Colouring

Bruno Ribeiro - Danilo Carvalho - Igor Penha - Lucas Gobbi

01

Contexto do Domínio

Contexto

- O domínio foi desenvolvido para a IPC 2023 de tracks HTN;
- Desenvolvido por Gregor Behnke, um dos organizadores da IPC;
- É um domínio *partial-order*, ou seja, as ações em um plano não são especificadas em uma ordem rígida. Em vez disso, as ações são especificadas com relações de precedência parciais, permitindo que elas ocorram em diferentes ordens, desde que as restrições de precedência sejam respeitadas. Isso reflete uma abordagem mais flexível ao planejamento, onde as ações podem ser reordenadas desde que os pré-requisitos e os efeitos sejam respeitados. Isso pode permitir uma maior flexibilidade na execução do plano;

Contexto

- Colouring não se trata apenas de um jogo de colorir, o domínio codifica uma versão do *problema de azulejos* [1], que é frequentemente usado para reduções de complexidade;

- Dado um conjunto de azulejos disponíveis, cada um com uma cor em uma de suas bordas, a tarefa é preencher um quadrado de (n x n) azulejos, de forma que as bordas tocantes tenham a mesma cor. A borda externa não tem uma cor obrigatória;

- Contato com o Gregor Behnke para a explicação do domínio.

Gregor Behnke
para mim

seg., 23 de out., 11:26 (há 23 horas)

inglês > português Traduzir mensagem

Desativar para: inglês

Dear Lucas,

There is little to no documentation on the domain -- so far. I.e. all there is has to be pieced together from the actual domain. I have recently written a very short explanation for a forthcoming journal article that roughly reads as follows:

\textbf{Colouring} (PO) encodes a version of the tiling problem [1], which is frequently used for complexity reductions.
Given a set of available tiles, each having a colour at one of its edges, the task is to fill an $n\times n$ square with these tiles, s.t., touching edges have the same colour.
The outer edge has no required colour.
This problem is $\mathbb{NP}$-complete for unary encoded n.
The encoding is based on the idea of proof encoding double-exponentially time-bounded Turing Machine [2] (specifically Thm. 6.1).

The two citations are
[1] http://people.irisa.fr/Francois.Schwarzentruber/sif2_thx/articles/tiling.pdf
[2] https://cdn.aaai.org/ojs/13721/13721-40-17239-1-2-20201228.pdf

This explains the purpose of the domain (your first question) and is all the documentation there is about it. As far as I know, there is no corresponding PDDL version of the domain. It should be possible to create one however -- but it might rely heavily on counters.

- Contato com o Gregor Behnke para a explicação do domínio.

Overall, the colouring domains is quite complex as it uses interlocking strings of actions (those started by the start_lines and the row) to correctly simulate an nxn filed of tiles. This inter-locking mechanism is based on all the count_*/set_*/delete_* actions in the domain. Its basic idea is explained in the proof for Thm. 6.1 of [2].

I hope this at least helps a little bit to understand the domain. Unfortunately, I don't really have the time to create a more in-depth documentation for the domain (which would probably take hours) - so if you have anything written at the end, I would be happy to take a look and if it checks out add it to the repo.

Cheers,
Gregor

Contexto

- Gregor Behnke, explicou que para criação do domínio ele se embasou em dois artigos.
  - O primeiro artigo explica a relação entre a Máquina de Turing com o problema do Tiling [2], fazendo a correlação entre o problema matemático de combinações com os limites do que pode ser computado.
  - O segundo artigo explica a relação entre a Máquina de Turing e “Hierarchical Task Network” [3] no caso as vantagens de usar um planejador hierarquico para resolver problemas polinomiais e recursivos.

Contexto

- Sobre o problema “Tiling”, é um problema de alta complexidade de execução mas baixa complexidade para o reconhecimento da solução.

- O problema consiste em um Azulejo ou ladrilho divido em 4 triângulos dentro de um quadrado, podendo ter 2, 3 ou 4 cores dentro do quadrado, sendo que as cores iguais não podem ser adjacentes no mesmo quadrado.

