IIA - SOKOBAN

Francisco Martinho (85088)

Mariana Pinto (84792)

MIECT – UA - Dezembro de 2020

ARQUITETURA

- A arquitetura do código está organizado pela:

1. Inicialização das variáveis para receber os valores iniciais do Keeper e das Caixas;
2. Procura do caminho possível da(s) caixa(s) até ao(s) goal(s);
3. Movimento do Keeper de acordo com a solução encontrada;
4. Colocação da(s) caixa(s) no(s) goal(s).

ALGORITMO USADO

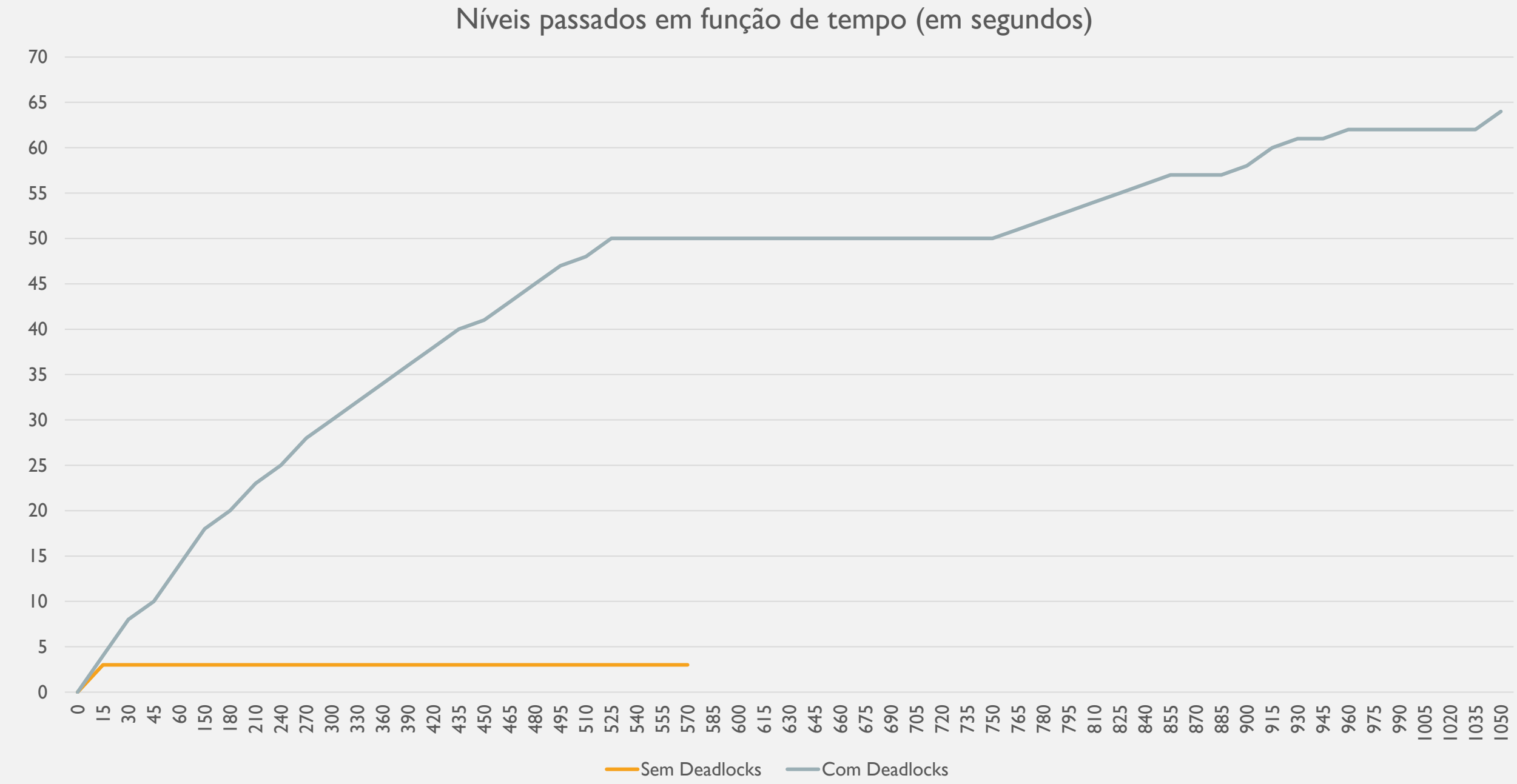
- O algoritmo usado para a pesquisa do caminho é o "***Minimum Matching Lower Bound***". Este calcula um custo mínimo de um gráfico bipartido, em que o custo é representado pela distância necessária para empurrar uma caixa até um goal. A cada caixa é designada um goal específico de forma a diminuir a distância total e a detetar deadlocks quando nem todas as caixas conseguem chegar ao goal designado.
- Usando uma abordagem "**Greedy**", as distância entre cada caixa e cada goal são guardadas numa lista por ordem crescente de distância. Existe outra lista que guarda as caixas comque já tenham correspondência, de forma a que duas caixas não fiquem com o mesmo goal. Com este algoritmo, é possível chegar até ao nível 64.

FUNÇÕES

- Ficheiro **state.py**: contém os estados possíveis para o Keeper e as caixas
- Ficheiro **tree_search.py**: módulo tree_search reutilizado das aulas práticas
- Ficheiro **sokosolver.py**: módulo que encontra a solução para cada nível
  - pair(self): associa a cada caixa um goal específico;
  - actions(Self,state): calcula as próximas ações possíveis a realizar;
  - result(self,state,next_move): calcula o resultado da ação num estado, ou seja, o estado seguinte;
  - heuristic(self, state, goal): calcula o custo estimado de chegar de um estado a outro, usado *Minimum matching lowe bound* com abordagem Greedy;
  - satisfies(self,state,goal): testa se um goal está no estado "state"

FUNÇÕES

- Ficheiro **translation.py**:
  - translate(solution): traduz os movimentos do Keeper de uma lista com os movimentos do teclado.
- Ficheiro **deadlock.py**: encontra os deadlocks possíveis para cada nível
  - deadlock_corner(box,obstacles): deadlock de um canto;
  - Deadlock_box(box,allboxes,obstacles): deadlock de duas caixas juntas, sem estarem no goal;
  - isWall(position,x,y,corner,obstacles): função auxiliar para verificar se dada posição é parede;
  - Deadlock_notgoal(box,goals,obstacles): deadlock para verificar se uma caixa está junto a uma parede e não consegue chegar ao goal


Níveis passados em função de tempo (em segundos)
70
65
60
55
50
45
40
35
30
25
20
15
10
5
0
0
15
30
45
60
150
180
210
240
270
300
330
360
390
420
435
450
465
480
495
510
525
540
555
570
585
600
615
630
645
660
675
690
705
720
735
750
765
780
795
810
825
840
855
870
885
900
915
930
945
960
975
990
1005
1020
1035
1050
Sem Deadlocks
Com Deadlocks