Simulation-based Assessment of Business Process Robustness

Robert Blümel[1,2], Alexander Kraus[3], Timotheus Kampik[2,4], and Han van der Aa[5]

[1] Doctoral School Computer Science, University of Vienna, Vienna, Austria
[2] SAP Signavio, Walldorf, Germany
{robert.bluemel,timotheus.kampik}@sap.com
[3] Data and Web Science Group, University of Mannheim, Mannheim, Germany
alexander.kraus@uni-mannheim.de
[4] Department of Computing Science, Umeå University, Umeå, Sweden
[5] Faculty of Computer Science, University of Vienna, Vienna, Austria
han.van.der.aa@univie.ac.at

Abstract. Business process robustness describes a process's ability to meet performance targets despite variations in parameters such as arrival rates or resource availability. Assessing business process robustness quantitatively is challenging because dependencies between process parameters and performance are complex and stochastic. Business process simulation avoids the need to specify those dependencies by simulating the process and observing the resulting behavior. However, the space of possible parameter combinations grows quickly, which makes it challenging to identify parameters that still meet the required performance targets. In this paper, we address this challenge by proposing a simulation-based approach that systematically explores parameter configurations to efficiently identify those that maintain acceptable performance. Our evaluation demonstrates that the proposed approach substantially reduces assessment complexity compared to our baselines while accurately identifying parameter combinations under which a process remains robust. The approach provides users with insights into how parameter variations influence performance, enabling a step toward a quantitative foundation for assessing and improving business process robustness.

Keywords: Business process robustness · Robustness assessment · Business process simulation · Process mining

1 Introduction

Organizations operate in environments characterized by constant change. These changes influence business processes and their parameters, such as arrival rates or resource availability [8]. Consequently, process performance measures, such as the average cycle time or resource cost, may fluctuate over time. When processes are expected to achieve specific performance targets, variations in process

parameters can cause these targets to be missed. Hence, it is essential to understand the extent to which process parameters can vary while maintaining desired performance levels. This relates to the concept of robustness [3,9,27], which captures the ability of a business process to sustain its performance targets despite changes in underlying parameters.

Despite the importance of business process robustness, its quantitative assessment remains a challenging and largely unexplored task in process mining. This challenge arises from the need to understand how specific parameter configurations influence process performance, a relationship that is often difficult to determine due to its inherent complexity. For example, while higher arrival rates generally increase cycle times, the precise impact is not straightforward to predict. The stochastic nature of processes [24] further adds variability to performance outcomes. Consequently, developing robustness assessment approaches is difficult, since they need to account for complex and stochastic dependencies between process parameters and performance outcomes.

One way of dealing with process complexity and stochasticity is to use business process simulation (BPS). Recent advances in BPS have enabled quantitative assessment of business process robustness. By modeling processes and simulating them under varying parameter configurations, BPS makes it possible to examine how parameter changes affect overall process performance [1]. However, existing BPS tools are mainly designed for simple what-if analyses and typically used for identifying parameters that optimize selected performance measures [19, 20, 22, 23], rather than for determining the complete set of configurations that maintain acceptable performance. As a result, current approaches cannot efficiently identify all parameter settings under which a process meets its performance targets. This gap highlights the need for a dedicated solution for robustness assessment.

To address this gap, we propose a simulation-based approach for assessing business process robustness. Our approach quantifies how much variation a process can withstand while maintaining acceptable levels of process performance. Our contribution is twofold. First, we formalize business process robustness as an algorithmic assessment problem. Second, we develop an approach that systematically explores parameter variations to efficiently identify configurations that ensure acceptable performance using information recorded about the process in an event log. Conducted evaluation experiments demonstrate that our proposed approach substantially reduces the assessment complexity compared to other search baselines, while accurately identifying the parameter configurations under which the process remains robust. As a result, our approach gives users clear insights into the parameter configurations under which a process meets its performance targets. This marks a first step toward a quantitative foundation for assessing and improving business process robustness.

The remainder of the paper is organized as follows. In Section 2, we discuss the problem of business process robustness assessment. Then, we describe our simulation-based approach for its assessment in Section 3 and evaluate it in Section 4. In Section 5 we discuss related work, before we conclude in Section 6.

2 Problem Statement

This section formalizes the problem of assessing business process robustness by defining its core elements and the related assessment challenges.

Event logs. An event log L contains data recorded by a process-aware information system during process execution [2]. Each event $e \in L$ includes at least a case ID, an activity, and a timestamp. Events with the same case ID ordered by time form a *trace*. An event log may also contain further attributes relevant for robustness analysis, such as resource information, costs, and lifecycle data.

Process parameters. Process parameters are factors that influence the behavior and performance of a business process. They are represented as a vector $X = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, where each x_i denotes a specific parameter, such as the number of available resources or the average processing time of an activity.

Each process parameter x_i takes values from a defined range $\mathbb{I}^*_{x_i} = [a_i, b_i]$, for example, the number of available resources between 1 and 20. The *parameter domain* D is the Cartesian product of these ranges, so that $X \in D \subset \mathbb{R}^n$. D thus represents all parameter configurations considered for robustness assessment.

Process performance indicators. Process performance indicators (PPIs) are quantifiable measures of process effectiveness or efficiency [25]. They are represented as a vector $P = (p_1, p_2, \ldots, p_m) \in \mathbb{R}^m$, where each p_i denotes a performance metric such as time, cost, quality, or resource usage. Common PPIs include average lead time, process execution cost, and resource utilization.

Each PPI p_i has an acceptable performance range $\mathbb{I}^*_{p_i} = [l_i, u_i]$, where l_i and u_i are the lower and upper bounds. If only one bound is specified, the other is set to $+\infty$, $-\infty$, or zero, depending on the context. The *acceptable performance space* is defined as $P^* = \mathbb{I}^*_{p_1} \times \mathbb{I}^*_{p_2} \times \cdots \times \mathbb{I}^*_{p_m}$.

