

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 3](#_Toc93103111)

[2 Algoritmo de Pesquisa Local (Trepa Colinas) 3](#_Toc93103112)

[2.1 Vizinhan¸ca 1 4](#_Toc93103113)

[2.2 Funções mais relevantes 4](#_Toc93103114)

[2.2.1 Função gera\_sol\_inicial 4](#_Toc93103115)

[2.2.2 Fun¸c˜ao de calcula\_fit 4](#_Toc93103116)

[2.2.3 Função gera\_vizinho 4](#_Toc93103117)

[2.3 Formulação de Hipóteses 5](#_Toc93103118)

[2.3.1. Hipótese 1 5](#_Toc93103119)

[2.3.2 Hipótese 2 5](#_Toc93103120)

[2.3.3 Hipótese 3 5](#_Toc93103121)

[2.3.4 Hipótese 4 5](#_Toc93103122)

[2.4 Verificação das Hipóteses (Experiência) 5](#_Toc93103123)

[2.4.1 Hipótese 1 5](#_Toc93103124)

[2.4.2 Hipótese 2 6](#_Toc93103125)

[2.4.3 Hipótese 3 6](#_Toc93103126)

[2.4.4 Hipótese 4 6](#_Toc93103127)

[2.5 Conclusão 7](#_Toc93103128)

[2.5.1 Hipótese 1 7](#_Toc93103129)

[2.5.2 Hipótese 2 7](#_Toc93103130)

[2.5.3 Hipótese 3 7](#_Toc93103131)

[2.5.4 Hipótese 4 7](#_Toc93103132)

[3 Algoritmo Gen´etico 9](#_Toc93103133)

[3.1 Recombinações 9](#_Toc93103134)

[3.1.2 Mutacões 10](#_Toc93103135)

[3.1.3 Torneios 10](#_Toc93103136)

[3.2 Funções mais relevantes 11](#_Toc93103137)

[3.2.1 Função init\_pop 11](#_Toc93103138)

[3.2.2 Função avaliaIndividual 11](#_Toc93103139)

[3.3 Formulação das Hipóteses 11](#_Toc93103140)

[3.3.1 Hipótese 1 11](#_Toc93103141)

[3.3.2 Hipótese 2 11](#_Toc93103142)

[3.3.3 Hipótese 3 11](#_Toc93103143)

[3.3.4 Hipótese 4 11](#_Toc93103144)

[3.3.5 Hipótese 5 11](#_Toc93103145)

[3.3.6 Hipótese 6 11](#_Toc93103146)

[3.4 Verificação das Hipóteses (Experiência) 11](#_Toc93103147)

[3.4.1 Hipótese 1 11](#_Toc93103148)

[3.4.2 Hipótese 2 11](#_Toc93103149)

[3.4.3 Hipótese 3 11](#_Toc93103150)

[3.4.4 Hipótese 4 11](#_Toc93103151)

[3.4.5 Hipótese 5 11](#_Toc93103152)

[3.4.6 Hipótese 6 11](#_Toc93103153)

[3.5 Conclusão 11](#_Toc93103154)

[3.51 Hipótese 1 11](#_Toc93103155)

[3.5.2 Hipótese 2 11](#_Toc93103156)

[3.5.3 Hipótese 3 11](#_Toc93103157)

[3.5.4 Hipótese 4 11](#_Toc93103158)

[3.5.5 Hipótese 5 11](#_Toc93103159)

[3.5.6 Hipótese 6 11](#_Toc93103160)

[3 Algoritmo H´ıbrido (combina¸c˜ao das duas abordagens anteriores) 14](#_Toc93103161)

[4.1 S.A. 1 15](#_Toc93103162)

[4.2 S.A. 2 15](#_Toc93103163)

[4.3 S.A. 3 16](#_Toc93103164)

[4.4 S.A. Misto 16](#_Toc93103165)

[4 Conclus˜ao 17](#_Toc93103166)

