mathbb{I}'$ and $\mathbb{I}''$ as well as a bijection b between $\mathbb{I}'$ and $\mathbb{I}''$. The bijection b can be thought of as a mechanism which prescribes how instances from $\mathbb{I}'$ are changed systematically, e.g., through exchanging data and solution method trajectories amenable to online optimization with data and solution method trajectories amenable to stochastic programming. Then, in analogy to the above definition of robustness for instances, we make the following definitions for robustness between two sets of instances:

- $\mathbb{I}'$ and $\mathbb{I}''$ are called objective-robust relative to each other when each pair $(\mathcal{I}', b(\mathcal{I}'))$ with $\mathcal{I}' \in \mathbb{I}'$ consists of two multi-stage process instances which are objective robust relative to each other w.r.t. to a fixed $\epsilon_{objective} > 0$.

- $\mathbb{I}'$ and $\mathbb{I}''$ are called decision-robust (or solution-robust) relative to each other when each pair $(\mathcal{I}', b(\mathcal{I}'))$ with $\mathcal{I}' \in \mathbb{I}'$ consists of two multi-stage process instances which are decision (or solution) robust relative to each other w.r.t. to a fixed $\epsilon_{decision} > 0$.

Finally, we consider another concept of robustness. To this end, consider a family of underlying multi-stage process instances

$$\mathbb{I}_{\mathcal{D}_{\leq T}} := \{(T, I_0, D_{\leq T}, c_{\leq T}, S_{\leq T}) \,|\, D_{\leq T} \in \mathcal{D}_{\leq T}\}$$

with fixed values for T, I_0, $c_{\leq T}$, $S_{\leq T}$ and varying $D_{\leq T} \in \mathcal{D}_{\leq T}$ where the set $\mathcal{D}_{\leq T}$ is the set of admissible data trajectories. Since the data $D_s = (I_s, \mathbb{F}_s, \texttt{paradigm}_s)$ of stage s allows for varying stage input data I_s, the set $\mathbb{I}_{\mathcal{D}_{\leq T}}$ is typically understood

as the set of all instances when the solution method is fixed upfront as is the case in general frameworks which do not explicitly integrate different solution methods. Moreover, let $dispersion(\mathbb{I}_{\mathcal{D}_{\leq T}})$ be a dispersion measure for the set of objective values obtained over all $D_{\leq T} \in \mathcal{D}_{\leq T}$. We call the value of the measure $dispersion(\mathbb{I}_{\mathcal{D}_{\leq T}})$ the fixed-solution-method robustness of $S_{\leq T}$ over instance set $\mathbb{I}_{\mathcal{D}_{\leq T}}$.

6 Outlook

The next steps are a thorough numerical analysis of the framework and the related evaluation module in several sample applications from the area of production and logistics. Problems both of basic character (such as lot sizing, vehicle routing, scheduling) will be considered as well as problems with real world character (such as production environments or traffic systems). In the latter case, the framework is embedded into discrete-event simulation models in order to obtain a realistic model. Methodologically, further steps will be taken to address the problem of finding the most appropriate snapshot problems and algorithms in the course of the online decision process. To this end, a rule-based approach will be followed which determines the selection of the most fitting snapshot problem and algorithm depending on a set of criteria which are formulated in advance and adopt the performance assessment of different solution approaches as provided by the evaluation module.