Performance function. We represent the relationship between process parameters and outcomes through a process's *performance function*. This function maps the parameter domain to PPIs and can be viewed as a sample function of a stochastic process [15], producing one realization of performance for a given parameter vector. Formally, it is defined as $f(X) = P$, where $X = (x_1, \ldots, x_n)$ is the parameter vector and $P = (p_1, \ldots, p_m)$ the resulting vector of PPIs.

Monotonicity. We assume that each performance function is strictly monotonic with respect to each parameter. Formally, let $f_j(X)$ denote the j-th PPI for configuration X. For any parameter x_i, and for any two configurations X and X' that differ only in the i-th coordinate, strict monotonicity requires $(x_i - x'_i)\big(f_j(X) - f_j(X')\big) > 0$ whenever $x_i \neq x'_i$. That is, changing one parameter while keeping all other parameters fixed changes the corresponding PPI in a consistent direction. Note that monotonicity does not require a linear or otherwise specific functional relationship between parameters and PPIs, only that each parameter affects a given PPI in a consistent direction. This assumption is reasonable in our context, since parameter effects often follow predictable monotonic trends. For example, adding additional resources generally reduces the average lead time, while more case arrivals increase it (with everything else remaining equal).

Robustness assessment challenges. Given a parameter domain D, an acceptable performance space P^*, and a process performance function f, the assessment of business process robustness is characterized by two main challenges:

Process stochasticity. Business processes are complex socio-technical systems [4] with many interacting elements, including human actors. Consequently, business processes are stochastic, and the same parameters may yield different PPI outcomes. For example, task durations depend on contextual factors such as worker experience. The performance function f is therefore stochastic and unknown, and must be approximated from an event log L.

State-space explosion. Even if an unbiased estimation of the performance function f is available, identifying the robustness space remains computationally challenging due to state-space explosion when the process parameter domain $D \subseteq \mathbb{R}^n$ is multidimensional. This makes an efficient search strategy necessary to determine the robustness space within reasonable computational limits.

Robustness space. Given these challenges, the objective of this work is to approximate business process robustness efficiently and with high accuracy. Therefore, we define a robustness space $S^{\alpha,\beta}$ as the set of parameter configurations that lead to acceptable process performance:

$$S^{\alpha,\beta} = \{\, X \in D \mid f(X) = P \in P^* \,\},$$

where:

- $\alpha \in (0, 1)$ defines the *confidence level*, i.e., the likelihood that parameter configurations within the robustness space yield acceptable performance. Formally, $\mathbb{P}(f(X) \in P^* \mid X \in S^{\alpha,\beta}) \geq \alpha$. This condition accounts for the stochastic nature of a process by ensuring that configurations in $S^{\alpha,\beta}$ meet the performance requirements with a confidence level of at least α, e.g., 0.90.
- $\beta \in (0, 1)$ defines the *accepted uncertainty fraction* of the parameter domain that may contain configurations that also yield acceptable performance but lie outside the estimated robustness space, i.e., $\mathbb{P}(X \in D \setminus S^{\alpha,\beta} \mid f(X) \in P^*) \leq \beta$. This condition addresses state-space explosion by allowing a part of the parameter domain, e.g., up to 10%, to remain uncertain, avoiding costly searches by trading off search efficiency and accuracy.

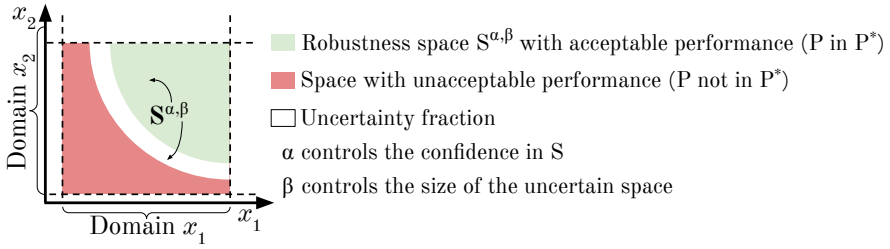


Fig. 1: Robustness space $S^{\alpha,\beta}$ (light green) with confidence level α and accepted uncertainty fraction β (white) within the parameter domain D.

Figure 1 illustrates such a space $S^{\alpha,\beta}$. It contains all parameters $X \in D$ for which the resulting performance $f(X) = P$ lies within P^*. The confidence level α specifies the confidence of points in the robustness space (light green), while β limits the size of the accepted uncertainty space in the parameter domain (white).

Finally, we formalize the problem of business process robustness assessment:
Problem statement. Given an event log L, a parameter domain D, an acceptable performance range P^*, a desired confidence level α, and an accepted uncertainty fraction β, the task of business process robustness assessment is to determine the approximated robustness space $S^{\alpha,\beta}$ such that every parameter configuration $X \in S^{\alpha,\beta} \subseteq D$ satisfies $f(X) = P \in P^*$, with $\mathbb{P}(f(X) \in P^* \mid X \in S^{\alpha,\beta}) \geq \alpha$ and $\mathbb{P}(X \in D \setminus S^{\alpha,\beta} \mid f(X) \in P^*) \leq \beta$.

3 Business Process Robustness Assessment Approach

This section presents our approach for approximating the robustness space of a business process. As shown in Figure 2, the approach takes as input an event log and a configuration that specifies the parameters to vary and the acceptable PPI ranges. It then iteratively simulates selected parameters and refines the search space through an adaptive search strategy until a termination condition is met. The result is an approximation of the robustness space, that is, the set of parameter configurations under which the process meets its performance targets.

