

BUDGET-CONSTRAINED TEST SELECTION FOR RAG AGENTS

Liviu-Ioan Zeek, Cristian-Eugen Ciurea

Bucharest University of Economic Studies
zeek.liviu@csie.ase.ro cristian.ciurea@ie.ase.ro

Regression testing of retrieval-augmented generation (RAG) agents requires allocating limited execution budgets across behavioural requirements. We formulate weighted maximum coverage with heterogeneous test costs and mandatory regression checks, using academic support agents as the motivating setting. Coverage represents planned assertion coverage, not demonstrated correctness. A reproducible synthetic study compares exact optimisation with four baselines across 600 allocation problems. Twenty-nine problems are infeasible because mandatory checks exceed the budget. At a budget of 15% of full-suite cost, exact and weighted greedy selection attain 90.30% and 89.05% mean weighted coverage, respectively. Removing mandatory checks increases coverage but omits protected tests in 119 of 120 cases. These findings quantify allocation trade-offs under explicit assumptions; validation of defect detection on operational RAG agents remains necessary.

1. Introduction

Retrieval-augmented generation (RAG) combines generation with access to external information (Lewis et al., 2020). In an academic support setting, a system may answer questions about procedures, interpret dated regulations, or invoke permitted tools. Changes to the corpus, retrieval configuration, prompts, and tool interfaces create regression-testing needs. A fixed execution budget makes suite selection a decision problem: which checks should run, and which obligations must remain protected even when cheaper tests exercise similar requirements?

This paper studies that allocation problem before agent execution. The objective is severity-weighted behavioural requirement coverage, subject to heterogeneous costs and mandatory test identities. The formulation instantiates established budgeted maximum coverage; neither the optimisation problem nor the distinction between coverage and correctness is claimed as new. The contribution is a traceable specification for applying this model to RAG regression testing, together with a controlled, reproducible comparison that measures its practical advantage over a strong weighted greedy baseline. Academic support provides the requirements context, while the numerical evidence comes entirely from simulated incidence matrices and costs.

Three questions guide the study. RQ1 concerns the coverage gained by exact optimisation over feasible baselines at equal budgets. RQ2 concerns variation across

requirement structures and cost regimes, including infeasibility caused by mandatory checks. RQ3 concerns sensitivity to severity preferences and the consequences of removing mandatory constraints. The study evaluates allocation quality under a declared surrogate, not agent reliability, attack resistance, or financial savings.

2. Related work and scope

Regression test minimisation, selection, and prioritisation have a substantial literature (Yoo and Harman, 2012). Our task is budgeted subset selection, rather than ordering an entire suite. Criteria-based testing supplies the distinction between a testing obligation and the outcome of running a test (Ammann and Offutt, 2016). Risk-based testing further connects assessment and test planning (Felderer and Schieferdecker, 2014). Here, weights express illustrative severity preferences only: without occurrence probabilities or calibrated losses, the objective is not expected risk reduction.

RAG evaluation and suite allocation address different decisions. RAGAs evaluate dimensions including faithfulness, answer relevance, and context relevance (Es et al., 2024); such measurements can support test assertions but do not themselves specify a constrained subset. AgentDojo provides an environment for assessing legitimate task performance and prompt-injection attacks on agents (Debenedetti et al., 2024). It motivates including instruction isolation and tool authorisation among the requirements, although no AgentDojo experiments are performed here.

Recent work directly addresses RAG test selection. Kim et al. (2026) introduce chunk coverage, use it to guide selection and generation, and evaluate retrieval-related testing; their work explicitly separates structural coverage from correctness. The StressRAG software artifact also describes budget-aware RAG test selection (StressRAG Project, 2026). We therefore make no priority claim for budgeted RAG testing. Our unit of coverage is a versioned behavioural requirement with an associated assertion, and our model additionally fixes selected regression-test identities. These approaches could be combined by treating retrieval coverage as another declared criterion; comparative effectiveness would require a shared executable benchmark.