The application of the proposed framework for multi-stage optimization problems under uncertainty inherently requires the iterative formulation of a well-suited snapshot problems and the utilization of snapshot algorithms. In terms of the overall outcome, an interesting research question asks for a profound methodology to determine these two components iteratively throughout the solution process. Answering this research question can be done by integrating machine learning techniques into the discussed mathematical optimization approaches in order to obtain system-state predictions about the performance of different solution method trajectories. The overall goal then consists of designing an automated methodology which decides upon snapshot problems and algorithms to be used throughout the online solution process. Automated algorithm configuration and selection is a promising stream of research addressing this question. Traditionally, it is concerned with tuning parameters of optimization algorithms in order to achieve favorable algorithm behavior (Kotthoff [2016], Kotthoff et al. [2017], Kerschke et al. [2018]). When the choice of the snapshot problem and the snapshot algorithm itself is a parameter, the setting can be interpreted as choosing the best candidate from a set of viable algorithms. However, algorithms to be used in an online decision process as parameters have not been considered so far in literature. Hence, this topic requires original research to be carried out with a focus on the setting as introduced in this article. A further method to algorithmically create algorithms are hyper-heuristics. These are heuristics to choose heuristics, i.e., they generate algorithms by combining elements from a set of heuristic components (Burke et al. [2013], Drake et al. [2019]). Hyper-heuristics intend to make algorithms

for new applications easily available in a meta-procedure which is as independent from an application as possible, but rather seeks within the available search space of heuristics to assemble a suitable algorithm for a given problem. Finally, considering the decision criterion as a single scalar-valued overall objective may represent an inappropriate approach to real world optimization problems which involve a multitude of stakeholders with different objectives. Therefore, another topic for future research lies in the extension of the methodology to multiple objective functions.

Acknowledgments This work is supported by the German Research Foundation (DFG) [Grants DU 1755/1-1 and NI 521/9-1, project no. 354864080]. This support is gratefully acknowledged.

References

R. K. Ahuja, R. H. Möhring, and C. D. Zaroliagis, editors. *Robust and Online Large-Scale Optimization: Models and Techniques for Transportation Systems*. Springer, 2009. doi:10.1007/978-3-642-05465-5. 10

S. Albers. The influence of lookahead in competitive paging algorithms. In T. Lengauer, editor, *European Symposium on Algorithms – ESA '93*, pages 1–12. Springer, 1993. doi:10.1007/pl00009158. 26

H. Bakker, F. Dunke, and S. Nickel. A structuring review on multi-stage optimization under uncertainty: Aligning concepts from theory and practice. *Omega*, 2019. doi:10.1016/j.omega.2019.06.006. 1, 8

R. Bellman. *Dynamic Programming*. Princeton University Press, 1957. doi:10.1126/science.153.3731.34. 6

S. Ben-David and A. Borodin. A new measure for the study of on-line algorithms. *Algorithmica*, 11(1):73–91, 1994. doi:10.1007/bf01294264. 12

A. Ben-Tal, B. Golany, A. Nemirovski, and J.-P. Vial. Retailer-supplier flexible commitments contracts: A robust optimization approach. *Manufacturing and Service Operations Management*, 7(3):248–271, 2005. doi:10.1287/msom.1050.0081. 12

A. Ben-Tal, L. El Ghaoui, and A. S. Nemirovskij. *Robust optimization*. Princeton series in applied mathematics. Princeton University Press, 2009. doi:10.1515/9781400831050. 10

D. Bertsimas and C. Caramanis. Finite adaptability in multistage linear optimization. *IEEE Transactions on Automatic Control*, 55(12):2751–2766, 2010. doi:10.1109/TAC.2010.2049764. 7

D. Bertsimas and C. McCord. From predictions to prescriptions in multistage optimization problems. *arXiv preprint arXiv:1904.11637*, 2019. 8

D. Bertsimas and D. Pachamanova. Robust multiperiod portfolio management in the presence of transaction costs. *Computers and Operations Research*, 35(1):3 – 17, 2008. doi:10.1016/j.cor.2006.02.011. 9

D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004. doi:10.1287/opre.1030.0065. 12

D. Bertsimas and A. Thiele. *Robust and Data-Driven Optimization: Modern Decision Making Under Uncertainty*, chapter 4, pages 95–122. INFORMS, 2006. doi:10.1287/educ.1063.0022. 11

