

From Global Policies to Local Strategies: Multi-Objective Optimization of Resource-Specific Handover Policies

Lukas Kirchdorfer^{1,2}[0000–0003–4713–9328], Artemis Doumeni³, Han van der Aa⁴[0000–0002–4200–4937], and Hugo A. López³[0000–0001–5162–7936]

¹ SAP Signavio, Walldorf, Germany
`lukas.kirchdorfer@sap.com`

² Data and Web Science Group, University of Mannheim, Germany

³ DTU Compute, Technical University of Denmark, Kongens Lyngby, Denmark
`hulo@dtu.dk`

⁴ Faculty of Computer Science, University of Vienna, Vienna, Austria
`han.van.der.aa@univie.ac.at`

Abstract. Efficient resource allocation is a key challenge in business process management, with direct implications for cost, throughput time, and utilization. While recent Reinforcement Learning (RL) approaches have shown promise in deriving adaptive allocation policies, they typically neglect inter-resource collaboration patterns that can strongly influence real-world task handovers. Recognizing this, this paper introduces the first approach for multi-objective optimization of resource-level decision-making, enabling the recommendation of person-specific handover policies. To achieve this, our work combines an existing Multi-Agent System-based process simulator with a multi-objective evolutionary algorithm. The resulting approach produces Pareto-optimal, resource-specific policies that optimize the process across multiple objectives. Experimental results on synthetic and real-world datasets show that our approach reduces costs by an average of 37% and waiting time by 58%, consistently outperforming heuristic baselines and demonstrating the potential of leveraging collaboration-aware optimization to improve process performance.

Keywords: Process mining · Multi-objective optimization · Resource allocation · Simulation · Multi-agent systems.

1 Introduction

Efficient resource allocation—the task of assigning resources, such as employees, machines, and systems, to process activities—is a central challenge in business process management, as it directly affects key performance indicators such as cost, throughput time, and resource utilization [24]. However, finding effective allocations is difficult due to the complexity and interdependence of concurrent process instances, the need for decisions under dynamic conditions, and the

long-term consequences of allocation choices [14]. Accordingly, a wide range of approaches has been proposed, ranging from rule- and scheduling-based methods [24] to more recent Reinforcement Learning (RL)-based approaches [2,14,23], which showed promising results by balancing short- and long-term objectives.

Despite this progress, existing RL-based approaches exhibit a key limitation: they learn *global* resource-allocation policies based only on (i) which tasks are awaiting execution and (ii) which resources are available to perform them. This allows them to exploit differences in skill, execution time, and cost, but it treats resources as passive recipients of assignments and ignores an important aspect of real organizational behavior: collaboration patterns among resources. In practice, however, resources—especially human resources—are active decision-makers during process execution, influencing not only which work they perform, but also with whom they collaborate. These inter-resource dependencies give rise to characteristic handover and collaboration patterns that shape how work flows through an organization. Concretely, we are interested in *handover policies* that specify, for a given resource and completed activity, to which resource the case should be passed next for the subsequent activity. While accounting for such *local*, resource-specific handovers has been shown to improve the fidelity of as-is process simulations [9], existing optimization approaches do not account for these local handovers, and therefore overlook collaboration structure as a leverage for process improvement.

This paper introduces the first approach for grounded multi-objective optimization of resource-specific handover policies in business processes. Our approach takes as input a Multi-Agent System (MAS)-based simulation model of the process that explicitly captures resource interdependencies, building on prior work [9]. On top of this model, we develop an analysis-and-design optimization framework that integrates the widely used multi-objective evolutionary algorithm NSGA-II [4] to search for Pareto-optimal handover policies. This approach offers two key advantages. First, it can be grounded in data, since the MAS simulation model taken as input can be discovered from event logs (cf. [9]). Second, it is generic with respect to any set of objectives, yielding a Pareto front of handover policies that offer practitioners actionable alternatives that reflect different operational trade-offs. To increase solution diversity, we incorporate four mutation variants into the genetic algorithm. Experiments on synthetic and real-world datasets, using cost and time as example objectives, demonstrate substantial improvements over the as-is process, with average reductions of 37% in cost and 58% in waiting time, also outperforming established heuristic baselines.

The remainder of this paper is structured as follows: Section 2 reviews related work, Section 3 defines the problem, Section 4 presents our approach, and Section 5 reports the evaluation. Finally, Section 6 concludes the paper.

2 Related Work

Our approach to optimizing resource-specific handover policies relates to three main research directions.

Resource allocation. A large body of work has addressed resource allocation in business processes, mostly focusing on optimizing task–resource assignments during process execution, i.e., at runtime. Early approaches rely on rule-based assignment and scheduling heuristics [24], which depend on predefined rules or short-term objectives and therefore adapt poorly to dynamic process conditions. Prescriptive Process Monitoring (PrPM) approaches [18, 22] extend these methods by recommending resources for ongoing cases, yet they still optimize decisions locally and often ignore their broader process-level impact.

To address longer-term and process-level effects, more advanced work integrates simulation and applies metaheuristic or learning-based optimization methods, such as genetic algorithms [6, 10, 17], heuristic search [13], or RL [8, 14, 15, 23]. Yet, only a subset considers multi-objective optimization as addressed in this paper. For example, Huang et al. [8] use RL to optimize cost and cycle time.

While our approach can be employed at runtime to determine handover decisions (which we will do in our evaluation), in principle, it optimizes a static model of resource collaboration, independently of runtime case information. Thus, we consider it as design-time. Within this design-time line of work, several approaches optimize the number of required resources per resource pool [7, 13, 16], solving a staffing problem rather than optimizing handovers. Closest to our work, Bejarano et al. [1] also address a design-time, simulation-based, multi-objective optimization problem, but their decision variables remain activity-to-resource assignments, whereas ours are handover decisions between resources.

