

Introdu¸c˜ao `a Inteligˆencia Artificial

TP2 - Problema de Otimiza¸c˜ao

2021 - 2022

**Nuno Santos Pedro Nogueira**

Licenciatura de Engenharia Inform´atica 15 de janeiro de 2022

**´Indice**

[1 Introdu¸c˜ao 4](#_Toc93241357)

[2 Algoritmo de Pesquisa Local (Trepa Colinas) 4](#_Toc93241358)

[2.1 Vizinhan¸ca 1 5](#_Toc93241359)

[2.2 Funções mais relevantes 5](#_Toc93241360)

[2.2.1 Função gera\_sol\_inicial 5](#_Toc93241361)

[2.2.2 Fun¸c˜ao de calcula\_fit 5](#_Toc93241362)

[2.2.3 Função gera\_vizinho 5](#_Toc93241363)

[2.3 Formulação de Hipóteses 6](#_Toc93241364)

[2.3.1. Hipótese 1 6](#_Toc93241365)

[2.3.2 Hipótese 2 6](#_Toc93241366)

[2.3.3 Hipótese 3 6](#_Toc93241367)

[2.3.4 Hipótese 4 6](#_Toc93241368)

[2.4 Verificação das Hipóteses (Experiência) 6](#_Toc93241369)

[2.4.1 Hipótese 1 6](#_Toc93241370)

[6](#_Toc93241371)

[2.4.2 Hipótese 2 7](#_Toc93241372)

[7](#_Toc93241373)

[2.4.3 Hipótese 3 7](#_Toc93241374)

[2.4.4 Hipótese 4 7](#_Toc93241375)

[2.5 Conclusão 8](#_Toc93241376)

[2.5.1 Hipótese 1 8](#_Toc93241377)

[2.5.2 Hipótese 2 8](#_Toc93241378)

[2.5.3 Hipótese 3 8](#_Toc93241379)

[2.5.4 Hipótese 4 8](#_Toc93241380)

[3 Algoritmo Gen´etico 10](#_Toc93241381)

[3.1 Recombinações 10](#_Toc93241382)

[3.1.2 Mutacões 11](#_Toc93241383)

[3.1.3 Torneios 11](#_Toc93241384)

[3.2 Funções mais relevantes 12](#_Toc93241385)

[3.2.1 Função init\_pop 12](#_Toc93241386)

[3.2.2 Função avaliaIndividual 12](#_Toc93241387)

[3.3 Formulação das Hipóteses 12](#_Toc93241388)

[3.3.1 Hipótese 12](#_Toc93241389)

[3.4 Verificação das Hipóteses (Experiência) 13](#_Toc93241390)

[3.4.1 Hipótese 1 13](#_Toc93241391)

[3.4.2 Hipótese 2 13](#_Toc93241392)

[3.4.3 Hipótese 3 13](#_Toc93241393)

[3.4.4 Hipótese 4 14](#_Toc93241394)

[3.4.5 Hipótese 5 14](#_Toc93241395)

[3.4.6 Hipótese 6 14](#_Toc93241396)

[3.5 Conclusão 15](#_Toc93241397)

[3.51 Hipótese 1 15](#_Toc93241398)

[3.5.2 Hipótese 2 15](#_Toc93241399)

[3.5.3 Hipótese 3 15](#_Toc93241400)

[3.5.4 Hipótese 4 15](#_Toc93241401)

[3.5.5 Hipótese 5 15](#_Toc93241402)

[3.5.6 Hipótese 6 15](#_Toc93241403)

[4 Algoritmo H´ıbrido (combina¸c˜ao das duas abordagens anteriores) 17](#_Toc93241404)

[4.1 Formulação das Hipóteses 17](#_Toc93241405)

[4.1.1 Hipóteses 18](#_Toc93241406)

[4.2 Verificação das Hipóteses (Experiências) 18](#_Toc93241407)

[4.2.1 Hipótese 1 18](#_Toc93241408)

[18](#_Toc93241409)

[4.2.2 Hipótese 2 19](#_Toc93241410)

