Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

Elinilson Louras Santos Vital

# Plataforma de avaliação automática de planejadores

Trabalho de Conclusão de Curso

São Paulo
2024

ELINILSON LOURAS SANTOS VITAL

# PLATAFORMA DE AVALIAÇÃO AUTOMÁTICA DE PLANEJADORES

Trabalho de Conclusão de Curso apresentado ao Bacharelado em Ciência da Computação da Universidade de São Paulo, como requisito para a obtenção do título de Bacharel.

Orientador: Prof. Dr. Leliane de Nunes Barros
Instituto de Matemática e Estatística - USP

Coorientador: MSc. Viviane Boanadia dos Santos
Instituto de Matemática e Estatística - USP

SÃO PAULO
2024

# RESUMO

VITAL, Elinilson. Plataforma de avaliação automática de planejadores. 2024. 52 f. Trabalho de Conclusão de Curso  Bacharelado em Ciência da Computação, Universidade de São Paulo. São Paulo, 2024.

Avaliar empiricamente sistemas de planejamento automatizado contribui para a evolução e melhoria da qualidade dos planejadores no contexto da inteligência artificial. Este trabalho aborda o desenvolvimento de uma plataforma com o objetivo de automatizar a avaliação de planejadores, permitindo aos usuários agendar, executar e analisar experimentos de maneira sistemática e reprodutível. A necessidade desse tipo de ferramenta surge da crescente complexidade dos domínios de planejamento e da importância de comparações consistentes e abrangentes entre diferentes abordagens para resolver problemas de planejamento. A plataforma foi projetada para centralizar a gestão de domínios e problemas predefinidos, assegurando uniformidade no processo de avaliação. Os resultados obtidos com a plataforma destacam sua eficiência em comparar planejadores distintos, aplicando métricas personalizadas aos mesmos domínios e condições controladas de acesso a recursos. Foram gerados gráficos comparativos que apresentam o tempo total de resolução de problemas, o tempo de busca e a cobertura normalizada em função do tempo, permitindo uma análise detalhada do desempenho de cada planejador; também foram disponibilizadas tabelas de dados para facilitar a incorporação dos resultados em publicações científicas. Essa capacidade de análise comparativa contribui para a evolução dos planejadores ao identificar pontos fortes e fracos, promovendo avanços nas abordagens utilizadas. Apesar dos resultados positivos, algumas limitações foram encontradas durante o desenvolvimento do sistema. A integração com planejadores externos demandou esforços adicionais devido à ausência de padrões unificados para entrada e saída de dados, e a configuração de múltiplos contêineres para assegurar isolamento e escalabilidade aumentou a complexidade técnica do sistema. A dependência de infraestrutura computacional robusta também pode restringir o uso da plataforma em contextos com recursos limitados. Entretanto, o trabalho demonstrou que a automação da análise empírica é uma ferramenta valiosa para melhorar a qualidade dos planejadores, fornecendo um ambiente mais consistente e acessível para estudos na área de planejamento automatizado. Com isso, este trabalho contribui significativamente para a padronização e reprodutibilidade de experimentos, consolidando uma base para avanços futuros na área.

**Palavras-chave**: planejamento automatizado, avaliação empírica, inteligência artificial, comparação de planejadores, automação de experimentos.

# ABSTRACT

VITAL, Elinilson. Automated planner evaluation platform. 2024. 52 f. Trabalho de Conclusão de Curso Bacharelado em Ciência da Computação, Universidade de São Paulo. São Paulo, 2024.

Empirically evaluating automated planning systems contributes to the evolution and improvement of planners quality in the context of artificial intelligence. This study presents the development of a platform designed to automate the evaluation of planners, enabling users to schedule, execute, and analyze experiments systematically and reproducibly. The need for such a tool arises from the increasing complexity of planning domains and the importance of consistent and comprehensive comparisons across different approaches to solving planning problems. The platform was designed to centralize the management of predefined domains and problems, ensuring uniformity in the evaluation process. The results obtained with the platform highlight its efficiency in comparing distinct planners by applying customized metrics to the same domains under controlled resource access conditions. Comparative graphs were generated, displaying the total problem-solving time, search time, and normalized coverage over time, allowing a detailed analysis of each planners performance. Data tables were made available to facilitate the incorporation of results into scientific publications. This comparative analysis capability supports the evolution of planners by identifying strengths and weaknesses, fostering advancements in the employed approaches. Despite the positive results, some limitations were encountered during the systems development. Integration with external planners required additional efforts due to the lack of unified standards for data input and output, and configuring multiple containers to ensure isolation and scalability increased the systems technical complexity. Dependence on robust computational infrastructure may also restrict the platforms use in contexts with limited resources. Nevertheless, the study demonstrated that empirical analysis automation is a valuable tool for improving planners quality, providing a more consistent and accessible environment for research in automated planning. Thus, this work significantly contributes to the standardization and reproducibility of experiments, establishing a foundation for future advancements in the field.

**Keywords**: automated planning, empirical evaluation, artificial intelligence, planner comparison, experiment automation.

## LISTA DE FIGURAS


Figura 1 – Tipos de políticas. Vértices representam os estados e as arestas, as ações. Arestas tracejadas representam ações não-determinísticas. Nas figuras, $s_0$ é o estado inicial e $s_3$ a meta. . . . . 3
Figura 2 – Mundo do robô carregador FOND representado por um grafo de transição de estados. Os vértices (retângulos) representam estados e as arestas rotuladas as ações. Arestas tracejadas representam ações não-determinísticas. Extraída de: (SANTOS, 2018) . . . . . 3
Figura 3 – Página Inicial da Plataforma Planalyzing . . . . . 21
Figura 4 – Configuração de Domínios e Problemas . . . . . 22
Figura 5 – Configuração dos Planejadores . . . . . 23
Figura 6 – Lista de Experimentos Agendados . . . . . 24
Figura 7 – Agendamento de Novo Experimento . . . . . 25
Figura 8 – Experimentos Processados . . . . . 26
Figura 9 – Gerando Nova Analise . . . . . 27
Figura 10 – Lista de Análises . . . . . 28
Figura 11 – Diagrama de Contexto da plataforma *Planalyzing* . . . . . 30
Figura 12 – Diagrama de Contêiner da Plataforma *Planalyzing* . . . . . 32
Figura 13 – Processo de Automação de Experimentos . . . . . 34
Figura 14 – Diagrama de Componentes do Orquestrador . . . . . 35
Figura 15 – Diagrama de Código do Orquestrador . . . . . 36
Figura 16 – Diagrama de Entidade e Relacionamento - ER . . . . . 38
Figura 17 – Processo de Automação . . . . . 41
Figura 18 – Arquivo de dados brutos e processados para o planejador Pactl-Sym . . . . 45
Figura 19 – Gráfico de Cobertura sobre o Tempo . . . . . 47
Figura 20 – Gráfico de Comparação da Performance dos Planejadores sobre o Tempo . 48
Figura 21 – Gráfico de Comparação da Performance dos Planejadores sobre o Tamanho 48

## LISTA DE TABELAS

Tabela 1 – Domínio e quantidade de problemas usados em competições internacionais de planejamento . . . . . 9
Tabela 2 – Descrição da competição IPC 2023, detalhando as trilhas, edições anteriores, última ocorrência, número de submissões em 2023, linguagem de entrada utilizada e os links das páginas correspondentes. . . . . 14
Tabela 3 – Tabela de Cobertura Normalizada . . . . . 46

## SUMÁRIO


**1 – INTRODUÇÃO . . . . . 1**
1.1 Domínios e Problemas em Planejamento . . . . . 1
1.2 Competições internacionais e formas de avaliação de planejadores . . . . . 9
1.2.1 As 5 Trilhas . . . . . 10
1.3 Motivação e objetivos . . . . . 11
1.4 Objetivos . . . . . 12
1.4.1 Objetivos específicos . . . . . 12
1.5 Principais contribuições . . . . . 12

**2 – TRABALHOS CORRELATOS . . . . . 13**
2.1 Competição Internacional de Planejamento (IPC) . . . . . 14
2.1.1 Métricas por Trilha na IPC 2023 . . . . . 15
2.2 A plataforma do FastDoward . . . . . 16
2.3 A plataforma do PRP2 . . . . . 17

**3 – PLANALYZING: ESTENDENDO E FLEXIBILIZANDO A PLATAFORMA DO PRP2 . . . . . 19**
3.1 Objetivo . . . . . 20
3.2 Apresentação da plataforma . . . . . 20
3.2.1 Configuração de Domínios e Problemas . . . . . 22
3.2.2 Agendamento de Experimentos . . . . . 24
3.2.3 Experimentos Processados . . . . . 26
3.2.4 Gerando Analises . . . . . 27
3.2.5 Lista de Análises . . . . . 28
3.3 Metodologia . . . . . 29
3.4 Modelagem da plataforma . . . . . 29
3.4.1 Contexto do Sistema . . . . . 29
3.4.2 Diagrama de Contêiner . . . . . 31
3.4.3 Diagrama de Componentes do Orquestrador . . . . . 33
3.4.4 Diagrama de Código do Orquestrador . . . . . 36
3.4.5 Modelo de Dados . . . . . 36
3.5 Processo de Automação . . . . . 39
3.5.1 Registro de Domínios, Problemas e Planejadores pelo Administrador 39
3.5.2 Agendamento de Experimentos pelo Pesquisador . . . . . 41
3.5.3 Execução dos Experimentos pela Plataforma Planalyzing . . . . . 42
3.5.4 Notificação e Análise Empírica dos Resultados . . . . . 42

**4 – USANDO A PLATAFORMA PLANALYZING COM O PLANEJADOR PACTL-SYM . . . . . 43**
4.1 O planejador PACTL-Sym . . . . . 43
4.2 Experimentos realizados . . . . . 44
4.3 Análise empírica . . . . . 45

**5 – CONCLUSÃO . . . . . 50**
5.1 Trabalhos futuros . . . . . 50

**Referências . . . . . 51**

# 1 INTRODUÇÃO

Impulsionado pela automação de tarefas complexas em áreas como robótica e logística, Planejamento Automatizado é um dos principais tópicos da Inteligência Artificial que se concentra na criação de técnicas e algoritmos capazes de gerar sequências de ações para alcançar um objetivo específico, partindo de um estado inicial. O planejamento automatizado busca soluções de forma autônoma e independentes do domínio de aplicação. Existem diferentes abordagens de planejamento, entre elas: planejamento determinístico, não determinístico, probabilístico e de múltiplos agentes. A seguir, cada uma dessas abordagens é descrita brevemente (GHALLAB et al., 1998):

- **Planejamento determinístico (ou clássico):** Esta abordagem faz diversas suposições restritivas do ambiente, dentre elas assume que o ambiente totalmente previsível, onde o resultado de cada ação é conhecido. A solução para um problema de planejamento clássico é uma sequência de ações que, partindo de um estado inicial, conduz o agente a um estado que satisfaz uma meta preestabelecida.
- **Planejamento não-determinístico e completamente observável:** Um domínio de planejamento não-determinístico é aquele em que a execução de cada ação pode ter mais de um resultado possível. Devido ao não-determinismo das ações, a solução para um problema desse tipo é uma **política**, isto é, um mapeamento que associa os estados às melhores ações a serem executadas nesses estados. Esta abordagem é descrita com mais detalhes na Seção 1.1.
- **Planejamento probabilístico:** Considera a incerteza no resultado das ações e no estado do ambiente, porém, neste caso os possíveis efeitos das ações possuem uma probabilidade associada a eles. Assim como abordagem anterior, a solução para um problema de planejamento probabilístico é uma política. Uma das principais abordagens para resolver esse tipo de problema baseia-se em Processos de Decisão Markovianos (MDPs). O planejamento probabilístico permite que o agente tome decisões ótimas em situações de incerteza, considerando a probabilidade de sucesso de cada ação.
- **Planejamento de multi-agentes:** Envolve a coordenação de múltiplos agentes para atingir um objetivo comum ou objetivos individuais. Este tipo de planejamento apresenta desafios adicionais, como a comunicação e a cooperação entre agentes.

## 1.1 Domínios e Problemas em Planejamento

No planejamento automatizado, os domínios são ambientes estruturados nos quais as tarefas de planejamento são definidas, envolvendo um conjunto de estados, ações e a relação de transição entre os estados. Neste trabalho, estamos interessados em domínios não-determinísticos.

Conforme citado anteriormente, no planejamento completamente observável e não-determinístico (*Fully-Observable Non-Deterministic Planning* - FOND), as ações possuem efeitos não-determinísticos, ou seja, uma ação pode levar a vários estados possíveis, porém sem que haja uma probabilidade associada a cada uma dessas possibilidades. Formalmente, um domínio de planejamento FOND pode ser definido como:

Domínio de Planejamento FOND

Um domínio de planejamento FOND é definido pela tupla $D = \langle S, A, T \rangle$, em que:

- $S$ é um conjunto finito de estados possíveis do ambiente;
- $A$ é um conjunto finito de ações não-determinísticas que o agente pode executar;
- $T : S \times A \to 2^S$ é uma função de transição não-determinística de estados, onde dado um estado e uma ação, a função de transição pode levar a um conjunto de estados possíveis (isto é, um subconjunto do conjunto potência $2^S$).

Um problema de planejamento FOND é dado pela tupla $P = \langle D, s_0, G \rangle$, onde:

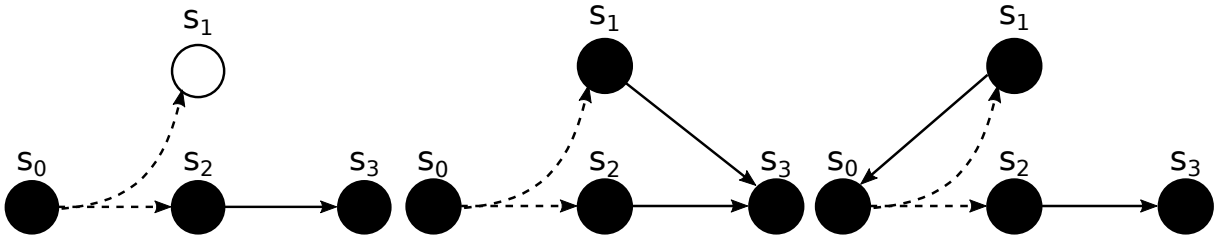
- $D$: domínio de planejamento, conforme definido acima;
- $s_0 \in S$: estado inicial do ambiente;
- $G \subseteq S$: conjunto de estados meta.

Política

A **solução** para um problema de planejamento FOND $P$ é uma **política** $\pi$, que é um mapeamento de estados em ações $\pi : S \to A$, indicando a melhor ação $\pi(s) \in A$ a ser executada no estado $s \in S$. As políticas podem ser classificadas em três tipos:

- **Política fraca** (Figura 1a): pode alcançar um estado meta; no entanto, não há garantia de sucesso devido ao não-determinismo das ações. A Figura 1a apresenta um exemplo de uma política fraca. Nessa figura, os vértices representam os estados, e as arestas direcionadas representam as ações. As arestas tracejadas que partem do mesmo estado indicam os efeitos não-determinísticos de uma mesma ação. Neste exemplo, $s_0$ é o estado inicial e $s_3$ é um estado que satisfaz a meta. Observe que, na figura, a execução de uma ação não-deterministica em $s_0$ pode levar tanto ao estado $s_1$ quanto ao estado $s_2$. No entanto, não é possível alcançar a meta a partir do estado $s_1$.
- **Política forte** (Figura 1b): garante que um estado meta será alcançado em um número finito de passos, independentemente do não-determinismo das ações.
- **Política forte-cíclica** (Figura 1c): pode conter ciclos, mas ainda assim garante que um estado meta será alcançado, considerando que o agente eventualmente sairá dos ciclos existentes na política.

Figura 1 – Tipos de políticas. Vértices representam os estados e as arestas, as ações. Arestas tracejadas representam ações não-determinísticas. Nas figuras, $s_0$ é o estado inicial e $s_3$ a meta.

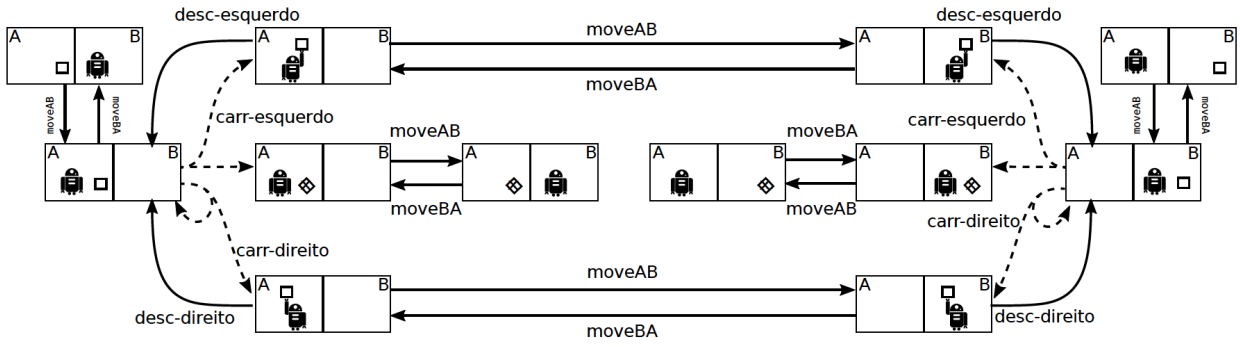



(a) Política fraca (b) Política forte (c) Política forte-cíclica

Exemplo de Planejamento FOND

A Figura 2 representa uma instância de problema do domínio de planejamento do Robô de Carga não-determinístico. Neste domínio, o agente deve transportar caixas entre salas, mas enfrenta incertezas relacionadas ao sucesso das ações dependendo do braço utilizado (direito ou esquerdo).

Figura 2 – Mundo do robô carregador FOND representado por um grafo de transição de estados. Os vértices (retângulos) representam estados e as arestas rotuladas as ações. Arestas tracejadas representam ações não-determinísticas. Extraída de: (SANTOS, 2018)




Um problema de planejamento pode ser representado explicitamente através de um grafo de transições de estados, conforme o exemplo da Figura 2. Contudo, essa representação torna-se inviável quando o conjunto de estados a ser representado cresce exponencialmente. Para contornar essa limitação, um problema de planejamento pode ser representado implicitamente através de uma linguagem capaz de descrever os domínios e problemas, como, por exemplo, a linguagem STRIPS, ADL, PDDL, entre outras.

A linguagem de Definição de Domínio de Planejamento (*Planning Domain Definition Language* - PDDL), foi criada especificamente para a competição de planejamento AIPS-98 (LONG et al., 2000), para encorajar o compartilhamento de problemas e algoritmos e permitir uma comparação do desempenho dos planejadores em diferentes problemas. A PDDL destina-se a expressar a *física* de um domínio, ou seja, quais predicados existem, quais ações são possíveis,

qual é a estrutura das ações compostas e quais são os efeitos das ações, o que permite descrever domínios e problemas de forma que os planejadores possam entendê-los e resolvê-los. Esta linguagem é derivada de vários formalismos, incluindo ADL, SIPE-2, Prodigy-4.0, UMCP, Unpop e UCPOP. Sua sintaxe e semântica incorporam recursos como ações básicas do STRIPS, efeitos condicionais, quantificação universal, axiomas de domínio, restrições de segurança, ações hierárquicas e a capacidade de gerenciar vários problemas em vários domínios com diferentes subconjuntos de recursos de linguagem (MCDERMOTT et al., 1998), sua sintaxe é baseada na notação do BNF Estendido (Extended Backus-Naur Form - EBNF) [1].

A especificação PDDL separa a descrição do domínio das descrições dos problemas. Um modelo de domínio descreve o ambiente e as ações, enquanto um problema de planejamento detalha o estado inicial e a meta, formando uma tarefa de planejamento quando combinado. O domínio e os problemas são fornecidos como entradas para os planejadores como arquivos separados; isso permite gerar soluções para os problemas de planejamento apresentados, reaproveitando o domínio em outros problemas. Essa abordagem permite separar a definição dos domínios dos problemas. A seguir, são descritos brevemente um domínio e um problema de planejamento, bem como algumas de suas principais estruturas para representar uma tarefa de planejamento em PDDL.

- **Domínio de planejamento:** Esse componente fornece uma descrição geral do ambiente e das ações disponíveis em um determinado domínio. Ele define os tipos de objetos, predicados e ações que podem ser usados no planejamento.
  - **Ações:** As ações são definidas com pré-condições e efeitos. Eles descrevem como o estado do mundo muda quando uma ação é executada. Resumidamente, as precondições de uma ação indicam as condições que devem ser verdadeiras em um estado para que a ação possa ser executada, enquanto os efeitos descrevem as alterações no estado resultantes da execução da ação.
  - **Predicados:** Eles são usados para descrever as precondições e efeitos das ações e definir os estados do domínio.
- **Problema de planejamento:** especifica os objetos concretos, o estado inicial e o estado da meta, descritos por meio dos predicados, para uma tarefa de planejamento específica. É uma instância específica do modelo de domínio, detalhando o cenário exato a ser abordado pelo mecanismo de planejamento .