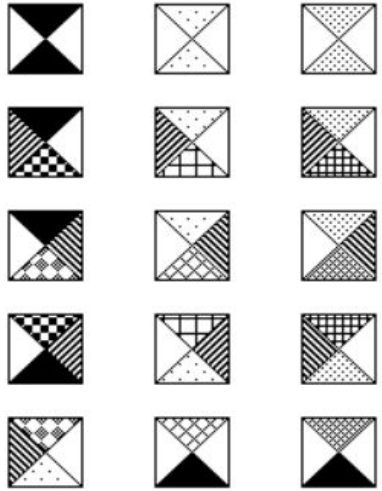


Figure 2: A sample of tile types

Contexto

- Objetivo do problema é conectar as cores das laterais dos azulejos de forma que elas sejam iguais.

- Para a realização da organização algumas regras devem ser seguidas, como :
  - Os azulejos não podem ser rotacionados, com exceção dos azulejos de 2 cores que podem ser rotacionados em 180 graus;
  - Os azulejos não podem ser refletidos;
  - Os azulejos podem ser movidos em qualquer sentido dentro do quadrado nxn.

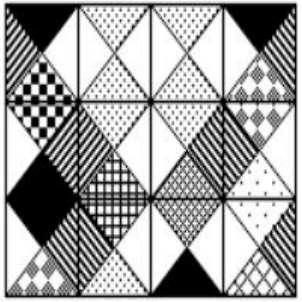


Figure 3: A region legally tiled using the tile types from Figure 2

Contexto

- Esse problema é NP-completo para *n* codificado de forma unária;
- A codificação se baseia na idéia de codificar uma Máquina de Turing limitada por tempo de forma duplamente exponencial [2] (especialmente o teorema 6.1);

O que é a Máquina de Turing?

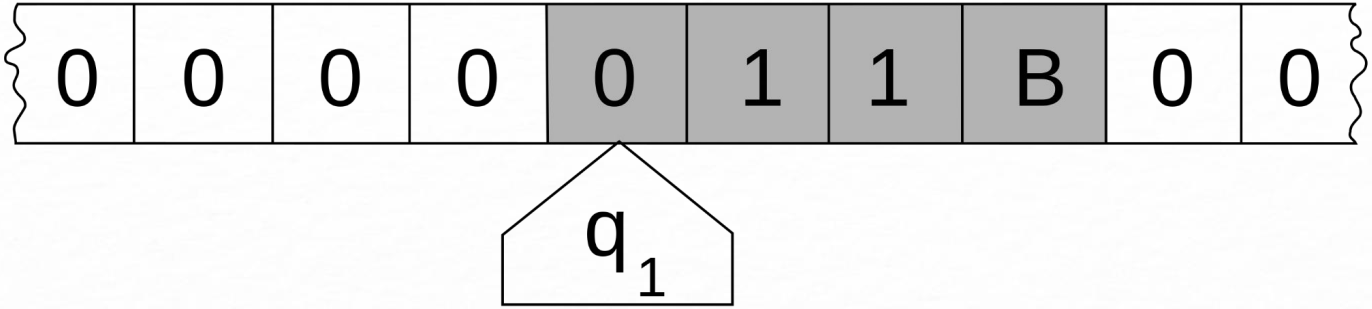
Uma máquina de Turing é um modelo abstrato de um dispositivo de processamento de informações desenvolvido pelo matemático britânico Alan Turing em 1936. Ela foi projetada para representar um sistema de computação idealizado que pode resolver uma ampla variedade de problemas matemáticos.

O que é a Máquina de Turing?

Uma máquina de Turing consiste em três principais componentes:

- Uma fita infinita dividida em células, onde cada célula pode conter um símbolo (geralmente 0 ou 1), e a fita pode ser movida para a esquerda ou direita.
- Uma cabeça de leitura/escrita que lê o símbolo atual na célula sob ela e pode escrever um novo símbolo nessa célula.
- Um conjunto finito de estados e regras que especificam como a máquina deve se comportar com base no estado atual, no símbolo sob a cabeça e em uma tabela de transições.

