

Forecast-Based Optimization: Applying Predictive Models for Decision-Making Under Uncertain Constraints

Edson M. Odake Junior  [Sidia & Department of Electrical Engineering, Federal University of Paraná |edson.junior@sida.com]

Fabio F. Oliveira  [Sidia & InovAI Lab, Federal University of Rio Grande do Norte |fabio.fonseca@sida.com]

Rafael S. Azevedo  [Sidia & Department of Electrical and Telecommunications Engineering, Fluminense Federal University |rafael.azevedo@sida.com]

Jorge M. S. Vasconcellos  [Sidia & Department of Administration, Federal University of Amazonas |jorge.vasconcellos@sida.com]

Leandro Andrade Vieira  [Sidia |leandro.andrade@sida.com]

Abstract

Process automation has become a commonly adopted concept of modern business operations, driven by increasing investments in accessible software and tools. Automation workflows streamline decision-making by reducing human error, aligning with company standards, and freeing resources for higher-value tasks. Intelligent systems represent the next step in automation, applying machine learning algorithms to go beyond rule-based decision-making. However, their implementation requires expert modeling and integration into daily routines, posing significant challenges for users to embrace. This paper discusses the end-to-end complexities associated with implementing an intelligent automated system and explores the benefits of integrating machine learning methods, especially forecasting algorithms, to improve the system's efficiency. It also provides a practical situation solved by introducing a two-stage hybrid approach based on the predict-then-optimize paradigm, addressing a problem with long-term intelligent decision-making. This methodology integrates forecasting models to predict future information for long-term evaluations, which are then utilized by an optimizer to recommend configurations that balance current and future outcomes. This approach reduces the limitations of traditional optimization, offering a practical solution for scenarios where future information is unavailable.

Keywords: *Forecasting Algorithms; Machine Learning; Contextual Optimization; Decision Making; Resource Allocation;*

1 Introduction

Process automation has attracted more investments as dedicated software and tools become more accessible, facilitating its adoption (Elhajjar et al., 2023). Automation is commonly performed through workflows in which data enters the system and follows a pre-organized sequence of validations and decisions based on how the company should interpret and act on new information (Zayas-Cabán et al., 2021). This approach offers multiple benefits, such as reducing vulnerability to human error, ensuring that decision-making aligns with company standards, and minimizing time spent on daily micro-decisions (Nielsen et al., 2023). As a result, employees and managers can allocate time and resources toward more complex, meaningful work. For example, strategies to deal with the declining birth rate include the implementation of process automation, increasing operational efficiency (Hsiung and Wang, 2022). However, the successful integration of process automation depends not only on technology availability but also on the expertise to model the company's workflow, adapt it to current contexts, and embed it into the daily routines of diverse teams.

In recent years, applications that integrate process automation and artificial intelligence (AI) for decision-making have also gained significant traction (Abildtrup et al., 2024). Rather than relying solely on fixed rules for decision-making,

these applications incorporate machine learning (ML) models to address scenarios with hidden patterns and increasing complexity (Ribeiro et al., 2021), enabling models to learn from the environment and provide more accurate results even when facing unpredictable scenarios. An example of such reliance on intelligent automated systems (IAS) is the stock market, where companies heavily depend on these technologies. Predicting stock prices remains a complex challenge, as research indicates that historical price data alone is insufficient due to the influence of current news, social media trends, and government actions (Sahu et al., 2023). Despite these obstacles, automated trading systems (ATS) execute pre-programmed trading instructions to automatically issue buy and sell orders to maximize profitability based on factors such as price, volume, mathematical models, and even social media sentiment analysis (Sahu et al., 2023).

Process automation has broad applicability across various industrial sectors. In insurance, it simplifies risk management analysis; in banking, it refines credit scoring; and in manufacturing, it optimizes inventory control (Kitsantas et al., 2024). However, achieving full process automation has been a challenge, and human intervention is often required through an interface layer to carry out activities that have not been automated. In exceptional or unusual scenarios, this interaction occurs in systems that have not been programmed or trained for such decision-making. (Mourtzis et al., 2023).

Due to the complexity of strategic decisions, their reliance on dynamic information, and their rare occurrence, they are generally less automated, making the cost of automation often unjustifiable (Van der Aalst et al., 2018). In contrast, operational management decisions involve numerous micro-decisions, leading to an exponential number of combinations to be explored. Therefore, analyzing the entire decision space in this context becomes highly time-consuming and, consequently, impractical. As a result, significant efforts have been directed toward automating operational decisions, as they are less dependent on external dynamic factors. Van der Aalst et al. (2018) highlighted that automation is most feasible for frequently occurring events that significantly impact the majority of cases. However, exceptional scenarios, such as strategic decisions, should remain human-driven, as their automation would entail higher costs and limited impact. Van der Aalst's work questioned the extent to which processes should be automated, while reaffirming the importance of prioritizing automation for operational decision-making.

Companies often face significant challenges when implementing intelligent systems. Due to limited prior knowledge, they may underestimate the effort, resources, and expertise required for a professional deployment (Hendrawan et al., 2024). Successful implementation demands careful planning from the outset, particularly because the associated costs naturally raise questions about the practical benefits of such systems. Still, much of the existing literature either provides a high-level discussion of intelligent automation without addressing the concrete technical challenges of integrating ML into operational systems or focuses narrowly on technical solutions with limited connection to the broader organizational context (Chakraborti et al., 2020) (Mendling et al., 2018) (Stoykova and Shakev, 2023) (Thiruvady et al., 2021) (Qin et al., 2025). In this sense, the contribution of this work lies not in proposing a new standalone forecasting or optimization method, but in showing how such techniques can be combined within an implementable intelligent system, linking organizational needs and data-driven decision support. Therefore, the paper bridges this gap by describing the main phases involved in implementing such a system and by presenting a case study that illustrates how forecasting and optimization can be used together to improve process efficiency.

The case study addresses a practical resource allocation problem in project management, where the objective is to minimize delivery lateness under uncertain future capacity. Therefore, we propose a two-stage hybrid architecture based on the predict-then-optimize paradigm (Sadana et al., 2025). The approach uses historical data to train ML models that forecast future capacity values, which are then incorporated into an optimization model responsible for recommending allocation decisions that account for both current conditions and expected future behavior. Although the proposed architecture has broader applicability, resource allocation is a common challenge in medium and large companies, making it a suitable setting to showcase how forecasting and optimization can be combined to support decision-making in uncertain operational environments.

Software engineering has moved beyond the development of isolated software artifacts, also encompassing the design, integration, and maintenance of complex data-driven sys-

tems embedded in organizational processes. In this context, intelligent components such as machine learning and optimization algorithms are becoming part of practical operational environments, making it essential to understand how such elements can be combined within implementable systems. Therefore, the main contributions of our paper are:

- **IAS Framework:** An end-to-end perspective for implementing intelligent automated systems, highlighting the main challenges, requirements, and expected benefits associated with their development.
- **Hybrid Architecture:** A hybrid architecture that integrates forecasting and optimization methods to support resource allocation decisions under uncertain future capacity.
- **Quantitative Analysis:** A proof-of-concept simulation analysis showing the potential benefits of the proposed architecture and comparing the impact of different forecasting methods on the resulting decisions.

The paper is structured as follows: Section 2 examines the practical challenges of implementing an automated decision-making system and outlines a roadmap for developing a sustainable intelligent system; Section 3 introduces the core concepts and mathematical formulations necessary for modeling generic forecasting-based optimization problems; Section 4 presents a practical case study, detailing the methods used to solve the problem and explaining the generation of synthetic data to ensure reproducibility in simulations; Section 5 analyzes the performance improvements achieved by our method and evaluates the accuracy of various forecasting techniques; Section 6 discusses the limitations of our proposal and suggests future research directions to address these challenges; Finally, Section 7 concludes the paper, summarizing our contributions.

2 Automated Decision Making

Business automation has experienced a significant paradigm shift with the rise of Robotic Process Automation (RPA) software. Historically, only well-organized and technologically advanced companies could implement automation in their processes. Today, RPA technologies are increasingly accessible to less technical organizations due to their flexible customization capabilities (Fersht and Slaby, 2012) (Elhajjar et al., 2023). While platforms like Blue Prism (Chappell, 2017) prioritize scalability and security, they require a more technical foundation and robust infrastructure. In contrast, alternatives such as UiPath (Tripathi, 2018) offer user-friendly interfaces and strong community support, making them more accessible for businesses of all sizes (Jovanović et al., 2018). Despite these advancements, proper training remains essential to ensure employees are proficient in using these tools (Hendrawan et al., 2024).

These platforms are tools to support the development of intelligent automation systems. However, the full implementation of such systems is not straightforward, and a three-phase methodology is proposed for the sustainable implementation of intelligent systems, as illustrated in the diagram of Figure 1. The diagram comprises the preparation, automation, and

data-driven stages, each designed to identify and break down requirements. Naturally, the implementation of each phase should rely on the maturity of the preceding phase.

This section examines the content specified in Figure 1, outlining requirements for companies seeking to advance their automation capabilities. Additionally, the remainder of this paper focuses on a specific implementation that combines optimization and ML to automate decision-making in resource allocation tasks, demonstrating its impact beyond simple repetitive tasks.

2.1 Preparation Phase

For the successful implementation of an automation system, organizations must consider a structured approach that addresses both technical and human factors (Hsiung and Wang, 2022) (McKinsey, 2023). First, the company's existing procedures must be mapped and documented to identify areas where automation can be effectively integrated. This process involves analyzing workflows, pinpointing inefficiencies, and defining clear objectives for the automation initiative. By establishing a detailed blueprint of current operations, organizations can ensure that the automation system aligns with their strategic goals and operational needs. Naturally, this mapping is required for ongoing processes. If the process does not yet exist, it is usually easier to implement process automation since there is no need to deal with transitions, and the process may already be designed to integrate with the system from the beginning.

In addition to procedural mapping, the organization must prepare its infrastructure to accommodate technological adjustments. This includes evaluating if the existing IT systems are compatible with new automation tools, and upgrading hardware or software as necessary (Hendrawan et al., 2024). A robust infrastructure is essential to support integration, scalability, and long-term sustainability of the automation system. Furthermore, organizations must protect sensitive data and maintain operational continuity during the transition (Siderska et al., 2023).

Another factor is the training and development of personnel to ensure they are comfortable using the new tools for implementation, maintenance, and utilization of the technology (Hendrawan et al., 2024). This includes training sessions, workshops, and continuous learning mentality to improve the new system (dos Santos Júnior et al., 2025). By providing employees with the right tools and expertise, organizations can mitigate resistance to change (Nama, 2022).

Given that automation technology directly impacts the daily activities of employees, it is important to develop a plan to smooth the transition. This plan should address potential disruptions, clear communication strategies, and provide ongoing support to employees throughout the implementation process (Siderska et al., 2023). Change management practices, such as stakeholder engagement and feedback, can help minimize resistance and ensure a smoother adoption of the system (Santos et al., 2025). Finally, organizations should establish metrics to monitor progress, evaluate the effectiveness of the automation system, and make adjustments as needed. These actions are related to the Technology Acceptance Model (TAM), which links the system's usage with

its perceived usefulness, simplicity, and management level engagement (Hsiung and Wang, 2022).

2.2 Automation Phase

Once the system is fully integrated into the daily operations of the organization, the next critical step is to automate decision-making processes based on predefined objectives and workflows. This phase marks the transition from manual to automated decision-making, which is guided by the company's strategic goals and operational requirements.

One of the simplest approaches in this phase is the implementation of rule-based automation, which operates on a set of predefined rules or conditions that dictate specific actions or responses (Silva et al., 2023). These rules are derived from the company's processes, policies, and best practices, ensuring that decisions are made consistently and in accordance with established guidelines. For example, in inventory management, a rule-based system might automatically reorder stock when inventory levels fall below a certain threshold, maintaining safe stock levels and preventing shortages. Rule-based automation is useful for tasks that involve repetitive, well-defined processes, as it reduces human error and frees up employees to focus on more complex activities.

In addition to rule-based automation, this phase emphasizes the importance of validation mechanisms to ensure the reliability of the workflow. Validations act as quality control mechanisms that verify the integrity of data and processes before decisions are executed. For instance, in financial transactions, validations might check for discrepancies in account balances, flagging potential errors or fraudulent activities before they are processed.

