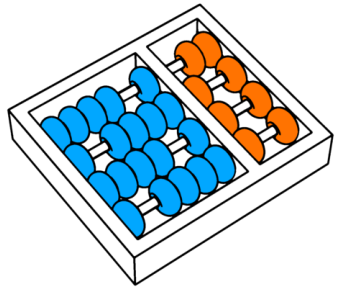
Partição de Grafos Eulerianos em Circuitos

Pedro Olímpio N. de O. Pinheiro



Orientador: Prof. Dr. Zanoni Dias
Coorientador: Prof. Dr. Cid C. de Souza

Sumário

1. Introdução
2. Fundamentação Teórica
3. Máxima Partição em Circuitos
4. Máxima Partição em Circuitos Alternados
5. Considerações Finais

Introdução

Introdução

- A partição de um conjunto consiste em dividir esse conjunto em subconjuntos disjuntos, em que cada elemento do conjunto original esteja em exatamente um subconjunto
- Nos problemas abordados neste trabalho, o objetivo é particionar o conjunto de arestas de um grafo, de forma que cada subconjunto induza um circuito no grafo original
- Essa partição só é possível em grafos eulerianos
- Queremos encontrar a partição de maior cardinalidade
- Chamamos esse problema de Máxima Partição em Circuitos (*Maximum Eulerian Cycle Decomposition* - `MAX-ECD`)

Introdução

- Também abordamos o problema da Máxima Partição em Circuitos Alternados (*Maximum Alternating-Cycle Decomposition* - `MAX-ACD`)
- Nesse problema, as arestas possuem cor e os circuitos devem possuir arestas com cores alternadas
- Ambos, `MAX-ECD` e `MAX-ACD`, são NP-difícil [2]
- Chen [4] apresentou algoritmos para o problema da Ordenação por Rearranjos que tem o `MAX-ACD` como subproblema
- Algoritmos de Ordenação por Rearranjos são frequentemente usados no estudo da distância evolucionária entre espécies

Fundamentação Teórica

Grafos

- $G = (V, E)$
- Subgrafo
- Trilha e Caminho
- Circuito e Ciclo
- Grafo euleriano
- Partição em circuitos

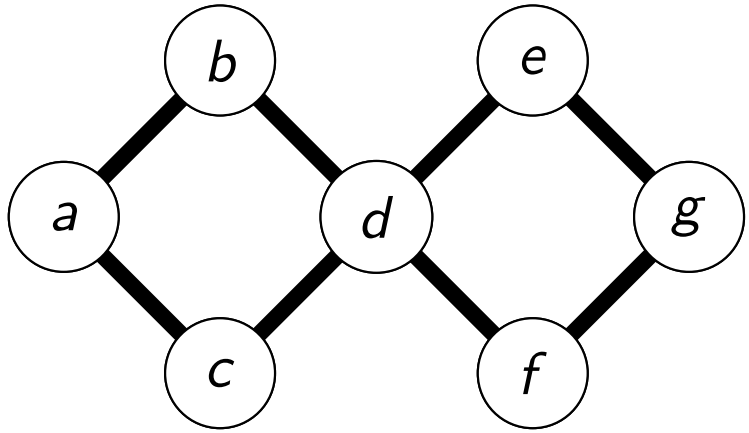


Figura: Exemplo de um grafo com 7 vértices e 8 arestas.

Grafos Bicoloridos

- $G = (V, P \cup C)$
- Circuito alternado

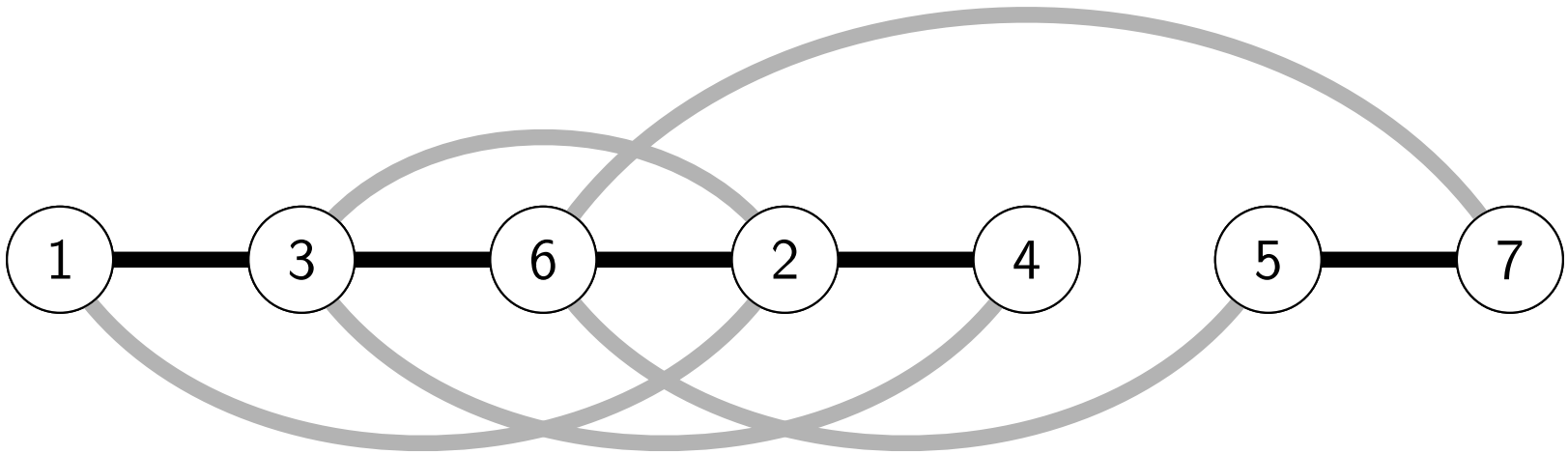


Figura: Exemplo de um grafo bicolorido.

