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: 12 de Março de 2025

Ano letivo 2024/2025

Índice

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 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 matemáticamente, sendo apenas necessário definir a qual dos grupos -ou lados da fronteira- cada ponto individual pertence.

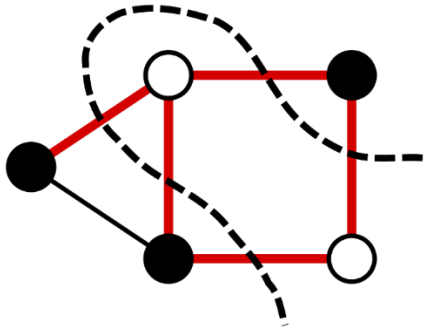

Figura 01. Exemplo gráfico de uma solução Max Cut para uma rede de pontos.

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 apresentavam 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 (e_{ij}) a considerar e apresentam a mesma formatação. Uma coluna descrevendo pontos de *origem* (P_i), uma segunda coluna descrevendo pontos de *destino* (P_j), e uma terceira coluna com o valor de peso da respetiva aresta entre os pontos (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

Uma vez que a solução deste problema foi feita com o algoritmo Simplex, poderíamos prever algumas complicações. Um dos problemas previstos foi em relação à necessidade de expressar a solução em forma binária, como será descrito na próxima secção. Por esta razão, foi possível prever que a solução deste problema pode conter valores fracionais, devido ao relaxament das condições binárias.

Outra previsão possível foi que, no caso da rede de pontos ser aproximadamente uma cadeia uni-dimensional de pontos unidos sequencialmente, a solução que maximiza o número de arestas cortadas nessa secção simplesmente colocaria os pontos em grupos alternados, de modo a cortar todas as arestas. Isto é verdade independentemente dos pesos, a não ser que o peso dado apresente um valor negativo.

Em contraste, é possível prever o comportamento oposto em secções de pontos radiais, onde um ponto partilha o maior número de arestas. Neste caso, a solução local colocará apenas um ponto num grupo, deixando os restantes no segundo grupo.

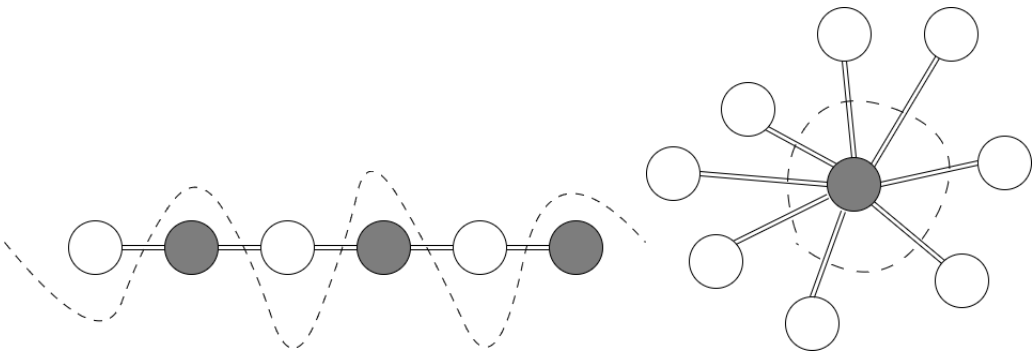

Figura 02. Ilustração da solução prevista numa secção de rede sequecial (esquerda) e radial (direita).

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

Um 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 do ponto 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.

Neste 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. Implementação

3.1. Relaxamento de condições binárias

Uma vez que o algoritmo Simplex apenas é capaz de otimizar sistemas de equações lineares (Linear Programming), o algoritmo não é capaz de convergir em soluções que contêm variáveis únicamente binárias. Isto, pelo menos, sem a adição de algoritmos como Branch & Bound.

De modo a tornar o problema compatível com este algoritmo, é então necessário aplicar um relaxamento de condições. Neste caso, todas as varíaveis binárias serão permitidas qualquer valor real entre **0** e **1**.

