Energy and AI 26 (2026) 100920

Contents lists available at ScienceDirect

Energy and AI

journal homepage: www.elsevier.com/locate/egyai

Review

S3TS: Stochastic Scenario-Structured Tree Search for advanced planning under uncertainty

Fabio Pavirani [a,*], Bert Claessens [b,a], Pierre Pinson [c], Chris Develder [a]

[a] IDLab, Ghent university – imec, Technologiepark Zwijnaarde 126, 9052 Gent, Belgium
[b] Beebop.ai, Belgium
[c] Imperial College London, London, UK

HIGHLIGHTS

- Proposes S3TS, a novel tree search framework with scenario-based uncertainty.
- Combines stochastic programming principles with Monte Carlo Tree Search.
- Handles non-linear dynamics and complex energy system models.
- Achieves near-optimal results in tractable stochastic problems.
- Improves robustness by reducing high-error outcomes under uncertainty.

ARTICLE INFO

Keywords:
Stochastic optimization
Monte Carlo Tree Search
Control theory
Multi-stage stochastic optimization
Tree search
Imbalance settlement
Grid balancing

ABSTRACT

Effective scheduling is essential to the reliable operation of electrical grids and their connected assets. An effective planning strategy must accommodate advanced, potentially non-linear system models while explicitly handling the uncertainties arising from the integration of renewable energy sources. Existing approaches can address either non-linearity (e.g., Monte Carlo Tree Search) or uncertainty (e.g., stochastic mathematical optimization), but rarely both simultaneously. To bridge this gap, we propose a Stochastic Scenario-Structured Tree Search (S3TS) algorithm that explicitly represents uncertainty through scenario trees while enabling the integration of advanced non-linear models. We evaluate S3TS on a simulated demand response signal publication problem, largely mimicking the imbalance settlement mechanism in Belgium. In linear, analytically tractable settings, its costs are within 14% of the mathematically optimal solution conditioned on the scenario trees. In highly non-linear scenarios, S3TS reduces cost by up to 51% and 5.3% compared to a myopic algorithm and deterministic MCTS, respectively, and consistently outperforms three online-sampling tree searches. Sensitivity studies over the tree configuration, the search parameters, and a misspecified scenario distribution confirm the robustness of S3TS.

1. Introduction

The growing complexity of electrical power systems, driven by the widespread electrification of energy sources, has made effective planning strategies essential for ensuring reliable grid operation. Accurate forecasting and strategic decision-making enable both grid operators and grid users to effectively optimize their objectives [1]. For instance, grid operators such as Transmission System Operators (TSOs) can forecast the grid load and imbalance, and plan balancing activations accordingly to favor grid stability. Similarly, grid users (such as Balance Responsible Parties (BRPs), residential, and commercial consumers) benefit from accurate forecasts of future energy prices, which allow them to adjust their consumption in line with expected grid conditions. This mechanism, also known as Demand Response (DR) [2], is significantly contributing to grid stability [3], and relies on effective planning by both operators and users.

A widely adopted approach for such planning tasks is mathematical optimization, particularly in the form of Model Predictive Control (MPC) [4,5]. MPC formulates sequential control tasks as optimization problems, aiming to minimize expected costs associated with controllable actions across the prediction horizon. This requires a dynamic model of the system to simulate state evolution and cost variations under different actions. Once an optimal solution has been found

* Corresponding author.
E-mail address: fabio.pavirani@ugent.be (F. Pavirani).

https://doi.org/10.1016/j.egyai.2026.100920
Received 29 June 2026; Received in revised form 14 September 2026; Accepted 27 September 2026
Available online 19 October 2026
2666-5468/© 2026 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

(usually through mathematical solvers), the resulting optimal actions are applied in a receding-horizon fashion, i.e., only the first action is executed before the optimization is repeated at the next time step.

Yet, as renewable energy sources become more prevalent, the resulting stochasticity in system dynamics poses a fundamental challenge to effective planning. In fact, if these uncertainties are not properly addressed, planning strategies can become fragile and unstable, potentially leading to substantial economic and operational losses [6,7]. To mitigate this, multi-stage stochastic optimization methods have been used [8]. These methods represent uncertainty through scenario trees constructed from sampled realizations of the underlying random variables. The scenario tree provides a structured representation of how uncertainty may evolve over time, with each branch corresponding to a possible future outcome. Decisions are then optimized across these scenarios using mathematical programming techniques. Stochastic MPC is a prominent example of this approach. Example of its application can be found for both grid operators [9,10] and grid users [11,12].

Despite their popularity, mathematical optimization methods face two major challenges. First, accurate modeling of system dynamics is critical for effective planning. As power systems grow more complex, increasingly advanced models with highly non-linear dynamics are typically required. Integrating such models into mathematical optimization frameworks is difficult due to their non-linear dynamics [7]. Second, evolving market designs introduce empirical rules and constraints that are hard to model analytically within conventional optimization techniques [13,14]. While simplified approximations can be used, they often fail to reflect the true complexity of modern power systems, resulting in suboptimal strategies.

These challenges thus call for planning techniques that can handle both non-linear models and complex market structures without sacrificing control robustness. Monte Carlo Tree Search (MCTS) has emerged as a promising alternative in the energy field, applied to problems such as residential heating [15,16], imbalance price publication [17], complex system operation and maintenance [18], and microgrid energy scheduling [19]; and is also extended in applications extending well beyond energy, to settings as diverse as safety-critical multi-agent coordination [20,21] and the search for symbolic solutions of differential equations [22,23]. MCTS explores the decision space by simulating multiple action sequences (trajectories) and selecting the most promising ones. Unlike mathematical optimization, MCTS imposes no structural constraints on the models beyond their ability to be simulated.

Standard MCTS, however, assumes deterministic transitions, which limits its applicability when uncertainty plays a dominant role. Several extensions have been proposed in the tree search literature to relax this assumption. The classical solution, originating in the game-tree community, is to interleave *decision nodes* with *chance nodes* in the search tree: at each chance node, an outcome is sampled from a (typically known) transition distribution, and the value backed up is the expectation over outcomes — e.g., in $*$-minimax [24] for stochastic games. Kearns et al. [25] formalized this idea for general MDPs through their sparse-sampling algorithm, which builds a finite-depth tree by drawing a fixed number of next-state samples per action, providing near-optimal value estimates with sample complexity independent of the state–space size. POMCP [26] extended chance-node MCTS to partially observable MDPs by sampling particles from the belief state at every simulation, while continuous-state stochastic dynamics are typically handled through double progressive widening [27], which gradually expands new outcome children as the visit count of a chance node grows. Although these methods cover a broad spectrum of stochastic settings, they share a common design choice: uncertainty is sampled online during simulation, drawing fresh outcomes from a transition oracle each time a chance node is visited. This couples the representation of uncertainty to the search procedure and presupposes cheap access to such an oracle, neither of which aligns naturally with the multi-stage stochastic programming tradition — where uncertainty is instead encoded upfront in a structured, finite scenario tree obtained through probabilistic forecasting and scenario reduction.

A more recent line of work tackles the deterministic-dynamics limitation by *learning* a stochastic transition model end-to-end. Building on MuZero [28], Antonoglou et al. [29] proposed Stochastic MuZero, where a variational autoencoder produces discrete latent outcomes that are then expanded as chance children during planning. While powerful in domains with abundant interaction data, this approach inherits two limitations relevant to our application: the latent transition model is opaque, and it is decoupled from the substantial body of work on scenario generation and reduction developed in stochastic programming [30–33], both of which are obstacles to adoption by real-world energy operators who require auditable forecasts. Our proposal sits at the intersection of these two traditions. Rather than sampling outcomes online from a transition oracle (as in chance-node MCTS, sparse sampling, or POMCP), or learning them implicitly via a latent model (as in Stochastic MuZero), we consider a *pre-built scenario tree* as input and use its topology to structure the search. Concretely, this differs from a chance-node MCTS along two axes: (i)the branching topology at every stochastic split is fixed externally by the scenario tree, hence no resampling occurs at visit time and the value estimates do not carry the variance of online Monte Carlo draws; and (ii)the probabilities attached to each branch come from a scenario reduction step (e.g., [30]) rather than from a sampling oracle, which makes the framework compatible with any probabilistic forecasting pipeline. A practical corollary is that the same scenario tree used for stochastic MPC is also used for our proposed tree search algorithm, enabling controlled comparisons that isolate the contribution of the optimization procedure from that of the uncertainty representation — a property we exploit in the experimental evaluation. The net effect is to bring the trustworthiness and modeling discipline of multi-stage stochastic programming into the flexibility of MCTS-style search.

Building on this positioning, we introduce *Stochastic Scenario-Structured Tree Search* (S3TS) and use it to bring the modeling discipline of multi-stage stochastic programming into MCTS-style search. The flexibility inherited from MCTS makes S3TS particularly attractive in settings where system dynamics are better captured through non-linear data-driven models — for instance, residential heating systems with neural-network-based thermal models [15,16] — or where the cost function contains components that hinder conventional optimization, such as discontinuous price ladders, min/max operators, or exponential terms combined with mixed-integer constraints. Anchoring the search to a pre-built scenario tree, in turn, aligns the resulting planner with established stochastic-programming pipelines: the same scenario generation, scenario reduction, and tree assembly steps already used to feed a stochastic MPC can directly drive S3TS, without modification. The key contributions of our work are:

1. We introduce S3TS (Section 3.4), an extension of MCTS that incorporates uncertainty through an externally constructed scenario tree rather than through online sampling from a transition oracle or a learned latent model. This makes the search topology, branch probabilities, and depth all explicit and inspectable, and exposes the same scenario-tree interface used by classical multi-stage stochastic optimization (as discussed in Section 3).
2. We formalize a Demand Response (DR) signal publication problem — inspired by the imbalance settlement mechanism currently in place in Belgium [34] — in both a linear and a non-linear formulation (Section 2), and solve the linearized version through Mixed-Integer Linear Programming to provide a reference optimum.
3. We empirically evaluate S3TS on this problem (Section 4). Our results (Section 5) show that in the linear formulation, S3TS achieves a total cost within 14% of the mathematically optimal solution obtained from the same scenario tree, demonstrating that the algorithm closely tracks the optimum even without

direct access to the analytical model. In the non-linear formulation — intractable for conventional solvers — S3TS reduces cost by 5.3% relative to deterministic MCTS, by 51% relative to a rule-based baseline, and consistently performs better than three online-sampling baselines, demonstrating the benefits of S3TS when jointly handling non-linear dynamics and explicit uncertainty. Moreover, we perform a comprehensive ablation study, analyzing the performance while varying algorithm parameters and a perturbed stochastic distribution, demonstrating the robustness of S3TS.

Section 6 summarizes our concluding remarks and directions for future work.

2. Problem description

To evaluate our S3TS algorithm, we apply it to a simulated DR market price (signal) publication problem. First, we evaluate S3TS in a simplified formulation of the problem (Section 2.1.1). Because this problem exhibits linear dynamics, the optimal stochastic solution can be computed analytically. This allows us to benchmark S3TS against an optimum provided by the true optimal solution, offering a clear measure of its efficiency. Then, we expand the problem by including highly non-linear dynamics (Section 2.1.2), making it analytically intractable for conventional mathematical optimization techniques. We use this second formulation to assess the S3TS performance when applied to more complicated environments. Since the optimal solution is then unknown, we compare S3TS against three families of baselines: (i)its deterministic counterpart, (ii)a range of online sampling tree search techniques, and (ii) a myopic rule-based strategy, highlighting the advantages of S3TS in complex, uncertainty-rich environments.

2.1. Price publication problem

We consider a problem formulation inspired by the imbalance settlement mechanism as operated in Belgium [17,34].[1] We assume a cooperative setting between two types of parties:

- The *system*, whose objective is to maintain its power balance.
- A group of *actors*, whose objective is to maximize their economic revenues.

The system health is described by its System Imbalance (SI) (i.e., the excess/lack of power in the system at a certain instant), and the system's goal would be to keep the SI magnitude as low as possible. While the actors can influence the state of the system by changing their energy production/consumption, their participation is subject to prices determined by the SI. These prices are computed at the end of each settlement period — each lasting $T \in \mathbb{N}^+$ timesteps (e.g., $T \doteq 15$ minutes in Belgium) — using a pricing formula based on the average SI observed during that period. The system benefits when actors adjust their actions to reduce SI; therefore, the price formula is designed to make helpful actors' influence (i.e., actors' actions that reduce the SI magnitude) profitable for them. However, since prices are only calculated after the settlement period, actors face uncertainty: their interventions might lead to financial losses if the final price does not match their expectations. In other words, if the SI significantly swings within a settlement period, the actors might struggle to understand which actions would be beneficial for the system; thus, potentially facing financial losses. This uncertainty discourages active participation. To incentivize actors, the system can publish an approximation of the final price *within* the settlement period. The more accurate this approximation is, the lower the risk for actors, making them more willing to participate and influence the SI with the goal of financial gains. This creates a mutual benefit: actors face less risk while the system improves its balance thanks to the more active actors' participation. Therefore, publishing accurate prices within the running settlement period is an important problem to enable effective cooperation between the two parties.

