Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

Resolvendo Xian-Xiang como Planejamento

Autor: Rodrigo Tiago Costa Lima
Orientador: Prof. Dr. Bruno César Ribas

Brasília, DF
2024

Rodrigo Tiago Costa Lima

Resolvendo Xian-Xiang como Planejamento

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. Bruno César Ribas

Brasília, DF

2024

Rodrigo Tiago Costa Lima
Resolvendo Xian-Xiang como Planejamento/ Rodrigo Tiago Costa Lima. – Brasília, DF, 2024-
52 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Bruno César Ribas

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2024.

1. Planejamento. 2. PDDL. I. Prof. Dr. Bruno César Ribas. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Resolvendo Xian-Xiang como Planejamento

CDU 02:141:005.6

Rodrigo Tiago Costa Lima

Resolvendo Xian-Xiang como Planejamento

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 01 de junho de 2013 – Data da aprovação do trabalho:

Prof. Dr. Bruno César Ribas
Orientador

Prof. Dr. John Lenon Cardoso Gardenghi
Convidado 1

Prof. Dr. Daniel Saad Nogueira Nunes
Convidado 2

Brasília, DF
2024

Resumo

No campo da IA conhecido como planejamento automatizado, existem aplicações que buscam determinar uma sequência de passos para solucionar um problema específico, a essa sequência de passos é dado o nome de plano. O objetivo deste trabalho foi desenvolver um modelo de resolução do jogo Xian-Xiang empregando planejamento automatizado. Por isso foi desenvolvido um domínio na linguagem Problem Domain Definition Language (PDDL) para abstrair o jogo e resolvê-lo utilizando uma abordagem de planejamento. Foram criados um gerador de problemas e um gerador de animações, o primeiro para auxiliar na experimentação e o último para auxiliar na visualização das soluções dos problemas. Um total de 105 instâncias do problema foram resolvidas. Dessas, 100 instâncias foram criadas aleatoriamente para testar o desempenho do domínio e 5 para testar casos específicos de borda. Como resultado, as soluções dessas instâncias foram obtidas, sendo que todas as respostas finalizam corretamente o jogo, e 2 finalizam com o melhor desempenho possível para o problema.

Palavras-chave: Planejamento; PDDL; Xian-Xiang; KadoKado.

Abstract

In the branch of AI called automated planning, you can find applications that try to work out a series of steps to solve a given problem. The aim of this work was to develop an algorithm for solving the Xian-Xiang game using automated planning. To this end, a domain was developed in the Problem Domain Definition Language (PDDL) to abstract the game and solve it using a planning approach. A problem generator and an animation generator were created, the first to help with experimentation and the latter to help visualize problem solutions.A total of 105 problem instances were solved. Of these, 100 instances were randomly created to test domain performance and 5 to test specific edge cases. As a result, the solutions to these instances were obtained, with all answers correctly finishing the game, and 2 finishing with the best possible performance for the problem.

Key-words: Planning; PDDL; Xian-Xiang; KadoKado.

Lista de ilustrações

Lista de tabelas

Lista de códigos

Lista de abreviaturas e siglas

IA	Inteligência Artificial
PDDL	Problem Domain Definition Language
IPC	International Planning Competition
SAT	Satisfiability Testing

Sumário

1 Introdução

O Planejamento automatizado é um ramo da Inteligência Artificial voltado para a criação de sistemas que geram uma sequência de ações (plano), que partem de um estado X para um estado desejado Y (SLEATH; BERCHER, 2024).

Para auxiliar na criação desses sistemas acima mencionados, foram criadas algumas ferramentas, por exemplo, a linguagem chamada Problem Domain Definition Language (PDDL), que é utilizada para definir domínios, no qual são descritos os tipos de objetos, predicados e ações de um contexto, e para definir problemas, no qual são abstraídos o estado inicial e o estado desejado (MCDERMOTT et al., 1998).

A linguagem PDDL é utilizada com outras ferramentas chamadas de planejadores, nos quais há a entrada tanto de um domínio, quanto de um problema, para então retornar um plano que resolva o problema (SLEATH; BERCHER, 2024).

Outra ferramenta desenvolvida, agora para visualizar sequencialmente a solução para um problema de planejamento, a ferramenta Planimation (CHEN et al., 2020) gera uma animação que facilita para ajudar na compreensão do plano obtido.

O jogo Xian-Xiang é um exemplo problema que pode ser modelado visando a solução por meio de um plano, nele há um tabuleiro com 42 peças que devem ser combinadas em duplas para retirá-las do tabuleiro, portanto conta com um estado inicial (início do jogo com todas as peças no tabuleiro) e desejado (final do jogo com a maior pontuação) claros, sendo assim classificado como um problema de planejamento clássico (MCDERMOTT, 2000).

Os objetivos deste trabalho foram modelar um domínio para solucionar o jogo Xian-Xiang utilizando a linguagem PDDL e testar diversas abordagens para procurar a mais adequada ao modelo, foram feitos também um gerador de problema e um gerador de arquivo base de animação, e em seguida foram realizados diversos experimentos para testar a qualidade do domínio criado.

No Capítulo 2 apresenta-se a fundamentação teórica para este trabalho. No Capítulo 3 apresenta-se os trabalhos relacionados, contando com artigos e outras monografias relacionadas a este trabalho. No Capítulo 4 apresenta-se uma breve descrição do jogo Xian-Xiang, suas regras e objetivos. No Capítulo 5 apresenta-se a formulação em PDDL com a explicação do domínio, os predicados, as funções e as ações, além de um resumo do código elaborado pelo gerador de problemas e pelo gerador de arquivos-base para a animação. No Capítulo 6 apresentam-se os experimentos e a análise dos resultados. Finaliza-se com a apresentação das conclusões deste trabalho.

2 Fundamentação Teórica

2.1 Planejamento Automatizado

O planejamento automatizado é uma área da Inteligência Artificial (IA) (GEFFNER, 2002), focada na criação de sistemas que geram uma sequência de ações (plano) para transitar de um estado inicial X para um estado desejado Y (SLEATH; BERCHER, 2024).

Este campo da IA lida com a resolução de problemas em uma classe de modelos, que definem o escopo do planejador, os tipos de problemas que ele deve resolver, a forma das soluções e quais soluções são consideradas ótimas (GEFFNER, 2002).

Os problemas de planejamento na IA são abordados de duas maneiras principais: abordagens independentes de domínio, que tentam resolver problemas gerais sem conhecimento específico de domínio, e abordagens dependentes de domínio, que utilizam heurísticas específicas para controlar a operação do planejador. As abordagens dependentes de domínio enfrentam desafios comuns na IA aplicada, como a necessidade de justificar soluções, a dificuldade de aquisição de conhecimento e a aplicabilidade limitada dos princípios de design a diferentes domínios (HENDLER; TATE; DRUMMOND, 1990). Nesse trabalho foi realizado foi utilizada a abordagem em que o domínio é conhecido.

Dentre os diversos ramos dentro do Planejamento automatizado, neste projeto resolveremos um problema que se encaixa no ramo de planejamento clássico, nele é fornecida uma situação inicial, um conjunto de definições de ações e um objetivo a ser alcançado. A solução é uma sequência de ações que, quando executadas a partir da situação inicial, resultam em uma situação na qual o objetivo é verdadeiro. Supõe-se que o planejador conhece todos os fatos verdadeiros na situação inicial e os efeitos de cada ação (MCDERMOTT, 2000).

2.2 Linguagem PDDL

A chamada Problem Domain Definition Language (MCDERMOTT, 2000) é considerada a linguagem padrão para a modelagem de problemas de planejamento, ela pode trabalhar com diversos planejadores diferentes, que tem propósitos diversos. Ela é utilizada na International Planning Competition (IPC) como linguagem padrão para o ramo de Planejamento Clássico.

Para resolver um problema de planejamento, é comum utilizar dois arquivos, um de domínio (que define o escopo) e outro de problema (que contém de fato o problema

a ser resolvido) como entrada para um planejador, que após receber algumas heurísticas, procura um plano.

A linguagem PDDL pretende expressar a física de um domínio, isto é, quais predicados existem, quais ações são possíveis, qual é a estrutura das ações compostas e quais são os efeitos das ações (MCDERMOTT et al., 1998).

2.2.1 Domínio

O arquivo de domínio é necessário para definir o escopo do problema, nele que consta os requisitos, tipos de objetos, predicados, funções e ações, que serão explicados a seguir. Para exemplificar como funciona um arquivo de domínio, será utilizado o um contexto de um sistema de irrigação, onde é necessário irrigar as plantas enquanto há o controle de água. O domínio pode ser definido como no Código 2.1.

```
1 (define (domain sistema-de-irrigacao)
2     ; Aqui ficam os requisitos, os predicados, as funcoes e as acoes
3 )
```

Código 2.1 – Código de exemplo para definição de domínio

Nesse código é definido o domínio sistema-de-irrigacao na linha 1, isso determina o começo do arquivo de domínio.

2.2.1.1 Requisitos

Logo após a nomenclatura do domínio, há a listagem dos requisitos, alguns exemplos são “equality”, que comunica ao planejador a necessidade de suportar a comparação de igualdade, e “negative-precondition”, que comunica que é necessário suportar precondições negativas. Os requisitos podem ser listados como no exemplo do Código 2.2

```
1     (:requirements :strips :fluents)
```

Código 2.2 – Código de exemplo para listagem dos requisitos

Os requisitos desse código são “strips”, que permite a negação do predicados, e “fluents”, que permite a utilização de elementos numéricos. Os requisitos podem ser listados em sequencia separados por espaço.