In summary, a crucial distinction sets our approach apart from all existing work: the underlying representation of resources. Existing methods treat resources as passive entities that are assigned to activities; optimization thus centers on determining which resource should execute which task. In contrast, we adopt a MAS perspective in which resources are first-class citizens, modeled as autonomous decision-making agents capable of choosing their collaboration partners. Consequently, instead of learning a single global allocation rule, our approach discovers *resource-specific handover policies* that capture and optimize the collaborative structure of the process. This enables a form of collaboration-aware optimization that has not been addressed in previous research.

Organizational and agent system mining. Complementary to optimization approaches, organizational and resource-oriented process mining methods analyze how resources collaborate [19]. More recently, agent system mining has been proposed to discover process control-flow models from an agent perspective [21]. However, these approaches remain descriptive: they reveal collaboration structures but do not treat them as decision variables for optimization.

Agent-based business process simulation. Traditional business process simulation typically follows a control-flow-first perspective, extending a process model (e.g., a Petri net) with temporal and resource-related parameters [11]. In contrast, agent-based approaches adopt a resource-first perspective in which resources are modeled as autonomous entities at the core of the simulation [9]. While such agent-based simulations can be used in a *what-if* manner to explore alternative collaboration patterns, this requires manually defining and evaluating

different configurations. Our approach automates this process by systematically exploring handover policies and presenting the user with a set of Pareto-optimal collaboration patterns and their corresponding trade-offs.

3 Problem Definition

Event logs. Event logs represent a key artifact of our approach, generated during the search for optimal resource policies. We define an event log L as a collection of traces. A trace $\sigma \in L$ is a finite sequence of events, $\langle e_1, \dots, e_n \rangle$, recording the execution of activities performed for a single case in a process. Each event e_i is a tuple $(case, act, ts_{start}, ts_{end}, res)$, where $case$ is the case’s identifier, act is the activity to which the event corresponds, ts_{start} and ts_{end} , respectively, are the start and end timestamps of the activity’s execution, and res is the resource that executed the activity.

Pareto optimization. Our approach aims to improve resource handover policies by optimizing a set of $k > 1$ objective functions. In our setting, an event log L captures the realization of the resource handover policy applied during the execution of its cases. Thus, we define the cost vector of an event log L as

$$\mathbf{c}(L) = [f_1(L), \dots, f_k(L)], \quad \text{where } f_i(L) = \sum_{\sigma \in L} f_i(\sigma), \quad f_i : L \rightarrow \mathbb{R}. \quad (1)$$

Since our goal is to minimize all components of the cost vector $\mathbf{c}(L)$, we seek to minimize the corresponding set of objective functions $\mathcal{K} = \{f_1(L), \dots, f_k(L)\}$. As these objectives are frequently in conflict, we focus on identifying solutions that represent optimal trade-offs, formalized through the notion of *Pareto optimality*. Let \mathcal{L}_C denote the set of all event logs defined over a set of cases C . For two logs $L, L' \in \mathcal{L}_C$, we follow [5] and say that L *dominates* L' if

$$\forall i \in 1, \dots, k : f_i(L) \leq f_i(L') \quad \wedge \quad \exists j \in 1, \dots, k : f_j(L) < f_j(L'). \quad (2)$$

In other words, L performs at least as well as L' on all objectives and strictly better on at least one. A log $L \in \mathcal{L}_C$ is *Pareto optimal* if no log $L' \in \mathcal{L}_C$ exists that dominates it. The collection of all non-dominated logs constitutes the *Pareto front*, representing the set of optimal trade-offs across the k objectives.

The resource allocation problem. Let \mathcal{A} denote a set of agents, where each agent $a \in \mathcal{A}$ corresponds to a resource $res \in \mathcal{RES}_L$ involved in the process. Likewise, let \mathcal{ACT}_L denote the set of all activities in the process, where $act, act' \in \mathcal{ACT}_L$. We denote each tuple (a, act, act') a *decision point* for agent a after completing activity act , in the sense that it needs to hand over to some agent a' to work on act' . To model this decision, we define a *handover policy* π as a conditional probability distribution:

$$\pi(a' \mid a, act, act') \quad \text{where } a, a' \in \mathcal{A}, \text{ and } act, act' \in \mathcal{ACT}_L. \quad (3)$$

Optimizing π thus corresponds to searching for a handover pattern that yields improved process performance with respect to the multi-objective cost vector defined in Equation 1.

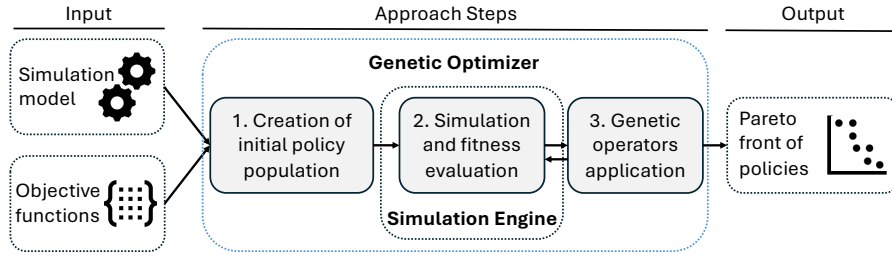


Fig. 1: Overview of the proposed approach.

