COMPASS: A Process Mining-based Methodology for Prompt Analysis of Large Language Model Agents

Rene Dorsch[1,*], Daniel Henselmann[1] and Andreas Harth[1,2]

[1]Fraunhofer IIS, Nordostpark 84, 90411 Nuremberg, Germany
[2]Friedrich-Alexander-Universität Erlangen-Nürnberg, Lange Gasse 20, 90403 Nuremberg, Germany

Abstract
Exploring and optimizing the behavior of LLM agents in complex environments is challenging. We present COMPASS, a methodology that systematically applies Process Mining to discover, analyze, and refine agent behavior. COMPASS consists of five phases: planning the project, extracting data from the LLM agent, transforming data into event logs, exploring and analyzing agent behavior, and providing feedback through conformance checking or prompt design guidelines. We evaluate COMPASS in a case study with an LLM agent that uses three tools to generate SPARQL queries while exploring a complex knowledge graph. Through COMPASS, we identified ineffective behavior patterns and optimized prompts to improve performance. Our methodology supports data-driven prompt optimization with interpretable behavioral models, enhancing explainability and reliability in complex environments. This ongoing research aims to systematically improve agent performance through exploratory behavioral analysis that increases transparency in agents' decision-making processes.

Keywords
Large Language Model, Agent, Prompt Optimization, Process Mining, Explainability

1. Introduction

The development of autonomous agents is flourishing due to the flexibility, reasoning, and planning capabilities provided by Large Language Models (LLMs) [1]. An LLM-based agent, or short agent, refers to a system that iteratively reasons and obtains information from its environment based on the LLMs' capabilities [2] to consume and produce reasonable text. Despite these advantages, LLMs add variation and complexity to agent-based systems, which can lead to unexpected behavior [3].

Current LLM optimization approaches rely on either hand-crafted prompts [4], which often yield higher quality results but require prompt engineers to conduct time-consuming trial-and-error explorations based on their intuition and experience, or automated prompt generation methods [5], which leverage larger collections of successful examples but depend on the availability of well-performing prompts to learn from and the pattern recognition and specification capabilities of other LLMs. Consequently, both methods are brittle and prone to missing information in environments requiring multi-step reasoning and fail to analyze complex agent behavior patterns or decision-making inconsistencies.

The absence of systematic behavior analysis necessitates methodologies that can discover, analyze, and formalize agent behavior. Process Mining (PM) techniques [6] — designed to discover structured workflow models from data — offer a framework to address the limitations of current prompt optimization approaches (see Sec. 2). By applying PM to the agents' produced text, we can derive formal behavioral models representing agent decision processes, identify successful and problematic patterns, and develop targeted prompt guidelines addressing specific behavioral inconsistencies.

Thus, we explore the research question, "How can PM support the systematic optimization of LLM agent behavior?" To address the question, we introduce COMPASS — a COMprehensive Process AnalySiS

Janis Grabis, Yves Wautelet, Emanuele Laurenzi , Hans-Friedrich Witschel, Peter Haase, Marco Montali, Cristina Cabanillas, Andrea Marrella, Manuel Resinas, Karolin Winter. Selected Papers of HybridAIMS and CAI Workshops. Co-located with CAiSE 2025
[*]Corresponding author.
rene.dorsch@iis.fraunhofer.de (R. Dorsch); daniel.henselmann@iis.fraunhofer.de (D. Henselmann); andreas.harth@iis.fraunhofer.de (A. Harth)
0000-0001-6857-7314 (R. Dorsch); 0000-0001-6701-0287 (D. Henselmann); 0000-0002-0702-510X (A. Harth)

methodology for prompt engineering that applies PM techniques to discover and guide agent behavior. The COMPASS methodology encompasses five phases: (1) planning of the optimization project, (2) extraction and evaluation of event data from an agent, (3) transformation of event data into event logs, (4) exploration and analysis of process models to discover expected and unexpected sequential patterns, and (5) specification and selection of guidelines for the prompt refinement. Guidelines refer to textual descriptions (of parts) of process models, such as workflow models or decision points.