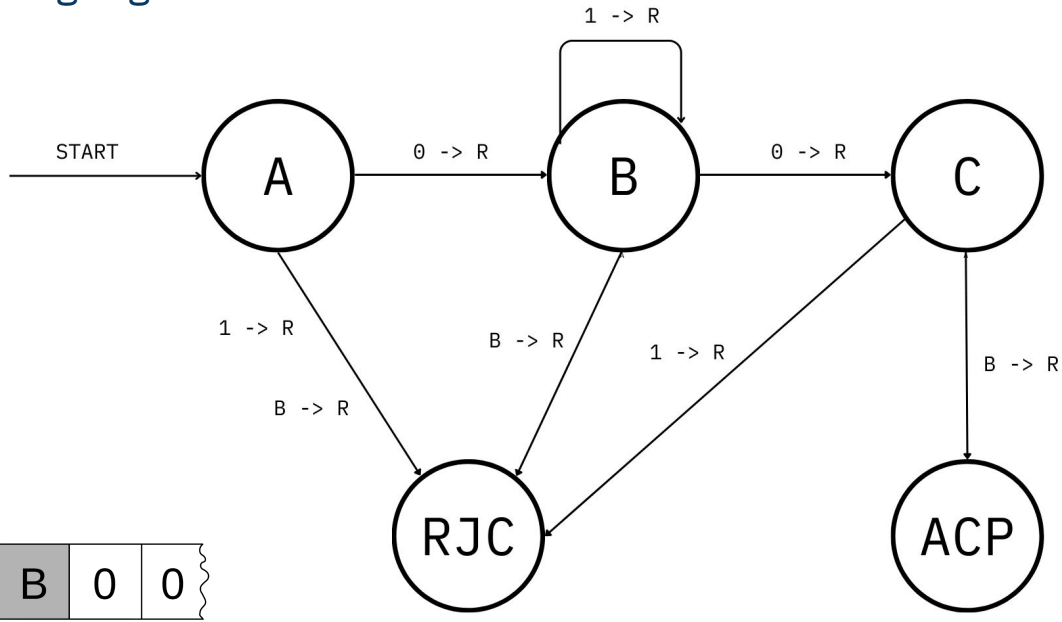
O que é a Máquina de Turing?

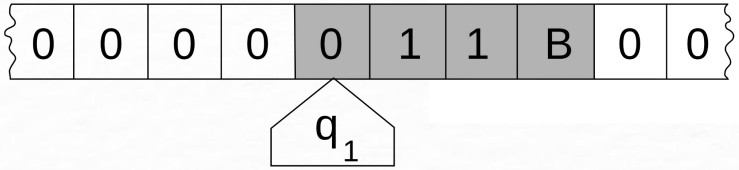


A máquina de Turing opera seguindo um algoritmo, movendo a fita e mudando de estado de acordo com as regras de transição.

O que é a Máquina de Turing?