Analytically speaking, the problem of publishing accurate prices (signals) can be formulated as follows

$$\begin{aligned} \min_{\{\lambda_t\}_{t=0,1,\dots,T-1}} \quad & \sum_{\tau=0}^{T-1} \left\| \lambda_\tau - \hat{\lambda}_T \right\|_q \\ \text{s.t.} \quad & \hat{\lambda}_T = \hat{\lambda}(\hat{x}_T), \\ & x_t = g(x_{t-1}) + h(\lambda_{t-1}) && \forall\, t = 1, \dots, T \\ & g(x_{t-1}) = a\, y(t) + (1-b)\, x_{t-1} + c\, w_t && \forall\, t = 1, \dots, T \end{aligned} \tag{1}$$

where:

- λ_t is the published price (signal) at timestep t,
- $\hat{\lambda}_T$ is the real price that will be charged to the actors, as determined at the end of the settlement timeslot,
- $q \in \mathbb{R}^+$ defines the polynomial cost of the price inaccuracies,
- $\hat{\lambda}(\hat{x}_T)$ is the function used to calculate the final price based on the averaged measured SI,
- x_t is the SI measured at timestep t,
- $\hat{x}_T$ is the average SI measured in the timesteps from 0 to T,
- $g(x_{t-1})$ describes the general trend of the SI, and it comprises (i)a *seasonal progression* $y(t) : \mathbb{N} \to \mathbb{R}$ with relevance $a \in \mathbb{R}^+$, (ii)a *balancing factor* $b \in [0, 1]$, that describes the general trend of the SI to tend toward 0, and (iii)a *stochastic fluctuation* part w_t with a relevance factor $c \in \mathbb{R}^+$ and sampled from a random variable distribution $\mathbb{W}_t$. Notice that this function is independent of the published prices — i.e., this describes the SI progression if the actors do not manifest their influence on the system.
- $h(\lambda_t)$ is the influence that the actors manifest over the system based on the published price (signal). This function can potentially be highly non-linear, as it describes the strategy that the actors follow to maximize their revenues (e.g., rule-based, empirical strategies, RL-based strategies).

Notice that the addition of $h(\lambda_t)$ potentially makes the problem non-tractable for a conventional optimization algorithm, due to the dependence of the function on the controllable variable (λ_t). If the function has a highly non-linear structure, the overall problem becomes hardly solvable for a mathematical solver, hindering the final solution.

In the deterministic formulation defined above, the values of the stochastic variable w_t will be represented by the forecasted expected or median value of the distribution $\mathbb{W}_t$.

In its stochastic formulation, the problem would also have a scenario tree $\mathcal{T}$ associated, describing possible future SI fluctuation outcomes sampled from the distribution functions $\mathbb{W}_t$. The nodes of the scenario tree are indicated using the subindex notation and the symbol n. In this formulation, the decisions of the optimization problem happen on a node level, rather than on a time level. Therefore, the resulting optimization problem would be:

$$\begin{aligned} \min_{\{\lambda_n\}_{n \in \mathcal{T} \setminus \mathcal{L}}} \quad & \sum_{l \in \mathcal{L}} \mathbb{P}(l) \sum_{n' \prec l} \left\| \lambda_{n'} - \hat{\lambda}_l \right\|_q \\ \text{s.t.} \quad & \hat{\lambda}_l = \hat{\lambda}(\hat{x}_l), \\ & x_n = g(x_{\text{par}(n)}) + h(\lambda_{\text{par}(n)}) && \forall\, n \in \mathcal{T} \setminus \{0\} \\ & g(x_{\text{par}(n)}) = a y(t_n) + (1-b) x_{\text{par}(n)} + c w_n && \forall\, n \in \mathcal{T} \setminus \{0\} \end{aligned} \tag{2}$$

where $\mathcal{L}$ is the set of leaves in the scenario tree, par(n) indicates the parent node of n, $n' \prec l$ is used to indicate all the nodes n' preceding (having a parental relation with) l, 0 indicates the root node, and t_n is the timestep associated with the node n. Note that now a probability term $\mathbb{P}(l)$ has been added, indicating the probability of the considered branch (trajectory to the leaf node $l \in \mathcal{L}$). In this formulation, we assume the leaf nodes correspond temporally to the beginning of the next settlement period (and therefore to the end of the current one); consequently, their state determines the final imbalance price.

[1] Note that the mechanism dynamics we considered are not unique to the Belgian system, but are rather generalizable to other European countries too.

2.1.1. Linear formulation

To allow commercial mathematical solvers to solve the problems described in Eq. (1) and (2), we can define the functions $h : \mathbb{R} \to \mathbb{R}$ and $\hat{\lambda} : \mathbb{R} \to \mathbb{R}$ with a *'low grade of non-linearity'*; in other words, we can ensure that those functions will allow a computationally affordable mathematical solution of the optimization problem by using standard mathematical solvers. Doing so allows us to obtain a mathematical bound on the obtainable objective given a certain scenario tree as input. This mathematical bound will give a clear measure of the S3TS efficacy. To allow for such a solution, we first considered a linear problem formulation:

$$h(\lambda) = \begin{cases} L_r, & \text{if } \lambda < \frac{L_r - b_r}{a_r}, \\ a_r\lambda + b_r, & \text{if } \frac{L_r - b_r}{a_r} \le \lambda \le \frac{U_r - b_r}{a_r}, \\ U_r, & \text{if } \lambda > \frac{U_r - b_r}{a_r}. \end{cases} \tag{3}$$

$$\hat{\lambda}(x) = \begin{cases} a_\lambda x + b_\lambda + c_\lambda^+, & \text{if } x \le 0, \\ a_\lambda x + b_\lambda + c_\lambda^-, & \text{if } x > 0, \end{cases} \tag{4}$$

where $a_r, b_r \in \mathbb{R}$, $L_r \in \mathbb{R}$, and $U_r \in \mathbb{R}$ (with $L_r < U_r$ and $a_r > 0$) define a saturated linear response function. Further, $a_\lambda, b_\lambda, c_\lambda^+, c_\lambda^- \in \mathbb{R}$ (with $a_\lambda < 0$) define a linear price formula with a discontinuous jump when the SI sign changes. With these definitions, both optimization problems described in Eq. (1) and (2) are solvable with a standard mathematical solver such as Gurobi [35]. In this linear formulation, the polynomial cost term used in Eq. (1) and (2) is fixed to $q \doteq 1$ to make the optimization solvable with a Mixed-Integer Linear Programming algorithm.

2.1.2. Non-linear formulation

The linear formulation in Section 2.1.1 is intentionally simple: it admits a tractable MILP encoding and provides a yardstick for measuring the optimality gap of S3TS. It does not, however, reflect several characteristics of real imbalance settlement mechanisms — e.g., actual price functions are tabulated rather than affine, actors' responsiveness varies during the settlement period, and penalties on inaccurate publications are potentially polynomial rather than linear. To assess the benefits of S3TS over deterministic alternatives in a more realistic regime, we therefore introduce a second formulation that incorporates these features. The added non-linearities make the problem intractable for standard mathematical solvers, while remaining tractable for S3TS, which only requires $\hat{\lambda}$ and h to be evaluable in simulation.

We model $\hat{\lambda}(x)$ as a discontinuous step ladder, plotted in Fig. 1. The ladder is inspired by the structure currently in place in the Belgian imbalance settlement [34]: price magnitude grows super-linearly with $|x|$, with steep penalties at large imbalance, and the function exhibits an asymmetric discontinuity at $x = 0$ separating the small-positive-x regime (mildly negative price) from the small-negative-x regime (mildly positive price). The asymmetry around zero reflects the asymmetric costs of upward and downward regulation that are characteristic of real-world balancing markets.

The actors' influence $h(\lambda; t)$ is modeled as the product of a spatial shape and a temporal scaling, clipped to a bounded range:

$$h(\lambda; t) \doteq -\mathrm{clip}\Big(\varphi(t)\, \bar{h}(\lambda),\ L_r,\ U_r \Big), \tag{5}$$

where the spatial shape $\bar{h} : \mathbb{R} \to \mathbb{R}$ is the quadratic-saturated function

$$\bar{h}(\lambda) = \begin{cases} L_r, & \lambda < \lambda^{\mathrm{lo}}, \\ L_r \left(\frac{\lambda}{\lambda^{\mathrm{lo}}}\right)^2, & \lambda^{\mathrm{lo}} \le \lambda < 0, \\ U_r \left(\frac{\lambda}{\lambda^{\mathrm{hi}}}\right)^2, & 0 \le \lambda \le \lambda^{\mathrm{hi}}, \\ U_r, & \lambda > \lambda^{\mathrm{hi}}, \end{cases} \tag{6}$$

and the temporal factor $\varphi : \mathbb{N} \to \mathbb{R}^+$ is

$$\varphi(t) \doteq \left| \cos\big(2\pi\, (t \bmod T)/60\big) \right| + \tfrac{1}{2}. \tag{7}$$

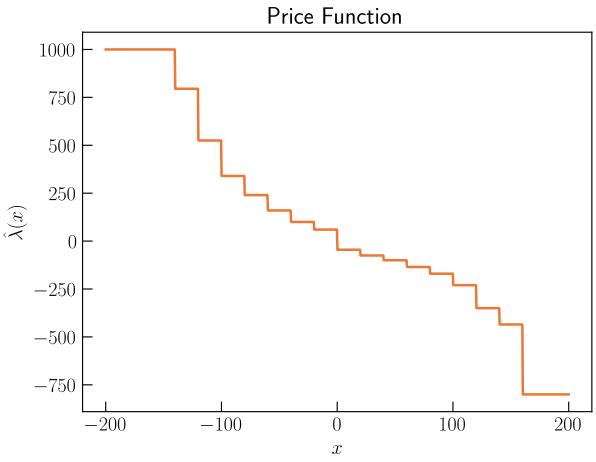


Fig. 1. Price function $\hat{\lambda}(x)$ used in the non-linear problem formulation.

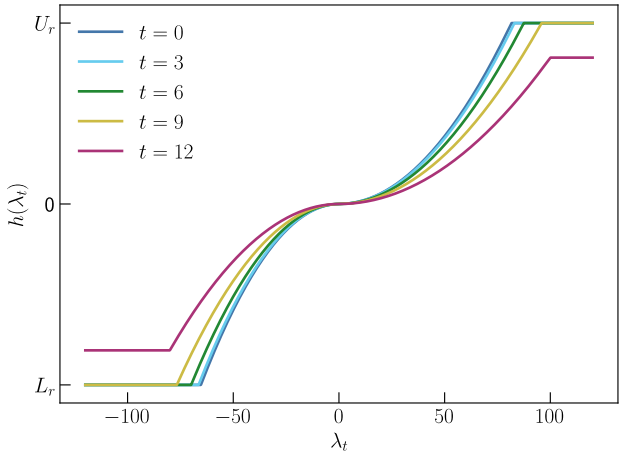


Fig. 2. Response function $h(\lambda; t)$ describing the actors' influence over the system in the non-linear problem formulation, plotted at five values of t within a settlement period. Curves at different t correspond to different temporal scalings $\varphi(t)$, leading to different effective steepness of the spatial shape.

Here $\lambda^{\mathrm{lo}} < 0 < \lambda^{\mathrm{hi}}$ are the price thresholds beyond which the actors' response saturates, and $U_r, L_r \in \mathbb{R}$ are the maximum and minimum response magnitude, respectively. The clip operator in Eq. (5) enforces the bound $L_r \le |h| \le U_r$ even when $\varphi(t) > 1$. The function is depicted at five values of t in Fig. 2.

We stress that these specific functional forms are illustrative; the S3TS framework can accommodate any pair $(\hat{\lambda}, h)$ that admits black-box simulation. More elaborate response models, such as the data-driven ones considered in [17,36], can be plugged in without modification of the algorithm. In this formulation, a quadratic cost exponent from Eq. (1) and (2) is also used ($q \doteq 2$), more aggressively penalizing large publication errors than the linear case.

The resulting problem combines three sources of analytical complexity: (i)the discontinuous price ladder $\hat{\lambda}$, which would require introducing one binary indicator per step at every leaf of the scenario tree (i.e., $\mathcal{O}(|\mathcal{L}|)$ binaries for each of the seventeen levels in Fig. 1); (ii)the non-convex quadratic-saturated response $\bar{h}$, which couples λ and x through equality constraints that are neither convex nor concave, and which interacts multiplicatively with the time-dependent factor $\varphi(t)$; and (iii)the quadratic publication cost. Together, these yield a non-convex Mixed-Integer Quadratically-Constrained Program. While such problems can in principle be encoded in modern solvers (e.g., Gurobi [35]), obtaining their global solution within a receding-horizon

time budget is generally infeasible. By contrast, S3TS evaluates $\hat{\lambda}$ and h only as black-box simulators along sampled trajectories, and remains therefore unaffected by their analytical complexity.

