Partição de Grafos

Integrantes

- Diogo Carrer de Macedo
- Jean Santos Diniz
- João Victor Carvalho dos Santos
- Rafael Rodrigues de Souza
- Vítor Pereira Resende

O que é a Partição de Grafos?

- É um problema NP-difícil.
- Dado um grafo não direcionado
- Quantidade par de vértices
- Dois subconjuntos de vértices de tamanho igual
- Minimizar o número de arestas que ligam vértices de subconjuntos diferentes.

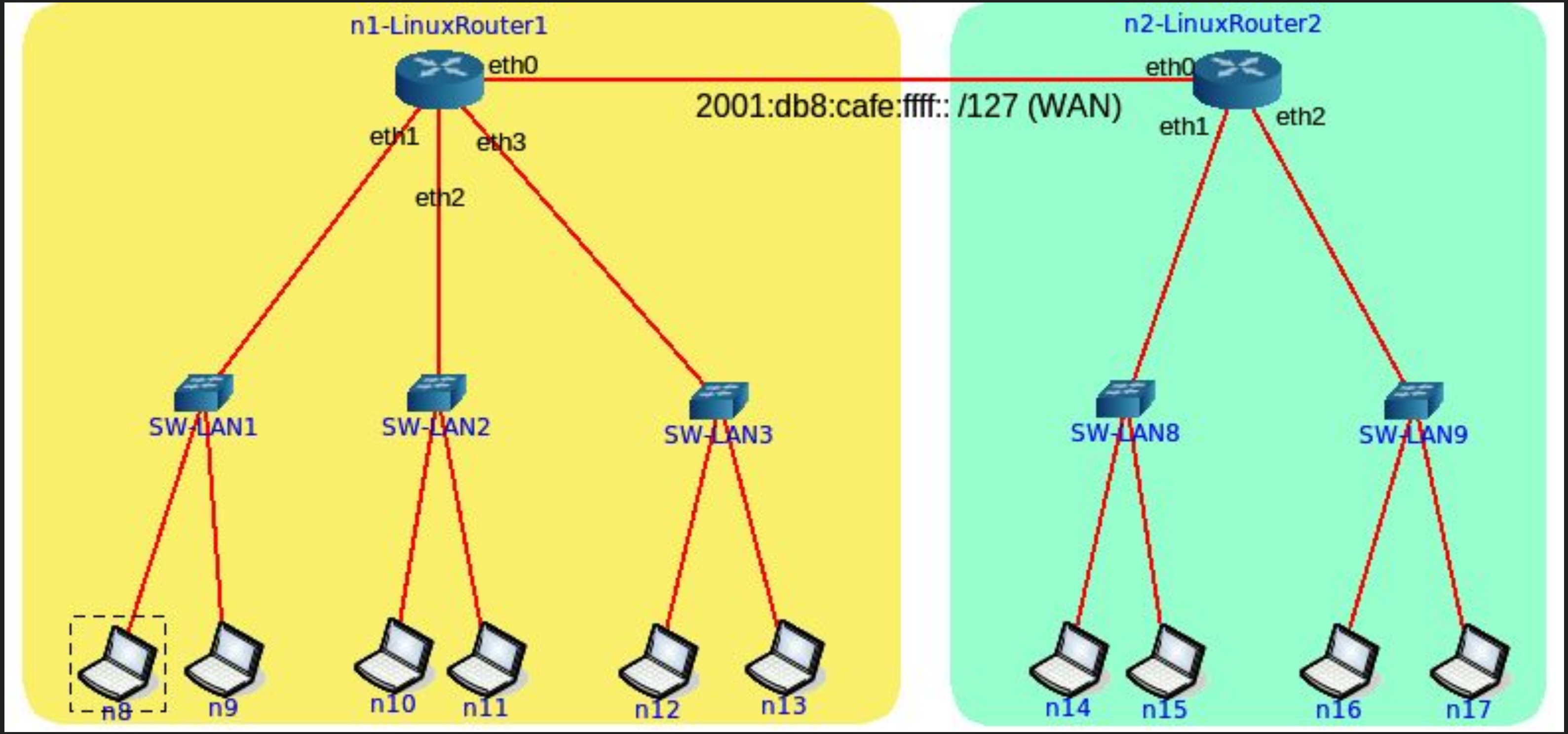
Desafios na Partição de Grafos

- Complexidade computacional
- Equilíbrio entre os subconjuntos
- Minimização das arestas.

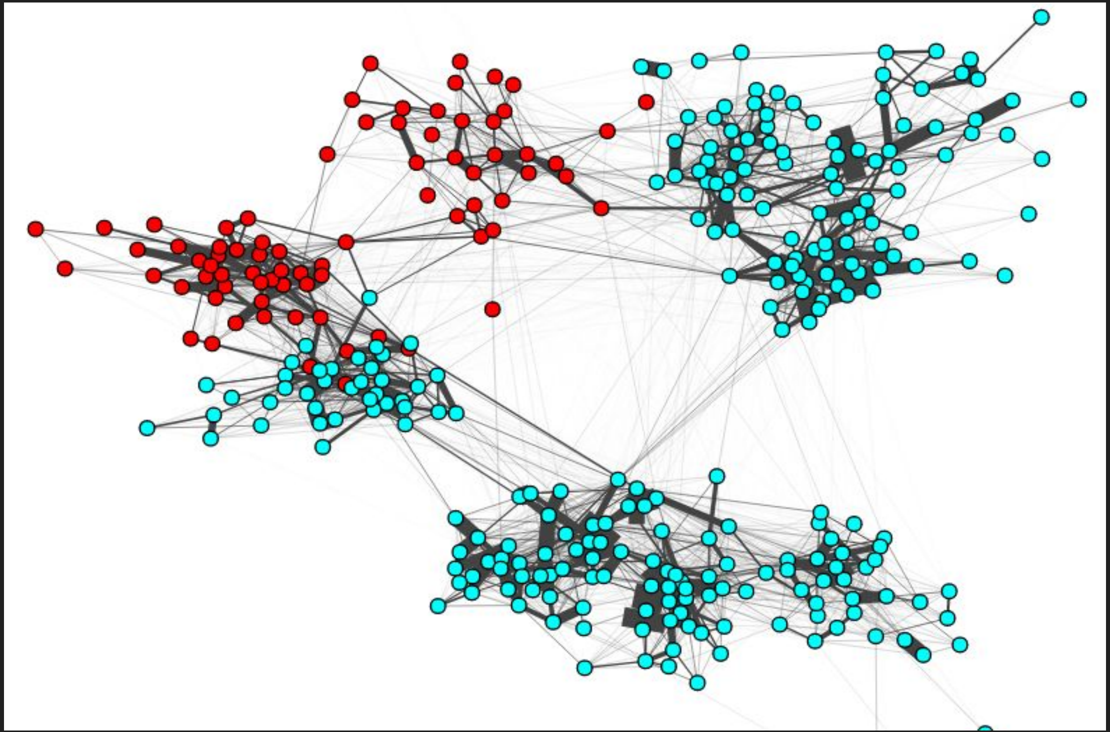
Aplicações reais de Partição de Grafos

- Divisão de redes de computadores em sub redes
- Divisão de circuitos elétricos
- Clusters em análise de dados
- Agendamento de tarefas em sistemas com multiprocessadores
- Associações em redes sociais.

Divisão de redes de computadores.



Associações em redes sociais



Algoritmo de Força Bruta

- Uma abordagem para o algoritmo é gerar todas as combinações possíveis entre as duas partições e identificar aquela que resulta no menor corte.
- Complexidade $O(2^n)$.

Algoritmo de Força Bruta

```
Algoritmo de força bruta
Entrada: grafo G(V,E) com |V| = 2n
Saída:  grafo particionado G(V,E)
particionaForcaBruta:
    corteMinimo ← INFINITO
    particoesOtimas ← par de listas

    FOR para cada combinação p1 e p2 IN combinacoes(G)
        corte ← calculaCorte(p1, p2)

        IF corte < corteMinimo:
            corteMinimo ← corte
            particoesOtimas ← (p1, p2)

    RETURN particoesOtimas
```

Algoritmo de Força Bruta

```
calculaCorte(p1, p2):
    valorCorte ← 0

    FOR para cada vertice v1 IN p1:
        FOR para cada vertice v2 IN p2:
            IF existe aresta entre v1 e v2:
                valorCorte ← valorCorte + 1

    RETURN valorCorte
```

A Heurística de Kernighan-Lin

- Introduzido por B. W. Kernighan e S. Lin em 1970.
- Complexidade $O(n^3)$.
- A estratégia do algoritmo é realizar trocas de vértices entre as duas partições de forma iterativa, buscando minimizar o número de arestas que cruzam a partição.
- Os dois vértices são escolhidos através de uma escolha gulosa (local).
- Se distancia da solução ótima quando o grafo tem muitas soluções quase ótimas.