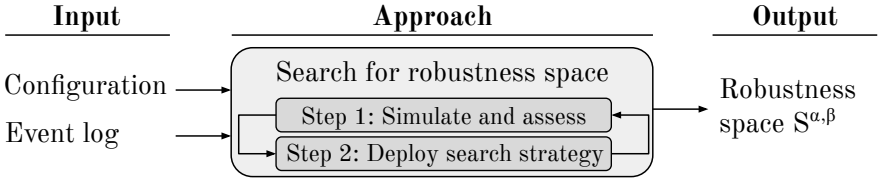


Fig. 2: Overview of the business process robustness assessment approach.

3.1 Step 1: Simulate and Assess

In the first step, we simulate the process for a selected parameter configuration and assess its resulting performance. The result indicates whether the process meets the robustness requirements under this configuration. This information guides the search in Step 2 by showing which spaces of the parameter domain require further analysis. In the following, we describe the discovery of the BPS model, performed once, and the process simulation and performance assessment procedure, applied to each evaluated parameter configuration.

BPS model discovery. To simulate a given process parameter configuration, we first discover a BPS model. This model can be discovered automatically from an event log L by using data-driven BPS techniques, such as Simod [6],

AgentSimulator [12], or DeepSimulator [5]. Alternatively, a BPS model can be constructed manually and provided to the approach.

Process simulation and performance assessment. Once the BPS model is available, we simulate the process under a parameter configuration X and assess the resulting performance. This combination of simulation and performance assessment yields an estimate of the performance function. To capture the stochastic nature of the process, each configuration is simulated N times. The resulting performance values are aggregated into an empirical distribution. We compute the fraction of outcomes that fall within the acceptable performance range for each PPI. A configuration X is classified as part of the robustness space $S^{\alpha,\beta}$ when this fraction reaches at least α for all PPIs; otherwise, X is outside of the robustness space.

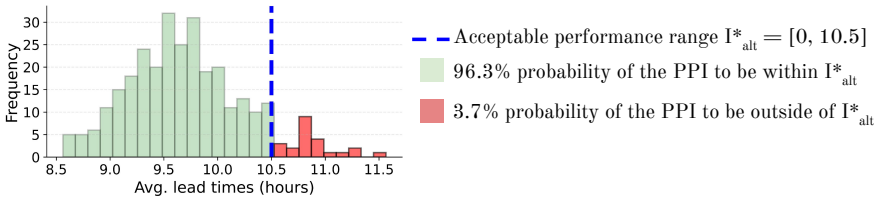


Fig. 3: Performance distribution after 300 process executions.

Example. Figure 3 illustrates the idea behind the performance assessment step, using the average lead time (*alt*) as the PPI of interest. To decide whether a parameter configuration X leads to acceptable process performance, we simulate the process $N = 300$ times and obtain the corresponding performance values. This results in the frequency plot shown in the figure and the associated empirical distribution of the target PPI. Assuming that the acceptable performance range is $\mathbb{I}^*_{alt} = [0, 10.5]$ (the upper bound is indicated by the dashed blue line), we observe that 96.3% of the simulated outcomes fall within this range. Therefore, for any confidence level α below 96.3%, configuration X leads to acceptable performance and is classified as part of the robustness space.

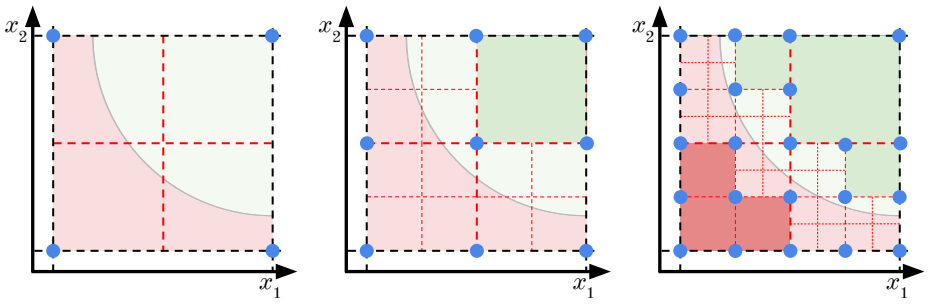
3.2 Step 2: Deploy Search Strategy

In this step, we deploy our search strategy to approximate the robustness space. We use a quadtree-based [26] search that adaptively explores the parameter domain and avoids exhaustive evaluation of all configurations. Since we apply this to a d-dimensional parameter domain, we will refer to this as a hyperquadtree (HQT) [16]. In the following, we provide an overview of the deployed search strategy and then detail its key steps.

Overview of the search strategy. The goal of the search strategy is to approximate the set of parameter configurations that yield acceptable process performance while keeping the number of performance evaluations low. Although

the exact performance function of a business process is unknown and stochastic, we assume that it is strictly monotonic (see Section 2). This assumption allows us to infer the behavior of a parameter subdomain from a limited number of simulated and assessed configurations using the hyperquadtree-based search strategy. This strategy represents the parameter domain as a hierarchical tree of nested hyperrectangular subspaces, referred to as nodes. Each node is defined by its hyperrectangular corners corresponding to specific parameter configurations in the domain. In an d-dimensional parameter domain, each node contains 2^d corner configurations. By evaluating the performance at these corner points, we can determine whether all configurations in the node are likely to meet or violate the performance requirements without sampling its interior. Nodes whose corner points all yield acceptable or unacceptable performance are classified accordingly. Nodes with a mix of acceptable and unacceptable outcomes remain uncertain and are recursively subdivided. This focuses simulations on the most informative parts of the domain, while large spaces can be classified with few evaluations.

Key steps of the search strategy. Our search consists of the following key steps: initialization, node classification, node subdivision, and termination. Figure 4 illustrates the first three iterations of these steps in a 2-dimensional domain.



(a) Iteration 1. (b) Iteration 2. (c) Iteration 3.

Fig. 4: Illustration of the hyperquadtree-based search strategy in 2 dimensions.

Initialization. We begin the search by considering the entire parameter domain as a *root node* with an unknown robustness status, as shown in Figure 4a. This root node serves as the basis for subsequent recursive classification and subdivision.