Taking automation a step further, this phase also explores the application of optimization algorithms to enhance decision-making capabilities. Optimization algorithms analyze input data and use mathematical models to determine the best available solution to a problem, given specific constraints and objectives (Yang, 2008). For example, in supply chain management, an optimization algorithm might calculate the most cost-effective distribution route for delivering products to multiple locations, minimizing transportation costs while meeting delivery deadlines. Therefore, organizations can achieve greater operational efficiency.

2.3 Data-Driven Phase

Optimization algorithms are highly effective in well-modeled deterministic scenarios, where the system's behavior and constraints can be precisely defined. These algorithms are useful in environments where the relationships between variables and outcomes are predictable and consistent, enabling organizations to deliver optimal performances. However, their effectiveness is limited in complex or uncertain scenarios where the system's behavior cannot be easily modeled or predicted, leading to suboptimal or impractical solutions (Sadana et al., 2025).

ML models are an alternative to address these limitations in scenarios where historical data is available, as they can be used to classify events, recognize anomalies, or forecast future fluctuations in advance (Xu and Saleh, 2021). ML mod-

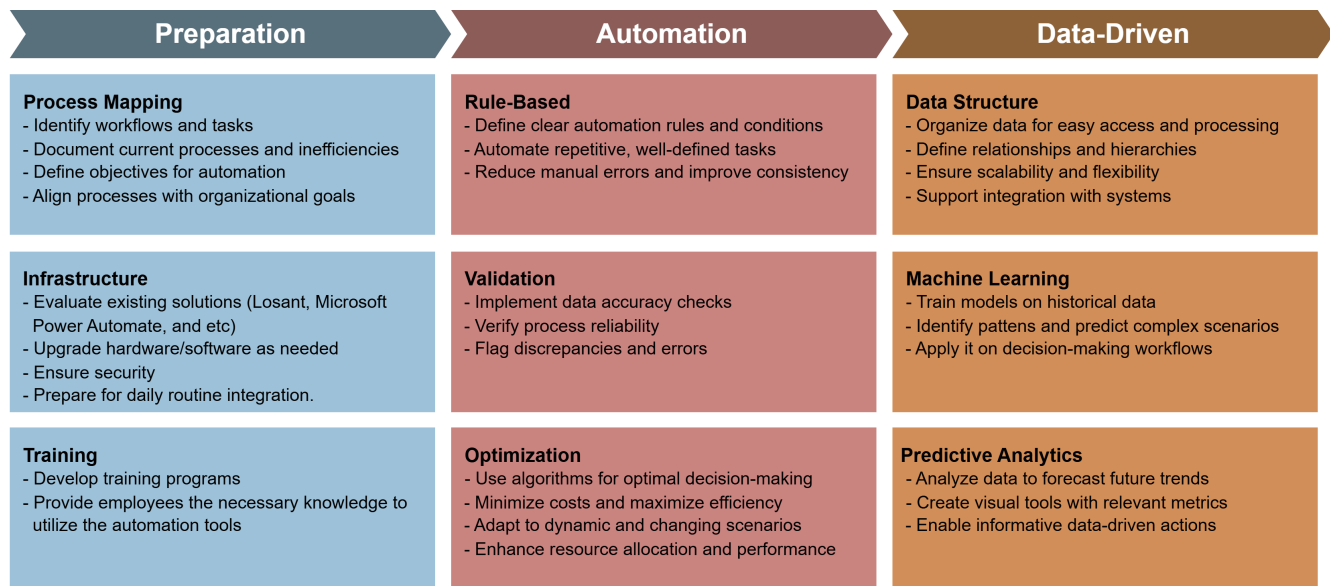


Figure 1. Implementation Phases for an Automated Decision Making Process.

els are trained on past data to identify patterns, trends, and relationships that may not be immediately apparent through deterministic modeling, and depending on the problem complexity, they can be even more reliable than human opinion (Salimzadeh et al., 2024). With historical information, these models can predict future behavior, even in complex or non-deterministic environments. For instance, in demand forecasting, ML algorithms can analyze historical sales data, market trends, and external factors to predict future demand. These predictions can be integrated into the decision-making workflow of RPA, enabling the system to make informed decisions in real time.

The capabilities of ML models extend beyond simple forecasting; they can also detect hidden patterns and correlations. For example, in customer service automation, they can predict customer behavior, identify issues, and recommend solutions, enhancing customer experience (Xu and Saleh, 2021). When these predictions are incorporated into the RPA workflow, organizations can automate complex decision-making processes that require more dynamic responses.

If well implemented, the previous phase provided an error-proof process with optimal performance, while this phase improves uncertain processes and uncovers opportunities. It is of core importance that the data are collected and organized, so the models can learn from it and suggest more customized views and actions that were not previously trivial. In addition to their decision-making applications, ML models enable organizations to achieve a higher level of understanding of their processes and create dashboards for stakeholders. These dashboards visualize metrics and variables in real-time, offering a general view of the system's performance and status (SAP, 2023). For instance, in supply chain management, a dashboard might display forecasted future trends of inventory levels, lead times, and delivery performance metrics, allowing stakeholders to monitor operations closely and address potential issues before they escalate.

3 Background

This section first reviews optimization methods that deal with uncertainty and then presents the fundamental concepts and algorithms used for forecasting and optimization in this work. It explains the forecasting algorithms used in the experiments and connects them to the optimization problem, providing a general formulation for resource allocation.

The main notational conventions adopted throughout this work are introduced here for clarity. The symbol y_t denotes the true value of the target variable at time step t , while $\hat{y}_t$ denotes its predicted value at the same time step. Bold symbols denote vectors, non-bold symbols denote scalar quantities, and uppercase symbols denote matrices.

3.1 Optimization Under Uncertainty

When an optimization model is deployed in practice, decisions often need to be made without full access to all variables that affect the problem (Mandi et al., 2024). For example, future demand, delivery capacity, or resource availability may change over time or remain unknown when the decision is made. Therefore, deterministic formulations become insufficient to represent the actual decision environment, since they assume that all relevant parameters are known in advance. This motivates optimization models combined with learning techniques that learn to deal with uncertainty and missing information. These methods can be categorized into decision rule optimization, sequential learning and optimization (SLO), and integrated learning and optimization (ILO) (Sadana et al., 2025).

Decision rule optimization trains a model to learn a mapping from the available system information to the action, without requiring the solution of an optimization problem after training (Sadana et al., 2025). However, feasibility and optimality are not always guaranteed. In sequential learning and optimization (SLO), a model is trained using historical data, and its output is used as input to the optimization model (Bertsimas and Kallus, 2020) (Sadana et al., 2025). The model

may output point forecasts, where a single predicted value is used, or uncertainty sets may be defined to represent plausible realizations. Robust optimization uses these sets to find decisions that perform well and are feasible given the uncertainty set, which avoids optimistic results when forecasting errors are high, but may also lead to conservative actions, since the optimizer must account for worst-case scenarios (Bertsimas and Sim, 2004) (Bertsimas et al., 2011). Integrated learning and optimization (ILO), also known as decision-focused learning, trains the predictive model according to the optimization objective instead of the prediction accuracy. The output of the learning model is passed to the optimizer, and the learning parameters are updated based on the quality of the resulting decision. For example, recent approaches calibrate uncertainty sets end-to-end (Yeh et al., 2024) (Chenreddy and Delage, 2024), improving the alignment between prediction and optimization, but also increasing solver complexity, since optimization problems are often non-differentiable, making gradient-based training difficult (Donti et al., 2017) (Elmachtoub and Grigas, 2022) (Mandi et al., 2024). In addition, because the predictive model focuses on minimizing the objective function instead of prediction accuracy, the resulting predictions may become less interpretable as estimates of the uncertain variables.

This work adopts a predict-then-optimize architecture with a rolling horizon, in which forecasting models estimate future capacity and an optimization model uses these estimates for task allocation, as described in Subsection 3.3. The objective is not to provide an exhaustive benchmark of all paradigms, but to analyze the behavior of a modular forecast-based optimization architecture in a controlled case study. This choice is consistent for the proof-of-concept scope of the experiments, since the forecasting and optimization components can be evaluated separately, visually inspected, and adapted to different practical scenarios. Decision rule optimization is not adopted because feasibility and optimality guarantees are less direct. ILO methods, although better aligned with the final optimization objective, require integrating the learner and the optimizer during training, which often makes the learning problem harder due to non-differentiable or discontinuous optimization mappings. Robust optimization and uncertainty-set methods are also relevant alternatives, but require additional assumptions about the uncertainty representation and may lead to more conservative decisions. Therefore, the predict-then-optimize rolling-horizon architecture is a practical, interpretable approach that can be adapted to different scenarios without major implementation changes, while also allowing future extensions to incorporate end-to-end learning, robust optimization, or calibrated uncertainty sets (Mandi et al., 2024) (Yeh et al., 2024).

3.2 Forecasting

Forecasting algorithms have been widely employed to generate point predictions or interval tolerance estimates for future values of monitored variables. These variables may include financial trends, demand seasonality, or other indicators that support informed decision-making for stakeholders. These models are trained using historical data from one or multiple time series, capturing information that correlates with the

target variable. However, forecasting values alone are insufficient for decision-making; they must be integrated into a pipeline to process the information and derive actionable decisions. Forecasting methods can be classified into statistical, ML, and deep learning (DL) approaches, as illustrated in Table 1.

Statistical approaches represent the most traditional techniques, specifically designed for time-series data. These methods are characterized by their formal mathematical formulations and computational efficiency. However, they are only suited for linear time-series data and often struggle to capture nonlinear patterns. This category can be divided into two main methodological approaches: autoregressive models, such as ARIMA, which capture autocorrelation within the data; and exponential smoothing methods, such as Holt-Winters, which incorporate concepts like trend and seasonality to describe the data (Hyndman and Athanasopoulos, 2018).

ML methods provide greater potential in interpreting nonlinear patterns. However, studies by Makridakis et al. (2020) indicate that traditional statistical methods often outperform purely ML algorithms. To address this limitation, hybrid approaches that combine multiple techniques have emerged, improving overall performance. ML models used for forecasting, such as Gaussian processes and Regression trees, are regression-based. When these models are fitted, and predictions are extrapolated beyond the monitored axis, they tend to exhibit overfitting as they are not specifically designed to handle time-series data. This may explain why purely statistical methods often outperform them in such contexts.

DL methods, based on neural networks with multiple layers, are often the preferred choice for handling complex data. These models capture a higher degree of data complexity compared to statistical and ML methods, frequently featuring millions of adjustable parameters. They are also highly flexible, allowing their architecture to be tailored to different data types. However, DL models require large datasets for training due to the extensive parameter tuning involved. Once trained, these models can be fine-tuned on smaller datasets if the knowledge from the original database is sufficiently similar, enabling the transfer of learned parameterizations (Weber et al., 2021).

No forecasting method guarantees the best results, as its accuracy depends on various factors, including the characteristics of the time-series data, the volume of available data, the input type, and even the specific requirements of the application. For instance, some applications prioritize precise point estimates, others require confidence intervals, and still others emphasize interpretability.

