CMC-15 - Projeto Buscas

Gabriel Martinz, Marina Moreira

1 Objetivo do Trabalho e Descrição da Implementação

1.1 Objetivo

O objetivo do trabalho foi aplicar os conhecimentos de algoritmos de busca, melhoria iterativa e resolução de problemas por satisfação de restrições obtidos na disciplina.

1.2 Descrição da implementação

A implementação dos problemas foi feita em um *notebook* Jupyter utilizando um kernel Python com um arquivo de *script* Python auxiliar contendo algumas constantes. Foi utilizada a IDE Visual Studio Code.

2 Resultados Obtidos

2.1 Busca em tabuleiro

Foi desenvolvido o algoritmo A^* para percorrer o tabuleiro. O tabuleiro foi criado como uma matriz do NumPy com todos os quadrados brancos com peso 1 e todos os pretos com peso -1. Com isso, pode-se bloquear o movimento de um quadrado branco para um quadrado preto.

Foi implementada uma classe para representar o nó no algoritmo A^*, contendo os valores de $f(n)$ e $g(n)$ em atributos, como também um atributo booleana dizendo se o nó já foi totalmente analisado e um atributo que mostra o seu nó pai no algoritmo.

A heurística $h'(n)$ utilizada no algoritmo foi a de distância euclideana, utilizando os índices no tabuleiro. Foi criada uma matriz de mesmo *shape* do tabuleiro contendo nós que se relacionam com as posições do tabuleiro. O algoritmo percorreu esta matriz de nós e retornou uma lista com as posições do melhor caminho encontrado. O caminho foi plotado utilizando o Matplotlib e está mostrado na figura 1

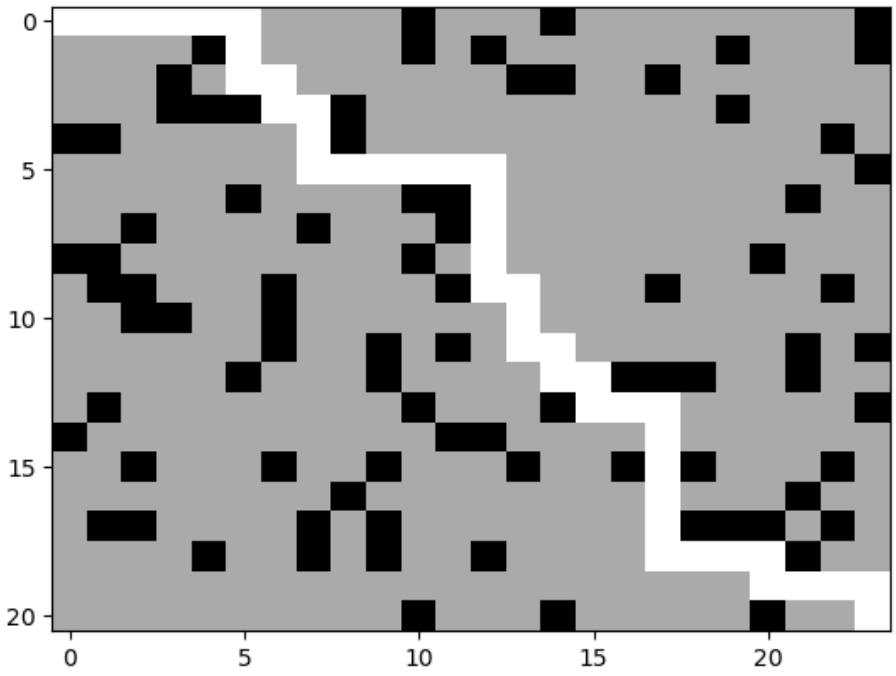

Figura 1: Caminho ótimo encontrado pelo algoritmo A^*.

2.2 Light-Up

2.2.1 Modelagem

A modelagem do problema foi feita considerando a técnica da satisfação de restrição (*constraint programming*).

Foram estabelecidos 3 restrições:

- Restrição 1: estabelece restrição para as lâmpadas que ficarão diretamente ao lado de blocos pretos que especificam números específicos. Nesse caso, a restrição é a de que o número de lâmpadas seja exatamente o número descrito pelo bloco preto.

- Restrição 2: estabelece restrição aos blocos brancos que estejam diretamente limitados por blocos pretos. Nessa restrição, tem-se uma questão de máximo. Nas semi-linhas ou semi-colunas (isto é, trechos de linhas e de colunas compostos apenas por blocos brancos e limitados em ambas as extremidades por blocos pretos), pode haver no máximo uma lâmpada.

- Restrição 3: estabelece restrição aos demais blocos brancos do tabuleiro. Nesse caso, pode haver no mínimo uma lâmpada.

2.2.2 Implementação

A implementação necessitou do módulo `constraint` para a abordagem de satisfação de restrições.

- Restrição 1: Como se trata de uma restrição que dita um número exato de itens, foi utilizada a função `ExactSumConstraint` do módulo `constraint`.
- Restrição 2: Como se trata de uma restrição que dita um máximo de itens, foi utilizada a função `MaxSumConstraint` do módulo `constraint`.
- Restrição 3: Como se trata de uma restrição que dita um mínimo de itens, foi utilizada a função `MinSumConstraint` do módulo `constraint`.