Esses componentes trabalham juntos para permitir a descrição e a resolução de tarefas de planejamento, permitindo que os mecanismos de planejamento gerem planos com base no domínio e nas especificações do problema. A seguir serão apresentados dois exemplos de PDDL, o primeiro que representa o domínio (Listagem 1.1) e o segundo a definição do problema que pode ser aplicado ao domínio (Listagem 1.2).

[1] O EBNF é uma forma de escrever as regras de um idioma de forma clara e estruturada. Nessa notação, cada regra é escrita como um elemento sintático, que faz parte da linguagem, seguido por dois pontos e um sinal de igual (:: =) e, em seguida, a expansão, que mostra do que o elemento pode ser feito (EARLEY, 1974).

Especificação do domínio robô carregador em PDDL

O domínio **robô carregador** (Listagem 1.1), descreve um cenário onde um robô equipado com duas garras, esquerda e direita, manipula caixas entre salas. Ele utiliza estruturas como tipos, predicados e ações para definir o comportamento e o estado do sistema. Os **tipos** especificados incluem `room` (salas), `box` (caixas) e `gripper` (garras). Constantes como `left` e `right` representam as garras disponíveis.

```
(define (domain gripper)
 (:requirements :strips :typing :non-deterministic)

 (:types room box gripper)

 (:constants left right - gripper)

 (:predicates
  (at-robby ?r - room)
  (at-box ?b - box ?r - room)
  (free ?g - gripper)
  (carry ?o - box ?g - gripper)
  (whole ?b - box))

 (:action move
  :parameters  (?from ?to - room)
  :precondition (at-robby ?from)
  :effect (and  (at-robby ?to) (not (at-robby ?from))))

 (:action pick-left
  :parameters (?obj - box ?room - room)
  :precondition  (and  (at-box ?obj ?room) (at-robby ?room) (
     free left) (whole ?obj) )
  :effect
   (oneof
    ( and )
    ( and (carry ?obj left) (not (at-box ?obj ?room)) (not (
       free left)) )
   )
 )

 (:action pick-right
  :parameters (?obj - box ?room - room)
```

```
32   :precondition  (and  (at-box ?obj ?room) (at-robby ?room) (
       free right) (whole ?obj) )
33   :effect
34    (oneof
35     ( and (not (whole ?obj)) )
36     ( and (carry ?obj right) (not (at-box ?obj ?room)) (not (
         free right)) )
37    )
38  )
39
40  (:action drop
41   :parameters (?obj - box ?room - room ?gripper - gripper)
42   :precondition (and (carry ?obj ?gripper) (at-robby ?room))
43   :effect (and (at-box ?obj ?room) (free ?gripper) (not (
       carry ?obj ?gripper))))
44 )
```

Listing 1.1 – Definição em PDDL do domínio de mundo do robô carregador

Os **predicados** modelam o estado do domínio. Por exemplo, `(at-robby ?r - room)` indica a localização do robô, enquanto `(at-box ?b - box ?r - room)` descreve a posição de uma caixa em uma sala. Outros predicados, como `(free ?g - gripper)` e `(carry ?o - box ?g - gripper)`, representam, respectivamente, se uma garra está livre ou se está carregando uma caixa. Na especificação do domínio dizemos que os predicados não estão instanciados, uma vez que descrevem as informações de maneira "genérica". Por exemplo, o predicado `(at-robby ?r - room)` não especifica em qual sala o robô está. A instanciação dos predicados se torna possível com a especificação do problema, que determina, por exemplo, quantas salas existem e quais são os seus identificadores. Para o mesmo domínio, podemos ter um problema com duas salas ou um problema com dez salas, dependendo da especificação do problema.

As **ações** descrevem as interações possíveis no domínio. A ação `move` permite ao robô se mover entre salas, atualizando sua posição com o predicado `(at-robby ?to)` e removendo a antiga posição `(not (at-robby ?from))`. Para manipular caixas, as ações `pick-left` e `pick-right` permitem ao robô pegar uma caixa com a garra esquerda ou direita, desde que ela esteja intacta `(whole ?b)`, a garra esteja livre e o robô esteja na mesma sala da caixa. Essas ações incluem efeitos não determinísticos, como falhar em pegar a caixa ou danificá-la.

Por fim, a ação `drop` permite que o robô solte uma caixa em uma sala específica, atualizando a posição da caixa `(at-box ?b ?r)` e liberando a garra `(free ?g)`.

Esse domínio exemplifica o uso do PDDL para modelar sistemas complexos, combinando estrutura lógica com ações que alteram o estado do ambiente. Ele inclui suporte para extensões

como tipagem e efeitos não determinísticos, ampliando sua expressividade para problemas de planejamento FOND.

Especificação de um problema do domínio robô carregador em PDDL

A Listagem 1.2 apresenta uma instância de problema do domínio `robô carregador` em PDDL. Este problema modela um cenário onde um robô deve transportar uma caixa de uma sala para outra, respeitando as condições iniciais e alcançando o estado desejado especificado na meta.

```
(define (problem gripper_1_boxs)
 (:domain gripper)

 (:objects
  rooma roomb - room
  box1 - box)

 (:init
  (free left)
  (free right)
  (at-robby rooma)
  (at-box box1 rooma)
  (whole box1)
 )

 (:goal
  (and
   (at-box box1 roomb)
   (whole box1)
  )
 )
)
```

Listing 1.2 – Definição em PDDL de um problema no domínio do mundo do robô carregador. Este problema possui duas salas e uma caixa. A caixa inicialmente está na sala A e deve estar na sala B na meta.

Estruturas do Problema

A definição do problema começa com:

```
(define (problem gripper_1_boxs) (:domain gripper)
```

Objetos

Os objetos usados no problema são:

```
(:objects rooma roomb - room box1 - box)
```

Aqui, `rooma` e `roomb` são as salas, enquanto `box1` representa a caixa.

Estado Inicial

O estado inicial do problema é definido como:

```
(:init (free left) (free right) (at-robby rooma) (at-box box1
rooma) (whole box1) )
```

No início, as duas garras (`left` e `right`) estão livres, o robô (`at-robby`) está na sala `rooma`, e a caixa `box1` está na mesma sala (`at-box box1 rooma`). Além disso, a caixa está intacta (`whole box1`). Note que, diferentemente da especificação do domínio, a descrição do estado inicial e da meta do problema são feitas através de predicados instanciados. Por exemplo, no estado inicial, é especificado que o robô está na sala "a" `(at-robby rooma)`.

Condições de Meta

A meta é definida como:

```
(:goal (and (at-box box1 roomb) (whole box1) ) )
```

O objetivo é que a caixa `box1` esteja na sala `roomb` (`at-box box1 roomb`), e que ela continue intacta (`whole box1`).

Esse problema exemplifica como o PDDL é usado para descrever um sistema de planejamento que envolve objetos, estado inicial e meta. Ele demonstra a modelagem de situações simples, como o transporte de um objeto, utilizando uma estrutura lógica clara e bem definida.

Em competições internacionais como a Competição Internacional de Planejamento (IPC), são elaborados benchmarks para avaliar as capacidades dos planejadores em vários domínios e problemas, garantindo uma análise abrangente de seu desempenho. Esses benchmarks fornecem um conjunto bem conhecido de domínios de referência, como por exemplo os domínios *acrobatics, beam-walk, blocks-world, first-responders e triangle-tireworld* e outros. Uma lista com o nome do domínio e a quantidade de problemas comumente utilizados para avaliar planejadores FOND é apresentada na Tabela 1, os problemas variam em tamanho e complexidade.

Na IPC de 2023, as avaliações de desempenho das trilhas foram determinadas por vários critérios, que variaram em diferentes faixas e subfaixas. Cada submissão recebeu até três

configurações por faixa, e as abordagens de planejamento do mesmo grupo foram consideradas submissões diferentes se seu funcionamento interno fosse suficientemente diferente. Os planejadores podiam usar até 8 GiB de RAM e 30 minutos de tempo de execução por instância.

A competição contou com cinco faixas distintas: a faixa clássica (determinística), a faixa numérica, a faixa Hierarchical Task Networks (HTN), a trilha de aprendizado e a trilha de aprendizado probabilístico e por reforço. Cada trilha avaliou as metodologias de planejamento por meio de uma ou mais subfaixas, com o objetivo de ultrapassar os limites do desempenho atual do planejador.

Tabela 1 – Domínio e quantidade de problemas usados em competições internacionais de planejamento

| Domínios | Número de Problemas |
|---|---|
| acrobatics | 8 |
| beam-walk | 11 |
| blocksworld | 37 |
| blocksworld-2 | 16 |
| blocksworld-new | 51 |
| chain-of-rooms | 10 |
| doors | 15 |
| earth-observation | 40 |
| elevators | 16 |
| ex-blocksworld | 15 |
| first-responders | 101 |
| forest | 90 |
| forest-new | 100 |
| islands | 60 |
| miner | 51 |
| rectangle-tireworld | 30 |
| rectangle-tireworld-noghost | 30 |
| st_blocksworld | 30 |
| st_first_responders | 75 |
| st_tires | 14 |
| tidyup-mdp | 10 |
| tireworld | 16 |
| tireworld-spiky | 11 |
| tireworld-truck | 74 |
| triangle-tireworld | 42 |
| zenotravel | 16 |
| Total (26) | 969 |

## 1.2 Competições internacionais e formas de avaliação de planejadores

As competições têm um papel essencial no campo do planejamento automatizado, servindo como um ambiente que combina colaboração e competitividade. Elas oferecem uma plataforma única para pesquisadores e profissionais trocarem ideias, compartilharem

experiências e testarem suas abordagens em cenários desafiadores. Além de fomentar a inovação, essas competições ajudam a estabelecer padrões, identificar melhores práticas e acelerar a implementação de soluções eficazes. Ao mesmo tempo, o caráter competitivo incentiva o desenvolvimento de tecnologias mais robustas, promovendo avanços significativos na área e impulsionando o estado da arte no planejamento automatizado.

A Competição Internacional de Planejamento (IPC), realizada no âmbito da Conferência Internacional sobre Planejamento e Agendamento (*International Conference on Planning and Scheduling* - ICAPS), é um evento de destaque que oferece uma plataforma para especialistas avaliarem e compararem seus planejadores em diversos domínios e problemas. Nesse ambiente competitivo, os participantes buscam desenvolver e demonstrar as abordagens mais eficazes para resolver os desafios propostos pela competição. Em sua edição de 2023 (IPC, 2023), a IPC foi organizada em cinco categorias distintas, chamadas de trilhas, cada uma dedicada a explorar um tipo específico de problema de planejamento, promovendo uma análise abrangente das capacidades dos planejadores (TAITLER et al., 2024).

### 1.2.1 As 5 Trilhas

1. Trilha Clássica (Determinística): Essa trilha lida com problemas em que tudo é previsível. Imagine planejar uma viagem em que você saiba exatamente quanto tempo levará cada parte da viagem. Essa trilha testa métodos que funcionam bem quando não há surpresas.
2. Trilha Numérica: Aqui, o foco está nos problemas envolvendo números. Por exemplo, se você está planejando um orçamento, precisa considerar quanto dinheiro tem e quanto precisa gastar. Esta trilha analisa o quão bem os planejadores lidam com esses tipos de detalhes numéricos.
3. Trilha de Redes de Tarefas Hierárquicas (HTN): Essa trilha trata de dividir grandes tarefas em tarefas menores e gerenciáveis. Pense nisso como planejar um casamento, onde você precisa organizar muitas tarefas menores, como reservar um local, enviar convites e organizar a comida. A HTN ajuda a organizar essas tarefas com eficiência.
4. Trilha de Aprendizagem: Essa trilha explora como os planejadores podem aprender com experiências passadas para tomar melhores decisões. É como um aluno que melhora em matemática praticando problemas e aprendendo com os erros.
5. Trilha de Aprendizagem Probabilística e por Reforço: Esta trilha trata da incerteza e do aprendizado com o meio ambiente. Imagine jogar um novo videogame em que você aprende as melhores estratégias experimentando movimentos diferentes e vendo o que funciona. Essa trilha testa planejadores capazes de lidar com incertezas e aprender com o tempo.

Note que, entre as trilhas existentes na competição, atualmente não há uma dedicada ao planejamento FOND. Em 2008, houve uma edição da competição para avaliar planejadores FOND, mas essa trilha não foi repetida desde então. Atualmente, além dos domínios apresentados na Tabela 1, que são comumente usados por autores de planejadores FOND para avaliar

seus planejadores, existe a plataforma PRP2, descrita em mais detalhes no Capítulo 2, que avalia planejadores FOND.

Esse cenário ressalta como a competição desempenha um papel fundamental no avanço da pesquisa em planejamento. O principal objetivo da competição é avaliar empiricamente os sistemas de planejamento de última geração em uma série de problemas de referência. Os objetivos da IPC são promover a pesquisa em planejamento, destacar os desafios enfrentados pela comunidade de planejamento e fornecer novos e interessantes problemas como benchmarks para pesquisas futuras.

Cada trilha da competição testa os planejadores e analisa o histórico desse tipo de planejamento, mostrando como diferentes abordagens se encaixam no panorama geral do campo. A competição também explora novas áreas e tendências no planejamento, evidenciando sua evolução. Ao testar diferentes planejadores em trilhas específicas, ela compartilha os resultados dos testes, contribuindo para o avanço da área e a descoberta de melhores soluções para problemas complexos.

## 1.3 Motivação e objetivos

A proposta de uma plataforma automatizada para análise empírica de planejadores surge como uma resposta à necessidade de lidar com a crescente complexidade dos domínios de aplicação e a explosão no número de problemas. Essa necessidade é particularmente relevante na análise de novos planejadores FOND. Atualmente, essa análise envolve a busca pelos benchmarks utilizados pela comunidade de planejamento por parte dos desenvolvedores e a criação de alguma automação para execução e visualização dos resultados dos experimentos, uma vez que a quantidade de domínios e problemas normalmente utilizados é grande, como mostrado na Tabela 1. A criação de uma plataforma automatizada para análise empírica permite a sistematização e padronização dos testes, facilitando tanto a validação quanto o aprimoramento de novos planejadores, bem como a comparação entre diferentes planejadores.

Ao pesquisar especificamente por planejadores FOND, encontramos a plataforma PRP2, apresentada no repositório MUISE (2024a), que avalia planejadores FOND. O código disponibilizado implementa testes em diversos domínios, problemas e planejadores, executados via linha de comando em sistemas Linux. Por meio de um Jupyter Notebook, os resultados são resumidos em tabelas e gráficos. Os gráficos comparam os planejadores com o PR2 (MUISE, 2024b), permitindo avaliar o desempenho dos demais em relação ao tempo de execução e ao tamanho das soluções geradas pelo PR2.

O código de avaliação do PR2 ainda está em desenvolvimento, mas permitiu explorar e compreender o processo de testes, incluindo a estrutura de diretórios, bem como os arquivos relacionados a domínios, problemas e planejadores. Com base nesse estudo preliminar, foi possível adicionar novos planejadores e realizar testes fora do escopo original do projeto. A partir desse entendimento, surgiu a proposta da plataforma Planalyzing, com o objetivo de automatizar esse processo, atualmente manual e que exige conhecimento técnico em linguagens

de programação, arquitetura de sistemas, ambientes Linux e virtualização de contêineres. A automação visa permitir que os pesquisadores concentrem seus esforços em questões de pesquisa, reduzindo a carga operacional e técnica associada à execução dos experimentos.

## 1.4 Objetivos

O objetivo principal deste trabalho é desenvolver uma plataforma integrada para automatizar a avaliação de planejadores, possibilitando aos usuários agendar, executar e analisar experimentos de maneira reprodutível. A plataforma permitirá a gestão centralizada de domínios e problemas predefinidos, garantindo consistência no processo de avaliação.

### 1.4.1 Objetivos específicos

- Identificar as principais técnicas de automação utilizadas na avaliação de sistemas de planejamento automatizado;
- Identificar as métricas mais relevantes avaliadas em competições de planejadores automatizados;
- Projetar uma arquitetura modular que permita a configuração de experimentos personalizados;
- Automatizar a execução dos experimentos, a partir das configurações dos experimentos registradas pelos pesquisadores, com as combinações de domínios, problemas e planejadores;
- Automatizar a extração das métricas personalizadas de cada planejador, para isso, serão utilizadas técnicas baseadas em expressões regulares.
- Apresentar os resultados em gráficos e tabelas;
- Avaliar a escalabilidade e flexibilidade da plataforma em cenários experimentais variados;
- Elaborar um estudo de caso com planejadores FOND.

## 1.5 Principais contribuições

A contribuição acadêmica deste trabalho está no desenvolvimento de uma ferramenta para avaliar planejadores automatizados em diversos contextos. A proposta inclui a automação do processo de geração de testes, incorporando múltiplas variáveis de entrada como domínios, problemas, métricas e planejadores em cenários experimentais variados, promovendo o avanço no desenvolvimento de novos planejadores.

No campo prático, essa abordagem beneficia áreas como logística, saúde, manufatura e robótica, que dependem fortemente de sistemas de planejamento automatizado. A avaliação sistemática e automatizada contribui para a criação de planejadores mais eficientes, capazes de oferecer soluções otimizadas para problemas reais.

# 2 TRABALHOS CORRELATOS

As dificuldades em avaliar planejadores envolvem o uso de benchmarks, entendidos como o conjunto de domínios comumente utilizados em experimentos de planejamento automatizado (ver Tabela 1), ou o desenvolvimento de novos, que devem incorporar a variedade e complexidade de contextos do mundo real nos quais os agentes funcionam. Particularmente, quando a abordagem de resolução de problemas utilizada pelo planejador é nova, isso se torna ainda mais desafiador. Essas dificuldades exigem uma avaliação de métricas que possam capturar com precisão o desempenho em vários cenários, garantindo que as soluções sugeridas sejam válidas e reprodutíveis. Para isso, é essencial estabelecer uma estrutura que descreva os processos e práticas a serem seguidos para testar os benchmarks propostos.

Os autores do artigo *Leveraging FOND planning technology to solve multi-agent planning problems* (MUISE et al., 2015) exploraram uma nova metodologia para questões de FOND, semelhante às mencionadas anteriormente, e como não havia padrões existentes, os pesquisadores desenvolveram um benchmark que incluiu três domínios: Blocksworld, Tic-Tac-Toe e Sokoban. Eles realizaram 1.000 testes para cada jogo, empregando simulações de Monte Carlo para imitar as ações de outros jogadores. O planejador em avaliação, denominado PRP, foi utilizado para formular estratégias, restritas a um máximo de 30 minutos e 2 gigabytes de memória. A eficácia dessa abordagem foi avaliada por meio de três métricas: taxa de sucesso (a porcentagem de metas alcançadas), tamanho da política (o número total de de pares estado-ação que compõem a política) e a duração do planejamento. O benchmark definido foi essencial para analisar o desempenho do PRP em um ambiente controlado, permitindo avaliar o planejador em contextos desconhecidos. Os resultados obtidos revelaram *insights* sobre a adaptabilidade do planejador e seu potencial para otimizar decisões em cenários novos.

Avaliar planejadores é uma tarefa desafiadora, especialmente na ausência de benchmarks padronizados e ferramentas de testes. A criação de benchmarks específicos para avaliar planejadores exige tempo, conhecimento detalhado do domínio, problemas e ferramentas de testes disponíveis. Essa dificuldade é ainda maior quando se deseja comparar diferentes planejadores, pois isso acrescenta uma compreensão profunda de como cada planejador opera, incluindo suas entradas, saídas e os detalhes técnicos para executar, coletar e avaliar os resultados. Nesse sentido, no trabalho (MUISE, 2024b) os autores apresentam um novo planejador, o PR2, também um planejador FOUND. Para obter seus resultados empíricos, foram implementados novos benchmarks para posicionar o planejador em relação aos que representam o estado da arte na resolução de problemas FOND. Na seção 2.3 apresentaremos mais detalhes.

Embora experimentos individuais sejam essenciais para validar novos planejadores, como os estudos realizados no desenvolvimento do PRP e PR2, eles frequentemente enfrentam limitações devido à falta de padrões e benchmarks uniformes. Essa abordagem isolada, embora valiosa, dificulta a comparação direta entre planejadores e a replicação dos resultados em

diferentes cenários. É nesse contexto que as competições internacionais, como a IPC, surgem como um modelo eficaz. Elas oferecem uma estrutura padronizada que supera as limitações de avaliações individuais, uniformizando os benchmarks, métricas e condições experimentais. Além de permitir uma comparação justa entre planejadores, essas competições criam um ambiente colaborativo que impulsiona a inovação e define novos parâmetros para o estado da arte na área. A seguir, resumiremos como a competição IPC 2023 foi estruturada e apresentaremos mais detalhes de algumas plataformas relacionadas a este trabalho.

## 2.1 Competição Internacional de Planejamento (IPC)

A IPC de 2023 avaliou métodos de planejamento em cinco faixas: clássica, numérica, HTN, aprendizado e aprendizado probabilístico/por reforço, como foi descrito na seção 1.2; cada uma representa uma metodologia distinta e requer abordagens diferentes para a resolução de problemas. Informações detalhadas estão disponíveis na Tabela 2.