2.2.1.2 Predicados

Os predicados são declarações que podem estar ligadas a diversos ou nenhum objeto. Abstraídos da matemática os predicados são geralmente entendidos como funções

booleanas podendo ter dois estados possíveis, os quais são “verdadeiro” e “falso”. Os predicados podem ser modificados com ao realizar ações. Os predicados podem ser listados como no exemplo do Código 2.3.

```
1    (:predicates
2        (sistema-ativo)
3        (planta-seca ?planta)
4        (agua-na-planta ?planta ?setor)
5    )
```

Código 2.3 – Código de exemplo para listagem dos predicados

Nesse código há um predicado com nenhum objeto sendo (sistema-ativo), que determina que o sistema de irrigação está ativo, um com um objeto (planta-seca ?planta), que indica se uma planta está seca, e um com dois objetos (agua-na-planta ?planta ?setor), que indica que a planta em um determinado setor foi regada.

2.2.1.3 Funções

Funções são variáveis numéricas, que também podem estar ligadas a objetos ou não. Geralmente estão ligadas a métricas visando a sua minimização ou maximização. As funções podem ser escritos como no exemplo do Código 2.4.

```
1    (:functions
2        (consumo-agua-total)
3    )
```

Código 2.4 – Código de exemplo para listagem de funções

Nesse código há a função (consumo-agua-total), que mantém o controle do consumo total de água.

2.2.1.4 Ações

Ações modificam os predicados e funções. Elas precisam ter um efeito e podem ter parâmetros, que são variáveis de objetos, e precondições, que podem ser predicados. As ações podem ser descritas como no exemplo do Código 2.5.

```
1    (:action ativar-sistema
2        :precondition (not (sistema-ativo))
3        :effect (and
4            (sistema-ativo)
5            (increase (consumo-agua-total) 5)
```

```
        )
    )

    (:action regar-planta
        :parameters (?planta)
        :effect (and
            (not (planta-seca ?planta))
            (increase (consumo-agua-total) 10)
        )
    )

    (:action regar-setor
        :parameters (?planta ?setor)
        :precondition (and
            (sistema-ativo)
            (planta-seca ?planta)
        )
        :effect (and
            (agua-na-planta ?planta ?setor)
            (not (planta-seca ?planta))
            (increase (consumo-agua-total) 20)
        )
    )
```

Código 2.5 – Código de exemplo para ações

Nesse código há três ações, a primeira é ativar-sistema, que ativa o sistema para começar irrigação, a segunda é regar-planta, que nega o predicado planta-seca, e a terceira regar-setor, que rega todo um setor.

2.2.2 Problema

O arquivo de problema descreve a condição atual a ser resolvida, para isso é necessário dar nome ao problema e definir o domínio a que se refere como no exemplo do Código 2.6. No contexto deste trabalho, ele conta com o bloco de objetos, o de estado inicial, o de objetivos e o de métrica.

```
(define (problem problema-irrigacao)
    (:domain sistema-de-irrigacao)
    ; Aqui ficam os objetos, o estado inicial, o objetivo e a metrica
)
```

Código 2.6 – Código de exemplo para definição de problema

Nesse código de problema, há a definição do problema problema-irrigacao, e a definição de qual é domínio de referência.

2.2.2.1 Objetos

Este bloco lista cada objeto presente no problema, como visto no Código 2.7.

```
(:objects
    planta1 planta2 - planta
    setor1 setor2 - setor
)
```

Código 2.7 – Código de exemplo para listagem de objetos

Nesse código há a definição de quatro objetos: planta1, planta2, setor1 e setor2.

2.2.2.2 Estado Inicial

Neste bloco consta os predicados verdadeiros no inicio do problema, além do valor de cada função. O exemplo do Código 2.8 mostra como pode ser escrito o estado inicial.

```
(:init
    (planta-seca planta1)
    (planta-seca planta2)
    (= (consumo-agua-total) 0)
)
```

Código 2.8 – Código de exemplo para estado inicial

Nesse código de estado inicial há duas declarações de predicados verdadeiros, (planta-seca planta1) e (planta-seca planta2), e a atribuição de 0 à função (consumo-agua-total).

2.2.2.3 Objetivos

Neste bloco há a descrição do estado final desejado após o planejamento, deve-se notar que podem haver predicados negados, e que os predicados que não estão neste bloco são ignorados, diferentemente do bloco de estado inicial, no qual todo predicado não presente é falso, neste bloco, todo predicado que não constar será ignorado e não ter como objetivo a sua negação. Um exemplo disso pode ser encontrado no Código 2.9

```
(:goal
    (and
        (agua-na-planta planta1 setor1)
        (agua-na-planta planta2 setor2)
    )
)
```

Código 2.9 – Código de exemplo para objetivo

Nesse código há a definição que o objetivo é ter dois predicados verdadeiros, (agua-na-planta planta1 setor1) e (agua-na-planta planta2 setor2).

2.2.2.4 Métrica

Tal como descrito na seção de funções, as métricas estão presentes ao se procurar a minimização ou maximização de alguma função do problema, ou ainda da soma ou subtração de funções, como demonstrado no exemplo do Código 2.10.

```
1  (:metric minimize
2        (consumo-agua-total)
3    )
```

Código 2.10 – Código de exemplo para as métricas

Nesse código há a métrica que minimiza a função (consumo-agua-total).

2.3 Planejadores

A linguagem PDDL é utilizada com outras ferramentas chamadas de planejadores, nos quais há a entrada tanto de um domínio, quanto de um problema, para então retornar um plano que resolva o problema (SLEATH; BERCHER, 2024). Ainda há planejadores com objetivos diferentes ou com heurísticas que variam sua utilidade, a International Planning Competition (IPC), que compara planejadores, divide em três grupos para o planejamento clássico:

- o grupo dos planejadores focados em encontrar um plano com agilidade, como o "madagascar";
- o grupo dos planejadores focados em encontrar planos cada vez melhores dentre de um tempo limite, como o "seq-sat-maidu;"
- e o grupo dos planejadores focados em encontrar somente o melhor plano, como o "scorpion".

2.4 Planimation

Planimation é uma plataforma modular e extensível para visualizar soluções sequenciais de problemas de planejamento especificados em PDDL (CHEN et al., 2020).

Esta ferramenta pode ser usada como uma extensão do *PDDL Editor*[1], sendo possível utilizá-la chamando um planejador, que resolveria o problema e depois a extensão montaria a animação, ou utilizá-la já com um plano já encontrado, assim a extensão somente montaria a animação.

Para isso, essa ferramenta é utilizada em conjunto com outro arquivo base de animação criado numa linguagem semelhante a PDDL que descreve as especificações dos elementos animados. Nesse arquivo há a descrição de como devem ser animados os objetos e de como cada predicado ou ação os modificam:

2.4.1 Definição

Assim como o domínio e o problema, a animação tem ser definida, mas não é necessário ter o nome domínio escrito no arquivo de animação, como é no arquivo de problema. Como pode ser visto no Código 2.11.

```
(define (animation sou-um-exemplo-de-animacao)
    ; Aqui ficam os predicados, os elementos visuais e imagens
)
```

Código 2.11 – Código de exemplo para definição de animação

2.4.2 Predicados

Os predicados modificam as propriedades dos objetos presentes na animação, causando assim efeitos visuais na animação. O exemplo do Código 2.12 mostra um predicado qualquer, assim que ele for verdadeiro, o efeito dele, que é igualar a coordenada "x" de dois objetos e a coordenada "y" com a altura de outro objeto, será ativado e ocorrerá na animação.

```
    (:predicate sou-um-predicado-qualquer-presente-no-dominio
        :parameters (?o1 ?o2)
        :effect(
        (equal (?o1 x) (?o2 x))
        (equal (?o2 y) (?o1 height))
        )
    )
```

Código 2.12 – Código de exemplo para um predicado na animação

[1] https://editor.planning.domains/

2.4.3 Elementos Visuais

Os elementos visuais são os objetos que estarão na animação, sendo eles presentes no arquivo de problema ou não. Cada objeto tem propriedades variadas que definem a sua imagem para representação, que deve estar em Base64, sua posição inicial, "x" e "y", se o nome do objeto deve estar presente na animação e o tamanho no objeto, como no exemplo do Código 2.13, no qual há um objeto presente no problema, contendo assim o ":type predefine", e um objeto que existe somente na animação, contendo assim o ":type custom".

```
1   (:visual objeto-existente-no-problema
2       :type predefine
3       :objects (objeto-existente-no-problema)
4       :properties(
5           (prefabImage imagem-em-Base64)
6           (showName TRUE)
7           (x 0)
8           (y 0)
9           (color WHITE)
10          (width 10)
11          (height 10)
12      )
13  )
14  (:visual objeto-nao-existente-no-problema
15      :type custom
16      :objects objeto-nao-existente-no-problema
17      :properties(
18          (prefabImage outra-imagem-em-Base64)
19          (showName FALSE)
20          (x 0)
21          (y 0)
22          (color WHITE)
23          (width 100)
24          (height 100)
25          (depth 0)
26      )
27  )
```

Código 2.13 – Código de exemplo para um elemento visual

2.4.4 Imagens

O bloco de imagens existe como um tipo de dicionário, no qual a primeira sequência de caracteres é a chave e a segunda sequência de caracteres que é a imagem representadas na Base64, como exemplificado no Código 2.14

```
1    (:image
2        (imagem-em-Base64 iVBORw0KGgoAAAANSUhEUgAA...
3        (outra-imagem-em-Base64 iVBORw0KGgoAAAANSUhEUgAAACgA...
4        ...
5    )
```

Código 2.14 – Código de exemplo para as imagens

3 Trabalhos Relacionados

Há uma série de trabalhos relacionados, que resolvem problemas similares, por exemplo, o artigo *"A Good Snowman is Hard to Plan"* (BOFILL et al., 2023) que resolve o jogo *A Good Snowman is Hard to Build*, que envolve mover e empilhar bolas de neve em uma grade discreta para construir bonecos de neve. No artigo, o jogo foi resolvido com modelagem em PDDL e três variações foram testadas: modelo básico, modelo com teletransporte (cheating) e modelo com predicados derivados para alcançabilidade; foi resolvido planejamento como SAT e isso permitiu resolver mais instâncias de forma otimizada, especialmente ao modelar a alcançabilidade. Após a construção dos modelos, a abordagem de SAT com alcançabilidade mostrou-se mais eficiente.

