Supplementary Material to the Paper "The Energy Impact of Domain Model Design Choices in Classical Planning: An Empirical Analysis"

Anonymous Author(s)

ACM Reference Format:
Anonymous Author(s). 2025. Supplementary Material to the Paper "The Energy Impact of Domain Model Design Choices in Classical Planning: An Empirical Analysis". In . ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Research Methodology

This appendix provides a detailed account of the research methodology briefly summarised in Section **??**. The methodological structure follows the principles of the PLANERGYM framework for energy measurement in AI planning systems [3]. In this framework, the first stage involves defining the objects and goals of measurement. The *measured objects* in our study are classical planners, while the *primary goal of the measurement* is to analyse how domain model design choices influence the energy consumption of these planners.

The subsequent stage involves defining measurement metrics. The study records several key metrics, including execution duration and energy consumption at the CPU package level, as well as CPU and memory utilisation at the system level.

Further elements of the PLANERGYM framework include the operationalisation of measured objects, the specification of the experimental workload, the design of measurement scenarios, and data evaluation methods. Accordingly, the following subsections detail the selection and preparation of classical planners, the selection of benchmark domains and generation of domain variants, the measurement collection procedure, and the data analysis approach.

1.1 Planner Selection

The AI planning field encompasses a wide range of planners developed over the past few decades, differing in their planning strategies, search techniques, support for planning languages, and other aspects. These differences must be taken into account when selecting planners for comparative analysis. To identify suitable planners for this study, we examined multiple sources, including IPCs and publicly available repositories.

Planner selection was guided by the following criteria: (i) availability of documentation, (ii) actively maintained open-source implementation with publicly accessible source code, (iii) support for PDDL, (iv) focus on classical planning, and (v) recent development to reflect the current state of the art. Applying these criteria narrowed the search to planners from the Agile Track of the IPC 2023 Classical Track, the most recent edition of the competition, which featured 23 planners.

To keep our experimental design tractable, given the factorial combination of planners × domains × configuration variants × repetitions, we adopted a framework-stratified sampling strategy. We selected a balanced set that spans two dominant open-source frameworks in contemporary classical planning (Fast Downward (FD) [4] and LAPKT[1]) and captures the diversity of search strategies within those frameworks. As a result, the final selection comprises five planners. Three of the selected planners are based on the FD framework, while the remaining two build upon the LAPKT framework. The FD-based planners include Fast Downward Stone Soup Agile (FDSSA) [1], Cerberus Agile (CA) [5], and DALAI Agile (DALAIA) [2]. The LAPKT-based planners are Approximate Novelty Search Tarski (ANST) [8] and Forward Backward Novelty Search (FBNS) [7].

1.2 Benchmark Selection and Variant Generation

The empirical investigation of how domain model design choices affect energy consumption requires a carefully constructed experimental workload. This workload must balance two competing demands: sufficient diversity to enable generalisation of findings across different planning tasks, and sufficient control to isolate the effects of specific modelling choices from confounding domain-specific characteristics. Domain selection establishes the foundation for this workload, while systematic variant generation operationalises the Domain Model Configuration Framework to produce the controlled variations necessary for analysis.

We select domains following the IPC guidelines, which stipulate that benchmarks should pose meaningful challenges for planners while enabling the identification of performance differences between them [11]. Domains must be neither too difficult so that most planners time out, nor too trivial so that performance variations vanish. To operationalise these principles and ensure reproducibility, our domain selection was guided by the following criteria: (i) a domain has appeared in an IPC edition and is encoded in PDDL 1.2, (ii) domain definitions and instances (or an instance generator) are publicly available, (iii) prior benchmark analyses or studies provide insight into domain structure, modelling practices, or planner behaviour, especially in relation to the design choice categories considered here, and (iv) instances exhibit a feasible range of difficulties, giving a meaningful mix of solved and unsolved runs across planners within the time limit.

[1] http://www.lapkt.org/

Conference'17, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Applying these criteria resulted in five classical benchmark domains that span 25 years of IPC history and represent diverse structural characteristics, problem complexities, and documented modelling sensitivities. As such, they provide a solid foundation for investigating the relationship between domain model design choices and energy consumption. Each domain is paired with ten selected problem instances. This number provides a sufficient variation in problem difficulty while keeping the experimental workload tractable, given its combinatorial expansion.

- Gripper (IPC 1998). This domain is relatively simple, but enables controlled investigation of specific modelling choices, with documented relevance to modelling redundancy [9]. We sampled problem instances by selecting every second instance from the original IPC set of twenty.
- Blocks World (IPC 2000). This domain is foundational in planning research with extensive prior use []. We sampled problem instances by selecting every tenth instance from the benchmark set, starting from the eleventh instance.
- Barman (IPC 2014). This domain has been documented to be sensitive to syntactic variations [10] and has undergone prior analyses of modelling redundancy [9]. We sampled problem instances from the IPC benchmark set by selecting every second instance from the original set of twenty, giving ten instances.
- Thoughtful (IPC 2014). This domain is a contrasting case with relatively low sensitivity to syntactic variation [10]. We obtained ten problem instances by taking every second instance from the original set of twenty.
- Ricochet Robots (IPC 2023). This domain appeared in the most recent IPC edition and exhibits moderate difficulty in Agile Track evaluations. As no predefined benchmark instances were publicly available, we generated ten problem instances using the domain's generator with default parameter settings.