A Heurística de Kernighan-Lin

Custo Interno

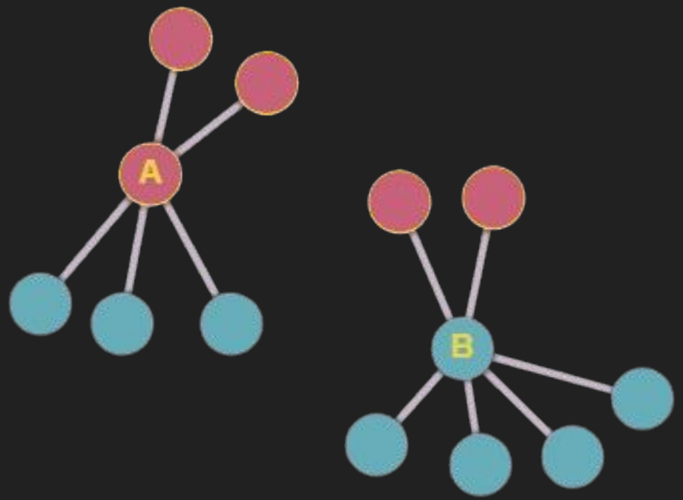
$$I(a) = \sum_{j \in A} m(a, j).$$

Custo Externo

$$E(a) = \sum_{j \in B} m(a, j)$$

Diferença D

$$D(a) = E(a) - I(a)$$



I(a) = 2 | E(a) = 3 I(b) = 4 | E(b) = 2

Custos internos e externos para os vértices A e B

A Heurística de Kernighan-Lin

- Calcular D inicial para todos os vértices.
- Calcular ganhos para cada par de vértices que estejam em partições diferentes.
- Selecionar maior ganho possível.
- Atualizar D para os vértices não trocados e repetir o passo.
- Escolher k de forma a maximizar o ganho total G.
- Realizar as trocas e, se G > 0 executar o algoritmo novamente.

Ganho ao trocar a e b

$$g(a,b) \;=\; D(a) + D(b) - 2m(a,b)$$

Atualizar D a partir de uma troca de a por b

$$D'(i) \;=\; D + 2m(i,a) - 2m(i,b)$$

Ganhos totais

$$G \;=\; \sum_{i=1}^{k} g_i$$

A Heurística de Kernighan-Lin

```
Kernighan_Lin(G) // Grafo G(V,E) com |V| ← 2n
(A,B) ← PARTICAO_INICIAL(G)
G ← INF
trancado ← arranjo de tamanho n

while (G > 0)
  ganhos ← ∅
  foreach (vertice v em V)
    trancado[v] ← false
    D[v] ← calculaD(v, G)
```

```
  for i←1 to n/2
    (gi, (a, b)) ← maiorGanho(G, A, B)
    ganhos.add(a,b)
    trancado[a] ← true
    trancado[b] ← true

    foreach (vertice v em ADJ(a) U ADJ(b))
      D[v] ← atualizaD(G, v, a, b)

  G, k ← ganhoMaximo(ganhos)

  if(G > 0)
    realizarTrocas(ganhos, k)

return A, B  // Duas partições disjuntas A e B
```

A Heurística de Kernighan- Lin

```
calculaD(v, G): // vértice v e grafo G
for cada aresta (v,u) in G
    if (particao(v) == particao(u))
      D[j]--; // Se estão na mesma partição = custo interno
    else
      D[j]++; // Se não estão na mesma partição = custo externo
```

```
atualizaD(G, v, a, b):  // Grafo G, vértice v, vértices trocados a e b
  if(particao(v) == particao(a))
    D[j] ← D[j] + (2 * MA[j][a]) - (2 * MA[j][b])
  else
    D[j] ← D[j] + (2 * MA[j][b]) - (2 * MA[j][a])
```

A Heurística de Kernighan-Lin

```
realizarTrocas(ganhos, k):
// Arranjo de ganhos de tamanho n/2 e índice k que maximize a soma
for(i=1 to k)
  A = A - ganhos[i].u
  B = B - ganhos[i].v

  A = A + ganhos[i].v
  B = B + ganhos[i].u

```

```
maiorGanho(G, A, B): // Grafo G, duas partições A e b
foreach vertice v in A
  if(trancado[v])
    continue

  foreach vertice u in B
      if(trancado[u])
        continue

      ganho ← D[j] + D[k] - 2 * MA[j][k];

      if (ganho > maiorGanho)
      {
        a = j;
        b = k;
        maiorGanho = ganho;
      }
```

Execução Força Bruta

Entrada:

4 (n vértices)

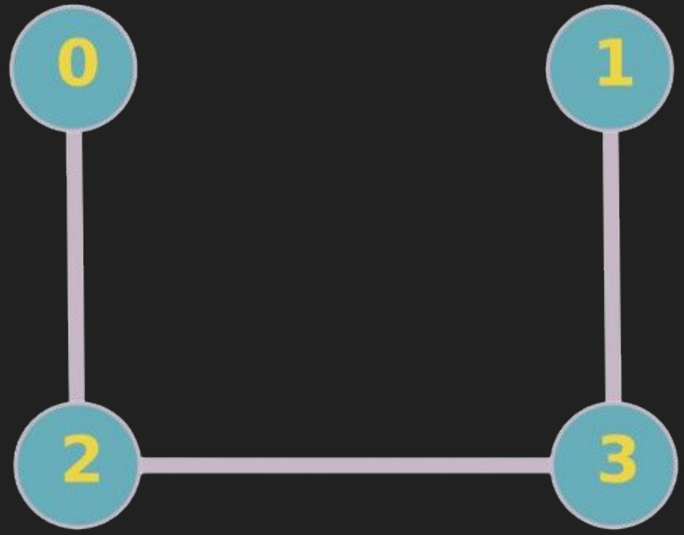
3 (m arestas)

0 2

1 3

2 3

arestas



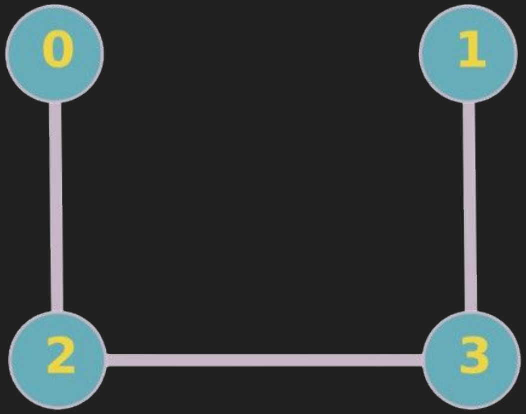
Execução Força Bruta

Particionamentos possíveis:

A = {0, 1}, B = {2, 3}

A = {0, 3}, B = {1, 2}

A = {0, 2}, B = {1, 3}

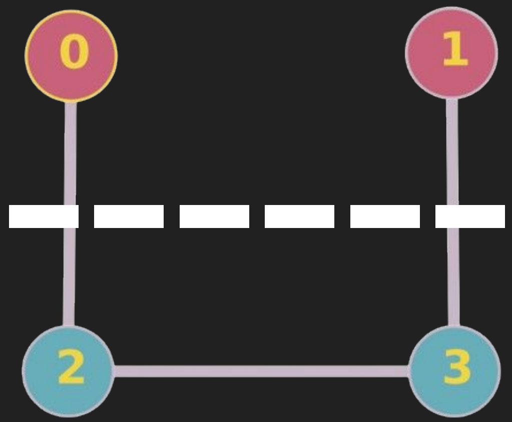


Execução Força Bruta

Particionamentos possíveis:

A = {0, 1}, B = {2, 3}

Corte: 2

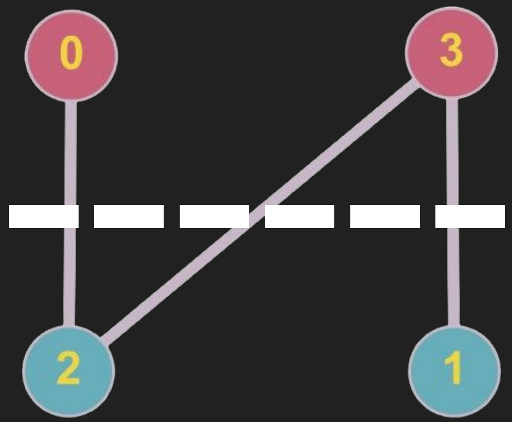


Execução Força Bruta

Particionamentos possíveis:

A = {0, 3}, B = {1, 2}

Corte: 3

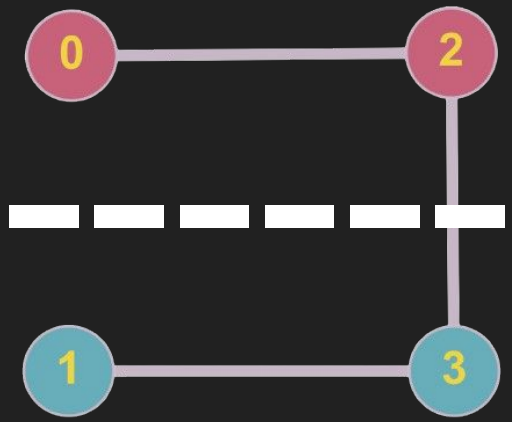


Execução Força Bruta

Particionamentos possíveis:

A = {0, 2}, B = {1, 3}

Corte: 1



Execução Força Bruta

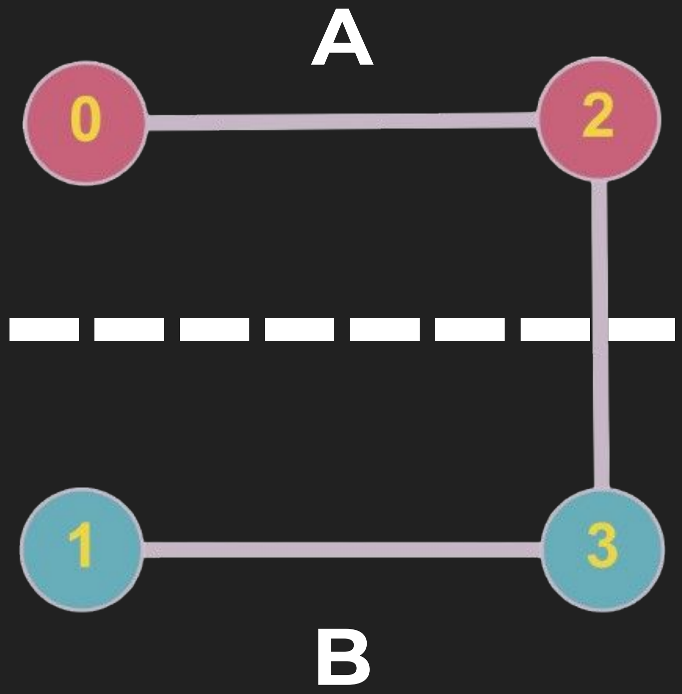
Saída:

Valor do Corte Mínimo: 1

Partição A: 0 2

Partição B: 3 1

O(2^n)



Execução Heurística

Entrada:

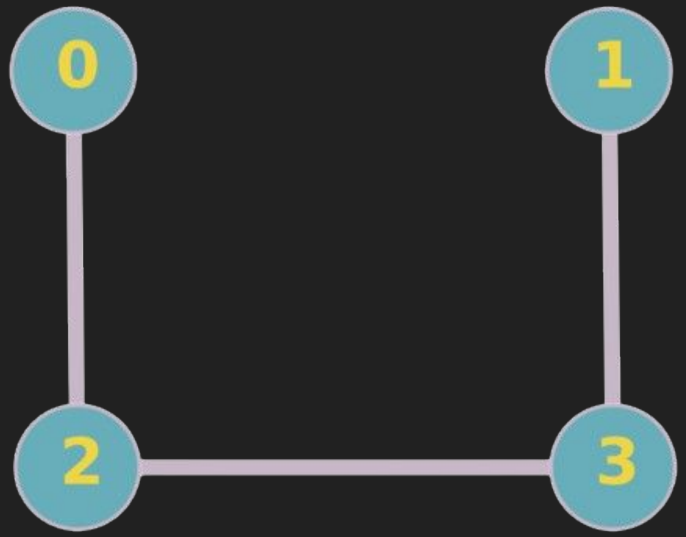
4 (n vértices)

3 (m arestas)

0 2

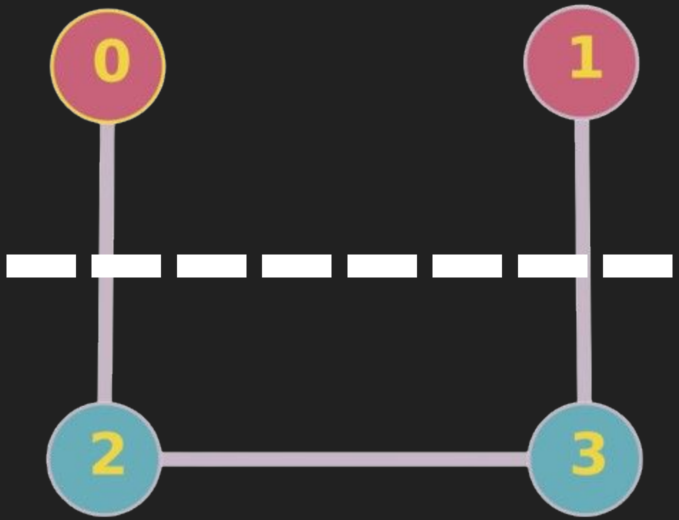
1 3 } arestas

2 3



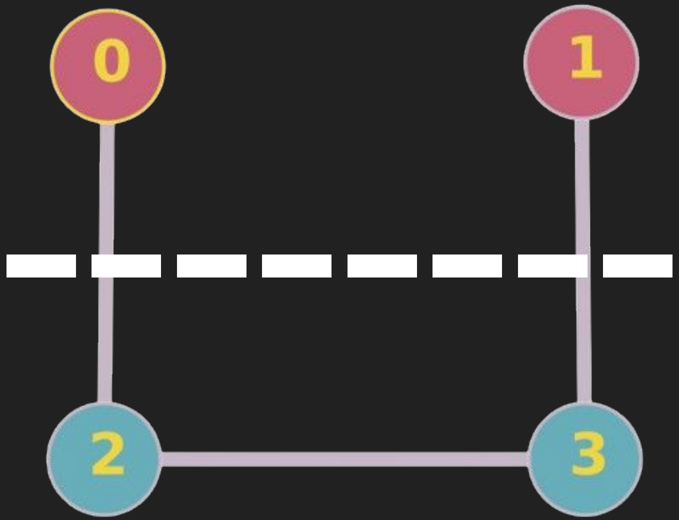
Execução Heurística

Partição Arbitrária



Execução Heurística

Partição Arbitrária



Cálculo de D (diferença):

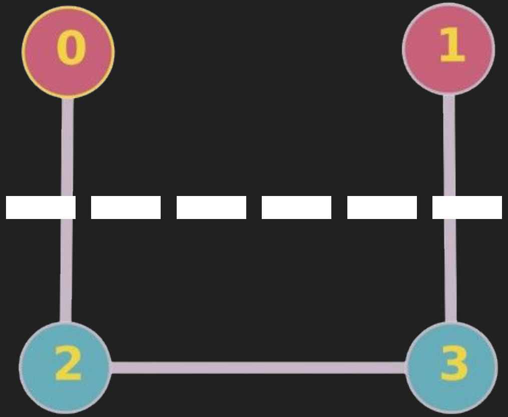
Custo externo - Custo interno

D[0] = 1 - 0 = 1
D[1] = 1 - 0 = 1
D[2] = 1 - 1 = 0
D[3] = 1 - 1 = 0

Execução Heurística

Cálculo de Ganhos

D[j] + D[k] - 2 * MA[j][k]



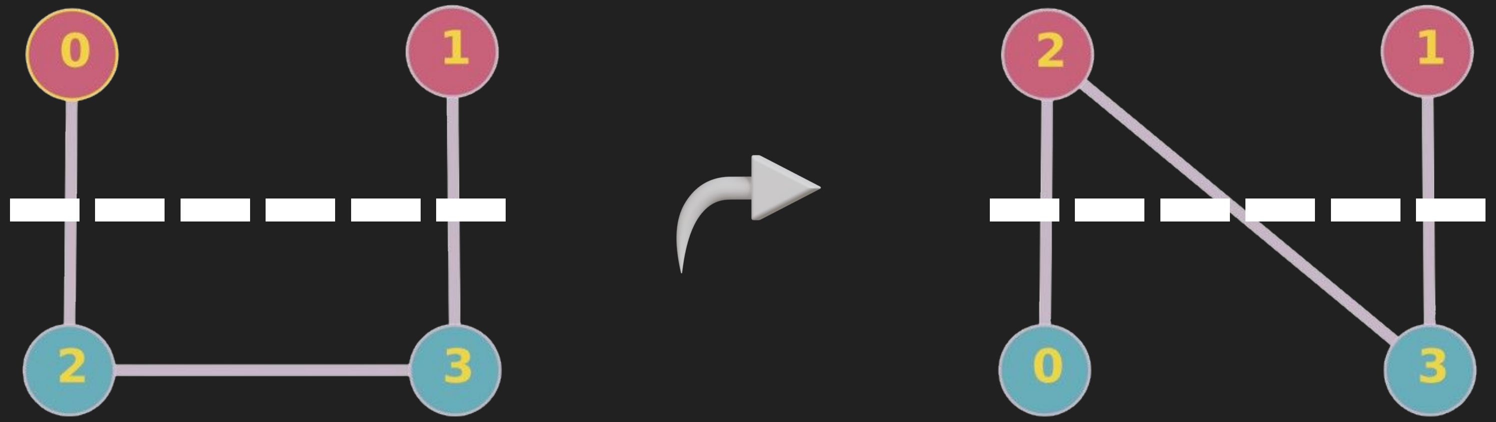
Execução Heurística

Cálculo de Ganhos

D[j] + D[k] - 2 * MA[j][k]