O artigo *"Challenges in Modelling and Solving Plotting with PDDL"* (ESPASA; MIGUEL; VILLARET, 2022a), que resolve o jogo *Plotting*, no qual o objetivo é remover um número alvo de blocos coloridos de uma grade atirando blocos sequencialmente. O artigo mostra que a modelagem em PDDL é difícil devido à complexidade das transições de estado após cada tiro e às limitações da linguagem em lidar com operações aritméticas e variáveis multi-valoradas.

Outro artigo que procurou resolver o jogo *Plotting* é *"Plotting: A Planning Problem with Complex Transitions"* (ESPASA; MIGUEL; VILLARET, 2022b), nele o jogo é modelado em PDDL e em Essence Prime. Foram apresentados dois modelos em Essence Prime e depois comparados com o modelo em PDDL, em que os modelos em Essence Prime mostraram um desempenho melhor. Esses modelos utilizam variáveis de decisão para representar o estado da grade e a ação tomada em cada passo, permitindo uma representação mais concisa e eficiente das transições de estado.

A Monografia *"Resolvendo Pipe Mania como Planejamento"* (BANCI, 2022), que resolve o jogo Pipe Mania, que é um puzzle em que o jogador deve conectar pedaços de tubulação em uma grade para criar um caminho com um comprimento mínimo dentro de um tempo limitado. A monografia compara quatro métodos diferentes, o primeiro e o segundo usando algoritmos de grafos e o terceiro e quarto utilizando modelos de planejamento de IA. O resultado dessa comparação mostra que a modelagem com um dos algoritmos para grafos foi a mais eficiente em termos de tempo e memória.

A Monografia *"Resolvendo Puzznic através de planejamento automatizado"* (PEREIRA, 2023), que resolve o jogo Puzznic, nele o jogador deve combinar peças de cores e formatos correspondentes para eliminá-las do tabuleiro, com o objetivo final de limpar todas as peças dentro de um tempo limite. O trabalho resolve o jogo utilizando a linguagem PDDL e o planejador ForwardPlanner na linguagem Julia, além de realizar

experimentos para validar o modelo, mostrando que ele representa adequadamente o jogo dentro do escopo planejado.

4 Xian-Xiang

Xian-Xiang é um jogo de quebra-cabeça desenvolvido pela Motion-Twin[1] e era disponibilizado no site KadoKado. A cada partida, o jogador recebe um tabuleiro com 42 peças dispostas em 7 linhas e 6 colunas, conforme o exemplo da Figura 1. O objetivo do jogo é eliminar todas as peças do tabuleiro ao combinar duas por vez.

Figura 1 – Exemplo de tabuleiro de Xian-Xiang no início da partida.

4.1 Regras do Jogo

As regras disponibilizadas do jogo são:

[1] A MOTION TWIN é um pequeno estúdio de desenvolvimento de jogos sediado na França. Organizado como uma cooperativa de trabalhadores, sem hierarquia, eles escolhem seus projetos, seus objetivos e sua estrutura de trabalho em conjunto. A MOTION TWIN busca criar os jogos que eles gostariam de jogar, concentrando-se na experiência do jogador.

- Combine peças similares ao clicar em duas;
- Pontuação:
  - Quanto mais atributos (cor, forma e simbolo) as peças compartilharem, mais pontos serão marcados;
  - Combinar duas peças com nenhum atributo igual: 1 ponto;
  - Combinar duas peças com 1 atributo igual: 50 pontos;
  - Combinar duas peças com 2 atributos iguais: 300 pontos; e
  - Combinar duas peças com 3 atributos iguais: 1000 pontos.

Tais atributos consideram os parâmetros de forma, cor e símbolo, conforme elencado na Tabela 1, a seguir:

FORMA	**COR**	**SÍMBOLO**
Triângulo	Vermelho	Quadrado
Quadrado	Verde	Quatro-pontos
Círculo	Azul	Cruz
-	Amarelo	Triângulo
-	-	Ondas

Tabela 1 – Tabela de atributos de cada peça.

A combinação de peças deve ser feita utilizando uma reta, seja horizontal ou vertical, ou por uma delas somada a uma perpendicular, sendo a junção de dois segmentos de reta (formato de "L"), tais quais mostram as Figuras 2, 3 e 4.

Figura 2 – Exemplo de combinação horizontal.

Figura 3 – Exemplo de combinação vertical.

Além disso, o tamanho de qualquer reta ou segmentos de reta possui apenas duas limitações, quais sejam: a) atravessar uma ou mais peças, conforme visto na Figura 5; e/ou b) as dimensões do tabuleiro, como demonstrado na Figura 6, observe-se:

Figura 4 – Exemplo de combinação com dois segmentos de reta.

Figura 5 – Exemplo de combinação inválida.

Figura 6 – Demonstração de limitação.

Tendo determinado o conjunto de regras a serem observadas para resolução do jogo Xian-Xiang, torna-se possível a formulação em PDDL.

5 Formulação em PDDL

Para a formulação em PDDL, foi seguida uma abordagem de considerar peça a peça, ignorando seus atributos de cor, forma e desenho, mas considerando a quantidade de atributos iguais entre elas e suas posições, isso para retirar as condições que seriam necessárias dentro das ações e evitar um grande número de predicados.

5.1 Objetos e Predicados

Para esse problema, não foi necessário criar nenhum tipo de objeto personalizado, tendo em vista que os únicos objetos são as peças dispostas no tabuleiro.

Inicialmente, para determinar a posição de cada peça, foram criados dois predicados:

- `(connected-horizontal ?p1 ?p2)`: que determina que uma peça está ao lado da outra;
- `(connected-vertical ?p1 ?p2)`: que determina que uma peça está acima ou abaixo da outra.

Após o início da partida, esses predicados também assumem o parte do papel de abstrair o desbloqueio das combinações, ou seja, as peças precisam estar conectadas ou subconectadas para serem combinadas, conforme é explicado a seguir com os demais predicados:

- `(connected-horizontal ?p1 ?p2)`: que determina que as peças conectadas podem ser combinadas horizontalmente;
- `(connected-vertical ?p1 ?p2)`: que determina que as peças conectadas podem ser combinadas verticalmente;
- `(subconnected ?p1 ?p2)`: que determina que as peças podem ser combinadas perpendicularmente, em “L”;

Explicados os predicados responsáveis pelo desbloqueio das jogadas, encontram-se também presentes no escopo os predicados que descrevem a semelhança entre as peças combinadas, eles são:

- `(three ?p1 ?p2)`: que determina que as peças têm três atributos iguais;

- `(two ?p1 ?p2)`: que determina que as peças têm dois atributos iguais;
- `(one ?p1 ?p2)`: que determina que as peças têm um atributo igual;
- `(zero ?p1 ?p2)`: que determina que as peças não têm atributos iguais;
- `(used ?p)`: que determina que a peça já foi combinada com outra.

Postos os objetos e predicados, faz-se necessário observar antes das ações, o sistema de pontuação que se dá por meio de função.

5.2 Função

Para determinar a pontuação, foi utilizada uma função, a qual é uma variável numérica.

Devido à escolha do planejador, que somente aceita uma sintaxe de somas e minimização da função, foi empregada uma lógica que soma custos. Dessa forma considera-se que qualquer jogada receberá um custo igual à pontuação da melhor jogada (a qual possui três atributos iguais) subtraída a pontuação da jogada efetuada. Assim, pode-se deduzir que quanto melhor a jogada, menos será acrescido à função utilizada. Percebe-se também que a melhor jogada possível custa 0.

Considerando a explicação acima, deve ser destacado que a pontuação final não é dada pela soma dos custos, mas sim pela equação a seguir:

$$PF = QC \cdot PJO - SDC \tag{5.1}$$

em que PF é a pontuação final, QC é a quantidade de combinações, PJO é a pontuação da melhor jogada e SDC é a soma dos custos.

A soma dos custos é representada no domínio pela função **(total-cost)**, presente no domínio conforme o Código 5.1:

```
1    (:functions
2        (total-cost) - number
3    )
```

Código 5.1 – Código da função que soma os custos do plano

Indicada a função e sua influência na pontuação final, cabe explanar as ações.

5.3 Ações

As ações modificam os predicados e a função, elas podem ser divididas em dois grupos. O primeiro garante que as regras do jogo serão respeitadas, e o segundo grupo que fragmenta a ação de combinar as peças.

5.3.1 Ações que garantem as regras do jogo

Essas ações garantem as regras do jogo limitando as jogadas para que não ocorram jogadas em mais de dois fragmentos de reta ou para que não ocorram jogadas que atravessam peças. As ações desse grupo são três:

- `CONNECT-HORIZONTAL`: que possibilita a combinação entre duas peças conectando-as horizontalmente; essa ação ocorre com três peças como parâmetro, as quais são *?p1*, *?p2* e *?p3*, e tem como precondição que a *?p1* já foi combinada, conectando as peças que estão à sua direita e à sua esquerda, criando assim um caminho válido para uma jogada horizontal, claro que deve-se salientar que mesmo *?p2* e/ou *?p3* podem já ter sido combinadas com outras peças e que essa conexão atuaria como intermediária para outra conexão futura. O código para essa para essa ação está no Código 5.2

```
1   (:action CONNECT-HORIZONTAL
2       :parameters (?p1 ?p2 ?p3)
3       :precondition (and
4           (connected-horizontal ?p1 ?p2)
5           (connected-horizontal ?p1 ?p3)
6           (used ?p1)
7           (not (= ?p1 ?p2))
8           (not (= ?p1 ?p3))
9           (not (= ?p2 ?p3))
10      )
11      :effect (and
12          (connected-horizontal ?p2 ?p3)
13          (connected-horizontal ?p3 ?p2)
14      )
15  )
```

Código 5.2 – Código da ação que garante as regras para jogadas horizontais.

