Trabalho Prático nº 2

Problema de Otimização

Relatório de Estágio

Francisco José Valente Ferreira

2019113494

Licenciatura em Engenharia Informática

Ramo de Desenvolvimento de Aplicações

Instituto Politécnico de Coimbra

Instituto Superior de Engenharia de Coimbra

09, janeiro de 2024

**Índice**

[Agradecimentos i](#_Toc68777782)

[Resumo iii](#_Toc68777783)

[Abstract v](#_Toc68777784)

[Índice de figuras ix](#_Toc68777785)

[Índice de tabelas xi](#_Toc68777786)

[Acrónimos xiii](#_Toc68777787)

[1 Introdução 1](#_Toc68777788)

[1.1 Entidade de acolhimento 1](#_Toc68777789)

[1.2 Instituto Superior de Engenharia de Coimbra 1](#_Toc68777790)

[1.3 Objetivos e plano de trabalhos 1](#_Toc68777791)

[1.4 Estrutura do relatório 2](#_Toc68777792)

[2 Mais um capítulo 5](#_Toc68777793)

[3 Mais um capítulo 7](#_Toc68777794)

[4 Mais um capítulo 9](#_Toc68777795)

[4.1 Requisitos 9](#_Toc68777796)

[4.2 Arquitetura 9](#_Toc68777797)

[4.3 Tecnologias e ferramentas usadas 9](#_Toc68777798)

[4.4 Implementação 9](#_Toc68777799)

[4.5 Testes e validação 9](#_Toc68777800)

[4.5.1 Método A 9](#_Toc68777801)

[4.5.2 Método B 10](#_Toc68777802)

[5 Mais um capítulo 11](#_Toc68777803)

[6 Conclusões e trabalho futuro 13](#_Toc68777804)

[6.1 Conclusões 13](#_Toc68777805)

[6.2 Trabalho Futuro 13](#_Toc68777806)

[Referências 15](#_Toc68777807)

[Anexos 17](#_Toc68777808)

[Anexo A: Proposta de Estágio A-1](#_Toc68777809)

[Anexo B: Especificação de Requisitos B-1](#_Toc68777810)

# Índice de figuras

[Figura 1.1 Explicação da figura 1](#_Toc68777811)

# Índice de tabelas

[Tabela 4.1 – Tecnologias e ferramentas usadas 9](#_Toc68777812)

# Introdução

No contexto da disciplina de Inteligência Artificial o projeto consistiu em Implementar em código C algoritmos de otimização, usando a trepa colinas, e variações de algoritmos genéticos em especifico o problema de maximização de arestas de um subgrafo.

Neste trabalho foram implementados 3 algoritmos, o trepa colina, um algoritmo genético, e um algoritmo de hibrido, sendo alteradas configurações para obter os melhores valores de otimização.

O objetivo foi efetuar um estudo comparativo aprofundado sobre o desempenho da otimização, usando exemplos disponibilizados de um pequeno conjunto de vértices até elevados valores e testar a otimização entre os diferentes ficheiros.

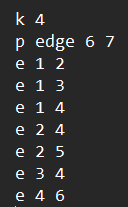
## Estrutura do relatório

O relatório vai falar sobre a representação do problema e abordagens para gerar soluções válidas, seguindo das configurações e resultados para o algoritmo de pesquisa local, configurações e resultados para o algoritmo evolutivo, configurações e resultados para os algoritmos híbridos.

# representação do problema

## Modelo de dados

Os valores dos dados são obtidos através ficheiros com estrutura pré-definida. São obtidos os valores do número de itens do subconjunto de vértices (k), o número vértices p = 6 e o número de arestas. A matriz de adjacências p\*p (Figura 1)é criada, onde são colocados 1s onde há ligações, só é preenchida a parte superior ou inferior, e a diagonal está preenchida a 0 senão significava que um ponto está ligado com ele mesmo e contar como aumento do custo.

 Uma imagem com texto, eletrónica, calculadora, teclado

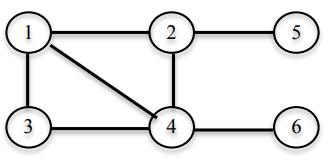
Descrição gerada automaticamente 

Figura Estrutura dos ficheiros, Matriz de adjacências, Desenho do grafo.

A representação de dados consiste num vetor de p bits em que 1 representa que faz parte do subconjunto k, 0 representa que não está. A fitness é a soma das arestas entre todos os elementos 1 desse vetor, de forma a obter mais arestas possíveis, sendo este um problema de maximização do fitness. Uma solução inválida existe quando não é respeitado o número k de elementos num grupo, ao existir mais ou menos que k.