Following this investigation, multiple forecasting algorithms are analyzed to comprehensively evaluate their performance in a specific scenario. The selected algorithms include some benchmark methods outlined by Makridakis et al. (2020), which are widely recognized as established approaches for time-series analysis, as well as additional methods representing ML and DL techniques. The methods span a range of techniques, from simple approaches that do not require training, such as the Naive method, to advanced DL methods like LSTMs. The forecasting algorithms chosen for evaluating this methodology are:

Model	Strengths	Weaknesses	Reference
ARIMA	Well-understood, built-in interval tolerance	Limited to linear models, doesn't handle seasonality	(Box and Jenkins, 1976)
Simple Exponential Smoothing (SES)	Simple, low computational cost	Short-term forecasting, no trend nor seasonality	(Brown, 1956)
Holt-Winters	Handles trend and seasonality well	Limited to linear models, parameter selection	(Winters, 1960)
SARIMA	Adaptation of ARIMA to handle seasonality	Limited to linear models, computationally intensive	(Box and Jenkins, 1976)
Comb	Improved accuracy, reduces bias	Loss of interpretability	(Bates and Granger, 1969)
Nonlinear autoregressive (NAR)	Captures nonlinear relationships	Parameter selection, stationarity, higher complexity	(Jones and Cox, 1978)
Damped Trend	Realistic trend projections	Limited to linear models, parameter selection	(Gardner Jr and McKenzie, 1985)
Theta	Incorporates long-term trends	May ignore short-term fluctuations	(Assimakopoulos, 2000)
Prophet	Incorporates events, handles missing data	Limited to linear additive model	(Taylor and Letham, 2018)
Linear Regression	Interpretable	Limited to linear models	(Eberly, 2007)
K-Nearest Neighbors (KNN)	Interpretable, non-parametric	High computational cost for large data	(Fix and Hodges, 1951)
Gaussian Process	Captures uncertainty	Computationally intensive, prone to overfitting	(Williams and Rasmussen, 2006)
Decision Trees	Simple, interpretable	Prone to overfitting	(Breiman et al., 1984)
Support Vector Machines (SVM)	Linear and non-linear depending on the kernel	Computationally intensive on large dataset	(Sain, 1996)
Gradient Boosting	Strong model, handles highly complex data	High computational cost, prone to overfitting	(Friedman, 2001)
Multilayer Perceptron (MLP)	Handles complex patterns	Prone to overfitting, require large dataset	(Rosenblatt, 1958)
Recurrent Neural Network (RNN)	Captures temporal dependencies	Vanishing gradient problem, require large dataset	(Rumelhart et al., 1986)
Convolutional Neural Networks	Effective for pattern recognition	High computational cost, require large dataset	(LeCun et al., 1989)
Long Short-Term Memory (LSTM)	Captures long-term dependencies	High computational cost, require large dataset	(Hochreiter and Schmidhuber, 1997)
Gated Recurrent Unit (GRU)	Less computationally cost than LSTM	Still high computational cost, require large dataset	(Cho et al., 2014)
Transformer Models	Long-range dependencies, highly scalable	Extremely expensive, require large dataset	(Vaswani et al., 2017)
Temporal Convolutional Networks	Captures causal temporal patterns	High computational cost, require large dataset	(Bai et al., 2018)

Table 1. Comparison of Forecasting Algorithms. The models are categorized into three groups: Statistical, Machine Learning, and Deep Learning, as indicated by the horizontal lines separating them.

3.2.1 Naive

The Naive forecasting method is a simple approach that assumes the next value in a time series will be equal to the most recent observed value. It's easy to implement and works well for stationary time series with low-frequency behavior. However, it struggles with series that have trends or undergo abrupt changes, often leading to inaccurate predictions in such cases. Despite its limitations, the Naive method can be useful for quick estimates, especially for non-technical organizations predicting less critical variables.

3.2.2 ARIMA

The ARIMA (AutoRegressive Integrated Moving Average) model (Box and Jenkins, 1976) is a forecasting method that combines autoregression (AR), differencing (I), and moving average (MA) components to capture patterns in time series data. The model is formulated as ARIMA(p, d, q), where p represents the number of autoregressive terms, d is the degree of differencing, and q denotes the number of moving average terms. Common methods to help define the model's order include using autocorrelation functions (ACF) and partial autocorrelation functions (PACF) (Montgomery et al., 2015), while metrics such as information criteria like AIC (Akaike Information Criterion) and BIC (Bayesian Information Criterion) are often used when applying a grid search to find the most appropriate model (Alghamdi et al., 2019). The model's formulation is expressed as:

$$\hat{y}_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \dots + \phi_p y_{t-p} + \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2} + \dots + \theta_q \epsilon_{t-q}$$

where c is a constant, ϕ are the autoregressive coefficients, θ are the moving average coefficients, and ϵ_t represents the error term. Differencing is not illustrated in the ARIMA formulation explicitly, as it is a previous step that involves subtracting the previous observation from the current observation to remove trends in the data. The first-order differencing (Δy) is calculated as $\Delta y_t = y_t - y_{t-1}$.

3.2.3 Holt-Winters

The Holt-Winters method (Winters, 1960) is a three-component exponential smoothing technique designed for time series data exhibiting trend and seasonality. It updates the level (l_t), trend (b_t), and seasonality (s_t) components at each time step using the following equations:

$$l_t = \alpha(y_t - s_{t-m}) + (1 - \alpha)(l_{t-1} + b_{t-1})$$

$$b_t = \beta(l_t - l_{t-1}) + (1 - \beta)b_{t-1}$$

$$s_t = \gamma(y_t - l_t) + (1 - \gamma)s_{t-m}$$

where α is the smoothing parameter for the level, and s_{t-m} is the seasonal component from the same season m periods ago, β is the smoothing parameter for the trend and γ is the smoothing parameter for seasonality. For forecasting, the method combines these components:

$$\hat{y}_{t+h} = l_t + hb_t + s_{t+h-m(k+1)}$$

where h is the forecast horizon, k is the integer part of $(h - 1)/m$, and m is the seasonal period.

3.2.4 Comb

This technique improves accuracy and reduces bias by averaging predictions from multiple models (Bates and Granger, 1969). In this case, it combines forecasts from ARIMA and Holt-Winters models. The final forecast is calculated as the average of the two models' predictions:

$$\hat{y}_{Comb} = \frac{\hat{y}_{ARIMA} + \hat{y}_{Holt-Winters}}{2}$$

3.2.5 Regression Tree

This model is based on decision trees (Breiman et al., 1984) and used to predict continuous numerical values. It recursively partitions the input data into subsets based on feature values and assigns a regression value to each leaf node. The tree is built by selecting the best feature and threshold at each node to minimize a loss function, such as Mean Squared Error (MSE). Once the tree reaches a stopping criterion (e.g.,

maximum depth or minimum samples per leaf), the regression value for each leaf is set as the average of the target variable in that subset. Regression trees are non-parametric, interpretable, and can capture nonlinear relationships, making them useful for various datasets. However, they are prone to overfitting and require careful tuning to balance bias and variance.

3.2.6 Gradient Boosting

This method builds a strong predictive model by sequentially training weak models, such as decision trees, where each tree focuses on correcting the errors of the previous ones (Friedman, 2001). The process begins with an initial simple model and iteratively improves by adding new trees that target the residuals from the current model. Each new tree is trained to predict these residuals, and its predictions are added to the existing model. This addition is controlled by a learning rate, which determines the contribution of each tree to the final model.

Gradient Boosting also necessitates careful parameter tuning, such as the learning rate, number of trees, and tree depth. If a model is defined with excessive depth, it may struggle with overfitting and require significant computational resources.

3.2.7 Gaussian Process Regression

Gaussian Process Regression (GPR) is a non-parametric Bayesian approach that can make predictions that incorporate uncertainty (Williams and Rasmussen, 2006). It uses a kernel function $k(x, x')$ to measure the similarity between input points, such as the RBF kernel (Wang, 2023):

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{(x - x')^2}{2l^2}\right)$$

where σ_f^2 is the variance and l is the length scale.

For a dataset X and observed outputs $\mathbf{y}$, GPR computes the covariance matrix $K(X, X)$ and adds noise variance σ_n^2 to account for observation noise. This term is analogous to the Tikhonov regularization term used in linear regression solvers, which yields smoother and more reliable results (OM, 2001):

$$K(X, X) + \sigma_n^2 I$$

For a new input X_* , Gaussian Process Regression calculates the covariance vector $K(X_*, X) = [k(x_*, x_1)k(x_*, x_2) \dots k(x_*, x_n)]$ and covariance scalar $K(X_*, X_*) = k(x_*, x_*)$. These variables are used to predict the mean and variance of the new input:

$$\mu_* = K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1} \mathbf{y}$$

$$\Sigma_* = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1} K(X, X_*)$$

The final prediction is a Gaussian distribution:

$$p(f_* | X_*, X, \mathbf{y}) = \mathcal{N}(\mu_*, \Sigma_*)$$

GPR provides both predictions and uncertainty estimates. However, it requires tuning of kernel hyperparameters and can be computationally intensive for large datasets.

3.2.8 Recurrent Neural Networks

RNNs are a class of neural networks designed to process sequential data by maintaining a memory of previous inputs (Rumelhart et al., 1986). Unlike traditional feedforward networks, RNNs have connections that form directed cycles, allowing them to capture temporal dependencies in sequences. At each time step t , an RNN updates its hidden state $\mathbf{h}_t$ based on the previous hidden state $\mathbf{h}_{t-1}$ and the current input x_t (Ghojogh and Ghodsi, 2023):

$$\mathbf{h}_t = g_1(W_h \mathbf{h}_{t-1} + U_h x_t + \mathbf{b}_h)$$

where W_h and U_h are weight matrices associated with the hidden state and input, respectively, $\mathbf{b}_h$ is the bias vector, and g_1 is an activation function. The output $\hat{y}_t$ is computed as:

$$\hat{y}_t = g_2(V \mathbf{h}_t + b_y)$$

where V is the output weight matrix, b_y is the bias term, and g_2 is an activation function.

RNNs are used for tasks involving sequential data, such as time series prediction. However, they face challenges such as vanishing gradients, which have motivated the development of more advanced variants.

3.2.9 Long Short-Term Memory Networks

LSTM networks are a specialized variant of recurrent neural networks (RNNs) designed to overcome the vanishing gradient problem and capture long-term dependencies in sequential data (Hochreiter and Schmidhuber, 1997). They incorporate memory cells and gating mechanisms that regulate the flow of information, making LSTMs suitable for time series prediction with long dependencies.

At each time step t , an LSTM unit computes the input gate $\mathbf{i}_t$, forget gate $\mathbf{f}_t$, and output gate $\mathbf{o}_t$ as follows:

$$\mathbf{i}_t = \sigma(W_i \mathbf{h}_{t-1} + U_i x_t + \mathbf{b}_i)$$

$$\mathbf{f}_t = \sigma(W_f \mathbf{h}_{t-1} + U_f x_t + \mathbf{b}_f)$$

$$\mathbf{o}_t = \sigma(W_o \mathbf{h}_{t-1} + U_o x_t + \mathbf{b}_o)$$

These gates are typically computed using the sigmoid activation function σ , which outputs values between 0 and 1 and determines how much information is allowed to pass. The input gate $\mathbf{i}_t$ controls the addition of new information to the memory cell, the forget gate $\mathbf{f}_t$ determines what information is discarded from the previous memory cell $\mathbf{c}_{t-1}$, and the output gate $\mathbf{o}_t$ regulates the flow of information from the memory cell to the hidden state $\mathbf{h}_t$.

The memory cell $\mathbf{c}_t$ and hidden state $\mathbf{h}_t$ are updated as follows:

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(W_c \mathbf{h}_{t-1} + U_c x_t + \mathbf{b}_c)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$

where $\odot$ denotes element-wise multiplication. This mechanism allows LSTMs to capture both short-term and long-term dependencies in sequential data, making them suitable for tasks involving complex temporal patterns. For a more in-depth explanation, refer to Staudemeyer's work (Staudemeyer and Morris, 2019).

3.3 Hybrid Optimization

Optimization methods are employed when faced with a vast array of possible decisions that cannot be exhaustively analyzed due to the complexity of combinatorial problems. Every optimization problem is characterized by a feasible search space defined by constraints applied to decision variables. Instead of conducting an exhaustive search across the entire space, an approach that is often computationally infeasible, optimization methods propose a more intelligent strategy for evaluating the search space. These algorithms utilize the cost function, constraint information, or empirical experiments to guide the search process (Yang, 2008). These methods identify better solutions while minimizing computational resources, avoiding the need for an exhaustive exploration of the entire search space.

A traditional optimization problem is represented by an objective function and a set of constraints that define the feasible space of decision variables. This formulation is illustrated in the equations below:

$$\begin{aligned} \text{objective: } & \min / \max \quad f(\mathbf{x}; \mathbf{z}) \\ \text{s.t.: } & g_i(\mathbf{x}; \mathbf{z}) \leq 0, \quad \forall i \in \{1, 2, \dots, M\} \\ & h_j(\mathbf{x}; \mathbf{z}) = 0, \quad \forall j \in \{1, 2, \dots, N\} \\ & \mathbf{x} \in \mathbb{S}^D \end{aligned}$$

The optimization problem involves finding the values of the decision variables $\mathbf{x}$ that minimize (or maximize) the objective function $f(\mathbf{x}; \mathbf{z})$, where $\mathbf{z}$ represents data associated with the application, such as deadlines, workloads, penalties, or other fixed parameters of the problem. This process must satisfy all inequality constraints $g_i(\mathbf{x}; \mathbf{z}) \leq 0$ and equality constraints $h_j(\mathbf{x}; \mathbf{z}) = 0$. The decision variables $\mathbf{x}$ belong to the space $\mathbb{S}^D$, where D is the dimension of the decision space, while M and N correspond to the number of inequality and equality constraints, respectively. The goal is to identify the optimal solution within the feasible region defined by the constraints, ensuring that the objective function is optimized while adhering to all given conditions.