Tabela 2 – Descrição da competição IPC 2023, detalhando as trilhas, edições anteriores, última ocorrência, número de submissões em 2023, linguagem de entrada utilizada e os links das páginas correspondentes.

| **Trilha** | **Ed.** | **Última** | **Submissões** | **Ling.** | **Webpage** |
|---|---|---|---|---|---|
| Clássica | 9 | 2018 | 65 | PDDL | https://ipc2023-classical.github.io/ |
| Numérica | 0 | - | 6 | PDDL2.1 | https://ipc2023-numeric.github.io/ |
| HTN | 1 | 2020 | 11 | HDDL | https://ipc2023-htn.github.io/ |
| Aprendizagem | 3 | 2014 | 6 | PDDL | https://ipc2023-learning.github.io/ |
| Probabilística | 6 | 2018 | 4 | RDDL | https://ataitler.github.io/IPPC2023/ |

fonte: (TAITLER et al., 2024)

A tabela 2, apresenta as informações sobre a competição, incluindo o número de edições, o último ano em que cada faixa foi realizada, as inscrições para 2023, a linguagem utilizada e links para mais informações. A faixa Clássica é a mais estabelecida, com 9 edições e 65 inscrições em 2023, enquanto a faixa Numérica foi realizada pela primeira vez em 2023. A faixa HTN, mais recente, teve sua última edição em 2020 e atraiu 11 inscrições. As faixas clássicas e probabilísticas utilizam linguagens diferentes (PDDL e RDDL), refletindo abordagens técnicas distintas. A diversidade de idiomas e o número de edições indicam um escopo amplo e em evolução da competição, com links que promovem o engajamento e o compartilhamento de conhecimento na comunidade. A variação nas inscrições pode indicar o interesse ou a acessibilidade das faixas.

As avaliações de desempenho das faixas foram determinadas por vários critérios, que variaram nas diferentes faixas e subfaixas. Cada submissão recebeu até três configurações por faixa, e as abordagens de planejamento do mesmo grupo foram consideradas submissões diferentes se seu funcionamento interno fosse suficientemente diferente. A seguir, serão detalhados os critérios de pontuação de cada faixa.

### 2.1.1 Métricas por Trilha na IPC 2023

As métricas utilizadas na IPC 2023 foram distribuídas entre as cinco trilhas principais, cada uma adaptada às características específicas dos problemas abordados. A seguir, detalhamos como as métricas foram calculadas para cada trilha:

1. **Trilha Clássica (Determinística)**: Avalia planejadores em ambientes totalmente observáveis e determinísticos, com três subtrilhas:
   - *Ótima*: Pontuação score $= 1$ se o planejador encontra o plano ótimo, $0$ caso contrário. Tempo limite: 30 minutos.
   - *Satisfatória*: Pontuação score $= \frac{C^*}{C}$, onde $C^*$ é o custo do plano de referência e $C$ é o custo do plano gerado. Planos não resolvidos recebem score $= 0$.
   - *Ágil*: Pontuação score $= 1 - \frac{\log(T)}{\log(300)}$, onde $T$ é o tempo de solução em segundos. score $= 1$ se $T \leq 1$ segundo, e score $= 0$ para planos não resolvidos em até 300 segundos.
2. **Trilha Numérica**: Avalia planejadores que lidam com variáveis numéricas e recursos, como energia e coordenadas.
   - Pontuação similar à subtrilha *Ótima* da Trilha Clássica, com score $= 1$ para planos ótimos e $0$ caso contrário.
   - Custos cumulativos foram usados como métricas comparativas em problemas de recursos.
3. **Trilha de Redes de Tarefas Hierárquicas (HTN)**: Focada em problemas hierárquicos, onde grandes tarefas são divididas em subtarefas menores.
   - Pontuação similar à Trilha Clássica (*Ótima*): score $= 1$ para planos hierarquicamente válidos e ótimos, $0$ caso contrário.
   - A avaliação considera a complexidade hierárquica dos problemas.
4. **Trilha de Aprendizagem**: Avalia planejadores que aprendem com experiências anteriores para resolver novos problemas dentro do mesmo domínio.
   - Pontuação baseada na subtrilha *Satisfatória*: score $= \frac{C^*}{C}$, onde $C^*$ é o custo do plano de referência.
   - Inclui análise de melhorias em planos subsequentes gerados pelo planejador.
5. **Trilha de Aprendizagem Probabilística e por Reforço**: Explora planejadores que lidam com incertezas e aprendizado por interação com o ambiente.
   - Pontuação baseada na subtrilha *Ágil*: score $= 1 - \frac{\log(T)}{\log(300)}$, onde $T$ é o tempo de solução.
   - A avaliação inclui a capacidade do planejador de se adaptar e aprender com iterações anteriores.

O processo de avaliação envolveu a execução de cada configuração do planejador em todas as instâncias de uma faixa por um período especificado, com 30 minutos alocados para as trilhas ideais e satisfatórias e 5 minutos para a trilha ágil, tudo dentro de um limite de memória de 8 GiB.

As pontuações foram atribuídas com base em critérios específicos para cada faixa. Na faixa ideal, uma pontuação de 1 foi dada para resolver uma instância de forma otimizada, enquanto na faixa satisfatória, a pontuação foi a relação entre o custo do plano mais barato descoberto e o custo de um plano de referência. A trilha ágil usava um sistema de pontuação que contabilizava o tempo gasto para resolver uma tarefa, com pontuações variando de 0 a 1 com base na escala logarítmica do tempo gasto.

Essa estrutura de avaliação permitiu uma comparação eficaz de diferentes configurações de planejadores, destacando seus pontos fortes e fracos em vários desafios de planejamento.

## 2.2 A plataforma do FastDoward

O Fast Downward(HELMERT, 2006) é um planejador clássico que utiliza busca heurística para resolver problemas de planejamento determinístico codificados no PDDL2.2. Esse sistema é capaz de lidar com recursos avançados, como condições de ADL, efeitos e predicados derivados (axiomas). Ao contrário de outros planejadores, ele não usa diretamente a representação proposicional do PDDL. Em vez disso, ele traduz a entrada em tarefas de planejamento de valores múltiplos, que tornam explícitas as restrições implícitas. Existe uma comunidade (HELMERT, 2024) que fornece uma plataforma flexível para a avaliação de algoritmos de planejamento, facilitando a execução de benchmarks e a análise de desempenho em diferentes contextos usando o Fast Downward. O sistema suporta uma ampla gama de funcionalidades, incluindo algoritmos de busca personalizados, heurísticas avançadas e configurações específicas para competições, como as utilizadas na IPC. O uso do Fast Downward envolve a definição de domínios e problemas no formato PDDL, que são processados pelo sistema para gerar soluções otimizadas. Por exemplo, a configuração `LAMA-2011`, utilizada em competições, pode ser executada com o comando:

```
./fast-downward.py --alias seq-sat-lama-2011 domain.pddl problem.pddl
```

Essa flexibilidade permite que pesquisadores adaptem o planejador a diferentes cenários, tornando-o uma ferramenta essencial para a comunidade científica.

Uma das formas de utilizar o **Fast Downward** é por meio de contêineres `Docker`, que simplificam a execução em ambientes padronizados. Para executar benchmarks, é possível montar um diretório local contendo arquivos PDDL no contêiner e rodar o planejador utilizando configurações específicas. Por exemplo, o comando:

```
docker run -v $(pwd):/benchmarks fast-downward --alias lama-first \
/benchmarks/domain.pddl /benchmarks/problem.pddl
```

permite resolver tarefas de planejamento diretamente no ambiente do `Docker`. O sistema também oferece suporte a múltiplas heurísticas e algoritmos de busca, como `A*` com heurísticas de corte de marcos e busca gulosa `best-first` com operadores preferenciais. Além disso, o **Fast Downward** fornece códigos de saída claros e uma documentação abrangente, garantindo

que usuários de diferentes níveis de experiência possam configurá-lo e utilizá-lo de maneira eficiente. Isso o torna ideal tanto para fins educacionais quanto para pesquisa avançada.

## 2.3 A plataforma do PRP2

O trabalho de Muise (2024b) resultou no desenvolvimento de uma plataforma de avaliação de planejadores, cujo código foi disponibilizado no repositório Muise (2024a). O principal objetivo dessa plataforma foi realizar uma análise detalhada do desempenho do planejador PR2, comparando-o com outros planejadores do tipo FOND. A avaliação foi conduzida com base em três aspectos principais: a cobertura, ou seja, a quantidade de problemas que cada planejador consegue resolver; o tamanho das soluções encontradas; e o tempo necessário para encontrar essas soluções (tempo de solução). Essa análise permitiu uma comparação entre os planejadores, destacando o desempenho do PR2 no conjunto de benchmarks listados na Tabela 1.

O PR2 foi desenvolvido utilizando o planejador Fast Downward (HELMERT, 2006), descrito na Seção 2.2, como base para sua arquitetura. Esse sistema auxilia na criação de planos ao dividir os problemas em partes menores e mais gerenciáveis. Além disso, os pesquisadores incorporaram componentes do planejador PRP (MUISE; MCILRAITH; BECK, 2012), como a estratégia de tradução de problemas e alguns scripts de suporte, para complementar o desenvolvimento do PR2. Essa abordagem de reaproveitamento de técnicas e ferramentas não é incomum no planejamento FOND. Outros planejadores, como MyND (MATTMüLLER et al., 2010) e FONDSAT (GEFFNER; GEFFNER, 2018), também adotam métodos semelhantes para interpretar e resolver os problemas de forma eficiente.

O PR2 foi comparado com outros quatro planejadores conhecidos: MyND, FONDSAT, PRP e Paladinus (PEREIRA et al., 2022). Cada um desses planejadores foi configurado da melhor maneira possível para garantir que a comparação fosse justa. Por exemplo, o FONDSAT usou um solucionador SAT moderno, que é uma ferramenta que ajuda a resolver problemas lógicos, para melhorar seu desempenho. Todos os planejadores receberam a mesma quantidade de memória de computador (4 GB) e tempo (60 minutos) para resolver cada problema.

Os testes foram executados em um computador chamado PowerEdge C6420, que usa um processador Intel 5218 e roda no Ubuntu, um sistema operacional Linux. O conjunto de benchmarks usados incluiu testes com 18 tipos diferentes de domínios. Cada domínio com um número diferente de problemas para resolver, variando de 8 problemas no menor domínio (acrobatics) a 190 problemas no maior domínio, apresentados na Tabela 1.

Como o número de problemas em cada domínio é diferente, os pesquisadores normalizam os resultados. Isso significa que eles ajustaram os resultados para que cada domínio pudesse ser comparado de forma justa, definindo a cobertura máxima possível para cada domínio como 1. Dessa forma, identificaram facilmente qual planejador resolveu mais problemas em cada domínio, independentemente de quantos problemas houvesse no total, que é o conceito de cobertura.

O desenvolvimento da plataforma representou uma contribuição para a avaliação de planejadores FOND, especialmente considerando a pouca atenção que essa categoria recebe em competições internacionais. Com a disponibilização do código em um repositório aberto, a plataforma estabelece uma base para a realização de novos experimentos, permitindo comparações detalhadas entre diferentes planejadores. Ao incorporar métricas como cobertura, tamanho das soluções e tempo de solução, foi possível realizar a análise do desempenho do planejador PR2 em benchmarks reconhecidos. Essa abordagem facilita a avaliação objetiva e promove a construção de um arcabouço consistente para o desenvolvimento e aprimoramento de planejadores FOND.

# 3 PLANALYZING: ESTENDENDO E FLEXIBILIZANDO A PLATAFORMA DO PRP2

A plataforma *Planalyzing* foi projetada para automatizar a avaliação empírica de planejadores em um conjunto de benchmarks amplamente reconhecidos. Ela ajuda na padronização e automação do processo de avaliação, permitindo a inserção de novos domínios e problemas de planejamento, o agendamento, a execução e a análise dos experimentos de forma reprodutível. Ao automatizar essas etapas, a plataforma elimina tarefas repetitivas, reduz esforços manuais e garante consistência na comparação de desempenho entre diferentes planejadores.

A avaliação de planejadores automatizados enfrenta desafios, principalmente devido à complexidade envolvida na escolha e no gerenciamento dos benchmarks utilizados nos experimentos. Esses benchmarks, definidos como o conjunto de domínios comumente usados em experimentos de planejamento automatizado, precisam ser consistentes para refletir a variedade e a complexidade dos problemas do mundo real. Além disso, a avaliação se torna ainda mais desafiadora quando novas abordagens de planejamento são utilizadas, exigindo a aplicação de métricas que possam capturar o desempenho de forma precisa, válida e reprodutível.

Uma das principais dificuldades está no relacionamento eficaz entre os diversos elementos necessários para a execução de experimentos de avaliação. Isso inclui a seleção de planejadores, domínios e problemas, além da coleta e padronização das métricas fornecidas pelos planejadores, que muitas vezes aparecem em formatos diferentes. Outro ponto crítico é a necessidade de combinar, em um único experimento, vários planejadores com diferentes domínios e problemas, garantindo que os resultados possam ser gerenciados e comparados de maneira clara e consistente. Algumas métricas coletadas podem ser obtidas pelo próprio Planalyzing, como, por exemplo, o tempo total que o planejador levou para finalizar sua execução. Porém, outras métricas, como o tamanho da política  isto é, a quantidade de pares estado-ação  são fornecidas pelos planejadores. Cada planejador pode fornecer essa informação de uma forma específica. Nesse caso, a plataforma permite configurar uma expressão regular que capturará a informação desejada a partir da saída fornecida pelo planejador.

As principais funcionalidades da *Planalyzing* são:

- Seleção automática de planejadores, domínios e problemas para experimentos;
- Padronização e coleta de métricas em diferentes formatos;
- Combinação flexível de planejadores, domínios e problemas em um único experimento;
- Gerenciamento de histórico de avaliações para rastreamento de resultados;
- Comparação visual e tabular entre experimentos, com geração de gráficos detalhados.

Com essas funcionalidades, a *Planalyzing* estabelece uma solução estruturada para superar as dificuldades na avaliação empírica de planejadores, promovendo reprodutibilidade e confiabilidade nos resultados.

## 3.1 Objetivo

A plataforma tem como objetivo principal reduzir a complexidade envolvida na análise empírica de planejadores automatizados, integrando funcionalidades, processos, recursos e benchmarks para automatizar os experimentos. Por meio de uma arquitetura distribuída que integra um banco de dados de documentos, ambientes virtuais baseados em contêineres e uma plataforma web intuitiva, a plataforma garante a execução reprodutível de experimentos em ambientes controlados. Dentre os objetivos específicos, destacam-se:

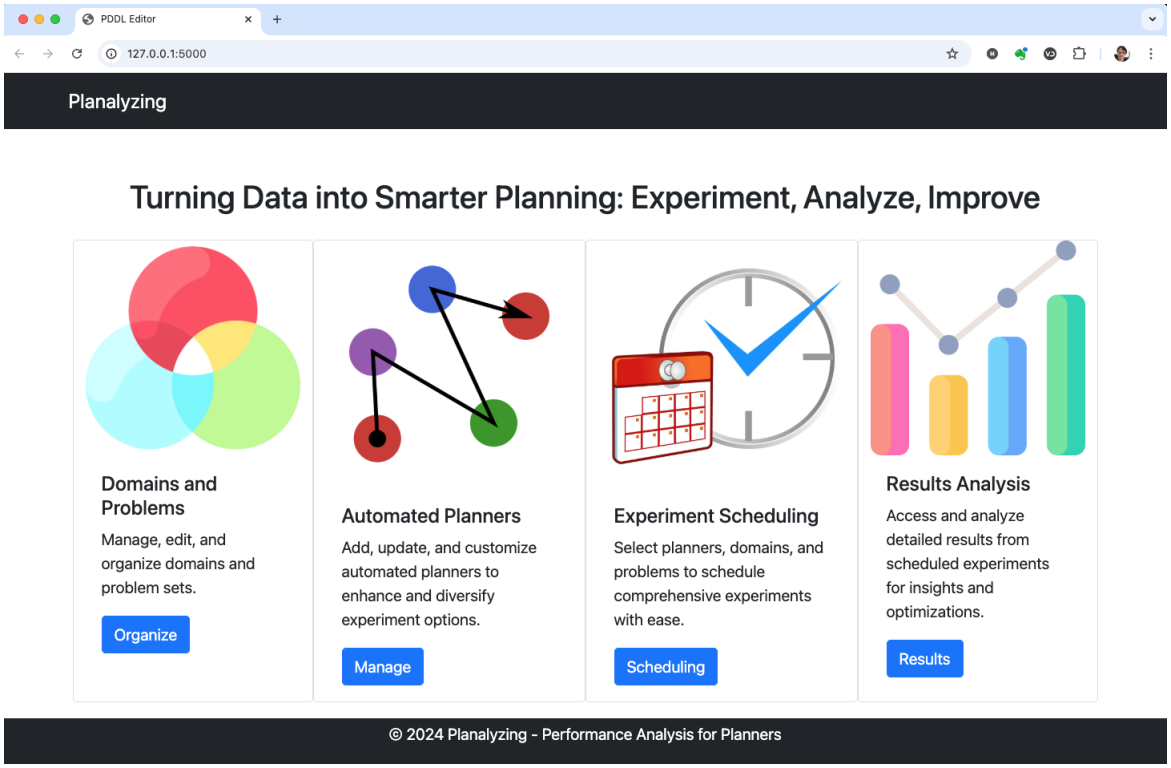
- Controle do tempo máximo de execução: Limita o tempo permitido para cada planejador resolver um problema, garantindo a consistência dos experimentos;
- Controle do uso de memória: Define o tamanho máximo de memória alocada para cada execução, prevenindo falhas ou sobrecargas nos recursos do sistema;
- Execução isolada em ambientes virtualizados: Utiliza contêineres Docker para criar ambientes controlados e isolados, minimizando riscos de conflitos entre softwares e assegurando a reprodutibilidade;
- Coleta e processamento automatizado de métricas: Garante que as métricas de desempenho sejam extraídas, processadas e armazenadas de forma padronizada;
- Geração de relatórios analíticos detalhados: Produz relatórios completos que apresentam os resultados de forma clara e estruturada, facilitando a interpretação e comparação dos experimentos.

Para executar os experimentos, serão utilizados contêineres Docker, proporcionando compatibilidade com diferentes ambientes e reduzindo os riscos de conflitos entre softwares. Essa abordagem também aumenta a segurança em ambientes compartilhados, ao isolar cada experimento, simplificando a operação em *clusters* de computação.

## 3.2 Apresentação da plataforma

A plataforma desenvolvida em interface web Figura 3, facilita a organização e gerenciamento de domínios, problemas e a avaliação comparativa entre planejadores automatizados. Com uma interface intuitiva, os usuários podem gerenciar, editar e estruturar conjuntos de domínios e problemas de forma centralizada.

Figura 3 – Página Inicial da Plataforma Planalyzing




O gerenciamento de planejadores automatizados possibilita adicionar, atualizar e personalizar planejadores. Essa flexibilidade aumenta as opções experimentais, incluindo novos planejadores com ajuste de parâmetros para atender a requisitos específicos em diferentes cenários.

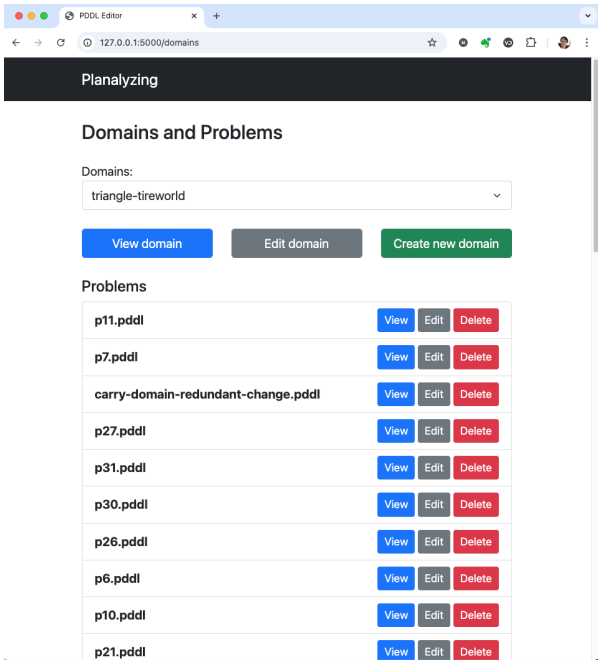
O módulo de agendamento de experimentos relaciona planejadores, domínios e problemas para criar testes personalizados, eliminando a necessidade de intervenções manuais complexas, permitindo uma execução estruturada de diferentes experimentos, melhorando o tempo de preparação e execução de testes.

A funcionalidade de análise de resultados oferece uma visão detalhada e visual dos experimentos realizados. Os usuários podem acessar tabelas e gráficos que destacam o desempenho dos planejadores em relação a métricas como tempo de execução e tamanho das políticas geradas. Essa abordagem analítica fornece informações valiosas para otimizações e comparação entre planejadores.