3. Methodology

3.1. Scenario tree construction

To solve multi-stage stochastic optimization problems such as the one in Eq. (2), we assume the existence of a scenario tree $\mathcal{T}$ that approximates the underlying random distribution influencing the problem dynamics. Scenario trees are required as inputs by both the stochastic MPC and the S3TS algorithm. We construct them in three sequential phases — scenario sampling, scenario reduction, and tree assembly — followed by a node-value refinement step, all described below. The full pipeline is illustrated in phases 1–3 of Fig. 3.

3.1.1. Scenario sampling

We first sample a finite number of scenarios (i.e., trajectories obtained through sequential realizations of the random variables governing the system) $\{\xi_m \in \mathbb{R}^{D\times H}\}_{m=1}^{N_\mathrm{o}}$ from the underlying stochastic distribution, where $N_\mathrm{o} \in \mathbb{N}^+$ is the number of sampled trajectories, $H \in \mathbb{N}^+$ is the prediction horizon, and $D \in \mathbb{N}^+$ is the dimension of each per-step realization ($D \doteq 1$ in the problem considered in Eq. (2), as the scenarios only entail the fluctuations of future SI scalar values). Each scenario ξ_m has an associated probability p_{ξ_m} such that $\sum_{m=1}^{N_\mathrm{o}} p_{\xi_m} = 1$. With uniform sampling, $p_{\xi_m} = \frac{1}{N_\mathrm{o}}$, $\forall m = 1, 2, \dots, N_\mathrm{o}$.

In our experiments, scenarios are drawn directly from the same stochastic process used to generate the test sequences. This is a deliberate simplification, intended to keep the focus on the control algorithms rather than on the quality of the distribution approximation. In a deployment setting, scenarios would instead be produced by a probabilistic forecasting model fitted on historical data — for instance, the technique described in [33], or any model capable of generating samples consistent with the conditional predictive distribution. The framework used by S3TS imposes no specific assumption on how scenarios are produced and is therefore agnostic to the choice of the underlying forecasting model.

3.1.2. Scenario reduction

Sampling a large number N_o of scenarios is desirable to faithfully approximate the underlying distribution. However, feeding all of them into a multi-stage optimization problem is computationally prohibitive: the number of decision variables in the resulting optimization grows linearly with the number of scenarios retained, and so does the size of the search tree explored by S3TS. To mitigate this, we reduce the original scenario set to a smaller representative subset of $N_\mathrm{r} \leq N_\mathrm{o}$ scenarios, using the algorithms described in [30,31]. These methods iteratively remove the least informative scenarios according to a Wasserstein-based distance metric on the trajectory space, and redistribute the probability mass of each removed scenario to its closest retained neighbor. The result is a reduced set $\{\xi_m\}_{m=1}^{N_\mathrm{r}}$ with updated probabilities still satisfying $\sum_{m=1}^{N_\mathrm{r}} p_{\xi_m} = 1$, and that approximates the original empirical distribution while remaining computationally tractable. The choice of N_r is a trade-off: a larger value provides a more faithful approximation but potentially yields a larger tree and therefore a higher computational cost in the downstream optimization.

3.1.3. Tree assembly

The reduced scenario set is then transformed into a scenario tree $\mathcal{T}$. Following the principles laid out in [32], we adopt a path-consistent agglomerative procedure based on stage-wise clustering on path prefixes, summarized in Alg. 1.

The procedure starts from a single root node containing all reduced scenarios, and progressively partitions them stage by stage. At each stage $t = 0, 1, \dots, H-1$, every existing tree node — corresponding to a subset μ_n of scenarios — is split into at most β_t children by clustering its scenarios according to their realizations up to (and including) time t.

In practice, we use the K-Means algorithm [37] to perform clustering at each node, with the requested β_t as the number of clusters. The branching schedule $\{\beta_t\}_{t=0}^{H-1}$ is a hyperparameter of the construction. In our experiments, we use $\beta_0 = 1$, $\beta_1 = 3$, $\beta_2 = 3$, and $\beta_t = 1$ for $t \geq 3$, concentrating the branching in the early stages where the impact on the immediate decision is largest. This is a common heuristic in stochastic programming to control the otherwise exponential growth of the number of leaves while preserving the most relevant uncertainty splits. A node with $\beta_t = 1$ effectively continues each scenario subset along a single branch, modeling the simplification that, once early-stage uncertainty has been resolved, residual uncertainty in later stages is approximated as a single conditional trajectory.

Algorithm 1 Scenario Tree Construction.

Require: Reduced scenarios $\{\xi_m\}_{m=1}^{N_\mathrm{r}}$, branching schedule $\{\beta_t\}_{t=0}^{H-1}$
Ensure: Scenario tree $\mathcal{T}$
  Initialize root node n_0 with $\mu_{n_0} \leftarrow \{1, \dots, N_\mathrm{r}\}$
  $\mathcal{N}_\mathrm{current} \leftarrow \{n_0\}$
  for $t = 0, 1, \dots, H-1$ **do**
    $\mathcal{N}_\mathrm{next} \leftarrow \emptyset$
    for each node $n \in \mathcal{N}_\mathrm{current}$ **do**
      $k \leftarrow \min\left(\beta_t,\ |\mu_n|,\ \left|\{\xi_m[t] :\ m \in \mu_n\}\right|\right)$
      $X \leftarrow \{\xi_m[0{:}t+1] : m \in \mu_n\}$
      $\{C_1, \dots, C_k\} \leftarrow$ K-Means(X, k)
      for each cluster C_j **do**
        Create child node n' with $\mu_{n'} \leftarrow C_j$
        Attach n' as child of n in $\mathcal{T}$
        $\mathcal{N}_\mathrm{next} \leftarrow \mathcal{N}_\mathrm{next} \cup \{n'\}$
      end for
    end for
    $\mathcal{N}_\mathrm{current} \leftarrow \mathcal{N}_\mathrm{next}$
  end for
  Set $p_n \leftarrow \sum_{\xi \in \mu_n} p_\xi$ for every $n \in \mathcal{T}$
  return $\mathcal{T}$

3.1.4. Node-value refinement

Once the tree topology is fixed, each node n is assigned a representative realization $\bar{\xi}_n \in \mathbb{R}^D$, computed as the probability-weighted conditional mean of the scenarios assigned to that node (indicated with μ_n), evaluated at the node's time index t_n:

$$\bar{\xi}_n \doteq \frac{\sum_{m\in\mu_n} p_{\xi_m}\, \xi_m[t_n]}{\sum_{m\in\mu_n} p_{\xi_m}}. \tag{8}$$

The trajectories along each root-to-leaf path are then redefined as the sequence of these conditional means, replacing the original raw scenarios. The refinement is what makes the node-level realization $\bar{\xi}_n$ well-defined as a property of the node alone. After the refinement, all scenarios sharing a path prefix in the tree share the same realizations along that prefix, which is the non-anticipativity condition required for the transition and reward functions as further discussed in Section 3.2. The choice of the conditional mean as the representative is, in turn, the natural one under a quadratic distance.

3.1.5. Receding-horizon tree construction

Both the stochastic MPC and the S3TS algorithm operate in a receding-horizon fashion: at every timestep t within the settlement period, a new tree is constructed using only the residual horizon $H - t$ (forecasting the SI beyond the end of the settlement period does not help the optimization, as the objective function optimizes over each settlement period independently). For a settlement period of length T, this yields T separate scenario trees per period, each with a progressively shorter horizon as time advances and uncertainty about the

past is resolved by observation. Sampling, reduction, and assembly are repeated independently at each step, conditioned on the latest observed state.

For consistency, we use exactly the same scenario trees for stochastic MPC and S3TS within a given experiment, and we share them across techniques whenever applicable. This guarantees that any performance gap between the two methods is attributable to the optimization procedure itself rather than to differences in how uncertainty is represented.

3.1.6. Notation summary

For the description of the S3TS algorithm in the remainder of the paper, we collect the notation introduced above. We indicate the reduced scenario set used to build the tree as $\vec{\xi} \doteq \{\xi_m\}_{m=1}^{N_\Gamma}$. Consistently with Eq. (2), we denote the nodes of the tree by $n \in \mathcal{T}$. Each node has an associated scenario subset $\mu_n \in \mathcal{P}(\vec{\xi})$, a node-level realization $\bar{\xi}_n$ obtained from Eq. (8), and a probability value $p_n \doteq \sum_{\xi \in \mu_n} p_\xi$. Finally, we describe the dynamics of the parental relations between nodes through a *scenario map* $\$: \mathcal{T} \to \mathcal{P}(\vec{\xi})$, which describes how the scenarios' uncertainty evolves through the tree by indicating, for each node, the partition induced on its scenario subset by its children.

3.2. MDP definition

Before describing the S3TS algorithm, we need to formalize the mathematical framework it works with, i.e., a Markovian Decision Process (MDP) [38]. An MDP is a sequential decision process model typically used with Reinforcement Learning (RL) algorithms, including the MCTS algorithm we expanded to build S3TS. Fixing a scenario ξ (i.e., SI trajectory), we obtain a *deterministic*[2] MDP defined as the tuple $(\mathcal{S}, \mathcal{A}, f_\xi, \rho_\xi)$, where

- $\mathcal{S}$ is the *State Space*,
- $\mathcal{A}$ is the *Action Space*,
- $f_\xi : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the *Transition Function*,
- $\rho_\xi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the *Reward function*.

By considering a set of scenarios $\vec{\xi}$, we obtain an approximation of the underlying stochastic process in the form of a set of deterministic MDPs $\{(\mathcal{S}, \mathcal{A}, f_\xi, \rho_\xi) \ ; \ \forall \xi \in \vec{\xi}\}$. Given a discount factor $\gamma \in [0, 1]$, the corresponding objective would then become to find the optimal policy $\pi^* : \mathcal{S} \to \mathcal{A}$ such that:

$$\pi^*(s) \in \arg\max_{a \in \mathcal{A}} Q^*(s, a) \ ; \ \forall s \in \mathcal{S}. \tag{9}$$

where:

$$Q^*(s, a) \doteq \max_\pi Q^\pi(s, a), \tag{10}$$

$$\begin{aligned} Q^\pi(s, a) &\doteq \mathbb{E}_{\xi \sim \vec{\xi}}\left[\rho_\xi(s, a) + \gamma R^\pi\left(f_\xi(s, a)\right)\right] \\ &= \sum_{\xi \in \vec{\xi}} p_\xi\left[\rho_\xi(s, a) + \gamma R^\pi_\xi\left(f_\xi(s, a)\right)\right], \end{aligned} \tag{11}$$

$$\begin{aligned} R^\pi(s_0) &\doteq \lim_{T \to +\infty} \mathbb{E}_{\xi \sim \vec{\xi}}\left[\sum_{t=0}^{T} \gamma^t \rho_\xi\left(s_t, \pi(s_t)\right)\right] \\ &= \lim_{T \to +\infty} \sum_{\xi \in \vec{\xi}}\left[p_\xi \sum_{t=0}^{T} \gamma^t \rho_\xi\left(s_t, \pi(s_t)\right)\right], \end{aligned} \tag{12}$$

$$R^\pi_\xi(s_0) \doteq \lim_{T \to +\infty} \sum_{t=0}^{T} \gamma^t \rho_\xi(s_t, \pi(s_t)) \tag{13}$$

$$\text{with} \quad s_{t+1} = f_\xi(s_t, a), \ \forall t \in \mathbb{N}$$

[2] By fixing a scenario as in the definition in Section 3.1, the process becomes deterministic as there is no uncertainty involved in the future values.

Notice that the optimal policy π^* as we just defined is not equivalent to the optimal policy of the stochastic process we wish to solve. Instead, π^* is only the optimal policy for the aggregated set of MDPs $\{(\mathcal{S}, \mathcal{A}, f_\xi, \rho_\xi) \ ; \ \forall \xi \in \vec{\xi}\}$, which is on its own an approximation of the actual underlying process we wish to solve. The quality of the policy π^* will then also depend on the quality of such an approximation; in other words, it will depend on how accurate the scenario tree used is in relation to the real random distribution steering the values of the stochastic variables involved.

As discussed in Section 3.1, $\vec{\xi}$ has been sampled and reorganized to build a scenario tree $\mathcal{T}$. To ensure that the just-defined optimization problem is consistent with the non-anticipativity property given by the scenario tree, we then impose the following conditions. Given a node in the scenario tree $n \in \mathcal{T}$, and a pair of state and action $s \in \mathcal{S}$, $a \in \mathcal{A}$, we have:

- $f_\xi(s, a) = f_{\xi'}(s, a) \ ; \ \forall \xi, \xi' \in \mu_n$
- $\rho_\xi(s, a) = \rho_{\xi'}(s, a) \ ; \ \forall \xi, \xi' \in \mu_n$

For simplicity, we will indicate the corresponding functions with f_n and ρ_n.