Node classification. We classify a node and, with that, the subspace it represents by assessing whether each of its corners meets the performance targets, using Step 1. Based on the obtained results, a node is classified as:

- *robust*, i.e., part of $S^{\alpha,\beta}$, if all corners meet the performance targets.
- *non-robust*, i.e., outside of $S^{\alpha,\beta}$, if no corners meet the performance targets.
- *uncertain*, if some corners meet the performance targets while others do not.

After classification, the parameter configurations of the robust nodes are added to $S^{\alpha,\beta}$. The non-robust nodes are excluded from further consideration. The

uncertain nodes are marked for recursive refinement in subsequent iterations. As a result, this strategy focuses computational effort on nodes where the boundary between robust and non-robust configurations is unclear.

For Iteration 1, shown in Figure 4a, we evaluate the root node using its four corner points, marked by the blue dots. The light green area represents the ground-truth robustness space, while the red area contains configurations that do not meet the performance requirements. Based on this ground truth, the root node is classified as uncertain (indicated by the white semi-transparent overlay) because the corner evaluations yield a mix of accepted and non-accepted performance outcomes. As a result, the root node is subdivided into four child nodes for the next iteration, shown in Figure 4.

Subdivision of uncertain nodes. In the next step, we subdivide all uncertain nodes. Specifically, each uncertain node is split at the midpoint of every parameter dimension, which produces 2^d child nodes that are considered for classification in the next iteration. Through this recursive refinement, the hierarchical structure adapts dynamically: nodes classified as robust or non-robust are not explored further, while uncertain nodes are examined in increasing detail. Figure 4 illustrates this refinement process across the first three iterations.

Search termination. The recursive search stops when two conditions are satisfied. First, a minimum refinement depth must be reached to avoid stopping at an overly coarse partition. Second, the fraction of the parameter domain covered by the uncertain nodes compared to the entire parameter domain must drop below the accepted tolerance threshold β. The recursive search is performed until both termination conditions hold. Then, no further simulations are performed, and the search returns the final approximation of the robustness space $S^{\alpha,\beta}$.

In our example, none of the termination conditions are met after Iteration 1 (Figure 4a), so the algorithm proceeds with Iteration 2 (Figure 4b). In this iteration, the node in the top right area of the domain shows acceptable performance for all of its corner points. This node is therefore classified as robust, and its parameter configurations become part of the robustness space $S^{\alpha,\beta}$. The remaining nodes show a mix of accepted and non-accepted performance outcomes at their corners and are thus classified as uncertain. These nodes are subdivided and evaluated in Iteration 3 (Figure 4c). In this third iteration, the search classifies three nodes as non-robust, three nodes as robust, and seven nodes as uncertain. In the next iteration, the search will explore these seven uncertain nodes, further reducing the uncertain space of the parameter domain and providing a more accurate approximation of the robustness space.

4 Evaluation

This section presents experiments conducted to assess the accuracy and efficiency of our approach in comparison to four baselines. We first describe the evaluation setup and then discuss the results. To support reproducibility, we provide our data, implementation, and results in our repository[6].

[6] `https://github.com/robert-l-b/robustness`

4.1 Setup

In this section, we discuss the relevant setup of our experiments.

Simulated process. We evaluated our approach on an established loan application process (cf. [8, Chapter 10.8]) that is represented by the BPMN model shown in Figure 5. The model comprises 12 activities, a loop, a parallel structure, and several exclusive gateways, resulting in broad behavioral variation. The configurations for the simulation model, such as the distributions underlying task durations and branching probabilities, are discovered from an event log of the process (cf. [7]) using Simod [6].

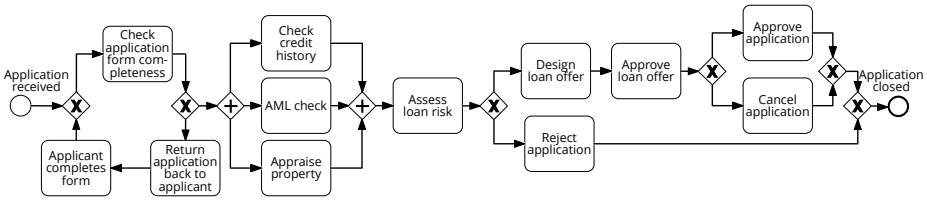


Fig. 5: BPMN model of the loan application process.

Input configurations. We consider the following input configurations for the robustness assessment of the loan application process:

Process parameters. We assess robustness with respect to parameters related to different process perspectives:

- *Arrival times (at)—(case perspective):* We model the arrival time by adjusting the mean of the discovered case inter-arrival time distribution. In our setup, this distribution was found to be a constant inter-arrival time that we vary from 0 to 125 minutes, that is, $\mathbb{I}^*_{at} = [0, 125]$.
- *Resources* (res)*—(resource perspective):* We consider a single resource type, where all resources follow the same seven-to-four schedule and are cross-trained, meaning that they can execute any activity when they are available. We vary the number of resources from 1 to 20, i.e., $\mathbb{I}^*_{res} = [1, 20]$.
- *Branching probability (bp)—(control-flow perspective):* We vary the branching probability at the XOR gateway following *Assess Loan Risk* from 0 to 1, that is, $\mathbb{I}^*_{bp} = [0, 1]$. This setting allows us to examine how varying strictness in risk assessment affects the robustness of the process.

Process performance indicators. We assess the robustness of the process with respect to two PPIs that cover the two remaining process perspectives:

- *Average lead time (alt)—(time perspective):* The average lead time is defined as the mean time between the first and the last event across all simulated cases. The accepted performance range $\mathbb{I}^*_{alt}$ is the interval $[0, 11]$ hours.
- *Cost (cost)—(data perspective):* The total cost is calculated as the working hours of all resources multiplied by a cost factor of 20 per hour. The corresponding accepted performance range $\mathbb{I}^*_{cost}$ is given by the interval $[0, 60{,}000]$.