4 Approach

This section introduces our genetic algorithm-based approach for discovering Pareto-optimal resource handover policies, as illustrated in Figure 1. Its input, main steps, and output are as follows:

Input. Our approach takes as input i) an agent-based simulation model \mathcal{M} and ii) a set of objective functions $\mathcal{K} = \{f_1(L), \dots, f_k(L)\}$, which instantiate the cost vector (see Equation 1) that guides the multi-objective optimization. We define the simulation model as $\mathcal{M} = (\mathcal{A}, \mathcal{S}, \pi_0, \beta)$, where \mathcal{A} is the set of agents, \mathcal{S} captures global simulation parameters, π_0 is the as-is handover policy, and β is the control-flow policy that determines the next activity in a running case. Each agent $a \in \mathcal{A}$ is characterized by its weekly availability calendar and its capabilities, comprising the set of activities it can perform and the corresponding execution-time distributions. The tuple \mathcal{S} captures global simulation parameters such as the case interarrival distribution and the arrival calendar. The simulation model used in our approach can either be specified manually with domain experts or automatically discovered from an event log L . In this work, we employ *AgentSimulator* [9], which automatically derives a multi-agent simulation model from an event log and supports the simulation of the corresponding process, thereby also serving as the simulation engine. For a detailed description of the quality of the discovered simulation models, we refer to the original publication. Note that other approaches may also be used to instantiate the engine.

Approach steps. The genetic optimizer is the central component of our approach. It takes as input the as-is model \mathcal{M} with its handover policy π_0 and optimizes the policy with respect to the specified set of objective functions \mathcal{K} . The optimization workflow consists of three steps:

1. *Creation of initial policy population.* Based on the as-is policy π_0 , we generate an initial population of handover policies, denoted by $\mathcal{P} = \{\pi_1, \pi_2, \dots, \pi_N\}$, consisting of N candidate policies.
2. *Simulation and fitness evaluation.* For each candidate policy $\pi_n \in \mathcal{P}$ in the population, the optimizer calls the simulation engine to simulate multiple runs. These runs yield event logs that are used to compute a multi-objective fitness score for the underlying policy, as defined in Equation 1.

3. *Genetic operators application.* The optimizer uses the feedback from the fitness evaluation to apply evolutionary operators to its population, generating a new, potentially superior set of candidate policies.

We employ NSGA-II [4], a genetic algorithm specialized for multi-objective optimization, to guide the evolutionary search. We instantiate NSGA-II to operate on individuals in the population that encode full probabilistic handover policies, and adapt the algorithm through custom genetic operators designed to preserve and meaningfully modify these probability distributions. It is important to note that while the selection of NSGA-II has been driven by speed and usability concerns against alternatives (e.g., NSGA-I [20], SPEA [25]), our approach is independent of the chosen algorithm.

Output. Our approach repeats steps 2 and 3 a predefined number of iterations, controlled by a hyperparameter $G \in \mathbb{N}$. After G generations, the algorithm yields a Pareto front of handover policies $\mathcal{P}^* = \{\pi_1, \pi_2, \dots, \pi_M\}$, that indicate how resources should collaborate to achieve an optimal trade-off.

In the remainder, we describe the details of its core steps: the creation of an initial population, the simulation and fitness evaluation, and the genetic operators.

4.1 Creation of the Initial Policy Population

The first step of the genetic optimizer involves constructing an initial population of handover policies, derived from the as-is policy π_0 . In this setting, each policy constitutes an *individual* (also known as the *chromosome*) in the population of the genetic algorithm.

Search space pruning. To reduce computational complexity, we prune the search space through a preprocessing step. We analyze the discovered handover policy and classify each decision point—i.e., each situation where a resource finishes an activity and must select a successor (see definition in Section 3)—based on the determinism of that decision point:

1. **Fixed:** Decision points with only one successor. These reflect non-negotiable business rules and are excluded from optimization.
2. **Variable:** Decision points with multiple successors. These represent flexible parts of the process and form the search space for optimization.

Therefore, we reduce the search space while preserving all meaningful degrees of freedom. Note that the fixed part of the policy π_{fix} will be used again during the simulation and fitness-evaluation steps.

Initial population. To balance *exploration* and *exploitation*, we construct an initial population \mathcal{P} of size N based on the variable part of the as-is handover policy π_0 . As a heuristic, half of the individuals are replaced with fully random policies to ensure broad coverage of the search space. The other half is generated by mutating the as-is policy, producing candidates that remain close to the observed process behavior. Finally, one individual is set to an exact copy of the as-is policy to ensure it is included. This hybrid strategy provides both diverse and domain-informed starting points for the evolutionary search.

Algorithm 1 Simulation and Fitness Evaluation

```

1: procedure EVALUATEFITNESS(var. policy  $\pi_n$ , fix. policy  $\pi_{\text{fix}}$ ,  $\mathcal{M}$ ,  $T$ ,  $\mathcal{K}$ )
2:    $\pi_n \leftarrow \text{MERGEPOLICYPARTS}(\pi_n, \pi_{\text{fix}})$       ▷ Reconstruct full handover policy
3:    $\mathcal{C} \leftarrow \emptyset$                                 ▷ Initialize set of cost vectors for all runs
4:   for  $t \leftarrow 1$  to  $T$  do
5:      $\mathcal{M}_t \leftarrow \text{COPY}(\mathcal{M})$                     ▷ Instantiate simulation model for run  $t$ 
6:      $\mathcal{M}_t.\pi \leftarrow \pi_n$                         ▷ Overwrite initial handover policy
7:      $L_n^t \leftarrow \text{RUNSIMULATION}(\mathcal{M}_t)$           ▷ Simulated event log
8:      $\mathbf{c}_t \leftarrow \text{CALCULATEMETRICS}(L_n^t, \mathcal{K})$     ▷ Cost vector for run  $t$ 
9:      $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathbf{c}_t\}$                 ▷ Append to set of cost vectors
10:   $\bar{\mathbf{c}} \leftarrow \text{VECTORMEAN}(\mathcal{C})$               ▷ Compute fitness as average cost vector
11:  return  $\bar{\mathbf{c}}$ 

```

4.2 Simulation and Fitness Evaluation

After generating the initial population of handover policies \mathcal{P} , the next step evaluates the fitness of each individual by means of simulation. The fitness evaluation treats the simulator as a black-box function whose outputs are then assessed. As summarized in Algorithm 1, the **fitness of a single chromosome** is computed through three stages: policy reconstruction, simulation, and evaluation.