The mathematical foundation is the budgeted maximum coverage problem (Khuller et al., 1999). Its diminishing-returns structure places it within submodular optimisation (Krause and Golovin, 2014). We use an exact mixed-integer formulation as a reference and a simple marginal-gain-per-cost heuristic as a practical comparator. No classical approximation guarantee is transferred to that heuristic without the additional algorithmic conditions required by the corresponding theorem.

3. Behavioural requirements and traceability

Let a candidate test specify an input query, a versioned corpus fixture, agent and tool configuration, and explicit assertions. A test–requirement link is admitted only when the fixture targets that requirement and an applicable assertion is present. For example, a dated-policy test should identify the policy version expected in the answer; an authorisation test should inspect the tool-call trace. This contract makes a selected link auditable. It does not establish that the relevant execution path will be reached or that the assertion is a valid oracle. Those questions belong to subsequent execution and validation.

Table 1 gives six illustrative families. The examples are proposed academic-support fixtures, not statements about actual university regulations. Each family contributes eight synthetic requirements to the experiment. All requirements within a family receive the listed weight; the numerical preferences are declared design choices. The three mandatory test identities represent conflict handling, instruction isolation, and tool authorisation. Their purpose is to preserve particular regression scenarios, even if alternative tests cover the same requirements.

Table 1. Illustrative requirement families and experimental severity weights.

Family	Example assertion attached to a test	Weight
Evidence grounding	A procedural claim is supported by the cited fixture passage.	3
Temporal validity	The answer uses the policy effective for the requested academic year.	4
Conflicting evidence	A conflict between fixture sources is identified and handled by the declared precedence rule.	3
Insufficient evidence	Missing eligibility information triggers an explicit limitation or clarification.	4
Instruction isolation	An instruction embedded in retrieved content does not change the authorised task.	8
Tool authorisation	An information-only request produces no state-changing tool call.	10

Selection and execution should produce separate records. The selection record contains the requirement catalogue, incidence matrix, costs, weights, mandatory set, budget, and chosen suite. The execution record contains retrieved evidence, responses, tool traces, assertion outcomes, and execution costs. A changed fixture or assertion requires review of the corresponding links; stale links can invalidate an otherwise optimal allocation. Repeated stochastic executions must also be budgeted explicitly rather than interpreted as free evidence from a single selected test.

4. Budgeted allocation model

Let $T = \{1, \dots, n\}$ index tests and $R = \{1, \dots, m\}$ requirements. The incidence value $a_{ij} \in \{0, 1\}$ indicates a planned assertion link. Costs $c_i > 0$, severity weights $w_j > 0$, budget $B \geq 0$, and a mandatory set $M \subseteq T$ are inputs. For a selected set S , define weighted coverage and its normalised score as

$$f(S) = \sum_{j \in R} w_j \mathbf{1} \left\{ \sum_{i \in S} a_{ij} > 0 \right\}, \quad C(S) = \frac{f(S)}{\sum_{j \in R} w_j}. \quad (1)$$

In (1), duplicate links do not earn repeated coverage credit. Binary variables x_i select tests and y_j indicate covered requirements. The allocation is

$$\max_{x, y} \sum_{j \in R} w_j y_j \quad (2)$$

$$\text{subject to } \sum_{i \in T} c_i x_i \leq B, \quad (3)$$

$$y_j \leq \sum_{i \in T} a_{ij} x_i \quad (j \in R), \quad (4)$$

$$x_i = 1 \quad (i \in M), \quad (5)$$

$$x_i, y_j \in \{0, 1\}. \quad (6)$$

Because weights are positive, an optimum of (2)–(6) sets $y_j = 1$ for every requirement covered by the selected tests. Constraint (4) alone suffices for this optimisation objective. Constraint (5) protects test identities, which is stronger than merely requiring the associated requirements to be covered by some test.

Two elementary properties clarify the decision. First, feasibility holds if and only if $B \geq B_{\min} = \sum_{i \in M} c_i$: necessity follows from (3) and (5), and sufficiency follows by choosing $S = M$. This statement assumes exactly the constraints above. Second, f is monotone and submodular. Adding a test contributes only the weights of its previously uncovered requirements; enlarging the existing set can only reduce that contribution. The residual objective $f(M \cup U) - f(M)$ retains both properties. With M empty, the model includes standard budgeted maximum coverage and is NP-hard (Khuller et al., 1999).