We illustrate the optimization problem of agents through Graf-von-Data in Sec. 3, an agent that explores complex knowledge graphs with specialized tools to generate SPARQL queries. After providing PM background in Sec. 2, we present COMPASS in Sec. 4 and demonstrate its application in Sec. 5, where we identified ineffective behaviors and developed prompt guidelines that reduced unexpected agent behavior. Sec. 6 examines COMPASS's benefits and limitations, while Sec. 7 summarizes results and outlines future work.

2. Background and Related Work

Process Mining (PM) is a set of techniques for discovering, monitoring, and enhancing processes based on event data (see Fig. 1). Event data (Fig. 1 A) — traces like KS, KSR, D, DR, S, SR, or END captured by information systems during process execution — undergoes preprocessing to create structured event logs. Event logs (Fig. 1 B) serve as primary input for most PM techniques, with each event containing at least a case identifier (uniquely identifying a process instance), an activity name (describing what happened), and a timestamp (establishing sequential relationships between events). Within PM three main techniques are used [6]: process discovery, conformance checking, and model enhancement. Process discovery automatically generates process models from event logs without requiring pre-existing models. For instance, in Fig. 1 C) the Petri net has been discovered by process discovery algorithms based on the event log in Fig. 1 B). The discovered process models capture workflows — the underlying decision logic and interaction patterns — that can be represented with different notations, such as Petri nets or state machines. Conformance checking (CC) compares process models with event logs to identify and quantify discrepancies. CC allows measurement of compliance with expected behaviors and regulations by highlighting where actual process executions deviate from predefined models. Fig. 1 D) illustrates CC by highlighting unexpected steps within the observed behavior. Model enhancement builds on existing process models by incorporating additional performance information such as processing times, waiting times, and resource utilization to help identify bottlenecks and inefficiencies.

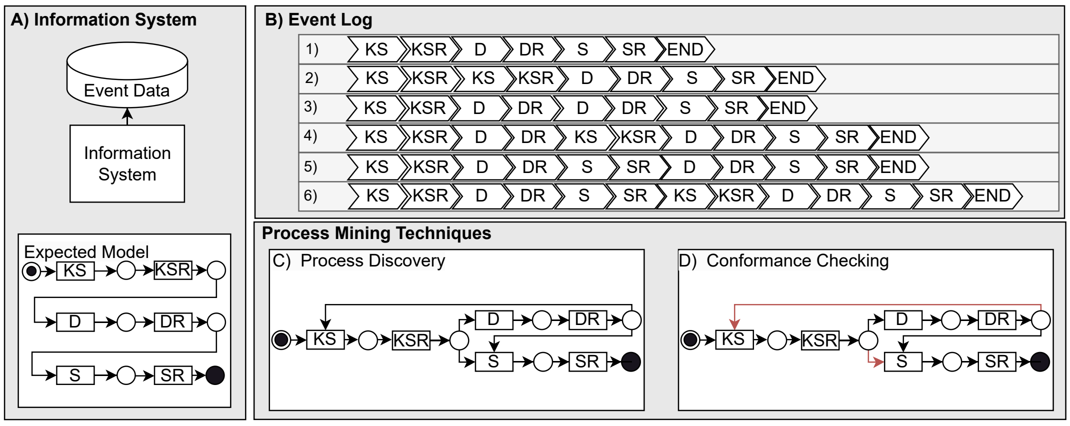
A) Information System
Event Data
Information System
Expected Model
KS
KSR
D
DR
S
SR
B) Event Log
1) KS KSR D DR S SR END
2) KS KSR KS KSR D DR S SR END
3) KS KSR D DR D DR S SR END
4) KS KSR D DR KS KSR D DR S SR END
5) KS KSR D DR S SR D DR S SR END
6) KS KSR D DR S SR KS KSR D DR S SR END
Process Mining Techniques
C) Process Discovery
D) Conformance Checking

Figure 1: Example showing an information system (A), event log with records from the information system (B), expected process model (C), and discovered model (D) of the information system