D. Bertsimas, D. B. Brown, and C. Caramanis. Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501, 2011a. doi:10.1137/080734510. 10

D. Bertsimas, V. Goyal, and X. Andy Sun. A geometric characterization of the power of finite adaptability in multistage stochastic and adaptive optimization. *Mathematics of Operations Research*, 36(1):24–54, 2011b. doi:10.1287/moor.1110.0482. 8

D. Bertsimas, V. Gupta, and N. Kallus. Robust sample average approximation. *Mathematical Programming*, 171(1-2):217–282, 2018. doi:10.1007/s10107-017-1174-z. 12

C. Bes and S. P. Sethi. Concepts of forecast and decision horizons: Applications to dynamic stochastic optimization problems. *Mathematics of Operations Research*, 13(2):295–310, 1988. doi:10.1287/moor.13.2.295. 9

J. R. Birge. State-of-the-art-survey-stochastic programming: Computation and applications. *INFORMS Journal on Computing*, 9(2):111–133, 1997. doi:10.1287/ijoc.9.2.111. 10

J. R. Birge and F. Louveaux. *Introduction to stochastic programming*. Springer, 2011. doi:10.1007/978-1-4614-0237-4. 10, 11, 12, 13

A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 2005. 12

J. Boyar, S. Irani, and K. Larsen. A comparison of performance measures for online algorithms. In *Proceedings of the 11th International Symposium on Algorithms and Data Structures*, pages 119–130, 2009. doi:10.1007/s00453-014-9884-6. 12

P. Brandimarte. Multi-item capacitated lot-sizing with demand uncertainty. *International Journal of Production Research*, 44(15):2997–3022, 2006. doi:10.1080/00207540500435116. 9

E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, 2013. doi:10.1057/jors.2013.71. 37

C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer, Second edition, 2010. doi:10.1007/978-0-387-68612-7. 9

F. J. C. T. de Ruiter, R. C. M. Brekelmans, and D. den Hertog. The impact of the existence of multiple adjustable robust solutions. *Mathematical Programming*, 160 (1):531–545, 2016. doi:10.1007/s10107-016-0978-6. 9

R. Dorrigiv, A. López-Ortiz, and J. Munro. On the relative dominance of paging algorithms. *Theoretical Computer Science*, 410(38-40):3694–3701, 2009. doi:10.1016/j.tcs.2009.04.023. 12

J. H. Drake, A. Kheiri, E. Özcan, and E. K. Burke. Recent advances in selection hyper-heuristics. *European Journal of Operational Research*, 2019. doi:10.1016/j.ejor.2019.07.073. 37

F. Dunke and S. Nickel. A general modeling approach to online optimization with lookahead. *Omega*, 63:134–153, 2016. doi:10.1016/j.omega.2015.10.009. 11, 12, 34

F. Dunke and S. Nickel. Online optimization with gradual look-ahead. *Operational Research*, pages 1–35, 2019. doi:10.1007/s12351-019-00506-z. 11

F. Dunke, I. Heckmann, S. Nickel, and F. Saldanha da Gama. Time traps in supply chains: Is optimal still good enough? *European Journal of Operational Research*, 264(3):813 – 829, 2016. doi:10.1016/j.ejor.2016.07.016. 4

A. Ek, M. G. de la Banda, A. Schutt, P. J. Stuckey, and G. Tack. Modelling and solving online optimisation problems. 2020. 8

L. F. Escudero, A. Garín, M. Merino, and G. Pérez. The value of the stochastic solution in multistage problems. *TOP*, 15(1):48–64, 2007. doi:10.1007/s11750-007-0005-4. 12

A. Fiat and G. Woeginger, editors. *Online Algorithms: The State of the Art*. Springer, 1998. doi:10.1007/BFb0029561. 11

B. Fleischmann, H. Meyr, and M. Wagner. Advanced planning. In H. Stadtler, C. Kilger, and H. Meyr, editors, *Supply Chain Management and Advanced Planning: Concepts, Models, Software, and Case Studies*, pages 71–95. Springer, Fifth edition, 2015. doi:10.1007/978-3-642-55309-7_4. 9