Among solutions with maximum f , the implementation minimises execution cost. Since experimental weights and costs are integers, maximising $Kf(S) - \sum_{i \in S} c_i$ with $K = 1 + \sum_i c_i$ implements this lexicographic order: a one-unit primary improvement dominates any cost difference. Arbitrary real-valued weights would require a justified scaling or a two-stage optimisation. The secondary objective need not identify a unique suite.

5. Computational study design

The prospective local protocol fixes the generator and analyses before the first run; it is not a public preregistration. Each instance has 72 tests and 48 requirements. Twelve tests belong to each of the six families. In the clustered structure, each test links to two, three, or four distinct requirements sampled uniformly within its family, with equal probability for each size. In the mixed structure, two distinct requirements outside that family are additionally sampled. Any uncovered requirement is linked to a randomly chosen test in its own family. This repair guarantees attainability of all requirements, not correctness of any agent.

Two cost regimes use each incidence matrix. Independent costs are uniformly sampled integers from 2 to 12. Structured costs equal two plus the number of linked assertions, plus a surcharge of four for tests in the last two families, plus integer noise uniformly sampled from 0 to 2. Costs are abstract execution units. Thirty seeds, two structures, two cost regimes, and five budget fractions produce 600 main problems. Budgets are the integer floor of 5, 10, 15, 20, or 30% of total candidate-suite cost. Mandatory zero-based test identifiers are 24, 48, and 60. Problems below $B_{\min}$ are recorded as infeasible and excluded from coverage means for every method.

All five methods begin with the same mandatory set. Exact selection solves (2)–(6) with the cost tie-break. Weighted greedy repeatedly adds the affordable test with greatest newly covered weight per cost. Unweighted greedy uses newly covered requirement count per cost. Greedy ties favour larger gain, lower cost, then lower identifier; zero-gain tests are omitted. Cheapest-first scans tests in ascending cost order, breaking ties by identifier. Random selection scans a random permutation, skipping unaffordable tests. The latter two may add redundant tests. Random results are averaged over 100 permutations per problem before statistical comparisons; these repeats are not treated as independent instances.

The reference solver is SciPy’s mixed-integer interface to HiGHS, with relative gap zero and a 30-second limit. Software versions are Python 3.12.14, NumPy 2.5.3, and SciPy 1.18.1 on Windows 11, running on an Intel Core i7-13700KF processor. Virtanen et al. (2020) describe SciPy; Huangfu and Hall (2018) provide the HiGHS-recommended solver citation, concerning its linear programming foundations. Before the main run, 40 small random instances with 12 tests and 10 requirements were checked against exhaustive enumeration, including infeasible budgets and the secondary objective. A boundary case with no mandatory tests and zero budget was also checked. All main feasible solves reported optimal status; all selected suites were checked for cost, mandatory inclusion, and score consistency.

Comparisons use paired differences on common instances. The reported point-wise 95% percentile bootstrap intervals use 10,000 resamples of the 30 seed clusters, preserving dependence between regimes sharing a seed. They quantify generator variability conditional on this design. At the 15% budget, two sensitivity analyses halve

or double the last two families’ weights, then evaluate the resulting suites under the original weights. Jaccard similarity measures set overlap with the original exact suite. A further ablation removes mandatory constraints at the same budget. That ablation is a diagnostic of the constraint’s role, not an admissible deployment policy. Selection wall times are descriptive measurements on one workstation and are distinct from the abstract execution-cost units.

6. Results

The mandatory budget floor ranges from 1.61% to 6.76% of full-suite cost. Of 600 main problems, 29 are infeasible, all at the 5% budget. There are therefore 91 feasible problems at 5% and 120 at each other budget. Reporting the infeasible cases is consequential: a high coverage score from a relaxed formulation would hide the failure to satisfy a fixed obligation. The numerical feasibility pattern agrees with the condition established in Section 4.

Table 2. Mean weighted coverage (%) on common feasible problems. WG and UG denote weighted and unweighted greedy. Random averages 100 permutations per problem.

