Feasibility-Aware Agentic Reinforcement Learning for Web Tasks Under Joint Cost and Failure Constraints

Marco Bianchi *, Giulia Rossi , Alessandro Conti

Posted Date: 12 March 2026

doi: 10.20944/preprints202603.0977.v1

Keywords: feasibility analysis; constrained reinforcement learning; web agents; failure risk management; budget-aware decision models

 doi:10.20944/preprints202603.0977.v1

Article

Feasibility-Aware Agentic Reinforcement Learning for Web Tasks Under Joint Cost and Failure Constraints

Marco Bianchi, Giulia Rossi and Alessandro Conti *

Department of Electronics, Information and Bioengineering (DEIB), Politecnico di Milano, Via Ponzio 34/5, 20133 Milan, Italy

* Correspondence: a.conti@polimi.it

Abstract

Not all web tasks are feasible under strict cost and safety requirements, yet standard reinforcement learning implicitly assumes feasibility. This study introduces a feasibility-aware agentic reinforcement learning framestudy that explicitly reasons about whether a task can be completed within given cost budgets and failure risk limits. A feasibility estimator is trained to predict the probability that any valid action sequence exists under current constraints. The agent uses this signal to adapt its strategy, prioritize feasible subtasks, or terminate early when feasibility is low. Evaluation on 800–1,400 constrained web tasks demonstrates that feasibility-aware decision-making reduces wasted interactions, prevents high-risk attempts, and improves overall system reliability. This study reframes web automation as a constrained decision problem where recognizing infeasibility is as important as optimizing success.

Keywords: feasibility analysis; constrained reinforcement learning; web agents; failure risk management; budget-aware decision models

1. Introduction

Web automation has evolved rapidly from rule-based scripts to agentic systems capable of translating natural-language instructions into multi-step browser actions. Early systems relied primarily on handcrafted rules and predefined workflows, which limited their ability to generalize across diverse websites and dynamic user interfaces. Recent progress in large language models and agent frameworks has enabled web agents to reason over complex instructions, interact with web environments, and autonomously complete multi-stage tasks. In particular, recent studies have explored reinforcement-learning–based decision models for web agents operating under multi-cost and failure-risk constraints, highlighting the importance of balancing task success with operational limitations in realistic web environments [1]. Over the past five years, the development of new benchmarks and datasets has further advanced this field by providing more realistic evaluation settings that include heterogeneous websites, dynamic layouts, and long-horizon task workflows [2,3]. These resources have significantly improved the ability to assess agent performance beyond simple navigation tasks.

At the same time, evaluation environments have expanded from basic browsing scenarios to knowledge-worker and enterprise-level tasks that require complex reasoning, information retrieval, and multi-step interaction across web services [4,5]. Such environments reveal a substantial gap between the success rates reported in academic benchmarks and the level of robustness required for real-world deployment. In practical settings, web agents must operate under strict constraints related to interaction cost, computational resources, and acceptable failure risk. These constraints often determine whether a system can be deployed reliably in production environments [6]. Consequently, maximizing task success alone is insufficient; practical web automation must also ensure efficiency,