- `CONNECT-VERTICAL`: que possibilita a combinação entre duas peças conectando-as verticalmente; tal qual a ação explanada anteriormente, essa ação ocorre com três peças como parâmetro, as quais são *?p1*, *?p2* e *?p3*, e tem como precondição que a *?p1* já foi combinada, conectando as peças que estão acima e abaixo, criando assim

um caminho válido para uma jogada vertical, claro que deve-se salientar que mesmo *?p2* e/ou *?p3* podem já ter sido combinadas com outras peças e que essa conexão atuaria como intermediária para outra conexão futura. O código para essa para essa ação está no Código 5.3.

```
(:action CONNECT-VERTICAL
    :parameters (?p1 ?p2 ?p3)
    :precondition (and
        (connected-vertical ?p1 ?p2)
        (connected-vertical ?p1 ?p3)
        (used ?p1)
        (not (= ?p1 ?p2))
        (not (= ?p1 ?p3))
        (not (= ?p2 ?p3))
    )
    :effect (and
        (connected-vertical ?p2 ?p3)
        (connected-vertical ?p3 ?p2)
    )
)
```

Código 5.3 – Código da ação que garante as regras para jogadas verticais.

- `SUBCONNECT`: que possibilita a combinação entre duas peças conectando-as por dois segmentos de retas; tal qual as ações explanada anteriormente, essa ação ocorre com três peças como parâmetro, as quais são *?p1*, *?p2* e *?p3*, e tem como precondição que a *?p1* já foi combinada, sub-conectando uma peça que está conectada a *?p1* horizontalmente e outra que está conectada verticalmente, criando assim um caminho válido para uma jogada com dois segmentos de reta, claro que deve-se salientar que diferentemente das outras ações, essa conexão não atua como intermediária para outra conexão futura. O código para essa para essa ação está no Código 5.4.

```
(:action SUBCONNECT
    :parameters (?p1 ?p2 ?p3)
    :precondition (and
        (connected-horizontal ?p1 ?p2)
        (connected-vertical ?p1 ?p3)
        (used ?p1)
        (not (= ?p1 ?p2))
        (not (= ?p1 ?p3))
        (not (= ?p2 ?p3))
```

```
10          )
11          :effect (and
12              (subconnected ?p2 ?p3)
13              (subconnected ?p3 ?p2)
14          )
15      )
```

Código 5.4 – Código da ação que garante as regras para jogadas com dois segmentos de reta.

5.3.2 Ações que combinam peças

As ações que combinam as peças são as que simulam as ações do jogador, para isso a ação de combinar foi fragmentada em quatro ações diferentes como demonstrado a seguir:

- `USE-THREE`: ação para as jogadas entre peças com três atributos iguais (por ser a melhor jogada, não tem custo). O código para essa ação pode ser encontrado no Código 5.5, deve-se destacar que foi colocado nesse código uma soma de 0 à função de custo por caráter didático e não modifica o custo real.

```
1       (:action USE-THREE
2           :parameters (?p1 ?p2)
3           :precondition (and
4               (not (used ?p1))
5               (not (used ?p2))
6               (or
7                   (connected-horizontal ?p1 ?p2)
8                   (connected-vertical ?p1 ?p2)
9                   (subconnected ?p1 ?p2)
10              )
11              (three ?p1 ?p2)
12          )
13          :effect (and
14              (used ?p1)
15              (used ?p2)
16              (increase (total-cost) 0)
17          )
18      )
```

Código 5.5 – Código da ação que combina as peças com três atributos iguais.

- `USE-TWO`: ação para as jogadas entre peças com dois atributos iguais (por ter uma pontuação de 700 a menos do que a melhor jogada, custa 700). O código para essa ação pode ser encontrado no Código 5.6.

```
(:action USE-TWO
    :parameters (?p1 ?p2)
    :precondition (and
        (not (used ?p1))
        (not (used ?p2))
        (or
            (connected-horizontal ?p1 ?p2)
            (connected-vertical ?p1 ?p2)
            (subconnected ?p1 ?p2)
        )
        (two ?p1 ?p2)
    )
    :effect (and
        (used ?p1)
        (used ?p2)
        (increase (total-cost) 700)
    )
)
```

Código 5.6 – Código da ação que combina as peças com dois atributos iguais.

- `USE-ONE`: ação para as jogadas entre peças com um atributo igual (por ter uma pontuação de 950 a menos do que a melhor jogada, custa 950). O código para essa ação pode ser encontrado no Código 5.7.

```
(:action USE-ONE
    :parameters (?p1 ?p2)
    :precondition (and
        (not (used ?p1))
        (not (used ?p2))
        (or
            (connected-horizontal ?p1 ?p2)
            (connected-vertical ?p1 ?p2)
            (subconnected ?p1 ?p2)
        )
        (one ?p1 ?p2)
    )
    :effect (and
```

```
14              (used ?p1)
15              (used ?p2)
16              (increase (total-cost) 950)
17          )
18      )
```

Código 5.7 – Código da ação que combina as peças com um atributo igual.

- `USE-ZERO`: para as jogadas entre peças com nenhum atributo igual (por ter uma pontuação de 999 a menos do que a melhor jogada, custa 999). O código para essa ação pode ser encontrado no Código 5.8.

```
1       (:action USE-ZERO
2           :parameters (?p1 ?p2)
3           :precondition (and
4               (not (used ?p1))
5               (not (used ?p2))
6               (or
7                   (connected-horizontal ?p1 ?p2)
8                   (connected-vertical ?p1 ?p2)
9                   (subconnected ?p1 ?p2)
10              )
11              (zero ?p1 ?p2)
12          )
13          :effect (and
14              (used ?p1)
15              (used ?p2)
16              (increase (total-cost) 999)
17          )
18      )
```

Código 5.8 – Código da ação que combina as peças com nenhum atributo igual.

Detalhados os aspectos relevantes do domínio, cabe mencionar o gerador de problema.

5.4 Gerador de problema

Visando a facilitar a experimentação dessa solução para o jogo Xian-Xiang, foi criado um gerador de arquivo de problema, devido a quantidade de mapas testados e a complexidade de cada arquivo de problema, que contam com a relação de atributos de todas as peças entre si, para isso foi escrito o trecho de Código 5.9.

```
    vector<string> pontuacoes;
    pontuacoes.push_back("zero");
    pontuacoes.push_back("one");
    pontuacoes.push_back("two");
    pontuacoes.push_back("three");
    int pontuacao;
    for (int i = 0; i < vetor.size(); i++)
        for (int j = i + 1; j < vetor.size(); j++)
        {
            pontuacao = 0;
            shape[i] == shape[j] ? pontuacao++ : pontuacao += 0;
            color[i] == color[j] ? pontuacao++ : pontuacao += 0;
            symbol[i] == symbol[j] ? pontuacao++ : pontuacao += 0;
            cout << "(" << pontuacoes[pontuacao] << " p" << vetor[i] <<
               " p" << vetor[j] << ") ";
        }
    printf("\n\t\t(= (total-cost) 0)\n");
```

Código 5.9 – Código em C++ utilizado para gerar arquivo de problema

Como entrada para o gerador de arquivos de problema, foi utilizado um arquivo que descreve o mapa em questão como o exemplo a seguir, no qual há o tamanho do mapa seguido pela descrição de cada peça de cima para baixo e da esquerda para direita:

```
6 7
circulo amarelo triangulo
quadrado amarelo quadrado
quadrado amarelo triangulo
circulo vermelho quadrado
quadrado vermelho quatro-pontos
circulo vermelho triangulo
...
```

Esses arquivos que descrevem os mapas foram criados, porque essa informação também é necessária para o gerador de animação que considera os atributos de cada peça.

Logo, contando com o domínio e problema, convém descrever também o gerador de animação desenvolvido para esta modelagem.

5.5 Gerador de animação

Com o objetivo de visualizar a solução para um problema, foi produzido um gerador de arquivo-base de animação que, partindo do mesmo arquivo de entrada para o gerador de problema, gera um arquivo-base de animação que é utilizado com o Planimation, mas

diferente do gerador de problema, em que a ordem dos atributos não importa, o gerador de arquivo base para animação necessita que a ordem dos atributos no arquivo de entrada seja forma, cor e desenho para as peças serem representadas corretamente, além de todas as letras serem minusculas e sem acentos. Como referência pode-se seguir a tabela 2:

Forma	Cor	Desenho
circulo	amarelo	cruz
quadrado	azul	ondas
triangulo	verde	quadrado
	vermelho	quatro-pontos
		triangulo

Tabela 2 – Tabela de referência para criação de arquivo de entrada.

Seguindo com o resultado de empregar o Planimation, três figuras que exemplificam como fica a animação, a Figura 7 para o início, a Figura 8 para o meio e a Figura 9 para o fim. Mesmo que no jogo original as peças só desapareçam ao serem jogadas, para facilitar o entendimento de cada jogada, foram modelados dois tabuleiros diferentes, um para as peças não combinadas (à esquerda) e outro para as peças combinadas (à direita).

Pode-se acessar o exemplo das imagens clicando no link[1].