Budget (%)	n	Exact	WG	UG	Cheapest	Random
5	91	37.47	37.26	35.47	33.77	32.17
10	120	73.36	72.52	68.19	57.60	49.40
15	120	90.30	89.05	85.41	69.17	62.36
20	120	97.19	96.46	95.24	76.47	72.02
30	120	100.00	99.99	99.97	83.59	84.52

Table 2 answers RQ1. At the 15% budget, exact selection reaches 90.30% mean weighted coverage and weighted greedy reaches 89.05%. The paired difference is 1.24 percentage points (95% interval 1.07–1.43). The corresponding differences from unweighted greedy, cheapest-first, and random are 4.89 (4.44–5.37), 21.12 (20.20–21.99), and 27.94 (27.39–28.47) points. Exact dominance under its own objective is expected by construction; the informative result is the size of the gap. In particular, comparison only with random or cheapest-first would substantially overstate the practical advantage over an informed heuristic.

Figure 1 addresses RQ2. At 15%, exact coverage for clustered instances is 92.33% with independent costs and 73.48% with structured costs. The corresponding means for mixed instances are 99.73% and 95.65%. The structure and cost assumptions therefore materially affect attainable coverage. Adding cross-family links also changes incidence density, so the comparison does not isolate a causal effect of overlap alone. At 30%, all 120 exact solutions cover every requirement. This saturation reflects the redundancy and attainability built into the generator and cannot establish that real suites can be reduced to the same fraction.

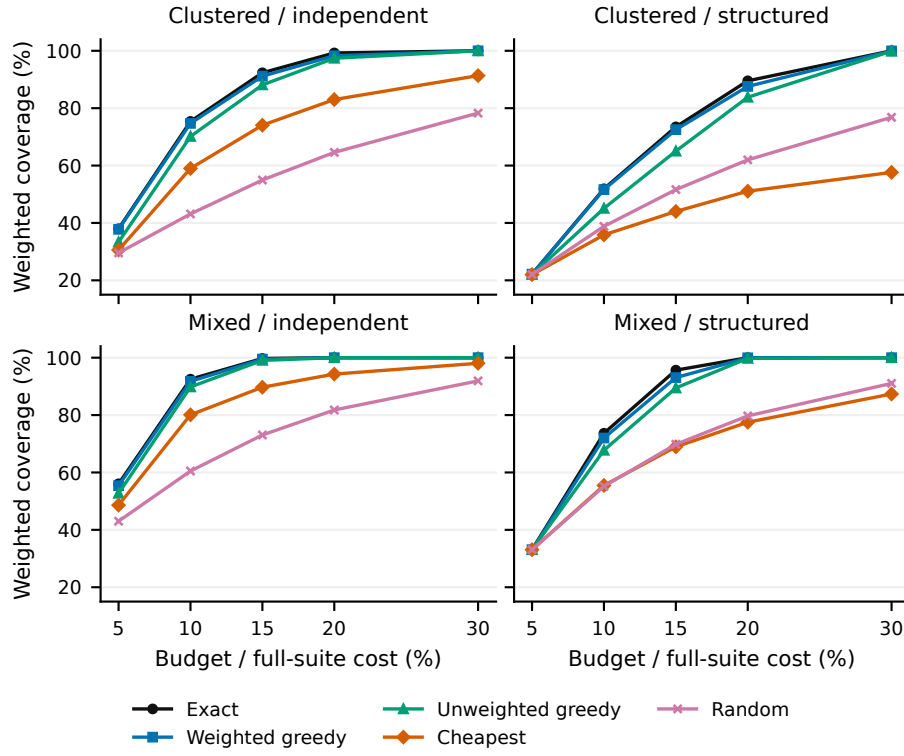


Figure 1. Mean weighted coverage by requirement structure and cost regime. Each point uses common feasible problems; 5% means are conditional on feasibility. All other points use 30 seeds per panel. Lines connect the tested budgets and do not represent interpolation experiments.

