INSTITUTO SUPERIOR TÉCNICO

Mestrado em Engenharia Mecânica

Unidade Curricular de Otimização e Decisão

Tema do projeto:

Max Cut (Adriaensen Datasets)

Part 1: Simplex method solution

Docentes: Professor Duarte Valério

Professor Filipe Santos

Trabalho Realizado por:

Nº	Nome
102560	Tiago Videira
113246	Frederico Kossack

Data de entrega: 2 de Abril de 2025

Ano letivo 2024/2025

Índice

1. Introdução....1
1.1. Descrição do problema Max Cut....1
1.2. Interpretação de dados....1
1.3. Previsão de resultados....2
2. Descrição matemática das condições....3
2.1. Definição de grupo de pontos....3
2.2. Definição de uma aresta cortada....3
2.3. Função de Contagem de Arestas....3
3. Programação Binária – Branch and Cut....3
3.1. Descrição geral....3
3.2. Linearização de Equações....4
3.3. Construção de matrizes Simplex....4
3.4. Transformação em Problema Inteiro....5
4. Meta-heurísticas – Ant Colony Optimization....5
4.1. Descrição geral....5
4.2. Formulação matemática....6
4.2.1. Depósito de feromonas....6
4.2.2. Evaporação de feromonas....6
4.2.3. Função de probabilidades....6
4.3. Implementação....7
5. Resultados....7
5.1. Programação binária (SCIPY)....7
5.2. Programação binária (PULP)....7
5.3. Ant Colony Optimization....8
5.4. Resultados otimizados....8
6. Conclusões....8
7. Bibliografia....9

1. Introdução

1.1. Descrição do Problema Max Cut

O problema Max Cut é um problema de otimização no qual, sendo providenciada uma rede de pontos unidos entre si por arestas, o objetivo é maximizar o número de arestas *cortadas*. Uma aresta pode ser entendida como *cortada* quando une dois pontos assinalados a grupos diferentes. Pode-se portanto entender a linha de corte como uma fronteira que separa dois grupos de pontos. Pelo método de agrupamento de pontos, a linha de corte não necessita ser expressa matematicamente, sendo apenas necessário definir a qual dos grupos -ou lados da fronteira- cada ponto individual pertence.

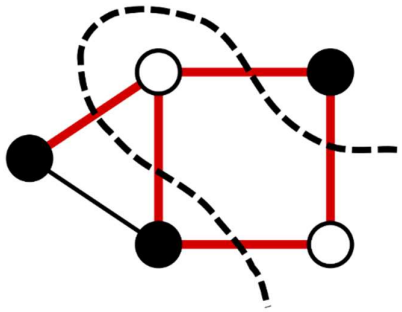

Figura 01. Exemplo gráfico de uma solução Max Cut para uma rede de pontos.

O problema Max Cut pode ser aplicado para uma rede de pontos de qualquer dimensão, caso as arestas entre pontos sejam conhecidas. Isto permite abstrair o problema de uma necessidade de representação gráfica. Redes como as criadas por *point clusters* como os usados em aprendizagem de máquina apresentam frequentemente altas dimensionalidades.

As arestas podem também ser assinaladas um *peso*, o qual indica a importância, ou prioridade de cortar uma determinada aresta. Este valor pode ser negativo. Uma possível analogia para este peso pode ser, por exemplo, um fluxo entre pontos. Nesse caso, o objetivo do algoritmo será maximizar o fluxo total através da fronteira de corte.

1.2. Interpretação de dados

Os dados providenciados para desenvolver esta solução apresentam um formato de uma série de ficheiros *.txt*. Cada um destes ficheiros representa uma rede de pontos e arestas diferentes.

Todos estes ficheiros descrevem as arestas $\boldsymbol{(e_{ij})}$ a considerar e apresentam a mesma formatação. Uma coluna descrevendo pontos de *origem* $\boldsymbol{(P_i)}$, uma segunda coluna descrevendo pontos de *destino* $\boldsymbol{(P_j)}$, e uma terceira coluna com o valor de peso da respetiva aresta entre os pontos $\boldsymbol{(w_{ij})}$. Apesar de neste documento os pontos serem descritos como *origem* e *destino*, estas arestas na realidade não apresentam direcionalidade.

Tabela 01. Exemplo da formatação de uma tabela de dados. A primeira linha descreve uma aresta entre o Ponto 1 e o Ponto 2, com um peso de 246.