MT para reconhecer a linguagem: L = 01*0



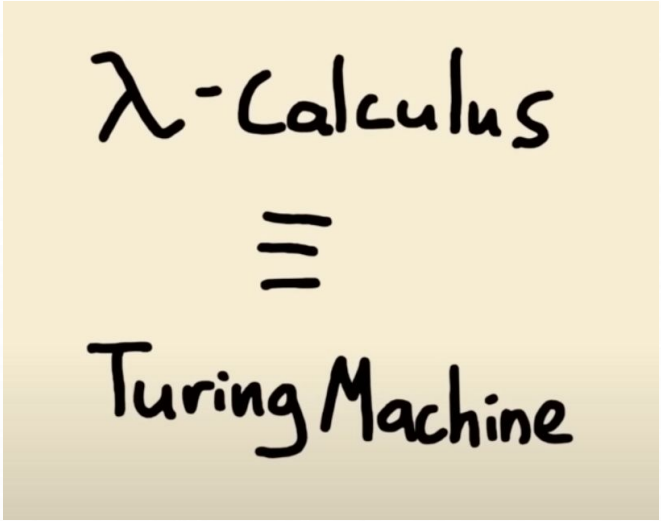


Máquina de Turing e o Cáculo λ

- Em 1930, Alonzo Church, um matemático americano, publicou um artigo intitulado "Uma Fórmula Definível de Função Numérica de Números Inteiros", onde introduziu o Cálculo Lambda. O Cálculo Lambda é uma notação matemática que descreve funções matemáticas computáveis.
- Church estava interessado em resolver o problema da decisão, ou seja, criar uma definição precisa de quais problemas matemáticos podem ser resolvidos algorítmicamente. Ele queria determinar quais funções eram computáveis por meio do Cálculo Lambda.
- Church demonstrou que as funções recursivas (um conjunto específico de funções matemáticas) podem ser representadas usando o Cálculo Lambda. Ele argumentou que essas funções eram a classe de funções computáveis.

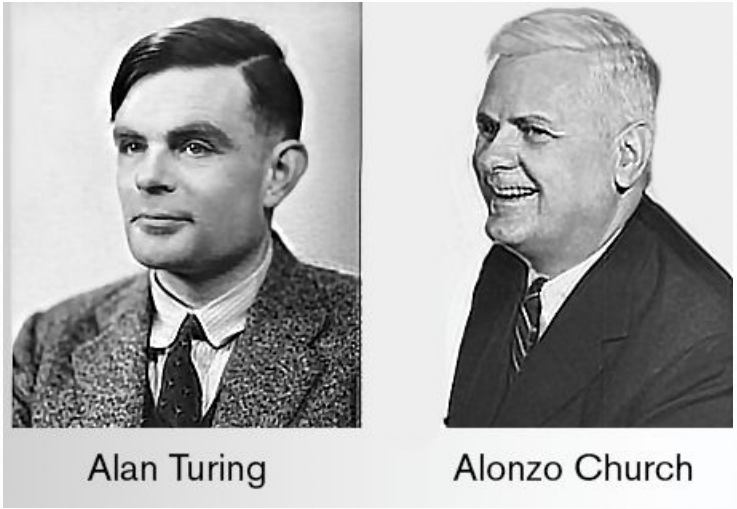
Máquina de Turing e o Cáculo λ

- Embora Church e Turing tenham desenvolvido modelos diferentes (Cálculo Lambda e Máquina de Turing), eles chegaram a conclusões equivalentes em termos de poder computacional. Ambos demonstraram que seus modelos podiam representar todas as funções computáveis.



Tese Church-Turing

"Qualquer coisa que possa ser calculada pode, em princípio, ser feita por uma máquina de Turing. Se uma máquina de Turing não puder calculá-la, então não é computável."



02

O Domínio

O Domínio

```
(:types number colour - object)

(:task deleting :parameters (?x - number))
(:task checking :parameters (?x - number))
(:task stepping :parameters (?x - number))

(:task start_line :parameters (?x - number))
(:task line :parameters (?c - colour))
(:task rows :parameters (?x - number))
(:task cells :parameters (?x - number))
(:task tile_choice :parameters (?c ?cc - colour))
(:task satisfy :parameters ())
```

O Domínio

```
(:predicates
  (next ?x ?y - number)
  (hasCounter ?x - number)
  (min ?x - number)
  (max ?x - number)
  (cur_colour_tape ?c - colour)
  (cur_colour_next ?c - colour)
  (isIGN ?c - colour)
  (tape)
  (tape2)
  (tile ?c1 ?c2 ?c3 ?c4 - colour ?n - number)
  (colok ?c1 ?c2 - colour)
)
```

O Domínio - Rows

```
(:method row_start
  :parameters (?c ?cc ?ccc - number ?curcol ?IGN - colour)
  :task (rows ?c)
  :precondition (and (min ?ccc) (next ?c ?cc) (isIGN ?IGN))
  :ordered-subtasks (and
    (reset_col_next ?IGN ?curcol)
    (cells ?ccc)
    (rows ?cc))
)

(:method no_row
  :parameters (?c - number)
  :task (rows ?c)
  :precondition (and (max ?c))
  :ordered-subtasks (and (satisfy))
)
```