The PM² methodology [7] provides a structured framework for PM projects. PM² organizes projects into six concise phases: (1) planning the project scope and objectives, (2) extracting relevant event data, (3) processing data into structured event logs, (4) mining and analyzing process models, (5) evaluating results against hypotheses, and (6) implementing improvements with ongoing monitoring. This methodology emphasizes iterative refinement between phases as project understanding evolves — a principle we incorporate in COMPASS. COMPASS differentiates from automated optimization techniques [5, 8] (e.g., Few Shot Learning, COPRO, MIPRO) by shifting the focus of the LLM agents' textual output, a final document, to extracting events from the output to create an event log. Moreover, unlike manual methods [4] where experts specify high-level attributes such as intent, output format, style, or role, COMPASS focuses on deriving actionable guidelines to structure the agent's workflow through decision-making recommendations.

3. Motivating Example

To illustrate the challenges of optimizing LLM agent behavior, consider Graf-von-Data (GvD)[1], an agent that transforms natural language questions into SPARQL queries by exploring a supply chain knowledge graph[2] with 33,755 triples that models semiconductor organizations, their sites, and supply relations. GvD implements the ReAct framework and utilizes three specialized tools: (1) search, which identifies entity URIs based on natural language keywords; (2) describe, which extracts triples from specified entity URIs; and (3) query, which executes SPARQL queries on the knowledge graph and returns result sets. The agent was deployed at Chat AI [9] using Meta-Llama-3 70B v3.3 with a p-value of 1.0 and a temperature of 0 to minimize variation in responses. After each query generation task, GvD is instructed to perform a structured self-assessment, classifying the outcome as success or failure.

Despite using fixed prompts and LLM parameters, GvD produces inconsistent results when given the same input. For example, when repeatedly asked 'Customers of TSMC that offer design services?', the agent generates semantically different SPARQL queries across runs varying in structure, triple patterns, and entity relationships. GvD illustrates the problem that an LLM, even with constrained tools, introduces significant variability in an agent. Despite only having three tools (search, describe, and query), GvD demonstrated different ways to combine these actions, creating diverse behavioral patterns. Furthermore, exploring the trajectories to identify sources for inconsistency is time-consuming and complex. In this context, a trajectory refers to a sequential trace of the agent's interaction with its environment and may includes the following components: actions (the explicit operations performed by the agent such as describe or self-assessment), thought processes (the agent's internal thoughts, including step-by-step reasoning, planning decisions, and evaluation of intermediate results), and observations (external feedback from the environment or tools in response to the agent's actions, such as search results or error messages).

4. COMPASS Methodology

This section presents COMPASS, a methodology that leverages PM techniques to optimize agent behavior. COMPASS serves two primary purposes: Improve agent performance through behavioral information and verify compliance with specified behavior. Our approach assumes that an agent with an initial prompt is already available.

The COMPASS methodology has five phases (see Fig. 2). Each phase is specified with its inputs, outputs, and activities. The methodology begins with a planning stage, defining the optimization goals, specifying the agent, and establishing evaluation metrics. Following this initial phase, one or more optimization iterations follow. Each iteration involves four key activities: (1) extracting behavioral trajectories from agent interactions, (2) processing these trajectories into structured event logs, (3) exploring and mining these logs to discover process models and key behavioral patterns, and (4) specifying guidelines derived from these patterns. Throughout this process, each optimization iteration maintains a focused approach — targeting either performance enhancement or compliance verification.