Evaluation scenarios. We define four evaluation scenarios (S1–S4) that cover increasing search space sizes, as well as single- and multi-objective performance constraints. Scenarios 1–3 use the same two-dimensional parameter domain, combining arrival times and number of resources, but differ in the PPIs: S1 evaluates average lead time, S2 evaluates cost, and S3 evaluates both. Scenario S4 increases the dimensionality by adding the branching probability as a third parameter and assesses both average lead time and cost.

Approach parameters. Across all scenarios, each configuration is simulated $N = 30$ times. Each run produces a log with 300 traces, a size determined using Welch's method [17] to remove warm-up effects and ensure stable estimates. We set the confidence level to $\alpha = 0.90$. For Scenarios 1–2, the accepted uncertainty fraction is $\beta = 0.15$, whereas for Scenarios 3–4, we set $\beta = 0.25$ to account for the increased complexity due to multi-objective constraints in both scenarios and the higher-dimensional parameter space. The minimum refinement depth is set to 3, so the search performs at least three rounds of subdivision before termination is checked. This limits the influence of any single corner evaluation on the resulting classification. Additionally, we conduct a sensitivity analysis, in which we vary $\alpha \in \{0.75, 0.80, 0.85, 0.90, 0.95, 0.99\}$ and $\beta \in \{0.10, 0.15, 0.20, 0.25\}$ to study their effect on the robustness assessment.

Baselines. We compare our approach against four baselines that represent alternative strategies for approximating the robustness space.

Grid search (GS). GS is a simple baseline that evaluates all parameter combinations on a predefined grid, providing a clear reference without relying on model assumptions or sampling strategies. The grid covers all parameter configurations defined by the domain step sizes: 50 options for arrival times, 20 for resources, and 11 for branching probability. Scenarios 1–3 therefore require 1,000 evaluations, and Scenario S4 requires 11,000. Since the true robustness space is unknown, GS provides a reliable, though costly, reference for evaluating the accuracy and efficiency of all other techniques.

Fixed space partitioning (FSP). FSP partitions the parameter domain into a fixed set of equally sized subspaces and represents a non-adaptive variant of the hyperquadtree approach. Each subspace is classified by evaluating its corner configurations, following the node classification logic described in Section 3.2. The classification of the subspaces continues until all subspaces are labeled or the uncertain space falls below β. To ensure comparability, we iteratively decrease the grid size until the resulting uncertainty fraction meets the target β, and report the number of evaluations at that grid size.

K-nearest neighbors (KNN). We consider KNN as a simple classification-based baseline. Parameter configurations are randomly sampled and labeled as described in Section 3.1, and the model is trained to distinguish between robust and non-robust configurations. The hyperparameter k is selected via a five-fold cross-validation. KNN is then applied to classify all configurations in the grid search reference, where a configuration is classified only if the predicted probability exceeds the threshold α. We report the setting that requires the fewest training points while satisfying the uncertainty criterion.

Regression tree (RT). We evaluate regression trees as a more expressive classification baseline, using the same setup as for KNN. Hyperparameters are selected via grid search with five-fold cross-validation on the sampled training set, with each fold containing an equal number of randomly assigned configurations.

Evaluation measures. We consider four measures that capture the efficiency (*Evals* and *Time*) and accuracy ($\hat{\beta}$ and *MCC*) for each scenario:

Number of evaluated configurations (Evals). Measures how many parameter configurations the approach simulates.

Computation time in minutes (Time). Measures the total runtime of the robustness assessment.

Uncertainty fraction ($\hat{\beta}$). Measures the proportion of unclassified evaluations.

Matthews correlation coefficient (MCC). Measures how well our approach classifies the configurations of the grid search based on $S^{\alpha,\beta}$. The MCC considers all entries of the confusion matrix and is defined as [21]:

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}},$$

where TP and TN are correct robust and non-robust classifications, and FP and FN are the corresponding misclassifications. Unclassified configurations in the uncertain space (fraction β) are excluded. MCC ranges from -1 to $+1$, where $+1$ means perfect classification, 0 random guessing, and -1 full disagreement.

Note that for KNN and RT, we repeat each experiment five times with different random samples and report the average results to account for variability.

Implementation. The evaluation of our approach and the baselines was carried out in the same technical environment. We implemented all search strategies in Python and executed them on an Apple MacBook Pro (macOS version 15) equipped with a 16-core M2 Pro processor and 16 GB of RAM. To avoid effects related to parallel execution, all experiments were run on a single core.

4.2 Results

In this section, we discuss the evaluation results of our experiment. We first examine the efficiency and accuracy of our approach in comparison to the four baselines and then conduct a sensitivity analysis to assess how the parameters α and β influence the performance of our approach.

Efficiency and accuracy. Table 1 reports the evaluation results for all scenarios, comparing our approach, hyperquadtree (HQT), with the four baselines. In the following, we first discuss these results for the two-dimensional Scenarios S1–S3, and then look at the three-dimensional Scenario S4.

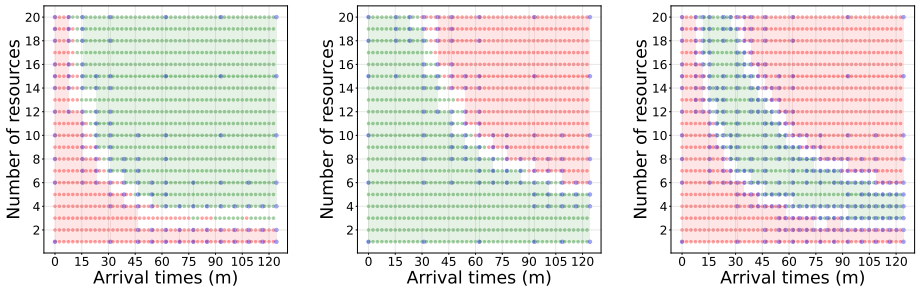
Scenarios S1–S3. Figure 6 presents the HQT results for the first three scenarios. All three scenarios share the same two-dimensional parameter domain but differ in the PPIs considered: average lead time (S1), cost (S2), and both measures (S3). The green and red dots show the configurations evaluated by GS, while the blue dots indicate the configurations evaluated by HQT. The red area marks configurations with unacceptable performance, and the green area represents the

Table 1: Experiment results of our approach for Scenarios 1–4.

