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*, [1] 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

- falar de como sao construidos: componente aletoria e determinisica ?

- Ensuring that no such solutions are tested more than once., como fiz isto

- quando é q o algortimo para?

Algoritmo 1 NOME DO ALGORTIMO

Entrada: SHDIASHDIUA
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
```

- complexidade

IV. Algoritmo de 2

dsadasd

Algoritmo 2 Simulated Annealing for Graph Partitioning

Entrada: matriz de adjacência G
Saída: subconjuntos S e T, peso do corte *weight*

Require: List of edges `edges`, number of nodes `n_nodes`, initial temperature `initial_temp`, cooling rate `cooling_rate`, minimum temperature `min_temp`
Ensure: Partition sets `S` and `T`, and maximum cut weight `best_cut`

while `temperature` > `min_temp` **do**
  `node` ← Randomly select a node from `nodes`
  Flip the partition of `node` in `partition`
  `new_cut` ← Compute weight of edges crossing the new partition
  `cost_diff` ← `new_cut` − `current_cut`
  if `cost_diff` > 0 **or** Random number < exp(`cost_diff`/`temperature`) **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

    and revert the partition change
  end if
  `temperature` ← `temperature` × `cooling_rate`
end while
`nodes` ← `range(n_nodes)`
`partition` ← Random assignment of each node to 0 or 1
`current_cut` ← Compute weight of edges crossing the partition
`temperature` ← `initial_temp`
`best_partition` ← `partition`
`best_cut` ← `current_cut`
`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

- falar de como sao construidos: componente aletoria e determinisica ?

- Ensuring that no such solutions are tested more than once., como fiz isto

- quando é q o algortimo para?

V. Algoritmo de 3

...

Algoritmo 3 Random Greedy Optimization for Max Weight Cut

Entrada: matriz de adjacência G
Saída: subconjuntos S e T, peso do corte *weight*

Require: List of edges `edges`, number of nodes `n_nodes`, iteration limit factor `itLim`
Ensure: Partition sets `S` and `T`, and maximum cut weight `cut_weight`

`partition` ← Random assignment of each node to 0 or 1
`cut_weight` ← Compute weight of edges crossing the partition
`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`
    `new_cut_weight` ← Compute weight of edges crossing the partition
    if `new_cut_weight` > `cut_weight` **then**
      `cut_weight` ← `new_cut_weight`
      `improved` ← `True`
      break ▷ Stop iteration for this node
    else
      Revert the partition of `node`
    end if
  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`

- complexidade
- falar de como sao construidos: componente aletoria e determinisica ?
- Ensuring that no such solutions are tested more than once., como fiz isto
- quando é q o algortimo para?

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] J. Buhler e S. Wagon, "Basic algorithms in number theory", *Algorithmic Number Theory*, vol. 44, 2008, https://pub.math.leidenuniv.nl/šstevenhagenp/ANTproc/02buhler.pdf. Accessed: 2024-11-02.