Traditional optimization faces challenges when decision variables or constraints are unknown. For instance, consider a constraint tied to the kilowatts produced by a solar plant over the next few months, a value that cannot be determined in advance. Similarly, decision variables may involve distributing funds across stock options, but the market's response and available funds the next day remain uncertain. These uncertainties limit the model, necessitating a reduction in the solution scope to focus on decisions associated with known constraints, when possible, resulting in suboptimal solutions. Addressing optimization problems under uncertain conditions is referred to as contextual optimization, a field that has recently gained attention (Sadana et al., 2025). However, much of the work in this area remains theoretical, with real-world applications of these methods often unclear.

In this scenario, full access to the constraints is unavailable. Without constraint information, a feasible space for optimization cannot be defined. Therefore, a forecasting formulation is incorporated to estimate the quantities associated with the constraint functions $g_i(\mathbf{x}; \mathbf{z})$ and $h_j(\mathbf{x}; \mathbf{z})$ of the system. In the present formulation, these quantities are incor-

porated as point forecasts, allowing the optimization problem to be defined in a direct and operationally simple manner. This choice facilitates the integration between forecasting and optimization and allows part of the uncertainty to be mitigated by incorporating anticipated future information into the optimization stage, although it does not fully remove uncertainty from the decision process. More elaborate formulations could also be adopted, such as robust optimization approaches (Bertsimas and Sim, 2004), in which forecast uncertainty is translated into conservative constraint values, for example by using lower bounds or safety margins derived from forecast intervals. Although such strategies may provide greater control over uncertainty, they also introduce additional design choices regarding the desired level of conservativeness. Since the proposed hybrid formulation is already sufficient to highlight the potential gains of integrating forecasting and optimization, these extensions are left outside the scope of the present work. Let $\mathbf{y}^{g_i} = [y_0^{g_i}, y_1^{g_i}, \dots, y_N^{g_i}]$ denote the actual historical data associated with the inequality constraint g_i , while $\hat{\mathbf{y}}^{g_i} = [\hat{y}_0^{g_i}, \hat{y}_1^{g_i}, \dots, \hat{y}_M^{g_i}]$ represents the forecasted estimations utilized to solve the optimization problem. The vector $\mathbf{p}^{g_i}$ is defined as the combination of these two sequences, thus making $\mathbf{p}^{g_i}$ a collection that includes both historical and forecast data:

$$\mathbf{p}^{g_i} = [\mathbf{y}^{g_i} \mid \hat{\mathbf{y}}^{g_i}]$$

$$\mathbf{p}^{g_i} = [y_0^{g_i}, y_1^{g_i}, \dots, y_N^{g_i}, \hat{y}_0^{g_i}, \hat{y}_1^{g_i}, \dots, \hat{y}_M^{g_i}]$$

When training and making predictions, slices of data containing the previous L values are used to predict the next step, where $0 < L < N$. Therefore, $\mathbf{p}_{a \rightarrow b}^{g_i}$ denotes a slice of the vector $\mathbf{p}^{g_i}$, specifically the sub-vector containing elements from index a to index b , where $0 \leq a < b$. This slice can be expressed as:

$$\mathbf{p}_{a \rightarrow b}^{g_i} = [p_a^{g_i}, p_{a+1}^{g_i}, \dots, p_b^{g_i}]$$

The forecast value $\hat{y}_{t+1}^{g_i}$ can be computed using the following formulation:

$$\hat{y}_{t+1}^{g_i} = F(\mathbf{p}_{t-L \rightarrow t}^{g_i}, \theta)$$

In this equation, $F(\mathbf{p}_{t-L \rightarrow t}^{g_i}, \theta)$ is a forecasting model parameterized by θ that maps the historical subset $\mathbf{p}_{t-L \rightarrow t}^{g_i}$, spanning from $t - L$ to t , to the forecast value $\hat{y}_{t+1}^{g_i}$.

Using these forecasting estimations, the objective function remains the same; however, the constraints are redefined as:

$$\begin{aligned} g_i(\mathbf{x}; \mathbf{z}, \hat{\mathbf{y}}^{g_i}) &\leq 0, \quad \forall i \in \{1, 2, \dots, M\} \\ h_j(\mathbf{x}; \mathbf{z}, \hat{\mathbf{y}}^{h_j}) &= 0, \quad \forall j \in \{1, 2, \dots, N\} \end{aligned}$$

The constraint functions $g_i(\mathbf{x}; \mathbf{z}, \hat{\mathbf{y}}^{g_i})$ and $h_j(\mathbf{x}; \mathbf{z}, \hat{\mathbf{y}}^{h_j})$ incorporate the forecasted values to define the constraints, ensuring that the solution $\mathbf{x}$ adheres to the forecast trends or patterns.

Numerous optimization methods are available (Yang, 2008). In this work, an exact optimization solver (SCIP (Achterberg, 2009)) was applied to evaluate how forecasting information can improve the quality of the resulting decisions under the proposed formulation. Computational time

is also reported to provide an empirical view of the cost associated with each method, while broader scalability aspects are discussed later. A visual representation of the process flow, linking the mathematical formulation to its practical implementation, is provided in Figure 2. The proposed hybrid approach is composed of two main parts: a forecasting model, which uses historical constraint values to estimate future quantities, and an optimization model, which uses these estimates together with the application variables to derive the final decision x^* .

4 Experimental Setup

This section presents a practical solution to a common challenge while offering readers a relatable example of the application of data-driven solutions. Three distinct scenarios are then examined for comparative evaluation: an idealized case where all information is available, a purely optimization-based approach, and the proposed hybrid architecture. Each method is implemented as detailed in the case study. Finally, the dataset used for evaluation is explained. A more detailed step-by-step pseudocode of the algorithm is provided in Appendix A.

4.1 Case Study

This study tackles a common challenge in decision-making: resource allocation. To provide a practical and relatable context, the problem is framed within the framework of agile methodologies, which differ significantly from traditional cascade project management approaches. Agile methodologies emphasize breaking tasks into the smallest possible units that still deliver value to the product. These tasks are organized in a backlog, a prioritized list of work items awaiting allocation to specific sprints. Mature agile teams further refine this process by assigning effort estimates to each task, often using the Fibonacci sequence to represent the time required for completion. Additionally, each team member is associated with a capacity value, indicating the number of hours they can contribute per sprint. While agile methodology is well-documented (see, for example, Sutherland (2014)), a detailed explanation of its principles is beyond the scope of this study.

The core objective is to allocate resources by selecting the decision variable x (representing different tasks) in a way that minimizes delivery lateness. In terms of the generic formulation presented in Subsection 3.3, the deadlines, workloads, unscheduled-task penalties, and planning horizon correspond to the known application data represented by z . This is formalized as the following optimization problem, where each task contributes a penalty that grows linearly with the number of sprints between its deadline and its actual allocation, yielding a cumulative tardiness measure:

$$Z_{\text{tardiness}} = \min \left(\sum_{i=1}^n \left(\sum_{t=T_i+1}^d x_{i,t} \cdot (t - T_i) + \text{Missing}(i) \right) \right)$$

where $x_{i,t}$ indicates whether task i is assigned to sprint t , $T_i \in \mathbb{N}$ is the deadline of task i , and $d \in \mathbb{N}$ is the total number of planned sprints. Since the inner summation starts at

$t = T_i + 1$, tasks allocated at or before their deadline do not contribute to the tardiness term; only delayed allocations are penalized. Additionally, $W_i \in \mathbb{R}_{>0}$ denotes the workload (or estimated duration) of task i , $C_t \in \mathbb{R}_{\geq 0}$ denotes the total available team capacity in sprint t , and $M_i \in \mathbb{R}_{\geq 0}$ denotes the penalty associated with leaving task i unscheduled. Since the objective function focuses on minimizing lateness penalties, the model could trivially avoid scheduling tasks to reduce costs. To prevent this behavior, an additional penalty term is introduced for unscheduled tasks:

$$\text{Missing}(i) = M_i \cdot \left(1 - \sum_{t=1}^d x_{i,t} \right)$$

In this study, all tasks are considered equally important. Therefore, M_i is taken as a constant sufficiently large to ensure that the optimizer prefers scheduling a task late rather than not scheduling it at all. The model is also subject to the following constraints:

Each task can be scheduled at most once:

$$\sum_{t=1}^d x_{i,t} \leq 1 \quad \forall i \in \{1, \dots, n\}$$

The total workload assigned to a sprint must not exceed the team's capacity:

$$\sum_{i=1}^n W_i \cdot x_{i,t} \leq C_t \quad \forall t \in \{1, \dots, d\}$$

Decision variables must be binary:

$$x_{i,t} \in \{0, 1\} \quad \forall i \in \{1, \dots, n\}, \forall t \in \{1, \dots, d\}$$

This optimization must be performed under constraints defined by the total capacity of the team members, which is itself subject to uncertainty. While team capacity is estimated, it is inherently inconsistent over time due to fluctuations in individual performance and external influences. These uncertainties are modeled as part of the problem, reflecting real-world complexities in resource allocation.

The problem is first solved in its ideal form, where capacity is treated as a deterministic known value. This scenario represents an unrealistic assumption, as it ignores uncertainties associated with resource availability. A more realistic scenario is then considered, where capacity is not directly observable and must be treated as an unknown variable. In this case, the problem is addressed using both traditional optimization techniques and the proposed contextual optimization architecture. These evaluations provide a concrete assessment setting for the proposed architecture, using a relevant operational decision-making problem to compare its behavior against a baseline approach and an idealized benchmark. The goal is not to provide an exhaustive comparison across all possible methods for optimization under uncertainty, but rather to examine how the proposed integration between forecasting and optimization behaves in a practical allocation setting. The scenarios are described as follows:

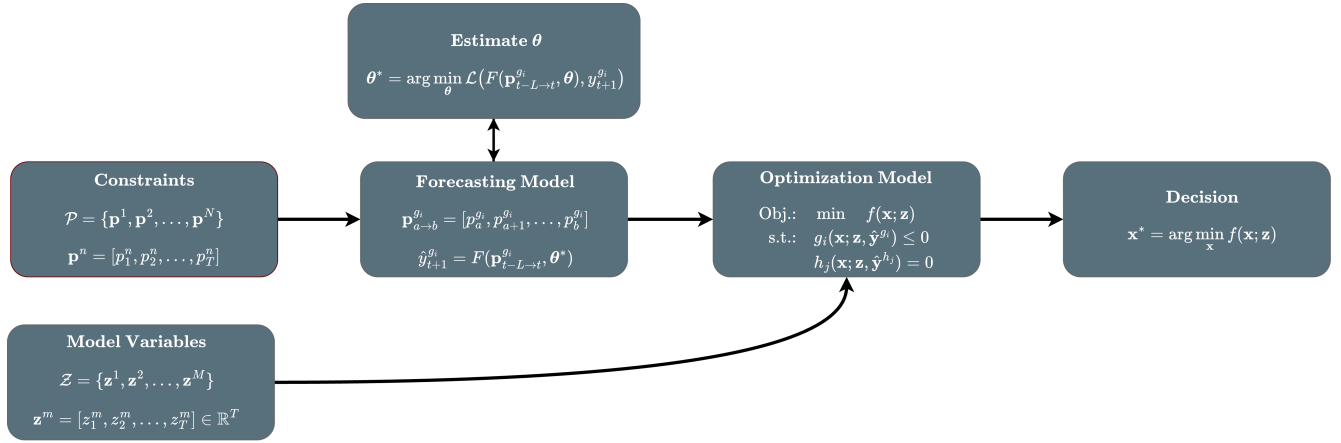


Figure 2. Forecasting-Based Optimization Diagram.

4.1.1 Ideal Scenario

Optimization algorithms can produce optimal decisions over the long term when all relevant data is available. For example, if an optimizer aims to allocate a set of activities over the next few days, given a known capacity for each time step, the system can be modeled, and an optimization method applied to achieve the best possible outcome under these constraints. This scenario assumes complete knowledge of the constraints. This solution serves as a benchmark or "golden standard" for assessing how resources should be allocated across N time steps to achieve optimal performance.

4.1.2 Traditional Optimization

In many practical scenarios, the constraints are unknown. Without this information, it becomes impossible to use an optimizer to guarantee the optimal configuration. In such cases, the problem's scope must be restricted to only the information that is known, leading to the identification of a suboptimal solution. Instead of defining configurations for the entire period, the focus shifts to making decisions on a per-time-step basis as new information becomes available. This approach is useful when future resource information is unavailable. The core idea is to narrow the decision-making scope by focusing on the constraints available for the current time step, rather than seeking the optimal configuration for the entire period. This involves allocating tasks for that specific time step, resulting in a locally optimal allocation each time step. While this method does not guarantee global optimality, it ensures the best available solution given the information available at the time of decision-making.