P_i	P_j	w_{ij}
1	2	246
1	3	600
2	15	270
...	...	...
3576	3200	700

1.3.Previsão de resultados

Em contraste à solução de método "Simplex", é possível prever que a falta de valores fracionários irá diminuir o valor do resultado obtido obtendo assim resultados mais próximos da realidade. No entanto, antecipamos que o método "Branch and Cut" irá trazer complicações em termos de tempos de processamento de resultados, uma vez que este método é construído a partir de um sistema de resoluções lineares, subsequentemente iterados até atingir uma solução com apenas números inteiros (no nosso caso números binários).

Já o método "Ant Colony", sendo um método meta-heurístico, resolve o problema consoante uma duração desejada, sacrificando tempo por uma melhor performance. Existe um elemento de aleatoriedade no resultado obtido através deste método e, conversamente, uma possibilidade do algoritmo não proceder para além de máximos locais, sendo o equilíbrio entre os dois casos atingido através da definição de constantes apropriadas.

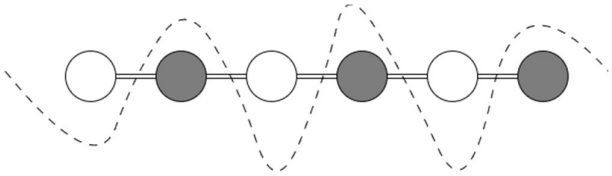

Figura 02. Ilustração da solução prevista numa secção de rede sequecial

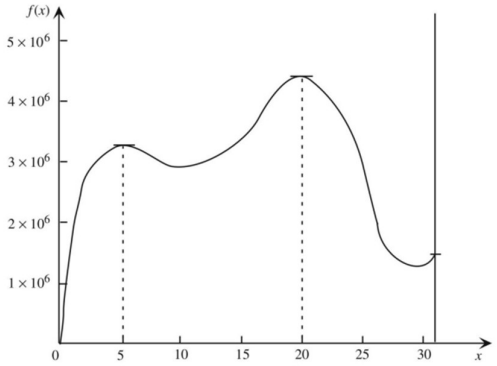

Figura 03. Ilustração de uma solução com vários máximos locais

2. Descrição matemática das condições

2.1. Definição de grupo de pontos

No contexto deste problema, um ponto ficar de um lado da fronteira ou do outro, pode ser representado por assinalar esse ponto a um de dois grupos, por exemplo Grupo A e Grupo B. Sendo apenas possível dois casos, podemos representar esta propriedade através de uma variável x_i, em que i representa o número do ponto que tem o valor **0**, para pertença ao Grupo A e **1** para pertença o Grupo B. Tem-se portanto que

(Eq. 1) $$x_i \in \{0, 1\}$$

2.2. Definição de uma aresta cortada

Uma aresta apenas pode existir em um de dois estados: cortada – ou seja, unindo dois pontos pertencentes a grupos diferentes – ou não cortada – o caso oposto. Tendo que o valor do grupo assinalado de cada ponto pode ser representado em formato binário, semelhantemente tem-se que

(Eq. 2) $$e_{ij} = |x_i - x_j|, \quad e_{ij} \in \{0, 1\}$$

onde i corresponde ao ponto de *origem*, j corresponde ao ponto de *destino* e e_{ij} indica se a aresta em questão é cortada.

Nesta condição, e_{ij} apenas pode resultar num valor de 1, se x_i e x_j tiverem valores distintos.

Tabela 02. Tabela de lógica gerada pela Eq. 2.

x_i	x_j	e_{ij}
0	0	0
0	1	1
1	0	0
1	1	1

2.3. Função de contagem de arestas

O valor da contagem de arestas, neste caso o valor o qual desejamos maximizar, representa a soma de todas as arestas assinaladas como cortadas. No caso do problema incluir um peso para o valor das arestas, este peso é multiplicado ao termo correspondente da aresta. Podemos então descrever esta expressão como

(Eq. 3) $$Maximize\ Z = \sum_i \sum_j w_{ij} e_{ij}$$

3. Programação Binária (Branch and Cut)

3.1. Descrição geral

"Branch and Cut" é uma extensão do método de otimização "Branch and Bound" para problemas de Programação Inteira, combinando técnicas de corte para melhorar a eficiência no processamento de um resultado final. O primeiro passo é a ramificação do problema original, subdividindo-o em subproblemas menores e criando uma *árvore de decisão* onde cada subproblema fixa variáveis

inteiras. Em cada nó da árvore de decisão, resolve-se a relaxação linear do problema, ignorando as restrições inteiras. Se a solução encontrada não for totalmente inteira, verifica-se se o valor obtido é um limite viável. Após a resolução de todos os ramos, estes são descartados caso: o limite inferior for maior que o valor encontrado até ao momento, se a relaxação linear for inviável ou se a solução da relaxação já seja inteira e melhor que outras conhecidas. Se mais nenhum nó possa ser cortado são adicionadas restrições para reduzir a região viável da relaxação linear e forçar soluções inteiras, evitando explorar ramos desnecessários e, por conseguinte, acelerando o processo.