[4.2.3 Hipótese 3 19](#_Toc93241411)

[4.2.4 Hipótese 4 19](#_Toc93241412)

[4.2.5 Hipótese 5 20](#_Toc93241413)

[20](#_Toc93241414)

[4.2.6 Hipótese 6 20](#_Toc93241415)

[4.3 Conclusões 20](#_Toc93241416)

[4.3.1 Hipótese 1 20](#_Toc93241417)

[4.3.2 Hipótese 2 20](#_Toc93241418)

[4.3.3 Hipótese 3 20](#_Toc93241419)

[4.3.4 Hipótese 4 21](#_Toc93241420)

[4.3.5 Hipótese 5 21](#_Toc93241421)

[4.3.6 Hipótese 6 21](#_Toc93241422)

[4.3.7 Nota comum 21](#_Toc93241423)

[5 Conclus˜ao 22](#_Toc93241424)

# Introdu¸c˜ao

Este trabalho consiste em conceber, implementar e testar m´etodos de otimizac¸˜ao que encontrem solu¸c˜oes de boa qualidade para diferentes instˆancias do problema a seguir descrito.

Dado um grafo não direcionado G = (V , A), composto por um conjunto V de vértices ligados entre si por arestas A, um subconjunto SV é chamado de conjunto estável quando não há nenhuma aresta entre os vértices de S. O objetivo do problema é encontrar um conjunto estável S tal que a sua cardinalidade (ou seja, o número de vértices que contém) seja máxima.

O objetivo do **problema do conjunto estável máximo** é portanto de maximização,

# Algoritmo de Pesquisa Local (Trepa Colinas)

O m´etodo de pesquisa local implementado foi o Trepa-Colinas (também conhecido por Hill-Climbing ou “Gradient-Descent”) .

O nome e ideia base provem de uma analogia com a decisão tomada por um agente que, perdido numa encosta, pretende atingir o topo, deslocar-se-á na direção “em que o caminho sabe”.

A implementação deste algoritmo segue a seguinte ordem:

* Parte de um estado inicial dado ou gerado aleatoriamente;
* Gera os estados sucessores do estado atual;
* Através de uma Função de Avaliação, avalia cada estado assim gerado e escolhe o de maior valor;
* Pára quando o estado selecionado tiver um valor inferior ao escolhido na iteração anterior
  + Isto significa que a solução “piorou” e que se “está a descer a colina, em vez de subir”.

Alguns problemas associados a este método são:

* Uma máximo local pode ser atingido sem que corresponda ao máximo absoluto (melhor solução)
* Nos “planaltos” é necessário escolher uma direção aleatoriamente
* Um cume pode ter lados tão inclinados que o passo seguinte conduz ao “outro lado do cume” e não ao seu topo. Neste caso a solução poderá “oscilar” nunca atingido o máximo pretendido

Será usada a variante do Trepa-Colinas “First-Choice” pois será o mais adequado ao nosso problema dado que este:

* Visita vizinhos de forma aleatória;
* Aceita um vizinho de melhor qualidade e termina a iteração, este caso é útil quando a vizinhança é grande mas também sabem que é um algoritmo não determinista.

### Vizinhan¸ca 1

A primeira vizinhan¸ca testada no nosso algoritmo foi uma vizinhan¸ca bastante simples, que se baseia em aleatoriedade apenas.

Esta funciona da seguinte forma: escolhe, aleatoriamente, 2 subconjuntos diferentes A e B, e troca, de A para B e de B para A, um elemento, tamb´em aleat´orio.

Nos testes efetuados ao algoritmo fizemos variar o nu´mero de vizinhos, fizemos variações entre aceitar soluções de custo igual e não aceitar e também penalização.

Em todos os testes realizados neste trabalho foram usadas 10 repeti¸c˜oes de modo a obter equilíbrio estatístico.

Os resultados foram avaliados em compara¸c˜ao com valores base obtidos com os seguintes parˆametros:

* Nvizinhos;
* Nrepetições;
* Aceitar soluções de custo igual;
* Penalização.