Cálculo de Trocar 0 e 2

1 + 0 - 2 * 1 = -1



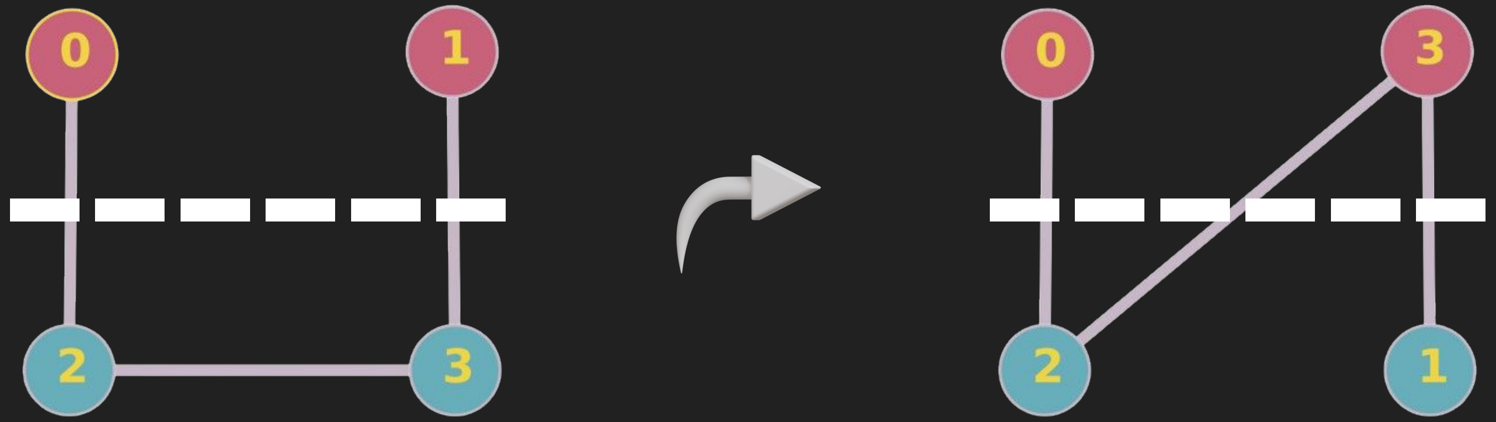
Execução Heurística

Cálculo de Ganhos

D[j] + D[k] - 2 * MA[j][k]

Cálculo de Trocar 1 e 3

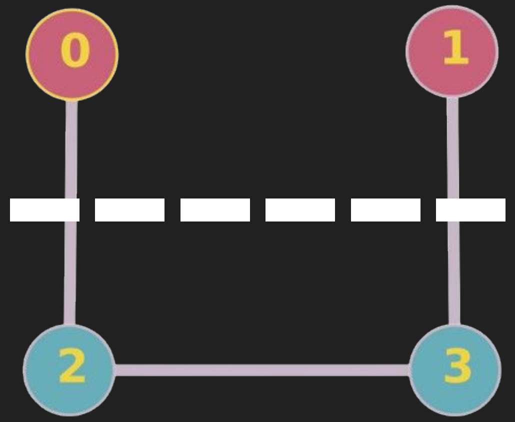
1 + 0 - 2 * 1 = -1



Execução Heurística

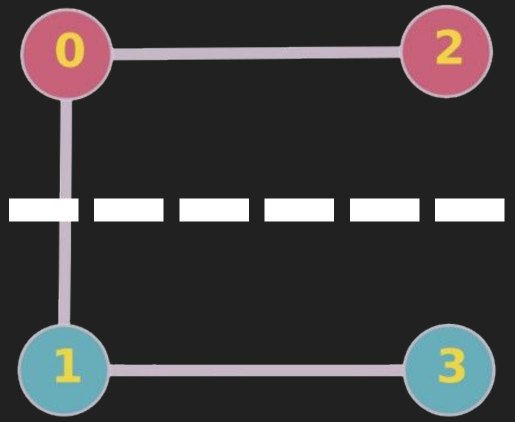
Cálculo de Ganhos

D[j] + D[k] - 2 * MA[j][k]



Cálculo de Trocar 1 e 2

1 + 0 - 2 * 0 = 1



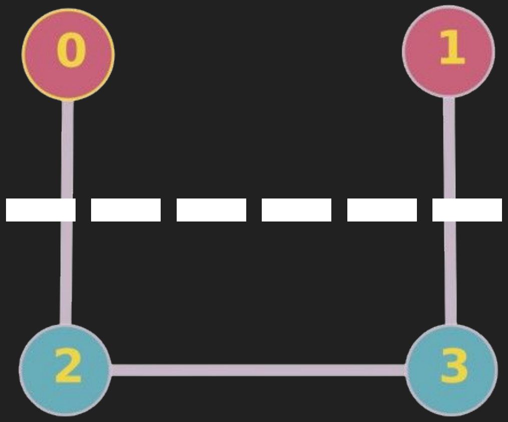
Execução Heurística

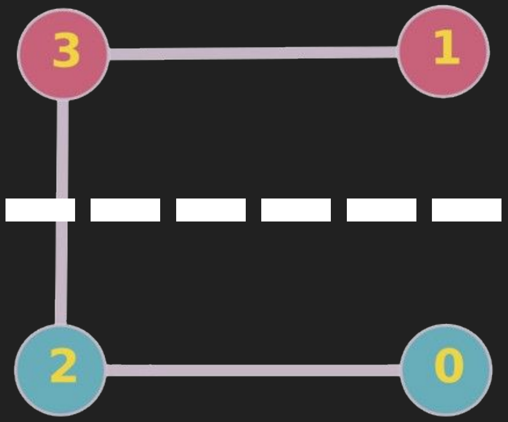
Cálculo de Ganhos

D[j] + D[k] - 2 * MA[j][k]

Cálculo de Trocar 0 e 3

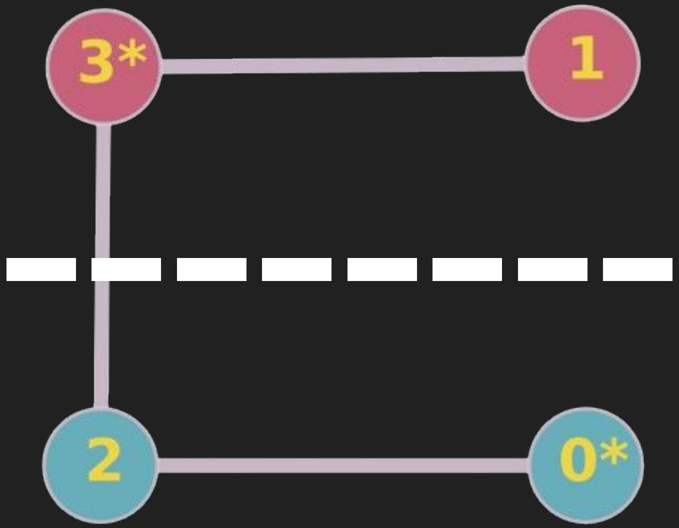
1 + 0 - 2 * 0 = 1





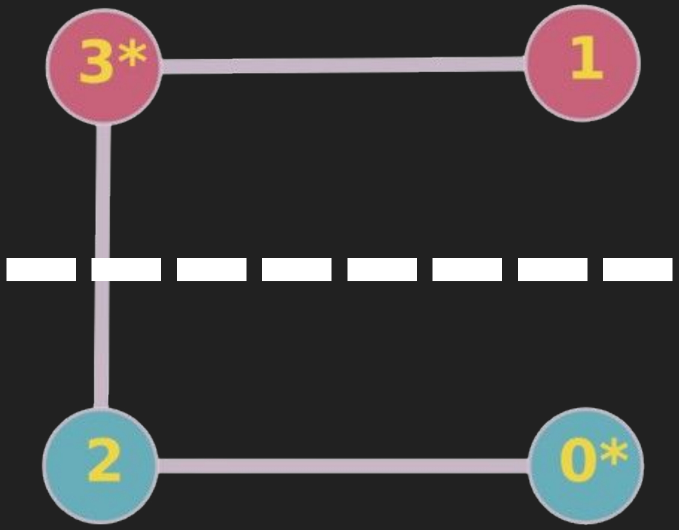
Execução Heurística

Nova Partição



Execução Heurística

Nova Partição



Cálculo de D (diferença):