[1]Available at: http://purl.org/compass/graf-von-data
[2]Available at: http://purl.org/compass/supplybench

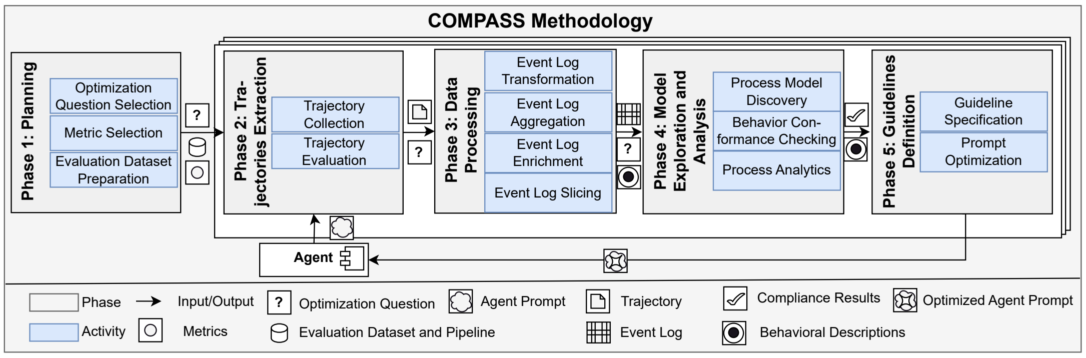

Figure 2: Overview of COMPASS methodology

4.1. Phase 1: Planning

Phase 1 focuses on planning the optimization project by establishing clear objectives and evaluation criteria. This phase takes an existing agent as input. It has three key outputs: focused optimization questions, specified metrics to measure progress, and a comprehensive evaluation dataset for comparing the optimized agent against its predecessors. The planning phase consists of three activities:

- **Optimization Question Formulation:** Questions are specified to guide the optimization process by identifying aspects that require improvement or verification.
- **Metric Selection:** Based on the optimization questions, appropriate evaluation metrics are selected. For performance improvement, typical measures include accuracy, recall, precision, or F1 score. For instance, the F1 score is particularly valuable for an agent like GvD as it measures functional correctness based on result set equivalence rather than syntactic similarity [10]. For compliance verification, specific constraints (e.g., ”Tool query must not be used before Tool search”) are defined and can be systematically evaluated.
- **Evaluation Dataset Preparation:** To contrast the behavior of agents, it is necessary to have a comprehensive dataset and an evaluation pipeline that covers diverse scenarios relevant to the optimization questions.

4.2. Phase 2: Trajectories Extraction

The extraction phase focuses on systematically gathering and evaluating agent behavior data by collecting trajectories. The extraction phase has two inputs: the agent with its current prompt configuration and the evaluation dataset from Phase 1. It produces evaluated trajectories that document the agents' behavior. The extraction phase consists of two activities:

- **Trajectory Collection:** This activity involves capturing detailed records of the agent's behavior as it processes inputs from the evaluation dataset. For the collection, telemetry frameworks like OpenTelemetry[3] or MLflow[4] can be integrated into the agent environment.
- **Trajectory Evaluation:** This activity requires applying the evaluation pipeline established in Phase 1 to analyze each collected trajectory, measuring it against the defined metrics for performance or compliance verification.

4.3. Phase 3: Data Processing

The data processing phase converts the trajectories from Phase 2 into event logs to facilitate the analysis of agent behavior with PM techniques. Phase 3 takes the trajectories from Phase 2 and the optimization questions from Phase 1 as inputs, producing event logs that capture the sequential patterns of the agent's interactions. The data processing phase consists of four activities:

[3]See https://opentelemetry.io/
[4]See https://mlflow.org/

- **Event Log Transformation:** In this step, the collected trajectories are transformed into an event log consisting of several cases. Each case consists of multiple events, where each event may represent an action, thought, or observation. An event contains at least three elements: a case identifier that uniquely identifies a trajectory for a specific example of the evaluation dataset, an activity descriptor that names the action, thought, or observation, and a timestamp that establishes sequential relationships between events. Furthermore, performance metrics for each case that differentiate successful from unsuccessful behaviors are included. The transformation may also incorporate additional attributes, such as the agent's internal reasoning process, to provide information for Phase 4.
- **Event Log Aggregation (optional):** The aggregation process reduces complexity by consolidating events and abstracting detailed sequences into higher-level patterns, making the resulting process models more interpretable [11].
- **Event Log Enrichment (optional):** During enrichment, the logs are augmented with additional information, either through analysis of existing elements from the trajectories, such as error classification or pattern identification, or through integration of external process knowledge [12].
- **Event Log Slicing (optional):** The slicing procedure segments the event logs to address the optimization questions, applying techniques such as Slice and Dice, variance-based, or compliance-based approaches to reveal different facets of agent behavior.