 doi:10.20944/preprints202603.0977.v1

predictability, and adherence to operational limits. Recent research has improved the grounding capabilities of web agents by linking language understanding to screen content and interface elements. Advances in multimodal perception and UI grounding enable agents to identify relevant components on the page, select appropriate actions, and adapt to common interface changes such as layout shifts or dynamic loading of content [7,8]. In parallel, agent frameworks integrating planning, tool usage, and iterative self-correction have enhanced multi-step reasoning and execution reliability in browser environments [9,10]. These approaches allow agents to decompose complex tasks into intermediate steps and recover from certain classes of execution errors. Despite these improvements, many current methods implicitly assume that the task is feasible within the given constraints. This assumption frequently breaks down in real web environments. In practice, web tasks may become infeasible due to factors beyond the agent's control [11]. Missing permissions, authentication barriers, paywalls, rate limits, unstable page layouts, and incomplete information can prevent successful task completion even when the instructions are clear. Under strict interaction budgets or safety constraints, continued exploration may only waste resources without increasing the likelihood of success. Reinforcement learning provides a natural framework for modeling sequential decision-making in such environments because it allows agents to learn policies over action sequences through interaction with the environment. Recent work has explored training web agents using demonstrations, offline interaction data, and iterative self-improvement techniques to improve generalization and reduce brittle behavior [12,13]. However, conventional reinforcement-learning objectives typically optimize expected return, which can encourage excessive exploration when success is uncertain. In web automation scenarios, such behavior is often undesirable because each interaction consumes time, tokens, or external API calls, and repeated trial-and-error increases the risk of triggering undesirable outcomes. Safe reinforcement learning introduces formal frameworks for learning under explicit constraints, including constrained Markov decision processes and risk-aware optimization objectives [14]. These methods aim to learn policies that maximize expected utility while satisfying predefined safety or cost limits. Research on risk-sensitive learning also emphasizes the importance of managing tail risk, as rare failures may have disproportionately large consequences in safety-critical applications [15]. Related developments in language-model alignment have demonstrated that constrained optimization can effectively balance task utility with safety costs during training rather than relying solely on post-hoc filtering mechanisms [16]. Nevertheless, most existing safe-RL approaches focus on identifying policies that satisfy constraints under the assumption that a feasible policy exists. In the context of web automation, this assumption may not hold because infeasibility often arises from external conditions that cannot be resolved through policy learning alone. Another limitation arises from current evaluation practices. Many web-agent benchmarks primarily report task success rates while providing limited information about interaction efficiency or safety-related behaviors [17,18]. As a result, an agent may appear successful even if it consumes excessive interaction budgets, enters repeated action loops, or approaches safety-violation boundaries. Safety-oriented evaluation suites demonstrate that agents can complete tasks while still producing actions that violate operational policies or expose users to risk [19]. Furthermore, experimental reports rarely present joint analyses of interaction cost, failure-risk exposure, and early-termination behavior. Without these metrics, it remains difficult to compare different approaches in terms of their suitability for real-world deployment under strict operational constraints. These observations motivate a reframing of constrained web automation. Instead of assuming that a feasible solution exists and focusing solely on policy optimization, web agents should also be capable of estimating whether a task is likely to be solvable under the current conditions and constraints. Feasibility recognition becomes a critical capability in environments where external restrictions and uncertainty are common. When the probability of feasibility is low, continued exploration may increase resource consumption and expose the system to unnecessary risk. Conversely, recognizing infeasibility early enables an agent to terminate execution gracefully or redirect effort toward feasible subtasks, improving reliability and resource efficiency.

 doi:10.20944/preprints202603.0977.v1

Motivated by these challenges, this study investigates feasibility-aware agentic reinforcement learning for web automation under joint interaction-cost and failure-risk constraints. The proposed framework introduces a feasibility estimator that predicts whether a valid action sequence is likely to exist given the current environment state and constraint limits. This signal guides the agent's decision-making process by prioritizing promising subtasks, avoiding high-risk exploration, and enabling early termination when the likelihood of satisfying constraints becomes low. The approach is evaluated on large-scale constrained web-task benchmarks comprising hundreds to over a thousand tasks with realistic operational limits. Evaluation metrics emphasize deployability, including reductions in wasted interactions, improved adherence to risk constraints, and more stable behavior under strict budgets. By treating infeasibility detection as a normal outcome rather than an exceptional case, the proposed framework aims to improve the robustness and reliability of web agents operating in real-world environments where constraints are unavoidable.

2. Materials and Methods

2.1. Sample and Task Set Description

Experiments used a constrained web-task benchmark built from real browser sessions and standardized instructions. The dataset contained 1,200 tasks, including 800 tasks for model development and ablation studies and 400 tasks for final testing. Tasks covered information lookup, sandboxed form filling, multi-step navigation, and checkout-like flows that were simulated without real payments. Each task included a natural-language goal, an initial browser state, and a constraint profile with (i) an action-step budget and (ii) a failure-risk limit defined over unsafe or irreversible operations. Task sampling targeted diversity in page layouts and difficulty. For each run, logs recorded success status, stop reason, steps used, and safety-related events.