Having established the baseline benchmark domains, we applied the Domain Model Configuration Framework (Section **??**) to generate controlled variants representing alternative design choices. For each baseline domain, the framework's transformation principles were applied to produce variants across the three design choice categories. Each generated variant was assigned a systematic identifier encoding its configuration parameters: the prefix ORG designates the original baseline domain, while SSC-, MRC-, and TDC- prefixes denote Syntactic Structure, Modeling Redundancy, and Task Design choices, respectively, followed by specific mechanism codes (see Table **??**). All variants were validated for conformance with the framework's validity constraints and PDDL syntax.

The generation process yielded 32 domain model variants per baseline domain (1 ORG + 20 SSC-* + 7 MRC-* + 4 TDC-* variants), resulting in a total of 160 domain configurations. Combined with ten problem instances per domain and five selected planners, this produces a factorial experimental design of 8,000 planner-domain-instance-configuration combinations, each executed with multiple measurement repetitions as described in Section **??**. All generated variants and instances are included in the experimental artifact to support reproducibility.

1.3 Measurement Collection Procedure

To ensure consistency, data validity, and reproducibility, we adopt a measurement collection procedure aligned with the PLANERGYM framework [3]. The procedure consists of two phases: a baseline phase and a planner phase. The baseline phase establishes a reference for baseline energy consumption by measuring system energy usage during an idle state with no user-level processes over a 300-second interval. The baseline phase is conducted once at the beginning of the measurement study. The planner phase follows a systematic execution protocol that captures energy consumption data across all measurement runs.

A measurement run corresponds to a complete execution of a planner on one domain model variant and its associated set of problem instances. The measurement scenario follows a standard usage pattern in automated planning: a planner is invoked once per problem instance (one-shot invocation [3]). The measurement window spans from planner invocation to either successful plan generation or timeout, whichever occurs first. A per-instance timeout of five minutes was established to strike a balance between coverage and computational feasibility. This timeout also aligns with the runtime constraint of the IPC Agile Track. To establish statistical robustness, each planner-domain-variant-instance combination is measured across 30 repetitions, as suggested in the PLANERGYM framework.

The execution protocol governing each measurement run proceeds through five phases: initialisation, stabilisation, measurement, post-measurement stabilisation, and logging. During the *initialisation phase*, the measurement tool validates planner availability, loads the domain definition and problem instance files, and clears any residual state from previous runs to ensure a clean start. The *stabilisation phase* introduces a brief idle period (5 seconds) to allow system processes to settle and CPU utilisation to reach baseline levels, thereby reducing measurement noise from background activity. If system activity remains elevated beyond a predefined threshold of 2%, the stabilisation phase is repeated up to twelve times. When this threshold cannot be met, the measurement is still recorded for completeness but flagged for potential exclusion. During the *measurement phase*, the planner is invoked on the current problem instance while energy consumption is captured at the processor level. Other aforementioned metrics are also recorded throughout the planning process. If execution exceeds the timeout, the measurement is terminated, and the timeout event is logged. The *post-measurement stabilisation phase* introduces another 5-second idle period to return the system to baseline conditions before the next measurement. Finally, during the *logging phase*, all recorded metrics are written to CSV files using a standardised schema.

Not all measurement runs produce valid data suitable for analysis. Invalid runs are systematically identified and excluded based on explicit criteria: (i) failure to achieve system stabilisation within the permitted retry limit, indicating unstable baseline conditions; (ii) system crashes or execution errors, compromising measurement integrity; or (iii) logging failures prevented complete data capture.

1.4 Data Analysis

The collected measurement data undergo processing to ensure validity, comparability, and interpretability. Initially, all data are filtered to remove invalid or incomplete runs as defined by the

measurement collection procedure. Subsequent analyses use only validated runs.

For each valid run, baseline correction is applied to isolate planner-specific energy consumption from background system overhead. The baseline energy contribution is calculated by scaling the baseline power draw to the planner execution duration. Net energz consumption is obtained by: $E_{\text{net}} = E_{\text{total}} - (P_{\text{baseline}} \times T_{\text{planner}})$, where E_{total} is the total energy measured during a run, P_{baseline} is the average baseline power draw, and T_{planner} is the measured planner execution time.

To characterise energy consumption patterns, several analytical metrics are computed for each planner-domain-variant combination. Mean energy (μ) and standard deviation (σ) are calculated across all problem instances and repetitions, capturing central tendency and variability of energy consumption and runtime.