Máxima Partição em Circuitos (MAX-ECD)

Máxima Partição em Circuitos

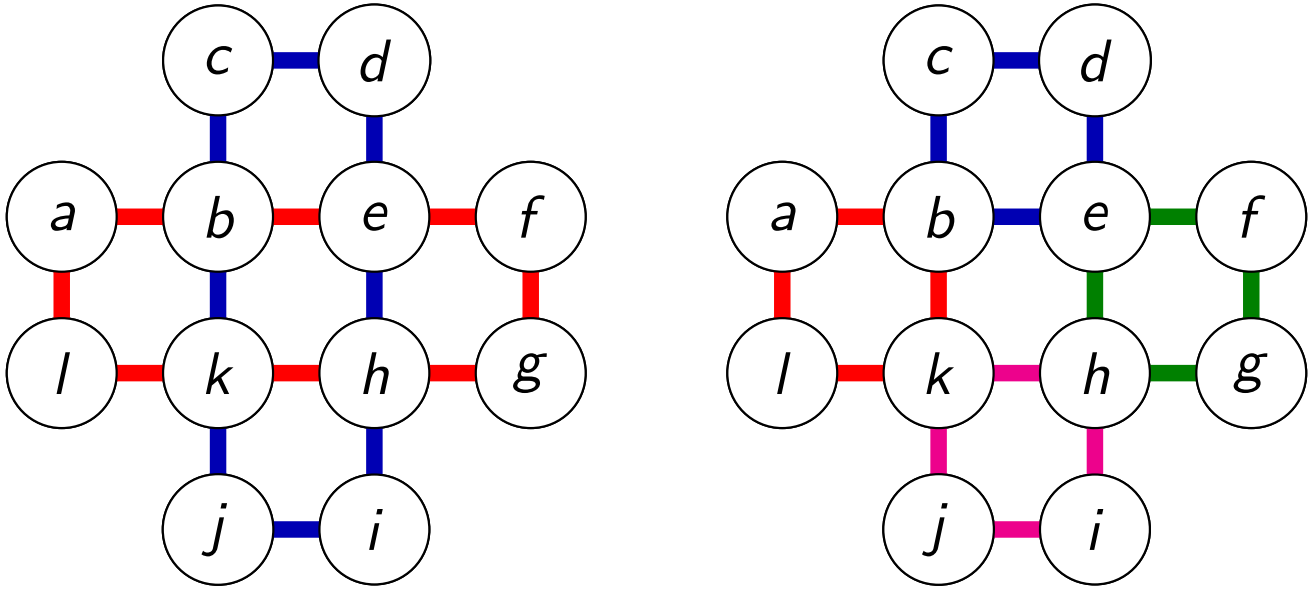


Figura: Exemplos de partições em circuitos de um grafo.

Máxima Partição em Circuitos

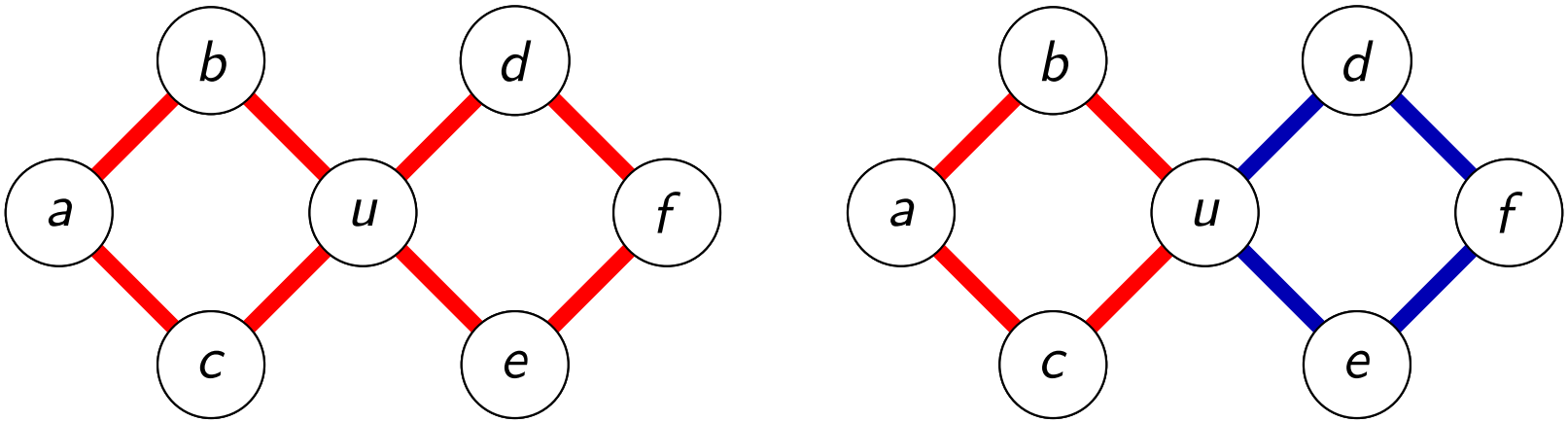


Figura: Exemplo de melhora na solução do `MAX-ECD` substituindo um circuito com repetição de vértices em (a) por dois ciclos em (b).