### 2.2 Funções mais relevantes

#### 2.2.1 Função gera\_sol\_inicial

Usámos um array dinâmico que é responsável por guardar a nossa matriz inicial.

É gerado um número aleatório entre 2 e o número de vértices que irá. É gerado também um número aleatório entre 0 e o número de vértices -1, irá ficar dentro de um ciclo onde só sairá quando a solução for igual a 1.

#### Fun¸c˜ao de calcula\_fit

Esta função recebe a solução, a matriz e o número de vértices e devolve o custo e o número de ligações (cardinalidade) que existem na solução.

#### 2.2.3 Função gera\_vizinho

É gerado um **pos** aleatório entre 0 e n-1 que é de seguida guardado na variável **pos\_ant**. O **pos** é continuamente gerado enquanto for igual ao **pos\_ant** garantido que ambos sejam índices diferentes. Quando ambos são diferentes, é feita uma substituição dos valores dos respetivos índices, gerando assim um novo vizinho.

### 2.3 Formulação de Hipóteses

#### 2.3.1. Hipótese 1

H1 : Trepa-Colinas com vizinhança 1.

#### Hipótese 2

H2 : Trepa-Colinas com vizinhança 1 e aceitando soluções de custo igual.

#### Hipótese 3

H3 : Trepa-Colinas com vizinhança 1, aceitando soluções de custo igual e com Penalização.

#### Hipótese 4

H4 : Trepa-Colinas com vizinhança 1 e com Penalização

### Verificação das Hipóteses (Experiência)

#### 2.4.1 Hipótese 1

#### Uma imagem com mesa Descrição gerada automaticamente

#### Hipótese 2

#### Uma imagem com mesa Descrição gerada automaticamente

#### Hipótese 3

#### Hipótese 4

### Conclusão

#### 2.5.1 Hipótese 1

H1 :

#### Hipótese 2

#### Hipótese 3

#### 2.5.4 Hipótese 4

# 3 Algoritmo Gen´etico

O algoritmo gen´etico ´e uma t´ecnica para resolu¸c˜ao de problemas que necessitem de otimizac¸˜ao.

E´ baseado na Teoria de Evolu¸c˜ao de Darwin e tem um forte v´ınculo com conceitos da biologia.

Neste algoritmo ao inv´es de melhorarmos uma u´nica solu¸c˜ao, manipulamos uma populac¸˜ao. Uma populac¸˜ao ´e um conjunto de indiv´ıduos (solu¸c˜oes) que por sua vez s˜ao compostos por cromossomas (no caso do nosso problema, pontos).

Os indiv´ıduos da popula¸c˜ao v˜ao-se alterando ao longo de v´arias gera¸c˜oes atrav´es de mecanismo gen´eticos como muta¸c˜ao e crossover. Os melhores indiv´ıduos de cada gera¸c˜ao, escolhidos atrav´es de torneios, s˜ao os que d˜ao origem aos descendentes, ou seja, a pr´oxima gera¸c˜ao ser´a composta por varia¸c˜oes dos melhores indiv´ıduos da anterior.

Nestas experiências o torneio escolhido foi sempre com o tamanho de 2.

Para fazermos este torneio, selecionamos duas soluções aleatórias e comparamos as suas fitness, a solução que tivesse maior fitness iria ser a solução escolhida para as próximas iterações.

### 3.1 Recombinações

Na recombinação (crossover) foi usado um ponto de corte aleatório que consiste em dividir duas soluções pai em dois e dividir ema parte para um filho e a outra parte para outro filho como mostra no exemplo em baixo. Depois da recombinação, existia também uma probabilidade de haver uma mutação, que alterava apenas um valor numa solução.