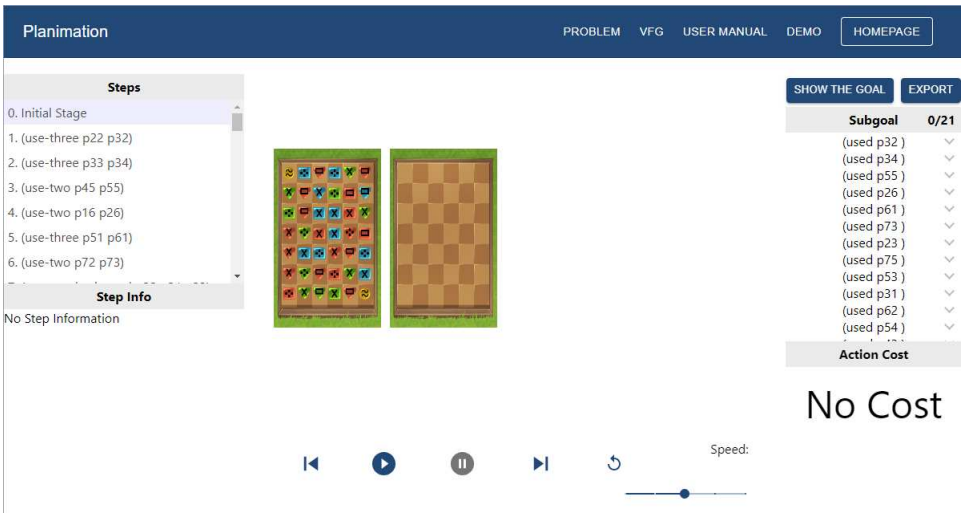

Figura 7 – Inicio da Animação.

[1] https://editor.planning.domains/#read_session=rq2ebGpOaE

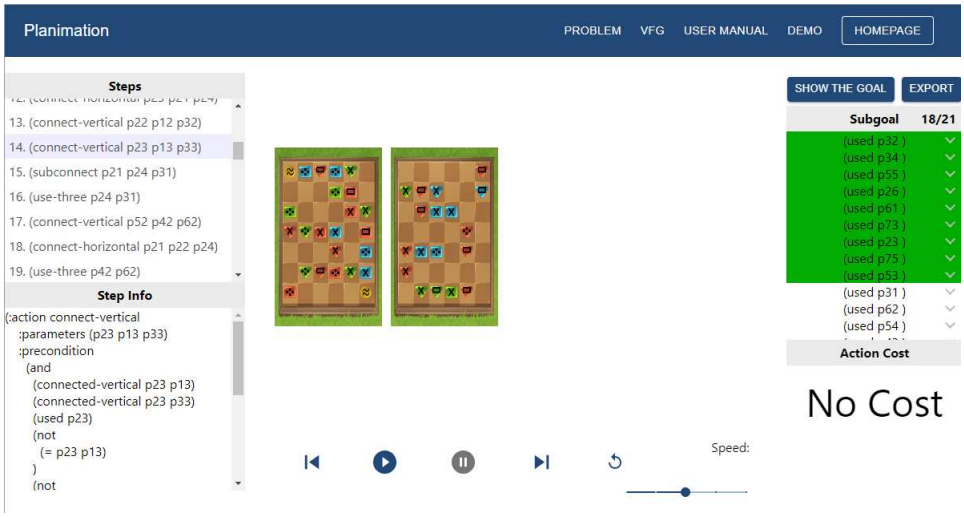

Figura 8 – Meio da Animação.

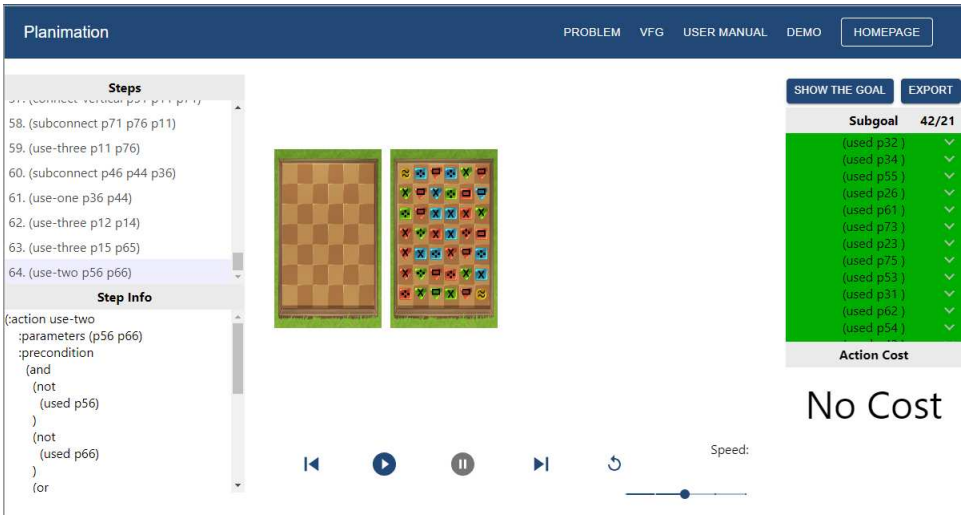

Figura 9 – Fim da animação.

O arquivo-base pode ser explicado considerando quatro partes diferentes:

5.5.1 Predicados

Mesmo contando com diversos predicados no domínio, o único que gera alguma consequência na animação é o predicado "used", utilizado quando uma peça é combinada com outra, ele tem um efeito de mover as peças de um tabuleiro para o outro. Para isso foi utilizado o Código 5.10.

```
1  (:predicate used
2    :parameters (?p)
3    :effect (and
```

```
4       (equal (?p x) (add (?p x) 130))
5     )
6   )
```

Código 5.10 – Código do predicado que movimenta as peças combinadas na animação.

5.5.2 Tabuleiros

Os 2 tabuleiros presentes na animação são inexistentes no problema e no domínio, então são criados tendo o tipo "custom" e variam somente na propriedade "x", assim colocando-os lado a lado. O código para um deles está 5.11:

```
1   (:visual Board
2       :type custom
3       :objects Board
4       :properties(
5           (prefabImage img-square)
6           (showName FALSE)
7           (x 130)
8           (y 0)
9           (color WHITE)
10          (width 120)
11          (height 200)
12          (depth 0)
13      )
14  )
```

Código 5.11 – Exemplo de código para criação do tabuleiro do jogo na animação.

5.5.3 Peças

As 42 peças foram modeladas individualmente seguindo o modelo a seguir no Código 5.12, tendo o tipo "predefine", o que significa que está diretamente ligada a um objeto do problema, além de variar nas demais propriedades de posicionamento e imagem da peça.

```
1   (:visual p11
2     :type predefine
3     :objects (p11)
4     :properties(
5       (prefabImage img-amarelo-quadrado-quadrado)
6       (showName FALSE)
7       (x 9)
```

```
8       (y 161)
9       (color WHITE)
10      (width 16)
11      (height 20)
12      (depth 1000)
13    )
14  )
```

Código 5.12 – Exemplo de código para criação das peças do jogo na animação.

5.5.4 Imagens

Já a parte "image" contém as imagens utilizadas no problema, as quais são as imagens das peças e do tabuleiro. Cada imagem é nomeada e definida por uma sequência de caracteres que são imagens representadas na Base64, como demonstrado no Código 5.13.

```
1     (:image
2         (img-amarelo-circulo-quatro-pontos iVBORw0KGgoAAAANSUhEUgAA...
3         (img-amarelo-circulo-triangulo iVBORw0KGgoAAAANSUhEUgAAACgA...
4         (img-amarelo-quadrado-quadrado iVBORw0KGgoAAAANSUhEUgAAACgA...
5         (img-amarelo-quadrado-triangulo iVBORw0KGgoAAAANSUhEUgAAACg...
6         (img-amarelo-triangulo-quadrado iVBORw0KGgoAAAANSUhEUgAAACg...
7         ...
8     )
```

Código 5.13 – Trecho de código das imagens presentes na animação

Isto posto, tendo os elementos aptos à resolução da problemática, bem como uma hipótese para sua aplicação, convém realizar experimentos acerca de sua funcionalidade.

6 Experimentos

Após a construção do domínio, dos problemas e animações, foram realizados dois tipos de experimentos diferentes, um para testar a validade da modelagem feita, e outro para testar como o modelo se comporta com abordagens diferentes de planejamento.

Para esses experimentos foi utilizado o planejador Fast Downward (HELMERT, 2006), que foi escolhido por contar com configurações de minimizações de custo, e de heurísticas otimizadas para competições, chamadas de *alias*, dentre as quais foi selecionada a "seq-sat-fdss-2018" (SEIPP; RÖGER, 2018) como base para ambos experimentos.

Para a realização dos experimentos, foi utilizada uma máquina com as seguintes especificações técnicas:

- Arquitetura: x86_64, com suporte para modos de operação de 32 e 64 bits.
- Processador: AMD Ryzen 7 2700 Eight-Core Processor, com 16 núcleos (8 físicos e 8 lógicos) e frequência máxima de 3200 MHz.
- Memória: 32 GB de RAM, com 20 GB de swap disponível.
- Cache: L1d: 256 KiB, L1i: 512 KiB, L2: 4 MiB, L3: 16 MiB.
- Sistema Operacional: Ubuntu 22.04.4 LTS (Jammy).
- Recursos de Virtualização: AMD-V, com suporte a várias extensões e mitigação de vulnerabilidades de segurança.

Essas especificações garantem um ambiente robusto e seguro para a execução dos experimentos, proporcionando desempenho e confiabilidade necessários para a obtenção de resultados precisos.

6.1 Validade da Modelagem

Após a modelagem, é necessário validar o domínio, para isso foram considerados mapas com condições específicas, assim testando se a modelagem não resulta em planos inválidos ou se não prejudica a busca por planos ótimos.