# 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

#### 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 uma 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

#### Hipótese 2

#### Hipótese 3

#### 2.4.4 Hipótese 4

### Conclusão

#### 2.5.1 Hipótese 1

#### 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.

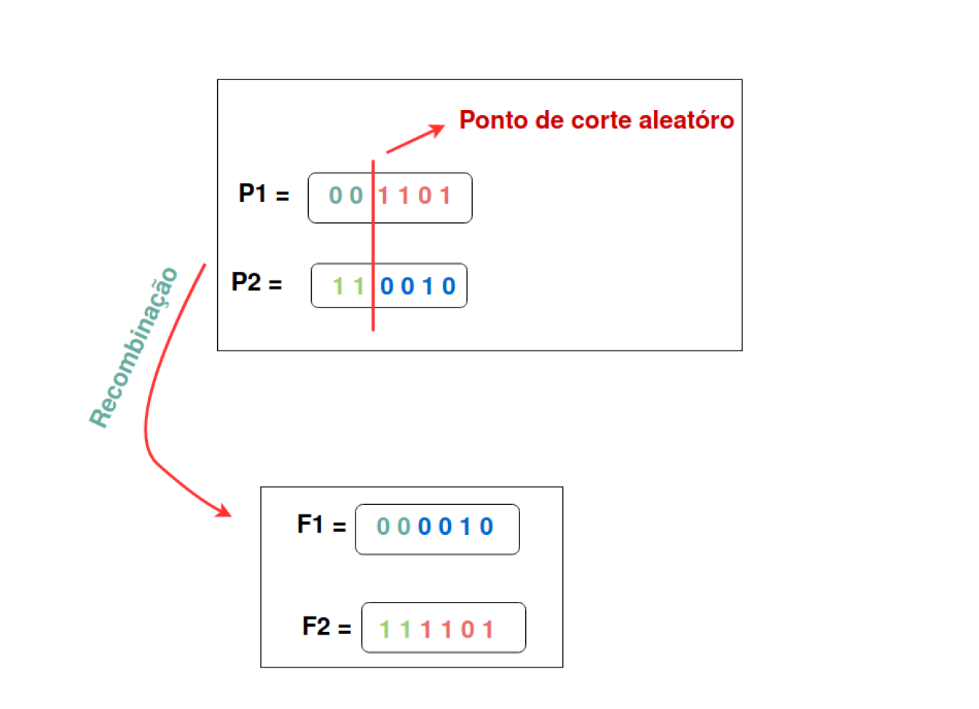


Figura : Exemplo de Recombinação

### 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.

### 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 incial e preencher um vetor binário consoante os valores

#### 3.2.2 Função avaliaIndividual

Nesta função usamos 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, 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

#### Hipótese 1

H1 : (file1.txt)

#### Hipótese 2

#### Hipótese 3

#### Hipótese 4

#### Hipótese 5

#### Hipótese 6

#### 

### 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.

Os m´etodos escolhidos foram os que obtiveram melhores resultados e os valores escolhidos para os parˆametros foram os que pareciam ter melhor rela¸c˜ao qualidade dos resultados / tempo de execu¸c˜ao, ou seja:

ˆ Algoritmo gen´etico:

* Repara¸c˜ao 2;
* Recombina¸c˜ao Uniforme;
* Muta¸c˜ao Aleat´oria;
* Popula¸c˜ao = 100;
* Prob. Reprodu¸c˜ao = 0.2;
* Prob. Muta¸c˜ao = 0.1;
* Tsize = 10.

ˆ Algoritmo de pesquisa local (recristaliza¸c˜ao simulada):

* Vizinhanc¸a 2;
* Arrefecimento geom´etrico;
* Nvizinhos = 10;
* F atualiza = 0.5;
* Tmax = 50;
* Tmin = 5.

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 indiv´ıduo escolhido.

## 4.1 S.A. 1

Nesta vers˜ao do algoritmo h´ıbrido o algoritmo de pesquisa local ´e invocado para melhorar a solu¸c˜ao inicial do algoritmo gen´etico.

Os testes foram realizados usando os m´etodos de muta¸c˜ao, repara¸c˜ao, recombina¸c˜ao, vizinhan¸ca e arrefecimento mencionados anteriormente.