Modelo PLI

- **Primal** [3]:

$$\max \sum_{C \in \mathcal{C}} x_C \tag{1}$$

sujeito a:

$$\sum_{C \in \mathcal{C}_e} x_C \leq 1, \qquad \forall e \in E \tag{2}$$

$$x_C \in \{0, 1\}, \qquad \forall C \in \mathcal{C} \tag{3}$$

- Geração de Colunas e *Branch & Price* (*B&P*)

Modelo PLI

- **Dual**:

$$\min \sum_{e \in E} y_e \tag{1}$$

sujeito a:

$$\sum_{e \in C} y_e \geq 1, \qquad \forall C \in \mathcal{C} \tag{2}$$

$$y_e \geq 0, \qquad \forall e \in E \tag{3}$$

- Encontrar o ciclo mínimo

Modelo PLI

- *Branch & Bound* (*B&B*)
  - Variáveis com valores fixos
- Variáveis com valor 1
- Variáveis com valor 0
  - Variáveis $x_{C_1}, x_{C_2}, \ldots, x_{C_k}$ com valor 0
  - Criamos cópias $G_{z_1,z_2,\ldots,z_k}$
    - $G_{1,1,\ldots,1}$
    - $G_{2,1,\ldots,1}$
    - $G_{3,1,\ldots,1}$
    - $G_{1,2,\ldots,1}$
    - $G_{2,2,\ldots,1}$
    - $G_{|C_1|,|C_2|,\ldots,|C_k|}$

Heurística Gulosa

Algoritmo 1 Heurística gulosa para o `MAX-ECD`

```
1:  função HEURÍSTICA-GULOSA(G = (V, E))
2:      H ← ∅
3:      enquanto |E| > 0 faça
4:          Escolha um vértice v
5:          C ← Ciclo-Mínimo(G, v)
6:          H ← H ∪ {C}
7:          E ← E \ {C}
8:      fim enquanto
9:      devolve H
10: fim função
```

Heurística Gulosa

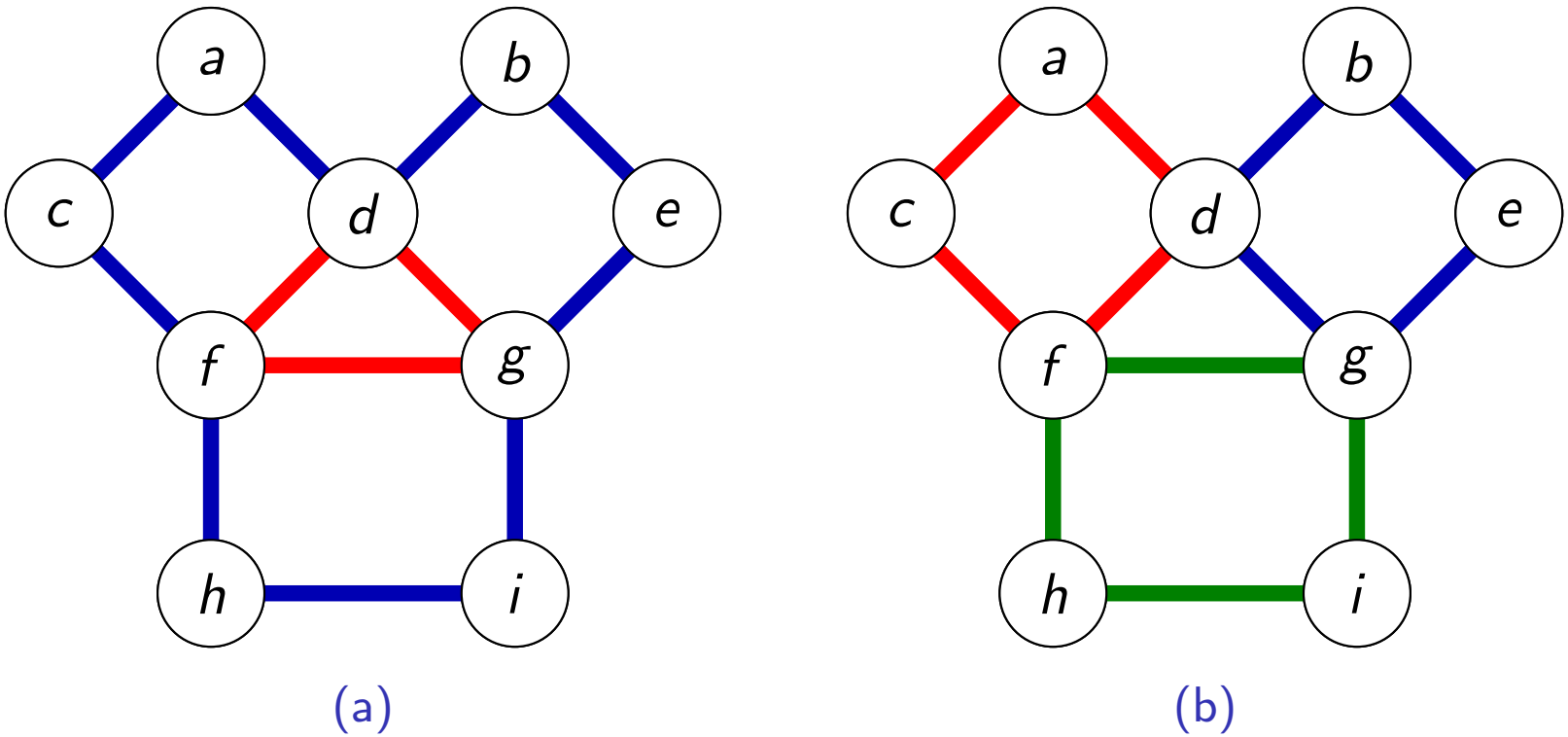


Figura: Soluções distintas da heurística gulosa para o mesmo grafo alterando o vértice escolhido em cada passo (linha 4).