M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979. 16

German Research Foundation (DFG). Sequential decision making under system-inherent uncertainty: Mathematical optimization methods for time-dynamic applications, 2020. URL `https://dol.ior.kit.edu/english/Projects_DFG.php`. accessed 2020-06-05. 1, 5

G. Ghiani, G. Laporte, and R. Musmanno. *Introduction to Logistics Systems Planning and Control*. Wiley, 2004. doi:10.1002/0470014040. 11

M. Goerigk and A. Schöbel. Algorithm engineering in robust optimization. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9220 LNCS:245–279, 2016. doi:10.1007/978-3-319-49487-6_8. 10

B. L. Gorissen, İ. Yanıkoğlu, and D. den Hertog. A practical guide to robust optimization. *Omega*, 53:124 – 137, 2015. doi:10.1016/j.omega.2014.12.006. 11

A. Gosavi. *Simulation-Based Optimization : Parametric Optimization Techniques and Reinforcement Learning*. Operations Research/Computer Science Interfaces Series. Springer, Second edition, 2015. doi:10.1007/978-1-4899-7491-4. 9

M. Grötschel, S. Krumke, and J. Rambau, editors. *Online Optimization of Large Scale Systems*. Springer, 2001a. doi:10.1007/978-3-662-04331-8. 8

M. Grötschel, S. Krumke, J. Rambau, T. Winter, and U. Zimmermann. Combinatorial online optimization in real time. In M. Grötschel, S. Krumke, and J. Rambau, editors, *Online Optimization of Large Scale Systems*, pages 679–704. Springer, 2001b. doi:10.1007/978-3-662-04331-8_33. 11

B. Hiller and T. Vredeveld. Probabilistic alternatives for competitive analysis. *Computer Science - Research and Development*, 27(3):189–196, 2012. doi:10.1007/s00450-011-0149-1. 12

G. N. Iyengar. Robust dynamic programming. *Mathematics of Operations Research*, 30(2):257–280, 2005. doi:10.1287/moor.1040.0129. 10

P. Jaillet and M. R. Wagner. *Online Optimization—An Introduction*, chapter 6, pages 142–152. 2010. doi:10.1287/educ.1100.0072. 11

P. Kall and J. Mayer. *Stochastic Linear Programming: Models, Theory, and Computation*. Springer, Second edition, 2011. doi:10.1007/b105472. 10

P. Kerschke, H. Hoos, F. Neumann, and H. Trautmann. Automated algorithm selection: Survey and perspectives. *Evolutionary Computation*, 27(1):3–45, 2018. doi:10.1162/evco_a_00242. 37

A. J. King and S. W. Wallace. *Modeling With Stochastic Programming*. Springer, 2012. doi:10.1007/978-0-387-87817-1. 13

K. Klamroth, E. Köbis, A. Schöbel, and C. Tammer. A unified approach to uncertain optimization. *European Journal of Operational Research*, 260(2):403–420, 2017. doi:10.1016/j.ejor.2016.12.045. 7

L. Kotthoff. Algorithm selection for combinatorial search problems: A survey. In C. Bessiere, L. De Raedt, L. Kotthoff, S. Nijssen, B. O'Sullivan, and D. Pedreschi, editors, *Data Mining and Constraint Programming: Foundations of a Cross-Disciplinary Approach*, pages 149–190. Springer, 2016. doi:10.1007/978-3-319-50137-6_7. 37

L. Kotthoff, C. Thornton, H.H. Hoos, F. Hutter, and K. Leyton-Brown. Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. *Journal of Machine Learning Research*, 18:1–5, 2017. doi:10.1007/978-3-030-05318-5_4. 37