Policy reconstruction. Each $\pi_n \in \mathcal{P}$ represents only the variable part of the handover policy. To obtain a complete policy compatible with the simulator, we merge this variable portion with the fixed part π_{fix} (Line 2) identified during the search space pruning. The result is a full, comprehensive handover policy that specifies all resource handover decisions in the process.

Simulation. Using the reconstructed policy $\pi_n \in \mathcal{P}$, we instantiate a new simulation model \mathcal{M}_t (Line 5). Specifically, we begin with the parameters of the initial simulation model \mathcal{M} —preserving all settings such as arrival rates and resource schedules—and overwrite only the handover policy parameter (Line 6). This yields a complete, valid model that can be executed to generate a simulated event log (Line 7). To mitigate stochastic variance and obtain stable performance estimates, the simulation is run multiple times for each policy, independently generating several event logs. Thus, per policy π_n , we generate T event logs, denoted by L_n^1, \dots, L_n^T .

Evaluation. For each simulated event log L_n^t , we compute the cost vector \mathbf{c}_t (Line 8) defined in Equation 1. The final fitness score of the policy π_n is obtained by averaging its cost vectors across all simulated logs L_n^1, \dots, L_n^T , resulting in a robust multi-objective fitness estimate (Line 10).

4.3 Genetic Operators

Having computed a fitness score for each policy in the population, the next step in our approach is to apply genetic operators to the current population \mathcal{P} . In evolutionary computation, genetic operators are mechanisms that modify or recombine individuals to drive the search process. Inspired by biological evolution,

they introduce variation, explore the search space, and steer the population toward higher-quality solutions [4]. In our approach, we employ NSGA-II’s three standard operators—selection, crossover, and mutation—applied sequentially, and customize the latter two to operate on probabilistic handover policies.

Selection. First, the algorithm performs a selection step to retain only the most promising individuals for survival and reproduction through the subsequent crossover and mutation operators. Individuals are ranked into non-dominated fronts based on their fitness scores. Afterwards, pairs of randomly drawn policies are compared, and the one with the superior rank is selected. If both belong to the same front, ties are broken using the *crowding distance*, which measures how isolated a policy is from its neighbors in the objective space, thereby favoring solutions in less-explored regions. This combination of non-domination ranking and crowding-based tie-breaking ensures that NSGA-II selects individuals that promote both convergence toward the Pareto front and diversity along it.

Crossover. Given the selected subset of policies, the crossover operator combines two candidates to produce new policies (*offspring*) that inherit characteristics from both *parents*. In our context, a handover policy consists of a set of decision points, each represented by a probability distribution over possible successor resources. Because mixing individual probabilities can yield invalid distributions, crossover must preserve each decision point as a coherent unit. Our custom operator therefore works at the level of entire decision points: for each decision situation, it randomly swaps the full probability distribution between the two parents. This ensures that the offspring inherit valid decision behaviors while still enabling the exploration of new combinations of strategies within the probabilistic constraints of the handover model. The crossover operation is applied to two parents with probability p_c . If there is no crossover, the offspring are exact clones of their parents. This hyperparameter p_c controls the degree of exploitation by determining how frequently the algorithm recombines strategies from high-performing solutions.

Mutation. Mutation serves as the main driver of novelty in our evolutionary search, applied to the previously generated offspring. Whereas crossover recombines existing decision strategies, mutation introduces new behavioral patterns by perturbing selected decision points within a handover policy. Its purpose is to prevent premature convergence and to support continuous exploration of alternative resource allocation behaviors. In the context of resource handover policies, different business environments may benefit from different mutation dynamics. Some scenarios require strong exploratory jumps to escape local optima, while others benefit more from careful refinements once the population has converged toward a promising region of the search space. To promote robust performance across diverse process structures, we implement several complementary mutation strategies. Mutation is applied to each offspring with probability p_m . If selected for mutation, one of its decision points is chosen at random, and one of four mutation variants is applied to the probability distribution at that point.

Random. The probability distribution at the selected decision point is reinitialized, providing a strong exploratory step by generating a completely new

handover strategy for that situation. Concretely, let the decision point specify a categorical distribution over its admissible successor resources in the variable part of the policy. The operator assigns a new positive weight to each successor, drawn independently from a uniform distribution, and normalizes these weights so that they again sum to one. Because this operator does not consult \mathcal{M} or the objective set \mathcal{K} , it can realize handover behaviors that are not reachable through small, heuristic-guided shifts. In our implementation, it plays the same role in initializing the population, where fully random variable policies are mixed with mutated variants of π_0 .

Greedy. Using a local heuristic derived from \mathcal{M} , the operator identifies the best and worst successor resources at the selected decision point and moves all probability mass away from the worst option—an aggressive exploitation step. Among the objectives in \mathcal{K} , the operator first draws one objective uniformly at random. It then scores each successor resource that carries a non-negligible probability at this decision point. The successors with the minimum and maximum scores are retained as best and worst, respectively. The entire probability allocated to the worst successor is set to zero and transferred to the best successor. If the best successor is associated with several successor activities, the incoming mass is distributed across them in proportion to their current probabilities (or uniformly if it currently receives no mass).