4.4. Phase 4: Model Exploration and Analysis

Within Phase 4, PM techniques are applied to transform event logs into partial and complete process models. This phase takes three key inputs: Event logs from Phase 3, the original optimization questions from Phase 1, and event logs derived from previous iterations. From these inputs, within Phase 4, two outputs are produced. The first output consists of identified behavioral patterns that reveal how the agent approaches different scenarios. The second output comprises compliance results that distinguish between expected behaviors that align with optimization goals and unexpected behaviors representing potential improvement opportunities. The activities within this phase may include:

- **Process Model Discovery:** Within this activity, process discovery algorithms [6] are applied to identify process models from event logs to reveal the agent's decision sequences, action patterns, and partial workflows. This can uncover whether an agent follows consistent patterns when approaching different tasks or exhibits variable behavior under similar conditions.
- **Behavior Conformance Checking:** This activity compares discovered process models against expected behaviors, detecting inconsistencies and deviations in the LLM agent's operation highlighting specific areas where prompt modifications might yield performance improvements. For instance, conformance checking might identify when agents deviate from optimal solution strategies or skip critical steps in complex reasoning tasks.
- **Process Analytics:** Leveraging supplementary analytical methods, including visual analytics and specialized data mining approaches, can be used to gain perspectives on the agents' behavior. These techniques help identify bottlenecks in the agent's decision process, compare performance across different input complexities, or visualize the distribution of behavioral variants across successful and unsuccessful outcomes.

4.5. Phase 5: Guidelines Definition

The guideline definition phase aims to create actionable prompt optimizations for the agent. This phase bridges behavioral analysis and implementation of the guidelines in the agent's prompt. Importantly, if the analysis reveals that the agent already performs adequately, no prompt modifications are necessary. This phase takes behavioral descriptions and compliance results from Phase 4 as its primary inputs. It has an optimized prompt as its output, incorporating only those changes needed to address identified behavioral issues. The main activities are:

- **Guideline Specification:** The specification process transforms behavioral patterns into clear guidelines for prompt modification. This includes prioritizing guidelines based on their potential impact on performance metrics or compliance objectives and validating each proposed guideline against the original optimization questions to ensure alignment with overall goals. A guideline might specify optimal tool usage sequences or define phase transitions between different cognitive operations like exploration and synthesis.
- **Prompt Optimization:** The final step integrates the selected and specified guidelines into the agent's prompt, drawing on established approaches for effective prompt engineering [4].

5. Case Study: Graf-von-Data

This section presents a comprehensive case study applying the COMPASS methodology to optimize GvD (Sec. 3). We employed during our analysis two software components: PM4Py [13] for Phase 2 to 3, and PMTK [14] for Phase 4. All materials used in this case study — including agent trajectories, evaluation questions and queries, knowledge graph, preprocessing pipeline, and prompts — are available[5].

Planning and Extraction: Based on the observed differences in Sec. 3, we established as our main objective to systematically understand the behavioral patterns of GvD and identify factors contributing to execution inconsistency (Phase 1). This objective led to two key optimization questions: (1) "Does the agent consistently follow expected behavioral patterns?" and (2) "What behavioral guidelines can reduce variability and improve F1 score performance?" For the extraction phase (Phase 2), we created an evaluation corpus and pipeline to collect the necessary trajectory data for analysis. Our evaluation corpus, based on the SupplyBench dataset, contains 69 questions spanning five domains within the semiconductor industry knowledge graph: organizational information, geolocations, financial data, employee profiles, and partnership networks. We organized the query according to complexity, ranging from simple single-hop patterns to four-hop patterns requiring more reasoning. To enable objective performance measurement, we paired each question with its reference SPARQL query, establishing a standard for evaluation. We implemented an evaluation pipeline that compares the result sets of generated SPARQL queries against reference queries. Rather than comparing query syntax, we adopted the result set equivalence for the F1 metrics proposed by Liu et al. [10], which measures functional query correctness. We executed each question 10 times to characterize the observed variation, generating 690 trajectories per optimization iteration.

