

Introdução à Inteligência Artificial

Trabalho Prático 2 – Problema de Otimização

Engenharia Informática

2022/2023

Gonçalo Leal Gonçalves Geraldes, a2021133456, P7

Miguel Ângelo Antunes Brandão, a2021133762, P6

# Índice

[Índice 1](#_Toc124199460)

[Índice de Imagens 2](#_Toc124199461)

[Introdução 3](#_Toc124199462)

[Problema de Subgrafo de Borda Máximo (*Maximum Edge Subgraph Problem*) 4](#_Toc124199463)

[Instâncias para teste 4](#_Toc124199464)

[Objetivo da otimização 5](#_Toc124199465)

[Algoritmo de Pesquisa Local (Trepa-Colinas) 6](#_Toc124199466)

[Aceitar soluções de custo igual 6](#_Toc124199467)

[Principais opções tomadas 6](#_Toc124199468)

[Resultados Obtidos 7](#_Toc124199469)

[Algoritmo Evolutivo 8](#_Toc124199470)

[Principais opções tomadas 8](#_Toc124199471)

[Resultados Obtidos 8](#_Toc124199472)

[Algoritmo Híbrido (combinação dos anteriormente abordados) 11](#_Toc124199473)

[Principais opções tomadas 11](#_Toc124199474)

[Resultados obtidos 11](#_Toc124199475)

[Conclusão 12](#_Toc124199476)

# Índice de Imagens

[Figura 1 - Resultados do trepa colinas 7](#_Toc124199520)

[Figura 2 - resultados teste.txt evolutivo 8](#_Toc124199521)

[Figura 3 - resultados file1.txt evolutivo 9](#_Toc124199522)

[Figura 4 - resultados file2.txt evolutivo 9](#_Toc124199523)

[Figura 5 - resultados file3.txt evolutivo 10](#_Toc124199524)

[Figura 6 - resultados hibrido 11](#_Toc124199525)

# Introdução

Este trabalho consiste em conceber, implementar e testar métodos de otimização que encontrem soluções de boa qualidade para diferentes instâncias do problema “*Maximum Edge Subgraph Problem*”.

# Problema de Subgrafo de Borda Máximo (*Maximum Edge Subgraph Problem*)

Dado um grafo e um valor inteiro *k*, o *Maximum Edge Subgraph Problem* consiste em

encontrar um subconjunto de k-vértices tal que o número de arestas dentro do

subconjunto seja máximo.

**Formalmente o problema é definido:**

**Dados:**

* um grafo não direcionado **G = (V, A),** composto por um conjunto **V** de vértices ligados entre si por arestas **A**
* um inteiro **k**

**Problema:**

* encontrar um subconjunto de vértices **S**, de tamanho **k**, tal que  
  **S ⊆ V**, de forma a maximizar o número de arestas desse subconjunto
* o objetivo deste problema é, portanto, de maximização.

## Instâncias para teste

Foram utilizados os ficheiros de teste fornecidos no Moodle, cada um complexidades diferentes para efetuar o estudo experimental dos vários algoritmos. Os ficheiros estão organizados da seguinte maneira:

A primeira linha contém o valor de *k*.

Depois disso surge uma linha iniciada por “*p edge*”, onde são indicados o número de vértices e o número de arestas.

Os vértices são identificados por números inteiros a partir de 1. Cada uma das linhas seguintes inicia-se com a letra e que especifica uma aresta entre dois vértices.

## Objetivo da otimização

Neste trabalho implementou-se e avaliou-se a capacidade de diferentes algoritmos de otimização para encontrar soluções de boa qualidade para o problema “*Maximum Edge Subgraph Problem”.*

Foram implementados 3 métodos e efetuado um estudo comparativo aprofundado sobre o desempenho da otimização.

# Algoritmo de Pesquisa Local (Trepa-Colinas)

Os algoritmos de Pesquisa Local são uma classe de algoritmos que visam encontrar uma solução válida para um problema dado, explorando a vizinhança da solução atual em busca de uma solução ainda melhor.

Neste trabalho, foi utilizado o algoritmo “**Trepa-Colinas *first choice***”, que parte de um estado inicial aleatório (entre 1 e o número de vértices) e avalia todos os seus vizinhos para encontrar a melhor solução. Se este melhor vizinho é melhor do que o estado atual, ele é aceito e uma nova iteração é iniciada a partir dele. Este processo é repetido até que todos os vizinhos tenham uma qualidade inferior ao estado atual.