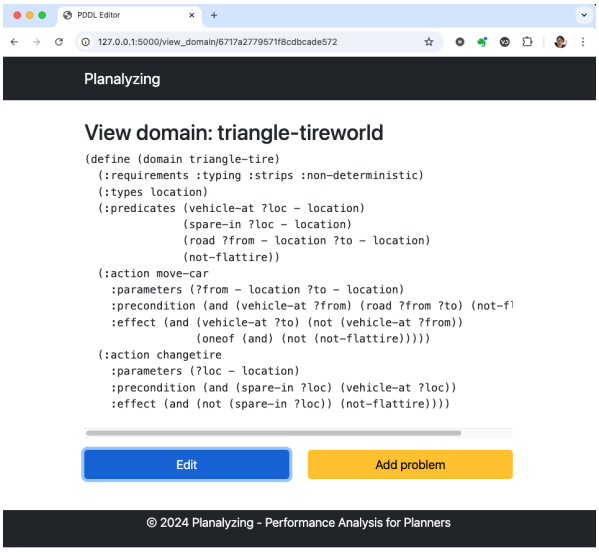
A seguir detalharemos as etapas necessárias para configurar um experimento e recuperar os resultados gerados por cada planejador nos problemas selecionados.

### 3.2.1 Configuração de Domínios e Problemas

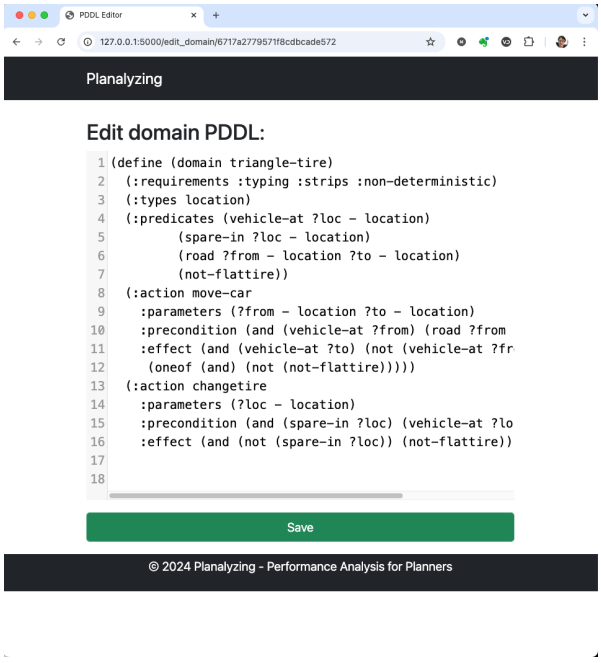
Figura 4 – Configuração de Domínios e Problemas



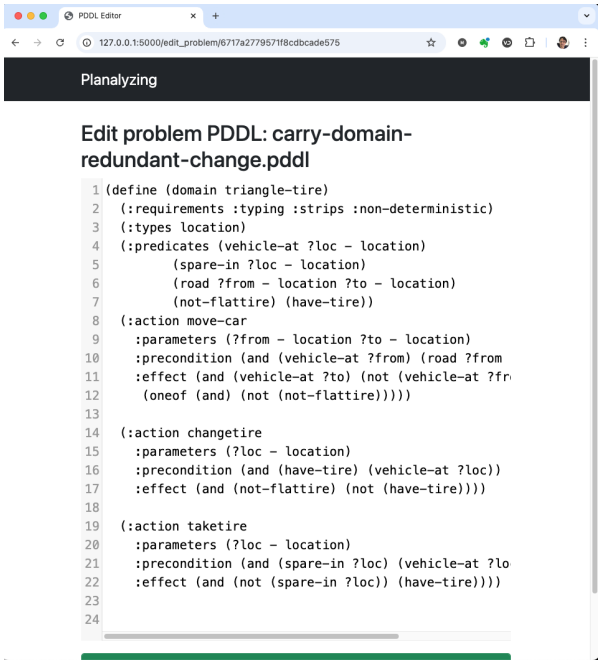

(a) Lista de domínios




(b) Detalhe do domínio




(c) Edição de domínio

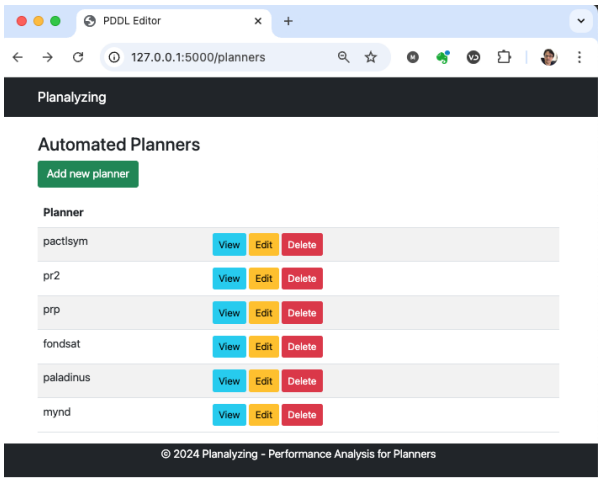



(d) Edição de problema

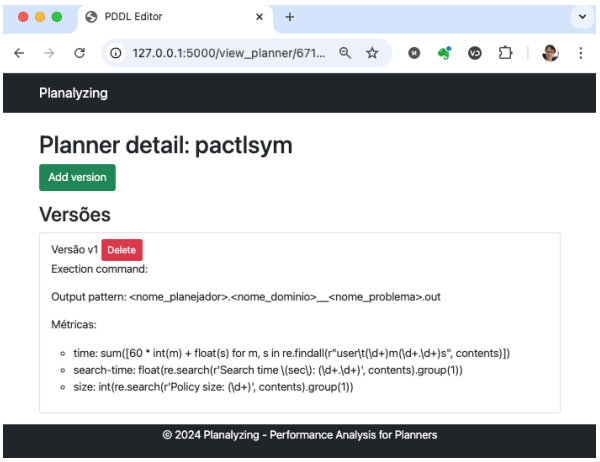
A Figura 4 apresenta as funcionalidades relacionadas à gestão de domínios e problemas no formato PDDL. A interface permite ao usuário visualizar e gerenciar domínios existentes, incluindo a listagem e seleção de problemas associados. É possível editar tanto os domínios quanto os problemas, modificando suas definições, como tipos, parâmetros, precondições e efeitos diretamente em um editor de texto integrado. A plataforma também oferece funciona-

lidades para adicionar novos domínios ou problemas, bem como excluir os existentes. Essas operações permitem ao usuário manter e ajustar facilmente os arquivos PDDL, garantindo uma gestão eficiente das configurações para análise e experimentação de planejadores.

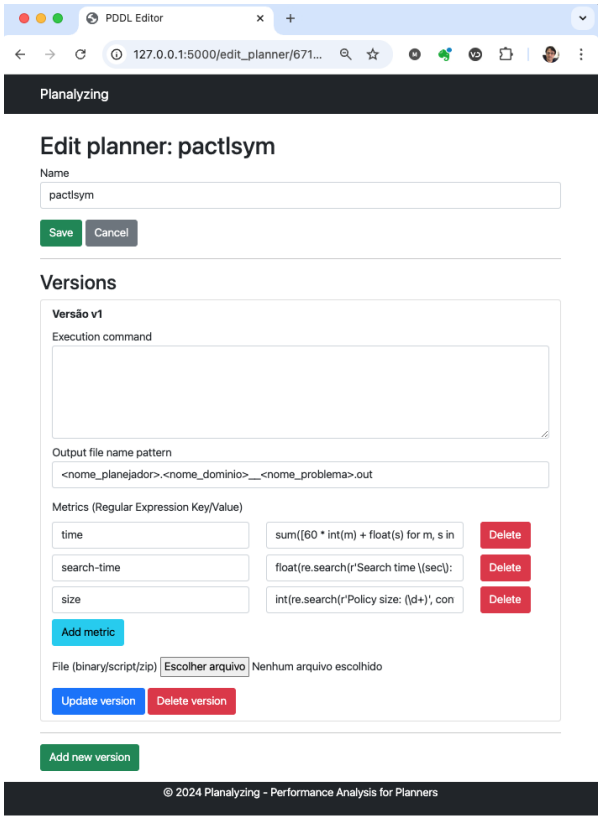
Figura 5 – Configuração dos Planejadores



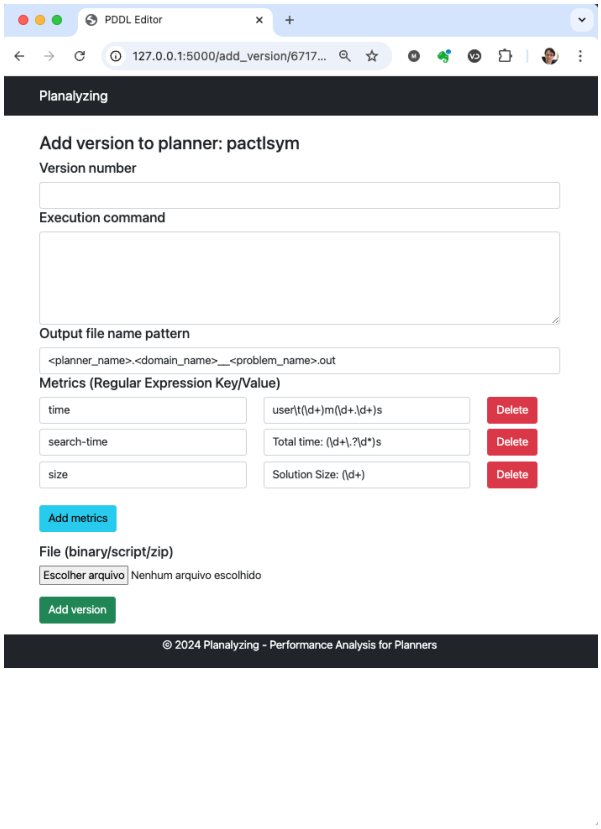

(a) Lista de planejadores




(b) Detalhe de planejadores




(c) Edição de planejadores



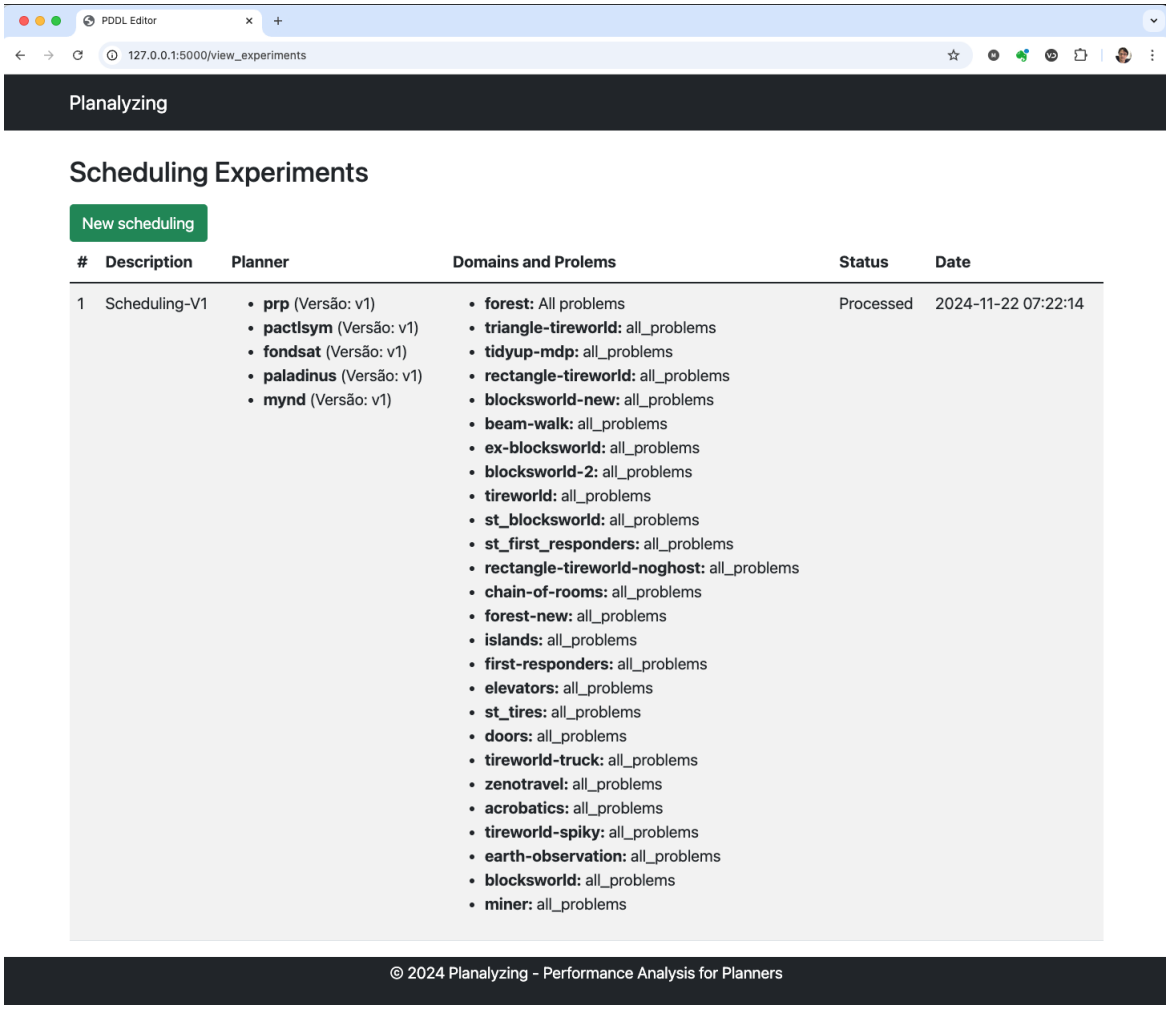

(d) Nova versão de planejadores

A Figure 5 apresenta as funcionalidades de configuração dos planejadores. A tela (a) exibe a lista de planejadores cadastrados, permitindo visualizar, editar ou excluir cada um deles, além de adicionar novos planejadores. A tela (b) detalha um planejador específico, exibindo suas versões, comandos de execução, padrão de nomeação dos arquivos e métricas extraídas

usando expressões regulares. Na tela (c), é possível editar um planejador, ajustando comandos, arquivos de saída e métricas associadas. A tela (d) permite adicionar uma nova versão ao planejador, definindo o número da versão, comandos, padrões de arquivo, métricas e arquivos binários ou scripts necessários.

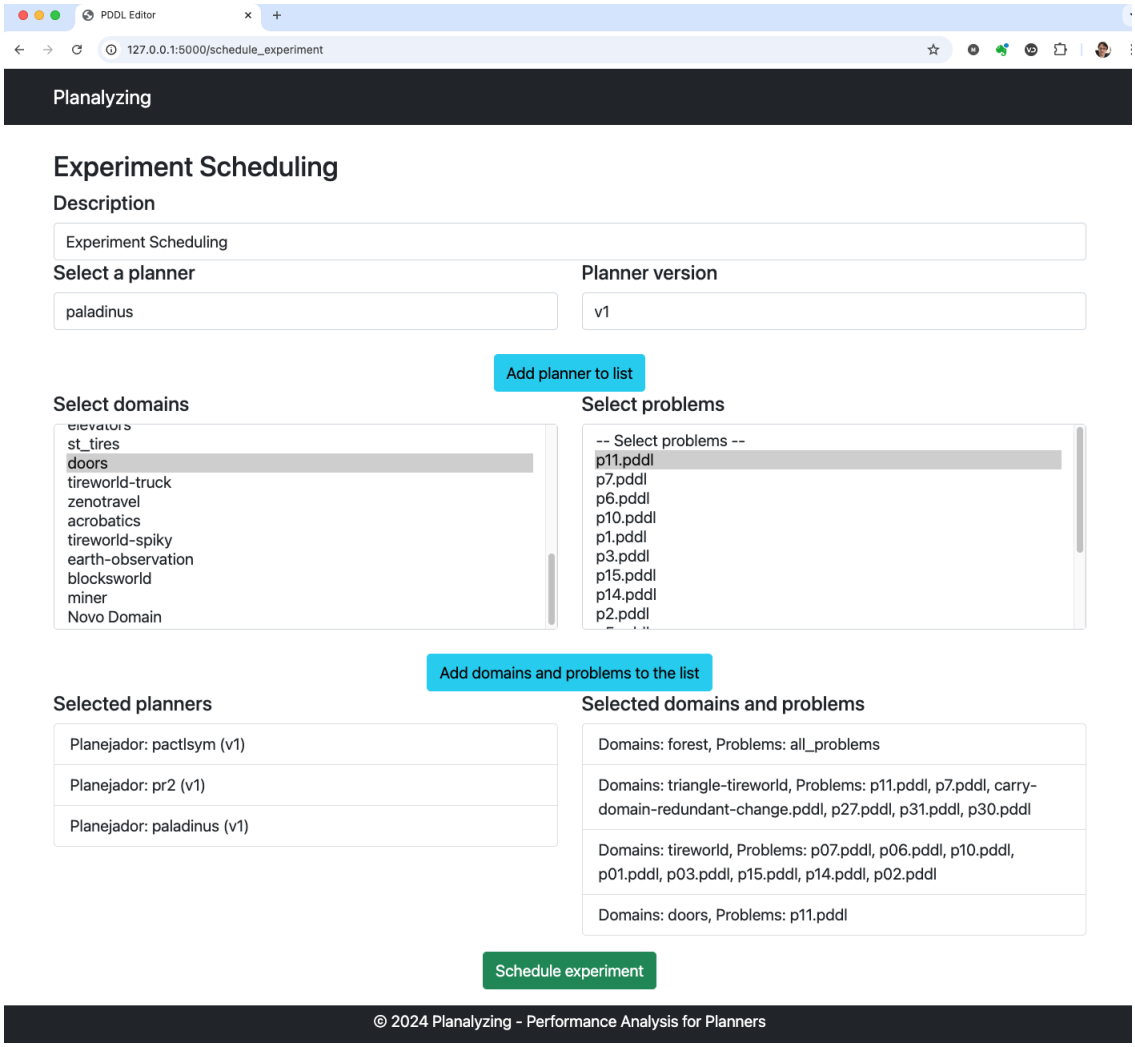
### 3.2.2 Agendamento de Experimentos

Figura 6 – Lista de Experimentos Agendados




A Figura 6 apresenta a lista de experimentos agendados , permitindo o acompanhamento detalhado de cada execução. A interface exibe informações organizadas em colunas, incluindo a descrição do experimento, planejadores utilizados com suas versões específicas, domínios e problemas associados, status de processamento e a data de execução. O experimento listado, Scheduling-V1, envolve múltiplos planejadores e abrange diversos domínios, nos quais todos os problemas foram incluídos. O botão New scheduling oferece a possibilidade de agendar novos experimentos, garantindo a flexibilidade e continuidade das análises. Essa funcionalidade centraliza a organização e o gerenciamento de experimentos, facilitando sua rastreabilidade e execução.

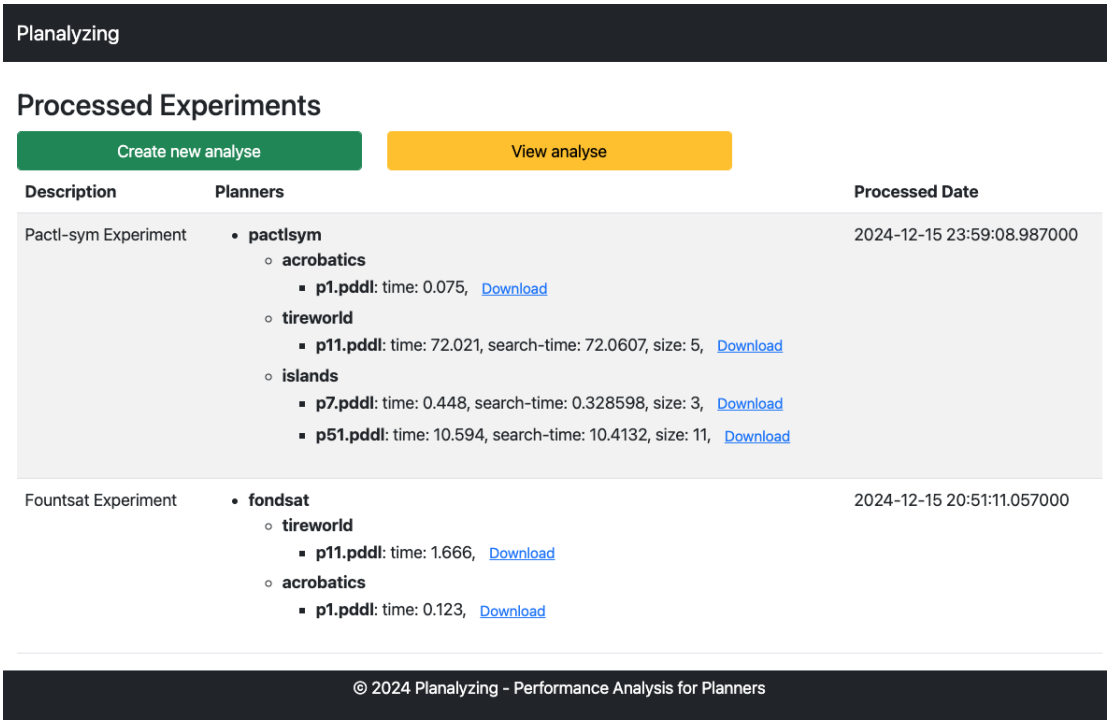
Figura 7 – Agendamento de Novo Experimento




A Figura 7 apresenta a funcionalidade de agendamento de novos experimentos . O usuário pode fornecer uma descrição para o experimento e selecionar um planejador, especificando a versão correspondente. Em seguida, é possível escolher domínios e problemas associados por meio de listas interativas. Após a seleção, os planejadores são adicionados à lista clicando em Add planner to list, enquanto os domínios e problemas são incluídos com o botão Add domains and problems to the list. A seção Selected planners e Selected domains and problems exibe as configurações adicionadas. Por fim, o botão Schedule experiment finaliza o agendamento, consolidando todas as informações configuradas.