Optimization 1) Conformance Assessment: In our first optimization iteration, we focused on discovering and documenting the actual behavioral patterns of GvD. We transformed the collected trajectories into event logs following the COMPASS methodology's data processing phase (Phase 3). Our event logs captured four essential elements for each trajectory event: (1) a case identifier uniquely referencing each trajectory, (2) an activity label describing the action (e.g. describe), (3) timestamps establishing sequential relationships between events, and (4) performance metrics differentiating successful from unsuccessful agent behaviors based on the self-assessment. We divided the event log into two subsets based on the agent's self-assessment results for a comparative analysis.

We explored the behavioral patterns of GvD using PMTK's process model visualization capabilities (Phase 4). The behavioral model of GvD is visualized in Figure 3. Our analysis revealed that in unsuccessful runs, the agent more often (14% instead of 9%) attempted to generate SPARQL queries immediately after receiving search results without gathering sufficient knowledge about the graph structure. This premature query generation triggered multiple guess-based query refinement cycles (switches between search and query), reducing overall efficiency and accuracy. Furthermore, we identified a system limitation that the prompt did not capture: the agent performs a maximum of 10 operation cycles per query task, a constraint that may influence GvD's approach to query generation.

Based on these findings, we implemented two guidelines in the GvD prompt (Phase 5): an explicit behavioral model that specifies allowed and restricted actions, especially during the initial exploration

[5]Available at: http://purl.org/compass/documentation

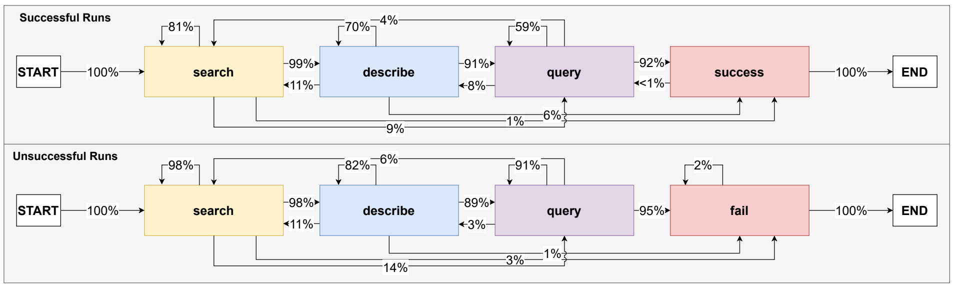

Figure 3: Comparison between successful and unsuccessful runs for the base prompt

cycles, and information about the 10-cycle limitation[6]. This model guides the LLM agent through the transition from search results to query generation and enforces more thorough knowledge graph exploration before query formulation.

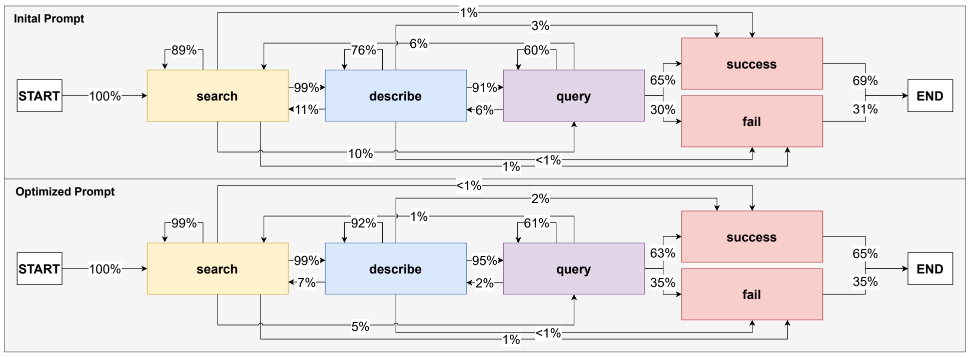

Figure 4: Comparison between the initial prompt and prompt with action restrictions

