Maximum Weight Cut

Hugo Veríssimo - 124348 - hugoverissimo@ua.pt

***Abstract* – abstrato em pt bla bla ns q in English**

***Resumo* – resumo baksdakdjsa in Portuguese**

I. Introdução

Atualmente, os problemas em grafos são amplamente estudados, pelo facto de terem a capacidade de modelar diversas situações reais, desde as mais palpáveis, como redes de computadores (problema *Minimum Spanning Tree*) até às mais abstratas, como física teórica (problema *Maximum Weight Cut*) [1].

Este relatório visa explorar o problema *Maximum Weight Cut*, conhecido em português por Corte de Peso Máximo, que consiste na divisão de um grafo não direcionado, $G(V, E)$, onde $|V| = n$ vértices e $|E| = m$ arestas de peso $w_{i,j} \geq 0 \ \forall \ (i, j) \in E$, em dois subconjuntos complementares, S e T, de forma a maximizar a soma dos pesos das arestas que ligam os dois conjuntos [2], isto é

$$\max \sum_{i \in S,\ j \in T} w_{i,j}$$

$$\begin{cases} S \cup T = V \\ S \cap T = \emptyset \end{cases}$$

Apesar do problema oposto, conhecido como *Minimum Weight Cut*, ter um algoritmo de resolução em tempo polinomial, em certas condições, o problema *Maximum Weight Cut* não o possui, sendo um problema *NP-Hard*. Isto implica que à medida que o tamanho do grafo aumenta, encontrar soluções exatas para este problema tornam-se computacionalmente caras [1].

Ao longo deste relatório, serão abordadas duas estratégias para a resolução do problema: uma busca exaustiva e uma heurística gulosa.

II. org e codigo

ns q gerar graficos, ficheiro no github?
etc etc

III. Algoritmo de Pesquisa Exaustiva

Tendo em conta o algoritmo de pesquisa exaustiva, este visa gerar todas as combinações possíveis de subconjuntos de V e, para cada subconjunto, calcular o peso do corte e comparar com o melhor corte encontrado até ao momento, ou seja, o algoritmo tem duas fases: a geração de todos os subconjuntos possíveis e a avaliação de cada subconjunto.

Este algoritmo garante a obtenção da solução ótima, no entanto, o seu custo computacional é exponencial, sendo impraticável para grafos de grande dimensão.

... Este algoritmo pode ser traduzido em pseudocódigo da seguinte forma:

Algorithm 1 Exhaustive Search

Input: *graph* matriz de adjacencia
Output: s, t, weight cut value

```
input_set ← {0, 1, . . . , len(graph) - 1}
subsets ← EMPTY LIST
n ← LENGTH OF input_set
                                  ▷ Generate all subsets
for r from 0 to n do
    for each subset in combinations(input_set, r) do
        Add subset to subsets
    end for
end for
best ← input_set
weight ← 0
                                  ▷ Evaluate each subset
for each S in subsets do
    new_weight ← CALCULATE_CUT_WEIGHT(graph, S)
    if new_weight > weight then
        best ← S
        weight ← new_weight
    end if
end for
S ← best
T ← input_set - best
return S, T, weight
```

IV. Algoritmo de Pesquisa Gulosa

técnica heurística gulosa (greedy), que procura uma solução aproximada de maneira eficiente, tomando decisões locais que parecem melhores no momento, embora sem garantia de que serão as ótimas globais.

bla bla inspirei me apos mta analise em https://arxiv.org/abs/2312.10895v1

... Este algoritmo pode ser traduzido em pseudocódigo da seguinte forma:

Algorithm 2 Max Weighted Cut Greedy (G)

Input: G matriz de adjacencia
Output: s, t, weight cut value

```
n ← len(G)
                              ▷ Extract edges and their weights
edges ← EMPTY LIST
for i from 0 to n - 1 do
    for j from i + 1 to n - 1 do
        weight ← G[i, j]
        Add (i, j, weight) to edges
    end for
end for
                              ▷ Sort edges in descending order by weight
Sort edges in descending order by weight
seen, S, T ← empty sets
                              ▷ Process each edge
for each (u, v, weight) in edges do
    if u not in seen and v not in seen then
        Add u to S, add v to T
        Update seen with {u, v}
    else if u in S and v not in seen then
        Add v to T, add v to seen
    else if u in T and v not in seen then
        Add v to S, add v to seen
    else if v in S and u not in seen then
        Add u to T, add u to seen
    else if v in T and u not in seen then
        Add u to S, add u to seen
    else          ▷ Both u and v have been seen, skip
    end if
end for
return S, T, CALCULATE_CUT_WEIGHT(G, S)
```

V. qqcena

jdsauhdisa
aa a a
oi

VI. qqcena

dasdsad sn q disse

References

[1] Wikipedia contributors, "Maximum cut — Wikipedia, the free encyclopedia", https://en.wikipedia.org/w/index.php?title=Maximum_cut&oldid=1247744814, 2024, [Online; accessed 27-October-2024].

[2] Noga Alon, Béla Bollobás, Michael Krivelevich, and Benny Sudakov, "Maximum cuts and judicious partitions in graphs without short cycles", *Journal of Combinatorial Theory, Series B*, vol. 88, no. 2, pp. 329–346, 2003.