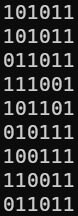


Figura Exemplos de soluções válidas para k = 4

Para obter a fitness da solução vai se a cada elemento do subconjunto k, ou seja, todos os elementos marcados com 1, de seguida procurasse outro elemento marcado com 1 e para a posição i, j dos referidos procurasse na matriz de adjacências que neste caso é um vetor 1d, *row major*.

# Métodos de otimização

## Trepa colinas first choice

O trepa colinas é um algoritmo de otimização heurístico que é utilizado para encontrar o máximo local de uma função. Que vai criando alterações iterativas na solução atual e escolhendo a alteração que resulta em maior fitness na solução.

A variante "first choice" de trepa colinas é um tipo específico de algoritmo de trepa colinas que considera apenas a primeira possível mudança que melhora a solução, em vez de considerar todas as mudanças possíveis e escolher a melhor. Isso pode ser uma estratégia útil quando o espaço de busca é grande e o custo de avaliar as mudanças potenciais é alto, pois permite que o algoritmo progrida mais rapidamente. No entanto, também é mais propenso a ficar preso em máximos locais, pois pode não considerar mudanças alternativas que poderiam potencialmente levar a uma solução melhor.

Para ilustrar como o trepa colinas funciona, exemplo: suponha que desejamos encontrar o valor máximo da função f(x) = x^2 no intervalo [0, 10]. Podemos começar escolhendo um ponto inicial, digamos x = 5, e então fazer alterações iterativas em x para ver se podemos melhorar o valor da função. Se aumentarmos x em 1, encontramos que f(6) = 36, o que é maior que f(5) = 25. Portanto, atualizamos nossa solução atual para x = 6. Em seguida, repetimos o processo, aumentando x em 1 novamente para encontrar que f(7) = 49, o que é maior que f(6). Continuamos esse processo até atingirmos o valor máximo da função em x = 10.

O algoritmo só termina quando o número de iterações especificado termina. Guardando o último melhor fitness que será a solução ótima.

Neste trabalho o algoritmo é lançado várias vezes em diversos pontos aleatórios, tentando assim encontrar ótimos globais.

### Vizinhança 1

A vizinhança um escolhe um valor aleatório numa posição 0 e procura outro valor aleatório numa posição 1 e troca os valores.

### Vizinhança 2

A vizinhança um escolhe um valor aleatório e se o elemento logo a sua direita for diferente troca

### Vizinhança 3

A vizinhança 3 faz uma troca como a vizinhança dois mais envés de ser logo o elemento logo a sua direita é o elemento logo a seguir, ou seja, i + 2.

### Vizinhança 4

A vizinhança 4 escolhe duas posições, uma de 0 até ao meio do vetor e outra acima dessa metade, e inverte esse subconjunto, ou seja, faz um reverse do subconjunto selecionado.

## Algoritmo genético

O algoritmo genético é composto por 3 fases. A Recombinação genética é o processo de misturar os genes de dois indivíduos diferentes para criar um novo indivíduo com características combinadas de ambos os pais. Isso ocorre naturalmente em seres vivos.

Mutação é o processo de alteração aleatória de um gene em um indivíduo. As mutações podem resultar em características novas ou inesperadas no indivíduo.

Seleção é o processo de escolha de indivíduos com base em suas características para reprodução. Em algoritmos genéticos, isso geralmente é feito de acordo com a aptidão do indivíduo, ou seja, sua capacidade de se adaptar e sobreviver em um determinado ambiente. Indivíduos com características mais desejáveis são mais propensos a serem selecionados.

### Recombinação de ponto único

Envolve a seleção de um único ponto no material genético das duas soluções parentais e a troca do material genético em ambos os lados desse ponto.

### Recombinação de dois pontos

Cruzamento de dois pontos: semelhante ao cruzamento de ponto único, mas envolve a seleção de dois pontos no material genético das duas soluções parentais e a troca do material entre os dois pontos para criar a descendência.

### Recombinação uniforme

Cruzamento uniforme: envolve a seleção aleatória de genes individuais de qualquer um dos pais para criar a prole. Isso pode resultar em descendentes mais semelhantes a um dos pais ou a outro, ou que são uma mistura dos dois.

# experimentação e análise

## Trepa colinas

Na experimentação e análise do trepa colinas inicial, foram feitas para 100, 200, 500, 1000 iterações, com configuração em que aceita apenas maiores, com o vizinho 1, ou seja aleatório trocas entre 1 e 0.