Custo externo - Custo interno

D[1] = 0 - 1 = -1

D[2] = 1 - 1 = 0

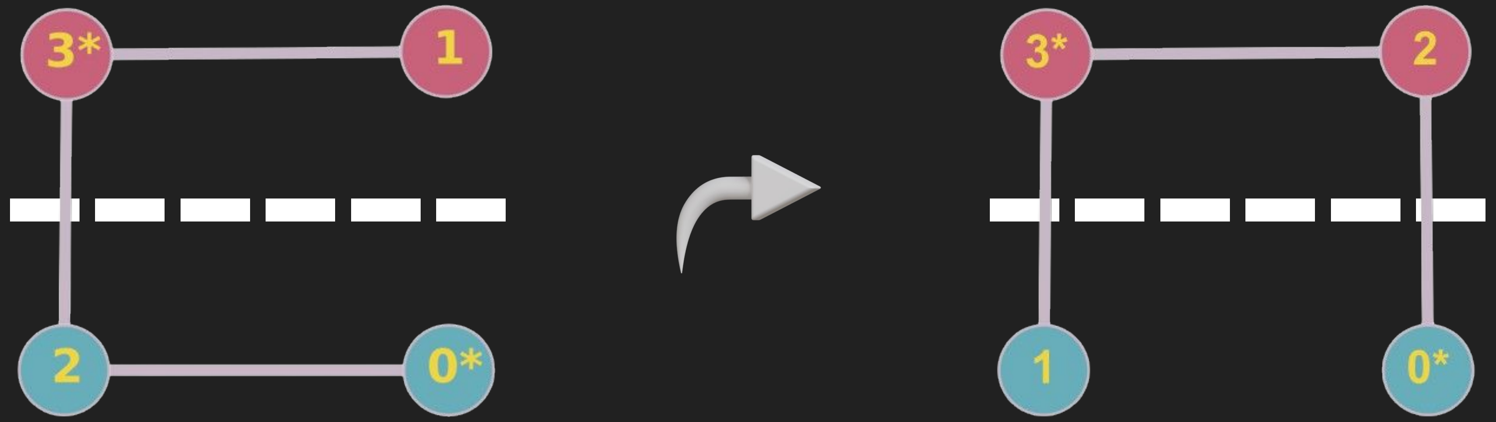
Execução Heurística

Cálculo de Ganhos

D[j] + D[k] - 2 * MA[j][k]

Cálculo de Trocar 1 e 2

-1 + 0 - 2 * 0 = -1



Execução Heurística

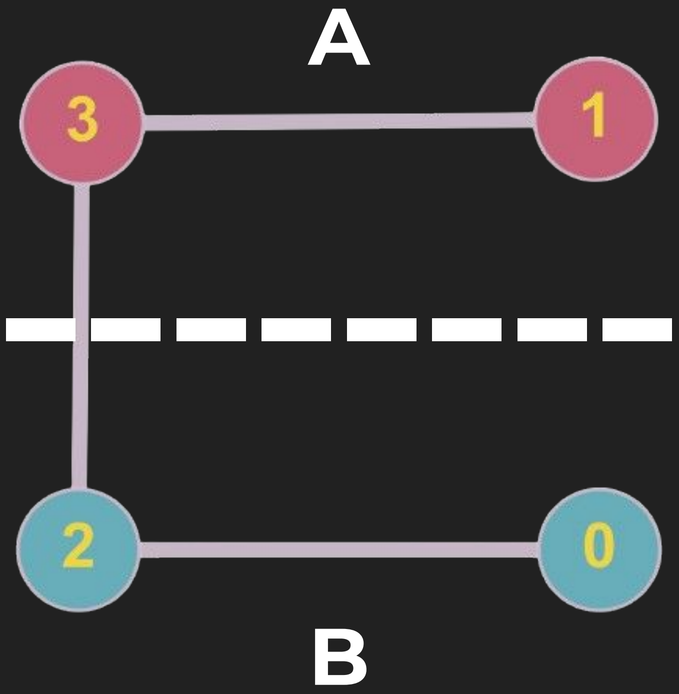
Saída:

Valor do Corte Mínimo: 1

Partição A: 3 1

Partição B: 2 0

O(n^3)



Análise Experimental dos Algoritmos

Especificações do sistema

- Sistema operacional: Linux
- Processador: Intel(R) Core(TM) i5-8265U @ 1.60GHz
- 8,00 GB de RAM

Análise Experimental dos Algoritmos

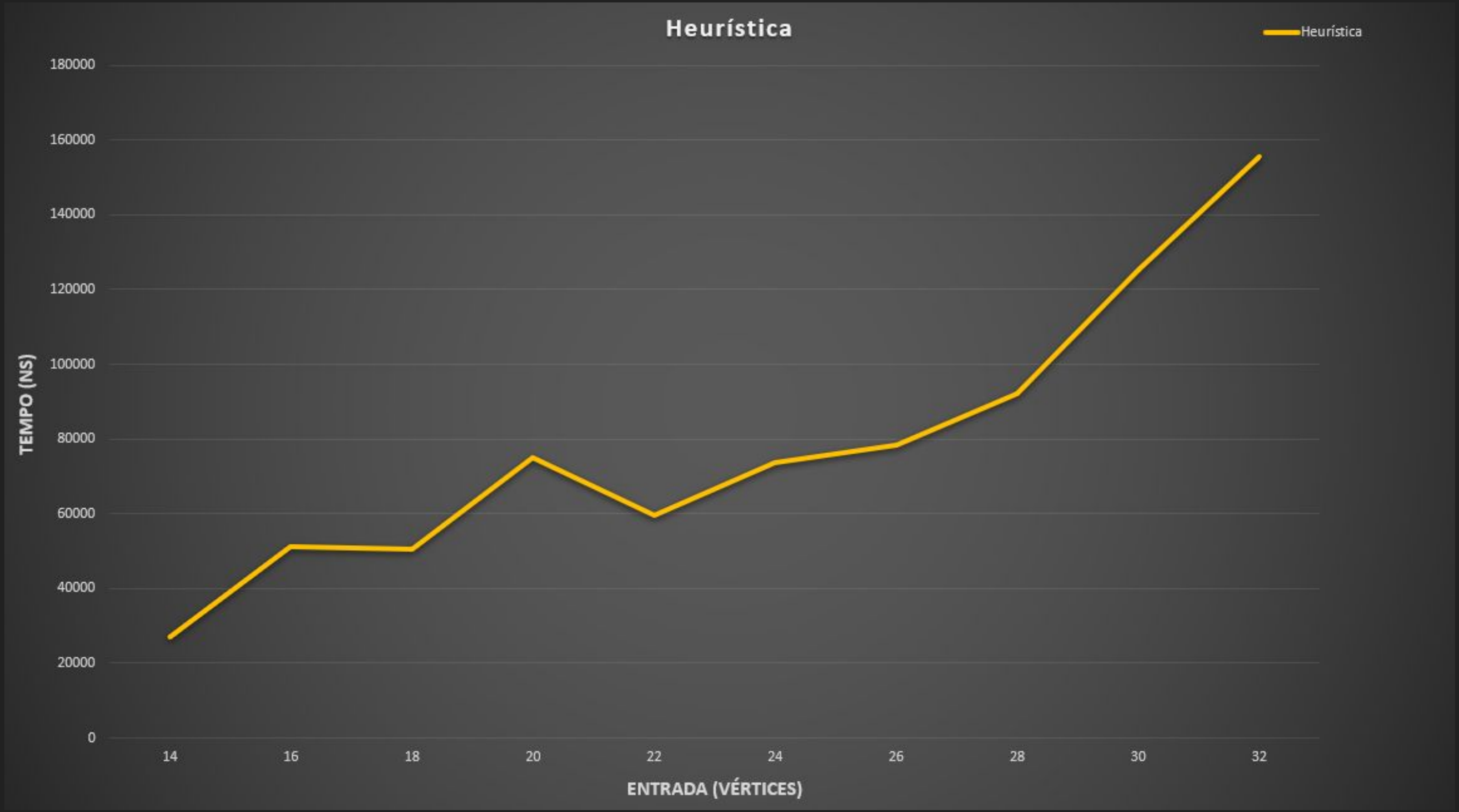
Especificações dos casos de teste

- Site: Geração de Grafos
- Grafos não direcionados e sem pesos nas arestas
- Conectividade aproximada de 50%
    - n(n-1) / 4