O Domínio - Lines

```
(:method line_start
  :parameters (?c - number ?curcol ?IGN
- colour)
  :task (start_line ?c)
  :precondition (and (isIGN ?IGN))
  :ordered-subtasks (and
    (stepping ?c)
    (set_colour ?IGN)
    (get_colour_tape ?curcol)
    (line ?curcol))
)
```

```
(:method next_line
  :parameters (?c - number ?oldcol
?curcol - colour)
  :task (line ?oldcol)
  :precondition (and (min ?c))
  :ordered-subtasks (and
    (stepping ?c)
    (set_colour ?oldcol)
    (get_colour_tape ?curcol)
    (line ?curcol))
)

(:method no_line
  :parameters (?c - colour)
  :task (line ?c)
  :ordered-subtasks (and)
)
```

O Domínio - Satisfy

```
(:method satisfy_something
  :parameters (?col2 - colour)
  :task (satisfy)
  :ordered-subtasks (and
    (set_tape2)
    (set_colour2 ?col2)
    (delete_tape2)
    (satisfy))
)

(:method no_satisfy
  :parameters ()
  :task (satisfy)
  :ordered-subtasks (and)
)
```

O Domínio - Cells

```
(:method cell_content_create
  :parameters (?c ?n ?nn - number ?col1 ?col2
?colDown - colour)
  :task (cells ?n)
  :precondition (and (min ?c) (next ?n ?nn) (colok
?colDown ?col1))
  :ordered-subtasks (and
    (deleting ?c)
    (set_tape)
    (delete_tape)
    (checking ?c)
    (get_colour ?colDown)
    (tile_choice ?col1 ?col2)
    (set_tape2)
    (set_colour2 ?col2)
    (delete_tape2)
    (cells ?nn))
)
```

```
(:method choose_tile_to_place
  :parameters (?cc ?col1 ?col2 ?col3
?col4 - colour ?n ?nn - number)
  :task (tile_choice ?col3 ?col4)
  :ordered-subtasks (and
     (do_tile ?cc ?col1 ?col2 ?col3
?col4 ?n ?nn))
)
```

```
(:method cell_content_end
  :parameters (?c - number)
  :task (cells ?c)
  :precondition (and (max ?c))
  :ordered-subtasks (and)
)
```

O Domínio - Round

```
(:method round_steps
  :parameters (?c ?cc - number)
  :task (stepping ?c)
  :ordered-subtasks (and
    (count_steps ?c ?cc)
    (stepping ?cc))
)

(:method round_steps_end
  :parameters (?c - number)
  :task (stepping ?c)
  :precondition (and (max ?c))
  :ordered-subtasks (and)
)
```

```
(:method round_delete
  :parameters (?c ?cc - number)
  :task (deleting ?c)
  :ordered-subtasks (and
    (count_delete ?c ?cc)
    (deleting ?cc))
)

(:method round_delete_end
  :parameters (?c - number)
  :task (deleting ?c)
  :precondition (and (max ?c))
  :ordered-subtasks (and)
)
```

O Domínio - Round

```
(:method round_check
  :parameters (?c ?cc - number)
  :task (checking ?c)
  :ordered-subtasks (and
    (count_check ?c ?cc)
    (checking ?cc))
)

(:method round_check_end
  :parameters (?c - number)
  :task (checking ?c)
  :precondition (and (max ?c))
  :ordered-subtasks (and)
)
```

O Domínio - Actions

```
(:action count_steps
  :parameters (?c ?cc - number)
  :precondition (and (next ?c ?cc) (tape))
  :effect (and (hasCounter ?c))
)

(:action count_delete
  :parameters (?c ?cc - number)
  :precondition (and (next ?c ?cc))
  :effect (and (not (hasCounter ?c)))
)

(:action count_check
  :parameters (?c ?cc - number)
  :precondition (and (next ?c ?cc) (hasCounter ?c))
  :effect (and)
)
```