(Eq. 4) $$x_i \in [0, 1], \quad e_{ij} \in [0, 1]$$

3.2. Linearização de equações

Como é indicado pelo nome, uma solução de Programação Linear não é capaz de resolver equações não lineares. 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. 5) $$e_{ij} \leq x_i + x_j \quad \leftrightarrow \quad -x_i - x_j + e_{ij} \leq 0$$

(Eq. 6) $$e_{ij} \leq 2 - (x_i + x_j) \quad \leftrightarrow \quad x_i + x_j + e_{ij} \leq 2$$

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

$$[A]\{x\} = \{b\}$$

Em que {x} contém variáveis as 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

$$\{x\} = \{ x_1 \ x_2 \ x_3 \ \dots \ x_{n\_pontos} \ e_{12} \ e_{13} \ e_{25} \dots \ e_{n\_arestas}\}$$

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].

A função a maximizar pode ser representada com o formato

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

Uma vez que a função linprog() incluída na biblioteca *scipy* minimiza o valor, para maximizar, simplesmente foram os simétricos de {c} e Z.

4. Resultados

Abaixo dois pares de tabelas demonstram, cada um, um exemplo de uma rede providenciada na forma de uma tabela descritiva de arestas, com a solução dos grupos assinalados aos respetivos pontos ao lado. Os exemplos escolhidos ilustram um conjunto de arestas com pesos inteiramente positivos, e um com pesos positivos e negativos.

Tabela 04. Dataset ‘g14.txt’ e solução Z=4694 (esquerda) e dataset ‘g.34.txt’ e solução Z=1976(direita).

Origem	Destino	Peso
1	7	1
1	10	1
1	12	1
...	...	...
762	765	1
773	792	1

Ponto	Grupo
1	0.5
2	0.5
3	0.5
...	...
799	0.5
800	0.5

Origem	Destino	Peso
1	1961	-1
1	41	-1
1	40	-1
...	...	...
1998	1999	1
1999	2000	-1

Ponto	Grupo
1	0
2	1
3	0
...	...
1999	1
2000	0

5. Conclusões

É possível através deste problema verificar o sucesso do método Simplex na resolução de problemas de otimização nos quais algumas variáveis são restritas a valores inteiros, específicamente binários.

Enquanto uma expressão não-linear pode muitas vezes ser simplificada para uma expressão linear compatível com o método Simplex, o mesmo não é o caso para as restições de solução binárias.

O relaxamento da condição para um intervalo de números reais entre **0** e **1** produz soluções que não são possíveis no contexto do problema. Neste caso específico, a solução produz pontos que são parcialmente parte de ambos os grupos em frações diferentes. O mesmo se aplica a arestas, as quais são parcialmente contadas como cortadas, resultado dos estados não binários dos pontos que estas unem.

Encontrámos que todas as soluções com pesos de aresta inteiramente positivos geravam soluções de pontos com um valor de 0.5. Isto, cremos, deve-se ao facto de que este valor permite utilizar todas as arestas, independemente da colocação dos pontos que elas unem. Em contraste, datasets com pesos negativos geram soluções com pontos com 1 e 0, pois este método evita cortar arestas indesejadas,

6. Próximas melhorias

Futuras soluções deste problema poderão envolver o método Branch and Bound, o qual resolve uma sequência de soluções simplex, nas quais as variáveis são sequencialmente impostas a obedecer condições que as restringem a valores de **0** e **1**, até convergir numa solução que maximize o resultado. Apesar de não ser simplesmente um algoritmo que testa todas as combinações possíveis, esta solução escala em complexidade rápidamente consoante o número de variáveis individuais no problema, necessitando no mínimo duas soluções Simplex para cada variável imposta a um estado binário. Tem-se portanto que a complexidade mínima de um problema destes é de $C = 2^n$, em que n corresponde ao número de pontos e arestas.

Durante o decorrer desta cadeira, iremos explorar outros métodos de resolver problemas de Programação de Números Inteiros e problemas de Programação de Números Inteiros e Linear mista, como *meta-heuristics*.

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.