Techniques	Scenario S1				Scenario S2				Scenario S3				Scenario S4			
	Evals	Time	$\hat{\beta}$	MCC	Evals	Time	$\hat{\beta}$	MCC	Evals	Time	$\hat{\beta}$	MCC	Evals	Time	$\hat{\beta}$	MCC
GS	1,000	88.0	–	–	1,000	79.8	–	–	1,000	92.3	–	–	11,000	975.5	–	–
FSP	220	18.2	0.14	**1.00**	180	13.4	**0.12**	**1.00**	300	31.1	0.20	**1.00**	6,730	561.7	0.24	**0.71**
KNN	150	24.9	0.14	0.99	100	16.6	0.15	0.99	200	37.7	0.20	0.98	4,000	738.8	0.24	0.68
RT	**75**	**12.7**	0.13	0.87	100	15.1	0.15	0.87	200	70.3	0.23	0.69	5,000	990.0	0.26	0.62
HQT	115	15.2	**0.08**	**1.00**	**54**	**8.5**	0.15	**1.00**	**176**	**25.9**	**0.14**	0.99	**2,874**	**243.6**	**0.14**	0.67

Note: The highlighted values indicate the best result in each column.

robustness space estimated by HQT. The figure shows that the robustness space identified by HQT closely matches that of GS and that the evaluation effort is efficiently concentrated along the boundary between the two areas.



(a) Scenario S1. (b) Scenario S2. (c) Scenario S3.

Fig. 6: HQT vs. GS comparison showing the robustness space and boundary-focused evaluation.

Quantitatively, as reported in Table 1, HQT achieves near-perfect classification with MCC values of 1.00 or 0.99 across all three scenarios relative to GS. This confirms that grid points in the green and red areas are correctly classified. For each scenario, HQT's unclassified fraction of the domain (shown in white) accounts for about 8%–15%. These results should be interpreted in light of the uncertainty inherent in GS at the boundary between robust and non-robust configurations, which amounts to 7.2%, 5.6%, and 12.6% for S1–S3, respectively. Moreover, GS exhibits locally mixed behavior. For example, in S1, along the row with three resources, green and red GS points alternate within the white space. This indicates that GS itself is an uncertain approximation of the ground truth.

In comparison to all baselines, HQT provides the most favorable efficiency–accuracy trade-off. In terms of evaluation effort, HQT requires only 5–12% of the GS evaluations in S1 and S2 (115 and 54 vs. 1,000), whereas the other baselines require substantially more, with RT needing around 75–100 evaluations, KNN

100–150, and FSP up to 220. In S3, HQT uses 18% of GS evaluations (176 vs. 1,000), whereas the other baselines require 200–300. In terms of runtime, S1 is the only scenario where HQT is slightly slower than RT, but in S2 and S3, it achieves speedups of a factor of 1.5–3 over all other baselines. In terms of accuracy, HQT maintains near-perfect MCC (1.00 in S1–S2, 0.99 in S3), whereas FSP achieves 1.00 but with substantially more evaluations, KNN reaches 0.98–0.99, and RT drops to 0.69–0.87. Furthermore, HQT consistently yields the smallest unclassified fraction $\hat{\beta}$ and reaches these results in a single adaptive run. In contrast, the classification-based baselines KNN and RT achieve competitive accuracy, but the reported results correspond to the smallest number of training points for which the resulting classification satisfies the uncertainty criterion. Finding this number required testing several values for the number of training points, and the simulation calls used in those preceding trials are not included in Table 1. The same holds for the partitioning depth required by FSP. Unlike the baselines, HQT does not require any prior choice of the number of training points or the partitioning depth. Instead, it adaptively concentrates evaluation effort on uncertain parts of the domain and terminates once this uncertainty falls below the accepted uncertainty fraction β.

Scenario S4. Figure 7 presents the results for S4, which considers a three-dimensional parameter domain and two PPIs. The red cubes denote parts of the domain with unacceptable performance, the white cubes represent the uncertain space, and the green cubes mark the detected robustness space.

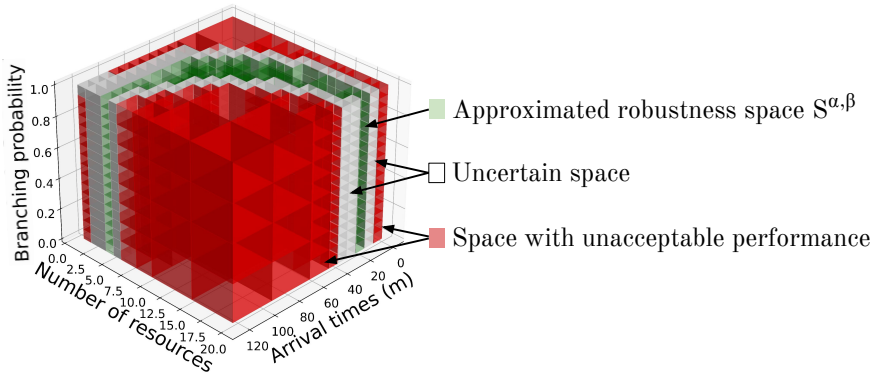


Fig. 7: Search results of HQT for Scenario S4.