O Domínio - Actions

```
(:action set_colour
  :parameters (?c - colour)
  :precondition (and (tape))
  :effect (and (cur_colour_tape ?c))
)

(:action set_colour2
  :parameters (?c - colour)
  :precondition (and (tape2))
  :effect (and (cur_colour_tape ?c))
)

(:action get_colour
  :parameters (?c - colour)
  :precondition (and (cur_colour_tape ?c))
  :effect (and (not (cur_colour_tape ?c)))
)
```

```
(:action get_colour_tape
  :parameters (?c - colour)
  :precondition (and (tape2)
(cur_colour_tape ?c))
  :effect (and (not (cur_colour_tape ?c)))
)
```

O Domínio - Actions

```
(:action set_tape
  :parameters ()
  :precondition (and)
  :effect (and (tape))
)

(:action delete_tape
  :parameters ()
  :precondition (and)
  :effect (and(not (tape)))
)
```

```
(:action set_tape2
  :parameters ()
  :precondition (and)
  :effect (and (tape2))
)

(:action delete_tape2
  :parameters ()
  :precondition (and)
  :effect (and (not
(tape2)))
)
```

O Domínio - Actions

```
(:action do_tile
  :parameters (?col1cur ?col1 ?col2 ?col3 ?col4 -
colour ?n ?nn - number)
  :precondition (and
    (tile ?col1 ?col2 ?col3 ?col4 ?n)
    (next ?nn ?n)
    (not (min ?n))
    (cur_colour_next ?col1cur)
    (colok ?col1cur ?col1))
  :effect (and
    (not (tile ?col1 ?col2 ?col3 ?col4 ?n))
    (tile ?col1 ?col2 ?col3 ?col4 ?nn)
    (not (cur_colour_next ?col1cur))
    (cur_colour_next ?col2))
)
```

```
(:action reset_col_next
  :parameters (?col1 ?col2 - colour)
  :precondition (and (cur_colour_next ?col2))
  :effect (and
    (cur_colour_next ?col1)
    (not (cur_colour_next ?col2)))
)
```

03

Comparação ao PDDL

Sobre PDDL

- Inicialmente, não encontramos uma versão em PDDL do problema.
- No entanto, durante nossa comunicação com Gregor Behnke, ele destacou a viabilidade de criar uma versão em PDDL, embora isso possa depender significativamente das contagens envolvidas.

Razões para escolher o HDDL

- A escolha pelo HDDL foi devido à capacidade e possibilidade de lidar com problemas de natureza exponencial e polinomial.
- A escolha também se baseia na flexibilidade da recursividade em que para executar uma ação primeiro já deveria ter executado outra.

04

Ferramentas de Apoio

Generators

- O domínio é acompanhado de dois scripts para geração de problemas:
  - C++ → script para escrever os arquivos de problema.
  - $ → script para nomeação e criação de diretórios e arquivos.

Script C++

```
int main(int argc, char *argv[]) {
	int n = atoi(argv[1]); // size of the field
	int s = atoi(argv[2]); // seed
	int p = atoi(argv[3]); // problem number
	int c = atoi(argv[4]); // colours

	srand(s);

	arr[0][0][LEFT] = rand()%c;
	FOR(j,0,n)
		arr[0][j][DOWN] = rand() % c;

	FOR(i,0,n){
		FOR(j,0,n){
			// select upper and right
			arr[i][j][UP] = rand() % c;
			arr[i][j][RIGHT] = rand() % c;
			arr[i+1][j][LEFT] = arr[i][j][RIGHT];
			arr[i][j+1][DOWN] = arr[i][j][UP];
		}
	}
```

```
map<vector<int>, int> tiles;

	FOR(i,0,n) {
		FOR(j,0,n) {
			//cout << arr[i][j][UP] << " " <<
arr[i][j][DOWN] << " " << arr[i][j][LEFT] << " " <<
arr[i][j][RIGHT] << " " << endl;

			vi v;
			v.push_back(arr[i][j][UP]);
			v.push_back(arr[i][j][DOWN]);
			v.push_back(arr[i][j][LEFT]);
			v.push_back(arr[i][j][RIGHT]);

			tiles[v]++;
		}
	}
	int maxN = n;

	for (auto [v,i] : tiles)
		if (i > maxN) maxN = i;
```