2.2. Experimental Design and Control Conditions

A controlled comparison was used to test the effect of feasibility-aware decision making under joint budget and risk limits. The proposed method combined a base web agent with a learned feasibility estimator and a feasibility-gated policy. Three baselines were included. The first baseline used the same base agent but without feasibility estimation and trained it to maximize success with a budget penalty. The second baseline used constrained RL with an explicit safety cost and updated a penalty coefficient during training, but it did not estimate feasibility. The third baseline used a rule-based early-stop strategy that ended an episode when the remaining budget fell below a fixed threshold. All methods were evaluated on the same tasks and constraint settings. Each task was executed up to five times with different random seeds to reflect stochastic web behavior.

2.3. Measurement Procedures and Quality Control

Reported outcomes included success rate, mean number of actions, violation rate of the failure constraint, and a reliability score that reflects both completion and safety compliance. A safety event was recorded when an action entered a predefined unsafe class, such as submitting irreversible forms, visiting restricted pages, or triggering rate limits, based on environment rules and action traces. Quality control relied on full trajectory logging, including DOM snapshots, action parameters, and terminal signals. Pages with unstable rendering were filtered out using a screening step that required consistent loading in three warm-up trials. Log files were validated with schema checks and hash verification to prevent missing fields and corruption. Metrics were computed only from runs that passed these checks, and sensitivity tests confirmed that results were not dominated by a small set of atypical tasks.

2.4. Data Processing and Model Formulation

 doi:10.20944/preprints202603.0977.v1

Trajectories were converted into step-level records containing state embeddings, action types, rewards, budget use, and safety indicators. The feasibility estimator was trained as a binary probabilistic model that outputs the chance that at least one valid action sequence exists under the remaining budget and risk limit. Labels were obtained from bounded rollouts and verified successful traces: a state–constraint pair was marked feasible when a constraint-compliant continuation was observed within a fixed horizon. The agent optimized a constrained objective that trades off task reward and failure-risk cost through a penalty term:

$$\max_{\pi} \; E[\sum_{t=0}^{T} \gamma^t r_t] - \lambda \, E[\sum_{t=0}^{T} \gamma^t c_t],$$

Where r_t is the step reward and c_t is the failure-risk cost. Feasibility gating used the estimator output $\hat{p}_t$ to control stopping:

$$\text{terminate at t if } \hat{p}_t < \tau \text{ or } B_t \leq 0,$$

where τ is a feasibility threshold and B_t is the remaining action budget. Hyperparameters were selected on the development set by searching over λ and τ, and results were reported on the held-out test set.

2.5. Implementation Details and Reproducibility

The web agent used a transformer encoder for page text and UI structure, followed by an action head that selected an operation type and its parameters. The feasibility estimator shared the encoder backbone to keep state representations consistent and was trained with cross-entropy loss using class-balanced sampling to reduce label imbalance. Training followed an offline-to-online schedule, with offline pretraining on logged trajectories and constrained on-policy fine-tuning in the interactive environment. All runs used fixed seeds per split, identical constraint schedules, and the same episode horizon, and key runtime settings were recorded. Results are reported with 95% confidence intervals from bootstrap resampling over tasks. Statistical comparisons used paired permutation tests on per-task differences to avoid distributional assumptions.

3. Results and Discussion

3.1. Constraint-Compliant Completion Under Joint Limits

On the held-out test split (400 tasks; five runs per task), the feasibility-aware agent showed a higher constraint-compliant completion rate than all baselines, with the largest gains on tasks that required long action chains under tight step budgets. Fewer episodes ended in late budget exhaustion, which indicates that the policy avoided extended recovery attempts when the remaining budget was no longer sufficient for a valid solution. Similar failure patterns have been reported in realistic web-agent evaluations, where dynamic interfaces and partial observability make late-stage recovery costly and often ineffective [20,21]. For context on visually grounded interaction and explicit marking of clickable elements, Fig.1 is cited as an external reference:Fig.1. Set-of-Marks augmented webpage screenshot with interactable elements labeled.