2.2.3 Testes

A apresentação de 3 tabuleiros de Light Up foi deixada indicada no notebook, bem como os resultados encontrados. Caso desejado, é possível rodá-las novamente, bem como inserir diferentes tabuleiros. Os três tabuleiros de teste oferecidos no código seguem visualmente representadas nas figuras 2, 3 e 4.

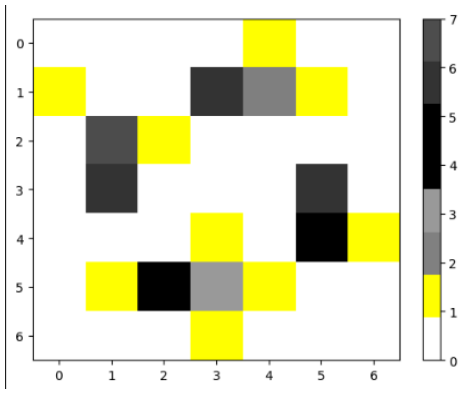

Figura 2: Resultado encontrado no tabuleiro 1

2.3 Melhoria Iterativa

Para maximizar a função $f(x,y) = 4e^{-(x-1)^2-(y-1)^2} + e^{-(x-3)^2-(y-3)^2} + e^{-(x-3)^2-(y+3)^2} + e^{-(x+3)^2-(y-3)^2} + e^{-(x+3)^2-(y+3)^2}$ foi utilizado o algoritmo de *simulated annealing* para minimização com a função de custo $g(x,y) = -f(x,y)$. Foi utilizado um *schedule* de temperatura do tipo $T(i) = \frac{T_0}{1+\beta \cdot i^2}$, onde i é a iteração e T_0 e β são hiperparâmetros.

Para o *simulated annealing*, a seleção de ponto vizinho foi aleatória, selecionando um ponto aleatório em um círculo (cujo raio também é um hiperparâmetro) ao redor do valor (x,y) atual dentro do algoritmo.

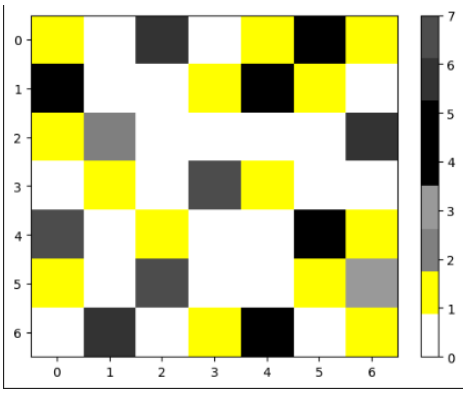

Figura 3: Resultado encontrado no tabuleiro 2

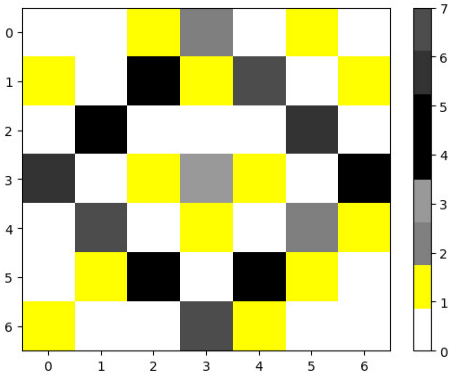

Figura 4: Resultado encontrado no tabuleiro 3

Após rodar o algoritmo, o ponto de máximo retornado por ele está sempre próximo do ponto $(1, 1)$, que é o máximo analítico da função. Podemos ver no gráfico da função mostrado na figura 5.

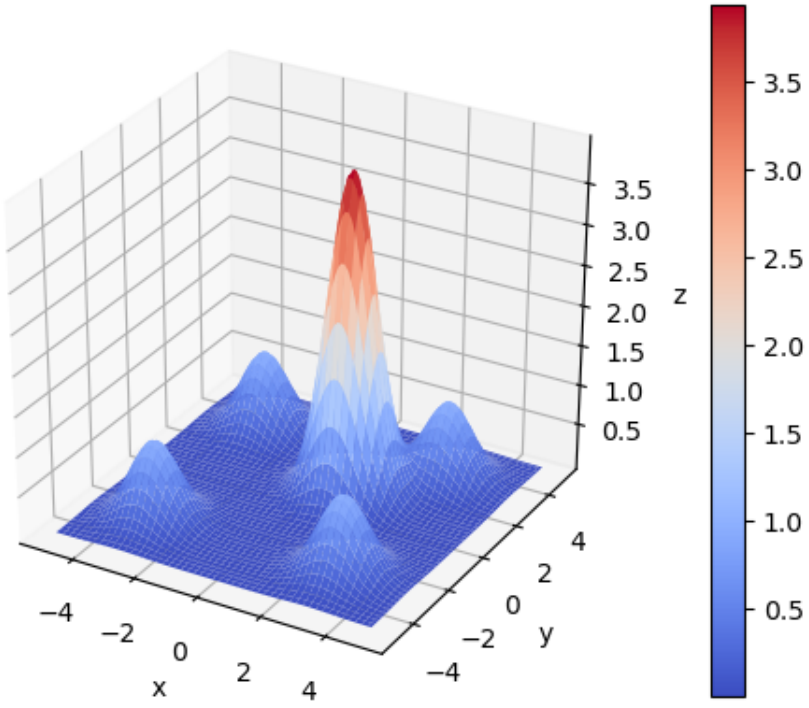

Figura 5: Gráfico 3D da função a ser maximizada na questão 3.