Script C++

```
        cout << "(define (problem tiling-" << p << ")" <<
endl;
        cout << "  (:domain game)" << endl;
        cout << "  (:objects" << endl;
        cout << "    ";
        FOR(i,0,maxN+1) cout << " n" << i;
        cout << " - number" << endl;
        FOR(i,0,c) cout << " c" << i;
        cout << " IGN - colour" << endl;
        cout << "  )" << endl;
        cout << "  (:init" << endl;
        FOR(i,0,maxN)
                cout << "    (next n" << i << " n" << i+1 <<
")" << endl;
        cout << "    (max n" << n << ")" << endl;
        cout << "    (min n0)" << endl;
        cout << "    (cur_colour_next IGN)" << endl;
        cout << "    (isIGN IGN)" << endl;
        FOR(i,0,c) cout << "    (colok IGN c" << i << ")" <<
endl;
        FOR(i,0,c) cout << "    (colok c" << i << " c" << i <<
")" << endl;
        for (auto [v,i] : tiles) {
                cout << "    (tile c" << v[0] << " c" << v[1]
<< " c" << v[2] << " c" << v[3] << " n" << i << ")" << endl;
        }
        cout << "  )" << endl;

        cout << "  (:htn" << endl;
        cout << "    :tasks (and" << endl;
        cout << "      (rows n0)" << endl;
        FOR(i,0,n)
                cout << "      (start_line n" << i << ")" <<
endl;
        cout << "    )" << endl;
        cout << "  )" << endl;

        cout << ")" << endl;
}
```

Script $

```
#!/bin/bash

g++ -O2 generator.cpp
mkdir problems

i=3
./a.out 2 $((420 + $i)) $i 2 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 2 $((420 + $i)) $i 10 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))

./a.out 3 $((420 + $i)) $i 1 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 3 $((420 + $i)) $i 10 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 3 $((420 + $i)) $i 10 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 3 $((420 + $i)) $i 10 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 3 $((420 + $i)) $i 7 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 3 $((420 + $i)) $i 7 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 3 $((420 + $i)) $i 7 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 3 $((420 + $i)) $i 5 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 3 $((420 + $i)) $i 5 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 3 $((420 + $i)) $i 5 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 3 $((420 + $i)) $i 5 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 3 $((420 + $i)) $i 4 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
```

Script $

```
./a.out 3 $((420 + $i)) $i 3 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 3 $((420 + $i)) $i 3 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 3 $((420 + $i)) $i 2 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 3 $((420 + $i)) $i 2 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
```

```
./a.out 4 $((420 + $i)) $i 50 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 4 $((420 + $i)) $i 50 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 4 $((420 + $i)) $i 10 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 4 $((420 + $i)) $i 10 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 4 $((420 + $i)) $i 5 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 4 $((420 + $i)) $i 5 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 4 $((420 + $i)) $i 3 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 4 $((420 + $i)) $i 3 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 4 $((420 + $i)) $i 2 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
./a.out 4 $((420 + $i)) $i 2 > problems/pfile$(printf "%02d" $i).hddl ; i=$((i+1))
```

05

Tabela de Execução

Tabela de execução

Maquina: gpu1 - Planner: pandaPI

Problem	Generated search nodes	Total costs of actions	Search time (s)
pfile01	511	71	0.004
pfile02	5189	71	0.04
pfile03	734	71	0.006
pfile06	13,210,684	181	73.06
pfile09	7,237,342	181	35.12
pfile12	12,018,250	178	61.09

Tabela de execução

Maquina: gpu1 - Planner: pandaPI

Problem	Generated search nodes	Total costs of actions	Search time (s)
pfile15	1,956,761	181	9.83
pfile18	216,140	175	0.845
pfile20	3,077	181	0.03
pfile09	7,237,342	181	35.12
pfile25	-	-	TLE
pfile30	-	-	TLE

Bibliografia

[1] Tiling Problems

[2] The convenience of tilings Peter van Emde Boas * ILLC, Faculteit WINS, Universiteit van Amsterdam, Plantage Muidergracht 24, 101

[3] Tight Bounds for HTN Planning

Thank you!

Colouring