For each planner–domain–variant combination, we construct an energy profile, defined as the vector of mean net energy consumptions across all problem instances. The Pearson correlation coefficient (r) is then computed between the energy profiles of the original domain model and its variants, capturing the similarity in consumption patterns. A high positive correlation indicates that a variant preserves the energy consumption structure of the original domain, while lower or negative correlations suggest that modelling choices alter energy profiles through problem-specific interactions.

The analysis strategy supports comparison across planners, domains, and design choice categories. Summary tables report the mean and standard deviation of energy consumption, the mean runtime, and the Pearson correlation coefficient for each planner-domain combination. Boxplots are included selectively to illustrate representative or notable effects, visualising energy consumption distributions across the ten problem instances, depicting median, quartiles, and range.

1.5 Experimentation Tooling

We implemented the measurement collection procedure and data analysis in a Python-based tool with two modules. The Baseline Measurement Module executes the baseline phase to estimate idle energy consumption. The Planner Measurement Module orchestrates the planner phase, implementing the five-stage execution protocol via a command-line interface that accepts planner, domain, and configuration parameters. It then autonomously performs measurement without further interaction.

Energy consumption is captured using pyRAPL [6], a Python library that interfaces with Intel's Running Average Power Limit (RAPL) interface, providing hardware-level energy readings at the CPU package level with a microsecond temporal resolution. Complementary system metrics (CPU utilisation, memory consumption) are recorded via psutil to contextualize energy data.

To ensure execution isolation and compatibility, each planner executes within a dedicated containerised environment. Following IPC practices, we employ Apptainer[2] (formerly Singularity) to encapsulate planner runtime environments. Containers were constructed from IPC definitions or adapted to ensure compatibility with measurement infrastructure, thereby guaranteeing identical software and dependency conditions across planners.

[2]https://apptainer.org/

Measurement results are stored in CSV format with standardised schema that includes planner identifiers, domain variants, instance identifiers, timestamps, execution durations, energy values, and system metrics.

The data analysis pipeline, implemented as a separate Python module, automates the workflow from raw logs to visualisations: data validation and filtering, baseline correction for net energy computation, descriptive statistics and correlation coefficient calculation, and comparative visualisations organized by design choice category.

The complete experimental artifact (measurement tool source code, planner container definitions, benchmark domains with generated variants, measurement data, and analysis scripts) is publicly available.

References

[1] Clemens Büchner, Remo Christen, Augusto B. Correa, Salome Eriksson, Patrick Ferber, Jendrik Seipp, and Silvan Sievers. 2023. Fast Downward Stone Soup 2023. In *International Planning Competition: Planner Abstracts–Agile Track of the Classical Track.*

[2] Clemens Büchner, Remo Christen, and Thomas Keller. 2023. DALAI – Disjunctive Action Landmarks All In. In *International Planning Competition: Planner Abstracts–Agile Track of the Classical Track.*

[3] Ilche Georgievski. 2025. What Is Hiding in the Energy Footprint of AI Planning? Initiating Energy Accountability. In *Workshop on Green-Aware Artificial Intelligence at the European Conference on Artificial Intelligence.* To appear.

[4] Malte Helmert. 2006. The fast downward planning system. *J. Artif. Int. Res.* 26, 1 (2006), 191–246.

[5] Michael Katz. 2018. Cerberus: Red-black heuristic for planning tasks with conditional effects meets novelty heuristic and enchanced mutex detection. In *IPC 2018 – Classical Tracks.* 47–51.

[6] Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K. Nurminen, and Zhonghong Ou. 2018. RAPL in Action: Experiences in Using RAPL for Power Measurements. *ACM Trans. Model. Perform. Eval. Comput. Syst.* 3, 2, Article 9 (2018).

[7] Anubhav Singh, Chao Lei, Nir Lipovetzky, Miquel Ramirez, and Javier Segovia-Aguas. 2023. Forward Backward Novelty Search. In *International Planning Competition: Planner Abstracts–Agile Track of the Classical Track.*

[8] Anubhav Singh, Nir Lipovetzky, Miquel Ramirez, and Javier Segovia-Aguas. 2023. Approximate Novelty Search. In *International Planning Competition: Planner Abstracts–Agile Track of the Classical Track.*

[9] Mauro Vallati and Lukáš Chrpa. 2019. On the Robustness of Domain-Independent Planning Engines: The Impact of Poorly-Engineered Knowledge. In *International Conference on Knowledge Capture.* Association for Computing Machinery, 197–204.

[10] Mauro Vallati, Lukáš Chrpa, Thomas Leo McCluskey, and Frank Hutter. 2021. On the Importance of Domain Model Configuration for Automated Planning Engines. *J. Autom. Reason.* 65, 6 (2021), 727ßß773.

[11] Mauro Vallati and Tiago Vaquero. 2015. Towards a Protocol for Benchmark Selection in IPC. In *Workshop on the International Planning Competition.*