4.1.3 Hybrid Architecture

The proposed architecture (shown in Figure 3) aims to improve outcomes in scenarios where future system parameters are unknown. Its central idea is to combine optimization algorithms with forecasting techniques so that decisions can be made using both current information and estimates of future system behavior. Historical data is used to predict the values associated with future constraints, and these estimates are incorporated into the optimization model to define an allocation plan over the decision period. However, at each iteration,

only the decision associated with the current time step is implemented. Once the next time step is reached and new system information becomes available, the forecasting model is retrained using the updated historical data, the forecast is refreshed, the remaining items are re-evaluated, and the optimization procedure is solved again. In this way, the method continuously adapts its decisions as new observations are incorporated.

In this approach, decisions are repeatedly recomputed as the system evolves, rather than being fixed once for the entire period. This differs from the traditional method, which relies only on the information available at the current time step and therefore restricts the scope of optimization to immediate conditions. By incorporating estimated future information, the proposed architecture defines configurations that may better approximate the ideal scenario, in which all future constraints are known in advance. The quality of the resulting decisions remains dependent on forecasting accuracy, since errors in the predicted values may propagate to the optimization stage. Even so, the approach provides a practical mechanism for improving decision-making in uncertain environments by allowing the allocation process to account for both present conditions and expected future behavior.

4.2 Dataset

The method is evaluated as a proof of concept using a synthetically generated dataset, designed to provide greater control over its characteristics and ensure reproducibility. The dataset is divided into two distinct types of data:

4.2.1 Project Backlog

This dataset includes individual tasks associated with projects that must be allocated while respecting the system's constraints. Each task is characterized by a due date and the estimated time required for execution. This setup is designed to represent the demand activities from a specific period within a company. An example of this data is provided in Table 2. The execution time is a positive real value sampled from random distributions with a closed interval, where the lower limit is a and the upper limit is b . The due dates are drawn from a multi-modal Gaussian distribution, which

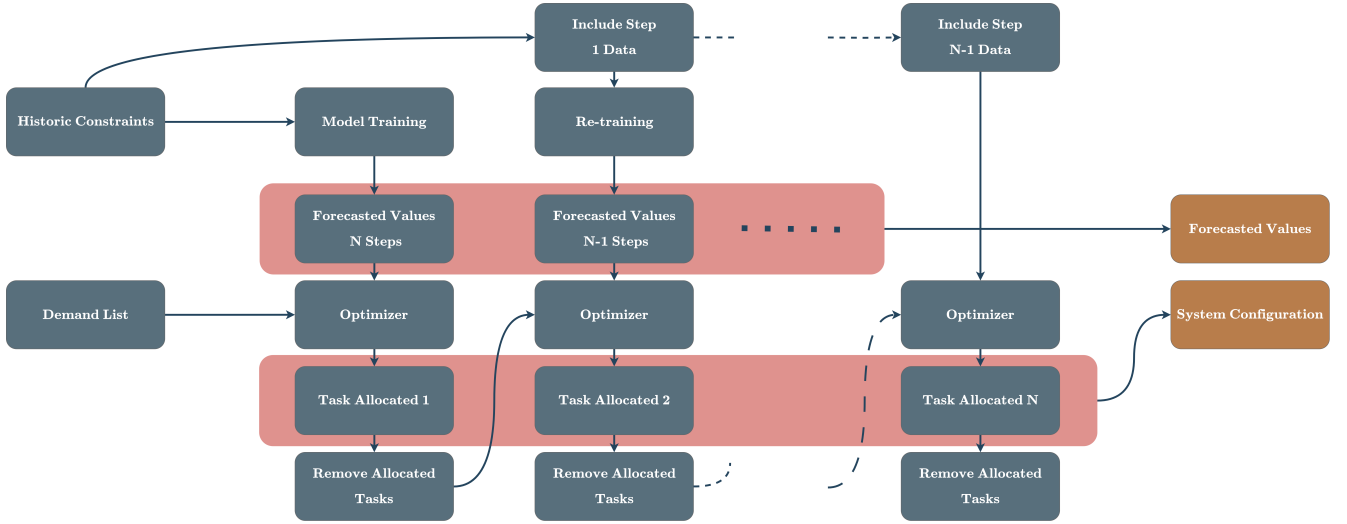


Figure 3. Hybrid Architecture for Resource Allocation.

reflects the demand with peaks and valleys in different time periods, depending on the centers of the Gaussians.

ID	Execution Time	Deadline
1	5h	1/3/2024
2	1.5h	1/9/2025
3	2h	1/8/2025
4	1h	1/8/2025

Table 2. List of items sampled to represent the project backlog. The first column represents a unique ID for each item; the second column defines how long each task takes to complete; the third column indicates the expected date when each task is scheduled for delivery.

Sampling from a random distribution is straightforward. However, for a multi-modal distribution, the process begins by defining the Gaussian probability density function for a single variable x , with mean μ and standard deviation σ , which is given by:

$$f(x | \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

The multi-Gaussian PDF combines multiple Gaussian distributions with different means, standard deviations, and weights. It is defined as:

$$f(x | \mu, \sigma, w) = \sum_{i=1}^n w_i \cdot f(x | \mu_i, \sigma_i)$$

where:

- $\mu = \{\mu_1, \mu_2, \dots, \mu_n\}$ is the set of means.
- $\sigma = \{\sigma_1, \sigma_2, \dots, \sigma_n\}$ is the set of standard deviations.
- $w = \{w_1, w_2, \dots, w_n\}$ is the set of weights.
- $f(x | \mu_i, \sigma_i)$ is the Gaussian PDF for the i -th component.

To ensure the generated data fall within a specified period, the multi-Gaussian probability density function (PDF) can be truncated to the interval $[a, b]$ by setting:

$$f(x | \mu, \sigma, w, a, b) = \begin{cases} f(x | \mu, \sigma, w) & \text{if } a \leq x \leq b \\ 0 & \text{otherwise} \end{cases}$$

The cumulative distribution function (CDF) is defined as:

$$F(x) = \frac{1}{C} \int_a^x f(t) dt$$

where $C = \int_a^b f(x) dx$ is the normalization constant, which is essential. This is because, in the process of generating the multi-modal distribution, multiple distributions are combined, and unwanted values are truncated, making the final function an invalid probability distribution since its integral does not converge to one.

Now, $F(x)$ is substituted with a random distribution $u \sim \mathcal{U}(0, 1)$, and x is isolated. For every random sample u , there is an associated x . Given a large number of samples, the distribution of x will approximate the same distribution as the previously modeled multi-Gaussian $F(x)$.

$$x = F^{-1}(u)$$

where F^{-1} is the inverse of the CDF.

4.2.2 Constraints History

These variables are unknown during real-time decision-making. Constraints represent any limitations that must be considered when making decisions. While multiple constraints can exist in a single problem, this dataset presents only one limitation, which can be represented as the amount of resources available. In this case, it is the team capacity available per time step for executing tasks. When deciding which items to allocate, this constraint must be respected, as the task allocation may not exceed this limit. A time series is constructed by summing three components: trend, seasonality, and noise. The general form of the time series is:

$$y = \text{Trend} + \text{Seasonality} + \text{Noise}$$

4.2.3 Trend

Represents the long-term movement or direction of the series, where β_0 is the series offset, and β_1 is the slope of the trend. This component is important since some models may present poor performance on non-stationary data

$$\text{Trend} = \beta_0 + \beta_1 t$$

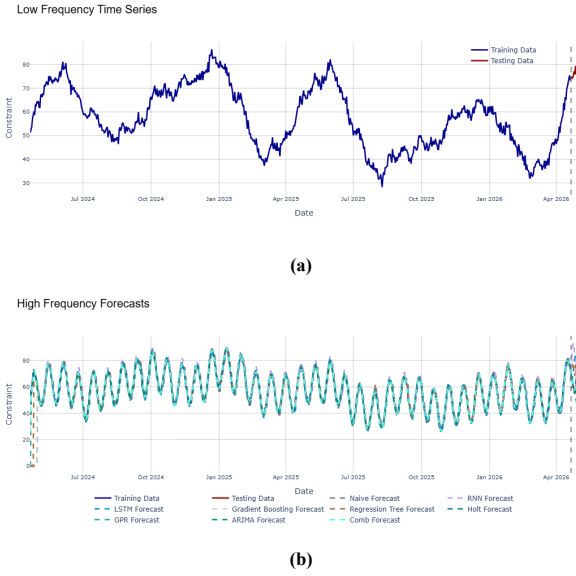


Figure 4. Constraint history generated from Trend, Seasonality and Noise formulation: (a) Low frequency time-series. (b) High frequency time-series.

4.2.4 Seasonality

Captures repeating patterns in the data, such as daily, weekly, or yearly cycles. It can be modeled using sine or cosine functions, where A_i and B_i are amplitudes, m_i is the seasonal period, and k is the number of seasonal components.

$$\text{Seasonality} = \sum_{i=1}^k A_i \sin\left(\frac{2\pi t}{m_i}\right) + B_i \cos\left(\frac{2\pi t}{m_i}\right)$$

4.2.5 Noise

Represents random fluctuations or residuals in the series, where $\mathcal{N}(0, \sigma^2)$ denotes a normal distribution with mean 0 and variance σ^2 .

$$\text{Noise} \sim \mathcal{N}(0, \sigma_n^2)$$

This evaluation can be extended by applying nonlinear trends, phase shifts in seasonality, and other characteristics to increase diversity. However, this setup provides sufficient customization and control to evaluate the methodology. Table 3 contains the parameters used in the data synthesis process for both low and high-frequency tests, presented for reproducibility purposes. It is important to note that the findings are not restricted to these specific parameters, and minor changes would likely maintain the same overall interpretation of the results. However, it is interesting to maintain an overloaded system; otherwise, the comparative results may have very similar metrics, as every task would be allocated correctly regardless of the constraint.

The data associated with the constraint is visualized in the plot of Figure 4. As shown, a time series with high seasonality frequency and another with low frequency were generated to evaluate the different results in each case. As Makridakis noted, different forecasting behaviors are observed in each scenario (Makridakis et al., 2020).

5 Results

The results are organized into two complementary analyses. First, a single-run analysis is presented to examine in greater detail how each forecasting algorithm behaves, how the lateness cost accumulates over time, and how the final performance is obtained. This analysis provides a more intuitive understanding of the internal behavior of the proposed methodology. Second, a multiple-run analysis is conducted to provide a more robust statistical evaluation. Instead of examining only one instance of the problem, multiple simulations are performed to assess how performance varies across runs, how consistently the forecast-based approach outperforms the standard method, the average magnitude of the improvement, and how likely this improvement is to occur.

5.1 Single-Run Analysis

The three approaches outlined in the Case Study section are compared using the dataset described earlier in order to evaluate the performance of the hybrid architecture. Since the ideal scenario provides the optimal solution, the main objective is to assess how closely the hybrid method approaches this reference and whether it outperforms the traditional approach in both the low- and high-frequency datasets. Figure 5 presents the results for the ideal and traditional scenarios, together with the outcomes of the hybrid method when combined with different forecasting algorithms. The figure reports the lateness score, which must be minimized, on the y-axis after the optimization process, while the x-axis identifies the evaluated methods. The results indicate that, in the scenarios considered, the hybrid architecture outperforms the traditional approach. In some cases, depending on the forecasting algorithm employed, the hybrid architecture achieves performance equal to that of the ideal case. Overall, the plot indicates that all forecasting algorithms evaluated in this study improve system performance relative to the traditional approach, although the magnitude of the improvement varies across methods.

The cumulative lateness of the allocated items was also evaluated at each time step. Notably, the traditional approach exhibits greater accumulated lateness than the other methods, while the ideal and hybrid approaches produce more similar, and in some cases identical, results. Figure 6 illustrates the cumulative lateness as time progresses.