Heurística PLI

- $\mathcal{C}' \subset \mathcal{C}$

$$\max \sum_{C \in \mathcal{C}'} x_C \tag{1}$$

sujeito a:

$$\sum_{C \in \mathcal{C}'_e} x_C \leq 1, \qquad \forall e \in E \tag{2}$$

$$x_C \in \{0, 1\}, \qquad \forall C \in \mathcal{C}' \tag{3}$$

Heurística PLI

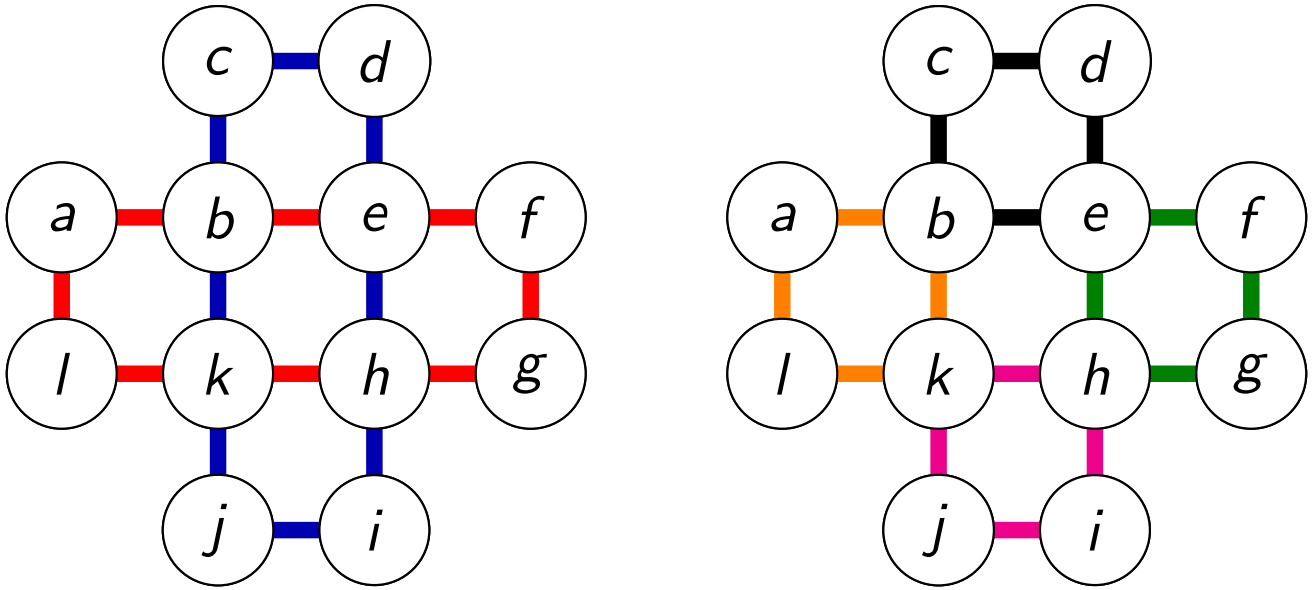


Figura: Ciclos de um conjunto $\mathcal{C}'$ da heurística PLI em que a solução ótima não é uma partição.

Ambiente Computacional

- Para cada $n \in \{10, 20, \ldots, 90, 100\}$ e $d \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$, criamos um conjunto contendo 20 grafos
  - Algoritmo de Hakimi [6]
- Processador Intel Core i7-8700T CPU com 12 núcleos de 2.40 GHz
- Memória RAM de 8 GB
- Sistema operacional Ubuntu 18.04.3
- Tempo limite de 1800 segundos
- Resolvedor `PL`: Gurobi 8.1.1 [5]
- Resolvedor `PLI` (*B&P*): SCIP 6.0.2 [1]
- Resolvedor `PLI`: Gurobi 8.1.1

Resultados dos Experimentos

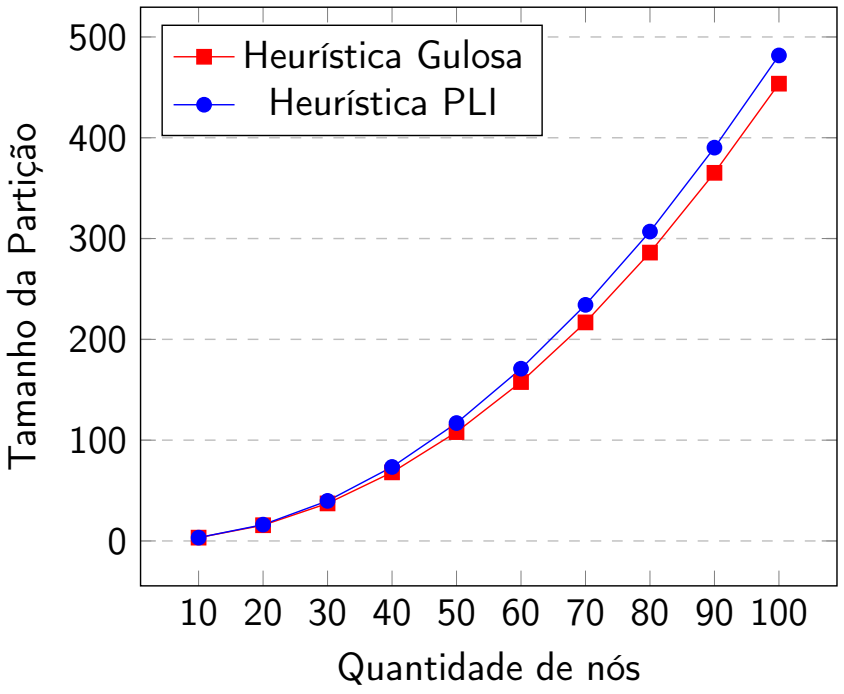
n \ d	10%	20%	30%	40%	50%
10	100	100	100	100	100
20	100	100	100	85	100
30	100	95	80	80	60
40	95	65	50	50	35
50	80	20	20	10	0
60	40	5	0	0	0
70	15	0	0	0	0
80	0	0	0	0	0
90	0	0	0	0	0
100	0	0	0	0	0

Tabela: Porcentagem de instâncias resolvidas usando `PLI` com *B&P*.