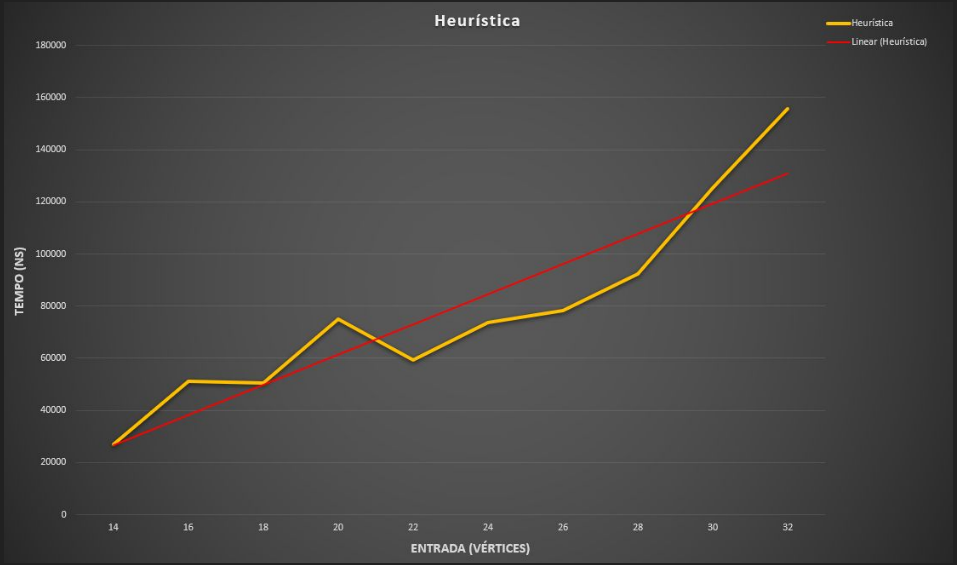
Análise Experimental dos Algoritmos

Heurística - Kernighan-lin							
Entrada	Nº Vértices	R. Heurística	1º Tentativa	2º Tentativa	3º Tentativa	4º Tentativa	Média
in.txt	14	21	25500 ns	26700 ns	28600 ns	27700 ns	27125 ns
in2.txt	16	25	43900 ns	50800 ns	52500 ns	57700 ns	51225 ns
in3.txt	18	28	46900 ns	64400 ns	46000 ns	44200 ns	50375 ns
in4.txt	20	34	64100 ns	77700 ns	93300 ns	65500 ns	75150 ns
in5.txt	22	34	56900 ns	58500 ns	59800 ns	62700 ns	59475 ns
in6.txt	24	38	67200 ns	68300 ns	85200 ns	73600 ns	73575 ns
in7.txt	26	60	84100 ns	77300 ns	74900 ns	77000 ns	78325 ns
in8.txt	28	60	91700 ns	97100 ns	91300 ns	89100 ns	92300 ns
in9.txt	30	80	113000 ns	134100 ns	151500 ns	103200 ns	123025 ns
in10.txt	32	96	180900 ns	196600 ns	125900 ns	119100 ns	155625 ns

Análise Experimental dos Algoritmos



Análise Experimental dos Algoritmos



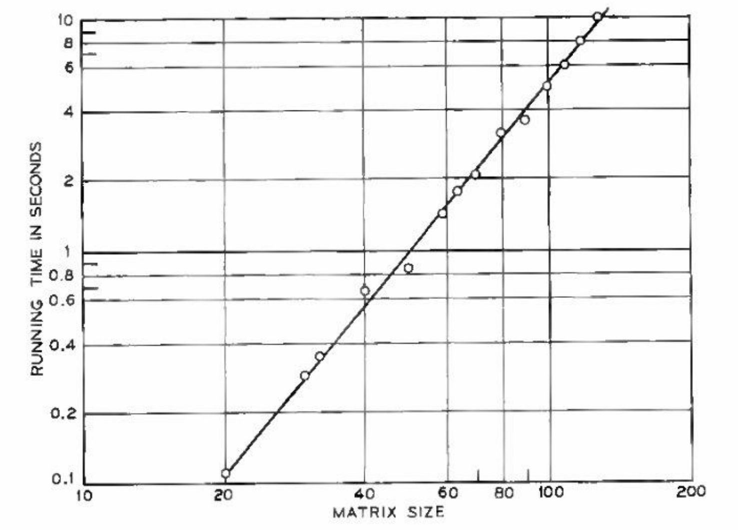
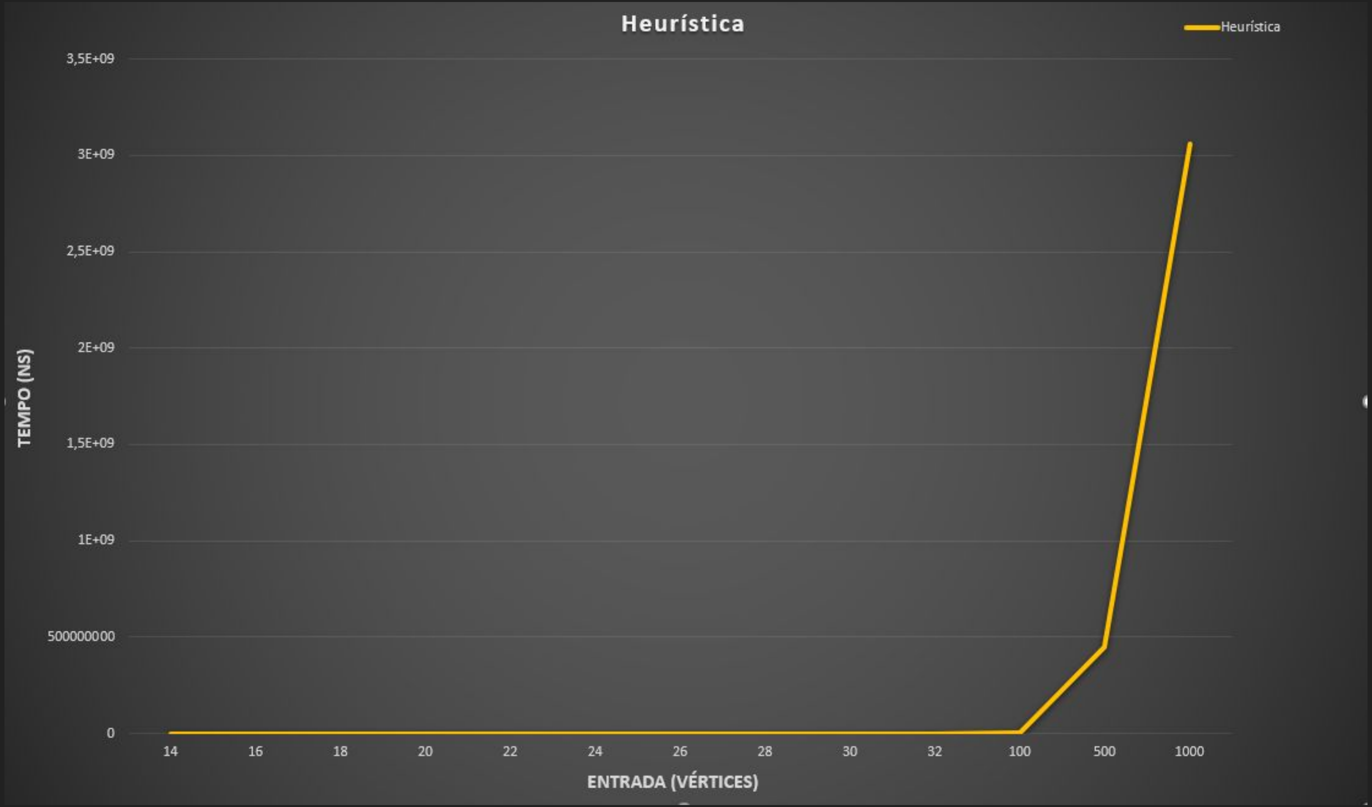


Fig. 2 — Running time.