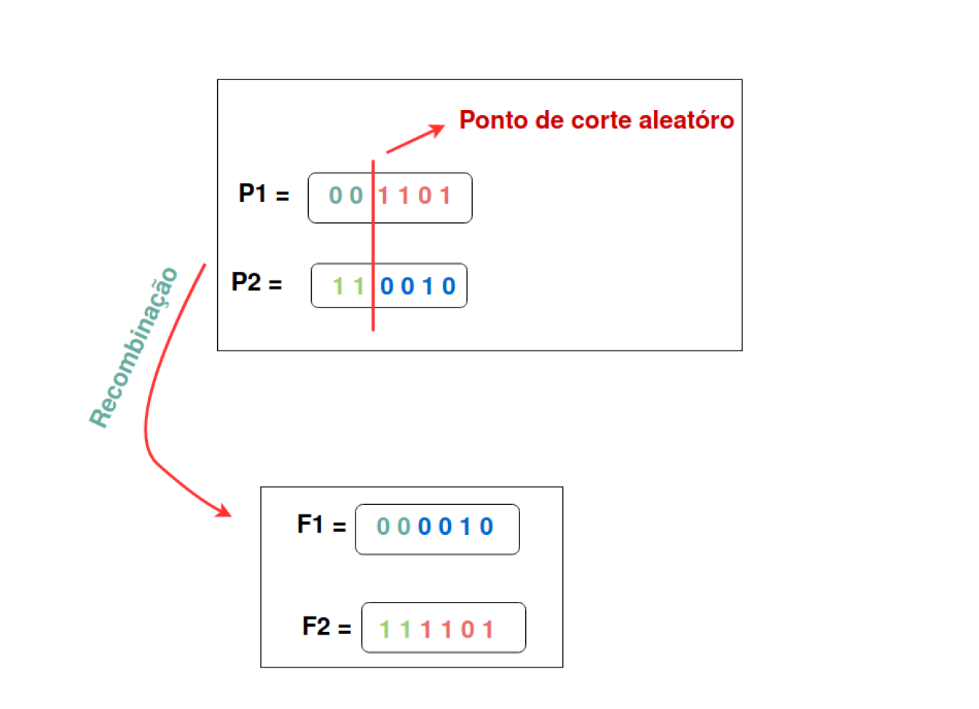


Figura : Exemplo de Recombinação

### 3.1.2 Mutacões

O processo de muta¸c˜ao consiste em alterar a unidade gen´etica mais b´asica (neste caso, o cro- mossoma) do indiv´ıduo. Isto possibilita a mudan¸ca subtil dos indiv´ıduos, permitindo desenvolver solu¸c˜oes diferentes (piores ou melhores) que possam ajudar a guiar as pr´oximas gerac¸˜oes a obter a melhor solu¸c˜ao poss´ıvel.

Os resultados foram avaliados em comparação com valores base obtidos com penalização cega, recombinação 1 e os seguintes parâmetros:

* População = 100;
* Gerações = 2500;
* Prob. Recombinação;
* Prob. Mutação.

### 3.1.3 Torneios

Os torneios servem como m´etodo de sele¸c˜ao: s˜ao usados para escolher os melhores indiv´ıduos de uma determinada gera¸c˜ao e determinar quais ”sobrevivem” para a pr´oxima gera¸c˜ao.

Um torneio consiste em escolher k indivíduos. Esses k indivíduos ”competem” entre eles, e o indiv´ıduo com maior qualidade ”ganha” o torneio e ´e adicionado ao vetor de indiv´ıduos sobreviventes para a pr´oxima gerac¸˜ao.

N˜ao h´a qualquer restri¸c˜ao na repeti¸c˜ao de indiv´ıduos: o mesmo indiv´ıduo pode ser adicionado v´arias vezes ao vetor de sobreviventes, fazendo com que os indiv´ıduos de maior qualidade tenham um peso maior na populac¸˜ao da nova gera¸c˜ao.

De modo a popular a nova gera¸c˜ao com o mesmo nu´mero de indiv´ıduos que a anterior, estes torneiros s˜ao realizados tantas vezes quanto o tamanho da populac¸˜ao, visto que s´o ´e escolhido um indiv´ıduo por torneio.

Um torneio onde k = 2 chama-se torneio bin´ario. Este ´e o tipo de torneio que tem vindo a ser utilizado at´e agora, e que vai ser alterado e testado para v´arios valores de k (tsize).