 doi:10.20944/preprints202603.0977.v1

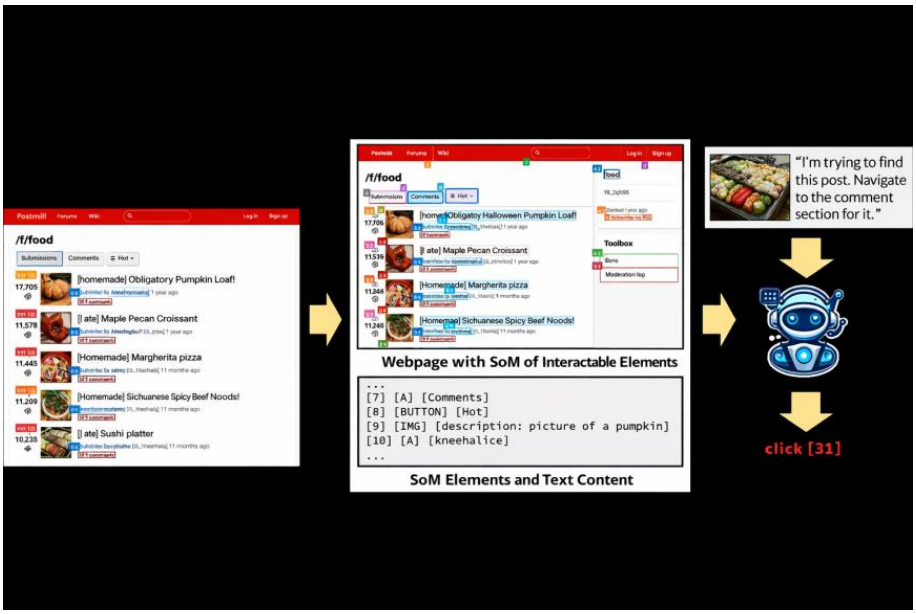

Figure 1. Set-of-Marks augmented webpage view highlighting interactable elements for visually grounded web-agent control.

3.2. Action Efficiency and Reduction of Wasted Steps

Feasibility-aware control reduced wasted steps, reflected by fewer actions in failed episodes and fewer repeated loops such as re-clicking unresponsive elements or re-issuing equivalent inputs after small UI changes. Under the same budget, baseline agents often continued exploring low-yield branches near the end of an episode, which increased cost without improving the chance of meeting constraints. In contrast, feasibility gating shifted effort earlier toward higher-probability subtasks, such as stabilizing navigation anchors, narrowing the target field, or consolidating required page transitions before committing to risky operations. This pattern complements prior work showing that planning and reflection can improve web agents, while also indicating that detecting low-feasibility states is a distinct capability that directly affects cost control [22,23].

3.3. Failure-Risk Control and Safety Behavior

Under strict failure-risk limits, the feasibility-aware agent reduced the violation rate relative to both a penalty-based constrained-RL baseline and a rule-based early-stop strategy. The difference mainly came from fewer high-risk actions during late recovery, where baseline methods tend to attempt irreversible clicks or repeated submissions after most budget has already been spent. This outcome is consistent with safe-RL findings that constraint satisfaction becomes harder when rare events dominate risk and when the environment includes irreversible transitions [24,25]. In this setting, feasibility estimation worked as a guardrail: when the state and remaining constraints were unlikely to support a valid continuation, the policy avoided escalation and instead terminated or shifted to a safer partial outcome.

3.4. Ablations, Calibration Effects, and Relation to Web Datasets

Ablation results showed that both feasibility prediction and feasibility-gated control were needed for stable gains. Removing feasibility gating preserved part of the step reduction but weakened safety compliance, while removing feasibility prediction made stopping behavior resemble a fixed rule that either stopped too late or quit too early. Results also depended on feasibility calibration: poorly calibrated scores increased premature stopping on hard-but-solvable tasks, whereas better calibration improved separation between infeasible states and recoverable states. This point aligns with dataset-driven web research, where diverse sites and interaction styles produce wide variation in solvability across states. As an external illustration of multi-step web trajectories,

 doi:10.20944/preprints202603.0977.v1

Fig.2 is cited as follows: Fig.2. Example data instance showing task context, webpage snapshots, and action transitions.