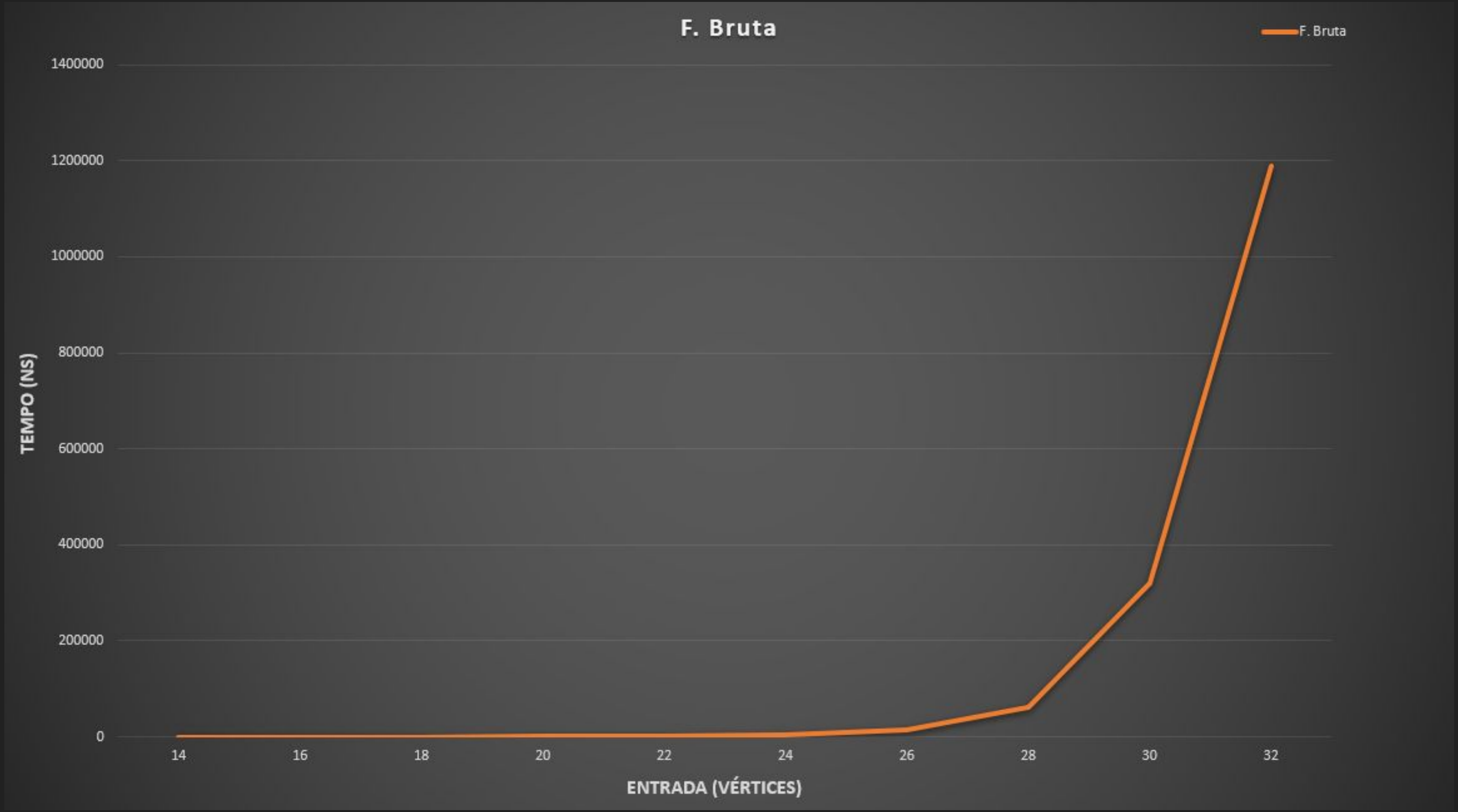
Análise Experimental dos Algoritmos



Análise Experimental dos Algoritmos

Força Bruta							
Entrada	Nº Vértices	R. Bruta	1º Tentativa	2º Tentativa	3º Tentativa	4º Tentativa	Média
in.txt	14	19	2 ms	2 ms	2 ms	2 ms	2 ms
in2.txt	16	23	11 ms	9 ms	9 ms	11 ms	10 ms
in3.txt	18	28	46 ms	41 ms	41 ms	42 ms	42 ms
in4.txt	20	31	171 ms	177 ms	177 ms	171 ms	174 ms
in5.txt	22	34	728 ms	722 ms	722 ms	742 ms	728 ms
in6.txt	24	37	3 s	3 s	3 s	3 s	3 s
in7.txt	26	55	14 s	14 s	14 s	14 s	14 s
in8.txt	28	60	60 s	62 s	62 s	63 s	62 s
in9.txt	30	78	5 min	5 min	5 min	5 min	5 min
in10.txt	32	91	22 min	19 min	19 min	18 min	19 min

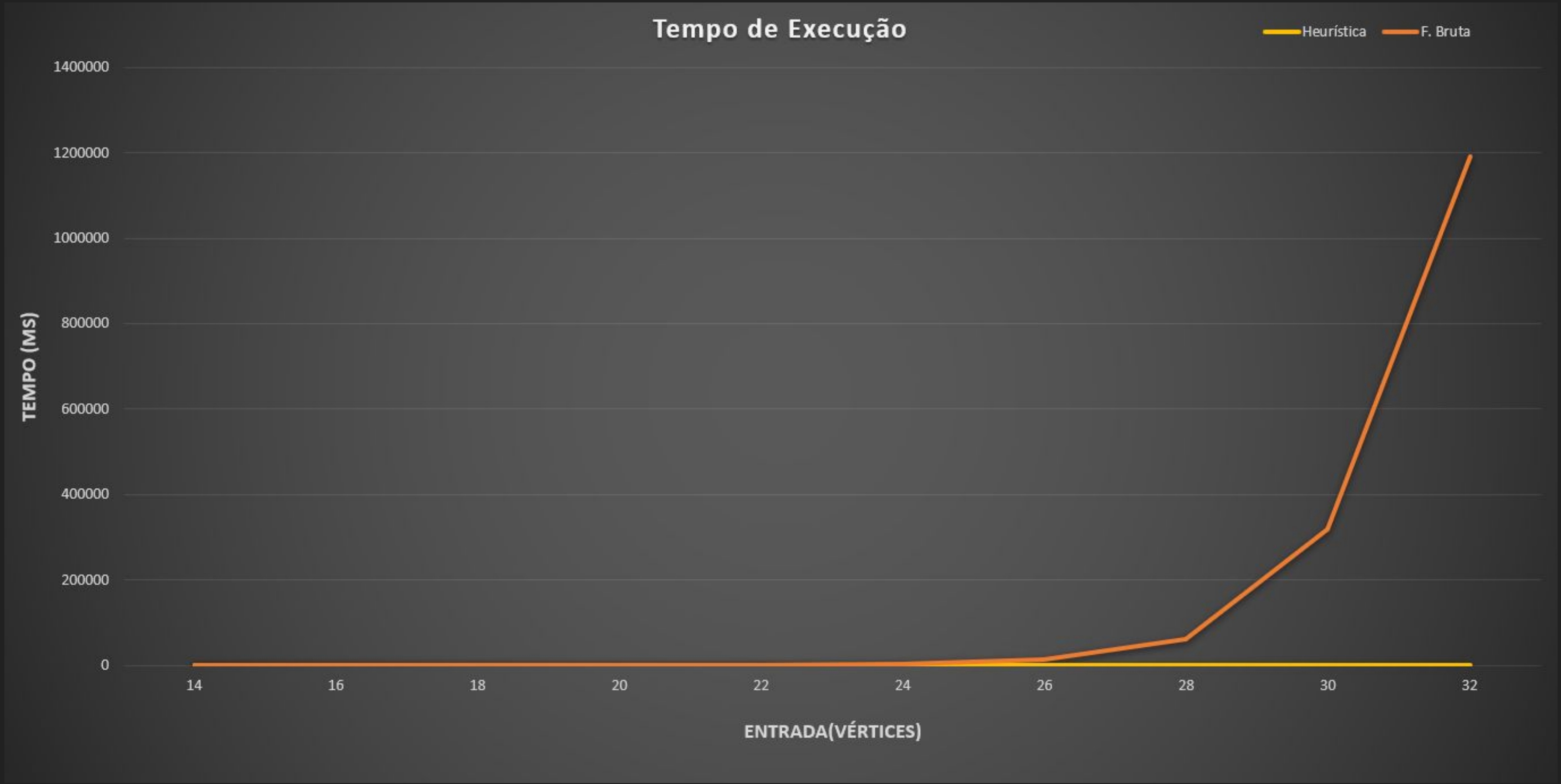
Análise Experimental dos Algoritmos



Análise Experimental dos Algoritmos

Entrada	Heurística	F. Bruta
14	0.027125 ms	2 ms
16	0.051225 ms	10 ms
18	0.050375 ms	42 ms
20	0.07515 ms	174 ms
22	0.059475 ms	728 ms
24	0.073575 ms	3129 ms
26	0.078325 ms	14270 ms
28	0.0923 ms	62261 ms
30	0.12545 ms	319023 ms
32	0.155625 ms	1190354 ms