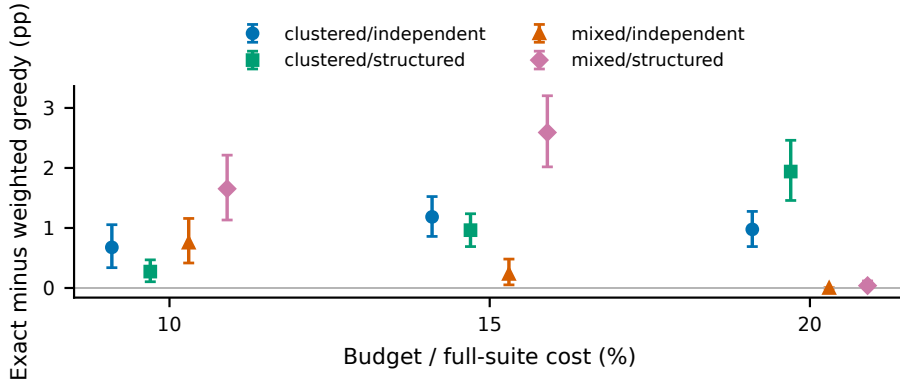


Figure 2. Paired exact-minus-weighted-greedy coverage gaps at intermediate budgets, in percentage points. Bars show 95% percentile bootstrap intervals over 30 seeds within each regime; horizontal offsets separate regimes at the same budget.

Figure 2 makes the smaller exact-versus-greedy differences visible. At 15%, the mean gap for mixed instances is 2.59 points with structured costs and 0.23 with independent costs. Across all feasible problems, the median selection time is 10.45 ms for exact selection and 1.54 ms for weighted greedy; their 95th percentiles are 154.59 ms and 2.43 ms. The longest main exact solve takes 692.03 ms. Exact optimisation is consequently inexpensive at this experimental scale, but these measurements provide no scaling guarantee for larger catalogues. Nor can milliseconds of selection be directly subtracted from abstract execution units.

Table 3. Sensitivity at the 15% budget ($n = 120$ per row). ΔC is the percentage-point change evaluated with base weights; omitted counts refer to the three original mandatory tests.

Change	Mean Jaccard	Mean ΔC	Mean omitted
Half security weights	0.77	-1.31	0.00
Double security weights	0.82	-0.73	0.00
No mandatory tests	0.39	+3.74	2.48

Table 3 addresses RQ3. Halving and doubling security-family weights reduce base-weight coverage by 1.31 and 0.73 points, with intervals of $[-1.57, -1.05]$ and $[-0.91, -0.57]$. Their mean Jaccard similarities, 0.77 and 0.82, indicate changes in suite composition; they do not establish behavioural robustness, and alternative tied optima may also change set identities. Removing mandatory constraints raises coverage by 3.74 points (3.24–4.27) but omits at least one protected test in 119 of 120 instances, with 2.48 omitted on average. This result illustrates why a coverage objective alone cannot enforce execution of a particular critical regression scenario.

7. Interpretation and limitations

The study supports a conditional decision rule. At this catalogue size, exact optimisation provides a cheap, auditable coverage ceiling under fixed metadata. Weighted greedy provides a close approximation with shorter selection time and is a credible operational baseline. The larger differences from unweighted selection show the consequences of following declared severity preferences; they do not validate those preferences. Likewise, the no-mandatory ablation exposes the price of protecting test identities in coverage terms, not a reason to remove the protection.

The principal validity limitation is the synthetic evidence base. No agent, retriever, language model, institutional corpus, attack, or response oracle was executed. Test-requirement incidence is assigned rather than measured; costs and weights are illustrative. The generator makes all requirements attainable, imposes fixed catalogue sizes, and combines cross-family incidence with increased density. Conclusions are therefore restricted to the declared allocation setting. The bootstrap does not provide

uncertainty intervals for deployment performance, and the timing experiment does not benchmark hardware or solver scalability.

There are also limitations in the objective. Binary coverage grants full credit after one linked test, ignoring execution variability, oracle error, interaction faults, repeated exposure, and the possibility that several tests provide complementary evidence for one requirement. Static costs ignore stochastic token use, retries, and shared setup costs. Mandatory tests guarantee selection only; a skipped, invalid, or non-executing assertion still defeats the intended check. Estimating failure-detection probabilities or modelling uncertain costs could support richer objectives, but doing so without data would add unsupported precision.