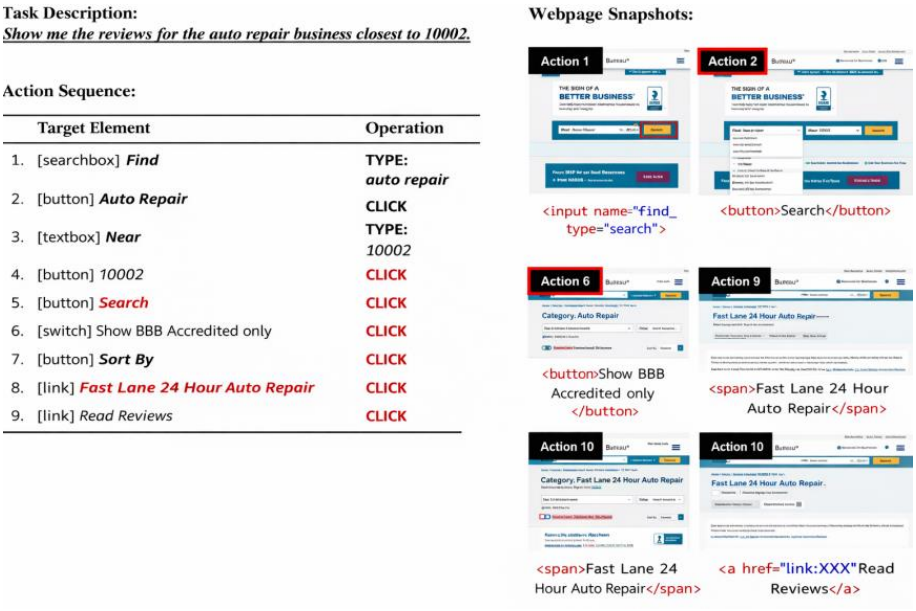

Figure 2. Example Mind2Web trajectory instance showing task context, webpage snapshots, and action transitions across steps.

4. Conclusion

Feasibility-aware agentic reinforcement learning strengthened constrained web automation by treating infeasibility as a normal outcome under joint limits on action budget and failure risk. In large-scale tests with strict constraints, combining feasibility prediction with feasibility-gated control reduced unnecessary steps, lowered violation rates, and increased constraint-compliant completion, which indicates more stable behavior when solvability depends on the remaining budget and risk margin. The main contribution is a decision framework that separates “hard but solvable” states from states that are unlikely to admit any valid continuation, showing that feasibility recognition can be as important as policy optimization for reliable web execution. This approach is relevant to customer-service workflows, enterprise operations, and regulated online processes where each interaction has a real cost and rare unsafe actions are unacceptable. Limitations remain in feasibility supervision and generalization: labels derived from bounded rollouts can be imperfect, and estimator calibration may degrade when websites, permissions, or hidden state change, which can cause early termination on difficult tasks. Further work should improve calibration and uncertainty handling, expand evaluation to longer horizons and broader sites, and develop more robust feasibility signals that remain reliable under web drift.

References

1. Ma, Q., Yue, L., Xu, S., Shi, Y., & Liu, H. (2026). Web Agent Agentic Reinforcement Learning Decision Model Under Multi-Cost and Failure Risk Constraints.
2. Goel, A., Chitale, P. A., Paliwal, B., Santra, B., & Sharma, A. HORIZON: A Benchmark for In-the-wild User Behaviour Modeling. In NeurIPS 2025 Workshop on Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling.
3. Gu, X., Yang, J., & Liu, M. (2025). Research on a Green Money Laundering Identification Framework and Risk Monitoring Mechanism Integrating Artificial Intelligence and Environmental Governance Data.
4. George, A. S., Baskar, T., Srikaanth, P. B., & Karthikeyan, M. (2025). The english paradigm: Natural language programming as the future of software development. Partners Universal multidisciplinary Research Journal, 52.