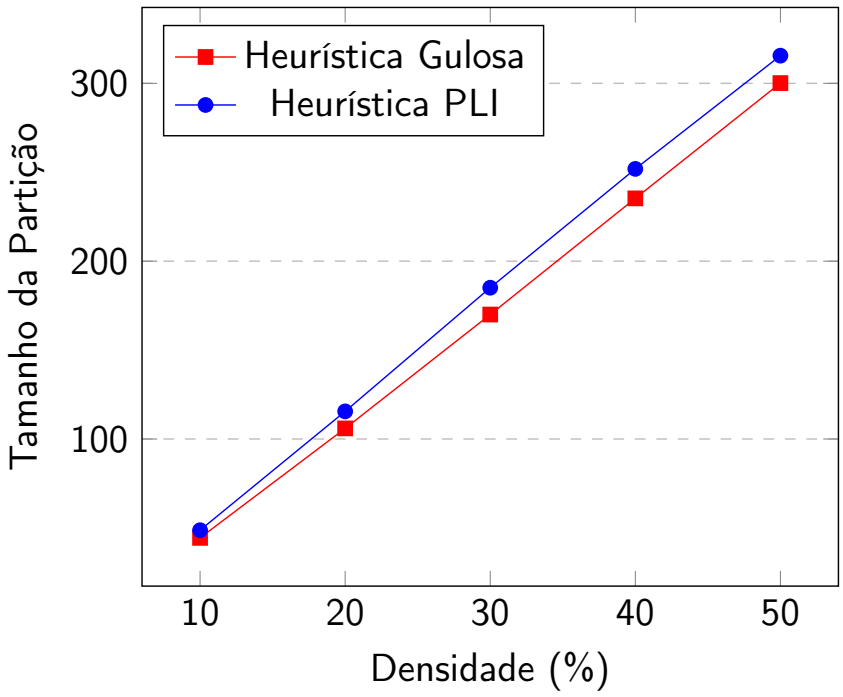
Resultados dos Experimentos

- Muitas instâncias com $n \geq 80$ sem solução no nó raiz da árvore de *B&B*
- Nessas instâncias, 74% do tempo foi gasto no *pricing*
- Para reduzir o tempo gasto no *pricing*, iniciamos o processamento com um conjunto não nulo de variáveis
  - Usamos todos os ciclos de 100 execuções da heurística gulosa
- Para diminuir o tamanho da árvore de *B&B*, visando encontrar mais soluções ótimas, buscamos limitantes primais com as heurísticas

Resultados dos Experimentos



(a)



(b)

Figura: Crescimento dos limitantes primais das heurísticas em relação à (a) quantidade de nós e à (b) densidade da instância.

Resultados dos Experimentos

	Guloso		PLI		PLI melhorado	
n	Média	Máximo	Primal	Ótimo	Primal	Ótimo
10	3.15	3.29	3.29	100%	3.29	100%
20	14.48	15.60	16.17	97%	16.18	100%
30	35.20	37.23	39.63	83%	39.79	100%
40	65.06	68.03	73.00	59%	73.43	95%
50	104.03	108.01	43.26	26%	116.99	91%
60	152.78	157.66	29.85	9%	170.89	82%
70	210.71	216.81	12.25	3%	234.23	53%
80	278.44	286.08	16.39	0%	306.92	41%
90	355.58	365.21	-	-	390.19	26%
100	442.37	453.65	-	-	481.67	9%

Tabela: Número de ciclos nas soluções obtidas do MAX-ECD.

Resultados dos Experimentos

n	Guloso	H_PLI	PLI	PLI 2
10	0.01	0.00	0.00	0.00
20	0.15	0.02	54.95	0.77
30	0.99	0.25	329.72	18.28
40	4.00	2.33	818.97	118.19
50	11.88	13.20	1394.53	173.40
60	27.92	184.92	1653.82	374.16
70	57.97	539.62	1765.97	852.39
80	110.86	1095.82	1800.00	1066.75
90	199.70	1355.89	1800.00	1339.50
100	348.64	1740.37	1800.00	1641.55

Tabela: Tempo de execução para obter as soluções do MAX-ECD.