Foram gerados 5 mapas para testar o comportamento do planejador, sendo os dois no quais as peças com três atributos iguais estão em posições favoráveis para combinação, e os três seguintes para descobrir o comportamento quando peças ótimas estão em posições mais distantes, isso devido a preocupação que a necessidade de criar um conexão entre as

peças antes de uma combinação pudesse gerar um impedimento na procura pelo melhor resultado. O resultado está descrito abaixo na Tabela 3,

Mapa	**Tempo Utilizado**	**Descrição**	**Resultado encontrado**
Mapa 1	224s	Mapa favorável para combinações, dado que as colunas são espelhadas a partir do meio para as extremidades.	O planejador encontrou a solução ótima.
Mapa 2	145s	Mapa favorável para combinações em que as peças ideais para uma combinação estão uma ao lado da outra.	O planejador encontrou a solução ótima.
Mapa 3	20 min	Mapa em que a primeira e a última peça da primeira coluna são iguais. O objetivo era verificar se essas peças se combinariam mesmo com a distância entre elas, caso a combinação fosse uma boa jogada.	A primeira e a última peça da primeira coluna foram combinadas.
Mapa 4	20 min	Mapa em que a primeira e a última peça da primeira linha são iguais. O objetivo era verificar se essas peças se combinariam mesmo com a distância entre elas, caso a combinação fosse uma boa jogada.	A primeira e a última peça da primeira linha foram combinadas.
Mapa 5	20 min	Mapa em a primeira e a última peças são as únicas totalmente semelhantes entre si. O objetivo era verificar se essas peças se combinariam mesmo com a distância entre elas, caso a combinação fosse uma boa jogada.	A primeira e a última peça foram combinadas.

Tabela 3 – Tabela com resultados para testes com mapas específicos.

Os resultados para esses 5 mapas mostraram o êxito do domínio ao modelar o problema, visto que os experimentos para os mapas de fácil solução resultaram em pla-

nos ótimos, enquanto os experimentos nos quais as peças com ótimas combinações estão distantes entre si tiveram essas peças combinadas, demonstrando que a necessidade de conectar as peças antes da combinação não é o principal fator que interfere na procura do plano.

6.2 Teste para diversas abordagens

O objetivo desse experimento foi testar diversas abordagens considerando vários planejadores e/ou várias configurações de um mesmo planejador visando encontrar uma abordagem que tivesse o melhor desempenho médio, assim foram utilizadas oito abordagens diferentes, para procurar planos que minimizassem o custo das jogadas utilizando a função descrita na seção 5.2.

As abordagens escolhidas para teste foram, em sua maioria, abordagens que visam diminuir o custo, que são "seq-sat-fdss-2018" (SEIPP; RÖGER, 2018), "seq-sat-fdss-2023" (BÜCHNER et al., 2023), "seq-sat-lama-2011" (RICHTER; WESTPHAL; HELMERT, 2011), "seq-sat-maidu" (CORRÊA et al., 2014), "scorpion" (SEIPP, 2023), "seq-opt-odin" (DREXLER; SEIPP; SPECK, 2023) e já as outras focam em encontrar uma solução rapidamente, sendo incluídas à comparação por curiosidade, são "madagascar" (RINTANEN, 2014) e "lama-first" (RICHTER; WESTPHAL; HELMERT, 2011).

Foi criado um conjunto de 100 mapas aleatórios com um gerador de mapas aleatórios, tais quais eram os do jogo original, e dado o tempo limite de 20 minutos de procura para cada abordagem em cada mapa, sendo ele utilizado integralmente por todas abordagens, exceto as que têm foco encontrar planos com rapidez. Para isso foi acrescido o parâmetro "–search-time-limit 1200", que delimita o tempo de procura do plano enquanto não contabiliza o tempo de leitura dos arquivos ou qualquer outro tipo de pré-processamento. Além desse parâmetro, não houve nenhum outro parâmetro que modificasse ou limitasse qualquer planejador.

A Tabela 4 apresenta o total da soma dos custos dos melhores planos encontrados. A Tabela 6 apresenta cada custo encontrado por mapa e abordagem.

Planejador	Soma dos Custos
seq-sat-fdss-2018	1128585
lama-first	1902252
madagascar	1870104
seq-sat-fdss-2023	1364598
seq-sat-lama-2011	1324812
seq-sat-maidu	1212235

Tabela 4 – Tabela do total da soma de todos os custos encontrados.

Após a experimentação, foram obtidas pontuações para somente 6 planejadores, pois dois planejadores, "seq-opt-odin" e "scorpion", por terem o foco em somente retornar o plano ótimo para cada mapa, não geraram nenhum plano devido ao tempo limite então não entraram na tabela.

Mesmo que seja possível avaliar que os custos encontrados pelos planejadores focados em encontrar uma solução rapidamente estão acima dos demais, foi adotado, para avaliar o desempenho das abordagens, o mesmo método utilizado na International Planning Competition (IPC), em que um planejador é utilizado como referência, no caso foi selecionado o *alias* "seq-sat-fdss-2018" arbitrariamente, para comparação entre os demais realizando o seguinte cálculo:

$$C^*/C$$

Em que para cada mapa $C*$ é valor de referência e C é o valor obtido pelo planejador avaliado. Após a comparação entre todos os mapas, há a soma da pontuação total para comparação.

Na Tabela 5 há o resultado das comparações entre as abordagens e a referência mapa a mapa e o total ao final, é importante salientar que nessa tabela há a pontuações expressas, então quanto maior melhor, diferente da Tabela 6, na qual estão descritos os custos encontrados para cada solução. Na Tabela 7 apresenta cada pontuação por mapa e abordagem.

Planejador	Soma das Comparações
seq-sat-fdss-2018	100,00
lama-first	59,35
madagascar	60,40
seq-sat-fdss-2023	83,42
seq-sat-lama-2011	87,49
seq-sat-maidu	93,52

Tabela 5 – Tabela do total da soma das pontuações encontradas.

Ao analisar o total, percebe-se que mesmo com planejadores mais atuais, como "seq-sat-fdss-2023" ou "seq-sat-maidu", o próprio "seq-sat-fdss-2018" se saiu melhor no geral. Faz-se ainda uma menção honrosa ao "seq-sat-lama-2011" por encontrar um plano com maior pontuação, pontuando assim 1,50 no Mapa 78.

7 Conclusão e Trabalhos Futuros

Neste trabalho foi descrita uma modelagem para solucionar o jogo Xian-Xiang sob uma abordagem de planejamento com PDDL. O domínio foi avaliado utilizando 105 mapas diferentes. Desses, 100 foram criados aleatoriamente, tais quais eram os mapas gerados pelo jogo original, e experimentados em oito planejadores diferentes, e 5 foram criados com propósitos específicos, de testar dois casos fáceis, um que teria as colunas espelhadas a partir do meio para as extremidades e outro em que as peças ideais para cada combinação estão lado a lado, além de três casos de borda, para testar o comportamento quando as peças que realizariam jogadas ótimas estiverem em extremos do mapa. Estas últimas experimentações com os 5 mapas criados foram feitas com somente um planejador e não apresentaram falhas no domínio.

Foram encontrados planos que resolvem todos os mapas criados, sejam aleatórios ou não. Os resultados apresentados nas Tabelas 3 e 6 mostraram uma dificuldade do planejador em encontrar ou, ainda, provar um plano ótimo para problemas aleatórios e de teste de comportamento, utilizando a formulação mencionada. Isso demonstra a complexidade dos problemas reais, tendo em vista que para casos fáceis (de menor complexidade) a melhor solução foi encontrada no tempo estipulado, como se pode observar nas duas primeiras linhas da Tabela 3, enquanto para os aleatórios, os quais são mais difíceis, o processo alcançou o objetivo de resolução dos problemas, porém não há garantia de melhor solução possível no tempo limite de 20 minutos, conforme se percebe ao analisar a Tabela 6.

Também foi criado nesse trabalho um gerador de arquivo-base para a animação de um plano, arquivo-base que utilizado como entrada junto aos arquivos de domínio e problema para a extensão Planimation, pode auxiliar na visualização do plano encontrado.

Destarte, fica proposto para um trabalho futuro, modelar uma solução seguindo um algoritmo de busca em grafos, em que cada estado seria passado para a busca, enquanto as ações modificariam os estados. Assim, o próprio grafo seria gerado ao longo da própria busca.

Referências

BANCI, G. A. D. Resolvendo pipe mania como planejamento. 2022. 56 f., il. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Software) — Universidade de Brasília, Brasília, 2022. Disponível em: <https://bdm.unb.br/handle/10483/33938>. Citado na página 22.

BOFILL, M. et al. A good snowman is hard to plan. 2023. Disponível em: <https://icaps23.icaps-conference.org/program/workshops/keps/KEPS-23_paper_9560.pdf>. Citado na página 22.

BÜCHNER, C. et al. Fast downward stone soup 2023. University of Basel, 2023. Citado na página 41.

CHEN, G. et al. *Planimation*. arXiv, 2020. ArXiv:2008.04600 [cs]. Disponível em: <http://arxiv.org/abs/2008.04600>. Citado 2 vezes nas páginas 12 e 18.

CORRÊA, A. B. et al. Scorpion maidu: Width search in the scorpion planning system. In: *Conference on Artificial Intelligence (ECAI 2012)*. [S.l.: s.n.], 2014. v. 540, p. 545. Citado na página 41.

DREXLER, D.; SEIPP, J.; SPECK, D. Odin: A planner based on saturated transition cost partitioning. In: *Tenth International Planning Competition (IPC-10): Planner Abstracts*. [S.l.: s.n.], 2023. Citado na página 41.

ESPASA, J.; MIGUEL, I.; VILLARET, M. Plotting: A Planning Problem with Complex Transitions. In: SOLNON, C. (Ed.). *28th International Conference on Principles and Practice of Constraint Programming (CP 2022)*. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. (Leibniz International Proceedings in Informatics (LIPIcs), v. 235), p. 22:1–22:17. ISBN 978-3-95977-240-2. ISSN 1868-8969. Disponível em: <https://drops.dagstuhl.de/opus/volltexte/2022/16651>. Citado na página 22.