E. Koutsoupias and C. Papadimitriou. Beyond competitive analysis. *SIAM Journal of Computing*, 30(1):300–317, 2000. doi:10.1137/S0097539796299540. 12

S. Lafortune. Discrete event systems: Modeling, observation, and control. *Annual Review of Control, Robotics, and Autonomous Systems*, 2(1):141–159, 2019. doi:10.1146/annurev-control-053018-023659. 9

S. Mitrović-Minić, R. Krishnamurti, and G. Laporte. Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38(8):669 – 685, 2004. doi:10.1016/j.trb.2003.09.001. 9, 12, 17

A. Nilim and L. El Ghaoui. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005. doi:10.1287/opre.1050.0216. 10

G. C. Pflug and A. Pichler. A distance for multistage stochastic optimization models. *SIAM Journal on Optimization*, 22(1):1–23, 2012. doi:10.1137/110825054. 7

W. B. Powell. Clearing the Jungle of Stochastic Optimization. In *Bridging Data and Decisions*, INFORMS TutORials in Operations Research, pages 109–137. INFORMS, 2014. doi:10.1287/educ.2014.0128. 13

W. B. Powell. A unified framework for stochastic optimization. *European Journal of Operational Research*, 2018. doi:10.1016/j.ejor.2018.07.014. 8, 15

W. B. Powell, P. Jaillet, and A. Odoni. Stochastic and dynamic networks and routing. In *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, pages 141 – 295. Elsevier, 1995. doi:10.1016/S0927-0507(05)80107-0. 9, 12

H.N. Psaraftis. Dynamic vehicle routing problems. In *Vehicle Routing: Methods and Studies*, volume 16 of *Studies in Management Science and Systems*, pages 223 – 248. North-Holland, 1988. 9

M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 2005. 6

M. C. Santos, M. Poss, and D. Nace. A perfect information lower bound for robust lot-sizing problems. *Annals of Operations Research*, 271(2):887–913, 2018. doi:10.1007/s10479-018-2908-x. 33

G. Schuh, V. Stich, T. Brosze, S. Fuchs, C. Pulz, J. Quick, M. Schürmeyer, and F. Bauhoff. High resolution supply chain management: Optimized processes based on self-optimizing control loops and real time data. *Production Engineering*, 5(4): 433–442, 2011. doi:10.1007/s11740-011-0320-3. 11

S. Sethi and G. Sorger. A theory of rolling horizon decision making. *Annals of Operations Research*, 29(1):387–415, 1991. doi:10.1007/BF02283607. 9

C. Shang, X. Huang, and F. You. Data-driven robust optimization based on kernel learning. *Computers & Chemical Engineering*, 106:464 – 479, 2017. doi:10.1016/j.compchemeng.2017.07.004. 11

A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on Stochastic Programming: Modeling and Theory, Second Edition*. MOS-SIAM Series on Optimization. SIAM, 2014. doi:10.1137/1.9781611973433. 10

E. A. Silver and H. C. Meal. A heuristic for selecting lot size quantities for the case of a deterministic time-varying demand rate and discrete opportunities for replenishment. *Production and inventory management*, 14(2):64–74, 1973. 21

R. S. Sutton and A. G. Barto. *Reinforcement Learning: an Introduction*. The MIT Press, Second edition, 2018. 7

P. Van Hentenryck and R. Bent. *Online Stochastic Combinatorial Optimization*. The MIT Press, 2006. doi:10.7551/mitpress/5140.001.0001. 7

P. Van Hentenryck, R. Bent, and E. Upfal. Online stochastic optimization under time constraints. *Annals of Operations Research*, 177(1):151–183, 2010. doi:10.1007/s10479-009-0605-5. 7

İ. Yanıkoğlu, B.L. Gorissen, and D. den Hertog. A survey of adjustable robust optimization. *European Journal of Operational Research*, 2018. doi:10.1016/j.ejor.2018.08.031. 11