In the price publication problem defined in Section 2, the states of the MDP would indicate the general condition of the system (e.g., SI, time), the actions would be the published price (λ), the transition function would be represented by the functions $g(x), h(\lambda)$, and the reward function would entail the objective functions defined in Eq. (1) and (2). Last, a definition of intermediate ρ is required for the tree-based methods. In fact, the optimization cost in Eq. (1) and (2) is defined at the end of the settlement period, once the final price $\hat{\lambda}_T$ is known. Tree search methods, however, backpropagate values through the search tree at every step, and therefore require an intermediate reward ρ_n at each node. We define this intermediate reward at timestep t in the settlement period as the negative mean deviation between the prices published so far in the period and the running estimate of the final price obtained from the current cumulative average SI:

$$\rho_t \doteq -\frac{1}{t+1} \sum_{\tau=0}^{t} \left|\lambda_\tau - \hat{\lambda}(\hat{x}_{t+1})\right|^q. \tag{14}$$

At $t = T - 1$ (i.e., the last publication step) the rolling estimate $\hat{\lambda}(\hat{x}_{t+1})$ coincides with $\hat{\lambda}_T$ and the sum of the terms in Eq. (14) recovers the total period cost in Eq. (1). This shaping is needed only for the tree search methods, since the mathematical-programming techniques operate directly on the end-of-period cost and do not require an intermediate signal.

3.3. Monte Carlo tree search

The MCTS algorithm models the MDP using a search tree structure, mapping the states to nodes and the actions to edges. The algorithm is a sequential repetition of four phases, roughly described as follows:

1. **Selection:** Starting from a root node s_0, iteratively traverse the tree by reaching a child node (i.e., a system state) $s_k \ ; \ k = 1, \dots, \ell$ by selecting an edge (action) $a_k \ ; \ k = 0, \dots, \ell - 1$ until a leaf node s_ℓ is reached. Each selection is made by balancing exploitation (selecting the actions that proved to be the most optimal in the previous iterations) and exploration (selecting the actions that have not been selected enough before, with the goal to find more profitable trajectories compared to the ones explored so far).
2. **Expansion:** Expand the tree by adding new children from the selected leaf node s_ℓ, i.e., roll out possible actions from that state.
3. **Simulation:** Evaluate the node value by performing Monte Carlo simulations.

4. **Backpropagation:** Propagate the information acquired (i.e., the rewards obtained in the trajectory) back to the root node. This information will update the value estimation of each node in the selected trajectory. These value estimations will then be used in the selection phase to indicate how promising actions are in the tree.

The execution of the four phases constitutes an iteration (or simulation). The more iterations are executed, the more accurate the tree search will become, improving the control efficacy of the policy. Notice that the nodes and edges of the search tree are identified solely by a depth value $k \in \mathbb{N}$. In general, a tree may contain multiple nodes and edges at the same depth. However, within the framework of the MCTS algorithm, such a finer distinction is unnecessary. During a single iteration, exactly one node and one edge are selected at each depth level. Therefore, referring to nodes and edges only by their depth is sufficient and does not introduce ambiguity for the purposes of the algorithm.

Algorithm 2 S3TS Algorithm.

procedure SIMULATION_ITER(s_k^μ)
  if s_k^μ is not a leaf **then**
    Select action a_k^μ ▷ Based on Eq. (15)
    Get n_k^μ from s_k^μ, a_k^μ ▷ Corresponding scenario tree node
    `v_list` ← [ ]
    for μ' in (n_k^μ) **do**
      $s_{k+1}^{\mu'} \leftarrow f_{n_k^{\mu'}}(s_k^\mu, a_k^\mu)$ ▷ Transition function
      $r_k^{\mu'} \leftarrow \rho_{n_k^{\mu'}}(s_k^\mu, a_k^\mu)$ ▷ Reward function
      Append $\left(\text{SIMULATION_ITER}(s_{k+1}^{\mu'}) + r_k^{\mu'},\ \mu',\ \mu\right)$ to `v_list`
    end for
    v ← AGGREGATE(`v_list`)
    Update $\overline{Q}_{\text{tree}}\left(s_k^\mu, a_k^\mu\right)$ ▷ Based on Eq. (16)
  else
    EXPAND(s_k^μ)
    v ← 0
  end if
  return v
end procedure

procedure AGGREGATE(`v_list`)
  `v_total` ← 0
  for (v, μ', μ) in `v_list` **do**
    `v_total` ← `v_total` + v $\cdot \frac{\mathbb{P}(\mu')}{\mathbb{P}(\mu)}$
  end for
  return `v_total`
end procedure

procedure EXPAND(s_k^μ)
  for a in $\mathcal{A}$ **do**
    Get n_k^μ from s_k^μ, a ▷ Corresponding scenario tree node
    for μ' in (n_k^μ) **do**
      Add $s_{k+1}^{\mu'}$ into the tree
    end for
  end for
end procedure

procedure STOCHASTIC_TREE_SEARCH($s_0^{\vec{\xi}}$)
  while computational budget is available **do**
    SIMULATION_ITER($s_0^{\vec{\xi}}$)
  end while
  return $\arg\max_{a\in\mathcal{A}} \left[Q_{\text{tree}}\left(s_0^{\vec{\xi}}, a\right)\right]$ ▷ Best action
end procedure

In our experiments, the Selection phase uses the following equation based on the UCT formula [39]:

$$a_k = \arg\max_{a\in\mathcal{A}^k} \left\{ \overline{Q}_{\text{tree}}(s_k, a) + \alpha \frac{\sqrt{N(s_k)}}{1 + N(s_k, a)} \right\}, \tag{15}$$

where $N(s) \in \mathbb{N}$ is the number of visits received by node s (i.e., the number of time the nodes has been selected before); $N(s,a) \in \mathbb{N}$ is the number of times action a has been selected from state s; $\alpha \in \mathbb{R}^+$ is a hyperparameter used to balance exploration and exploitation; and $\overline{Q}_{\text{tree}}(s,a)$ is the normalized version of a state–action value function approximation that is tuned in the backpropagation phase as follows:

$$Q_{\text{tree}}(s_k, a_k) = \frac{N(s_k, a_k)\, Q_{\text{tree}}(s_k, a_k) + G^{k+1}}{N(s_k, a_k) + 1}, \quad \forall k = \ell - 1, \dots, 0 \tag{16}$$

where $Q_{\text{tree}}(s_k, a_k)$ is initialized as $\rho(s_k, a_k)$ and G^k is the discounted accumulated reward in the tree branch defined as:

$$G^k = \begin{cases} \rho(s_{\ell-1}, a_{\ell-1}) & \text{if } k = \ell \\ \rho(s_{k-1}, a_{k-1}) + \gamma\, G^{k+1} & \text{if } k = \ell - 1, \dots, 1 \end{cases}; \tag{17}$$

with $\gamma \in [0,1]$ the discount factor. The normalized value $\overline{Q}_{\text{tree}}(s,a)$ is obtained using min/max values in the tree:

$$\overline{Q}_{\text{tree}}(s,a) = \frac{Q(s,a) - Q^{\min}}{Q^{\max} - Q^{\min}} \in [0,1], \tag{18}$$

where

$$\begin{cases} Q^{\min} \doteq \min_{s',a' \in \text{Tree}} Q_{\text{tree}}(s', a') \\ Q^{\max} \doteq \max_{s',a' \in \text{Tree}} Q_{\text{tree}}(s', a') \end{cases}. \tag{19}$$

3.4. Stochastic scenario-structured tree search (S3TS)

The standard MCTS algorithm was developed with the goal of working with deterministic MDPs. That is because it is expected that, given a certain state (node), an action (edge) would deterministically and uniquely lead to a defined next state. To allow the algorithm to handle uncertainties in the problem, we extended the MCTS algorithm described above by integrating scenario trees in the tree search. To do so, we introduce the distinction between state nodes (i.e., the nodes mapped to a state of the control problem, as used in standard MCTS) and *scenario nodes* (i.e., nodes that are mapped to the nodes of a scenario tree); and action edges (i.e., the edges mapped to an action of the control problem, as used in standard MCTS) with *scenario edges* (i.e., the edges that are mapped to the edges of a scenario tree). Roughly speaking, we naturally integrate a scenario tree in a search tree by adding its corresponding scenario nodes and edges between the state nodes. In particular, each action edge will now go from a state node (the state in which the control action takes place) to a scenario node mapped to the corresponding node in the scenario tree. Then, the scenario node will solve the uncertainty dynamics of the scenario tree by having multiple scenario edges ending at a state node, each corresponding to a specific child of the scenario tree node. Note that, differently from the deterministic version, the depth k is now not sufficient to indicate each selected node and action. Instead, it is now necessary to also indicate the corresponding subset of scenarios $\mu \in \mathcal{P}(\vec{\xi})$ that the node is associated with (based on the scenario tree structure). For this reason, state nodes and action edges are now indicated with the notation s_k^μ, a_k^μ, respectively. Analogously, we indicate with n_k^μ the corresponding node in the scenario tree and with r_k^μ the corresponding reward obtained.

At a high level, the algorithm follows the same four-phase loop as standard MCTS — selection, expansion, simulation, and backpropagation — but treats stochastic transitions explicitly. Starting from the root state node, the algorithm uses the UCT criterion of Eq. (15) to select an action; this action leads deterministically to a scenario node mapped

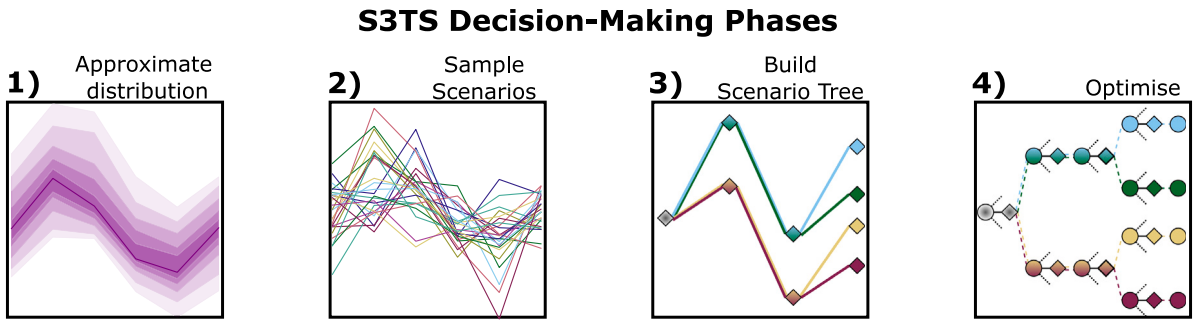


Fig. 3. Four phases repeated at each timestep to use the S3TS algorithm. First, the random distribution underneath the stochastic process gets approximated (phase 1). Then, using the approximated distribution, a certain number of scenarios are sampled from it (phase 2). The scenarios are aggregated in a scenario tree (phase 3) that is then optimized using the S3TS algorithm (phase 4), obtaining the optimal action from it. Notice that the structure is very similar to the generic stochastic MPC one, where the first three phases are equivalent, and phase 4 can be changed with other optimization algorithms (e.g., mathematical optimization).

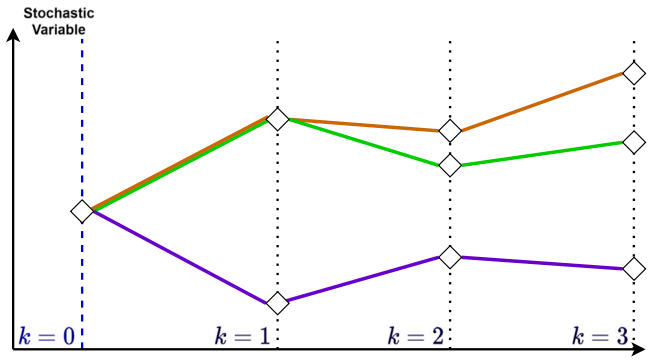


Fig. 4. Example of a scenario tree with horizon 3.