Resultados dos Experimentos

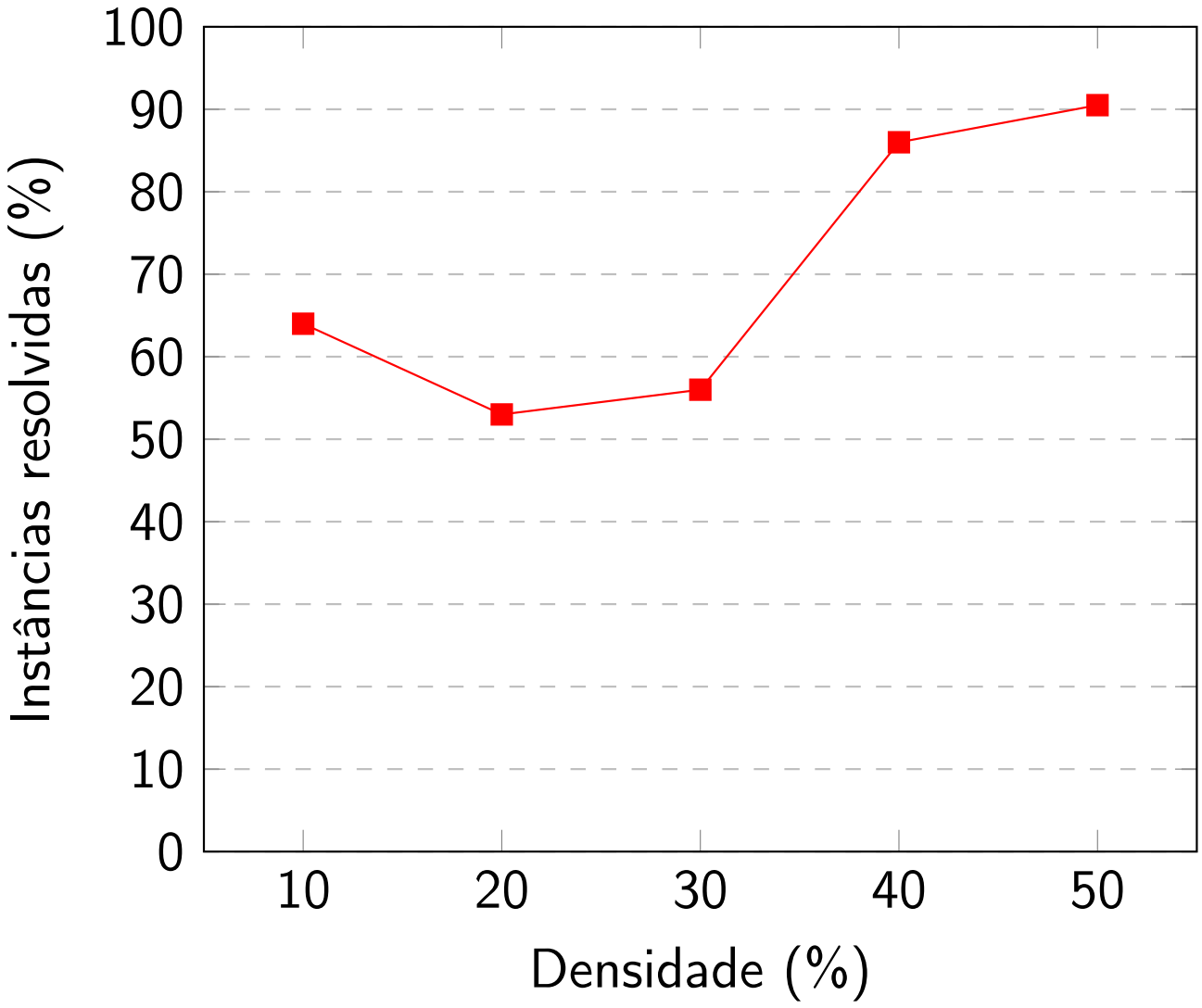


Figura: Percentual de instâncias resolvidas por densidade.

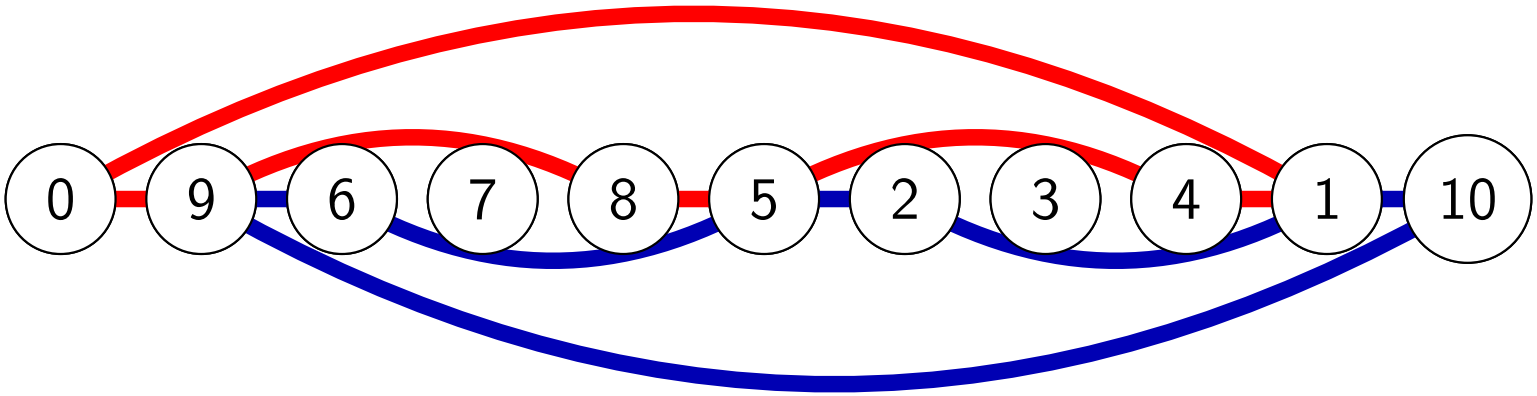
Resultados dos Experimentos

n \ d	10%	20%	30%	40%	50%
10	0.00	0.00	31.67	59.41	67.19
20	0.00	36.81	63.30	81.43	91.87
30	24.34	53.46	75.13	89.19	97.63
40	27.89	60.65	83.40	94.27	99.31
50	34.09	67.29	86.40	98.18	99.93
60	37.41	72.42	92.56	99.41	100.00
70	38.54	75.57	94.89	99.77	99.75
80	42.74	77.85	95.34	99.30	99.33
90	46.10	81.34	97.37	99.45	99.45
100	48.53	83.44	98.45	99.45	99.50