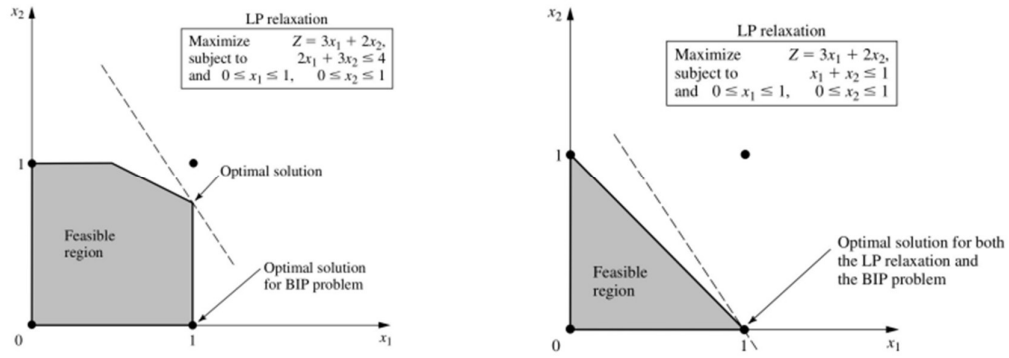

Figura 04. Relaxação linear incluindo a região viável antes (esquerda) e após (direita) a adição de restrições

3.2. Linearização de equações

O método "Branch and Cut", uma vez requerendo primeiramente resolver o problema no formato de Programação Linear, necessita estabelecer uma linearização das equações limitantes acima. No contexto deste problema, existe uma equação não linear, devido à imposição de um valor absoluto (Eq. 2). Sendo o resultado da equação um valor contido em **[0, 1]**, a linearização da condição pode ser feita através das seguintes inequações, impostas em simultâneo:

(Eq. 4) $$e_{ij} \leq x_i + x_j \quad \leftrightarrow \quad -x_i - x_j + e_{ij} \leq 0$$

(Eq. 5) $$e_{ij} \leq 2 - (x_i + x_j) \quad \leftrightarrow \quad x_i + x_j + e_{ij} \leq 2$$

O resultado destas condições pode ser comprovado através da tabela abaixo:

Tabela 03. Tabela de lógica gerada pelas Eq. 5 e Eq. 6.

x_i	x_j	e_{ij} ***(Eq. 5)***	e_{ij} ***(Eq. 6)***
0	0	0	2
0	1	1	1
1	0	1	1
1	1	2	0

Tendo imposto a restrição de que e_{ij} não pode exceder **1**, este valor só igualará **1** quando os pontos que a aresta une pertencem a grupos diferentes.

3.3. Construção de matrizes Simplex

Uma vez obtidas as equações que definem o problema de uma forma linear e não restrita a números inteiros, podemos então expressar o problema em forma de matriz.

As condições têm o formato

(Eq. 6) $$[A]\{x\} = \{b\}$$

Em que {x} contém as variáveis x_i para i correspondendo ao número de pontos existentes; e variáveis e_{ij}, em que ij corresponde a todas as combinações de pontos unidos por uma aresta. Abaixo um exemplo do vetor de variáveis

(Eq. 7) $$\{x\} = \{ x_1 \ x_2 \ x_3 \ \dots \ x_{n\_pontos} \ e_{12} \ e_{13} \ e_{25} \dots e_{n\_arestas}\}$$

A matriz [A] é composta de duas matrizes [A1] e [A2], cada uma correspondente ao grupo de equações geradas pelas **Eqs. 5 e 6**, respetivamente. Ambas têm um número de colunas correspondente ao comprimento do vetor {x}, e um número de linhas correspondente a $2 \times n_{pontos}$, o valor obtido por variar i e j.

O vetor {b} corresponde a um vetor {0 0 0 ... 0} com o mesmo número de linhas que [A1] e um vetor {2 2 2 2 ... 2} com o mesmo número de linhas que [A2], tratando-se estes vetores do lado direito das respetivas inequações.