Guided. Also relying on the local heuristic above, this variant transfers only a small, random fraction of probability from the worst to the best successor. It is designed for fine-grained adjustments in an already promising region of the search space. After selecting the same guidable objective from \mathcal{K} and ranking successors as in greedy mutation, guided mutation draws a transfer share uniformly between 10% and 50% of the probability mass currently assigned to the worst successor. This amount is subtracted from the worst successor’s activities in proportion to their shares and added to the best successor’s activities using the same proportional rule as in the greedy operator. The worst successor, therefore, retains most of its former probability, yielding a conservative, heuristic-biased step along one dimension of \mathcal{K} while preserving a valid distribution at the decision point.

Hybrid. To balance exploration and exploitation, the operator selects one of the above strategies with predefined probabilities: 60% guided, 20% greedy, and 20% random. This weighting reflects the intuition that, as the search progresses, small refinements are often most effective, while still allowing for larger changes.

Together, these mutation strategies enable the genetic algorithm to flexibly adapt its search behavior, supporting both incremental improvements and substantial shifts in handover logic when needed. While we employ these four mutations, other heuristics can also be incorporated.

Thus, the genetic operators generate a new population of handover policies, which is then again used for simulation and fitness evaluation. We repeat this evolutionary cycle for a predefined number of G iterations, ultimately yielding a Pareto front of resource handover policies \mathcal{P}^* .

5 Evaluation

This section presents the experiments used to evaluate the performance of our multi-objective optimization approach. We instantiate the set of objective functions with *waiting time* and *labor cost*, thus assessing whether we can learn resource-specific policies that improve both cost and time compared to the as-is scenario and other baselines. For internal validity, we assess the effectiveness of our approach in a controlled setting designed to test its performance across diverse scenarios. For external validity, we evaluate the effectiveness of our approach on seven widely used event logs from business process simulation research, providing an ideal competitive benchmark that lets us assess whether our approach can learn more optimal policies for these processes. In the remainder, Section 5.1 describes the experimental setup, followed by the results in Section 5.2. Implementations and additional results can be found in our repository⁵.

5.1 Experimental Setup

The experimental setup is identical across both experiments, differing only in the datasets used.

Internal validity. To investigate the performance of our approach, we study 5 variants of an existing synthetic *Loan Application* process [3], allowing us to assess the capability of our approach to learn effective policies across varying process configurations. The process consists of 12 activities, including parallelism, choices, and a loop. By default, the process is executed by 19 distinct resources, divided into 6 roles. To create the variants, we employ *AgentSimulator* [9] to discover a simulation model of the as-is process from the original event log, which we then modify. For each variation of the simulation model, we generate an event log containing 1000 cases, consistent with the size of the original log. Specifically, we create the following variants:

- Loan_{AD} : Cost and duration of activities are equal for all resources.
- Loan_{JS} : For each activity, resources are split into junior and senior employees, where juniors are cheaper but require more time to execute the task.
- Loan_{RC} : Creating resource contention by halving the available resources.
- Loan_{SCDT} : All resources cost the same, but differ in execution times.
- Loan_{ARR} : Increasing workload through a higher rate of case arrivals.

External validity. To study how our approach could optimize logs in the process mining community, we employ seven widely used event logs (see Table 1) from business process simulation research [9, 12] as a basis for optimization scenarios. They are well-suited for our analysis as they provide both start and end timestamps for each event, which is required for simulation. The datasets span multiple domains, including financial services, education, and procurement, and differ in size and complexity (datasets available in our repository).

A limiting factor for external validity is that existing logs do not include information on resource cost allocation. This information is essential for our

⁵ <https://github.com/lukaskirchdorfer/BPS-MAS-Handover-Optimizer>

Table 1: Description of event log properties.

Log	Type	Traces	Events	Activities	Agents	Avg. cycle time
Loan App	syn	1000	7492	12	19	10
P2P	syn	608	9119	21	27	515
C1000	syn	1000	38160	42	14	22
C2000	syn	2000	77418	42	14	20
ACR	real	954	6870	18	432	357
BPI12W	real	8616	59302	6	52	214
BPI17W	real	30276	240854	8	136	304

multi-objective analysis, for which we decided to provide an approximation. We define hourly cost values for each resource. While assigning uniform costs to all resources provides a simple baseline, it prevents the exploration of cost-time trade-offs. Therefore, we derived a performance-correlated cost model for each event log. This reflects the real-world principle that more efficient or specialized resources are typically more expensive. To achieve this, we created a performance profile for each resource based on skills and activity durations. We then clustered the profiles into five distinct tiers using k-means, assigned an hourly cost range to each tier with higher-performing tiers associated with higher costs (e.g., Tier 1: 10-25\$/hr, . . . , Tier 5: 76-90\$/hr). We randomly sampled a value for each resource from its corresponding cost range.

Benchmark approaches. We compare against several baselines⁶:

- **As-Is:** The as-is resource handover patterns discovered by AgentSimulator.
- **Avail:** Always hand over to the earliest available resource.
- **Random:** Select a random resource.
- **LC:** Always choose the cheapest resource.
- **SPT:** Select the resource with the shortest expected processing time.

Hyperparameters. We use the following hyperparameter configuration for all experiments. We use a population size $N = 100$ and run $G = 100$ iterations. To account for simulation stochasticity, each policy is evaluated using $T = 3$ simulated logs. The probability that crossover is applied is set to $p_c = 70\%$. The probability that a mutation is applied to an offspring is set to $p_m = 30\%$. As detailed in Section 4, we employ four mutation variants for our genetic operator (*Random, Greedy, Guided, Hybrid*), all of which we evaluate in our experiments.

Metrics. We use two types of metrics. The first assesses overall process performance under different handover policies by measuring total process cost in \$ (sum of event durations in hours multiplied by the resource’s hourly rates) and waiting time (WT), defined as the sum of idle times. The second type assesses the

⁶ While we initially wanted to include [8] in our evaluation, their method employs tabular Q-learning, which requires storing all state-action values and thus does not scale to realistic BPM state spaces. In addition, the approach lacks a reproducible implementation and was evaluated only in a custom simulator. We therefore exclude it from the empirical comparison.