Assim, os valores base obtidos foram os seguintes:

|  |  |  |
| --- | --- | --- |
| **Ficheiro** | **Melhor** | **MBF** |
| **n010.txt** | 1228 | 1228.00 |
| **n012.txt** | 1000 | 1000 |
| **n030.txt** | 5194 | 5179.42 |
| **n060.txt** | 18513 | 18394.66 |
| **n120.txt** | 45516 | 45477.28 |
| **n240.txt** | 149252 | 149082.46 |

Em compara¸c˜ao com os valores base nota-se uma ligeira melhoria nos resultados obtidos.

## 4.2 S.A. 2

Nesta vers˜ao do algoritmo h´ıbrido o algoritmo de pesquisa local ´e invocado para melhorar a melhor solu¸c˜ao de cada execu¸c˜ao.

Os testes foram realizados usando os m´etodos de muta¸c˜ao, repara¸c˜ao, recombina¸c˜ao, vizinhan¸ca e arrefecimento mencionados anteriormente.

Assim, os valores base obtidos foram os seguintes:

|  |  |  |
| --- | --- | --- |
| **Ficheiro** | **Melhor** | **MBF** |
| **n010.txt** | 1228 | 1228.00 |
| **n012.txt** | 1000 | 1000.00 |
| **n030.txt** | 5194 | 5176.78 |
| **n060.txt** | 18505 | 18461.82 |
| **n120.txt** | 45480 | 45308.18 |
| **n240.txt** | 149669 | 148835.66 |

## 4.3 S.A. 3

Nesta vers˜ao do algoritmo h´ıbrido o algoritmo de pesquisa local tem chance de ser invocado entre gerac¸˜oes para um ou mais indiv´ıduos de modo a melhorar a qualidade total da solu¸c˜ao.

Os testes foram realizados usando os m´etodos de muta¸c˜ao, repara¸c˜ao, recombina¸c˜ao, vizinhan¸ca e arrefecimento mencionados anteriormente.

Assim, com uma probabilidade de 10% de ser invocado, os valores base obtidos foram os seguintes:

|  |  |  |
| --- | --- | --- |
| **Ficheiro** | **Melhor** | **MBF** |
| **n010.txt** | 1228 | 1228.00 |
| **n012.txt** | 1000 | 999.68 |
| **n030.txt** | 5194 | 5188.48 |
| **n060.txt** | 18615 | 18425.2 |
| **n120.txt** | 45352 | 45286.3 |
| **n240.txt** | 149993 | 149781.22 |

Em compara¸c˜ao com os valores anteriormente obtidos no algor´ıtmo h´ıbrido, podemos notar uma melhoria razo´avel. Para al´em da melhoria nos valores foi tamb´em verificado um enorme aumento no tempo de execu¸c˜ao.

### 4.4 S.A. Misto

Esta vers˜ao do algoritmo h´ıbrido ´e uma combina¸c˜ao de todas as vers˜oes mostradas anteriormente, ou seja, o algoritmo de pesquisa local ´e invocado para melhorar a solu¸c˜ao inicial, melhorar a melhor solu¸c˜ao de cada execu¸c˜ao e tem chance de ser invocado entre gerac¸˜oes para um ou mais indiv´ıduos de modo a melhorar a qualidade total da solu¸c˜ao.

Os testes foram realizados usando os m´etodos de muta¸c˜ao, repara¸c˜ao, recombina¸c˜ao, vizinhan¸ca e arrefecimento mencionados anteriormente.

Assim, com uma probabilidade de 10% de ser invocado entre gera¸c˜oes, os valores base obtidos foram os seguintes:

|  |  |  |
| --- | --- | --- |
| **Ficheiro** | **Melhor** | **MBF** |
| **n010.txt** | 1228 | 1228.00 |
| **n012.txt** | 1000 | 999.68 |
| **n030.txt** | 5194 | 5191.9 |
| **n060.txt** | 18612 | 18588.46 |
| **n120.txt** | 45619 | 45381.28 |
| **n240.txt** | 149744 | 149448.14 |

Este algoritmo foi o que apresentou os melhores resultados mas tamb´em demorou substancial- mente mais que todos os outros a executar.

# 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.