Consistent with the previous scenarios, HQT maintains a favorable efficiency–accuracy balance in S4 despite the increased dimensionality of the parameter space. As reported in Table 1, HQT evaluates 2,874 configurations—about 26% of GS—and reduces runtime to 243 minutes, a factor of 3.8 faster. The MCC is 0.67, with 14% of the domain unclassified. The decrease in accuracy is primarily due to configurations near the uncertain boundary of the three-dimensional

space. Compared to that, FSP reaches a slightly higher MCC of 0.71 but requires 6,730 evaluations—more than twice the effort of HQT. KNN and RT require 4,000 and 5,000 evaluations, respectively, to achieve similar coverage, with RT still slightly failing to meet the target unclassified fraction $\hat{\beta} = 0.25$. Their MCC values (0.68 for KNN and 0.62 for RT) remain at or below that of HQT. Overall, these results confirm that HQT concentrates evaluations on uncertain regions, achieving higher efficiency while maintaining comparable accuracy.

Table 2: Sensitivity analysis results for varying α and β (Scenario S3).

	$\boldsymbol{\beta_{0.25}}$			$\boldsymbol{\beta_{0.20}}$			$\boldsymbol{\beta_{0.15}}$			$\boldsymbol{\beta_{0.10}}$		
	Evals	Time	MCC	Evals	Time	MCC	Evals	Time	MCC	Evals	Time	MCC
$\alpha_{0.75}$	66	17.7	1.00	176	29.5	0.99	176	29.2	0.99	438	44.7	1.00
$\alpha_{0.80}$	66	17.2	1.00	176	26.1	0.99	176	29.4	1.00	436	38.4	1.00
$\alpha_{0.85}$	66	17.8	1.00	176	30.5	0.99	176	29.9	1.00	436	45.7	1.00
$\alpha_{0.90}$	66	17.2	1.00	176	25.9	1.00	176	30.0	0.99	436	38.6	1.00
$\alpha_{0.95}$	66	16.3	1.00	176	26.3	1.00	176	27.3	0.99	434	40.3	0.98
$\alpha_{0.99}$	66	16.3	1.00	176	28.9	0.99	176	27.3	0.99	438	45.7	0.97

Sensitivity analysis. Table 2 shows the sensitivity of efficiency and accuracy to the parameters α and β for Scenario S3. Our results are only slightly affected by these parameters. A higher β reduces the number of evaluations and computation time, while accuracy stays high with MCC values near 1.00. This is expected, since a larger uncertainty fraction reduces the need to classify points near the transition between robust and non-robust spaces. Changes in α have a minor impact: the number of evaluations remains nearly constant, and the MCC stays close to 1.00. Only for high confidence levels (0.95 and 0.99) do we observe a small rise in computation time and a slight drop in MCC of 1 to 2 percentage points.

Overall, our evaluation shows that HQT can accurately approximate the robustness space of a business process across several scenarios when compared with naive GS and provides a favorable efficiency–accuracy trade-off against other baselines. It offers an efficient and effective method for assessing business process robustness and represents an initial step toward more robust business processes.

5 Related Work

In this section, we discuss work relevant to business process robustness assessment from three perspectives: robustness in other domains, related process characteristics, and approaches in BPM addressing similar search challenges.

Robustness in other domains. Studying robustness has a long tradition in science, yet "is rich in still insufficiently studied methodological and philosophical implications" [28]. In domains where it has been studied—software systems [10],

biological [3] and pharmaceutical processes [9], and production systems [27]—robustness is typically understood as the ability of a system to continue functioning properly under varying conditions. In business process management (BPM), robustness has largely been considered only as a property of algorithms, denoting their ability to produce outcomes with guarantees despite imperfect input, such as noise [18]. As a property of the process itself, however, it has not been assessed.

Related process characteristics. Business process robustness relates to two process characteristics, process flexibility and resilience, both of which concern how processes respond to changes. *Flexibility* refers to "the ability to react to changes" [8], emphasizing active adaptation to changes. Robustness, by contrast, refers to the extent to which a process can withstand changes without requiring adaptation. *Resilience*, in turn, describes a process's ability to restore performance after a disruption [13]. While resilience addresses short-term disruption, robustness emphasizes long-term stability and requires continuous performance within acceptable bounds rather than returning to them after a deviation.

Related approaches in BPM. Several approaches in BPM relate to our work, either by addressing a closely connected problem using BPS or by employing a similar search strategy but addressing a different problem. The closest line of related approaches concerns *process optimization*. These approaches use BPS to evaluate different parameter configurations, exploring varying activity batching or resource allocation policies, to identify settings that produce performance-optimal outcomes [20, 22, 23]. Although methodologically related, these approaches aim to locate an optimal configuration rather than to search for the full space of configurations that satisfy given performance requirements. Another related approach applies *tree-based adaptive search* to find parameter configurations of Markov models [11], which are also commonly used in process prediction [14]. While the search strategy aligns with ours, it is applied to formally verify properties of a given model, whereas our work uses it to assess the robustness of a business process based on simulation. Together, these studies illustrate approaches that are related, but none of them address the problem of identifying the set of parameter configurations under which a process consistently meets its performance requirements.

6 Conclusion

In this paper, we introduced the problem of business process robustness assessment and proposed an approach for its simulation-based approximation that leverages information recorded about the process in an event log. Our approach efficiently identifies the robustness space, i.e., the range of parameter configurations under which a process maintains acceptable performance. The evaluation demonstrated that our approach substantially reduces assessment complexity compared to baseline techniques while accurately identifying parameter combinations under which a process meets its performance targets. The approach provides clear, decision-oriented insights into how parameter variations affect performance and offers a quantitative basis for managing processes under change.