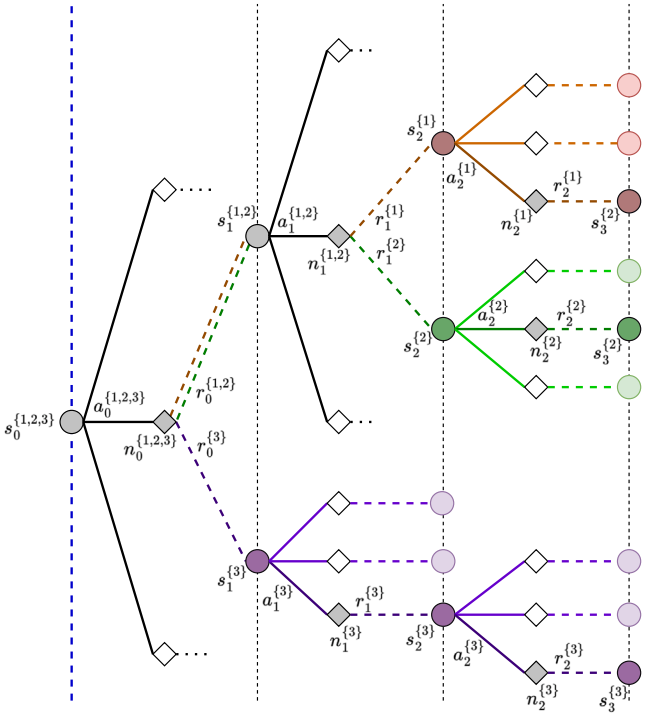


Fig. 5. Notation of the S3TS framework; using $\{1,2,3\}$ as the scenario space, with 1 representing the orange scenario of Fig. 4, 2 the green one, and 3 the purple one. Darker nodes indicate the selected trajectories in the current tree search iteration.

onto the corresponding node of the scenario tree. From there, the scenario node resolves the stochastic transition by branching into all of its scenario children, and the algorithm recurses into each child state node. The values returned by the recursive calls are aggregated into a single probability-weighted estimate, which is backpropagated up the tree to update the Q_{tree} value of the selected action via Eq. (16). When the recursion reaches a state node that has not yet been expanded, the algorithm grows the tree at that point. For each available action, it adds a scenario node and, for each child of that scenario node, a new state node. The current simulation then terminates without recursing further; subsequent simulations that reach this point will be able to descend through the newly added children. This loop is repeated until the computational budget available for each action is exhausted, at which point the action with the highest Q_{tree} value at the root is returned. An example of a search tree integrated with a scenario tree is shown in Fig. 5 corresponding to the scenario tree in Fig. 4. The full algorithm is described in Alg. 2.[3] The phases iterated upon to use the S3TS algorithm are depicted in Fig. 3.

4. Experimental setup

4.1. Data generation

To evaluate the algorithms on the problems described in Eq. (1) and (2), we considered a synthetic stochastic process $\mathbb{W}$ governing the SI fluctuations of g. We then generated a timeseries of datapoints sequentially sampled from g and divided it into 1000 independent settlement periods of $T \doteq 15$ timesteps of 1 min each. Each period is approached independently by every algorithm to produce a sequence of published prices $(\lambda_t\ ;\ \forall t = 0, 1, \dots, T)$. A graphical illustration of the underlying random distribution is shown in Fig. 6, with probabilistic quantiles obtained through Monte Carlo sampling. The process is defined as a first-order autoregressive stochastic system, following the structure used in Eq. (1) and (2). The seasonal progression component $y(t)$ follows a sinusoidal profile with period 15 (corresponding to T). The balancing factor is fixed to $b \doteq 0.5$. The stochastic fluctuations w are modeled as time-dependent normal noise using a cosine modulation. More precisely, the process can be written as

$$x_{t+1} = 15 \sin\left(\frac{2\pi t}{15}\right) + (1-b)x_t + w_t, \tag{20}$$

where

$$w_t \sim \mathcal{N}\left(0, \sigma_t^2\right), \qquad \sigma_t = 20 + 5\cos\left(\frac{2\pi(t \bmod 15)}{60}\right). \tag{21}$$

[3] The implementation code is available at https://github.com/loud-mime/S3TS/tree/main.

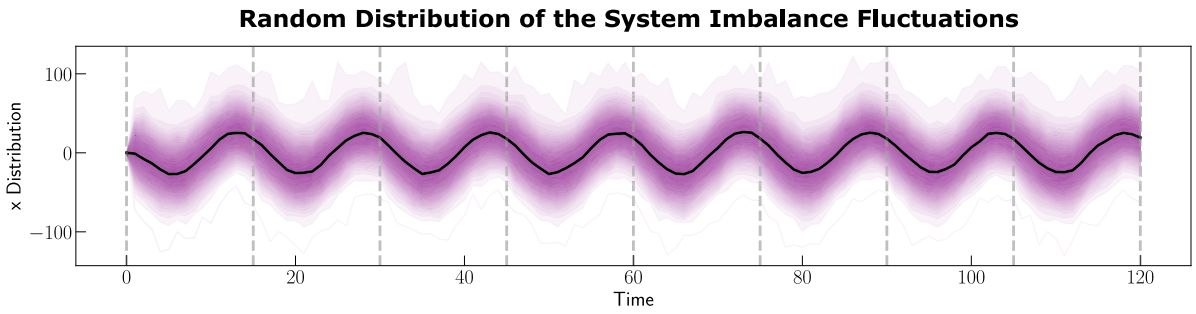


Fig. 6. Visualization of the random distribution used to generate the SI fluctuations in our experiments. The distribution is approximated through Monte Carlo sampling, and different quantiles are visualized in the figure.

To build the scenario trees used by the stochastic methods (Stochastic MPC and S3TS) when solving Eq. (2), at every timestep within each settlement period we drew $N_o = 10000$ SI trajectories from the underlying distribution conditional on the latest observed state, reduced them to $N_r = 100$ representatives following Section 3.1.2, and assembled the resulting scenario tree using the path-consistent procedure of Alg. 1. For consistency, Stochastic MPC and S3TS share exactly the same scenario tree at every receding-horizon step within an experiment — this guarantees that any performance gap between the two methods is attributable to the optimization procedure itself rather than to differences in uncertainty representation. The deterministic-equivalent baselines are obtained by feeding the very same code path with degenerate trees containing a single trajectory (the element-wise median of the original scenario set). All techniques operate in a receding horizon, i.e., in every timestep the forecasts are regenerated, then the problem is solved with the latest information, and only the upcoming action is applied before the iteration repeats.

4.2. Baselines

The evaluation compares nine techniques that span four pairings of interest: (i)tree search against mathematical optimization, (ii)stochastic against deterministic uncertainty handling, (iii)pre-built scenario trees against online outcome sampling, and (iv)S3TS against a rule-based baseline. To avoid implementation artifacts confounding the comparison, every algorithm variant runs through the same code path and shares the same input scenario trees whenever applicable. The baselines based on mathematical solvers (i.e., Perfect Knowledge and the Stochastic and Deterministic MPCs) are reported only in the linear regime experiment's results, since the non-linear regime is intractable for MILP; conversely, the three online-sampling baselines are reported only in the non-linear regimes, where the pre-built scenario tree is the sole remaining source of uncertainty representation.

- **Rule-Based.** A naive heuristic that publishes the price obtained by applying $\hat{\lambda}$ to the cumulative average SI observed so far in the current settlement period. This baseline carries no forecasting and no optimization, but it is usually applied in practice, and it serves to quantify the value of advanced planning.
- **Perfect Knowledge.** An MILP solution to Eq. (1) computed under the assumption of perfect foresight — i.e., with the realized SI trajectory of the period revealed in advance. This is a clairvoyant oracle, not a stochastic optimum, i.e., it provides an unattainable optimum on the theoretical best policy obtainable.
- **Stochastic MPC.** An MILP solution of Eq. (2) given the input scenario tree. This is the optimum of the stochastic problem *conditional on the input tree*, and is the most informative reference point for assessing S3TS: any gap between Stochastic MPC and S3TS measures how well S3TS approximates the optimum of the same multi-stage stochastic problem the two methods share.
- **Deterministic MPC.** The MILP solution of Eq. (1) fed with the elementwise median of the sampled scenarios as a point forecast. This isolates the cost of replacing the scenario tree with a single representative trajectory, while keeping the optimization procedure (MILP) unchanged.
- **S3TS.** The algorithm proposed in this work, instantiated as in Alg. 2.
- **MCTS.** The deterministic counterpart to S3TS, obtained by running Alg. 2 on a singleton scenario tree whose unique trajectory is the element-wise median of the original scenario set. This baseline shares its implementation, action space, expansion policy, UCT exploration constant, and value-update rule with S3TS; the only difference is the input scenario tree. As a consequence, any performance gap between this baseline and S3TS is attributable solely to the explicit handling of uncertainty in the planning framework, with all other algorithmic factors kept unchanged.
- **Sparse Sampling.** A stochastic counterpart to S3TS in which the pre-built scenario tree is replaced by $C \in \mathbb{N}^+$ outcomes drawn online from the transition process at every stochastic split, following [25].[4] The outcomes are sampled from the same process that generated the S3TS scenarios and conditioned on the same observed state. Hence, any performance gap with ST3S is attributable solely to whether the uncertainty is represented by a pre-built structured tree or by online draws.
- **β-Matched Sampling.** Sparse Sampling constrained to the branching schedule of the published scenario tree, instead of branching C ways at every stage. Since a tree branching C^d wide cannot descend as deep within a fixed wall-clock budget, this method controls how wide the tree is allowed to grow (i.e., how many online samples are considered), ensuring the same number of scenario splits in the same positions as S3TS.
- **Quantile Sampling.** This follows Sparse Sampling, but places the outcomes at the conditional quantiles of the transition process and rescales them to reproduce its variance, rather than drawing them independently. This implies the use of evenly spread representative points, which is conceptually similar to the scenario reduction methods that we used for S3TS.

The tree search techniques (S3TS, MCTS, and the three sampling baselines) require a discrete action space; we adopt the same construction as in [17]. The action space is recomputed at every timestep based on the current SI estimate by considering the price associated with the current SI using the corresponding formula ($\hat{\lambda}$) and adding equidistant actions by adding and subtracting a fixed amount to obtain a certain fixed-sized action space.

[4] The value of C has been selected with a parameter search, obtaining the lowest error with $C = 9$ for $q = 1$ and $C = 12$ for $q = 2$.

4.3. Computational budget protocol

A core methodological aim of our evaluation is to disentangle the contribution of S3TS from the contribution of the computational budget involved. Tree search methods can, in principle, be made arbitrarily accurate by running additional simulations. We therefore adopt a matched-budget protocol: every technique is given the same per-publication wall-clock budget, and we report results across a grid of budgets to expose how performance scales with available compute.

Concretely, we run each experiment at four budgets: $B \in \{1, 2, 4, 8\}$ seconds per published price. The budget is enforced as wall-clock time inside the inner solve loop: the loop checks elapsed time after every completed simulation and exits as soon as the budget is exceeded. The budget is applied uniformly to all techniques. The Rule-Based heuristic terminates in microseconds and is therefore always well below budget; the MILP solves (Perfect Knowledge, Stochastic MPC, Deterministic MPC) typically reach close-to-optimal results within a fraction of a second on the linear formulation, and consequently also finish below budget and are not affected by our variations of B.

The specific hardware specifics used in our experiments are listed in detail in Appendix.

4.4. Validation metrics

We evaluate each technique using two cost metrics that mirror the cost function of Eq. (1) and (2): the *Mean Absolute Error* (MAE) corresponds to $q \doteq 1$, and the *Mean Squared Error* (MSE) to $q \doteq 2$, both computed from the published prices (λ_t) and the realized price $\hat{\lambda}_T$ difference at the end of the settlement period. Since q enters the objective each technique optimizes, we first report three experiments: the linear regime under $q \doteq 1$ (MAE) — the only setting admitting a tractable MILP encoding thanks to the linear dynamics and therefore the only one we can benchmark against mathematical optimization (i.e., the MPCs and Perfect Knowledge baselines); the non-linear regime under $q \doteq 1$ (MAE), which isolates the effect of the non-linear dynamics while keeping the cost exponent fixed; and the non-linear regime under $q \doteq 2$ (MSE), matching a quadratic-cost setting. We also performed comprehensive ablation and sensitivity analyses in the non-linear dynamics case. Each metric is averaged over the 1000 independent settlement periods. Together with each mean, we also report the corresponding interquartile (Q1–Q3) interval taken over the same 1000 periods.

5. Results

5.1. Linear model results

Table 1 reports the MAE achieved by each technique on the linear regime with $q = 1$ across the four budget settings. At 8 s, S3TS achieves 26.03 MAE against Stochastic MPC's 22.90, placing it within 13.7% of the optimum of the multi-stage stochastic problem conditioned to the same input scenario tree; indicating S3TS to be a valuable control algorithm. The gap narrows with budget, from 15.9% at 1 s to 13.7%, indicating that part of the distance to the optimum can also be attributed to the simulation budget rather than to a structural limitation of the algorithm (e.g., discretization of the action space).

A separate and complementary observation concerns the two deterministic-equivalent techniques: Deterministic MPC and MCTS consume the same median scenario as input but differ in their solver, and the tree search method wins by a substantial margin (MAE of 27.62 vs. 32.86 at 8 s, a 15.9% advantage that is consistent across budgets). This is initially counter-intuitive: MPC solves the deterministic problem to optimality given the median forecast, whereas MCTS is a sample-based approximation working within a finite simulation budget — one might a priori expect MPC to do at least as well. Our intuition is that the MPC's optimality is conditional on the forecast being correct across the entire settlement window, i.e., it commits to a full sequence of decisions based on the whole median trajectory, propagating any forecast error all the way to the cost incurred at the end of the period. Because the SI random process accumulates the uncertainties later in the prediction horizon, the median trajectory is a particularly poor predictor of the late-period dynamics that ultimately determine $\hat{\lambda}_T$. MCTS, by contrast, is implicitly horizon-limited: a limited simulation budget gives each rollout only finite depth, so the algorithm weights near-term consequences more heavily than far-future ones when selecting actions. This horizon bias is harmful in general, but protective when the long-horizon forecast is unreliable.