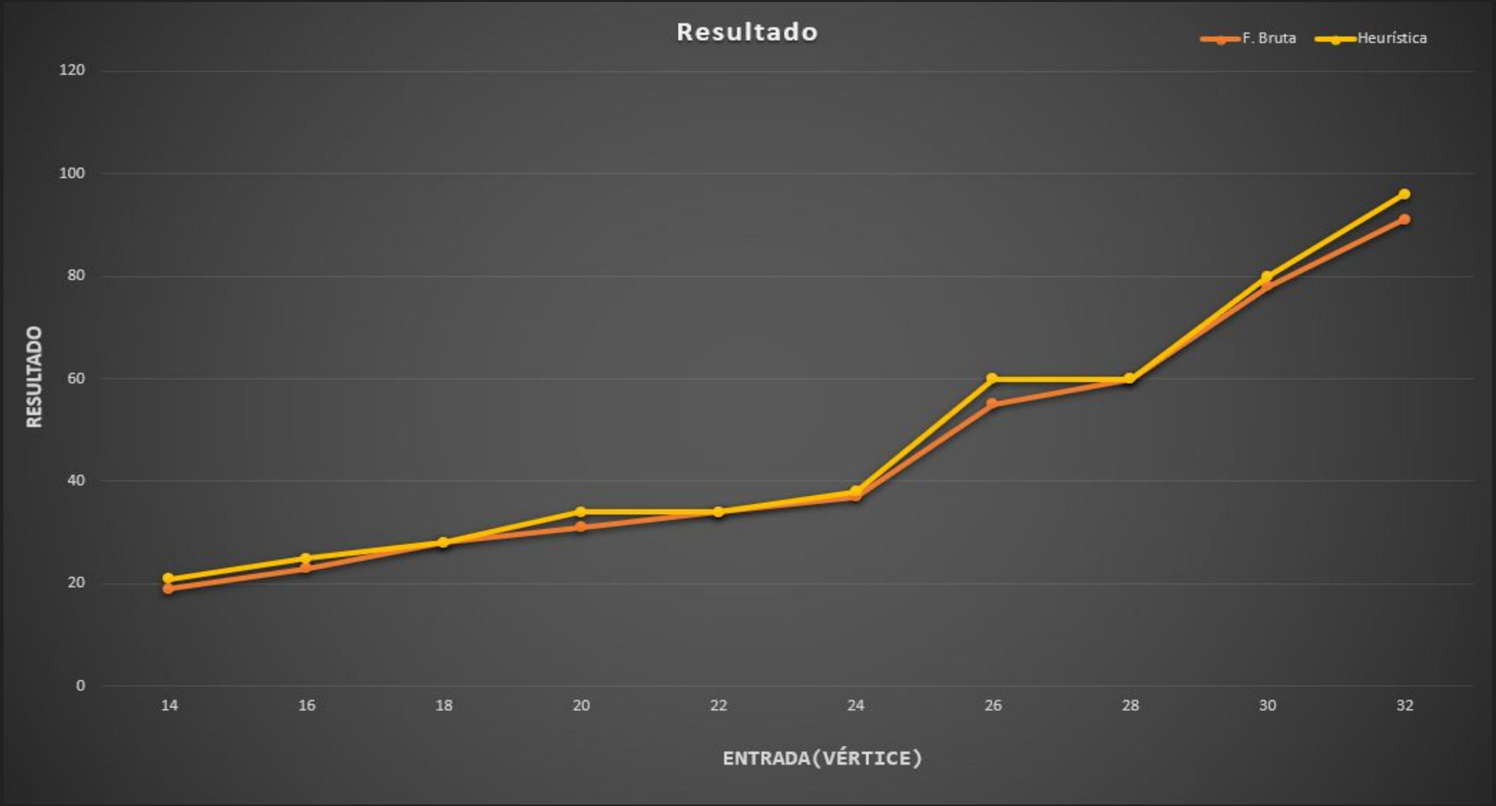
Análise Experimental dos Algoritmos



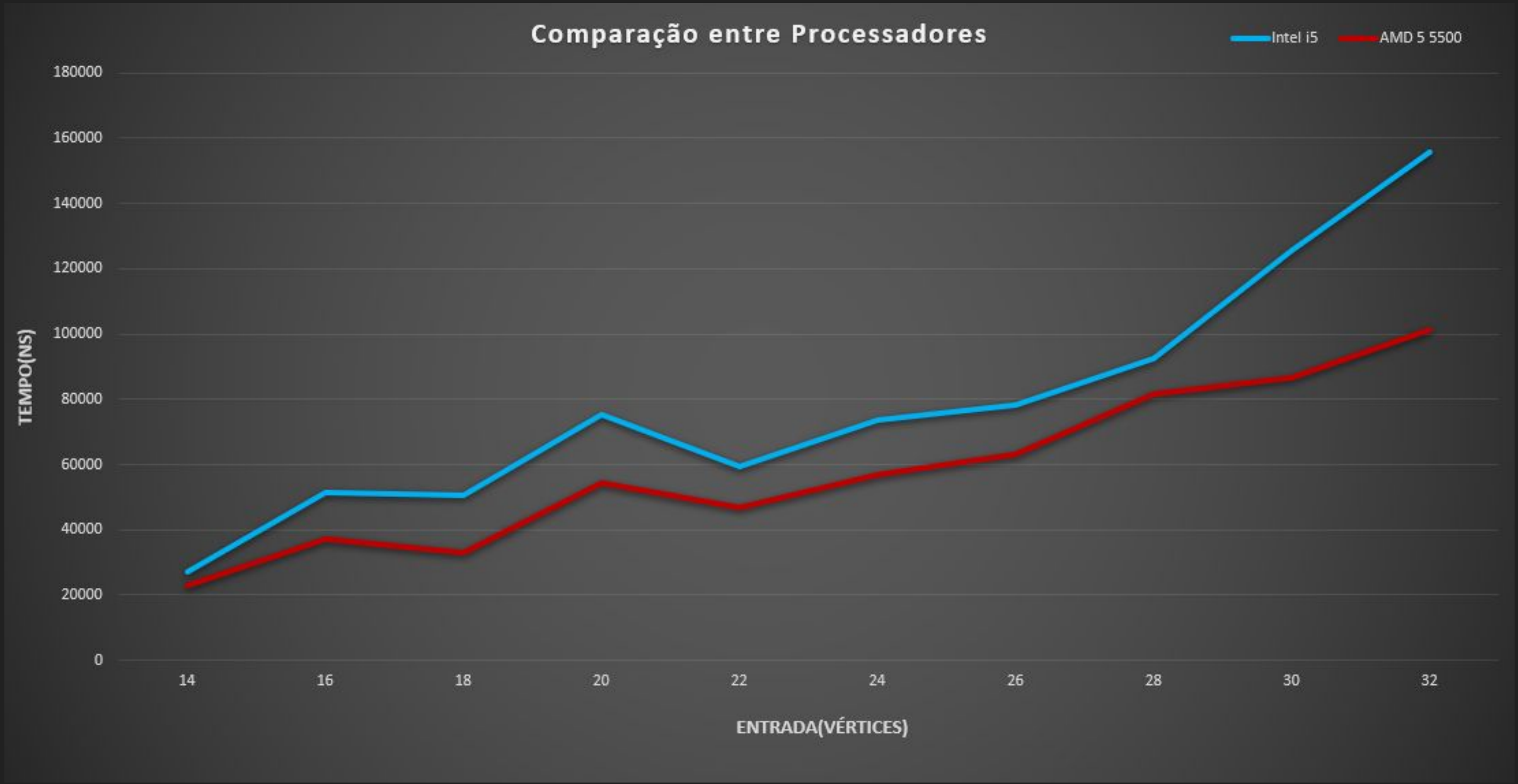
Análise Experimental dos Algoritmos

Entrada	F. Bruta	Heurística	F. Aproximação
14	19	21	1,105263158
16	23	25	1,086956522
18	28	28	1
20	31	34	1,096774194
22	34	34	1
24	37	38	1,027027027
26	55	60	1,090909091
28	60	60	1
30	78	80	1,025641026
32	91	96	1,054945055

Análise Experimental dos Algoritmos



Análise Experimental dos Algoritmos



Referências

- KERNIGHAN, B. W.; LIN, S. An Efficient Heuristic Procedure for Partitioning Graphs. Bell System Technical Journal, v. 49, n. 2, p. 291–307, fev. 1970.

- Kang, Andrew. et al. VLSI Pjysical Design: From Graph Partitioning to Timing Closure. Nova York: Springer, 2011.

Obrigado pela Atenção!