**Introdução à Inteligência Artificial**

Licenciatura em Engenharia Informática -Curso Europeu

2ºAno – 1º Semestre

**2019/2020**

T R A B A L H O  P R Á T I C O  N º  2

# Problema de Otimização

## Minumum Bandwitdth Problem

*Ana Videira – a21250074*

## Introdução

Na sequência da unidade curricular de Introdução à Inteligência Artificial foi proposto a realização deste segundo trabalho pratico em linguagem C. Este trabalho prático tem como objetivo conceber, implementar e testar métodos de otimização referidos durante as aulas.

De acordo com o enunciado proposto, o problema da largura de banda de grafos ( Minumum Bandwitdth Problem ) é um problema de combinação aleatória cujo objetivo é rotular diversos vértices de um grafo de modo a minimizar a largura de banda do mesmo.

Assim, neste trabalho pratico, é explorado este problema com recurso a diversos algoritmos de pesquisa explorados durante o período letivo. As fichas 7 e 8 realizadas durante as aulas praticas foram um precioso guia para a realização do programa necessário á implementação do métodos exigidos.

## Representação & Objetivo do Problema

No início do programa é pedido ao utilizador para indicar os parâmetros a testar como, por exemplo, o ficheiro texto a utilizar , as iterações necessárias, entre outros.

Com estas informações clarificadas são carregados o número de vértices, de arestas e uma matriz de adjacências. Esta matriz de adjacências possui a informação sobre quais dos vértices formam cada aresta.

Tendo como objetivo minimizar a banda larga de um grafo, diminuindo a maior diferença entre os vértices adjacentes, foi criada uma solução inicial aleatória de inteiros não repetidos que rotulam cada um dos vértices.

Assim, através da função de avaliação e de algoritmos como o Trepa colinas, é possível saber o custo da solução gerada e melhorá-la de acordo com o pretendido.

## Função de Avaliação

*A função avaliação está presente no ficheiro evaluate.c e evaluate.h*

A função de avaliação desenvolvida tem como guia a função matemática apresentada no enunciado: $B_f(G) = max\{ |f(v_i) - f(v_j)| : v_iv_j \in E\}$

A função de avaliação recebe a solução gerada, a matriz de adjacências, o número de vértices e de arestas. Assim procede então a calcular o modulo da diferença dos valores dos dois vértices que foram uma aresta. Á medida que calcula esta diferença em cada aresta, guarda sempre o valor máximo encontrado. Finalmente devolve esse valor máximo encontrado, devolve o custo da solução.

Assim esta função permite aos algoritmos utilizados ajustar a solução gerada para que seja possível um valor máximo cada vez menor (problema de minimização).

## Trepa-Colinas

*A função deste algoritmo está presente no ficheiro localSearch.c e localSearch.h*

Trepa-Colinas é um algoritmo de pesquisa local em que o seu funcionamento pode ser comprado a um alpinista que pretende chegar ao local mais alto.

Resumidamente, este algoritmo parte de um estado inicial, gera estados sucessores ao estado atual e, através da função avaliação, escolhe o melhor.

No contexto deste trabalho, este algoritmo foi testando sobre condições controladas para a uma melhor análise dos soluções geradas e da chegada ao objetivo requerido: encontrar a solução com o menor custo.

## Trepa-Colinas – Testes

*A função deste algoritmo está presente no ficheiro localSearch.c e localSearch.h*

Todos os teste efetuados sobre este algoritmo tiveram como parâmetros:
*Vértices: 5, 39,49,118,199 e 234*
*Runs: 10*
*Iterações: 100, 1000, 5000 e 10000*
*Vizinhanças: 1 e 2*
Foram recolhidos os valores médios a cada 10 runs e a melhor solução encontrada.

Através da análise dos valores recolhidos pode concluir-se que as soluções de valores mais baixos, as melhores soluções, são encontradas por o número de iterações máximo: 10000.

| **Tabela Resumo Trepa-Colinas (10000 iterações)** | | | | |
|---|---|---|---|---|
| | **V1** | **V1a** | **V2** | **V2a** |
| 5 vertices | 2 | 2 | 2 | 2 |
| 39 vertices | 22 | 7 | 16 | 8 |
| 49 vertices | 27 | 11 | 20 | 9 |
| 118 vertices | 84 | 51,1 | 80 | 46 |
| 199 vertices | 176 | 139 | 161 | 136 |
| 234 vertices | 175 | 124 | 158 | 120 |

v1 - vizinhança 1
v1a - vizinhança 1 aceitando soluções com o mesmo custo

De acordo com a tabela apresentada podemos também afirmar que conseguimos também melhores resultados , aceitando soluções com o mesmo custo.