Before the optimization process, the forecasting algorithms are executed to predict the next N constraint values. For a more detailed analysis, Figures 7 and 8 present the target values alongside the predictions generated by each algorithm. As described in the architecture flow, the forecasting procedure is repeated multiple times; however, for illustrative purposes, only the first forecasting step is shown. In addition, the figures display the error between the true values and the corresponding predictions, providing further insight into forecast accuracy. The plots include both in-sample (training) and out-of-sample (testing) predictions. Several observations can be made from Figures 7 and 8:

- **Performance:** While several machine learning models achieved lower in-sample errors, the statistical methods

Variable	Description	Value
$n_{historic}$	Number of historical days generated for training	730
$n_{forecast}$	Number of forecasted days generated for testing	10
n_{items}	Number of items	100
μ	Multi-modal Gaussian distribution mean	$\{3, 7\}$
σ	Multi-modal Gaussian distribution standard deviation	$\{1, 3\}$
w	Multi-modal Gaussian distribution weights	$\{1.0\}$
a	Sampling truncation lower limit	0
b	Sampling truncation upper limit	10
β_0	Trend component offset	50
β_1	Trend component slope	0.02
A_h	Sine seasonal amplitudes (high-frequency)	$\{2, 7, 13, 15, 2, 2\}$
A_l	Sine seasonal amplitudes (low-frequency)	$\{15, 7, 13, 2, 2, 2\}$
m_l	Senoidal seasonal periods	$\{180, 120, 750, 20, 40, 80\}$
B	Cosine seasonal amplitudes	-
σ_n^2	Noise standard deviation	1.5

Table 3. Description and definition of the parameters and their corresponding values used in the simulations.

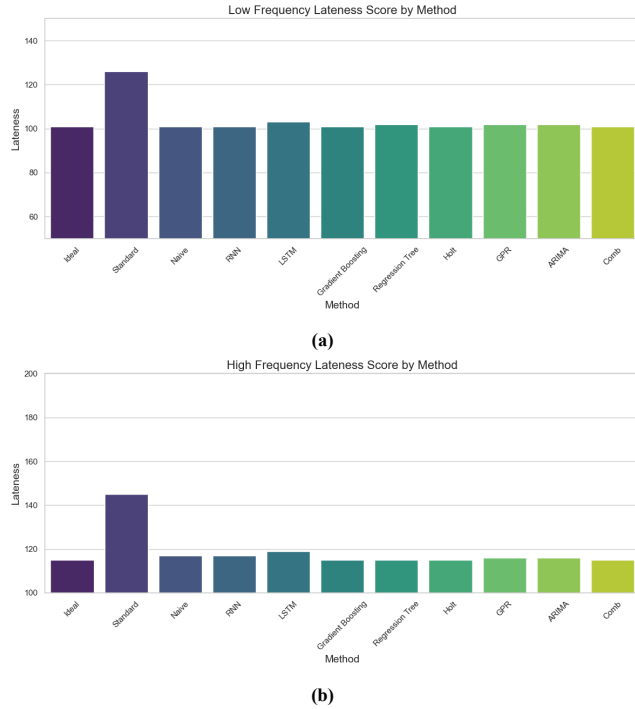


Figure 5. Bar plot comparing lateness score ($Z_{tardiness}$) after the optimization process using the ideal scenario, traditional approach and the hybrid architecture evaluated with several forecasting algorithms: (a) Low-frequency scores. (b) High-frequency scores.

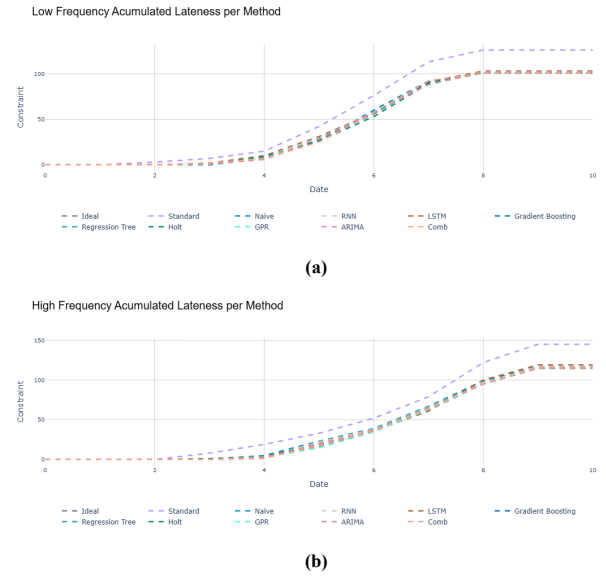


Figure 6. Cumulative lateness score ($Z_{tardiness}$) per time step of each optimization method: (a) Low-frequency cumulative value. (b) High-frequency cumulative value.

provided better and more consistent out-of-sample forecasts in the scenarios considered here. Holt, ARIMA, and Comb remained more robust on unseen data, whereas more flexible models such as RNN, LSTM, Gaussian Process, and Gradient Boosting showed larger generalization errors. This suggests that, in the present setting, simpler models were better matched to the structure of the series and offered a more favorable performance. Machine learning approaches may become more competitive when larger datasets or additional explanatory variables, such as categorical or other exogenous information, are available. Figure 9 presents a cleaner plot, showcasing only the target values and the best-performing forecasting algorithm.

- **Frequency:** The algorithms were evaluated on two time series with distinct frequency characteristics. Although additional simulations would be required to draw broader conclusions, the results indicate that some methods are more sensitive to this change than others. In particular, the Naive method is directly affected, since it assumes that future values remain equal to the most recently observed value. This assumption is more reasonable in the low-frequency series, but becomes less accurate in the high-frequency series case, where the signal varies more abruptly. More flexible machine learning models also showed some degradation under the high-frequency setting, suggesting that their configuration may need to be adjusted according to the temporal structure of the series.
- **Overfitting:** Most methods followed the target values closely on the training data. However, several machine learning models exhibited larger out-of-sample errors than in-sample errors, indicating limited generalization in this setting. In contrast, the statistical approaches remained more stable on unseen data. This is a common issue in forecasting, where models that fit the training data well may still fail to generalize to unseen observations in practice.
- **Error Accumulation:** Time series forecasting inherently involves uncertainty, and the fitted model is rarely a perfect representation of the underlying process. As a result, forecast errors tend to accumulate over the prediction horizon. Although a method may predict the next value accurately, errors introduced at earlier steps propagate to subsequent predictions, causing the forecasts to gradually diverge from the target values. This effect is evident in the out-of-sample error plots, where the error generally increases as the forecasting horizon becomes longer.

As shown in Figure 5, the hybrid architecture outperforms the traditional optimization approach in the evaluated scenarios and, in some cases, achieves results equal or very close to the ideal solution. Even when the forecasted values are not perfectly accurate, the architecture is still able to improve the final score. However, a perfect match with the ideal solution can only be guaranteed when the forecasted values coincide with the true future values. Another observation is that higher forecasting accuracy does not necessarily lead to a better optimization outcome. Therefore, although the hybrid architec-

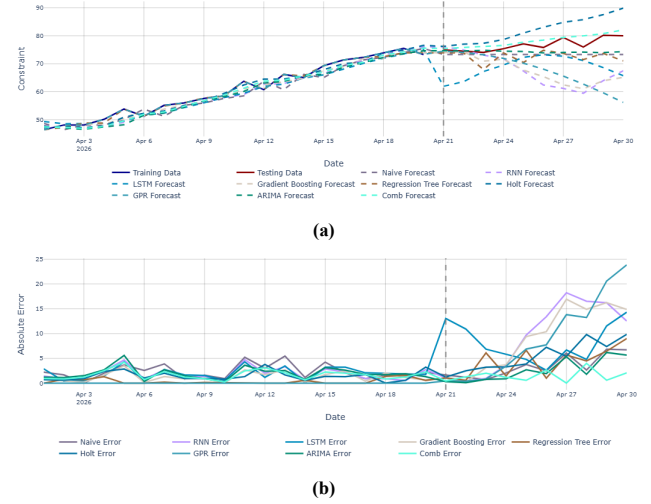


Figure 7. (a) Plotting the low-frequency time series forecast values of each algorithm using both in-sample and out-of-sample data; **(b)** Plotting the error of each forecasting algorithm. (The vertical line separates the training and testing data).

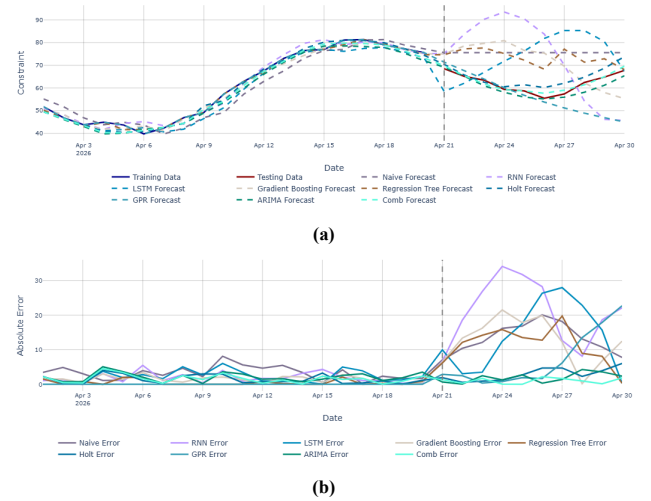


Figure 8. (a) Plotting the high frequency time series forecast values of each algorithm using both in-sample and out-of-sample data; **(b)** Plotting the error of each forecasting algorithm. (The vertical line separates the training and testing data).

ture consistently improves upon the traditional approach, the relationship between forecasting performance and optimization quality is not direct.

To support the previous observation, Tables 4 and 5 summarize the forecasting errors, the final optimization cost, and the runtime of each method. The reported runtime encompasses the complete rolling-horizon procedure, including model training, forecasting, and solving the optimization problem at each step. In both tables, the Comb method presents the best forecasting accuracy and also achieves an optimization score equal to the ideal scenario. This result follows the intuition that better forecasts may lead to a better optimization score. However, this relationship is not strictly monotonic. For example, Gradient boosting and Regression tree present worse forecasting accuracy than the Naive method, but still achieve better optimization scores. Therefore, due to the non-differentiable nature of the optimization process, it is not trivial to associate forecasting error with the final optimization score.

Beyond improving the optimization results, forecasting

Method	Score	In-sample Error	Out-of-sample Error	Runtime
Ideal	101.0	-	-	2.41
Standard	126.0	-	-	0.06
Naive	101.0	1.72 ± 1.31	2.04 ± 1.95	5.97
Holt	101.0	1.22 ± 0.95	4.51 ± 2.79	7.63
ARIMA	102.0	1.49 ± 1.10	2.50 ± 2.18	34.53
Comb	101.0	1.29 ± 0.97	1.17 ± 1.06	34.65
Gradient Boosting	101.0	1.06 ± 0.81	6.88 ± 6.36	9.15
Regression Tree	102.0	0.36 ± 0.57	2.46 ± 2.02	3.51
Gaussian Process	102.0	0.00 ± 0.00	7.98 ± 7.45	931.64
RNN	101.0	1.68 ± 1.23	7.48 ± 6.49	203.41
LSTM	103.0	1.74 ± 1.33	6.15 ± 3.54	307.04

Table 4. Performance metrics of each method on low-frequency data: the first column lists the method name; the second column shows the optimization score ($Z_{tardiness}$); the third and fourth columns present the mean and standard deviation of the in-sample (training) and out-of-sample (testing) errors, respectively; and the last column reports the runtime.

Method	Score	In-sample Error	Out-of-sample Error	Runtime
Ideal	115.0	-	-	3.05
Standard	145.0	-	-	0.08
Naive	117.0	3.31 ± 2.15	6.20 ± 4.20	3.08
Holt	115.0	1.21 ± 0.98	1.66 ± 1.39	14.05
ARIMA	116.0	1.48 ± 1.13	2.16 ± 1.49	49.06
Comb	115.0	1.27 ± 1.00	0.97 ± 0.69	52.60
Gradient Boosting	115.0	1.31 ± 1.00	11.59 ± 7.84	5.98
Regression Tree	115.0	0.33 ± 0.60	9.98 ± 6.77	2.03
Gaussian Process	116.0	0.00 ± 0.00	17.48 ± 15.23	599.60
RNN	117.0	1.91 ± 1.45	14.36 ± 9.94	275.97
LSTM	119.0	1.81 ± 1.43	14.17 ± 10.07	424.19

Table 5. Performance metrics of each method on high-frequency data: the first column lists the method name; the second column shows the optimization score ($Z_{tardiness}$); the third and fourth columns present the mean and standard deviation of the in-sample (training) and out-of-sample (testing) errors, respectively; and the last column reports the runtime.