Within a second iteration of Optimization 1), we applied the new prompt to GvD on the same questions to generate a new corpus (Phase 2). We used the same data processing steps as before (Phase 3). We compared, within Phase 4, the initial prompt with the new prompt using PMTK's process models and identified significant differences in GvD's behavior (see Fig. 4). The premature query generation dropped from 10% to 5%, showing that our prompt adjustments directly influenced GvD's behavior. Furthermore, we investigated the variation between the initial and optimized prompt answers by comparing the standard deviations. We identified a reduction in standard deviation between the runs from 3.3% to 3.0%, with 24 of 69 questions having a lower standard deviation for the new prompt across 10 runs, indicating a decrease of variation through the added behavioral guidelines.

Optimization 2) Process Model Optimization: Our second optimization iteration focused on enhancing GvD's performance to address our second research question. We collected and evaluated trajectories from GvD with the revised prompt (Phase 2), then created an event log that incorporated F1 score as a performance metric in our analysis. To facilitate comparative analysis, we segmented the event log into four distinct subsets based on the agent's self-assessment results and an F1 threshold of 0.9 (Phase 3). We employed the variant explorer to compare these event logs and investigate sequential patterns (Phase 4). Our analysis revealed that variants in which GvD alternates between graph exploration (search and describe) and query generation (query) phases consistently achieved higher F1 scores. Based on this insight, we formulated a guideline emphasizing switching between graph exploration and query generation phases (Phase 5). We integrated this guideline into the prompt[7] and evaluated GvD's performance with our dataset.

Figure 5 shows the F1 scores across different prompts for GvD. Both prompt variations that use the process model-based optimizations improved F1 scores compared to the base prompt, indicating that

[6]Available at: http://purl.org/compass/action-restriction-prompt
[7]Available at: http://purl.org/compass/phase-transition-prompt

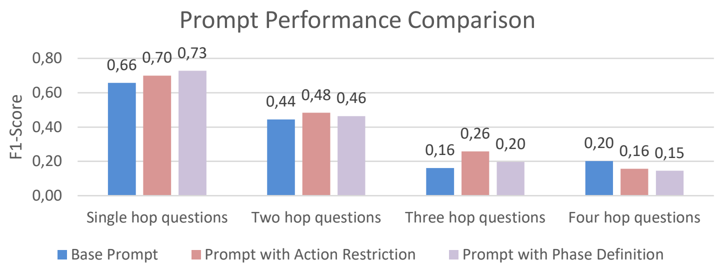

Figure 5: Performance of the process model based prompt optimization

guidelines describing expected behavior enhance performance. However, we observed that performance decreased for the optimized prompts with four triple patterns. We attribute this to GvD‘s fixed steps, which make execution more reliable but also more rigid.

6. Discussion

In this section, we first discuss the benefits and then examine the limitations of COMPASS. Process models discovered with COMPASS improve the explainability of agent behavior in several ways. First, they provide interpretable, high-level process models that serve as a starting point for exploring event logs in detail. Second, these models enable a systematic comparison between different agent versions, highlighting specific changes in behavioral patterns. Finally, the methodology generates comprehensive documentation throughout the optimization process. Performance improvements are also achieved through generalized guidelines using our PM-based approach. Guidelines such as action restrictions and phase definitions go beyond LLM-specific optimizations and represent transferable knowledge about effective agent behavior. These results validate the efficacy of the PM approach to behavior optimization, showing that data-driven guidelines can improve both explainability and performance.

Despite its benefits, COMPASS faces several key limitations. First, implementation requires dual expertise in PM and prompt engineering, creating adoption barriers for organizations lacking interdisciplinary teams. Second, patterns discovered are often domain-specific, limiting direct transferability between domains, such as knowledge graph exploration versus code generation. Third, effective PM demands data volume to identify significant patterns, which can be resource-intensive with LLMs.