É possível ver o aumento do custo com o numero de iterações, mas a estabilização de da média e até inicio de declínio em maiores iterações em ficheiros mais pequenos(file2, file3).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 100 | Média | Custo | 200 | Média | Custo |
| teste.txt | 5.000000 | 5 | teste.txt | 5.000000 | 5 |
| file1.txt | 19.660000 | 20 | file1.txt | 19.684999 | 20 |
| file2.txt | 13.910000 | 15 | file2.txt | 13.980000 | 15 |
| file3.txt | 105.040001 | 108 | file3.txt | 104.970001 | 110 |
| file4.txt | 31.709999 | 46 | file4.txt | 30.795000 | 46 |
| file5.txt | 20.629999 | 32 | file5.txt | 20.379999 | 32 |
|  | | | | | |
| 500 | Média | Custo | 1000 | Média | Custo |
| teste.txt | 5.000000 | 5 | teste.txt | 5.000000 | 5 |
| file1.txt | 19.730000 | 20 | file1.txt | 19.731001 | 20 |
| file2.txt | 14.116000 | 15 | file2.txt | 14.067000 | 15 |
| file3.txt | 104.942001 | 111 | file3.txt | 104.893997 | 111 |
| file4.txt | 30.782000 | 50 | file4.txt | 30.348000 | 54 |
| file5.txt | 20.686001 | 35 | file5.txt | 20.760000 | 38 |

### Análise de aceitação de maiores custos ou custos iguais.

Análise de diferença entre aceitar maior ou aceitar maior ou igual para 500 iterações. Permitiu verificar o impacto de aceitar soluções iguais o que aumenta as médias. Porém para um determinado número de iterações nem sempre dá o melhor resultado geral pois pode seguir outro caminho que não faria se aceitasse soluções iguais.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 500 | Média | Custo | 500 | Média | Custo |
| teste.txt | 5.000000 | 5 | teste.txt | 5.000000 | 5 |
| file1.txt | 19.730000 | 20 | file1.txt | 19.784000 | 20 |
| file2.txt | 14.116000 | 15 | file2.txt | 14.520000 | 15 |
| file3.txt | 104.942001 | 111 | file3.txt | 105.269997 | 110 |
| file4.txt | 30.782000 | 50 | file4.txt | 30.521999 | 56 |
| file5.txt | 20.686001 | 35 | file5.txt | 20.856001 | 38 |

### Análise de vizinhos

Na análise de vizinhos foi possível análise 4 algoritmos para 500 iterações cada ficheiro, aceitando soluções maiores ou igual, ver que tipo de alterações provoca mais qualidade nas soluções geradas. O vizinho 1 foi obtido nas aulas, que apenas encontra 2 números diferentes e troca o que permite diferenças possivelmente mais relevantes. O vizinho 2 foi o que obteve piores resultados, a diferença colocada no algoritmo é que apenas troca 2 valores diferentes adjacentes o que pode levar a trocas pouco significativas para o mesmo número de iterações.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | Média | Custo | 2 | Média | Custo |
| teste.txt | 5.000000 | 5 | teste.txt | 3.890000 | 5 |
| file1.txt | 19.784000 | 20 | file1.txt | 18.840000 | 20 |
| file2.txt | 14.520000 | 15 | file2.txt | 12.536000 | 15 |
| file3.txt | 105.269997 | 110 | file3.txt | 102.209999 | 108 |
| file4.txt | 30.521999 | 56 | file4.txt | 16.332001 | 40 |
| file5.txt | 20.856001 | 38 | file5.txt | 9.160000 | 23 |

O vizinho 3 aumentou a distância do vizinho 2, ou seja a troca de dois elementos, mas envés de diretamente adjacentes foi incrementado um bit para a direita, o que melhorou notoriamente a qualidade de resultados da média. O vizinho 4 faz um reverse de um ponto da metade inferior até um ponto da metade superior, o pode aumentar a diferença das soluções e por vezes escapar soluções pois faz alterações muito significativas, para ficheiros maiores teve mais dificuldade.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 3 | Média | Custo | 4 | Média | Custo |
| teste.txt | 4.164000 | 5 | teste.txt | 4.918000 | 5 |
| file1.txt | 19.058001 | 20 | file1.txt | 19.242001 | 20 |
| file2.txt | 12.116000 | 15 | file2.txt | 12.640000 | 15 |
| file3.txt | 102.671997 | 108 | file3.txt | 101.585999 | 106 |
| file4.txt | 18.548000 | 36 | file4.txt | 19.488001 | 34 |
| file5.txt | 11.490000 | 32 | file5.txt | 14.128000 | 30 |

# Mais um capítulo

…

# Conclusões e trabalho futuro

...

## Conclusões

...

## Trabalho Futuro

…

# Anexos

Proposta de Estágio

… conteúdo Anexo A

… conteúdo Anexo A

… conteúdo Anexo A

Especificação de Requisitos

… conteúdo Anexo B

… conteúdo Anexo B

… conteúdo Anexo B