### 3.2.3 Experimentos Processados

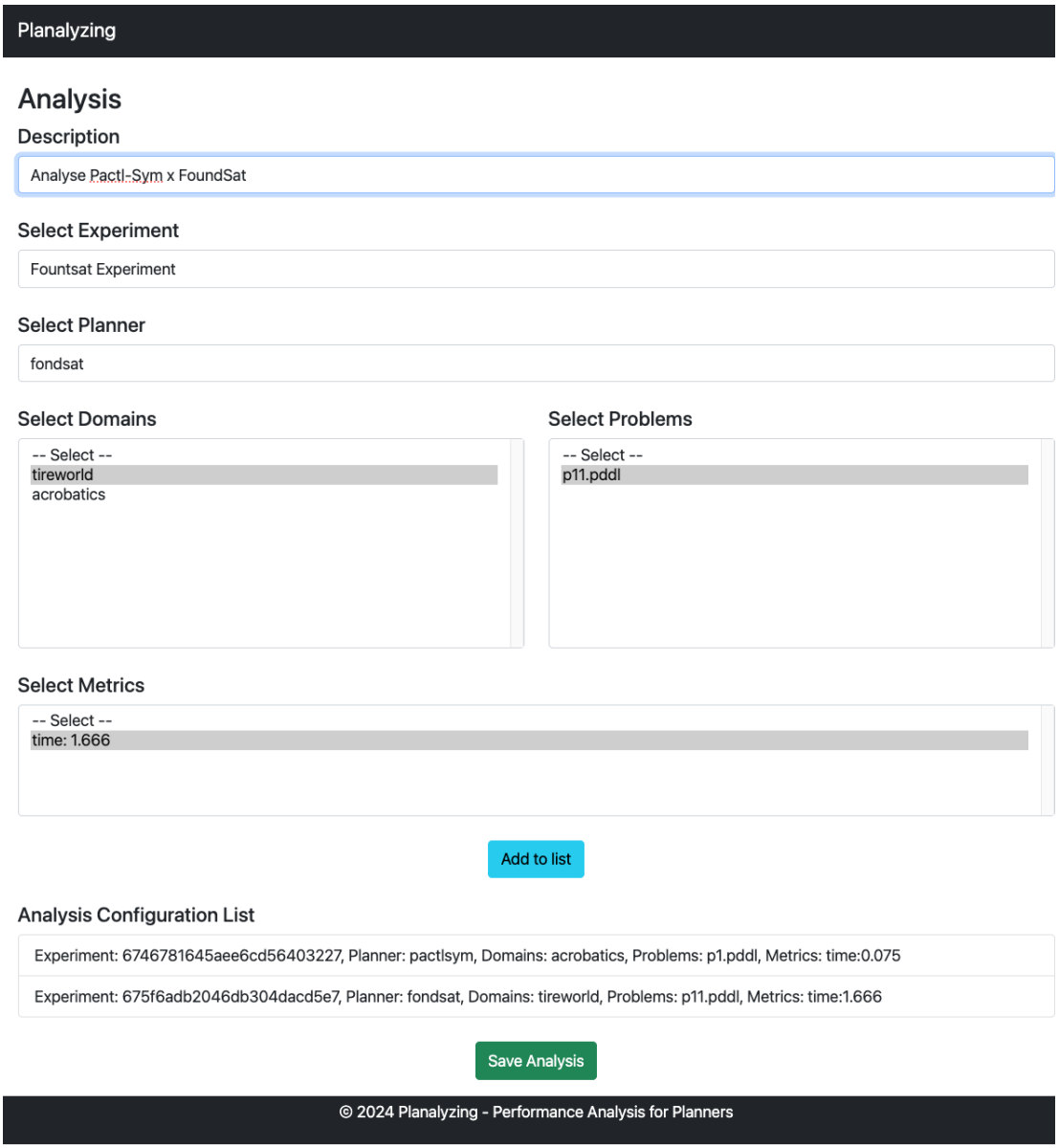
Figura 8 – Experimentos Processados




A Figura 8 apresenta a tela de experimentos processados . A interface lista os experimentos concluídos com suas respectivas descrições, planejadores utilizados, domínios, problemas processados e métricas registradas, como tempo de execução (time), tempo de busca (search-time) e tamanho (size). Para cada problema, há uma opção de Download para os resultados processados. A tabela inclui a data de processamento dos experimentos, facilitando a organização e rastreabilidade das execuções. Os botões Create new analyse e View analyse permitem iniciar uma nova análise ou visualizar análises já realizadas. Essa funcionalidade centraliza os resultados, tornando o gerenciamento e a análise de experimentos mais eficientes.

### 3.2.4 Gerando Analises

Figura 9 – Gerando Nova Analise



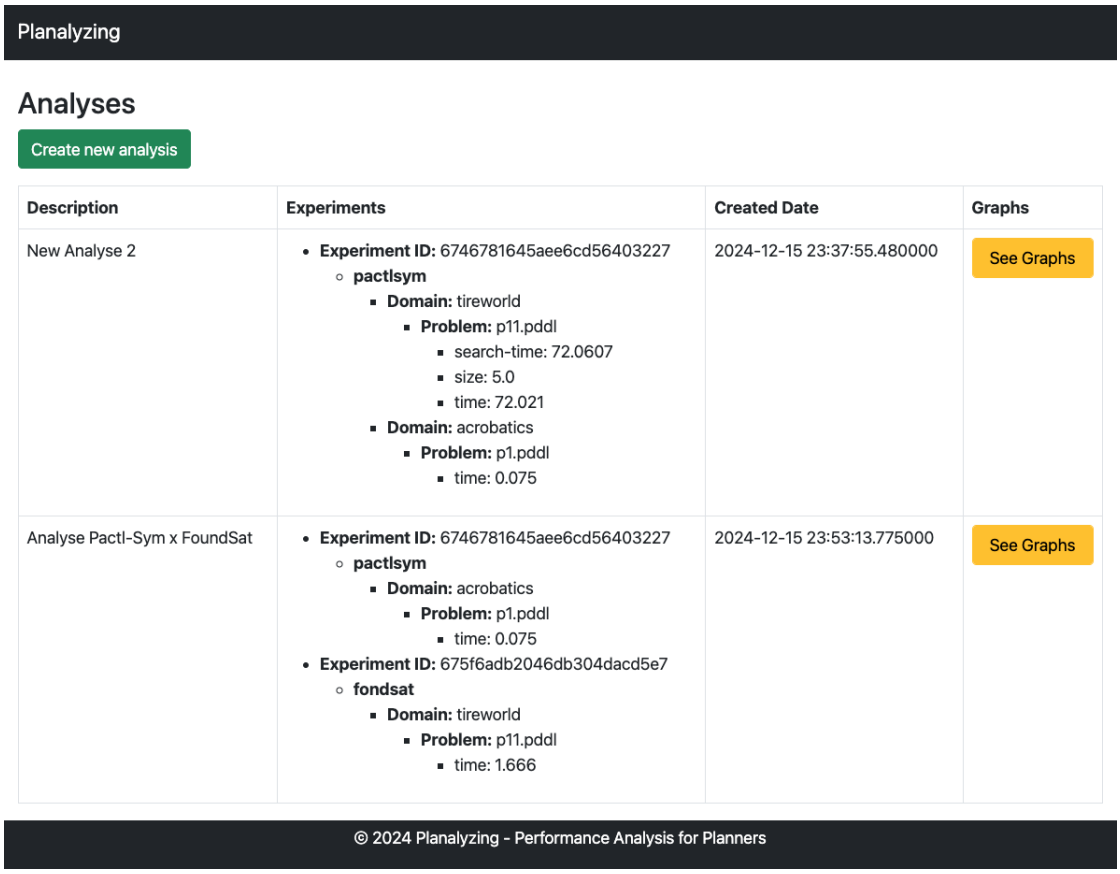

A Figura 9, mostra a tela para criar uma análise, o usuário deve fornecer uma descrição no campo correspondente para identificar os experimentos que serão comparados. Em seguida, deve selecionar o experimento desejado no campo Select Experiment, o que fará com que o sistema carregue automaticamente os planejadores relacionados e os resultados coletados. Após isso, o usuário escolhe um planejador na lista Select Planner, preenchendo o campo Select Domains com os domínios disponíveis. Com os domínios selecionados, são exibidos os problemas associados na lista Select Problems, e as métricas coletadas das saídas dos planejadores são exibidas no campo Select Metrics.

Depois de configurar todas as seleções, basta clicar no botão Add to List para incluir a configuração atual na lista de análise, onde são exibidos os detalhes do experimento, planejador,

domínios, problemas e métricas na seção Analysis Configuration List. Para finalizar, basta clicar em Save Analysis para salvar a análise na base de dados.

### 3.2.5 Lista de Análises

Figura 10 – Lista de Análises




A Figura 10 exibe a lista de análises registradas. Cada análise inclui uma descrição, os experimentos associados, a data de criação e uma opção para visualizar gráficos por meio do botão See Graphs. Os experimentos são detalhados com o ID correspondente, planejadores utilizados, domínios processados, problemas específicos e métricas obtidas, como tempo de execução (time), tempo de busca (search-time) e tamanho (size). A interface organiza as informações de forma clara, facilitando o acompanhamento e comparação dos resultados obtidos. A funcionalidade Create new analysis permite iniciar novas análises, contribuindo para a continuidade do processo de avaliação e interpretação dos experimentos.

## 3.3 Metodologia

Para modelar o sistema, usamos a linguagem C4. O Modelo C4 é uma abordagem estruturada para a visualização de arquiteturas de sistemas de software. Ele se concentra em oferecer uma visão estática do sistema, destacando tanto a estrutura quanto a interação entre seus diferentes componentes.

Essa abordagem é organizada em quatro níveis hierárquicos de abstração, cada um capturado por um tipo de diagrama específico:

1. Diagrama de Contexto: fornece uma visão geral do sistema, destacando como ele se relaciona com os usuários e outros sistemas externos.
2. Diagrama de Contêiner: detalha os principais contêineres de software do sistema (por exemplo, aplicações, serviços e bancos de dados) e suas interações.
3. Diagrama de Componente: foca na estrutura interna de cada contêiner, descrevendo os componentes e suas interdependências.
4. Diagrama de Código: apresenta os detalhes mais granulares da implementação, como classes, métodos e funções, quando necessário.

Ao decompor a arquitetura em diferentes níveis de abstração, o modelo C4 permite uma comunicação clara entre as partes interessadas, desde desenvolvedores até gerentes de projeto e outros membros não técnicos da equipe.

Para avaliar a plataforma, será submetido à análise o planejador PACTL-Sym (SANTOS et al., 2022), um planejador simbólico, que implementa a semântica da lógica $\alpha$-CTL para planejamento totalmente observável não-determinístico (FOND). Esse planejador, usado como referência para avaliar a plataforma, foi desenvolvido por alunos de pós-graduação do curso de Ciência da Computação da Universidade de São Paulo.

## 3.4 Modelagem da plataforma

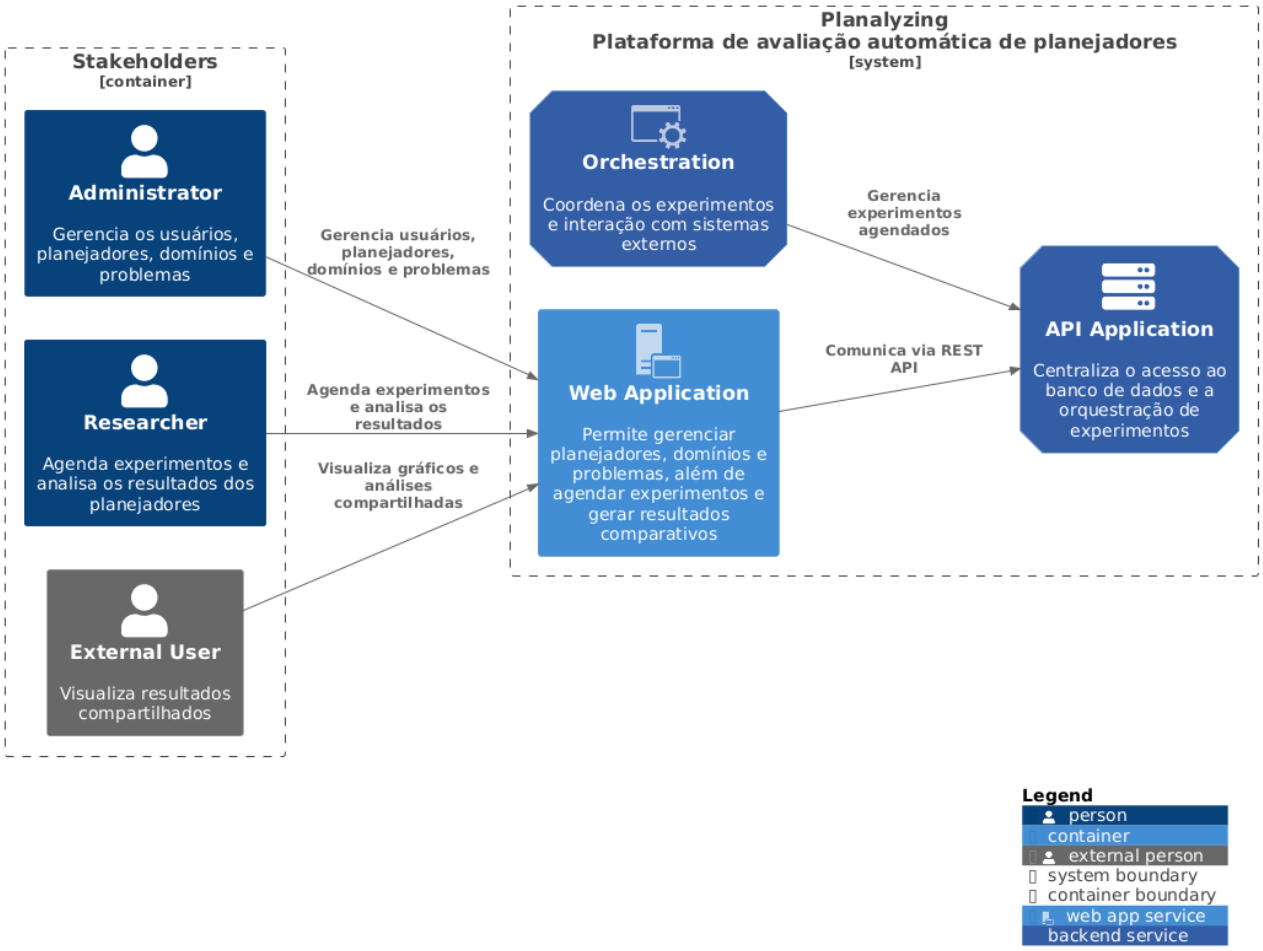
A seguir, é apresentada a modelagem da plataforma baseada na linguagem C4.

### 3.4.1 Contexto do Sistema

O diagrama de contexto representa a visão de mais alto nível de um sistema de software, oferecendo uma perspectiva de como o sistema se integra ao ecossistema em que está inserido. Nesse diagrama, o sistema é retratado como uma única entidade (ou caixa), cercada por seus usuários, também chamados de atores, e pelos outros sistemas com os quais interage.

Essa visualização serve para destacar os principais relacionamentos e fluxos de informação entre o sistema e seu ambiente externo. Como ponto de partida, o diagrama de contexto estabelece uma visão geral fundamental, preparando o terreno para os diagramas subsequentes, que exploram os detalhes internos do sistema em níveis mais granulares. A Figura 11, mostra o contexto da plataforma.

Figura 11 – Diagrama de Contexto da plataforma *Planalyzing*




O diagrama apresentado na Figura 11, ilustra como o sistema interage com seus principais atores (*stakeholders*) e seus componentes internos. A plataforma é representada como um único sistema composto por diferentes contêineres que desempenham papéis específicos.

- Stakeholders (Usuários)
  1. *Administrator*: Responsável por gerenciar usuários, planejadores, domínios e problemas no sistema.
  2. *Researcher*: Agenda experimentos, analisa resultados dos planejadores e visualiza gráficos e análises compartilhadas.
  3. *External User*: Visualiza os resultados compartilhados gerados pelos pesquisadores.
- Plataforma *Planalyzing* (Componentes do Sistema)
  1. *Orchestration*: Coordena os experimentos e gerencia interações com sistemas externos, garantindo que os experimentos agendados sejam processados de forma eficiente.
  2. *Web Application*: Serve como interface principal para gerenciar planejadores, domínios e problemas. Permite aos usuários agendar experimentos e visualizar resultados comparativos, funcionando como um ponto de interação para os administradores e

pesquisadores.

3. *API Application*: Centraliza o acesso ao banco de dados e gerencia a orquestração dos experimentos, comunicando-se via *REST API* com outros componentes.

A seguir são descritos os fluxos de integração entre os contêineres apresentados anteriormente:

- *Administrator*: Interage diretamente com a *Web Application* para gerenciar usuários, planejadores, domínios e problemas, populando os dados básicos que serão utilizados no agendamento e execução dos experimentos.
- *Researcher*: Também utiliza a *Web Application* para agendar experimentos, analisar resultados e visualizar gráficos.
- *Orchestration*: Responsável por coordenar a execução dos experimentos agendados por meio da aplicação web. Esse componente se comunica diretamente com a *API Application* para acessar o banco de dados e obter as informações necessárias para a execução dos experimentos. Além disso, estabelece conexões com outros sistemas externos, garantindo a integração e o fluxo de informações para o funcionamento da plataforma.
- *External User*: Acessa resultados compartilhados, que são disponibilizados através da interface Web.

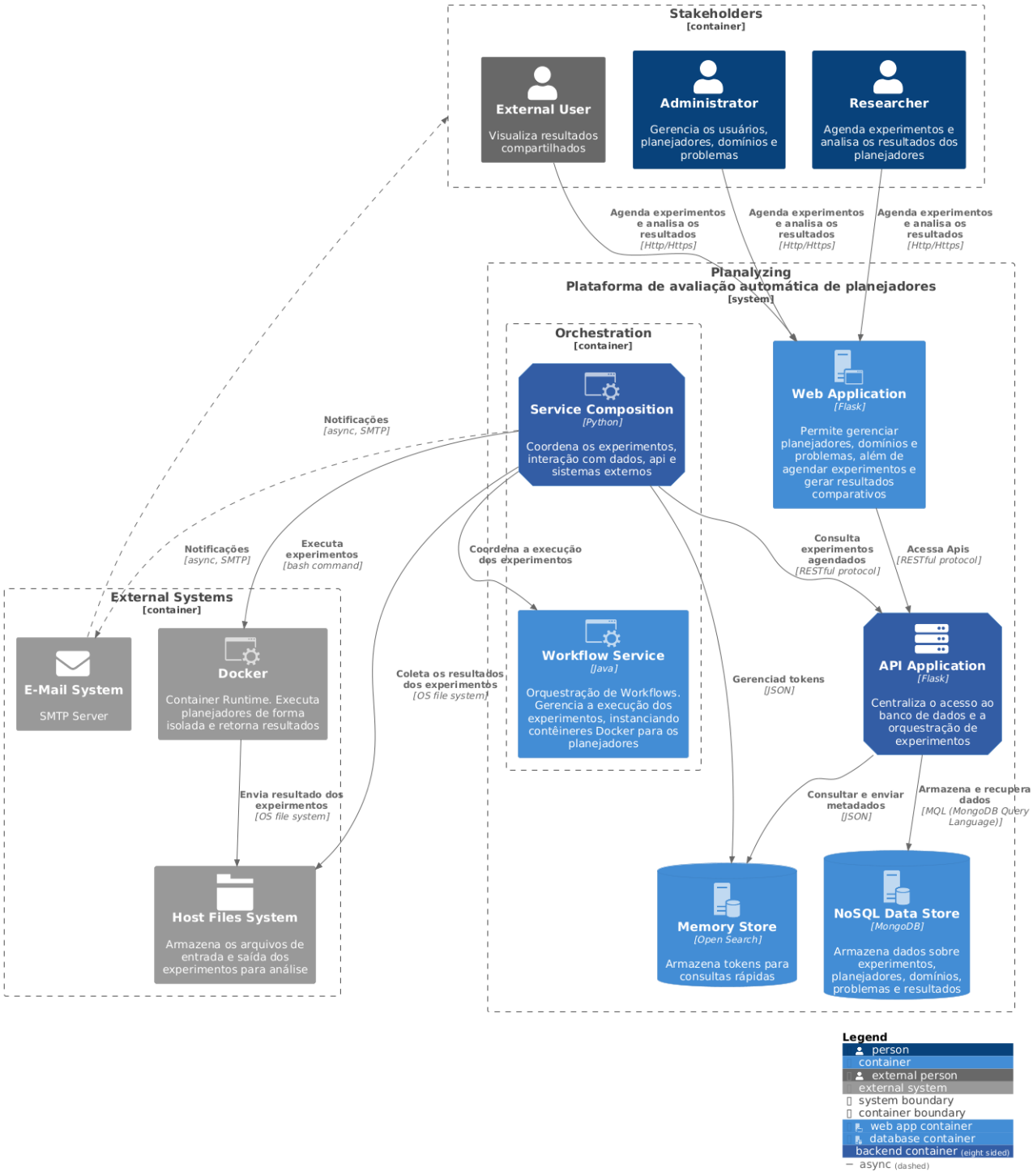
### 3.4.2 Diagrama de Contêiner

O Diagrama de contêiner oferece uma visão mais detalhada da arquitetura do sistema, mantendo um nível de abstração suficiente para ser acessível e facilmente compreendido por partes interessadas não técnicas. Ele destaca os principais contêineres que compõem o sistema e suas interações, possibilitando uma visão clara e simplificada da estrutura geral. A Figura 12, descreve a arquitetura do sistema que é composta por diversos contêineres, cada um desempenhando funções específicas e interdependentes, formando um ecossistema integrado para a avaliação de planejadores automatizados.