A função a maximizar pode ser representada com o formato

$Z = \{c\}^T\{x\}$, sendo $\{c\}$ composto de {0 0 0 ... 0}para todas as variáveis x_i e o respetivo valor de $\{w_{ij}\}$ para cada variável e_{ij}.

Uma vez que a função *linprog()* incluída na biblioteca *scipy* – a qual utilizámos para obter a solução deste problema – minimiza o valor de Z, para maximizar, simplesmente foram utilizados os simétricos de $\{c\}$ e Z.

3.4. Transformação em Problema Inteiro

Para transformar os problemas lineares em problemas inteiros utilizamos dois *solvers* diferentes: o *mixed-integer linear programming* (milp) do pacote scipy e o *LpProblem* do *pacote* pulp. Enquanto que o milp é uma pequena variação do código de simplex anterior, o pulp é um biblioteca desenvolvida com a intenção de resolver problemas como estes, não necessitando a pre-assinalação da matriz [A] e o vetor {b} sendo apenas necessária a transcrição das equações linearizadas. Utilizamos estes dois processos diferentes para perceber a diferença entre algo mais familiar e algo mais industrializado.

4. Meta-Heurísticas – Ant Colony Optimization

4.1. Descrição geral

"Ant Colony Optimization" é um nome dado a um algoritmo de otimização, o qual se inspira no método utilizado por formigas para determinar o melhor caminho para um dado objetivo – através da colocação e evaporação de feromonas. Cada formiga toma decisões relativamente à direção em que caminhar, baseando-se nas feromonas que consegue detetar. Ao tomar uma decisão, a formiga deposita feromonas no seu rasto, as quais evaporam com o tempo, caso não sejam colocadas novamente pelas subsequentes formigas.

Abstraindo deste conceito, podemos descrever o sistema como sendo um número de iterações que cria um conjunto de soluções progressivamente inspirados no conjunto anterior (formigas). Cada conjunto adiciona uma constante para a sua decisão, aumentando a probabilidade do próximo conjunto tomar uma decisão semelhante (feromonas). Em cada conjunto gerado, todas estas probabilidades decrescem por igual (evaporação). É importante notar que o "caminho" escolhido em cada conjunto pode ser abstraído para um conjunto de decisões, permitindo utilizar este algoritmo para soluções não-gráficas.

Nesta solução, as decisões tomadas por cada "formiga" correspondem à decisão de assinalar os pontos na extremidade de cada aresta ao grupo A ($x_i = 0$) ou ao grupo B ($x_i = 1$).

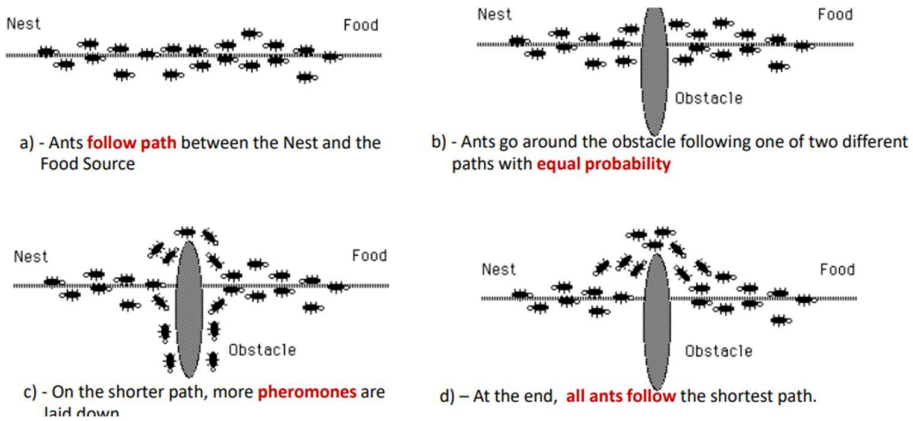

Figura 05. Ilustração do comportamento das formigas na seleção de um itenerário

4.2.Formulação Matemática

4.2.1. Depósito de feromonas

Cada vez que uma formiga assinala pontos ao grupo A ou B, para cada aresta que a formiga decidiu cortar – em que os pontos em cada extremidade da aresta pertencem a grupos diferentes, um valor de feromonas é acrescentado a essa aresta em particular. O valor acrescenta é descrito por:

(Eq. 8) $$\Delta\tau_{ij} = \frac{1}{Z_{max,global}}$$

Em que $Z_{max,global}$ corresponde ao melhor resultado cortado até à presente iteração.

