Maximum Weight Cut Problem

Hugo Veríssimo - 124348 - hugoverissimo@ua.pt

***Abstract* – ... abstrato em ingles**

***Resumo* – Este relatório apresenta a implementação e comparação de dois métodos para resolver o problema *Maximum Weight Cut*: uma pesquisa exaustiva e uma heurística gulosa. O problema *Maximum Weight Cut* con ESTE É O ANTIGO FAZER NOVO**

I. Introdução

ja se analisou no outro relatorio a descrição do problema *Maximum Weight Cut*, e ns q, super fixe

este relatoria visa explorar algoritmos com um certo grau de estocacidade/aletorieda com vista em otimizar a complexidade e as solucoes.

para alem disso os resultados são comparados aos obtidos anteriormente

serao entao implexmentados 3 algoritmos, nomeadamente: ... e ...

II. Metodologia da Análise

vamos usar o python por ter o modulo random e outros

ns q vamos usar os ficheiro tal e tal

e para testar os algortimos serão testados os graficos do Gset e criados por nós com o ficheiro tal

Graphs for the Computational Experiments: mine and elearnig ou links and gset

III. Algoritmo de 1

este algoritmo é o tipico "aleatorio" que consite na geracao de solucoes aleatorias, e a comparacao das mesmas, e a escolha da melhor solucao, nao havendo qualquer componente deterministica (?)

para garantir que n ha solucoes testadas mais q uma vez, as solucoes ja testadas sao guardadas num set, e antes de testar uma nova solucao, verifica se esta ja foi testada, e se sim, passa para a proxima solucao evitando o calculo do peso do corte, uma operacao que é cara

este algortimo ou quando atingir o numero de solucoes a gerar ou quando todas as solucoes possiveis foram testadas, ou seja, quando o numero de solucoes testadas for igual a 2^n

Algoritmo 1 NOME DO ALGORTIMO

Entrada:
- lista de arestas e respetivos pesos (*edges*)
- número de vértices (*n_nodes*)
- número de soluções a gerar (*solutions*)

Saída: subconjuntos S e T, peso do corte (*weight*)

```
best_solution ← None
weight ← 0
seen_solutions ← empty set
for i ← 1 to solutions do
    partition ← random partition of the nodes
    if length(seen_solutions) = 2^n_nodes then
        break
    end if
    partition_hash ← hash the partition
    if partition_hash ∈ seen_solutions then
        continue
    end if
    Add partition_hash to seen_solutions
    new_cut_weight ← compute the cut weight
    if new_cut_weight > weight then
        weight ← new_cut_weight
        best_solution ← copy of partition
    end if
end for
S ← set of nodes assigned to 0 in best_solution
T ← set of nodes assigned to 1 in best_solution
return S, T, weight
```

quando a complexidade, este algortimo, a parte mais cara é o loop que corre no maximo `solutions` vezes, e dentro dele, a complexidade é O(n + m) por gerar uma particao aleatoria e calcular o peso do corte, logo a complexidade final é $O((m + m) \times \texttt{solutions})$, tendendo para $O(n^2)$ para grafos densos e um n grande

IV. Algoritmo de 2

o segundo algortimo a ser implementado é o Simulated Annealing, que é um algoritmo de otimizacao global, que procura a melhor solucao possivel, e que é baseado no processo de arrefecimento de metais, que consiste em arrefecer um metal a uma taxa controlada, para que os atomos se organizem de forma a minimizar a energia do sistema. ns q, o algortimo simulated Annealing consite em ... e é heuristico e ns q e random (referencias)

este algortimo já foi implementado no problema max cut por exemplo [1]

neste caso tem como componente aleatorio a selecao de uma solucao inicial e a aceitacao de solucoes piores, com uma probabilidade que decresce com o tempo, mas

sendo esta aceite smp que a solucao for melhor que a anterior (determinisico)
por ser sensivel a solucao inicial, é interessante testar diferentes solucoes iniciais, e por isso o algoritmo deve ser corrido um numero de vezes
neste algoritmo não ha garantia de que cada sol so é testada uma vez, por opção propria. visto que a probabilidade de testar a mesma solucao é dada por:
...
para grafos de maiores dimensões, esta prob torna-se muito baixa, e por isso a probabilidade de testar a mesma solucao é baixa, fazendo com que o processo de comparar solucoes ja testadas possa prejudicar a eficiencia do algoritmo
o algortimo para quando a temperatura é menor que 10^{-3}, pelo que o numero de iteracoes é variavel e depende do valor da temperatura inicial e da taxa de arrefecimento

Algoritmo 2 *Simulated Annealing*

Entrada:
- lista de arestas e respetivos pesos (*edges*)
- temperatura (*Temp*)
- taxa de arrefecimento (*cooling_rate*)

Saída: subconjuntos S e T, peso do corte (*best_cut*)