7. Conclusion

We presented COMPASS, a methodology that applies Process Mining (PM) techniques to optimize LLM-based agent behaviors, addressing our research question: ”How can PM support the systematic optimization of LLM agent behavior?” Through five phases — planning, trajectory extraction, data processing, model exploration, and guideline definition — COMPASS enables systematic discovery and analysis of agent behavior patterns.

We demonstrated COMPASS’s effectiveness by applying it to GvD, an agent that generates SPARQL queries from natural language by exploring knowledge graphs. Through PM, we identified inefficient patterns such as premature querying without sufficient exploration, enabled comparison of different prompt versions through their discovered process models, and generated evidence-based guidelines that measurably improved both performance and consistency.

Our future research will extend the exploration of PM for agent optimization in three directions: (1) evaluating COMPASS across diverse agent architectures and environments to assess its generality, (2) adapting the methodology for multi-agent systems using object-centric PM techniques, and (3) developing an intuitive interface that improves visualization and analysis capabilities, making COMPASS more accessible to practitioners.

Acknowledgements.

The authors gratefully acknowledge the scientific support and HPC resources provided by Erlangen National High Performance Computing Center (NHR@FAU). This work was funded by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) through the Antrieb 4.0 project (Grant No. 13IK015B).

Declaration on Generative AI

While preparing this work, the author(s) used Claude and Grammarly to reword, and check grammar and spelling. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] X. Liu, et al., AgentBench: Evaluating LLMs as Agents, 2023. doi:10.48550/arXiv.2308.03688.
[2] S. Yao, et al., ReAct: Synergizing reasoning and acting in language models, in: International Conference on Learning Representations (ICLR), 2023. doi:10.48550/arXiv.2210.03629.
[3] A. Salinas, F. Morstatter, The Butterfly Effect of Altering Prompts, in: Findings of the Association for Computational Linguistics ACL 2024, Association for Computational Linguistics, Bangkok, Thailand, 2024, pp. 4629–4651. doi:10.18653/v1/2024.findings-acl.275.
[4] S. Schulhoff, et al., The Prompt Report: A Systematic Survey of Prompt Engineering Techniques, 2025. doi:10.48550/arXiv.2406.06608.
[5] K. Opsahl-Ong, et al., Optimizing Instructions and Demonstrations for Multi-Stage Language Model Programs, 2024. doi:10.48550/arXiv.2406.11695.
[6] W. Van Der Aalst, Process Mining: Data Science in Action, Springer Berlin Heidelberg, Berlin, Heidelberg, 2016. doi:10.1007/978-3-662-49851-4.
[7] M. L. van Eck, et al., PM^2: A Process Mining Project Methodology, in: Advanced Information Systems Engineering, Springer International Publishing, Cham, 2015. doi:10.1007/978-3-319-19069-3_19.
[8] O. Khattab, et al., DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines, 2023. doi:10.48550/arXiv.2310.03714.
[9] A. Doosthosseini, et al., Chat AI: A Seamless Slurm-Native Solution for HPC-Based Services, 2024. doi:10.48550/ARXIV.2407.00110.
[10] S. Liu, et al., SPINACH: SPARQL-Based Information Navigation for Challenging Real-World Questions, Association for Computational Linguistics, Miami, Florida, USA, 2024. doi:10.18653/v1/2024.findings-emnlp.938.
[11] V. Zelst, et al., Event abstraction in process mining: Literature review and taxonomy 6 (2021) 719–736. doi:10.1007/s41066-020-00226-2.
[12] P. Filipp, R. Dorsch, EVErPREP: Towards an Event Knowledge Graph enhanced Workflow Model for Event Log Preparation (2024). doi:10.1007/978-3-031-82225-4_3.
[13] A. Berti, et al., PM4Py: A process mining library for Python, Software Impacts 17 (2023) 100556. doi:10.1016/j.simpa.2023.100556.
[14] A. Berti, et al., The Process Mining ToolKit (PMTK): Enabling Advanced Process Mining in an Integrated Fashion (Extended Abstract) (2021).