## Aceitar soluções de custo igual

Em alguns casos, pode ser útil considerar soluções de custo igual como válidas quando se está a utilizar um algoritmo de pesquisa local. Isto pode ser útil em situações em que o objetivo é encontrar uma solução viável o mais rapidamente possível, mesmo que esta não seja a melhor.

## Principais opções tomadas

Uma das principais opções tomadas foi na função *calcula\_fit()* que é utilizada para calcular o custo. Temos de ter em conta todos os vértices anteriormente calculados e os vértices que interessam para o cálculo de maneira que dê para contar o número de arestas corretamente.

## Resultados Obtidos

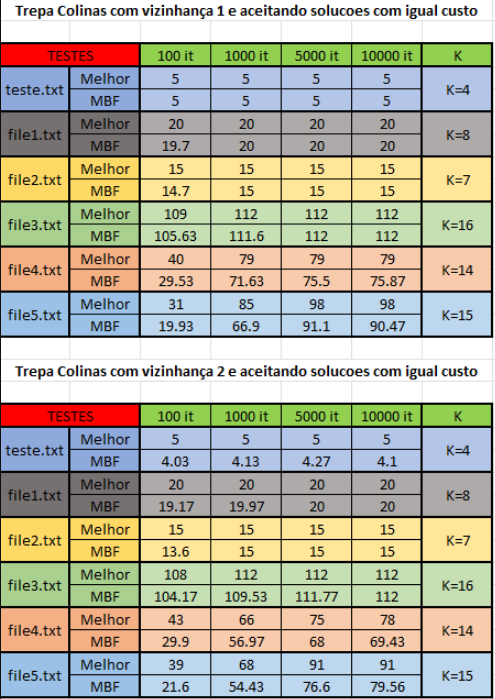


Figura - Resultados do trepa colinas

# Algoritmo Evolutivo

Este algoritmo segue o conceito de sobrevivência do mais apto, dentro de uma população específica, com o objetivo de gerar soluções cada vez mais eficientes. Em cada geração de um algoritmo genético, é criado um conjunto de soluções através do processo de seleção, no qual os indivíduos são escolhidos de acordo com sua capacidade de solucionar o problema. Esses indivíduos são então reproduzidos utilizando operadores derivados de mecanismos de adaptação natural, como mutação e crossover. O objetivo final é encontrar a solução ideal para o problema em questão.

## Principais opções tomadas

Uma das principais opções tomadas foi na função *eval\_individual()* que é utilizada para calcular o custo. Temos de ter em conta todos os vértices anteriormente calculados e os vértices que interessam para o cálculo de maneira que dê para contar o número de arestas corretamente.

# Resultados Obtidos

Uma imagem com mesa

Descrição gerada automaticamente

Figura - resultados teste.txt evolutivo

Uma imagem com mesa

Descrição gerada automaticamente

Figura - resultados file1.txt evolutivo

Uma imagem com mesa

Descrição gerada automaticamente

Figura - resultados file2.txt evolutivo

Uma imagem com mesa

Descrição gerada automaticamente

Figura - resultados file3.txt evolutivo

# Algoritmo Híbrido (combinação dos anteriormente abordados)

O algoritmo híbrido combina o Trepa Colinas com o Algoritmo Evolutivo para aproximar-se da solução ótima para o problema em questão.

## Principais opções tomadas

Uma das principais opções tomadas foi na função *eval\_individual()* que é utilizada para calcular o custo. Temos de ter em conta todos os vértices anteriormente calculados e os vértices que interessam para o cálculo de maneira que dê para contar o número de arestas corretamente.

## Resultados obtidos

Uma imagem com texto, montado

Descrição gerada automaticamente

Figura - resultados hibrido

# Conclusão

Após analisar os resultados obtidos, podemos tirar algumas conclusões importantes. Primeiro, o algoritmo Trepa Colinas mostrou-se mais eficaz em arquivos de tamanho pequeno, mas teve um declínio significativo em conjuntos de dados maiores. Além disso, observamos que quanto maior a probabilidade de recombinação no algoritmo evolutivo, pior eram os resultados, pois isso aumentava a chance de soluções inválidas. Também percebemos que, quanto menor era a população, mesmo aumentando o número de gerações, o algoritmo se tornava menos eficiente, o que indica que o tamanho da população é um fator importante para alcançar soluções ótimas. A mutação não teve grande impacto nos resultados, apenas ocasionalmente causando uma queda insignificante na qualidade. No entanto, o algoritmo híbrido mostrou uma notável melhoria nas soluções e aproximou-se dos valores ótimos. Para atingir esses valores ótimos com o algoritmo híbrido, seria necessário aumentar o número de iterações.