A interação entre os componentes da plataforma é mediada principalmente pela *Orchestration*, que centraliza a coordenação dos experimentos, conectando usuários, sistemas internos e externos. A *Web Application* funciona como a interface principal para os usuários, oferecendo ferramentas para gerenciar planejadores, domínios e problemas, além de agendar experimentos e visualizar resultados. Os *stakeholders*, como administradores e pesquisadores, utilizam esta aplicação para interagir com a plataforma, enquanto os usuários externos têm acesso restrito para visualizar resultados compartilhados.

A *Web Application* comunica-se diretamente com a *API Application*, que é responsável por centralizar o acesso aos dados armazenados no banco de dados e gerenciar as operações associadas, como a consulta de experimentos e o gerenciamento de *tokens* de autenticação. Essa estrutura permite que o sistema seja flexível e responsivo às necessidades dos usuários, mantendo uma clara separação entre interface, lógica de aplicação e dados.

Figura 12 – Diagrama de Contêiner da Plataforma *Planalyzing*




O *Orquestrador* é um componente central no sistema, cuja finalidade é gerenciar a execução de experimentos e coordenar as interações com outros serviços e sistemas externos. Esse contêiner está subdividido em dois módulos principais: o *Service Composition*, que controla as interações de alto nível, e o *Workflow Service*, que orquestra a execução de workflows e instanciamento de contêineres *Docker* para execução de planejadores.

O *Workflow Service* é responsável por garantir que os experimentos sejam realizados

de forma isolada e controlada, utilizando contêineres para executar os planejadores e coletar os resultados, que são posteriormente armazenados em sistemas de arquivos. Paralelamente, o *Service Composition* gerencia a comunicação assíncrona, enviando notificações aos usuários sobre o progresso dos experimentos via sistemas de e-mail externo. Essa segmentação permite que o sistema mantenha um fluxo de execução e comunicação, garantindo a integridade dos experimentos e a entrega de notificações aos *stakeholders*.

Os dados gerados e consumidos pelo sistema são armazenados e gerenciados por dois contêineres dedicados: o NoSQL Data Store e o Memory Store. O NoSQL Data Store, baseado em *MongoDB*, armazena informações sobre experimentos, planejadores, domínios, problemas e resultados, oferecendo uma base de dados para consultas e análise. Já o Memory Store é utilizado para armazenar dados, como pares chave/valor, otimizando a performance em operações de consulta rápidas.

Esses dois contêineres interagem diretamente com a *API Application*, que atua como um intermediário para gerenciar as requisições dos usuários e da *Orchestration*. Por fim, o sistema também depende de serviços externos, como o *E-Mail System* para envio de notificações e o *Host Files System* para armazenar entradas e saídas dos experimentos.

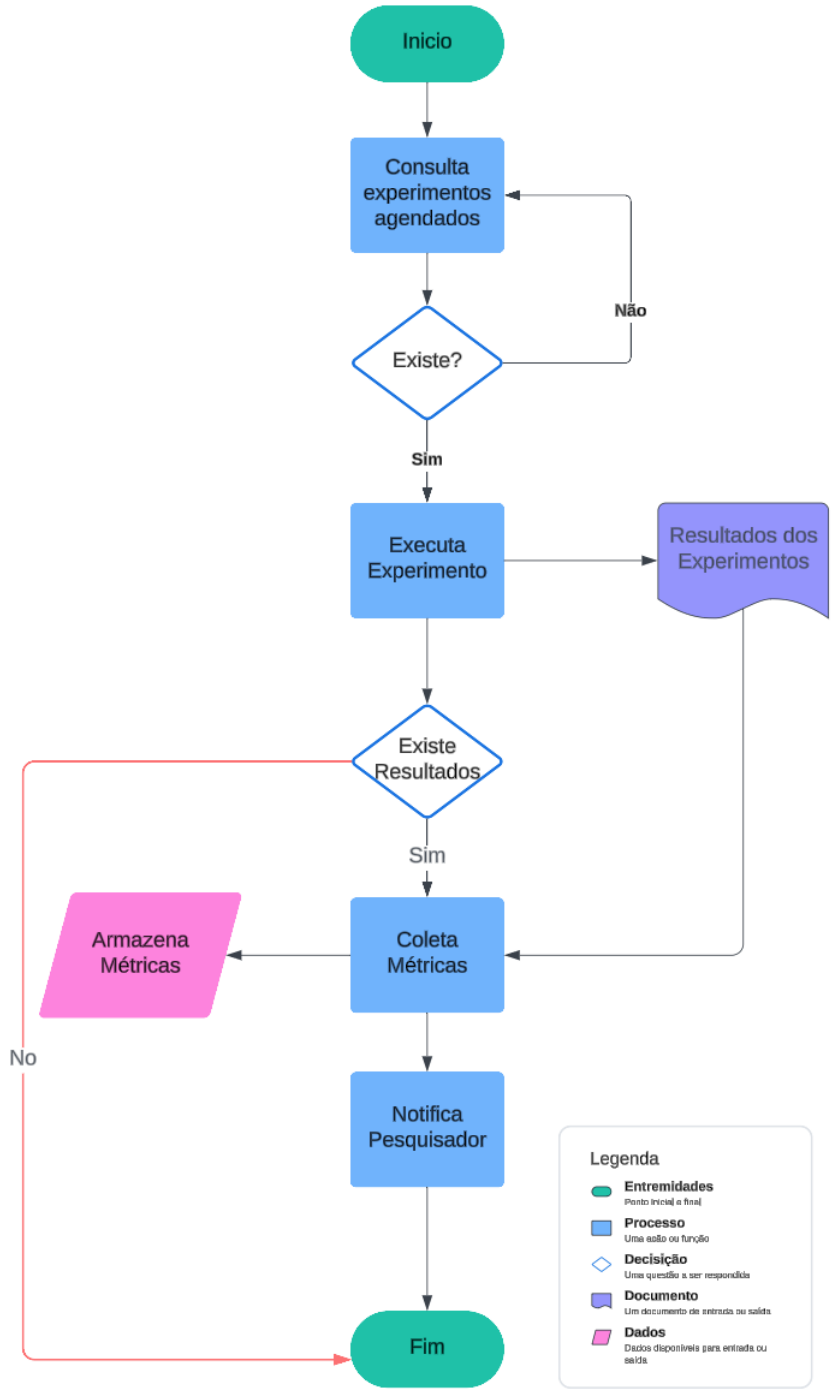
### 3.4.3 Diagrama de Componentes do Orquestrador

O Diagrama de componentes é o terceiro nível de abstração no modelo C4. Ele detalha cada contêiner, dividindo seus componentes principais e revelando suas interações e relacionamentos. Esses componentes podem ser classes, interfaces ou módulos que oferecem funcionalidades específicas. Esse nível de detalhamento é especialmente útil para os desenvolvedores, pois fornece uma visão detalhada da estrutura interna do sistema e facilita a compreensão da arquitetura.

Nessa modelagem, descreveremos em detalhes o processo de automação, mostrado na Figura 13. Cada etapa é detalhada através do diagrama de componentes do orquestrador Figura 14, que é dividido em dois módulos principais, o *Workflow Service* e o *Service Composition*, que coordenam a execução dos experimentos em diferentes níveis de granularidade.

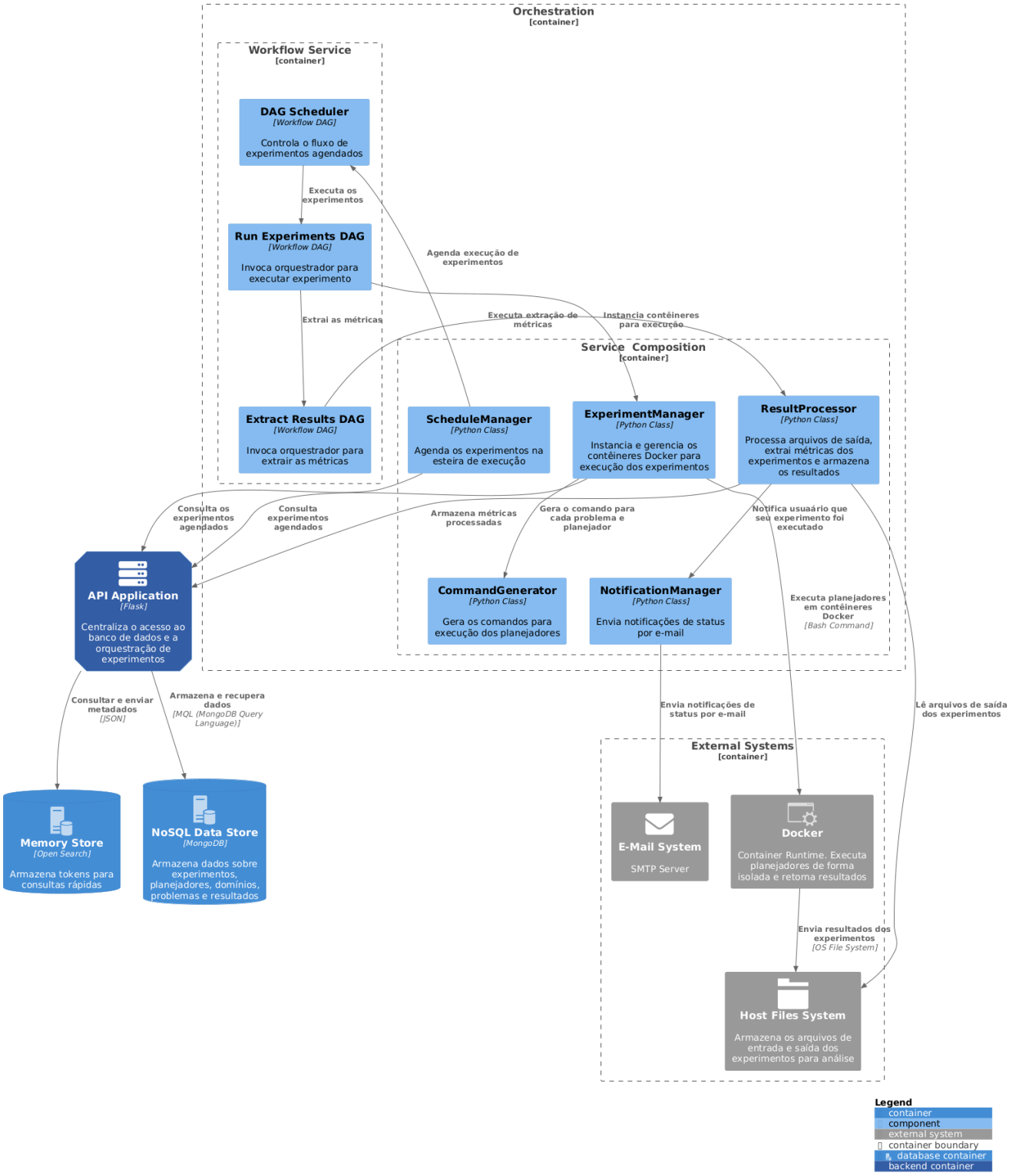
No *Workflow Service*, elementos como o *DAG Scheduler* gerenciam o fluxo dos experimentos agendados, enquanto processos mais específicos, como o *Run Experiments DAG* e o *Extract Results DAG*, são responsáveis por executar experimentos e extrair métricas. Esses componentes trabalham em conjunto para garantir que as tarefas sejam realizadas de maneira orquestrada e com integração entre os serviços internos e externos. Esse comportamento é refletido no fluxograma, que descreve as etapas sequenciais do processo, desde a consulta de experimentos agendados até a coleta e notificação dos resultados.

Figura 13 – Processo de Automação de Experimentos




No *Service Composition*, componentes como o *ScheduleManager*, *ExperimentManager* e *ResultProcessor* desempenham papéis essenciais na automação e gerenciamento das execuções. O *ScheduleManager* agenda experimentos na esteira de execução, alinhado à lógica do fluxograma, onde os experimentos são validados antes de sua execução. O *ExperimentManager*, por sua vez, instancia contêineres *Docker* para executar planejadores de forma isolada, garantindo a reprodutibilidade dos experimentos e a integridade dos dados, enquanto o *ResultProcessor* analisa os arquivos de saída, extrai métricas e as armazena no banco de dados para posterior consulta e análise.

Figura 14 – Diagrama de Componentes do Orquestrador




Paralelamente, componentes como o *CommandGenerator* e o *NotificationManager* geram comandos para execução e notificações para os usuários, garantindo que o sistema seja responsivo às interações humanas e automatize os feedbacks necessários, como mostrado no fluxograma na etapa de notificação dos resultados.

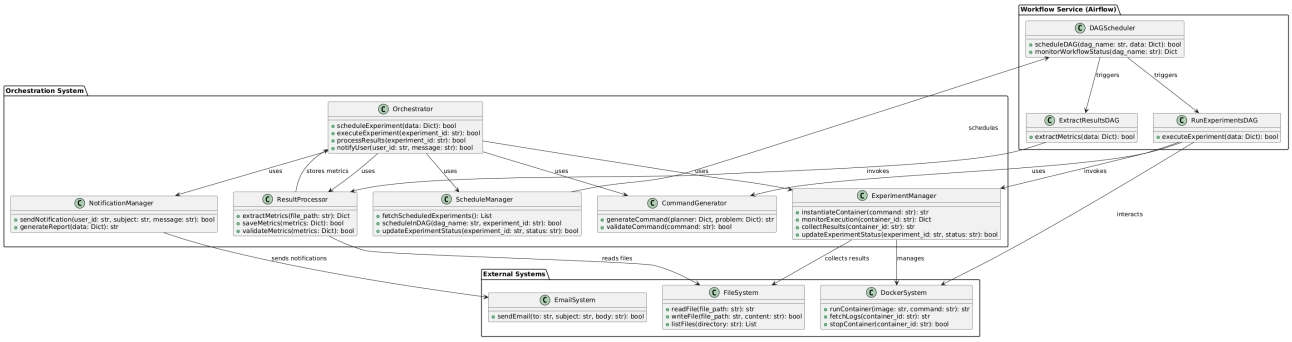
A consulta de experimentos agendados pelo *DAG Scheduler* corresponde ao início do processo no fluxograma, enquanto a execução de experimentos e extração de métricas são etapas representadas pelos componentes *Run Experiments DAG* e *Extract Results DAG*. O ciclo se fecha com a coleta de métricas, realizada pelo *ResultProcessor*, e a notificação ao pesquisador, representada no fluxograma como a etapa final antes do encerramento do processo.

A integração entre os módulos do orquestrador e os sistemas externos, como o *Docker* e o *E-Mail System*, garante que cada etapa seja realizada com total automação, garantindo a capacidade de gerenciar experimentos complexos de forma escalável e confiável.

### 3.4.4 Diagrama de Código do Orquestrador

A Figura 15, representa o nível final de abstração, mostra os detalhes de componentes individuais, classes importantes, interfaces e seus relacionamentos, normalmente usado por desenvolvedores para projetar e implementar o sistema.

Figura 15 – Diagrama de Código do Orquestrador




O orchestrator usa o *ScheduleManager* para agendar experimentos e o *Command-Generator* para criar comandos de execução. Durante a execução, o *ExperimentManager* gerencia os contêineres *Docker* e coleta os resultados. Após a execução, o *ResultProcessor* lê os resultados do sistema de arquivos, extrai métricas e salva no banco de dados. Por fim, o *NotificationManager* informa os usuários sobre o status e a conclusão do experimento.

### 3.4.5 Modelo de Dados

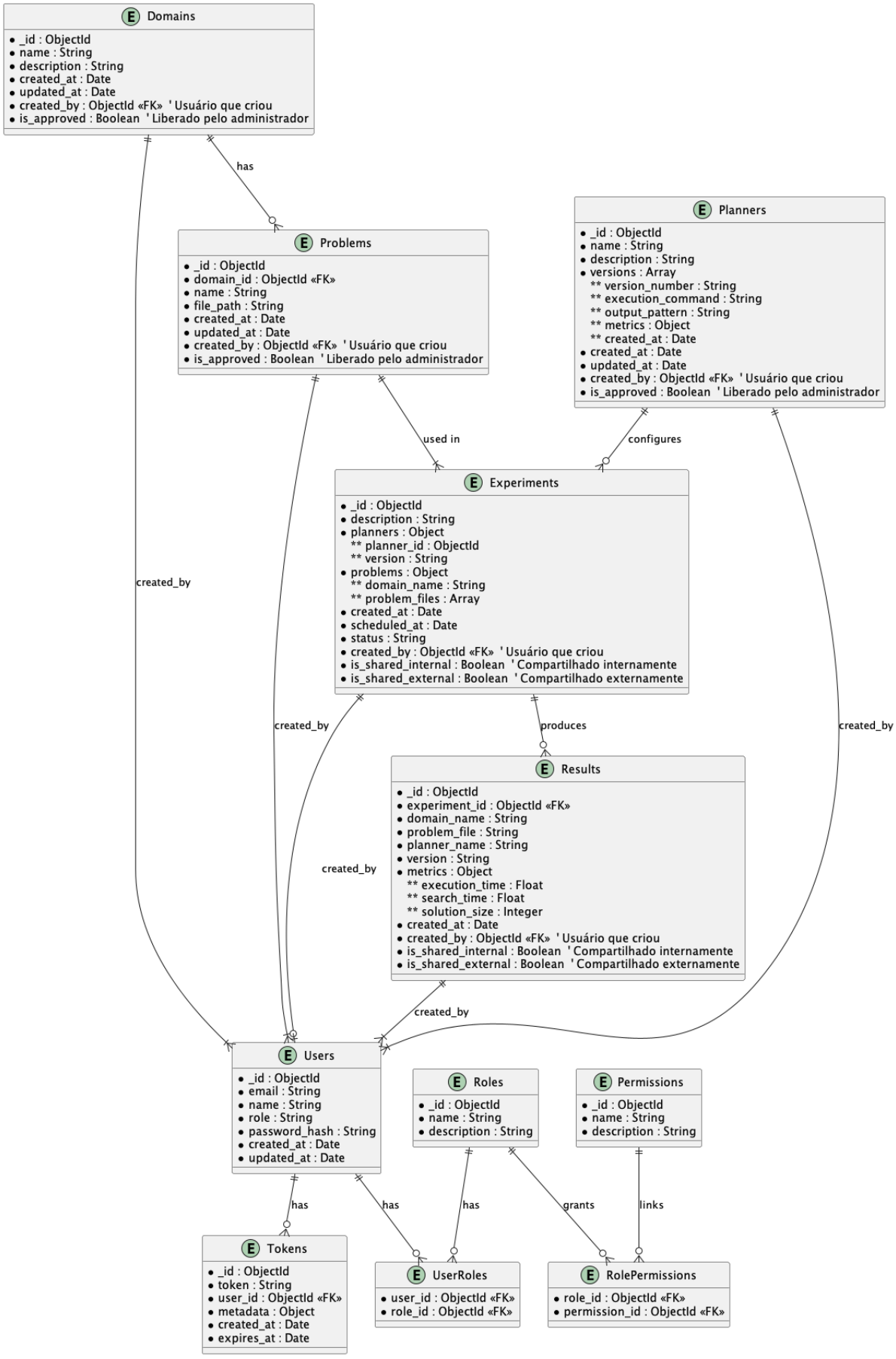
O modelo de dados da Figura 16 é estruturado em coleções no *MongoDB*, cada uma representando um conceito fundamental no sistema. A coleção `Domains` armazena os domínios de planejamento, definidos por um identificador único (_id), um nome descritivo, uma descrição detalhada, e as datas de criação e atualização. Cada domínio também inclui o campo `created_by`, que identifica o usuário que registrou o domínio, e o campo `is_approved`, que indica se o domínio foi aprovado por um administrador. Apenas domínios aprovados estão disponíveis para serem listados ou utilizados. Esses domínios servem para agrupar e organizar

os problemas de planejamento, que são armazenados na coleção `Problems`. Cada problema está associado a um domínio através do campo `domain_id` e inclui informações como o nome do problema, o caminho do arquivo que contém sua definição (`file_path`), além de metadados sobre criação e atualização. Assim como nos domínios, problemas possuem os campos `created_by` e `is_approved`, garantindo controle sobre quem os registrou e se estão disponíveis para uso.