Table 2: Results for the *Loan Application* variants. The best results per variant are highlighted in bold. Cost measured in thousand \$, WT in hours.

	As-Is		Avail		Random		LC		SPT		Ours	
	Cost	WT	Cost	WT	Cost	WT	Cost	WT	Cost	WT	Cost	WT
Loan _{AD}	213	16.81	161	0.43	238	1.18	145	6.22	164	23.93	120	0.52
Loan _{JS}	246	1.46	231	0.65	333	1.39	194	26.79	212	1.00	166	0.59
Loan _{RC}	67	2.30	74	2.20	76	2.10	59	4.14	85	2.57	49	2.03
Loan _{SCDT}	254	2.78	209	0.71	287	1.27	204	0.65	237	26.85	179	0.60
Loan _{ARR}	88	0.74	83	0.59	81	0.73	57	3.84	97	1.38	53	0.69

quality of the discovered Pareto fronts, evaluating how effectively each mutation variant explores a diverse and high-quality set of optimal solutions, following the methodology of López-Pintado et al. [13]. These metrics include the Hyperarea Ratio, Purity, Averaged Hausdorff Distance, and Delta Spread. As this analysis is not our primary focus, we summarize the key findings here, while detailed metric definitions and results are available in our repository.

5.2 Results

Internal validity. Table 2 shows the performance of the different approaches in terms of WT and cost across the variants of the *Loan Application* process. Unlike the baseline methods, our approach produces an entire set of trade-off solutions. For comparability, we report the performance of the *most balanced solution* from the discovered Pareto front, defined as the solution with the smallest Euclidean distance to the ideal point in the objective space (i.e., zero cost and zero WT).

Overall, our approach consistently outperforms the as-is baseline in both cost and time across all five scenarios. For example, in the Loan_{SCDT} scenario—where resources share identical hourly costs but differ in execution times—our approach reduces the total cost from \$254k to \$179k and the WT from 2.78 to 0.60 hours. This demonstrates that our approach effectively enhances individual handover policies by prioritizing faster resources while still engaging slower ones to prevent queues and excessive waiting times.

Compared to the other baselines, our approach achieves the lowest cost in all scenarios and the lowest waiting time in three scenarios, while ranking second-best in the remaining two. These results demonstrate that we learn more effective resource handover policies than those currently implemented. Notably, it also surpasses greedy heuristics such as *LC*, which always selects the cheapest resource, by accounting for the long-term consequences of assignment decisions and jointly optimizing cost and temporal performance.

To better understand these effects, we analyze the resulting handover patterns for the Loan_{JS} log in Figure 2, which captures how often resources hand over work to others in the simulated event logs based on the *As-Is*, *LC*, and

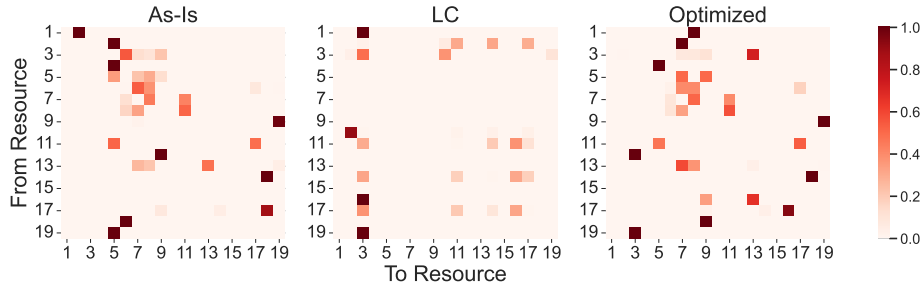


Fig. 2: Resource handover frequencies in the Loan_{JS} log based on the *As-Is*, *LC*, and our optimized policy. Darker cells indicate more frequent handovers.

our optimized policy. The differences between the three are striking. In the *As-Is* baseline, for example, resource 1 always hands over to resource 2. In the *LC* baseline, which optimizes purely for cost, resource 1 instead always hands over to resource 3—who performs the same role but at a lower hourly rate. However, as all resources follow this greedy strategy, costly but faster resources are excluded entirely, leading to severe queue formation, as evident from the many blank cells in Figure 2. Our approach, in contrast, learns a more balanced policy. Again, looking at resource 1, it now primarily hands over to resource 8—a senior costing \$150/hr compared to resource 2’s \$50/hr, but completing tasks six times faster. This makes resource 8 both quicker and more cost-efficient overall. Moreover, unlike the *LC* baseline, where work accumulates at cheap but slow resources, our approach learns individual policies for each resource that, through multi-agent interactions, lead to more complementary collaboration patterns and a more balanced task distribution, resulting in a globally superior outcome.

External validity. The results across the seven existing event logs are presented in Table 3. Overall, consistent with the findings from the first experiment, our approach achieves the best overall performance across most event logs. Importantly, it improves cost and waiting time compared to the *As-Is* baseline in all datasets, underscoring the robustness of our approach in balancing efficiency and cost-effectiveness across diverse real-world settings. While our approach yields particularly robust results in terms of WT, it does not always attain the lowest cost, ranking above the *LC* baseline in four of the seven datasets. Moreover, these gains come at the cost of higher computation time: whereas the baselines require about three minutes on average, our approach is roughly 35 times slower (cf. Table 5). Like the baselines, runtime increases with the number of events and agents in the log. At the same time, the optimization component scales comparably: for example, BPI17W takes 39 times longer than LoanApp for the baselines and 41 times longer for our approach.