ESPASA, J.; MIGUEL, I.; VILLARET, M. Plotting: A Planning Problem with Complex Transitions. *LIPIcs, Volume 235, CP 2022*, v. 235, p. 22:1–22:17, 2022. ISSN 1868-8969. Artwork Size: 17 pages, 981797 bytes ISBN: 9783959772402 Medium: application/pdf Publisher: Schloss Dagstuhl – Leibniz-Zentrum für Informatik. Disponível em: <https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CP.2022.22>. Citado na página 22.

GEFFNER, H. Perspectives on Artificial Intelligence Planning. p. 1013–1023, 2002. Citado na página 13.

HELMERT, M. The fast downward planning system. *Journal of Artificial Intelligence Research*, v. 26, p. 191–246, 2006. Citado na página 39.

HENDLER, J. A.; TATE, A.; DRUMMOND, M. Ai planning: Systems and techniques. *AI magazine*, v. 11, n. 2, p. 61–61, 1990. Citado na página 13.

MCDERMOTT, D. et al. Pddl - the planning domain definition language. 08 1998. Disponível em: <https://api.semanticscholar.org/CorpusID:59656859>. Citado 2 vezes nas páginas 12 e 14.

MCDERMOTT, D. M. The 1998 ai planning systems competition. *AI Magazine*, v. 21, n. 2, p. 35, Jun. 2000. Disponível em: <https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1506>. Citado 2 vezes nas páginas 12 e 13.

PEREIRA, S. d. S. B. Resolvendo puzznic através de planejamento automatizado. 2023. 62 f., il. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Software) — Universidade de Brasília, Brasília, 2023. Disponível em: <https://bdm.unb.br/handle/10483/39173>. Citado na página 22.

RICHTER, S.; WESTPHAL, M.; HELMERT, M. Lama 2008 and 2011. In: ICAPS FREIBURG, GERMANY. *International Planning Competition*. [S.l.], 2011. p. 117–124. Citado na página 41.

RINTANEN, J. Madagascar: Scalable planning with sat. *Proceedings of the 8th International Planning Competition (IPC-2014)*, v. 21, p. 1–5, 2014. Citado na página 41.

SEIPP, J. Scorpion 2023. *Tenth International Planning Competition (IPC-10): Planner Abstracts*, 2023. Citado na página 41.

SEIPP, J.; RÖGER, G. Fast downward stone soup 2018. *IPC2018–Classical Tracks*, p. 72–74, 2018. Citado 2 vezes nas páginas 39 e 41.

SLEATH, K.; BERCHER, P. Detecting AI Planning Modelling Mistakes – Potential Errors and Benchmark Domains. In: LIU, F. et al. (Ed.). *PRICAI 2023: Trends in Artificial Intelligence*. Singapore: Springer Nature Singapore, 2024. v. 14326, p. 448–454. ISBN 978-981-9970-21-6 978-981-9970-22-3. Series Title: Lecture Notes in Computer Science. Disponível em: <https://link.springer.com/10.1007/978-981-99-7022-3_41>. Citado 3 vezes nas páginas 12, 13 e 18.

Apêndices

APÊNDICE A – Primeiro Apêndice

Mapa	seq-sat-fdss-2018	lama-first	madagascar	seq-sat-fdss-2023	seq-sat-lama-2011	seq-sat-maidu	seq-opt-odin	scorpion
Mapa 00	10599	18299	18348	10599	12598	10599	TLE	TLE
Mapa 01	12298	18843	19793	15099	15394	15799	TLE	TLE
Mapa 02	10050	18798	18298	13100	12298	12400	TLE	TLE
Mapa 03	11300	19994	19793	15300	12000	11750	TLE	TLE
Mapa 04	14399	19146	19048	15300	11148	14399	TLE	TLE
Mapa 05	10448	19044	18294	15149	16147	13199	TLE	TLE
Mapa 06	11598	18294	18798	14550	13994	12449	TLE	TLE
Mapa 07	11599	19994	19744	13699	14044	11599	TLE	TLE
Mapa 08	11999	19945	19847	13499	13347	12999	TLE	TLE
Mapa 09	12299	18945	19097	13199	12998	12950	TLE	TLE
Mapa 10	9949	19097	18245	14695	12594	13048	TLE	TLE
Mapa 11	9649	18896	18000	12900	7648	10349	TLE	TLE
Mapa 12	12400	19896	17946	13899	16844	12400	TLE	TLE
Mapa 13	11098	18994	18548	14599	11049	11348	TLE	TLE
Mapa 14	11349	19298	19249	11950	13396	11950	TLE	TLE
Mapa 15	11197	19543	17946	13199	14596	12749	TLE	TLE
Mapa 16	10149	19146	19146	12048	11046	12048	TLE	TLE
Mapa 17	11500	17799	18950	12400	11648	13400	TLE	TLE
Mapa 18	12200	19994	20097	14699	16746	12200	TLE	TLE
Mapa 19	11750	18994	19190	15800	14898	12749	TLE	TLE
Mapa 20	9199	17443	16546	14846	12996	15493	TLE	TLE
Mapa 21	11749	18646	17049	13698	15746	12900	TLE	TLE
Mapa 22	8000	17447	18049	10198	12548	8000	TLE	TLE
Mapa 23	10448	19646	19048	10849	12598	10800	TLE	TLE
Mapa 24	10800	19945	19396	13499	15296	10800	TLE	TLE
Mapa 25	11500	18548	19048	11950	13748	11799	TLE	TLE
Mapa 26	10599	18392	16746	13650	11598	11299	TLE	TLE
Mapa 27	12500	19646	19298	12549	14497	12500	TLE	TLE
Mapa 28	8200	18146	17696	13650	8147	9150	TLE	TLE

Mapa 29	11500	17897	18147	13199	9597	12000	TLE	TLE
Mapa 30	11897	19494	18896	12200	12900	12200	TLE	TLE
Mapa 31	11099	19146	18146	14750	14546	12400	TLE	TLE
Mapa 32	12450	19597	17446	12450	10948	12450	TLE	TLE
Mapa 33	8950	18744	19445	15796	13393	10649	TLE	TLE
Mapa 34	12000	18593	16898	14498	14149	12900	TLE	TLE
Mapa 35	12299	18798	18798	13699	11099	12700	TLE	TLE
Mapa 36	10845	19744	19597	12749	15042	12749	TLE	TLE
Mapa 37	13150	18843	18196	13400	14047	13400	TLE	TLE
Mapa 38	10198	18593	19293	17794	9048	11549	TLE	TLE
Mapa 39	10800	19543	19396	11549	12798	10800	TLE	TLE
Mapa 40	12900	18647	19396	15750	16397	13600	TLE	TLE
Mapa 41	10649	18544	18995	12200	9949	12200	TLE	TLE
Mapa 42	10149	18593	17843	13200	13745	10149	TLE	TLE
Mapa 43	12650	19396	19097	12650	14842	12650	TLE	TLE
Mapa 44	6800	19347	19249	16197	10948	7050	TLE	TLE
Mapa 45	11549	19097	19445	13650	13397	13400	TLE	TLE
Mapa 46	11700	18548	18048	14349	15243	11950	TLE	TLE
Mapa 47	12700	19592	18048	14148	15992	12700	TLE	TLE
Mapa 48	12400	17697	17398	13048	15046	13048	TLE	TLE
Mapa 49	11550	18646	18749	12450	10849	11550	TLE	TLE
Mapa 50	11549	19945	18593	13899	15046	13899	TLE	TLE
Mapa 51	13400	18896	19048	14850	16347	14350	TLE	TLE
Mapa 52	11999	18745	17295	11999	10497	11999	TLE	TLE
Mapa 53	12548	19396	19744	13699	9048	14149	TLE	TLE
Mapa 54	11750	19548	18499	14497	11746	13150	TLE	TLE
Mapa 55	10100	19445	19494	12249	11897	10100	TLE	TLE
Mapa 56	12499	19445	18847	12950	10547	12749	TLE	TLE
Mapa 57	9350	19249	18950	12650	14596	10349	TLE	TLE
Mapa 58	11500	18642	19396	13249	12147	11500	TLE	TLE
Mapa 59	11099	20391	17897	12048	13195	11799	TLE	TLE
Mapa 60	12900	19494	18298	15750	11349	13400	TLE	TLE
Mapa 61	8749	19146	19146	14100	8098	10350	TLE	TLE
Mapa 62	9400	19597	18999	13600	16946	10100	TLE	TLE
Mapa 63	10100	18999	18950	11749	14649	11749	TLE	TLE
Mapa 64	11950	19097	19298	14100	16643	13199	TLE	TLE
Mapa 65	10198	17348	18995	12098	15046	10198	TLE	TLE