The same mechanism explains why the gap between S3TS and MCTS widens with budget rather than closing, from 3.7% at 1 s to 5.8% at 8 s, even though the two methods share their entire implementation and differ only in their input scenario tree. The MCTS trajectory in Table 1 is essentially flat across 1–4 s (27.49–27.55) and ticks up slightly at 8 s (27.62). A larger budget allows the algorithm to explore deeper into the simulation tree and incorporate more of the median trajectory into its value estimates — which, by the argument above, degrades the publication rather than improving it. The deterministic baseline thus shows a mild overfitting-to-forecast effect, asymptoting toward the failure mode of Deterministic MPC as compute time grows. S3TS does not exhibit this pattern because its scenario tree expresses the uncertainty in the late-period SI through multiple branches rather than committing to a single misleading trajectory: deeper search lets it weight those branches more accurately, not propagate a wrong point forecast further. The result is that uncertainty representation and compute budget interact constructively for S3TS and destructively for the deterministic baseline.

A sizeable gap separates all techniques from Perfect Knowledge (15.34 MAE), with 49.3% remaining even for Stochastic MPC. We emphasize that Perfect Knowledge is not the optimum of the stochastic problem the techniques are solving: it is a clairvoyant oracle that observes the realized SI trajectory in advance, and as such represents an unattainable target. Its distance from Stochastic MPC quantifies the irreducible cost of stochasticity in this problem — even an optimal stochastic policy cannot, on average, close to within 49% of clairvoyance under the SI fluctuations we considered. This is intrinsic to the random process, not to any specific algorithm, and is consistent with the empirical observations that publishing imbalance settlement prices remains a difficult problem even given strong forecasting tools.

The distributional view in Fig. 7 adds qualitative support to the table. Stochastic MPC and S3TS exhibit MAE distributions more concentrated than Deterministic MPC, which shows a longer upper tail of high-MAE periods — the conservative-strategy effect of uncertainty-aware planning, avoiding the worst-case publications at the cost of occasionally accepting slightly higher errors on easy periods. A representative within-period example is shown in Fig. 8, illustrating how the uncertainty-aware methods anticipate the final sign of the SI in a period where the average crosses zero late.

5.2. Non-linear dynamics with linear cost term

The non-linear dynamics make the problem analytically intractable, so the comparison narrows to the techniques whose machinery does not depend on a closed-form model. We first discuss the comparison between S3TS and simpler baselines such as Rule-Based, Deterministic MCTS. In Section 5.4, we extend our comparison with stochastic-aware baselines. Table 2 reports the MAE achieved by each technique on the non-linear regime under $q = 1$.

Both planning methods remain substantially better than the rule-based baseline — a 24.8% improvement for S3TS and 22.0% for MCTS at 8 s. The advantage of advanced planning therefore carries over from the linear regime: the discontinuous price ladder and the non-linear actors' response still allow either tree search method to extract structure that a naive heuristic cannot. S3TS continues to outperform its deterministic MCTS counterpart at every computational budget, where the difference widens with increasing budget: from 2.4% at

Table 1
Linear regime ($q = 1$): published-price MAE across wall-clock budget settings. Each cell reports the across-period mean ± the Q1–Q3 interquartile interval, computed over 1000 settlement periods.

Technique	Wall-clock budget			
	1 s	2 s	4 s	8 s
Perfect Knowledge	15.34 ± [7.15, 23.87]	15.34 ± [7.15, 23.87]	15.34 ± [7.15, 23.87]	15.34 ± [7.15, 23.87]
Rule-Based	30.28 ± [22.28, 38.36]	30.28 ± [22.28, 38.36]	30.28 ± [22.28, 38.36]	30.28 ± [22.28, 38.36]
Stochastic MPC	22.90 ± [13.70, 30.39]	22.90 ± [13.70, 30.43]	22.90 ± [13.69, 30.55]	22.90 ± [13.72, 30.55]
Deterministic MPC	32.86 ± [20.31, 45.20]	32.86 ± [20.31, 45.20]	32.86 ± [20.31, 45.20]	32.86 ± [20.31, 45.20]
MCTS	27.55 ± [22.36, 32.61]	27.51 ± [22.03, 32.30]	27.49 ± [21.83, 32.42]	27.62 ± [21.83, 32.66]
S3TS (our)	**26.53** ± [22.21, 30.97]	**26.33** ± [22.20, 30.76]	**26.24** ± [22.15, 30.65]	**26.03** ± [21.86, 30.51]

Table 2
Non-Linear regime ($q = 1$): published-price MAE across wall-clock budget settings. Each cell reports the across-period mean ± the Q1–Q3 interquartile interval, computed over 1000 settlement periods.

Technique	Wall-clock budget			
	1 s	2 s	4 s	8 s
Rule-Based	53.31 ± [35.00, 70.00]	53.31 ± [35.00, 70.00]	53.31 ± [35.00, 70.00]	53.31 ± [35.00, 70.00]
MCTS	41.83 ± [30.67, 50.92]	41.73 ± [30.73, 50.58]	41.88 ± [30.48, 51.19]	41.58 ± [30.33, 51.17]
S3TS (our)	**40.81** ± [31.29, 49.33]	**40.70** ± [31.17, 49.27]	**40.13** ± [30.83, 48.75]	**40.08** ± [30.67, 48.75]

Table 3
Non-Linear regime ($q = 2$): published-price MSE across wall-clock budget settings. Each cell reports the across-period mean ± the Q1–Q3 interquartile interval, computed over 1000 settlement periods.

Technique	Wall-clock budget			
	1 s	2 s	4 s	8 s
Rule-Based	5670.8 ± [3735.0, 7410.0]	5670.8 ± [3735.0, 7410.0]	5670.8 ± [3735.0, 7410.0]	5670.8 ± [3735.0, 7410.0]
MCTS	2992.6 ± [1792.0, 3844.9]	2998.3 ± [1758.6, 3850.2]	2930.3 ± [1706.0, 3773.9]	2908.0 ± [1673.5, 3757.2]
S3TS (our)	**2925.4** ± [1878.1, 3767.5]	**2888.7** ± [1805.3, 3733.9]	**2821.9** ± [1766.0, 3606.4]	**2753.6** ± [1766.7, 3523.6]

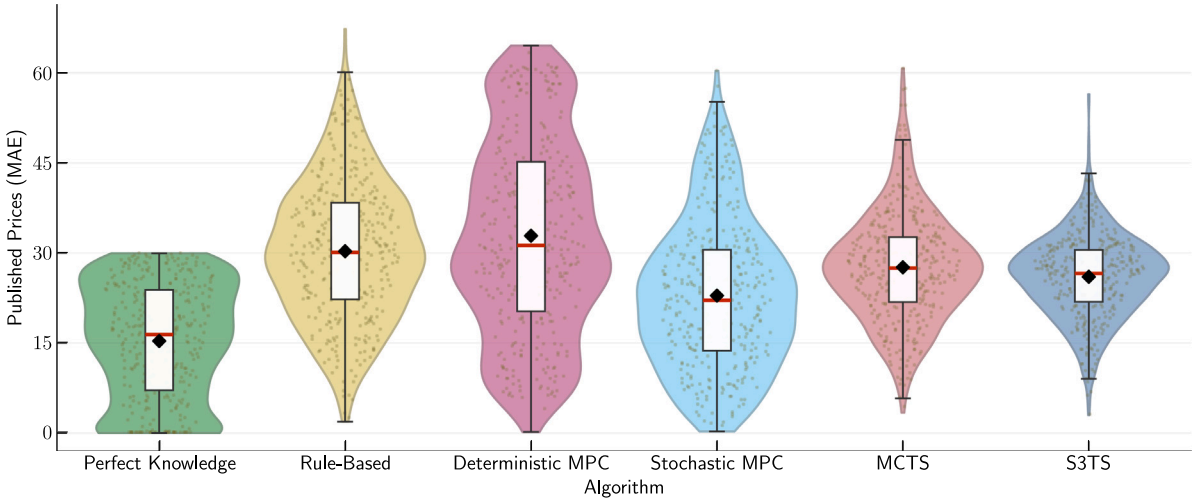


Fig. 7. Distribution of the published price MAE vs. the actual one in the linear problem formulation. The red lines indicate the median values, while the black diamonds indicate the means.

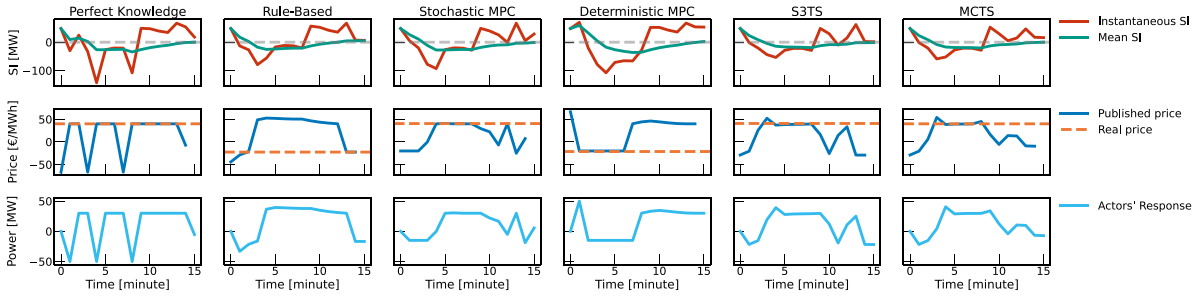


Fig. 8. Example of price publication in the linear problem formulation using different techniques. The rolling average SI faces sign instability throughout the settlement period, adding significant uncertainty to the final price.

1 s to 3.6% at 8 s, with S3TS improving monotonically while MCTS stays mostly flat. This is the same pattern observed in the linear case, now reproduced under genuinely non-linear dynamics and without an available analytical reference. The continued benefit of explicit uncertainty handling under these conditions is the central finding of this experiment: it indicates that the strength of S3TS does not rest on the tractability of the surrounding problem, and that the gains we attributed to scenario-tree reasoning in the linear case can still be reaped for non-smooth dynamics and when the cost function is no longer affine.

The distributional view in Fig. 9 adds qualitative support to our observation for Table 2. The Rule-Based's MAE distribution is broadly spread across the MAE axis, with substantial mass in the high-error bins $[55, 83[$ and a non-negligible tail extending beyond MAE 80. This spread reflects the heuristic's vulnerability to the discontinuous price ladder: whenever late-period SI dynamics push the final price across a discontinuity that the running average failed to anticipate, the heuristic incurs a large publication error. Both tree search methods relocate the distribution toward lower errors, but they do so in different ways. S3TS concentrates its mass tightly in the central $[28, 55[$ range, accepting consistent mid-range errors in exchange for sharply suppressing the high-error tails. MCTS, by contrast, spreads more broadly: while it places more periods in the low-error bin $[14, 28[$ than S3TS does, it also retains noticeably more mass in the $[55, 83[$ tails. The conservative strategy of S3TS therefore manifests not only as an average reduction in error, but also as a deliberate trade — getting fewer exceptionally good publications, but also fewer exceptionally bad ones.

5.3. Non-linear dynamics with quadratic cost term

Table 3 reports the MSE achieved by each technique on the non-linear regime under the quadratic cost exponent ($q = 2$). The qualitative story carries over from the $q = 1$ case: both tree search methods substantially outperform Rule-Based, S3TS outperforms Deterministic MCTS at every budget, and the S3TS–MCTS gap widens with computational budget (from 2.2% at 1 s to 5.3% at 8 s, with S3TS improving monotonically while MCTS improves only marginally).

What changes under the quadratic cost is the magnitude of the advantages. The gap between Rule-Based and the tree search methods roughly doubles, from 24.8% under MAE to 51.4% under MSE. This is expected from the form of the cost function: by squaring the publication error, MSE places disproportionate weight on the high-error tail of the per-period distribution, and as the histogram in Fig. 10 shows, the Rule-Based distribution extends substantially further into that tail than either tree search method. The S3TS–MCTS gap also widens slightly under MSE — from 3.6% in the $q = 1$ experiment to 5.3% with $q = 2$ at the largest budget — for the same reason: S3TS's conservative-strategy effect suppresses moderate-to-high errors more aggressively than MCTS does, and that suppression matters more when errors are weighted quadratically.

The distributional view in Fig. 10 makes this concrete. The histogram exhibits the same risk-trading pattern already discussed in Section 5.2, now amplified by the quadratic cost: S3TS concentrates its mass tightly around the modal bin, MCTS retains visibly more mass in both the low-error and the high-error bins, and Rule-Based spreads across the full range with a thin tail at the high end. The difference is one of emphasis rather than kind: under quadratic cost, the bins on the high-error side of the modal bin contribute disproportionately to the mean, so the portion of the MCTS distribution sitting in those bins weighs more heavily here than the equivalent portion did under MAE. The 5.3% S3TS–MCTS gap at 8 s reflects almost entirely this tail asymmetry.