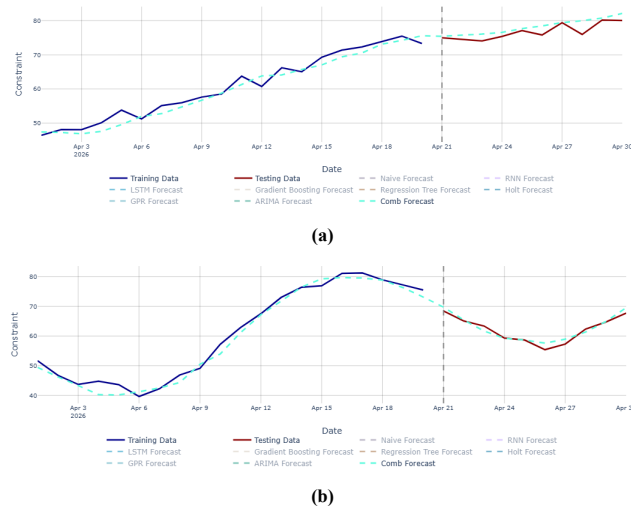


Figure 9. Showcasing in-sample and out-of-sample forecasts of the most accurate model and comparing with the target values: (a) Low-frequency best model. (b) High-frequency best model.

methods are also useful for building dashboards to monitor critical variables over time. In the present evaluation, classical approaches such as ARIMA and Holt-Winters appeared to be the most suitable options, as they required relatively little tuning while providing competitive predictive performance. In addition, because these methods have modest computational cost, a grid search over their parameter orders can be carried out with little difficulty.

5.2 Multiple-Run Analysis

To provide a more complete analysis, the evaluation is extended beyond a single execution and repeated over multiple runs. In addition to the stochasticity already present in the synthetic data generation process, extra variability is introduced in the magnitude of the constraint time series by perturbing both the trend and the seasonal components within a range of $\pm 5\%$ of their original values. This creates a more stressful setting, allowing the analysis to better reflect how the methods behave under small but relevant changes in the monitored process. In total, 50 independent simulations are performed in order to collect a sufficiently large sample for statistical analysis. In each run, every method is executed independently, and two quantities are recorded for further evaluation: the final lateness score and the computational time required by the method. This allows not only a comparison

in terms of decision quality, but also an analysis of the computational cost associated with obtaining such decisions.

The aggregated results are presented in Table 6, which reports the mean and standard deviation of the lateness score and execution time for each method across all runs. As expected, every forecast-based method outperforms the standard approach and remains considerably closer to the ideal scenario. This result is important, since it indicates that the benefit of incorporating forecasting is not restricted to a single favorable execution, but is instead maintained under repeated perturbations of the problem. The computational time varies substantially among the forecast-based methods. Simpler forecasting models remain relatively inexpensive, whereas more complex models, especially LSTM, RNN, and GPR, require significantly more execution time. This higher computational cost may limit their practicality in larger-scale applications or settings requiring frequent model updates.

To further investigate the behavior of each method across repeated runs, the deviation of its lateness score from the ideal solution is analyzed. More specifically, for each method and each run, the deviation is computed as the difference between the method's lateness score and the corresponding ideal lateness score. Smaller values indicate solutions closer to the ideal scenario, whereas larger values indicate worse performance. These deviations are visualized in Figure 10. For visualization purposes, a kernel density estimate (KDE) is fitted for each method in order to provide a smooth approximation of the empirical distribution across the 50 runs. Although the KDE should be interpreted only as a descriptive aid, it helps reveal the relative location and spread of the deviations. The traditional approach is shifted toward larger deviation values, indicating a higher probability of producing solutions farther from the ideal, whereas the forecast-based methods remain more concentrated near zero.



Figure 10. Distribution of the deviation from the ideal solution across multiple runs. Values closer to zero indicate better performance.

Based on these observed differences, a statistical hypothesis test is also conducted to evaluate whether forecast-based decision-making provides a consistent advantage over the traditional approach. Instead of analyzing each forecasting model separately, the forecasting methods are considered jointly. For each run r , the average improvement over the traditional approach, normalized by the ideal score, is defined as

$$\Delta_r = \frac{1}{K} \sum_{k=1}^K \frac{Z_r^{\text{tra}} - Z_r^{(k)}}{Z_r^{\text{ideal}}} \times 100,$$

where K is the number of forecasting methods, Z_r^{tra} and

Z_r^{ideal} are the lateness scores of the traditional and ideal approaches in run r , respectively, and $Z_r^{(k)}$ is the lateness score of the hybrid architecture using forecasting method k . This aggregation is used because the goal is not to test each forecasting algorithm individually, but to evaluate whether the forecast-based architecture consistently improves the results in the proposed experimental setting when forecasting information is incorporated. This run-level quantity is then used to assess, across a sequence of target improvement percentages τ , whether the mean improvement remains greater than the defined threshold.

For each target improvement percentage τ , a one-sided one-sample t -test (Montgomery and Runger, 2010) is performed to assess

$$H_0 : \mathbb{E}[\Delta_r] \leq \tau \quad \text{against} \quad H_1 : \mathbb{E}[\Delta_r] > \tau.$$

The test statistic is computed as

$$t = \frac{\hat{\mu}_\Delta - \tau}{\hat{\sigma}_\Delta / \sqrt{R}},$$

where $R = 50$ is the number of independent runs, $\hat{\mu}_\Delta = \frac{1}{R} \sum_{r=1}^R \Delta_r$ is the sample mean of the run-level improvements, and $\hat{\sigma}_\Delta = \sqrt{\frac{1}{R-1} \sum_{r=1}^R (\Delta_r - \hat{\mu}_\Delta)^2}$ is the sample standard deviation estimated across runs. This test assumes that the simulation runs are independent and that the sampling distribution of the mean improvement is approximately normal. The resulting p -values are shown in Figure 11, which illustrates how the statistical evidence changes as the required improvement percentage increases. The null hypothesis that the mean improvement is 20% or less is rejected at the 5% significance level, with a p -value of 0.044, indicating that the average gain exceeds this threshold. The results provide statistical evidence that the forecast-based architecture improves over the traditional method in the proposed synthetic experiment. Although the simulation runs are independent, the same set of runs is used for all values of τ , so the p -values along the curve should not be interpreted as independent tests.

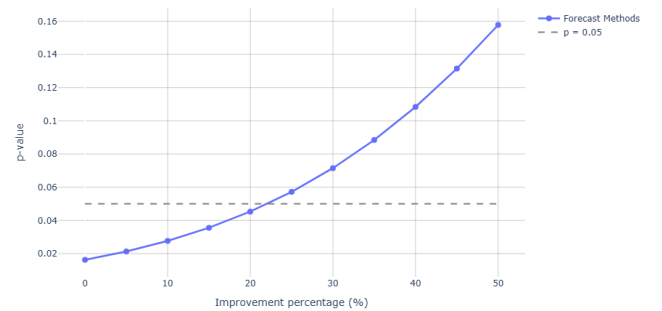


Figure 11. One-sided one-sample t -test p -values for testing whether the mean improvement of the forecast-based architecture, considered as a group, exceeds each target improvement percentage τ over the traditional approach. The x -axis shows the target improvement percentage τ , and the y -axis shows the corresponding p -value. The solid curve represents the p -value as a function of τ , while the dashed horizontal line marks the 0.05 significance level.

Metric	Ideal	Standard	Naive	Holt	ARIMA	Comb
Lateness Score	125.12 $\pm$ 74.33	152.82 $\pm$ 70.02	126.40 $\pm$ 74.66	126.48 $\pm$ 74.64	126.34 $\pm$ 74.56	126.46 $\pm$ 74.53
Computational Time	2.95 $\pm$ 3.61	0.07 $\pm$ 0.01	35.91 $\pm$ 191.94	9.35 $\pm$ 12.74	35.94 $\pm$ 12.02	39.25 $\pm$ 9.49

Metric	Gradient Boosting	Regression Tree	Gaussian Process	LSTM	RNN
Lateness Score	126.38 $\pm$ 74.78	126.68 $\pm$ 74.58	128.22 $\pm$ 74.41	126.46 $\pm$ 74.20	126.66 $\pm$ 74.71
Computational Time	9.83 $\pm$ 16.83	4.01 $\pm$ 5.15	604.41 $\pm$ 2095.44	438.14 $\pm$ 825.34	218.36 $\pm$ 31.29

Table 6. Mean and standard deviation of the lateness score and computational time across multiple runs.

6 Discussion

This paper contributes to understanding how intelligent automated systems can be structured in practice, connecting organizational process needs and implementation concerns from a shared perspective. Rather than treating forecasting and optimization as isolated techniques, this paper uses the case study to illustrate how these components can be integrated into a decision-making pipeline. In this way, the contribution of the paper is not restricted to the performance of a specific method, but also to the clarification of how data-driven and optimization-based components may be combined to support operational decisions in practical environments.

The proposed architecture does not eliminate uncertainty, since forecast errors remain an inherent limitation of any predictive system. The use of future estimates allows the optimization stage to anticipate trends and reduce part of the uncertainty, therefore the quality of the final allocation still depends on the reliability of the forecasting stage. This issue should become more relevant as the planning horizon increases, since uncertainty naturally accumulates over time. In this work, point forecasts were adopted as a practical mechanism to connect prediction and optimization. However, more elaborate treatments of uncertainty, such as prediction intervals or tolerance margins, may further improve the resulting decisions and represent important directions for future research.

Another important point concerns the scope and external validity of the experimental evaluation. Since datasets for this type of decision problem are not easily available in a public and structured form, the analysis was conducted as a proof of concept using a synthetic dataset designed to reproduce stochastic variations in capacity while preserving controlled and repeatable conditions. To reduce the dependence on a single realization, the study was extended to multiple runs, allowing a broader statistical analysis of the behavior of the methods under perturbed scenarios. Even so, the experimental setting remains a simplified representation of real organizational contexts, and the results should not be interpreted as a full validation of the architecture in real-world deployments. In addition, other optimization approaches for uncertain environments were not explored in this study. The comparative setting adopted here was designed to provide a clear operational contrast between an ideal full-information benchmark, a traditional optimization baseline, and the proposed forecast-informed architecture. Therefore, the goal was not to provide an exhaustive comparison across the broader literature on optimization under uncertainty, which also includes approaches such as robust optimization (Bertsimas and Sim, 2004) and end-to-end decision-focused learning

(Wilder et al., 2019). For this reason, the results should be interpreted primarily as evidence of the potential of the proposed architecture within a controlled proof-of-concept setting, rather than as an exhaustive comparison among all possible methodologies for optimization under uncertainty.

Finally, scalability remains a relevant aspect when considering more realistic applications. As the number of tasks and planning periods increases, the size of the optimization model also grows, which may increase computational cost and gradually reduce the practicality of exact solution methods. In addition, the rolling-horizon architecture repeatedly retrains the forecasting model and solves the optimization problem at each step, which may require more efficient update strategies in larger-scale applications, such as retraining or re-solving the optimization problem only after a fixed number of steps. Approximate approaches and metaheuristics, such as genetic algorithms or simulated annealing, are also alternatives depending on the desired trade-off between computational effort and solution quality. Moreover, real decision environments may involve several objectives, additional operational constraints beyond the team-capacity constraint considered here, and more complex dependencies between activities, which were intentionally excluded from the current formulation to preserve clarity in the analysis. Future investigations may advance in several directions, including the use of real-world data, the incorporation of richer uncertainty-aware optimization schemes, the analysis of multi-objective formulations, and the exploration of alternative hybrid architectures for practical deployment.

7 Conclusion

This paper explored the challenges faced when implementing an intelligent automated system and incorporating it into the daily processes of a company. A three-stage plan was proposed to illustrate the resources required to develop the system sustainably. Then, a practical problem was presented to illustrate how data-driven methods can support system efficiency. To this end, a forecast-based optimization architecture was proposed to improve decision-making for task allocation in companies operating under uncertain constraints. The method is a combination of forecasting and optimization algorithms evaluated as a proof of concept using a stochastic synthetic dataset and compared against a traditional approach. Therefore, we provided an end-to-end analysis of how to implement an intelligent system and illustrated its potential to improve operational efficiency through a practical example. However, further experiments with more realistic datasets are still necessary to assess how the proposed methodology behaves in practical deployment scenarios.

Acknowledgments

This article is the result of the R&D project PANGEIA, with contributions from members of the STAR project, carried out by Sidia Institute of Science and Technology in partnership with Samsung Eletrônica da Amazônia Ltda., using resources provided under Federal Law No. 8,387/1991, with its disclosure and publication in accordance with Article 39 of Decree No. 10,521/2020.