4.2.2. Evaporação de feromonas

As feromonas assinaladas a cada aresta perdem o seu valor com tempo, com base numa constante pré-assinalada, dada pela expressão:

(Eq. 9) $$\tau_{ij}(i+1) = \tau_{ij}(i) * (1 - \rho)$$

Onde $\boldsymbol{\rho}$ corresponde à constante de evaporação.

4.2.3. Função de probabilidades

A probabilidade de uma dada formiga escolher cortar cada aresta, isto é, a probabilidade dos grupos assinalados a ambos os pontos permanecerem inalterados perante o que a formiga anterior escolheu, pode-se representar com a expressão:

(Eq. 10) $$p_{ij} = {\tau_{ij}}^{\alpha} * {w_{ij}}^{\beta}$$

Nesta expressão, são usados os valores de feromona assinalados para cada aresta $\boldsymbol{\tau_{ij}}$, elevado a uma constante pré-assinalada de importância de feromonas $\boldsymbol{\alpha}$; e valor heurístico de peso da aresta $\boldsymbol{w_{ij}}$, elevado a uma constante pré-assinalada de importância heurística $\boldsymbol{\beta}$.

4.3. Implementação

Abaixo, um pseudo-código ilustrativo do funcionamento do algoritmo, benefeciando das expessões acima descritas:

```
Para todas as iterações
|   Para todas as formigas
|   |   Para todas as arestas ij
|   |   |   Calcular probabilidade {0, 1} (Eq. 10)
|   |   |   Se número aleatório < probabilidade ij
|   |   |   |   Assinalar pontos i e j a grupos opostos
|   |   |   Adicionar feromonas se aresta foi cortada (Eq. 8)
|   |   Se Z_formiga > Z_max,global
|   |   |   Atualizar melhor conjunto
|   |   Evaporar feromonas de todas as arestas (Eq. 9)
|   Caso excedidos os limites de tempo, itera
```

5. Resultados

5.1. Programação Binária (SCIPY)

Número de Arestas no dataset	1500	2000	4694
Resultado	1350	1716	Indeterminado
Tempo(s)	15.27	39.28	Indeterminado

5.2. Programação Binária (PULP)

Número de Arestas no dataset	1500	2000	4694
Resultado	1350	1716	Indeterminado
Tempo(s)	24.87	118.06	Indeterminado

5.3. Ant Colony Optimization

Número de Arestas no dataset	1500	2000	4694
Resultado	912/945	945/1223	2898/2952
Tempo(s)	15/25	39/118	30/3600

5.4. Resultados otimizados

Número de Arestas no dataset	1500	2000	4694
Resultado	1350	1716	3052

6. Conclusões

Os métodos de programação binária, apesar de apresentarem resultados ótimos, aumentam o tempo necessário de processamento exponencialmente com o número de variáveis a analisar. Isto porque cada variável imposta a uma condição de número inteiro implica uma nova iteração da solução do sistema de equações/inequações. Para dados de alta complexidade, estas soluções tornam-se inoperáveis em tempos práticos.

As soluções meta-heurísticas apesar de não garantirem um resultado otimizado, são muito menos dependentes da complexidade do dataset, pois o seu número de iterações é linear com os dados. Adicionalmente, a natureza iterativa das soluções meta-heurísticas permite impor restrições costumizadas de interrupção, necessitando apenas grandes tempos de processamento quando desejado por resultados.

Todos os ficheiros usados neste projeto, como scripts, datasets e outputs gerados podem ser encontrados em https://github.com/HoaxFK/Otimization_Decision.

7. Bibliografia

F. Hillier and G. Lieberman (2015) ***Introduction to Operations Research***, 10th Edition. McGrawHill.

Adriaensen, S., Ochoa, G., Nowe, A. (2015). ***A benchmark set extension and comparative study for the HyFlex framework***. 2015 IEEE Congress on Evolutionary Computation, CEC 2015- Proceedings, 784 791.

Y. Wang, K. Li, and W. Wang, "AntCut: An ant colony optimization heuristic for partitioning graphs," *J. China Univ. Geosci.*, vol. 19, no. 4, pp. 331–335, 2008. doi: 10.1016/S1002-0071(08)00221-9.

M. Dorigo, *Real ants, artificial ants & ant colony optimization*. [Online]. Available: https://iridia.ulb.ac.be/~mdorigo/ACO/RealAnts.html. [Accessed: **4/2/2025**].