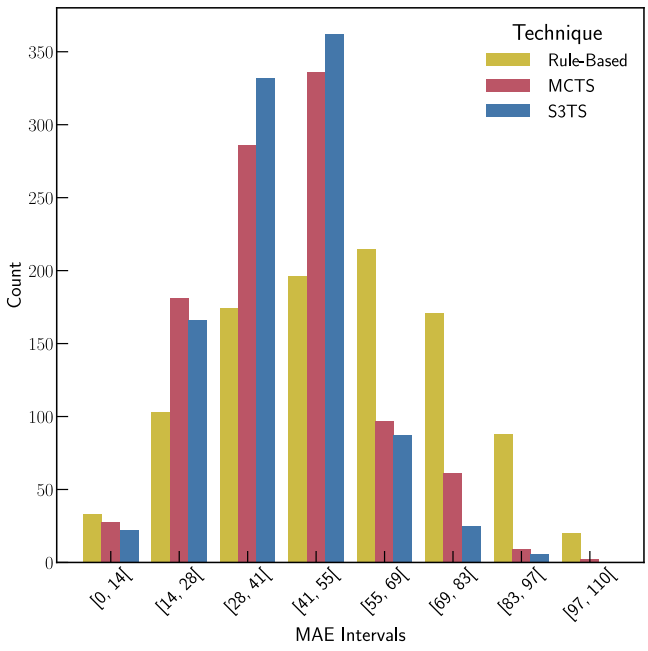


Fig. 9. Publication MAE histogram in the non-linear regime with $q \doteq 1$.

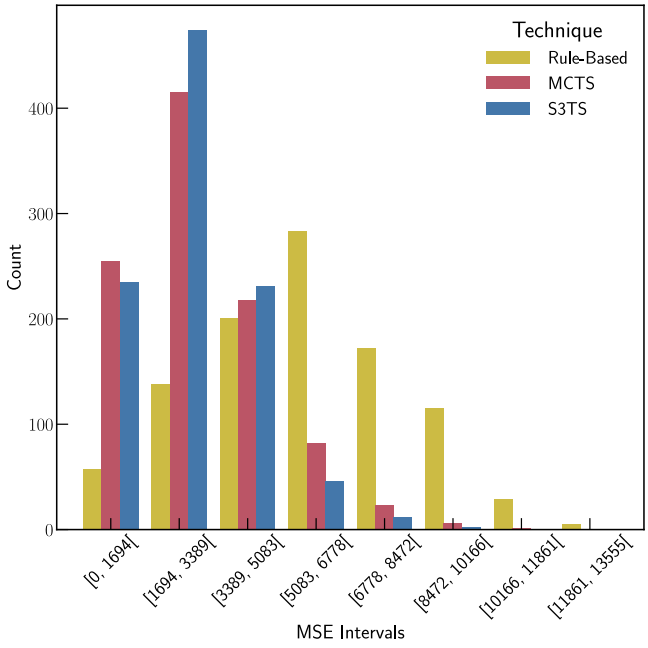


Fig. 10. Publication MSE histogram in the non-linear regime with $q \doteq 2$.

5.4. Comparison against online outcome sampling

Table 4 compares S3TS against three stochastic-based baselines at the largest budget with non-linear dynamics for both linear ($q = 1$) and quadratic ($q = 2$) cost terms.

S3TS outperforms all three baselines in both regimes. Against sparse sampling the margin is 7.9% under $q = 1$ and 22.3% under $q = 2$. The β-matched control is essentially as effective as the unmatched one at $q = 1$, while still gaining a substantial advantage under quadratic costs ($q = 2$); probably thanks to more extended tree search depth springing from the contained sampling splits. The quantile baseline closes roughly half of the gap at $q = 1$, reducing the margin from 7.9% to 4.4%. This delimits the claim that a substantial part of the distance to naive sparse

Table 4
Comparison against online outcome sampling at the 8 s budget. Each cell reports the across-period mean ± the Q1–Q3 interquartile interval over 1000 settlement periods; the win rate is the fraction of periods in which S3TS achieves the lower error.

Technique	Non-Linear, $q = 1$ (MAE)		Non-Linear, $q = 2$ (MSE)	
	Error	Win rate	Error	Win rate
Sparse sampling	43.52 ± [33.23, 52.17]	0.724	3546.1 ± [2291.2, 4496.2]	0.863
β-matched sampling	43.51 ± [32.92, 51.77]	0.663	3359.3 ± [2190.1, 4238.1]	0.818
Quantile sampling	41.92 ± [33.00, 51.08]	0.685	3302.5 ± [2136.0, 4234.1]	0.806
S3TS (our)	**40.08** ± [30.67, 48.75]	—	**2753.6** ± [1766.7, 3523.6]	–

sampling is attributable to the quality of the sampler rather than to the scenario tree as such. The residual margin, however, consistently persists, with S3TS achieving lower errors in 68.5% of settlement periods. Scenario reduction amortizes the cost of constructing a good distributional summary across decisions, whereas online sampling must pay it inside every decision and, at these budgets, cannot afford enough samples to match it. Consistently with this reading, the margin does not shrink under the quadratic cost, where it stands at 16.6%.

5.5. Sensitivity to the scenario-tree configuration

To further support S3TS benefits in the use case considered, we performed a sensitivity analysis of the algorithm parameters, studying *both* S3TS and the deterministic ablation at every setting. We varied four of the parameters we found to be more impactful on the techniques' accuracy: (i)the size N_r of the reduced scenario set used to build the scenario trees, (ii)the branching schedule β that determines the shape of the tree, (iii)the UCT exploration constant α, and (iv)the resolution and size of the action space.

Table 5 reports the outcome at the 8 s budget on the non-linear regime with $q = 1$. It is important to note small variations in the deterministic MCTS performance when varying parameters that do not directly affect the algorithm (branching and set size). We attribute this to small variations in the tree searches under the wall-clock budget enforcement, making each experiment slightly different based on the number of simulations performed within the same computational budget, which might still vary slightly despite the shared hardware settings. Also, note that the configuration used for the results previously discussed in Section 5.2 is itself one of the cells of Table 5, which we indicate as the '*Base*' row.

The performance of S3TS stays relatively fixed with variations of the scenario tree characteristics, namely the size of the reduced set N_r and the branching factor defining the tree topology. The action space granularity and size also do not seem to significantly affect either S3TS or MCTS. The single factor with a more substantial effect is the exploration constant α. A higher α favors S3TS performance, while a lower one hurts MCTS considerably more than S3TS. Importantly, the advantage of S3TS over its deterministic counterpart is preserved in every configuration. Taken together, our sensitivity analysis indicates that the benefit of scenario-structured search is a property of the method rather than of a particular parameterization, and that the tree configuration can be varied over a wide range without eliminating it.

5.6. Robustness to a misspecified scenario distribution

As a final ablation experiment, we perturbed the distribution used to generate the scenarios while leaving the process that produces the test trajectories untouched, so that both S3TS and MCTS plan against a model that is wrong in a controlled and measurable way.

Two families of perturbations are considered. The first shifts the conditional mean of the sampling kernel by a bias $\delta \in \{\pm 5, \pm 10\}$ MW, representing a forecast that systematically over- or under-predicts the SI. The second scales its conditional standard deviation by a factor $\sigma_{\text{scale}} \in \{0.75, 1.25, 1.5\}$, representing a forecast that is over- or under-confident about the magnitude of the fluctuations. The correctly specified cell is retained as a control and is identical to the unperturbed setting we considered in the previous result analysis. Table 6 reports the outcome at the 8 s budget on the non-linear regime with $q = 1$.

Under the most severe negative bias, S3TS's relative advantage over MCTS rises from 3.6% to 9.5%. The mechanism is visible in how the two techniques degrade: moving from the control to $\delta = -10$ MW costs S3TS 3.82 MAE but costs MCTS 6.94. We thus find that the deterministic baseline absorbs roughly twice the damage of S3TS. This is what one would expect from the argument developed in Section 5.1. Representing uncertainty explicitly therefore buys not only accuracy under a correct model, but tolerance to an incorrect one.

We further note that the effect is markedly asymmetric in the sign of the bias. A positive bias improves the absolute accuracy of *both* techniques — S3TS from 40.08 to 38.54 MAE and MCTS from 41.58 to 39.85 at $\delta = +10$ MW — which deserves comment, since a deliberately wrong forecast model would rather be expected *not* to help. The perturbation happens to cancel a systematic offset already present in the correctly-specified setting. In fact, under no distribution misspecification, both planners publish prices well above the price the period eventually settles at. Because the publication price is a decreasing function of the SI estimate, shifting the sampling distribution upward moves the published prices down, and thereby toward the realized settlement price. We therefore read the positive-bias cells of Table 6 ($\delta = +5$ and $\delta = +10$) as an indication that the publication policies of both planners carry a correctable bias under the nominal configuration — a finding of independent interest, and one that suggests calibrating the sampling distribution against realized settlement prices as a straightforward improvement to either method.

Misspecifying the variance produces a milder effect. In this case, what it changes is not the position of the sampled trajectories (i.e., there is no shift to the mean and median), but instead the width of the scenario tree, and that is a quantity only S3TS can see. Since the element-wise median of a set of trajectories is invariant to a symmetric rescaling of the noise about the conditional mean, σ_{scale} reaches the ablation only through Monte Carlo error in estimating that median. It is accordingly unmoved by it, and its paired changes against the control are relatively small. S3TS, by contrast, responds monotonically — with a MAE variation of +0.74 at $\sigma_{\text{scale}} = 0.75$, −0.08 at 1.25, and −0.22 at 1.5.

6. Conclusions

6.1. Strengths and limitations

While the results presented are promising, further discussion is needed to contextualize the strengths and limitations of the S3TS algorithm. As discussed above, the major strength of S3TS lies in its direct handling of stochasticities through a multi-stage stochastic optimization framework, while retaining the MCTS flexibility to handle highly non-linear models. However, several limitations need to be considered.

First, some important limitations spring from the specific experimental setup we presented. While a different range of experiments has been presented, the results rely on synthetic data; an evaluation with real-world data would further add credibility to the algorithm's

Table 5
Sensitivity analysis of the S3TS advantage, non-linear regime with $q = 1$ at the 8 s budget, over 1000 settlement periods when varying the algorithm parameters. We also report the gap (MCTS - S3TS) for easier reference.

Factor	Setting	S3TS	MCTS	Gap
Base	$N_r = 100,\ \beta = (1, 3, 3, 1, \dots),\ \alpha = 3,\ \lvert\mathcal{A}\rvert = 5$	40.08	41.58	+1.50
Reduced set	$N_r = 50$	40.15	41.77	+1.62
	$N_r = 200$	40.20	41.85	+1.65
Branching	$\beta = (1, 2, 2, 2, 2, 1, \dots)$	40.27	41.71	+1.44
	$\beta = (1, 3, 3, 3, 1, \dots)$	40.37	41.82	+1.45
Exploration	$\alpha = 1.5$	40.24	42.49	+2.25
	$\alpha = 6$	39.84	41.79	+1.95
Action space	$\lvert\mathcal{A}\rvert = 9$	40.73	42.08	+1.35

Table 6
Robustness to a misspecified scenario distribution, non-linear regime with $q = 1$ at the 8 s budget, over 1000 settlement periods. The distribution used to generate the scenarios has its conditional mean shifted by δ or its standard deviation scaled by σ_{scale}, while the test trajectories continue to follow the true process. Each cell reports the mean across all settlement periods. We also report the gap (MCTS - S3TS) for easier reference.

Perturbation	Setting	S3TS	MCTS	Gap
None	Correctly specified	40.08	41.58	+1.49
Bias	$\delta = -10$ MW	43.90	48.52	+4.62
	$\delta = -5$ MW	41.38	44.10	+2.72
	$\delta = +5$ MW	39.16	40.42	+1.26
	$\delta = +10$ MW	38.54	39.85	+1.31
Variance	$\sigma_{\text{scale}} = 0.75$	39.69	41.92	+2.23
	$\sigma_{\text{scale}} = 1.25$	40.41	41.82	+1.41
	$\sigma_{\text{scale}} = 1.50$	40.85	42.12	+1.27

effectiveness. Moreover, only one use case (imbalance price publication) has been presented; the algorithm should be tested and evaluated on multiple applications to further assess its effectiveness. Second, the algorithm's nature gives rise to several limitations that need to be carefully reviewed by practitioners when deciding upon using S3TS. To integrate the scenario tree into the search, more computational effort is required to address each scenario trajectory. This essentially means that simulations are now more computationally expensive compared to standard MCTS, and the added cost directly depends on the size of the scenario tree used. Consequently, given the same computational budget, S3TS will have shallower search trees compared to MCTS, as every simulation (intended as the execution of a selection, expansion, simulation, and backpropagation iteration; as in Section 3.3) is more computationally demanding. This is shown in Table 7, where we show the average number of simulations and the depth of S3TS and MCTS for the tree search performed in our experiments. While we proved the effectiveness of S3TS in this specific use case, the computational burden can become a limiting factor for S3TS vs. MCTS for use cases where a long depth is essential (which particularly may be a concern in applications that require fast decision-making).

