

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* com ESTE É O ANTIPO 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/aleatorieda com vista em otimizar a complexidade e as solucoes.

para alem disso os resultados são comparados aos obtidos anteriormente

serao entao implexmmentados 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 aleatoria 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:

- lista de arestas e os seus pesos (*edges*)

- número de vértices (*n_nodes*)

- número de soluções a gerar (*solutions*)

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

```

1: best_solution ← None
2: weight ← 0
3: seen_solutions ← empty set
4: for i ← 1 to solutions do
5:   partition ← random partition of the nodes
6:   if length(seen_solutions) = 2n_nodes then
7:     break
8:   end if
9:   partition_hash ← hash the partition
10:  if partition_hash ∈ seen_solutions then
11:    continue
12:  end if
13:  Add partition_hash to seen_solutions
14:  new_cut_weight ← compute the cut weight
15:  if new_cut_weight > weight then
16:    weight ← new_cut_weight
17:    best_solution ← copy of partition
18:  end if
19: end for
20: S ← set of nodes assigned to 0 in best_solution
21: T ← set of nodes assigned to 1 in best_solution
return S, T, weight

```

- complexidade

IV. ALGORITMO DE 2

dsadasd

Algoritmo 2 *Simulated Annealing***Entrada:**

- lista de arestas e os seus pesos (*edges*)
- temperatura (*Temp*)
- taxa de arrefecimento (*cooling_rate*)

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

```

1: partition ← random partition of the nodes
2: best_partition ← partition
3: current_cut ← compute the cut weight
4: best_cut ← current_cut
5: while Temp > 10-3 do
6:   node ← randomly select a node
7:   Flip the partition of node in partition
8:   new_cut ← compute the new cut weight
9:   cost_diff ← new_cut - current_cut
10:  if cost_diff > 0 or random number ∈ [0, 1]
    < ecost_diff/Temp then      ▷ Accept the move
11:    current_cut ← new_cut
12:    if new_cut > best_cut then
13:      best_cut ← new_cut
14:      best_partition ← partition
15:    end if
16:  else      ▷ Reject the move
17:    Revert the partition of node in partition
18:  end if
19:  Temp ← Temp × cooling_rate
20: end while
21: S ← set of nodes assigned to 0 in best_partition
22: T ← set of nodes assigned to 1 in best_partition
23: return S, T, best_cut

```

Algoritmo 3 NOME DO ALGORTIMO**Entrada:**

- lista de arestas e os seus pesos (*edges*)
- número de vértices (*n_nodes*)
- fator de ajuste do máximo de iterações (*itLim*)

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

```

1: partition ← random partition of the nodes
2: cut_weight ← compute the cut weight
3: improved ← True
4: it_limit ← len(edges) × itLim
5: while improved and it_limit > 0 do
6:   it_limit ← it_limit - 1
7:   improved ← False
8:   for node in range(n_nodes) do
9:     Flip the partition of node in partition
10:    new_cut_weight ← compute the cut weight
11:    if new_cut_weight > cut_weight then
12:      cut_weight ← new_cut_weight
13:      improved ← True
14:      break      ▷ Stop iteration for this node
15:    end if
16:    Revert the partition of node in partition
17:  end for
18: end while
19: S ← Set of nodes assigned to 0 in partition
20: T ← Set of nodes assigned to 1 in partition
   return S, T, cut_weight

```

- complexidade
- falar de como sao construidos: componente aleatoria 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

V. ALGORITMO DE 3

...

- complexidade

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

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

- quando é q o algortimo para?

...

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/~stevenhagenp/ANTproc/02buhler.pdf>. Accessed: 2024-11-02.