### 3.2 Funções mais relevantes

#### 3.2.1 Função init\_pop

Esta função serve para criar a população inicial e preencher um vetor binário consoante os valores, a função é idêntica à gera\_sol\_inicial, no entanto cada solução, neste caso é inserida num array que pertence a uma estrutura.

#### 3.2.2 Função avaliaIndividual

Nesta função usámos os mesmos princípios da **calcula\_fit**, uma vez que é responsável por calcular a fitness de uma solução numa população.

Como existe a recombinação e a mutação, há uma probabilidade da solução a ser avaliada ser inválida, logo, foi preciso criar uma função chamada **verifica\_validade** que tem como objetivo validar uma solução e caso esta seja válida o programa prossegue, caso contrário, é marcada como inválida e sai da **avaliaIndividual**.

### Formulação das Hipóteses

Todas hipóteses abaixo efetuadas serão efetuadas para um algoritmo base sem penalização e outro com penalização.

Os parâmetros a testar serão os mesmos para todas as hipóteses (6), mas para ficheiros diferentes, podendo assim ser possível obter uma conclusão consistente e bem fundamentada.

#### Hipótese

* File 1 | File 2 | File 3 | File 4 | File 5 | File 6

Uma imagem com mesa

Descrição gerada automaticamenteUma imagem com mesa

Descrição gerada automaticamente

### Verificação das Hipóteses (Experiência)

#### Hipótese 1

#### Hipótese 2

#### Hipótese 3

#### Hipótese 4

#### Hipótese 5

#### Hipótese 6

### Conclusão

#### 3.51 Hipótese 1

#### 3.5.2 Hipótese 2

#### 3.5.3 Hipótese 3

#### 3.5.4 Hipótese 4

#### 3.5.5 Hipótese 5

#### 3.5.6 Hipótese 6

# Algoritmo H´ıbrido (combina¸c˜ao das duas abordagens anteriores)

Este algoritmo ´e uma combina¸c˜ao dos algoritmos de pesquisa local (Trepa-Colinas) e do algoritmo gen´etico discutidos anteriormente.

Usamos o algoritmo hibrido de duas maneiras diferentes. A primeira foi usada aquando da criação da população inicial, ou seja, em vez de começarmos com uma população aleatória, melhoramos a população incial com o Trepa-Colinas e esta serviu de ponto de partida para o algoritmo evolutivo.

A segunda maneira foi deixar executar o algoritmo evolutivo primeiro e no final usar o trepa colinas na população final e vendo se era possível ainda melhorar um pouco mais a mesma.

Os resultados obtidos ser˜ao comparados com os resultados obtidos para os mesmos parˆametros e m´etodos no algoritmo gen´etico.

A diferen¸ca entre este algoritmo h´ıbrido e o algoritmo gen´etico anteriormente falado ´e que este invoca, em diferentes etapas da execu¸c˜ao, o algoritmo de pesquisa local, de modo a melhorar o indivíduo escolhido.

### 4.1 Formulação das Hipóteses

Todas hipóteses abaixo efetuadas serão efetuadas para o algoritmo base híbrido i) e algoritmo base híbrido ii).

Os parâmetros a testar serão os mesmos para todas as hipóteses (6), mas para ficheiros diferentes, podendo assim ser possível obter uma conclusão consistente e bem fundamentada.

#### 4.1.1 Hipóteses

* File 1 | File 2 | File 3 | File 4 | File 5 | File 6

Uma imagem com mesa

Descrição gerada automaticamenteUma imagem com mesa

Descrição gerada automaticamente

### 4.2 Verificação das Hipóteses (Experiências)

#### 4.2.1 Hipótese 1

#### Uma imagem com mesa Descrição gerada automaticamente

#### 4.2.2 Hipótese 2

Uma imagem com mesa

Descrição gerada automaticamente

#### 4.2.3 Hipótese 3

Uma imagem com mesa

Descrição gerada automaticamente

#### 4.2.4 Hipótese 4

Uma imagem com mesa

Descrição gerada automaticamente

#### 4.2.5 Hipótese 5

#### Uma imagem com mesa Descrição gerada automaticamente

#### 4.2.6 Hipótese 6

Uma imagem com mesa

Descrição gerada automaticamente

### Conclusões

#### Hipótese 1

H1 : É possível observar na tabela da hipótese 1 dos algoritmos híbridos em comparação à tabela da hipótese 1 dos algoritmos evolutivos que o custo manteve-se mas o MBF melhorou , o que faz sentido, uma vez ser esse o nosso objetivo.