We also used these event logs to investigate the performance of the four different mutation variants of our genetic operator in Table 4, reporting the most balanced solution per variant. This comparison reveals that the *Guided* variant achieves the greatest average improvement across all datasets for both cost and WT relative to the *As-Is* baseline. However, the performance gap between vari-

Table 3: Results over the existing logs. The best results per log are highlighted in bold. Cost measured in thousand \$, WT in hours.

	As-Is		Avail		Random		LC		SPT		Ours	
	Cost	WT	Cost	WT	Cost	WT	Cost	WT	Cost	WT	Cost	WT
LoanApp	91	0.71	91	0.64	90	0.70	62	2.70	88	0.85	70	0.61
P2P	98	17.10	116	0.79	144	2.91	61	15.26	103	2.45	44	3.62
C1000	160	1.68	133	1.58	170	1.81	120	1.53	121	1.49	98	0.75
C2000	316	1.05	251	1.03	297	1.05	247	1.22	237	1.03	211	0.71
ACR	103	8.97	140	0.89	232	1.01	23	3.50	96	1.64	45	0.51
BPI12W	244	0.03	177	0.02	265	0.20	24	0.04	113	0.07	176	0.02
BPI17W	654	0.38	714	0.02	675	0.16	209	2.61	317	0.43	480	0.01

Table 4: Relative improvement in cost and WT per mutation variant compared to the As-Is baseline, averaged across the seven existing logs.

Random		Greedy		Guided		Hybrid	
Cost	WT	Cost	WT	Cost	WT	Cost	WT
33.46%	51.64%	32.84%	54.56%	36.88%	57.71%	33.07%	55.07%

ants is small, indicating that the approach is stable with respect to the mutation strategy. An analysis of the quality of the resulting Pareto fronts reveals that, while no single variant consistently dominates, the *Guided* and *Hybrid* variants stand out, whereas *Random* and *Greedy* tend to produce less dominant solutions.

Table 5: Runtimes for all 7 logs (in minutes).

	LoanApp	P2P	C1000	C2000	ACR	BPI12W	BPI17W	Average
Baselines	0.28	0.35	1.78	3.58	1.72	2.17	10.83	2.96
Ours	9.33	16.03	70.67	142.60	41.83	67.31	379.17	103.85

We further illustrate the solution space and Pareto front for the real-life ACR event log in Figure 3, comparing our solutions with the baselines. The *Guided* and *Hybrid* variants clearly dominate, offering Pareto-optimal trade-offs with process costs between \$22k and \$85k and waiting times from 0.2 to nearly 2 hours. While the most balanced solution reported in Table 3 corresponds to \$45k and 0.51 hours, our approach offers a diverse set of high-quality alternatives, enabling practitioners and process owners to select policies aligned with their specific operational goals. For instance, if minimizing waiting time is prioritized over cost, one could choose a policy achieving around 0.2 hours, representing an improvement of 0.3 hours over the most balanced solution, 0.7 hours over the heuristic policy that always selects the next available resource, and a remarkable 8.7 hours improvement compared to the as-is status. Likewise, the Pareto front also includes solutions that outperform the *LC* heuristic in terms of cost, showing that our approach can offer both broad coverage and best-in-class solutions.

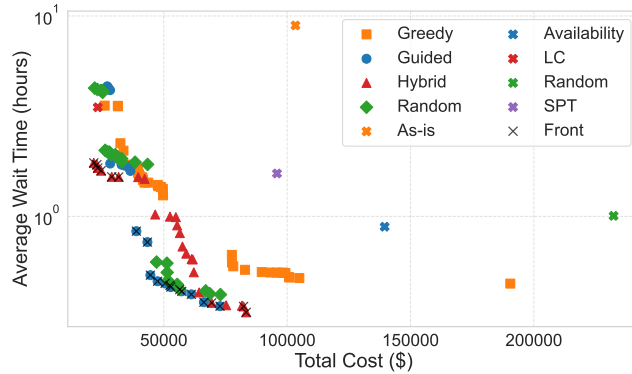


Fig. 3: Solution space and Pareto front for ACR event log.

6 Discussion and Conclusion

In this work, we presented a novel approach for optimizing resource-specific handover policies in business processes, enabling the discovery of more effective collaboration strategies. Building on a multi-agent business process simulation model where resources and their interactions are represented as autonomous agents, we integrated a genetic algorithm to optimize process performance across multiple dimensions. The approach is designed to be fully generic: it can incorporate any set of optimization objectives. Its output is a Pareto front of optimized handover policies, providing decision-makers with a transparent set of trade-offs. We validated the approach on synthetic and real logs with cost and time as objectives, achieving substantial improvements compared to the as-is process.

Threats to validity. Our approach relies on a simulation model to evaluate candidate handover policies, which introduces several validity threats. First, optimization outcomes depend on simulation fidelity: if the simulator does not accurately reproduce real process dynamics, estimated improvements may not fully transfer to reality. To mitigate this risk, we deliberately use a data-driven simulation approach that has been shown in prior work to achieve state-of-the-art accuracy [9]. Moreover, the simulation model itself is not part of our optimization method but serves as an interchangeable input artifact; therefore, improvements in simulation fidelity can directly benefit optimization outcomes without changes to the proposed approach. Nevertheless, future work should explicitly quantify simulator accuracy and analyze the sensitivity of optimization results to modeling errors. Second, policies are optimized and evaluated within the same model, which creates a risk of overfitting to simulator-specific dynamics. In practice, this could lead to policies that perform well in simulation but less so in reality. We partially mitigate this threat by grounding simulations in discovered models derived from real event logs and by validating the approach on multiple datasets. A promising direction for future research is robustness

analysis across alternative simulation models or perturbed model parameters to assess the stability of discovered policies. Third, our experiments augment event logs with hourly resource cost information that is typically not recorded in standard logs. While such augmentation is common in process optimization studies, it introduces assumptions that may affect absolute performance estimates. We partially mitigate this threat by using realistic cost assumptions that correlate with performance and by relying on the same assumptions for as-is and optimized policies. Future work could incorporate empirically observed cost data or uncertainty-aware optimization to reduce this dependency further. Furthermore, our approach assumes that resources can choose whom to collaborate with, or at least that handovers between resources are subject to change. While this assumption holds for many real-world processes—particularly knowledge-intensive processes in which resources have some degree of decision-making autonomy—it may be less applicable to highly orchestrated workflow processes. We refer to [9] for details on the types of processes benefiting from resource-specific modeling.