References

- Abildtrup, A. et al. (2024). The rise of robotic process automation in the banking sector: streamlining operations and improving efficiency. *Journal of Computing and Natural Science*, 4(1):31–40.
- Achterberg, T. (2009). Scip: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41.
- Alghamdi, T., Elgazzar, K., Bayoumi, M., Sharaf, T., and Shah, S. (2019). Forecasting traffic congestion using arima modeling. In *2019 15th international wireless communications & mobile computing conference (IWCMC)*, pages 1227–1232. IEEE.
- Assimakopoulos, V. (2000). The theta model: a decomposition approach to forecasting. *International journal of forecasting*, 16(4):521–530.
- Bai, S., Kolter, J. Z., and Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.
- Bates, J. M. and Granger, C. W. (1969). The combination of forecasts. *Journal of the operational research society*, 20(4):451–468.
- Bertsimas, D., Brown, D. B., and Caramanis, C. (2011). Theory and applications of robust optimization. *SIAM review*, 53(3):464–501.
- Bertsimas, D. and Kallus, N. (2020). From predictive to prescriptive analytics. *Management Science*, 66(3):1025–1044.
- Bertsimas, D. and Sim, M. (2004). The price of robustness. *Operations research*, 52(1):35–53.
- Box, G. and Jenkins, G. (1976). Analysis: Forecasting and control. *San francisco*.
- Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). Classification and regression trees. *wadsworth int. Group*, 37(15):237–251.
- Brown, R. G. (1956). *Exponential smoothing for predicting demand*. Little.
- Chakraborti, T., Isahagian, V., Khalaf, R., Khazaeni, Y., Muthusamy, V., Rizk, Y., and Unuvar, M. (2020). From robotic process automation to intelligent process automation: –emerging trends–. In *International Conference on Business Process Management*, pages 215–228. Springer.
- Chappell, D. (2017). Introducing blue prism. *Robotic Process Automation for the Enterprise*. San Francisco, CA: Chappell & Associates.
- Chenreddy, A. and Delage, E. (2024). End-to-end conditional robust optimization. *arXiv preprint arXiv:2403.04670*.
- Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.
- Donti, P., Amos, B., and Kolter, J. Z. (2017). Task-based end-to-end model learning in stochastic optimization. *Advances in neural information processing systems*, 30.
- dos Santos Júnior, P. S., Ruy, F. B., Omêna, M. S., and Barcellos, M. P. (2025). A preliminary panoramic view of continuous software engineering adoption in brazilian organizations. *Journal of Software Engineering Research and Development*, 13(1):13–89.
- Eberly, L. E. (2007). Multiple linear regression. *Topics in biostatistics*, pages 165–187.
- Elhajjar, S., Yacoub, L., and Yaacoub, H. (2023). Automation in business research: systematic literature review. *Information Systems and e-Business Management*, 21(3):675–698.
- Elmachtoub, A. N. and Grigas, P. (2022). Smart “predict, then optimize”. *Management Science*, 68(1):9–26.
- Fersht, P. and Slaby, J. R. (2012). Robotic automation emerges as a threat to traditional low-cost outsourcing. *Horses for Sources, London*, 1:18.
- Fix, E. and Hodges, J. L. (1951). Discriminatory analysis, nonparametric discrimination.
- Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.
- Gardner Jr, E. S. and McKenzie, E. (1985). Forecasting trends in time series. *Management science*, 31(10):1237–1246.
- Ghojogh, B. and Ghodsi, A. (2023). Recurrent neural networks and long short-term memory networks: Tutorial and survey. *arXiv preprint arXiv:2304.11461*.
- Hendrawan, S. A., Chatra, A., Iman, N., Hidayatullah, S., and Suprayitno, D. (2024). Digital transformation in msme: Challenges and opportunities in technology management. *Jurnal Informasi dan Teknologi*, 6(2):141–149.
- Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Hsiung, H.-H. and Wang, J.-L. (2022). Research on the introduction of a robotic process automation (rpa) system in small accounting firms in taiwan. *Economies*, 10(8):200.
- Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts.
- Jones, D. A. and Cox, D. R. (1978). Nonlinear autoregressive processes. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 360(1700):71–95.
- Jovanović, S. Z., Đurić, J. S., and Šibalija, T. V. (2018). Robotic process automation: overview and opportunities. *International Journal Advanced Quality*, 46(3-4):34–39.
- Kitsantas, T., Georgoulas, P., and Chytis, E. (2024). Integrating robotic process automation with artificial intelligence for business process automation: Analysis, applications, and limitations. *Journal of System and Management Sciences*, 14(7):217–242.

- LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.
- Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2020). The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74. M4 Competition.
- Mandi, J., Kotary, J., Berden, S., Mulamba, M., Bucarey, V., Guns, T., and Fioretto, F. (2024). Decision-focused learning: Foundations, state of the art, benchmark and future opportunities. *Journal of Artificial Intelligence Research*, 80:1623–1701.
- McKinsey (2023). The imperatives for automation success.
- Mendling, J., Decker, G., Hull, R., Reijers, H. A., and Weber, I. (2018). How do machine learning, robotic process automation, and blockchains affect the human factor in business process management? *Communications of the Association for Information Systems*, 43(1):19.
- Montgomery, D. C., Jennings, C. L., and Kulahci, M. (2015). *Introduction to time series analysis and forecasting*. John Wiley & Sons.
- Montgomery, D. C. and Runger, G. C. (2010). *Applied statistics and probability for engineers*. John Wiley & sons.
- Mourtzis, D., Angelopoulos, J., and Panopoulos, N. (2023). The future of the human-machine interface (hmi) in society 5.0. *Future Internet*, 15(5):162.
- Nama, P. (2022). Cost management and optimization in automation infrastructure. *Iconic Research and Engineering Journals*, 5(12):276–285.
- Nielsen, I. E., Piyatilake, A., Thibbotuwawa, A., De Silva, M. M., Bocewicz, G., and Banaszak, Z. A. (2023). Benefits realization of robotic process automation (rpa) initiatives in supply chains. *IEEE Access*, 11:37623–37636.
- OM, A. (2001). Ridge regression and inverse problems. *Stockholm University, Department of Mathematics*.
- Qin, P., Ye, B., Li, Y., Cai, Z., Gao, Z., Qi, H., and Ding, Y. (2025). Hybrid bilstm-arima architecture with whale-driven optimization for financial time series forecasting. *Algorithms*, 18(8).
- Ribeiro, J., Lima, R., Eckhardt, T., and Paiva, S. (2021). Robotic process automation and artificial intelligence in industry 4.0—a literature review. *Procedia Computer Science*, 181:51–58.
- Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.
- Sadana, U., Chenreddy, A., Delage, E., Forel, A., Frejinger, E., and Vidal, T. (2025). A survey of contextual optimization methods for decision-making under uncertainty. *European Journal of Operational Research*, 320(2):271–289.
- Sahu, S. K., Mokhade, A., and Bokde, N. D. (2023). An overview of machine learning, deep learning, and reinforcement learning-based techniques in quantitative finance: recent progress and challenges. *Applied Sciences*.
- Sain, S. R. (1996). The nature of statistical learning theory.
- Salimzadeh, S., He, G., and Gadiraju, U. (2024). Dealing with uncertainty: Understanding the impact of prognostic versus diagnostic tasks on trust and reliance in human-ai decision making. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pages 1–17.
- Santos, G., Dutra, E., Mattos, C. S., Fortuna, A., dos Santos, R. P., Ramos, L. F., and da Cruz Andrade, Á. J. (2025). What leads to the success of agile transformation initiatives? a study on critical success factors. *Journal of Software Engineering Research and Development*, 13(1):13–1.
- SAP (2023). What is process automation?
- Siderska, J., Aunimo, L., Süße, T., von Stamm, J., Kedziora, D., and Aini, S. N. B. M. (2023). Towards intelligent automation (ia): literature review on the evolution of robotic process automation (rpa), its challenges, and future trends. *Engineering Management in Production and Services*, 15(4).
- Silva, B., Hak, F., Guimaraes, T., Manuel, M., and Santos, M. F. (2023). Rule-based system for effective clinical decision support. *Procedia Computer Science*, 220:880–885.
- Staudemeyer, R. C. and Morris, E. R. (2019). Understanding lstm—a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*.
- Stoykova, S. and Shakev, N. (2023). Artificial intelligence for management information systems: Opportunities, challenges, and future directions. *Algorithms*, 16(8):357.
- Sutherland, J. (2014). *Scrum: the art of doing twice the work in half the time*. Crown Currency.
- Taylor, S. J. and Letham, B. (2018). Forecasting at scale. *The American Statistician*, 72(1):37–45.
- Thiruvady, D., Morgan, K., Bedingfield, S., and Nazari, A. (2021). Allocating students to industry placements using integer programming and ant colony optimisation. *Algorithms*, 14(8):219.
- Tripathi, A. M. (2018). *Learning Robotic Process Automation: Create Software robots and automate business processes with the leading RPA tool—UiPath*. Packt Publishing Ltd.
- Van der Aalst, W. M., Bichler, M., and Heinzl, A. (2018). Robotic process automation. *Business & information systems engineering*, 60(4):269–272.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
- Wang, J. (2023). An intuitive tutorial to gaussian process regression. *Computing in Science & Engineering*.
- Weber, M., Auch, M., Doblander, C., Mandl, P., and Jacobsen, H.-A. (2021). Transfer learning with time series data: A systematic mapping study. *Ieee Access*, 9:165409–165432.
- Wilder, B., Ewing, E., Dilkina, B., and Tambe, M. (2019). End to end learning and optimization on graphs. *Advances in Neural Information Processing Systems*, 32.
- Williams, C. K. and Rasmussen, C. E. (2006). *Gaussian pro-*

- cesses for machine learning, volume 2. MIT press Cambridge, MA.
- Winters, P. R. (1960). Forecasting sales by exponentially weighted moving averages. *Management science*.
- Xu, Z. and Saleh, J. H. (2021). Machine learning for reliability engineering and safety applications: Review of current status and future opportunities. *Reliability Engineering & System Safety*, 211:107530.
- Yang, X.-S. (2008). Introduction to mathematical optimization. *From linear programming to metaheuristics*.
- Yeh, C., Christianson, N., Wu, A., Wierman, A., and Yue, Y. (2024). End-to-end conformal calibration for optimization under uncertainty. *arXiv preprint arXiv:2409.20534*.
- Zayas-Cabán, T., Haque, S. N., and Kemper, N. (2021). Identifying opportunities for workflow automation in health care: lessons learned from other industries. *Applied Clinical Informatics*, 12(03):686–697.

A Appendix

This appendix provides a simple step-by-step code of the functions used to generate both the Project Backlog information and the Constraint History utilized in the experiments. It also details the execution process of the hybrid architecture from Figure 3.

Listing 1. Function for generating the project backlog items used in the simulation

```

FUNCTION generate_synthetic_items:
  Input: n_items, mu, std, w, a, b
  Output: items
  Step 1: Generate an array of gaussian distributions using the mean (mu) and standard deviation (std)
  Step 2: Apply the weights (w) to scale the gaussian distributions
  Step 3: Sum the distributions into a single probability function
  Step 4: Truncate the values outside the low (a) and high (b) limits to zero
  Step 5: Normalize the truncated distribution
  Step 6: Compute the cumulative density function (CDF) from the distribution
  for item in range(n_items):
    Step 7: Generate a value using a uniform distribution
    Step 8: Use the generated value to sample from the CDF
  return items

```

Listing 2. Pseudocode for generating synthetic time series using a trend, seasonality, noise model

```

FUNCTION generate_synthetic_time_series:
  Input: n_historic, b_0, b_1, A_h, A_l, std_n
  Output: time_series
  Step 1: Create the timestamps from n_historic
  Step 2: Generate a trend using the trend slope (b_1) and offset (b_0)
  Step 3: Generate the seasonality component (A_h, A_l)
  Step 4: Generate the noise component using a Gaussian distribution (std_n)
  Step 5: Sum the trend, seasonality and noise components
  return time_series

```

Listing 3. Pseudocode implementation of the proposed hybrid architecture to generate the optimal solution based on the item backlog and historical data

```

FUNCTION simulate_forecasting_optimization:
  Input: time_series, items, historic, forecasting_method
  Output: schedule, total_lateness, forecast_values
  FOR each day IN time_series:
    Step 1: Update known capacities and historic data
    Step 2: Train forecasting model and predict future constraints
    Step 3: Remove already allocated items from backlog
    Step 4: Schedule items on the current day
  Step 5: Calculate the total lateness of all items
  RETURN schedule, total_lateness, forecasts

```