Our approach and its evaluation have some limitations. First, the approach assumes that simulation tools are accurate, which may not always be the case. Second, it also assumes that process performance is monotonic in the process parameters. While this is generally reasonable, there may be cases where the assumption does not hold. For example, adding an extra resource can reduce team productivity for certain process types due to increased communication and coordination effort. Third, the evaluation of our approach is based on a single process, with an illustrative set of parameters and performance indicators. In future work, we aim to relax the monotonicity assumption by exploring search strategies that do not rely on it and remain effective in more general settings. We also seek further optimization by adjusting the number of simulation runs and simulated cases based on the stability of the performance targets over time. Building on these extensions, we plan to evaluate the approach on a broader set of real-life processes and parameters. Despite these limitations, our work provides a foundation for a systematic assessment of business process robustness.

References

1. Van der Aalst, W.: Business process simulation survival guide. In: Handbook on Business process management 1: introduction, methods, and information systems, pp. 337–370. Springer (2014)
2. Van der Aalst, W.: Process Mining: Data Science in Action. Springer, Berlin Heidelberg (2016)
3. Becker, L., Sturm, J., Eiden, F., Holtmann, D.: Analyzing and understanding the robustness of bioprocesses. Trends in Biotechnology **41**(8), 1013–1026 (2023)
4. Brocke, J.v., van der Aalst, W., et al.: Process science: the interdisciplinary study of socio-technical change. Process Science **1**, 1 (2024)
5. Camargo, M., Báron, D., Dumas, M., González-Rojas, O.: Learning business process simulation models: A hybrid process mining and deep learning approach. Information Systems **117**, 102248 (2023)
6. Camargo, M., Dumas, M., González-Rojas, O.: Automated discovery of business process simulation models from event logs. Decision Support Systems **134**, 113284 (2020)
7. Chapela-Campa, D., Benchekroun, I., Baron, O., Dumas, M., Krass, D., Senderovich, A.: A framework for measuring the quality of business process simulation models. Information Systems **127**, 102447 (2025)
8. Dumas, M., Rosa, M., Mendling, J., Reijers, H.: Fundamentals of business process management. Springer (2018)
9. Glodek, M., Liebowitz, S., McCarthy, R., McNally, G., Oksanen, C., Schultz, T., Sundararajan, M., Vorkapich, R., Vukovinsky, K., Watts, C., et al.: Process robustness—a pqri white paper. Pharm. Eng **26**(6), 1–11 (2006)
10. Gribble, S.: Robustness in complex systems. In: Proceedings Eighth Workshop on Hot Topics in Operating Systems. pp. 21–26 (2001)
11. Junges, S., Ábrahám, E., Hensel, C., Jansen, N., Katoen, J.P., Quatmann, T., Volk, M.: Parameter synthesis for markov models: covering the parameter space. Formal methods in system design **62**(1), 181–259 (2024)
12. Kirchdorfer, L., Blümel, R., Kampik, T., van der Aa, H., Stuckenschmidt, H.: Discovering multi-agent systems for resource-centric business process simulation. Process Science **2**(1), 4 (2025)

13. Kraus, A., Rehse, J., van der Aa, H.: Data-driven assessment of business process resilience. Process Science **1**(1), 4 (2024)
14. Lakshmanan, G.T., Shamsi, D., Doganata, Y.N., Unuvar, M., Khalaf, R.: A markov prediction model for data-driven semi-structured business processes. Knowledge and information systems **42**(1), 97–126 (2015)
15. Lamperti, J.: Stochastic processes: a survey of the mathematical theory, vol. 23. Springer Science & Business Media (2012)
16. Laszlo, M., Mukherjee, S.: A genetic algorithm using hyper-quadtrees for low-dimensional k-means clustering. IEEE transactions on pattern analysis and machine intelligence **28**(4), 533–543 (2006)
17. Law, A., Kelton, D.: Simulation modeling and analysis. Mcgraw-hill NY (2007)
18. Leemans, S.: Robust Process Mining with Guarantees - Process Discovery, Conformance Checking and Enhancement, Lecture Notes in Business Information Processing, vol. 440. Springer (2022)
19. López-Pintado, O., Rosenbaum, J., Berx, J., Dumas, M.: Optimos: A tool for simulation-driven business process optimization. In: BPM (Demos/Resources Forum). pp. 126–130 (2024)
20. López-Pintado, O., Rosenbaum, J., Dumas, M.: Optimization of activity batching policies in business processes. In: BPM. pp. 397–414. Springer (2025)
21. Matthews, B.: Comparison of the predicted and observed secondary structure of t4 phage lysozyme. Biochimica et Biophysica Acta (BBA) - Protein Structure **405**(2), 442–451 (1975)
22. Meneghello, F., Middelhuis, J., Genga, L., Bukhsh, Z., Ronzani, M., Di Francescomarino, C., Ghidini, C., Dijkman, R.: Optimizing resource allocation policies in real-world business processes using hybrid process simulation and deep reinforcement learning. In: BPM. pp. 167–184. Springer (2024)
23. Middelhuis, J., Bianco, R., Sherzer, E., Bukhsh, Z., Adan, I., Dijkman, R.: Learning policies for resource allocation in business processes. Information Systems **128**, 102492 (2025)
24. Polyvyanyy, A., Moffat, A., García-Bañuelos, L.: An entropic relevance measure for stochastic conformance checking in process mining. In: ICPM. pp. 97–104 (2020)
25. del Río-Ortega, A., Resinas, M., Cabanillas, C., Ruiz-Cortés, A.: On the definition and design-time analysis of process performance indicators. Information Systems **38**(4), 470–490 (2013)
26. Samet, H.: The quadtree and related hierarchical data structures. ACM Computing Surveys (CSUR) **16**(2), 187–260 (1984)
27. Stricker, N., Lanza, G.: The concept of robustness in production systems and its correlation to disturbances. Procedia CIRP **19**, 87–92 (2014)
28. Wimsatt, W.: Robustness, Reliability, and Overdetermination (1981), pp. 61–87. Springer Netherlands, Dordrecht (2012)