#### Hipótese 2

H2 : Nesta hipótese em comparação com a hipótese 2 dos algoritmos evolutivos já verificamos que foi possível obter um custo mais baixo e por sua vez também foi possível obter um MBF melhor.

#### Hipótese 3

H3 : Nesta hipótese também nos é possível observar que em comparação com a hipótese 2 dos algoritmos evolutivos, que foi possível obter um custo mais baixo e também um melhor MBF.

#### Hipótese 4

H4 : Nesta hipótese em comparação com a hipótese 2 dos algoritmos evolutivos já verificamos que foi possível obter um custo mais baixo e por sua vez também foi possível obter um MBF melhor.

#### Hipótese 5

H5 : Nesta hipótese em comparação com a hipótese 2 dos algoritmos evolutivos já verificamos que foi possível obter um custo mais baixo e por sua vez também foi possível obter um MBF melhor.

#### Hipótese 6

H6 : Nesta hipótese em comparação com a hipótese 2 dos algoritmos evolutivos já verificamos que foi possível obter um custo mais baixo e por sua vez também foi possível obter um MBF melhor.

#### Nota comum

Repare que, á medida que o número de vértices/arestas aumenta a diferença de valores obtidos entre o algoritmo evolutivo e o algoritmo híbrido é maior, e faz sentido acontecer, visto que algoritmos híbridos são mais fiáveis em termos de resultados.

Em relação ao algoritmo base híbrido 1 e ao algoritmo base híbrido 2 não se nota resultados muito significativos o que leva a concluir que ambos foram bons comparado com o algoritmo evolutivo.

# Conclus˜ao

Assim, e concluindo o trabalho efetuado, depois de analisados os resultados, marcamos como principais os seguintes pontos gerais:

ˆ Quando o algoritmo se baseia em l´ogica altamente aleat´oria, os resultados n˜ao s˜ao de todo est´aveis e, de certa forma, prejudicam o estudo estat´ıstico dos mesmos;

ˆ Os resultados obtidos dependem largamente da criatividade e qualidade das fun¸c˜oes de viz- inhan¸ca, muta¸c˜ao, recombina¸c˜ao e repara¸c˜ao, sendo que h´a v´arias possibilidades para estas (com qualidade semelhante);

ˆ Em geral, um nu´mero elevado de itera¸c˜oes (caso da pesquisa local) ou gera¸c˜oes / nu´mero de indiv´ıduos da populac¸˜ao aumenta a qualidade final dos resultados, mas requer mais tempo e mem´oria para o algoritmo;

ˆ No caso do algoritmo gen´etico, o aumento da probabilidade de reprodu¸c˜ao e muta¸c˜ao pode ter ou n˜ao efeitos ben´eficos ao problema, devendo ser estudada a sua variac¸˜ao e escolhido o melhor valor para cada fun¸c˜ao de muta¸c˜ao / reprodu¸c˜ao;

ˆ Um bom algoritmo requer, inevitavelmente, uma maior complexidade temporal para melhores resultados em problemas maiores, sendo que, para alcan¸car a solu¸c˜ao ´otima nestes problemas,

´e necess´ario uma grande capacidade de computa¸c˜ao e tempo;

ˆ N˜ao ´e necess´ario um algoritmo com alta complexidade para resolver problemas mais simples, como os dos ficheiros **n010.txt** e **n012.txt**.

Tendo alcan¸cado o objetivo deste trabalho, damos o mesmo por conclu´ıdo.