A coleção `Planners` armazena informações sobre os planejadores disponíveis no sistema. Cada planejador possui um identificador único, um nome, uma descrição, e uma lista de versões. Cada versão inclui um número de versão (`version_number`), o comando necessário para sua execução (`execution_command`), o padrão dos arquivos de saída gerados (`output_pattern`) e as métricas que podem ser extraídas desses arquivos. Essas métricas são armazenadas como pares de nome e expressão regular, permitindo a análise dos resultados após a execução. Cada planejador também contém os campos `created_by` e `is_approved`, garantindo que apenas planejadores aprovados possam ser utilizados em experimentos. Esses planejadores são fundamentais para configurar experimentos, que são registrados na coleção `Experiments`.

A coleção `Experiments` representa os experimentos agendados e executados no sistema. Cada experimento é descrito por um identificador único, uma descrição textual, uma lista de planejadores e suas versões, e os problemas associados. Os problemas são organizados por nome do domínio e arquivos de problemas específicos ou, quando aplicável, a indicação de que todos os problemas do domínio devem ser utilizados. Adicionalmente, o experimento contém os campos `is_shared_internal` e `is_shared_external`, que controlam o compartilhamento de resultados entre usuários do sistema e usuários externos, respectivamente. O experimento também armazena informações sobre a data de criação, o agendamento (`scheduled_at`) e o status atual (como agendado ou concluído). Após a execução de um experimento, os resultados são armazenados na coleção `Results`.

Figura 16 – Diagrama de Entidade e Relacionamento - ER



A coleção `Results` armazena os dados extraídos dos experimentos. Cada resultado está associado a um experimento específico (`experiment_id`) e inclui informações como o domínio, o arquivo de problema, o nome do planejador, a versão utilizada e as métricas extraídas. Essas métricas incluem tempo de execução total (`execution_time`), tempo de busca (`search_time`) e o tamanho da solução gerada (`solution_size`) como padrão, mas podem ser estendidas. Assim como nos experimentos, os resultados possuem os campos `is_shared_internal` e `is_shared_external`, permitindo controle sobre a visibilidade dos dados. Esses resultados são a base para as análises posteriores e são acessados por usuários, que são gerenciados na coleção `Users`.

A coleção `Users` armazena informações sobre os usuários registrados no sistema. Cada usuário é identificado por seu e-mail, nome, função (como administrador, pesquisador ou usuário externo), e uma senha criptografada (`password_hash`). Os usuários podem ser associados a diferentes papéis, cujas permissões são definidas por associações na coleção `Roles` e `Permissions`. Para controle de autenticação, a coleção `Tokens` armazena tokens de acesso vinculados aos usuários. Esses tokens incluem informações como endereço IP, agente do usuário e data de expiração, permitindo autenticação segura e controle de sessões.

Esse modelo de dados suporta todo o ciclo de vida do sistema, desde o registro de domínios e problemas até a execução de experimentos e a análise de resultados, além de oferecer mecanismos de controle de acesso e compartilhamento de informações.

## 3.5 Processo de Automação

Uma das principais dificuldades na avaliação de um conjunto de planejadores em um lote de problemas, considerando um conjunto de métricas de interesse, está na seleção do(s) planejador(es), seleção do(s) domínio(s) e problema(s), associação de métricas diferentes por planejador, visualização e gerenciamento do histórico de resultados. Associados a isso, na execução dos experimentos, existe a preocupação com o controle máximo do tempo permitido por problema, limite de memória e o uso de ambientes virtuais (Docker) para criar ambientes reprodutíveis e isolados.

A plataforma Planalyzing foi projetada para automatizar a avaliação de planejadores, permitindo que os usuários realizem o agendamento, a execução e a análise de experimentos de forma reprodutível em um conjunto de domínios e problemas conforme o processo descrito na Figura 17. Essa abordagem otimiza o processo de avaliação, eliminando tarefas repetitivas e reduzindo os esforços manuais.

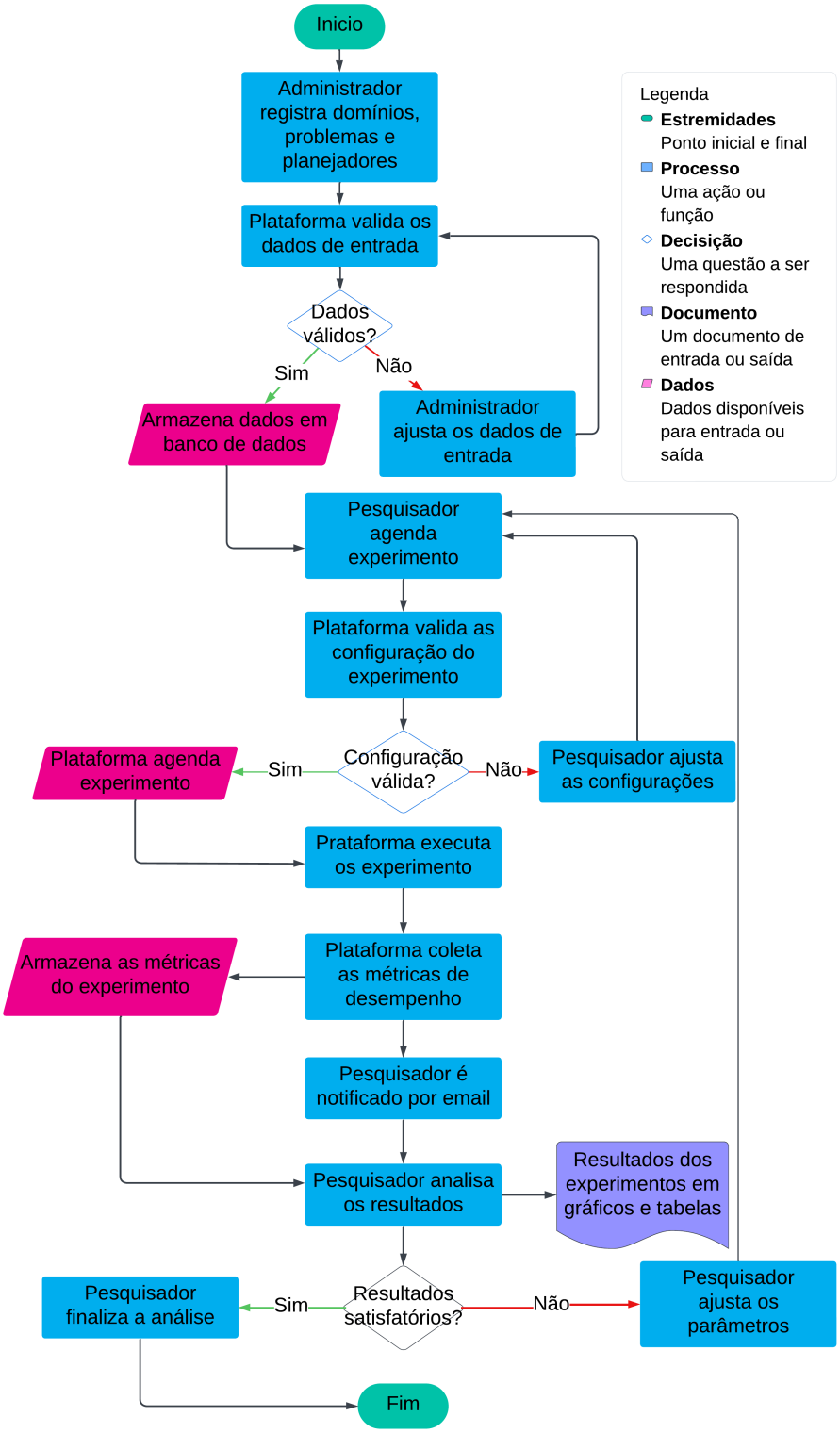
### 3.5.1 Registro de Domínios, Problemas e Planejadores pelo Administrador

O processo de automação da plataforma Planalyzing começa com o administrador registrando os dados básicos no sistema, incluindo domínios, problemas e planejadores. O administrador acessa a interface web e insere as informações detalhadas de cada domínio,

como nome, descrição e definição em PDDL (Planning Domain Definition Language). Em seguida, são registrados os problemas relacionados a cada domínio, associando as definições do problema também no formato PDDL. Para completar os requisitos mínimos necessários para um experimento, o administrador também adiciona os planejadores ao sistema, incluindo seus nomes, descrições e configurações de execução, como comandos de linha e padrões de saída.

Cada planejador pode ter múltiplas versões, e para cada versão são configuradas métricas específicas que serão extraídas dos resultados experimentais. Para cada versão, existe um conjunto de métricas padrões sugeridas pela plataforma, que é registrado no formato chave/valor, onde o valor é uma expressão regular que será aplicada sobre o arquivo de saída para cada problema resolvido pelo planejador, são elas: o tempo de execução, o tempo de busca e tamanho da solução. Após o registro, a plataforma valida automaticamente os dados inseridos, verificando consistência, integridade e possíveis erros de configuração. Caso os dados sejam considerados inválidos, o administrador é notificado para ajustá-los. Quando os dados são aprovados, eles são armazenados no banco de dados, ficando disponíveis para uso pelos pesquisadores durante o agendamento dos experimentos.

Figura 17 – Processo de Automação




### 3.5.2 Agendamento de Experimentos pelo Pesquisador

Com os dados registrados pelo administrador, os pesquisadores podem acessar a plataforma para configurar experimentos. O agendamento começa com o pesquisador selecionando domínios e problemas de interesse, podendo escolher problemas específicos ou incluir todos os

problemas associados a um domínio. Em seguida, o pesquisador seleciona os planejadores e as versões que deseja testar.

Após configurar o experimento, a plataforma valida as informações fornecidas para garantir que a configuração está consistente com os dados do sistema. Se a configuração for considerada inválida, o pesquisador é notificado e pode ajustar os parâmetros antes de reenviar. Quando tudo está correto, o experimento é agendado no sistema, com os dados sendo registrados no banco de dados. A configuração é então passada para o módulo de orquestração, que coordena a execução do experimento por meio de workflows em um gerenciador de tarefas.

### 3.5.3 Execução dos Experimentos pela Plataforma Planalyzing

Após o experimento ser agendado, a plataforma Planalyzing inicia o processo de execução utilizando o sistema de orquestração. O agendador de tarefas ativa o DAG (Directed Acyclic Graph) responsável por executar os experimentos, iniciando com a recuperação dos dados de configuração do banco de dados. O módulo CommandGenerator gera os comandos necessários para cada combinação de planejador e problema, baseando-se nos parâmetros fornecidos. Esses comandos são passados para o ExperimentManager, que instancia contêineres Docker isolados para executar os planejadores. Cada contêiner processa todos os problemas de um planejador, garantindo que os experimentos sejam executados em paralelo para otimizar o tempo de execução. Os resultados gerados pelos planejadores são gravados no sistema de arquivos em um formato padronizado, seguindo o padrão de nomeação especificado durante a configuração.

Após a execução, o ResultProcessor é acionado para ler os arquivos de saída, aplicar expressões regulares e extrair as métricas definidas. Essas métricas são então armazenadas no banco de dados, vinculadas ao experimento correspondente.

### 3.5.4 Notificação e Análise Empírica dos Resultados

Quando as métricas do experimento são armazenadas, o pesquisador é automaticamente notificado por e-mail por meio do módulo NotificationManager. O e-mail informa que o experimento foi concluído e fornece um link direto para a interface de análise. Na plataforma, o pesquisador pode acessar gráficos e tabelas interativas que comparam os resultados obtidos entre diferentes planejadores, problemas e domínios. Esses gráficos destacam métricas como desempenho, tempo de execução e tamanho das soluções.

Caso os resultados não sejam satisfatórios, o pesquisador pode ajustar os parâmetros do experimento diretamente na plataforma, refinando domínios, problemas, planejadores ou métricas, e reexecutar o processo. Se os resultados atenderem às expectativas, o pesquisador pode finalizar a análise e exportar os dados e gráficos para relatórios. Esse ciclo de experimentação e análise empírica permite aos pesquisadores explorar a eficácia de diferentes planejadores em diversos cenários, promovendo uma avaliação detalhada e sistemática dos algoritmos testados.

# 4 USANDO A PLATAFORMA PLANALYZING COM O PLANEJADOR PACTL-SYM

Este capítulo apresenta uma análise empírica de planejadores utilizando a plataforma Planalyzing. Em particular, essa análise teve como motivação avaliar o desempenho do planejador PACTL-Sym.

## 4.1 O planejador PACTL-Sym

O planejador PACTL-Sym (SANTOS, 2018) é a versão simbólica do planejador PACTL (PEREIRA, 2007), projetado para resolver problemas de planejamento em ambientes não-determinísticos com observação completa (FOND). Este planejador é baseado em verificação de modelos e na semântica da lógica temporal $\alpha$-CTL, uma extensão da lógica temporal CTL (*Computation Tree Logic*). Usualmente, os planejadores baseados em verificação de modelos utilizam a lógica CTL. Contudo, a semântica da CTL possui limitações para lidar com problemas de planejamento. A lógica $\alpha$-CTL foi proposta para superar essas limitações (PEREIRA, 2007).

O planejamento não-determinístico aborda cenários em que as ações podem ter efeitos incertos, exigindo a formulação de políticas em vez de planos sequenciais (SANTOS, 2018). Nesse contexto, o PACTL-Sym é capaz de resolver problemas de planejamento considerando os três tipos de políticas: fracas, fortes e forte-cíclicas. A maioria dos planejadores que resolvem problemas FOND buscam apenas por soluções forte-cíclicas. Além disso, o PACTL-Sym é capaz de resolver problemas com metas de alcançabilidade estendida. Uma meta de alcançabilidade estendida expressa, além da condição que deve ser satisfeita no estado final, condições que devem ser satisfeitas ao longo do caminho para a meta. Os planejadores para problemas FOND considerados estado-da-arte, como por exemplo PRP e PR2, não são capazes de lidar com este tipo de meta (SANTOS et al., 2022). Conforme citado anteriormente, as políticas fracas garantem que pelo menos um caminho satisfaça a meta, as políticas fortes asseguram que todos os caminhos possíveis a partir do estado inicial satisfaçam a meta, enquanto as políticas forte-cíclicas garantem a satisfação mesmo na presença de ciclos.

Em sua implementação simbólica, o PACTL-Sym representa conjuntos de estados e ações como fórmulas lógicas e planeja raciocinando diretamente sobre essas fórmulas por meio de operações entre diagramas de decisão binária (*Binary Decision Diagram* - BDD), uma estrutura de dados que representa funções booleanas de forma compacta, permitindo operações eficientes sobre essas funções (HUTH; RYAN, 2004), que levam à síntese de políticas mais eficazes e escaláveis.

Neste trabalho, integramos o planejador PACTL-Sym às avaliações realizadas na plataforma Planalyzing permitindo comparar seu desempenho com outros planejadores em diferentes domínios. Ao incluir o PACTL-Sym nas avaliações, exploramos suas capacidades de resolver problemas de planejamento não-determinístico com dois tipos de políticas: fracas

e forte-cíclicas. Essa integração permitiu comparar o PACTL-Sym com outros planejadores considerados estado-da-arte, analisando o tempo de execução, tempo de busca, tamanho da política e cobertura.

## 4.2 Experimentos realizados

Conforme mencionado anteriormente, foram conduzidos diversos experimentos para validar a plataforma desenvolvida, comparando o desempenho do planejador PACTL-Sym com outros planejadores FOND, incluindo PR2, PRP, MyND, FONDSAT e Paladinus.

Para iniciar a análise, é necessário realizar uma etapa preliminar de configuração, onde os domínios e problemas são adicionados conforme detalhado na Figura 4. Em seguida, os planejadores mencionados anteriormente são registrados com suas informações de execução, padrões de arquivos e métricas, como ilustrado na Figura 5. Após essa configuração, os planejadores são incluídos em um experimento, juntamente com suas versões, domínios e problemas a serem analisados conforme pode ser visto na Figura 7. A plataforma registra o experimento e realiza a execução em paralelo, aproveitando a independência das saídas dos planejadores para otimizar o processamento.

Durante a execução, os planejadores resolvem os problemas definidos para cada domínio, gerando arquivos individuais com as métricas de desempenho, como tempo de execução, tempo de busca e tamanho da solução. Um exemplo da saída gerada, considerando o planejador Pactl-Sym e o domínio *islands* pode ser visto na Figura 18a. Os resultados são armazenados em pastas temporárias do sistema operacional e, posteriormente, processados pela plataforma, quando são extraídas as métricas necessárias para a avaliação empírica. A Figura 18b mostra as métricas coletadas armazenadas em banco de dados, incluindo o arquivo com os dados brutos que são armazenados compactados.

Com os experimentos concluídos, os resultados podem ser analisados comparativamente. Para isso, é necessário criar uma análise, selecionando os experimentos processados, planejadores envolvidos, domínios, problemas e as métricas desejadas para gerar uma análise Figura 9. A plataforma organiza as configurações e exibe os resultados em gráficos e listas detalhadas Figuras (19, 20, 21) e a Tabela 3, permitindo a comparação direta do desempenho entre os planejadores. Cada análise pode ser salva para referência futura, facilitando o acompanhamento e refinamento contínuo. Esse fluxo integrado, desde o agendamento até a análise, proporciona uma avaliação sistemática e eficiente, permitindo identificar padrões, diferenças de desempenho e possíveis melhorias para os planejadores automatizados em diversos cenários experimentais. Além das métricas citadas, a plataforma também permite configurar limite de tempo para resolver cada problema e limite de memória disponível para o planejador.

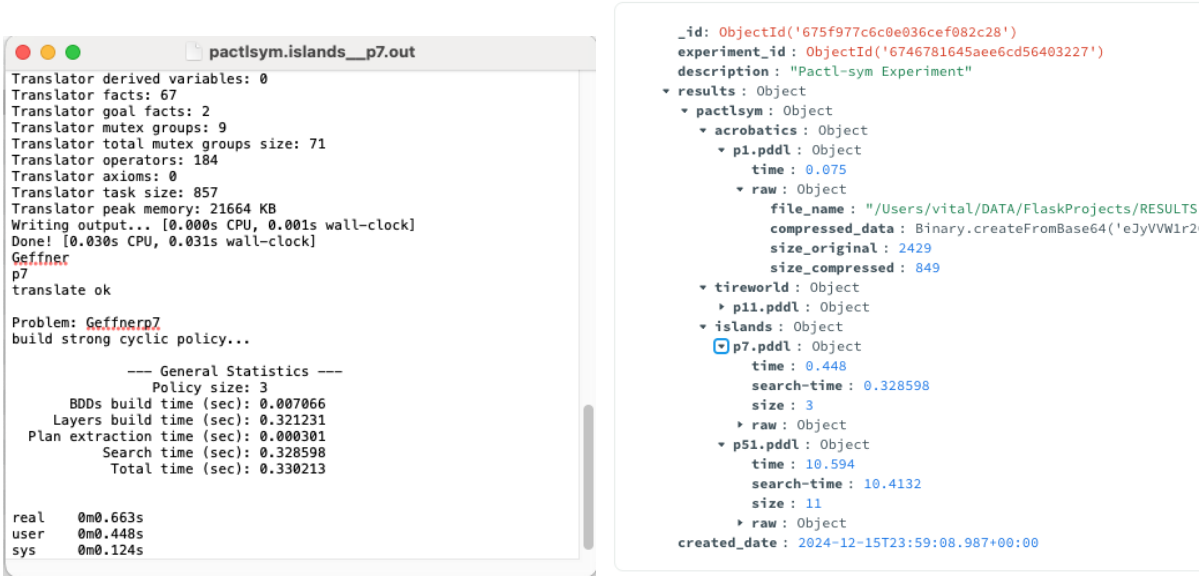
A partir das configurações definidas anteriormente, foi possível analisar empiricamente o planejador Pactl-Sym em alguns contextos, com diferenças no total de recursos computacionais disponibilizados e configurações particulares do planejador. A seguir, listamos algumas:

- Execução do PACTL-Sym para encontrar políticas fracas sem considerar limite de tempo

ou memória;

- Execução do PACTL-Sym para encontrar políticas fracas considerando limite de 1h para resolver cada problema e 4GB de memória;
- Execução do PACTL-Sym para encontrar políticas forte-cíclica considerando sem considerar limite de tempo ou memória;
- Execução do PACTL-Sym para encontrar políticas forte-cíclica considerando limite de 1h para resolver cada problema e 4GB de memória;
- Execução dos demais planejadores considerando limite de 1h para resolver cada problema e 4GB de memória. Por padrão esses planejadores buscam por uma solução forte-cíclica.