 doi:10.20944/preprints202603.0977.v1

5. Qiu, Y., & Wang, J. (2023, October). A machine learning approach to credit card customer segmentation for economic stability. In Proceedings of the 4th International Conference on Economic Management and Big Data Applications, ICEMBDA (pp. 27-29).
6. Ghelani, H. (2024). AI-driven quality control in PCB manufacturing: Enhancing production efficiency and precision. Valley International Journal Digital Library, 12(10), 1549-1564.
7. Zhu, W., Yao, Y., & Yang, J. (2025). Real-Time Risk Control Effects of Digital Compliance Dashboards: An Empirical Study Across Multiple Enterprises Using Process Mining, Anomaly Detection, and Interrupt Time Series.
8. Dritsas, E., Trigka, M., Troussas, C., & Mylonas, P. (2025). Multimodal interaction, interfaces, and communication: a survey. Multimodal Technologies and Interaction, 9(1), 6.
9. Gullí, A. (2025). Reasoning Techniques. In Agentic Design Patterns: A Hands-On Guide to Building Intelligent Systems (pp. 241-263). Cham: Springer Nature Switzerland.
10. Li, T., Xia, J., Liu, S., & Hong, E. (2025). Strategic Human Resource Leadership in Global Biopharmaceutical Enterprises: Integrating HR Analytics and Cross-Cultural.
11. Kampik, T., Mansour, A., Boissier, O., Kirrane, S., Padget, J., Payne, T. R., ... & Zimmermann, A. (2022). Governance of autonomous agents on the web: Challenges and opportunities. ACM Transactions on Internet Technology, 22(4), 1-31.
12. Mao, Y., Ma, X., & Li, J. (2025). Research on Web System Anomaly Detection and Intelligent Operations Based on Log Modeling and Self-Supervised Learning.
13. Patel, A., Hofmarcher, M., Leoveanu-Condrei, C., Dinu, M. C., Callison-Burch, C., & Hochreiter, S. (2024). Large language models can self-improve at web agent tasks. arXiv preprint arXiv:2405.20309.
14. Li, T., Xia, J., Liu, S., & Jiang, Y. (2025). Digital Transformation of Human Resources: From Consulting Frameworks to AI-Enabled Learning Management Systems.
15. Eziokwu, U. J., Duruemeruo, U. C., Akande, N. A., & Makinde, O. F. (2023). Cost-Aware and Risk-Sensitive Learning for Autonomous Robots A Conceptual Framework.
16. Gu, X., Liu, M., & Yang, J. (2025). Application and Effectiveness Evaluation of Federated Learning Methods in Anti-Money Laundering Collaborative Modeling Across Inter-Institutional Transaction Networks.
17. Kuntz, T., Duzan, A., Zhao, H., Croce, F., Kolter, Z., Flammarion, N., & Andriushchenko, M. (2025). Os-harm: A benchmark for measuring safety of computer use agents. arXiv preprint arXiv:2506.14866.
18. Zhu, W., Yang, J., & Yao, Y. (2025, October). How Compliance Maturity Translates to Risk Reduction: A Multi-Case Comparison of Global Operations Using fsQCA and Hierarchical Bayesian Methods. In Proceedings of the 2025 2nd International Conference on Digital Economy and Computer Science (pp. 672-676).
19. Adabara, I., Sadiq, B. O., Shuaibu, A. N., Danjuma, Y. I., & Maninti, V. (2025). Trustworthy agentic AI systems: a cross-layer review of architectures, threat models, and governance strategies for real-world deployment. F1000Research, 14(905), 905.
20. Cai, B., Bai, W., Lu, Y., & Lu, K. (2024, June). Fuzz like a Pro: Using Auditor Knowledge to Detect Financial Vulnerabilities in Smart Contracts. In 2024 International Conference on Meta Computing (ICMC) (pp. 230-240). IEEE.
21. Liu, S., Feng, H., & Liu, X. (2025). A Study on the Mechanism of Generative Design Tools' Impact on Visual Language Reconstruction: An Interactive Analysis of Semantic Mapping and User Cognition. Authorea Preprints.
22. Assaf, G., & Assaad, R. H. (2025). A decision-support system for choosing between traditional and alternative project delivery methods for bundled projects. Engineering, construction and architectural management, 32(11), 7181-7216.
23. Du, Y. (2025). Research on Deep Learning Models for Forecasting Cross-Border Trade Demand Driven by Multi-Source Time-Series Data. Journal of Science, Innovation & Social Impact, 1(2), 63-70.
24. Ghosh, R. (2025). Safe Reinforcement Learning for Multi-Agent Systems with Risk Constraints.
25. Mao, Y., Ma, X., & Li, J. (2025). Research on API Security Gateway and Data Access Control Model for Multi-Tenant Full-Stack Systems.

 doi:10.20944/preprints202603.0977.v1