`partition` ← random partition of the nodes
`best_partition` ← `partition`
`current_cut` ← compute the cut weight
`best_cut` ← `current_cut`
while `Temp` $> 10^{-3}$ **do**
  `node` ← randomly select a node
  Flip the partition of `node` in `partition`
  `new_cut` ← compute the new cut weight
  `cost_diff` ← `new_cut` − `current_cut`
  if `cost_diff` > 0 **or** random number $\in [0, 1]$ $< e^{\texttt{cost\_diff}/\texttt{Temp}}$ **then** ▷ Accept the move
    `current_cut` ← `new_cut`
    if `new_cut` > `best_cut` **then**
      `best_cut` ← `new_cut`
      `best_partition` ← `partition`
    end if
  else ▷ Reject the move
    Revert the partition of `node` in `partition`
  end if
  `Temp` ← `Temp` × `cooling_rate`
end while
`S` ← set of nodes assigned to 0 in `best_partition`
`T` ← set of nodes assigned to 1 in `best_partition`
return `S`, `T`, `best_cut`

- complexidade on4 ?
tendo em conta a complexidade do algortimo, o gerar uma particao inicial envolve o(n) pq vai vertice a evrtice atribuir determinado subt. depois o valvulo do current cut envolve o(m) visto q corre a lista de todos os vertices
depois corre o loop K vezes
esolhe um nó e muda a sua partição o(1), recalcula o novo peso do corte o(m) e depois compara o novo corte com o anterior o(1) e se for melhor atualiza o corte e a particao o(1)

ou seja a complexidade é O(m) * K

K é dado por:

$$
\begin{aligned}
&T_0 \cdot (\text{cooling\_rate})^k \leq 10^{-3} \\
\Leftrightarrow k &\geq \frac{\log\left(\frac{10^{-3}}{T_0}\right)}{\log(\text{cooling\_rate})} \\
\Leftrightarrow k &= \left\lceil \frac{\log\left(\frac{10^{-3}}{T_0}\right)}{\log(\text{cooling\_rate})} \right\rceil
\end{aligned}
$$

logo a complexidade final é dada por $O(m) \approx O(n^2)$ para grafos densos , pq k é uma constante que não depende nem de m nem de n

V. ALGORITMO DE 3

este algoritmo consiste numa heuristica gulosa, que consiste em iterar por todos os vertices e trocar a sua particao, verificando se a solucao é melhor, e se for, atualiza a solucao, e quanto iterar por todos e nao melhorar em nenhum para.

contudo isto tornava o algoritmo muito lento, apesar dos bons repertidos, para isso foi adicionado um fator de ajuste do maximo de iteracoes (*itLim*), que é o numero de arestas vezes o fator de ajuste, evitando assim que o algoritmo corra indefinidamente

como o algoritmo é guloso, a solucao final depende da solucao inicial, gerada aleatoriamente, e por isso o algoritmo deve ser corrido varias vezes, para garantir uma maior probabilidade a melhor solucao é encontrada

neste algoritmo, como a unica componente aleatoria é a particao inciial e como todas as alteracoes sao feitas em diracao a melhor solucao, o algoritmo nunca ira testar a mesma solucao duas vezes, pelo que nao é necessario guardar as solucoes ja testadas

Algoritmo 3 NOME DO ALGORTIMO

Entrada:
- lista de arestas e respetivos pesos (*edges*)
- número de vértices (*n_nodes*)
- fator de ajuste do máximo de iterações (*itLim*)

Saída: subconjuntos S e T, peso do corte (*weight*)

```
partition ← random partition of the nodes
cut_weight ← compute the cut weight
improved ← True
it_limit ← len(edges) × itLim
while improved and it_limit > 0 do
    it_limit ← it_limit −1
    improved ← False
    for node in range(n_nodes) do
        Flip the partition of node in partition
        new_cut_weight ← compute the cut weight
        if new_cut_weight > cut_weight then
            cut_weight ← new_cut_weight
            improved ← True
            break        ▷ Stop iteration for this node
        end if
        Revert the partition of node in partition
    end for
end while
S ← Set of nodes assigned to 0 in partition
T ← Set of nodes assigned to 1 in partition
return S, T, cut_weight
```

quanto a compelxidade, gerar a particao inicial e calcular o seu peso é O(n + m), pq corre a lista de vertices e a lista de arestas

depois com o ciclo, ira correr no maximo O(itLim x m) e dentro dele a compelxidade é O(n) por correr os nós todos x O(m) por calcular o peso a cada vertice q passa

logo a complixidade final é $O(m \times \texttt{itLim} \times n \times m)$ que tende para $O(n^5)$ para grafos densos

VI. Análise dos Resultados

Compare the results of the experimental and the formal analysis.

todos os grafos devem ser corridos pelo menos 5 vezes, e a media dos resultados deve ser calculada e mediana do tempo , por causa dos tempos e da aleatoriedade dos resultados

Graphs for the Computational Experiments: mine and elearnig and gset

asdasds

A. (1) the number of basic operations carried out

dsadasds

B. 2 the execution time

- Determine the largest graph that you can process on your computer, without taking too much time.
- Estimate the execution time that would be required by much larger problem instances.

dsadasd

C. solution

asdad

C.1 (3) the number of solutions / configurations tested

sadsad

C.2 precision

asdasd

Bibliografia

[1] Tor G. J. Myklebust, "Solving maximum cut problems by simulated annealing", 2015.
URL: https://arxiv.org/abs/1505.03068