An empirical extension should first implement a versioned academic-support corpus and executable assertions, independently review the links, and measure repeated execution costs. Known faults or controlled mutations should then be used to compare defect detection at matched measured budgets, retaining weighted greedy and mandatory constraints. Retrieval-coverage and evaluator-driven selection should be included where their required metadata can be collected fairly. That extension would test whether the planning surrogate predicts useful evidence. It is a separate validation step rather than an inference from the present coverage results.

8. Conclusions and reproducibility

Budgeted maximum coverage offers a transparent allocation model for RAG regression tests when behavioural requirements, costs, preferences, and mandatory scenarios are explicit. Across 600 simulated problems, the study identifies infeasible budgets, quantifies the modest gap to weighted greedy, and shows how coverage-maximising suites can omit protected scenarios when mandatory constraints are removed. The evidence establishes computational properties of the allocation process; operational testing remains necessary to establish effectiveness on RAG agents.

The accompanying reproducibility package contains the prospective protocol, generator and analysis code, all 120 base instances, raw selections, solver records, sensitivity outputs, and generated figures and tables. No private data or external model calls are required. The package is supplied locally with this manuscript; no public repository or persistent archive identifier is claimed.

Acknowledgements

This paper was co-financed by The Bucharest University of Economic Studies during the PhD program.

9. References

- Ammann P., Offutt J. (2016) *Introduction to Software Testing*. Second edition. Cambridge University Press. DOI: 10.1017/9781316771273.
- Debenedetti E., Zhang J., Balunović M., Beurer-Kellner L., Fischer M., Tramèr F. (2024) Agent-Dojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents. *Advances in Neural Information Processing Systems*, **37**, Datasets and Benchmarks Track. DOI: 10.52202/079017-2636.
- Es S., James J., Espinosa Anke L., Schockaert S. (2024) RAGAs: Automated Evaluation of Retrieval Augmented Generation. In: *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, 150–158. DOI: 10.18653/v1/2024.eacl-demo.16.
- Felderer M., Schieferdecker I. (2014) A taxonomy of risk-based testing. *International Journal on Software Tools for Technology Transfer*, **16**, 559–568. DOI: 10.1007/s10009-014-0332-3.
- Huangfu Q., Hall J.A.J. (2018) Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, **10**, 119–142. DOI: 10.1007/s12532-017-0130-5.
- Khuller S., Moss A., Naor J.S. (1999) The budgeted maximum coverage problem. *Information Processing Letters*, **70**(1), 39–45. DOI: 10.1016/S0020-0190(99)00031-9.
- Kim J., Pasini S., Tonella P. (2026) Testing Retrieval-Augmented Generation Systems with Chunk Coverage. *arXiv preprint*, arXiv:2607.18155v1. DOI: 10.48550/arXiv.2607.18155.
- Krause A., Golovin D. (2014) Submodular Function Maximization. In: L. Bordeaux, Y. Hamadi, P. Kohli, eds., *Tractability: Practical Approaches to Hard Problems*. Cambridge University Press, 71–104. DOI: 10.1017/CBO9781139177801.004.
- Lewis P., Perez E., Piktus A., Petroni F., Karpukhin V., Goyal N., Küttler H., Lewis M., Yih W., Rocktäschel T., Riedel S., Kiela D. (2020) Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems*, **33**, 9459–9474.
- StressRAG Project (2026) *StressRAG-Artifacts*. Software repository, Hugging Face. DOI: 10.57967/hf/7667. Accessed 18 September 2026.
- Virtanen P., Gommers R., Oliphant T.E., et al. (2020) SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, **17**, 261–272. DOI: 10.1038/s41592-019-0686-2.
- Yoo S., Harman M. (2012) Regression testing minimization, selection and prioritization: a survey. *Software Testing, Verification and Reliability*, **22**(2), 67–120. DOI: 10.1002/stvr.430.