Mapa 66	9650	18946	19847	12500	11198	12500	TLE	TLE
Mapa 67	12049	18847	18847	12450	14747	12049	TLE	TLE
Mapa 68	12250	19945	19396	13850	15398	13449	TLE	TLE
Mapa 69	12499	19347	19146	16246	15747	12499	TLE	TLE
Mapa 70	11500	17897	17750	12249	15898	11500	TLE	TLE
Mapa 71	10849	18147	19347	12700	15595	11050	TLE	TLE
Mapa 72	12000	19044	18397	14100	15295	12450	TLE	TLE
Mapa 73	10149	18397	17897	10600	11546	10149	TLE	TLE
Mapa 74	10849	18647	18495	12250	10297	11750	TLE	TLE
Mapa 75	13400	18798	18700	13400	15496	13400	TLE	TLE
Mapa 76	12299	17995	18749	13650	10997	13650	TLE	TLE
Mapa 77	13150	19592	19391	14649	16594	13199	TLE	TLE
Mapa 78	12099	19195	18593	12099	8049	12099	TLE	TLE
Mapa 79	11000	18891	19244	14399	15493	11000	TLE	TLE
Mapa 80	11099	19945	18696	15599	10596	12548	TLE	TLE
Mapa 81	12650	19293	19744	16848	15644	13150	TLE	TLE
Mapa 82	10800	18397	19097	14350	12946	11050	TLE	TLE
Mapa 83	13900	20244	19847	15599	15046	13900	TLE	TLE
Mapa 84	13600	19793	18641	15500	15496	14100	TLE	TLE
Mapa 85	11050	19494	18646	12949	12798	11050	TLE	TLE
Mapa 86	11750	19347	18896	16545	13400	12499	TLE	TLE
Mapa 87	12344	18196	18196	13449	13646	12344	TLE	TLE
Mapa 88	12999	19195	18450	14898	16495	14600	TLE	TLE
Mapa 89	11950	19744	19244	13649	12700	13649	TLE	TLE
Mapa 90	10300	19244	16594	17745	14346	16344	TLE	TLE
Mapa 91	8999	18745	17496	14698	8397	8999	TLE	TLE
Mapa 92	12250	19396	19695	13650	15644	12250	TLE	TLE
Mapa 93	10600	19195	18946	11849	13699	11050	TLE	TLE
Mapa 94	11500	18945	18945	14599	16446	11500	TLE	TLE
Mapa 95	10800	18700	18749	11049	9596	10349	TLE	TLE
Mapa 96	7750	20195	18848	11349	12046	9150	TLE	TLE
Mapa 97	11000	17099	18348	14796	12945	11000	TLE	TLE
Mapa 98	13150	18848	18299	14800	16397	14800	TLE	TLE
Mapa 99	8200	19249	17700	11750	10800	8900	TLE	TLE
TOTAL	1128585	1902252	1870104	1364598	1324812	1212235		

Tabela 6 – Tabela de custo para cada abordagem testada.

APÊNDICE B – Segundo Apêndice

Mapa	seq-sat-fdss-2018	lama-first	madagascar	seq-sat-fdss-2023	seq-sat-lama-2011	seq-sat-maidu
Mapa 00	1,00	0,58	0,58	1,00	0,84	1,00
Mapa 01	1,00	0,65	0,62	0,81	0,80	0,78
Mapa 02	1,00	0,53	0,55	0,77	0,82	0,81
Mapa 03	1,00	0,57	0,57	0,74	0,94	0,96
Mapa 04	1,00	0,75	0,76	0,94	1,29	1,00
Mapa 05	1,00	0,55	0,57	0,69	0,65	0,79
Mapa 06	1,00	0,63	0,62	0,80	0,83	0,93
Mapa 07	1,00	0,58	0,59	0,85	0,83	1,00
Mapa 08	1,00	0,60	0,60	0,89	0,90	0,92
Mapa 09	1,00	0,65	0,64	0,93	0,95	0,95
Mapa 10	1,00	0,52	0,55	0,68	0,79	0,76
Mapa 11	1,00	0,51	0,54	0,75	1,26	0,93
Mapa 12	1,00	0,62	0,69	0,89	0,74	1,00
Mapa 13	1,00	0,58	0,60	0,76	1,00	0,98
Mapa 14	1,00	0,59	0,59	0,95	0,85	0,95
Mapa 15	1,00	0,57	0,62	0,85	0,77	0,88
Mapa 16	1,00	0,53	0,53	0,84	0,92	0,84
Mapa 17	1,00	0,65	0,61	0,93	0,99	0,86
Mapa 18	1,00	0,61	0,61	0,83	0,73	1,00
Mapa 19	1,00	0,62	0,61	0,74	0,79	0,92
Mapa 20	1,00	0,53	0,56	0,62	0,71	0,59
Mapa 21	1,00	0,63	0,69	0,86	0,75	0,91
Mapa 22	1,00	0,46	0,44	0,78	0,64	1,00
Mapa 23	1,00	0,53	0,55	0,96	0,83	0,97
Mapa 24	1,00	0,54	0,56	0,80	0,71	1,00
Mapa 25	1,00	0,62	0,60	0,96	0,84	0,97
Mapa 26	1,00	0,58	0,63	0,78	0,91	0,94
Mapa 27	1,00	0,64	0,65	1,00	0,86	1,00
Mapa 28	1,00	0,45	0,46	0,60	1,01	0,90

Mapa 29	1,00	0,64	0,63	0,87	1,20	0,96
Mapa 30	1,00	0,61	0,63	0,98	0,92	0,98
Mapa 31	1,00	0,58	0,61	0,75	0,76	0,90
Mapa 32	1,00	0,64	0,71	1,00	1,14	1,00
Mapa 33	1,00	0,48	0,46	0,57	0,67	0,84
Mapa 34	1,00	0,65	0,71	0,83	0,85	0,93
Mapa 35	1,00	0,65	0,65	0,90	1,11	0,97
Mapa 36	1,00	0,55	0,55	0,85	0,72	0,85
Mapa 37	1,00	0,70	0,72	0,98	0,94	0,98
Mapa 38	1,00	0,55	0,53	0,57	1,13	0,88
Mapa 39	1,00	0,55	0,56	0,94	0,84	1,00
Mapa 40	1,00	0,69	0,67	0,82	0,79	0,95
Mapa 41	1,00	0,57	0,56	0,87	1,07	0,87
Mapa 42	1,00	0,55	0,57	0,77	0,74	1,00
Mapa 43	1,00	0,65	0,66	1,00	0,85	1,00
Mapa 44	1,00	0,35	0,35	0,42	0,62	0,96
Mapa 45	1,00	0,60	0,59	0,85	0,86	0,86
Mapa 46	1,00	0,63	0,65	0,82	0,77	0,98
Mapa 47	1,00	0,65	0,70	0,90	0,79	1,00
Mapa 48	1,00	0,70	0,71	0,95	0,82	0,95
Mapa 49	1,00	0,62	0,62	0,93	1,06	1,00
Mapa 50	1,00	0,58	0,62	0,83	0,77	0,83
Mapa 51	1,00	0,71	0,70	0,90	0,82	0,93
Mapa 52	1,00	0,64	0,69	1,00	1,14	1,00
Mapa 53	1,00	0,65	0,64	0,92	1,39	0,89
Mapa 54	1,00	0,60	0,64	0,81	1,00	0,89
Mapa 55	1,00	0,52	0,52	0,82	0,85	1,00
Mapa 56	1,00	0,64	0,66	0,97	1,19	0,98
Mapa 57	1,00	0,49	0,49	0,74	0,64	0,90
Mapa 58	1,00	0,62	0,59	0,87	0,95	1,00
Mapa 59	1,00	0,54	0,62	0,92	0,84	0,94
Mapa 60	1,00	0,66	0,70	0,82	1,14	0,96
Mapa 61	1,00	0,46	0,46	0,62	1,08	0,85
Mapa 62	1,00	0,48	0,49	0,69	0,55	0,93
Mapa 63	1,00	0,53	0,53	0,86	0,69	0,86
Mapa 64	1,00	0,63	0,62	0,85	0,72	0,91
Mapa 65	1,00	0,59	0,54	0,84	0,68	1,00

Mapa 66	1,00	0,51	0,49	0,77	0,86	0,77
Mapa 67	1,00	0,64	0,64	0,97	0,82	1,00
Mapa 68	1,00	0,61	0,63	0,88	0,80	0,91
Mapa 69	1,00	0,65	0,65	0,77	0,79	1,00
Mapa 70	1,00	0,64	0,65	0,94	0,72	1,00
Mapa 71	1,00	0,60	0,56	0,85	0,70	0,98
Mapa 72	1,00	0,63	0,65	0,85	0,78	0,96
Mapa 73	1,00	0,55	0,57	0,96	0,88	1,00
Mapa 74	1,00	0,58	0,59	0,89	1,05	0,92
Mapa 75	1,00	0,71	0,72	1,00	0,86	1,00
Mapa 76	1,00	0,68	0,66	0,90	1,12	0,90
Mapa 77	1,00	0,67	0,68	0,90	0,79	1,00
Mapa 78	1,00	0,63	0,65	1,00	1,50	1,00
Mapa 79	1,00	0,58	0,57	0,76	0,71	1,00
Mapa 80	1,00	0,56	0,59	0,71	1,05	0,88
Mapa 81	1,00	0,66	0,64	0,75	0,81	0,96
Mapa 82	1,00	0,59	0,57	0,75	0,83	0,98
Mapa 83	1,00	0,69	0,70	0,89	0,92	1,00
Mapa 84	1,00	0,69	0,73	0,88	0,88	0,96
Mapa 85	1,00	0,57	0,59	0,85	0,86	1,00
Mapa 86	1,00	0,61	0,62	0,71	0,88	0,94
Mapa 87	1,00	0,68	0,68	0,92	0,90	1,00
Mapa 88	1,00	0,68	0,70	0,87	0,79	0,89
Mapa 89	1,00	0,61	0,62	0,88	0,94	0,88
Mapa 90	1,00	0,54	0,62	0,58	0,72	0,63
Mapa 91	1,00	0,48	0,51	0,61	1,07	1,00
Mapa 92	1,00	0,63	0,62	0,90	0,78	1,00
Mapa 93	1,00	0,55	0,56	0,89	0,77	0,96
Mapa 94	1,00	0,61	0,61	0,79	0,70	1,00
Mapa 95	1,00	0,58	0,58	0,98	1,13	1,04
Mapa 96	1,00	0,38	0,41	0,68	0,64	0,85
Mapa 97	1,00	0,64	0,60	0,74	0,85	1,00
Mapa 98	1,00	0,70	0,72	0,89	0,80	0,89
Mapa 99	1,00	0,43	0,46	0,70	0,76	0,92
TOTAL	100,00	59,35	60,40	83,42	87,49	93,52

Tabela 7 – Tabela de pontuação para cada abordagem testada.