## Recristalização Simulada

*A função deste algoritmo está presente no ficheiro localSearch.c e localSearch.h*

Este algoritmo é semelhante ao trepa colinas mas procura evitar estagnar em máximos locais.

Em cada iteração é selecionada aleatoriamente uma nova solução mas a substituição da solução atual diferencia-se do método trepa-colinas: A substituição da solução atual por uma vizinha depende do custo da solução mas também da probabilidade aleatória que se vai ajustando ao longo do decorrer do processo.

Assim, este algoritmos foi também testado sobre determinadas condições de modo a poder serem retiradas as conclusões necessárias.

## Recristalização Simulada – Testes

*A função deste algoritmo está presente no ficheiro localSearch.c e localSearch.h*

Todos os teste efetuados sobre este algoritmo tiveram como parâmetros:
*Vértices: 5, 39,49,118,199 e 234*
*Runs: 10*
*Iterações: 1000 e10000*
*Vizinhanças: 1 e 2*
*Tmax-Tmin: 100-5 ; 10-0.5 ; 1-0.05; 100-0.01 e 0.99-0.0001*
Foram recolhidos os valores médios a cada 10 runs e a melhor solução encontrada.

À semelhança da análise anterior, quanto maior o número de iterações melhores soluções são encontradas. Deste modo foram analisados com maior pormenor os resultados recolhidos com 10000 iterações.

| **Tabela Resumo Recristalização Simulada (10000 iterações)** | | | | |
|---|---|---|---|---|
| | **V1** | **V1a** | **V2** | **V2a** |
| 5 vertices | 2 | 2 | 2 | 2 |
| 39 vertices | 19 | 18 | 16 | 16 |
| 49 vertices | 25 | 23 | 23 | 24 |
| 118 vertices | 80 | 79 | 77 | 75 |
| 199 vertices | 165 | 171 | 167 | 161 |
| 234 vertices | 172 | 173 | 166 | 161 |

v1 - vizinhança 1
v1a - vizinhança 1 aceitando soluções com o mesmo custo

Podemos concluir, através da tabela apresentada que melhores soluções proveem dos testes realizados aceitando soluções de igual custo e de menor diferença entre Tmax e Tmin.

## Trepa Colinas VS Recristalização Simulada

Considerando as melhores soluções de cada algoritmo, consegue-se a seguinte tabela:

| | Trepa Colinas | Recristalização Simulada |
|---|---|---|
| 5 vertices | 2 | 2 |
| 39 vertices | 8 | 16 |
| 49 vertices | 9 | 24 |
| 118 vertices | 46 | 75 |
| 199 vertices | 136 | 161 |
| 234 vertices | 120 | 161 |

Os resultados apresentados proveem de testes com 10000 iterações, vizinhança 2 e aceitando soluções de igual custo.

Assim, podemos afirmar que  o Trepa Colinas, mesmo com as suas limitações superou o algoritmo Recristalização simulada, entregando um custo de solução menor , ou seja melhores soluções .

# Conclusão

Neste trabalho prático houve uma tentativa de implementar os algoritmos de pesquisa local Trepa-Colinas e Recristalização Simulada. Tentando respeitar as regras e conceitos de implementação destes algoritmos foi criado um programa de forma a gerar e avaliar soluções ao problema apresentado.

Ambos os algoritmos apresentam melhores soluções no maior número de iterações possível e aceitando soluções com o mesmo custo. Na recristalização simulada é também ainda importante a diferença ente tmax e tmin: quanto menor a diferença entre tmax e tmin, melhores soluções.

Os resultados entre os dois algoritmos são bastante semelhantes mas podemos considerar como “vencedor” o algoritmo trepa Colinas pois conseguiu as soluções com o custo mais baixo.

Analisando os resultados adquiridos pode concluir-se que , embora não despropositados, os valores adquiridos ficam aquém dos resultados pretendidos. Em comparação aos resultados fornecidos pelos docentes , as soluções encontradas pelo programa aqui descrito não são ótimas o que pode dever-se a possíveis erros de implementação dos algoritmos.