Future work. Beyond validity considerations, the approach entails higher computational effort due to evolutionary search. Although NSGA-II provides a strong baseline for multi-objective optimization, other algorithms may be preferable as the number of objectives grows. Reducing computational overhead—e.g., by narrowing the search space, identifying the most influential decision points, or optimizing only selected handovers—remains an important direction. Furthermore, extending the policy space towards richer collaboration strategies (e.g., dynamic parallelization or adaptive routing) is a promising avenue for future work. Finally, our evaluation relies on heuristic baselines for comparison. This is partly due to the lack of suitable existing benchmarks addressing the same problem of resource handover optimization. Future work may add stronger optimization-oriented resource allocation baselines that optimize a single objective but add further insights into the value of our approach.

Acknowledgments This work received support from VILLUM FONDEN (grant VIL57420) through the Center for Digital Compliance (DICE), the Innovation Fund Denmark projects “Explainable Hybrid-AI for Computational Law and Accurate Legal Chatbots” (XHAILe, grant 4355-00018B) and Predictive and Prescriptive Process Analytics for Industry 4.0 (P3AI4, grant 4105-00045B).

References

1. Bejarano, J., Barón, D., González-Rojas, O., Camargo, M.: Discovering optimal resource allocations for what-if scenarios using data-driven simulation. *Frontiers in Computer Science* **Volume 5** (2023)
2. Branchi, S., Buliga, A., Di Francescomarino, C., Ghidini, C., Meneghello, F., Ronzani, M.: Recommending the optimal policy by learning to act from temporal data. *arXiv preprint arXiv:2303.09209* (2023)
3. Chapela-Campa, D., Benčekroun, 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)

4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE TEVC* **6**(2), 182–197 (2002)
5. Diaz, J.F., López, H.A., Quesada, L., Rosero, J.C.: Pareto-optimal trace generation from declarative process models. In: *BPM Workshops*. pp. 314–325 (2024)
6. Djedović, A., Žunić, E., Avdagić, Z., Karabegović, A.: Optimization of business processes by automatic reallocation of resources using the genetic algorithm. In: *BIHTEL*. pp. 1–7 (2016)
7. Filatov, V.O., Yerokhin, M.A.: Improved multi-objective optimization in business process management using r-nsga-ii. *RADIO ELECTRON COMPU* (3) (2023)
8. Huang, Z., van der Aalst, W.M., Lu, X., Duan, H.: Reinforcement learning based resource allocation in business process management. *DKE* **70**(1), 127–145 (2011)
9. 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**, 4 (2025)
10. Lee, H., Kim, S.S.: Integration of process planning and scheduling using simulation based genetic algorithms. *IJAMT* **18**, 586–590 (2001)
11. Ling, S., Schmidt, H.: Time petri nets for workflow modelling and analysis. In: *SMC*. vol. 4, pp. 3039–3044. *IEEE* (2000)
12. López-Pintado, O., Dumas, M.: Business process simulation with differentiated resources: Does it make a difference? In: *BPM*. Springer (2022)
13. López-Pintado, O., Dumas, M., Berx, J.: Discovery, simulation, and optimization of business processes with differentiated resources. *Inf. Syst.* **120**, 102289 (2024)
14. 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)
15. Middelhuis, J., Bianco, R.L., Sherzer, E., Bukhsh, Z., Adan, I., Dijkman, R.M.: Learning policies for resource allocation in business processes. *Inf. Syst.* **128**, 102492 (2025)
16. Peters, S.P.F., Dijkman, R.M., Grefen, P.W.P.J.: Resource optimization in business processes. In: *EDOC*. pp. 104–113 (2021)
17. Si, Y.W., Chan, V.I., Dumas, M., Zhang, D.: A petri nets based generic genetic algorithm framework for resource optimization in business processes. *Simul Model Pract Theory* **86**, 72–101 (2018)
18. Sindhgatta, R., Ghose, A., Dam, H.K.: Context-aware analysis of past process executions to aid resource allocation decisions. In: *CAISE*. pp. 575–589 (2016)
19. Song, M., van der Aalst, W.M.P.: Towards comprehensive support for organizational mining. *Decis. Support Syst.* **46**(1), 300–317 (2008)
20. Srinivas, N., Deb, K.: Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation* **2**(3), 221–248 (1994)
21. Tour, A., Polyvyanny, A., Kalenkova, A.: Agent system mining: Vision, benefits, and challenges. In: *IEEE Access*. vol. 9, pp. 98336–98350 (2021)
22. Wibisono, A., Nisafani, A.S., Bae, H., Park, Y.J.: On-the-fly performance-aware human resource allocation in the business process management systems environment using naïve bayes. In: *AP-BPM*. pp. 70–80. Springer (2015)
23. Zbikowski, K., Ostapowicz, M., Gawrysiak, P.: Deep reinforcement learning for resource allocation in business processes. In: *ICPM*. pp. 177–189 (2022)
24. Zhao, W., Yang, L., Liu, H., Wu, R.: The optimization of resource allocation based on process mining. In: *ICIC*. pp. 341–353 (2015)
25. Zitzler, E., Thiele, L.: An evolutionary algorithm for multiobjective optimization: The strength pareto approach. *TIK report* **43** (1998)