Figura 18 – Arquivo de dados brutos e processados para o planejador Pactl-Sym




(a) Dados brutos gerados como saída do planejador Pactl-Sym

(b) Métricas processadas e armazenada em banco de dados

## 4.3 Análise empírica

Para as análises empíricas, apresentados nessa seção, os experimentos foram realizados utilizando um sistema com 48 CPUs, 96 GB de memória RAM e 100 GB de armazenamento, executando o Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1015-aws x86_64). O processamento foi conduzido ao longo de 3 dias, utilizando Python com suporte a multithreading para otimizar a execução. Foram avaliados os seis planejadores distintos em 18 domínios, com uma quantidade variável de problemas por domínio. A infraestrutura permitiu uma execução paralelizada, gerando as saídas como as mostradas na Figura 18a. Cada planejador gerou um arquivo por problema resolvido, onde estão as informações com as métricas definidas para cada planejador.

A Tabela 3 apresenta a cobertura normalizada para todos os domínios avaliados. A primeira coluna lista o nome de cada domínio, com o número total de problemas indicado entre parênteses. As demais colunas exibem os resultados obtidos por cada planejador nos respectivos domínios, sendo que a Coluna 7 destaca os resultados do PACTL-Sym na tentativa

de encontrar uma política forte-cíclica para os problemas. Os valores foram normalizados em uma escala onde 1 representa a resolução de todos os problemas do domínio pelo planejador, enquanto 0 indica que nenhum problema foi resolvido.

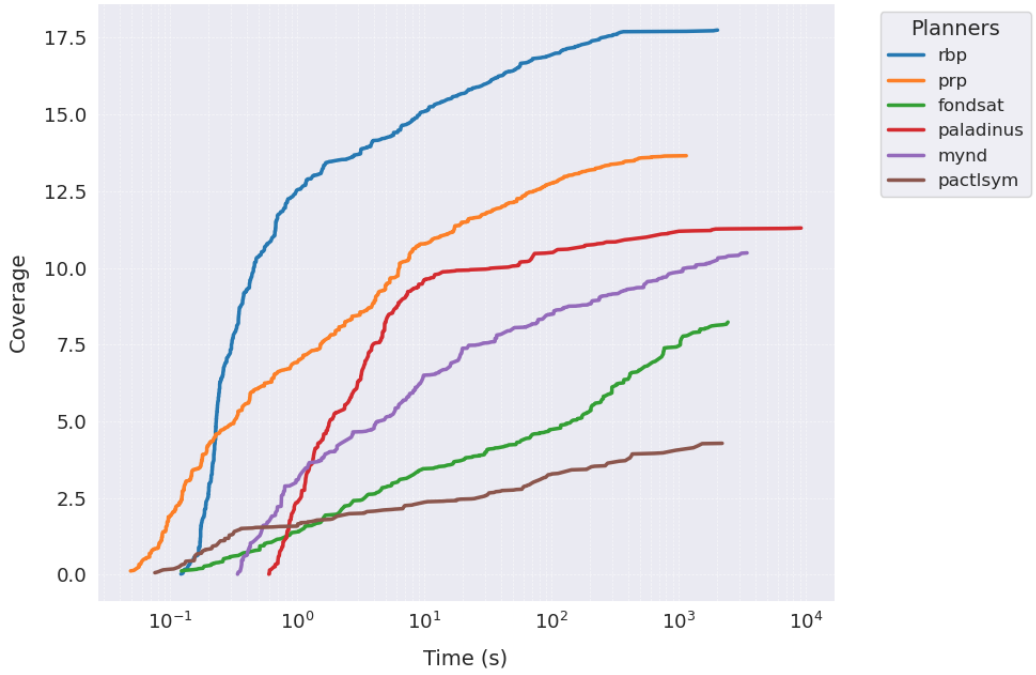
No total, foram analisados 18 domínios, somando 800 problemas. A análise revela que o PACTL-Sym não conseguiu resolver nenhum problema em oito domínios, o que motivou uma investigação detalhada sobre suas limitações ao lidar com algumas das entradas fornecidas. Por outro lado, o PACTL-Sym resolveu todos os problemas do domínio Islands, projetado para conter diversos caminhos que aparentam levar rapidamente à meta, mas que, na verdade, são enganosos e não garantem alcançar o objetivo. O desempenho do PACTL-Sym nesse domínio demonstrou sua capacidade de identificar e evitar esses caminhos enganosos (GEFFNER; GEFFNER, 2018).

Tabela 3 – Tabela de Cobertura Normalizada

| domain (size) | pr2 | prp | fondsat | paladinus | mynd | pactlsym |
|---|---|---|---|---|---|---|
| acrobatics (8) | 1.00 | 1.00 | 0.50 | 1.00 | 1.00 | 0.00 |
| beam-walk (11) | 1.00 | 1.00 | 0.27 | 0.82 | 1.00 | 0.00 |
| bw-new (50) | 0.82 | 0.84 | 0.16 | 0.36 | 0.42 | 0.08 |
| chain (10) | 1.00 | 1.00 | 0.10 | 1.00 | 1.00 | 0.20 |
| earth-obs (40) | 1.00 | 1.00 | 0.17 | 0.65 | 0.78 | 0.00 |
| elevators (15) | 1.00 | 1.00 | 0.47 | 0.53 | 0.93 | 0.67 |
| faults-new (190) | 1.00 | 1.00 | 0.04 | 0.12 | 1.00 | 0.00 |
| first-new (88) | 0.99 | 0.99 | 0.06 | 0.06 | 0.09 | 0.05 |
| forest-new (100) | 0.94 | 0.88 | 0.10 | 0.18 | 0.16 | 0.30 |
| tidyup-mdp (10) | 1.00 | 0.00 | 0.10 | 0.40 | 1.00 | 0.00 |
| tire (12) | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.92 |
| tri-tire (40) | 1.00 | 1.00 | 0.10 | 0.20 | 0.17 | 0.07 |
| zeno (15) | 1.00 | 1.00 | 0.33 | 0.53 | 0.60 | 0.33 |
| doors (15) | 1.00 | 0.80 | 1.00 | 0.87 | 0.73 | 0.67 |
| islands (60) | 1.00 | 0.52 | 0.95 | 1.00 | 0.22 | 1.00 |
| miner (51) | 1.00 | 0.25 | 0.98 | 1.00 | 0.00 | 0.00 |
| tire-spiky (11) | 1.00 | 0.09 | 0.91 | 0.91 | 0.18 | 0.00 |
| tire-truck (74) | 1.00 | 0.28 | 0.99 | 0.68 | 0.20 | 0.00 |
| TOTAL (800) | 17.75 | 13.65 | 8.23 | 11.30 | 10.49 | 4.28 |

A Figura 19 apresenta um gráfico que ilustra a cobertura ao longo do tempo, mostrando a quantidade de problemas resolvidos por cada planejador conforme o experimento avança. Observa-se que, embora o PACTL-Sym tenha apresentado um desempenho inferior em relação aos demais planejadores, esses resultados refletem também os domínios nos quais o PACTL-Sym não conseguiu processar corretamente a especificação da entrada. O gráfico considera especificamente os problemas resolvidos pelo PACTL-Sym ao buscar uma política forte-cíclica, permitindo uma análise mais detalhada de seu comportamento ao longo do tempo.

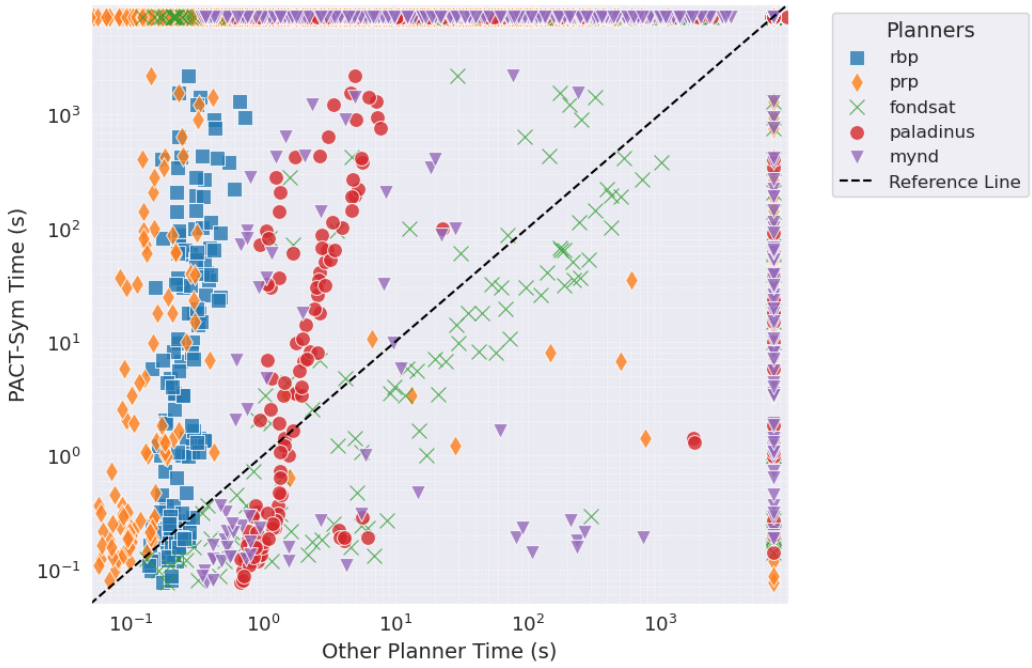
Figura 19 – Gráfico de Cobertura sobre o Tempo




Para avaliar o tempo de resolução dos problemas, a plataforma gera um gráfico de dispersão que compara o tempo de execução de um planejador de referência com outros planejadores. A Figura 20 apresenta o gráfico considerando o PACTL-Sym como referência ao buscar uma política forte-cíclica. Neste gráfico, a interpretação dos pontos é a seguinte:

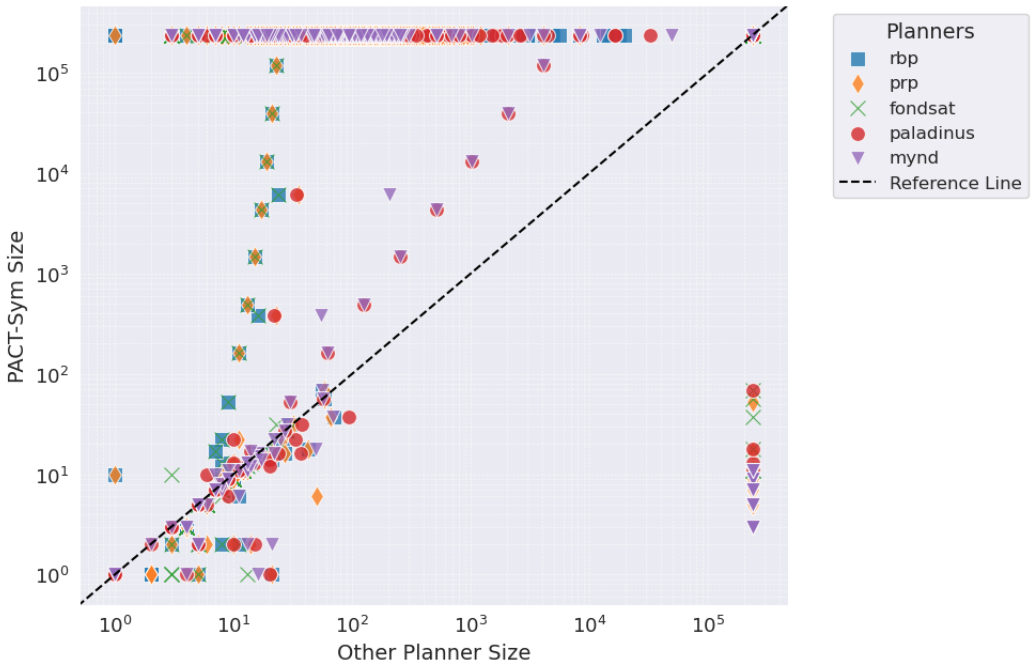
- Pontos na linha x = y: Indicam que o tempo do PACTL-Sym e do planejador analisado foram semelhantes.
- Pontos acima da linha x = y: Representam os casos em que o tempo de execução do PACTL-Sym (eixo Y) foi maior do que o tempo do planejador comparado (eixo X).
- Pontos abaixo da linha x = y: Mostram que o PACTL-Sym apresentou um desempenho superior, resolvendo os problemas em menos tempo que o planejador analisado.

Figura 20 – Gráfico de Comparação da Performance dos Planejadores sobre o Tempo




A Figura 21 apresenta um gráfico que mostra o tamanho das políticas obtidas por cada um dos planejadores. O tamanho da política é um exemplo de métrica coletada a partir das informações fornecidas pelos planejadores, ou seja, essas informações são geradas em arquivos de saída pelos planejadores e capturadas pela plataforma utilizando expressões regulares configuradas no experimento. O gráfico apresentado considera o PACTL-Sym ao tentar gerar uma política forte-cíclica como referência.

Figura 21 – Gráfico de Comparação da Performance dos Planejadores sobre o Tamanho




Os experimentos realizados na plataforma colaboraram com o desenvolvimento contínuo

do PACTL-Sym. A análise comparativa realizada envolveu diversos domínios que não haviam sido testados com o PACTL-Sym até o desenvolvimento deste trabalho e pôde revelar pontos fortes, bem como limitações e oportunidades de melhoria, contribuindo para ajustes em seu algoritmo. Como exemplo, uma das contribuições da plataforma ao planejador PACTL-Sym foi identificar que, para alguns problemas, a saída gerada pelo planejador era excessivamente grande, chegando a cerca de 12 GB. Esse problema resultou em um novo desafio para a equipe de desenvolvimento, que atualmente está analisando as causas. Para a plataforma Planalyzing, esse resultado gerou um novo requisito: limitar o tamanho dos arquivos de saída gerados pelos planejadores, que posteriormente são usados para criar gráficos e armazenados no banco de dados.

# 5 CONCLUSÃO

O trabalho teve como objetivo desenvolver uma plataforma integrada para a avaliação automatizada de planejadores, possibilitando agendamento, execução e análise de experimentos de maneira sistemática e reprodutível. Para modelar o sistema, utilizamos o Modelo C4, que se destacou por sua abordagem hierárquica e estruturada, facilitando a visualização da arquitetura em diferentes níveis de abstração. Através dos diagramas de Contexto, Contêiner, Componente e Código, foi possível capturar a complexidade do sistema, detalhar os componentes internos e documentar as interações entre módulos de software e sistemas externos.

A metodologia adotada possibilitou a construção de uma plataforma, que atende às demandas de pesquisadores na área de planejamento automatizado. Os resultados obtidos demonstraram a contribuição do sistema na comparação entre diferentes planejadores, utilizando métricas personalizadas, aplicadas em benchmarks reconhecidos e sob condições controladas. Gráficos gerados pela plataforma permitiram avaliar o desempenho dos planejadores em termos de tempo total de execução, tempo de busca e cobertura normalizada. Esses resultados possibilitaram validar a plataforma como uma ferramenta prática para a análise empírica de planejadores automatizados.

No entanto, algumas limitações foram encontradas durante o desenvolvimento. A integração com planejadores externos exigiu esforços adicionais devido à ausência de padrões unificados para entrada e saída de dados. Outra dificuldade foi a necessidade de configurar múltiplos contêineres Docker para garantir isolamento e escalabilidade na execução dos experimentos, o que aumentou a complexidade técnica do sistema. Além disso, a dependência de uma infraestrutura computacional robusta pode limitar a utilização da plataforma em contextos com recursos mais restritos. Apesar dessas dificuldades, o trabalho demonstrou que a automação da análise empírica pode contribuir para melhorar a qualidade dos planejadores, possibilitando um ambiente mais consistente e acessível para pesquisas nessa área.

## 5.1 Trabalhos futuros

Para trabalhos futuros, propõe-se a expansão das funcionalidades da plataforma, incorporando planejadores que utilizem outras abordagens, como a probabilísticas e numéricas, além de suportar domínios e problemas especificados em outras linguagens além do PDDL. Ademais, a criação de um repositório público contendo domínios, problemas e resultados experimentais poderia incentivar a colaboração entre pesquisadores, facilitar a replicação de experimentos e contribuir para o estabelecimento de novos benchmarks no campo do planejamento automatizado.

# Referências


EARLEY, J. Syntax extension using a run time model. **International Journal of Computer & Information Sciences**, v. 3, n. 3, p. 189–196, 1974. ISSN 1573-7640. Disponível em: <https://doi.org/10.1007/BF00977254>. Citado na página 4.

GEFFNER, T.; GEFFNER, H. Compact policies for fully observable non-deterministic planning as sat. In: **Proceedings of the International Conference on Automated Planning and Scheduling**. [S.l.: s.n.], 2018. v. 28, p. 88–96. Citado 2 vezes nas páginas 17 e 46.

GHALLAB, M. et al. **PDDLThe planning domain definition language**. [S.l.], 1998. Citado na página 1.

HELMERT, M. The fast downward planning system. **Journal of Artificial Intelligence Research (J. Artif. Int. Res.)**, v. 26, n. 1, p. 191–246, 2006. ISSN 1076-9757. Citado 2 vezes nas páginas 16 e 17.

HELMERT, M. **Fast Downward resources**. 2024. <https://www.fast-downward.org/>. Accessed: 2024-09-03. Citado na página 16.

HUTH, M.; RYAN, M. **Logic in Computer Science: Modelling and Reasoning About Systems**. [S.l.]: Cambridge University Press, 2004. Citado na página 43.

IPC. **International Planning Competition 2023**. 2023. Accessed: 2024-12-13. Disponível em: <https://ipc2023.github.io/>. Citado na página 10.

LONG, D. et al. The aips-98 planning competition. **AI Magazine**, v. 21, p. 13–33, 06 2000. Citado na página 3.

MATTMüLLER, R. et al. Pattern database heuristics for fully observable nondeterministic planning. In: **Proceedings of the International Conference on Automated Planning and Scheduling**. [S.l.: s.n.], 2010. v. 20, p. 105–112. Citado na página 17.

MCDERMOTT, D. et al. **PDDL - The Planning Domain Definition Language Manual**. New Haven, CT, USA, 1998. Based on the UCPOP language manual written by researchers from the University of Washington, including Anthony Barrett, Dave Christianson, Marc Friedman, Chung Kwok, Keith Golden, Scott Penberthy, David E. Smith, Ying Sun, and Daniel Weld. Citado na página 4.

MUISE, C. **PR2 is the next iteration in the evolution of SoA non-deterministic planning**. 2024. <https://github.com/QuMuLab/pr2>. Accessed: 2024-09-03. Citado 2 vezes nas páginas 11 e 17.

MUISE, C. **PR2 Rebooted**. 2024. <https://mulab.ai/project/pr2/>. Accessed: 2024-09-03. Citado 3 vezes nas páginas 11, 13 e 17.

MUISE, C. et al. Leveraging fond planning technology to solve multi-agent planning problems. **Distributed and Multi-Agent Planning (DMAP-15)**, p. 83, 2015. Citado na página 13.

MUISE, C.; MCILRAITH, S.; BECK, C. Improved non-deterministic planning by exploiting state relevance. In: **Proceedings of the International Conference on Automated Planning and Scheduling**. [S.l.: s.n.], 2012. v. 22, p. 172–180. Citado na página 17.

PEREIRA, R. F. et al. Iterative depth-first search for fond planning. In: **Proceedings of the International Conference on Automated Planning and Scheduling**. [S.l.: s.n.], 2022. v. 32, p. 90–99. Citado na página 17.

PEREIRA, S. do L. **Planejamento sob incerteza para metas de alcançabilidade estendidas**. Tese (Tese de Doutorado) — Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, novembro 2007. Citado na página 43.

SANTOS, V. B. d. et al. Symbolic fond planning for temporally extended goals. In: **Workshop on Knowledge Engineering for Planning and Scheduling**. [S.l.: s.n.], 2022. Citado 2 vezes nas páginas 29 e 43.

SANTOS, V. B. dos. **Planejamento baseado em Verificação Simbólica de Modelos**. Dissertação (Dissertação de Mestrado) — Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, outubro 2018. Durante o desenvolvimento deste trabalho, a autora recebeu auxílio financeiro do CNPq. Citado 3 vezes nas páginas , 3 e 43.

TAITLER, A. et al. The 2023 international planning competition. **Ai Magazine**, 2024. Citado 2 vezes nas páginas 10 e 14.