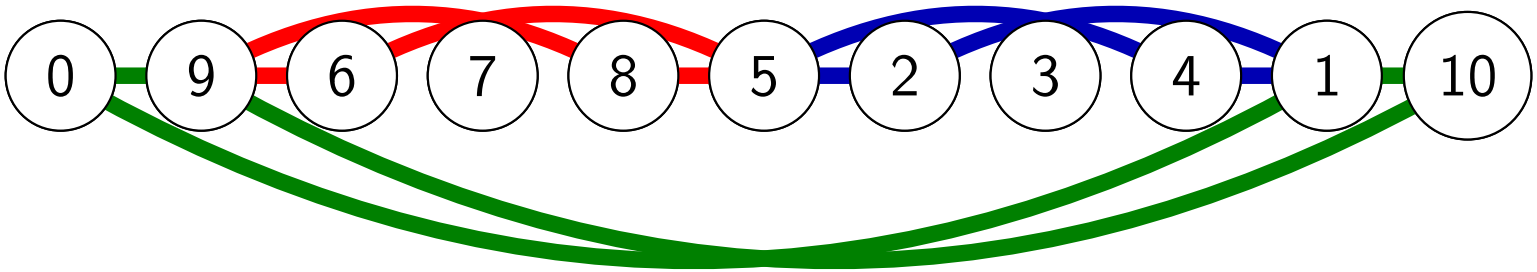
Tabela: Proporção de triângulos das soluções do `MAX-ECD` retornadas pela heurística PLI.

Máxima Partição em Circuitos Alternados (MAX-ACD)

Máxima Partição em Circuitos Alternados



(a)



(b)

Figura: Exemplos de partições em circuitos alternados de um grafo bicolorido.

Máxima Partição em Circuitos Alternados

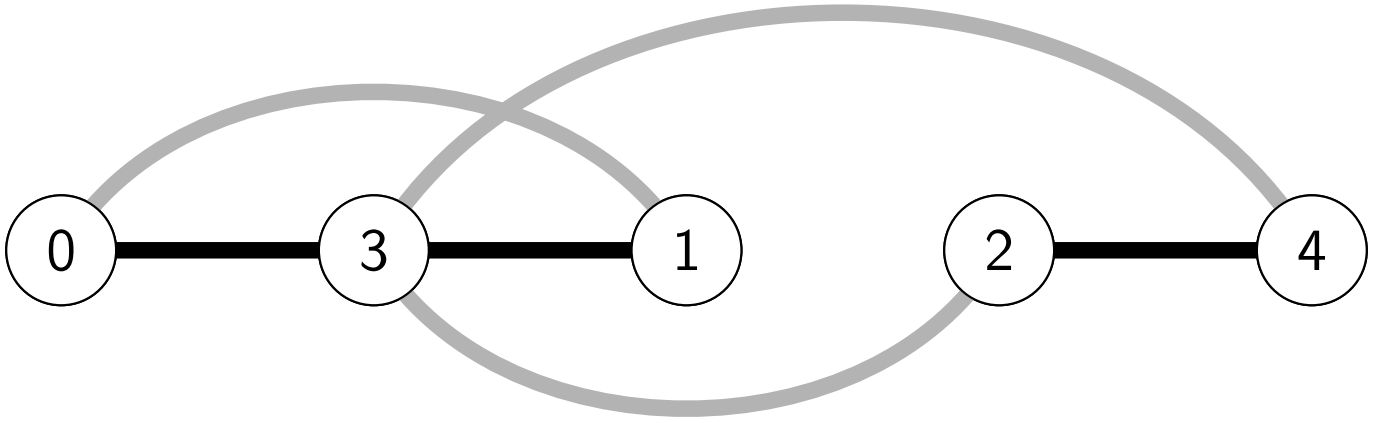


Figura: Exemplo de um grafo bicolorido que exige um circuito para particionar as arestas.

Grafos de *Breakpoint*

- Grafos de *breakpoint* são construídos com base em uma permutação π

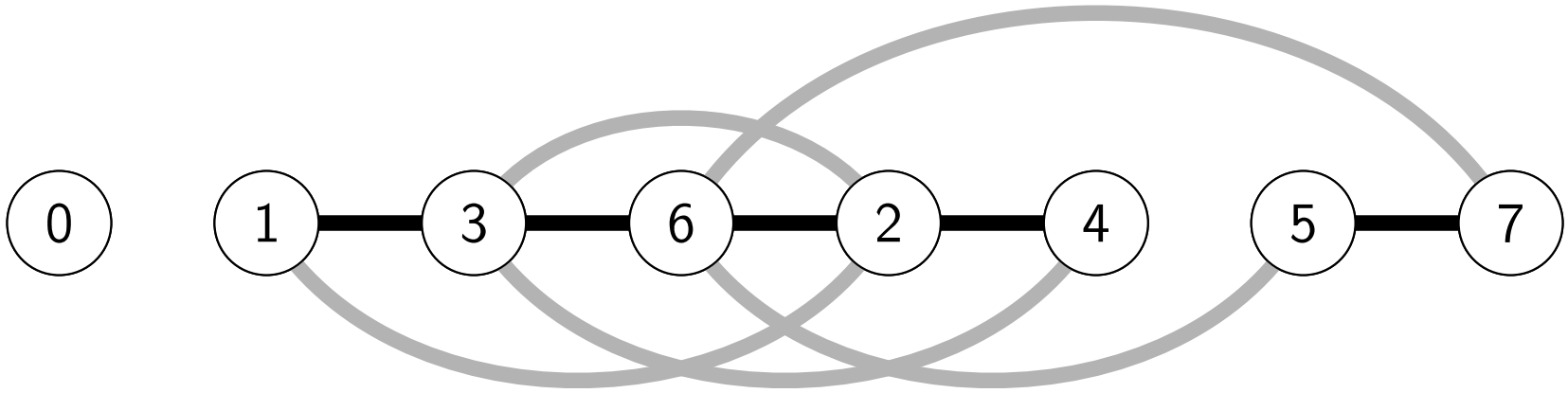


Figura: Exemplo de grafo de *breakpoint* para a permutação $\pi = (1, 3, 6, 2, 4, 5)$.