6.2. Future directions

Several avenues remain to enhance the algorithm. The most immediate is to enrich the search phase with prior knowledge, in the spirit of [40]: value estimates could be attached to state nodes, but also — perhaps more interestingly — to scenario nodes, allowing the tree search to directly compute toward the genuinely informative branches. This could also help close the matched-budget gap to Stochastic MPC observed in the linear regime, since much of the residual 14% is plausibly attributable to the simulation budget rather than to a structural limitation of the algorithm. A second direction is to evaluate S3TS across a more diverse set of problem domains. The demand response problem considered here was chosen as a tractable proof-of-concept, but the S3TS algorithm's flexibility suggests it could be equally applied to residential heating, microgrid scheduling, and complex maintenance optimization, among others. Finally, the algorithm's objective function allows for a straightforward extension to risk-sensitive criteria such as CVaR, which would let it explicitly target the tail behavior that the present quadratic-cost results already show to be the dominant differentiator of our S3TS compared to conventional MCTS.

6.3. Final discussion

We introduced S3TS, an adaptation of MCTS to multi-stage stochastic optimization through scenario trees. S3TS inherits the flexibility of the MCTS framework with respect to model complexity, supporting non-linear dynamics and non-linear cost functions, which both fall outside the capabilities of conventional mathematical-programming solvers. At the same time, by taking a pre-built scenario tree as input rather than sampling outcomes online or learning them through a latent model (as in most popular stochastic MCTS alternatives), it aligns with the multi-stage stochastic optimization tradition and can be driven by the same scenario-generation pipelines that already feed stochastic MPC.

We evaluated S3TS on a demand response signal publication problem, inspired by the imbalance settlement mechanism currently in place in Belgium. The evaluation spanned three experiments and a matched wall-clock budget protocol of 1, 2, 4, and 8 s per publication.[5] In the linear regime (i.e., with simplified linear dynamics), S3TS came within 14% of the optimum of the multi-stage stochastic problem solved with MILP — a proof of concept that the algorithm closely tracks the stochastic optimum even without direct access to the analytical model. In the two non-linear experiments (with linear and quadratic cost terms), where no MILP reference is available, S3TS outperformed both a rule-based baseline and deterministic MCTS at every budget, as well as three alternative online sampling tree searches that model uncertainties by sampling trajectories within the tree search. Two findings recurred across all three experiments and deserve emphasis: the S3TS–MCTS gap *widens* rather than narrows with budget, indicating that the benefit of explicit uncertainty representation grows with available compute rather than diminishing; and the distributional pattern is consistent across regimes, with S3TS trading a small number of exceptionally low-error publications for substantially fewer high-error ones — a conservative-strategy effect that pays off particularly under quadratic cost.

[5] The computational budgets considered allow for realistic applications, as publications typically happen at a minute pace.

Table 7
Search effort per decision, measured at the first timestep of each of the 1000 settlement periods. Cells report the average number of simulations and the average tree search depth.

		B = 1 s		B = 2 s		B = 4 s		B = 8 s	
Regime	Method	Sims	Depth	Sims	Depth	Sims	Depth	Sims	Depth
Non-linear $q = 1$	S3TS	301	5.00	583	5.81	1083	6.00	1953	6.58
	MCTS	2387	6.97	3668	7.20	8109	8.14	16,042	9.12
Non-linear $q = 2$	S3TS	287	4.96	534	5.10	1018	5.97	1944	6.14
	MCTS	2352	6.71	4329	7.07	8794	8.04	16,128	8.87
Linear	S3TS	252	4.96	531	5.17	807	5.89	1773	6.14
	MCTS	1850	6.27	4127	7.33	8085	8.19	15,610	9.21

CRediT authorship contribution statement

Fabio Pavirani: Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Bert Claessens:** Writing – review & editing, Supervision, Methodology, Conceptualization. **Pierre Pinson:** Writing – review & editing, Supervision, Conceptualization. **Chris Develder:** Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was partly funded by the Flemish Government through the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme as well as a travel grant from the Research Foundation - Flanders, by the Energy Transition Fund.

Appendix. Computational hardware

Because the per-decision budget in this study is enforced as wall-clock time rather than as a fixed number of simulations, the execution environment is part of the experimental design. The number of simulations a technique completes within B seconds depends on the machine it runs on and on what else that machine is doing.

All experiments were executed on identical bare-metal nodes. Each node carries a six-core Intel Core i5-9500 CPU (3.00 GHz base, 4.40 GHz maximum, simultaneous multithreading disabled) and 62 GB of RAM, and runs Ubuntu 22.04.5 with Linux kernel 5.15.0 and uses Gurobi 13.0.2 [35]. No node is virtualized or shared with another tenant. Only one job runs per node at a time, so no result is obtained under a different level of contention than any other.

Data availability

No data was used for the research described in the article.

References

[1] Zahraoui Y, Pokhrel BR, Oleinikova I, Gros SN. Assessment of TSO-dso coordination to enhance flexible grid operation. Curr Sustainable/Renewable Energy Rep 2025;12(1):20.
[2] Sezgen O, Goldman C, Krishnarao P. Option value of electricity demand response. Energy 2007;32(2):108–19.
[3] Silva C, Faria P, Vale Z, Corchado J. Demand response performance and uncertainty: A systematic literature review. Energy Strat Rev 2022;41:100857.
[4] Bakirtzis GA, Biskas PN, Chatziathanasiou V. Generation expansion planning by MILP considering mid-term scheduling decisions. Electr Power Syst Res 2012;86:98–112.
[5] He H, Harlim J, Huang D, Li Y. Efficient MPC-based energy management system for secure and cost-effective microgrid operations. 2025, arXiv preprint arXiv:2510.03241.
[6] Haugen M, Farahmand H, Jaehnert S, Fleten S-E. Representation of uncertainty in market models for operational planning and forecasting in renewable power systems: a review. Energy Syst 2023;1–36.
[7] Roald LA, Pozo D, Papavasiliou A, Molzahn DK, Kazempour J, Conejo A. Power systems optimization under uncertainty: A review of methods and applications. Electr Power Syst Res 2023;214:108725.
[8] Birge JR, Louveaux F. Introduction to stochastic programming. Springer; 1997.
[9] Patrinos P, Trimboli S, Bemporad A. Stochastic MPC for real-time market-based optimal power dispatch. In: 2011 50th IEEE conference on decision and control and European control conference. IEEE; 2011, p. 7111–6.
[10] Lei Y, Wang D, Jia H, Li J, Chen J, Li J, Yang Z. Multi-stage stochastic planning of regional integrated energy system based on scenario tree path optimization under long-term multiple uncertainties. Appl Energy 2021;300:117224.
[11] Smets R, Bruninx K, Bottieau J, Toubeau J-F, Delarue E. Strategic implicit balancing with energy storage systems via stochastic model predictive control. IEEE Trans Energy Mark Policy Regul 2023;1(4):373–85.
[12] Madahi SSK, Bruninx K, Claessens B, Develder C. Model predictive control-guided reinforcement learning for implicit balancing. 2025, arXiv preprint arXiv:2510.04868.
[13] Anderson E, Ferris M, Philpott A, Anitescu M, Cramton P, Geng S, Green R, Homem-de Mello T, Huber O, Leclère V, et al. Ten challenges for mathematical modeling of the green-energy transition. Curr Sustainable/Renewable Energy Rep 2025;12(1):1–13.
[14] Kotzur L, Nolting L, Hoffmann M, Groß T, Smolenko A, Priesmann J, Büsing H, Beer R, Kullmann F, Singh B, et al. A modeler's guide to handle complexity in energy systems optimization. Adv Appl Energy 2021;4:100063.
[15] Pavirani F, Gokhale G, Claessens B, Develder C. Demand response for residential building heating: Effective Monte Carlo tree search control based on physics-informed neural networks. Energy Build 2024;311:114161.
[16] Kiljander J, Sarala R, Rehu J, Pakkala D, Pääkkönen P, Takalo-Mattila J, Känsälä K. Intelligent consumer flexibility management with neural network-based planning and control. IEEE Access 2021;9:40755–67.
[17] Pavirani F, Van Gompel J, Madahi SSK, Claessens B, Develder C. Predicting and publishing accurate imbalance prices using Monte Carlo tree search. Appl Energy 2025;392:125944.
[18] Hao Z, Di Maio F, Zio E. Monte carlo tree search-based deep reinforcement learning for flexible operation & maintenance optimization of a nuclear power plant. J Saf Sustain 2024;1(1):4–13.
[19] Shuai H, He H. Online scheduling of a residential microgrid via Monte-Carlo tree search and a learned model. IEEE Trans Smart Grid 2020;12(2):1073–87.
[20] Lin Z, Lan J, Liu S, Tian Z, Zhao D, Wei C. Hierarchical multi-agent MCTS for safety-critical coordination in mixed-autonomy roundabouts. IEEE Trans Veh Technol 2026. http://dx.doi.org/10.1109/TVT.2026.3682689, [Early access].
[21] Lin Z, Wu L, Tian Z, Lomuscio A, Lan J. Scalable and safe multi-agent coordination with reconstructed level-k Monte Carlo tree search. In: Proceedings of the 25th international conference on autonomous agents and multiagent systems. Paphos, Cyprus; 2026.
[22] Dana Mazraeh H, Parand K. Approximate symbolic solutions to differential equations using a novel combination of Monte Carlo tree search and physics-informed neural networks approach. Eng Comput 2025;41(4):2707–35. http://dx.doi.org/10.1007/s00366-025-02127-x.
[23] Dana Mazraeh H, Parand K. A three-stage framework combining neural networks and Monte Carlo tree search for approximating analytical solutions to the Thomas–Fermi equation. J Comput Sci 2025;87:102582. http://dx.doi.org/10.1016/j.jocs.2025.102582.
[24] Ballard BW. The*-minimax search procedure for trees containing chance nodes. Artificial Intelligence 1983;21(3):327–50.
[25] Kearns M, Mansour Y, Ng AY. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. Mach Learn 2002;49(2):193–208.
[26] Silver D, Veness J. Monte-Carlo planning in large POMDPs. Adv Neural Inf Process Syst 2010;23.

[27] Couëtoux A, Hoock J-B, Sokolovska N, Teytaud O, Bonnard N. Continuous upper confidence trees. In: International conference on learning and intelligent optimization. Springer; 2011, p. 433–45.
[28] Schrittwieser J, Antonoglou I, Hubert T, Simonyan K, Sifre L, Schmitt S, Guez A, Lockhart E, Hassabis D, Graepel T, et al. Mastering atari, go, chess and shogi by planning with a learned model. Nature 2020;588(7839):604–9.
[29] Antonoglou I, Schrittwieser J, Ozair S, Hubert TK, Silver D. Planning in stochastic environments with a learned model. In: International conference on learning representations. 2021.
[30] Heitsch H, Römisch W. Scenario reduction algorithms in stochastic programming. Comput Optim Appl 2003;24(2):187–206.
[31] Growe-Kuska N, Heitsch H, Romisch W. Scenario reduction and scenario tree construction for power management problems. In: 2003 IEEE bologna power tech conference proceedings, vol. 3, IEEE; 2003, p. 7–pp.
[32] Heitsch H, Römisch W. Scenario tree modeling for multistage stochastic programs. Math Program 2009;118(2):371–406.
[33] Pinson P, Madsen H, Nielsen HA, Papaefthymiou G, Klöckl B. From probabilistic forecasts to statistical scenarios of short-term wind power production. Wind Energy: An Int J Prog Appl Wind Power Convers Technol 2009;12(1):51–62.
[34] Elia Transmission Belgium. Tariffs for maintaining and restoring the residual balance of individual access responsibles (2024–2027 period). Technical report, Elia; 2024, URL https://www.elia.be/-/media/project/elia/elia-site/public-consultations/2024/20240606_proposal-for-modification-of-the-tariff-for-maintaining-and-restoring-the-residual-balance/20240606_tarifs-desequilibre-2024-2027_en-clean.pdf. [Accessed: 16 January 2026].
[35] Gurobi Optimization, LLC. Gurobi optimizer reference manual. 2024, URL https://www.gurobi.com.
[36] Madahi SSK, Claessens B, Develder C. Distributional reinforcement learning-based energy arbitrage strategies in imbalance settlement mechanism. J Energy Storage 2024;104:114377.
[37] Lloyd S. Least squares quantization in PCM. IEEE Trans Inform Theory 1982;28(2):129–37.
[38] Busoniu L, Babuska R, De Schutter B, Ernst D. Reinforcement learning and dynamic programming using function approximators. CRC Press; 2017.
[39] Kocsis L, Szepesvári C. Bandit based monte-carlo planning. In: European conference on machine learning. Springer; 2006, p. 282–93.
[40] Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A, Lanctot M, Sifre L, Kumaran D, Graepel T, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. 2017, arXiv preprint arXiv:1712.01815.