- Usando `MAX-ACD` em grafos de *breakpoint*, Lin e Jiang [7] desenvolveram um algoritmo para o problema da Ordenação por Reversão (*Minimum Sorting by Reversals* - `MIN-SBR`)

Heurística Gulosa

Algoritmo 2 Heurística gulosa para o `MAX-ACD`

1: **função** HEURÍSTICA-GULOSA($G = (V, E)$)
2: $\quad \mathcal{H} \leftarrow \emptyset$
3: $\quad$ **enquanto** $|E| > 0$ **faça**
4: $\quad\quad$ Escolha um vértice v
5: $\quad\quad$ $C \leftarrow$ `Circuito-Alternado-Mínimo`(G, v)
6: $\quad\quad$ $\mathcal{H} \leftarrow \mathcal{H} \cup \{C\}$
7: $\quad\quad$ $E \leftarrow E \setminus C$
8: $\quad$ **fim enquanto**
9: $\quad$ **devolve** $\mathcal{H}$
10: **fim função**

Heurística Gulosa

- A escolha do vértice na Linha 4 do Algoritmo 2 influencia diretamente no resultado final
- Propomos 4 variações
  - `FIRST`
  - `RANDOM`
  - `MAX`
  - `ALL`
- Fizemos ainda uma alteração no algoritmo para ter o mesmo fator de aproximação do algoritmo de Lin e Jiang para o `MIN-SBR`

Heurística Gulosa

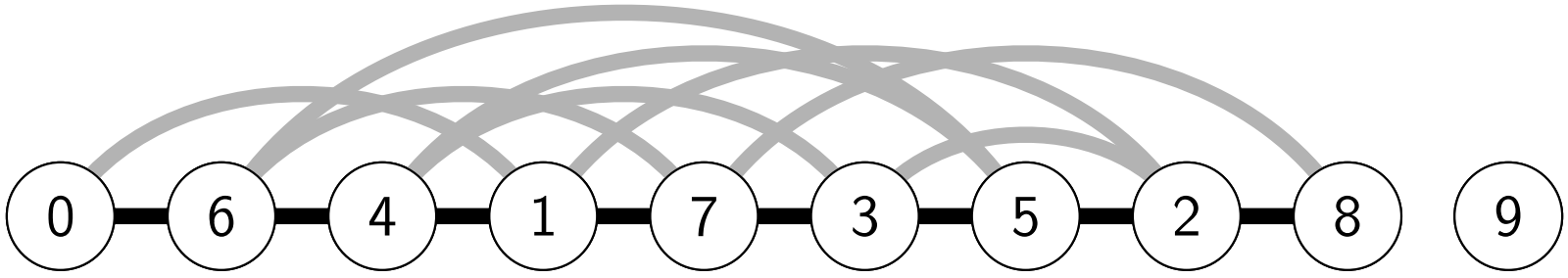


Figura: Exemplo um grafo bicolorido.

Heurística Gulosa

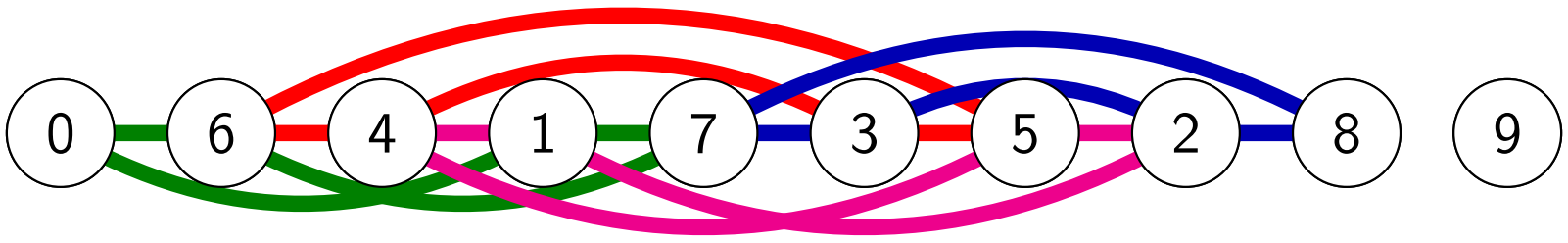


Figura: Solução com as variações `FIRST`, `MAX` e `ALL`.

Heurística Gulosa

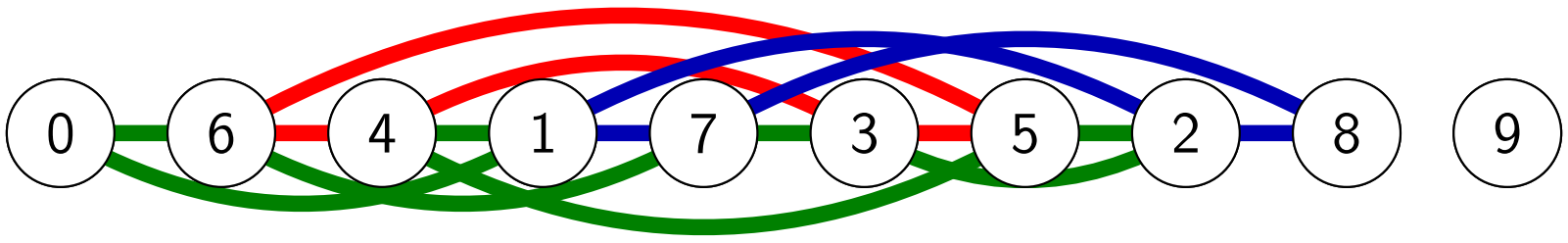


Figura: Solução com a variação RANDOM (vértices escolhidos: 6, 1 e 0).

Busca Tabu

- Solução inicial dada pela heurística gulosa
- Dois movimentos
  - O primeiro melhora a solução
  - O segundo permite a busca continuar após encontrar um ótimo local

Busca Tabu

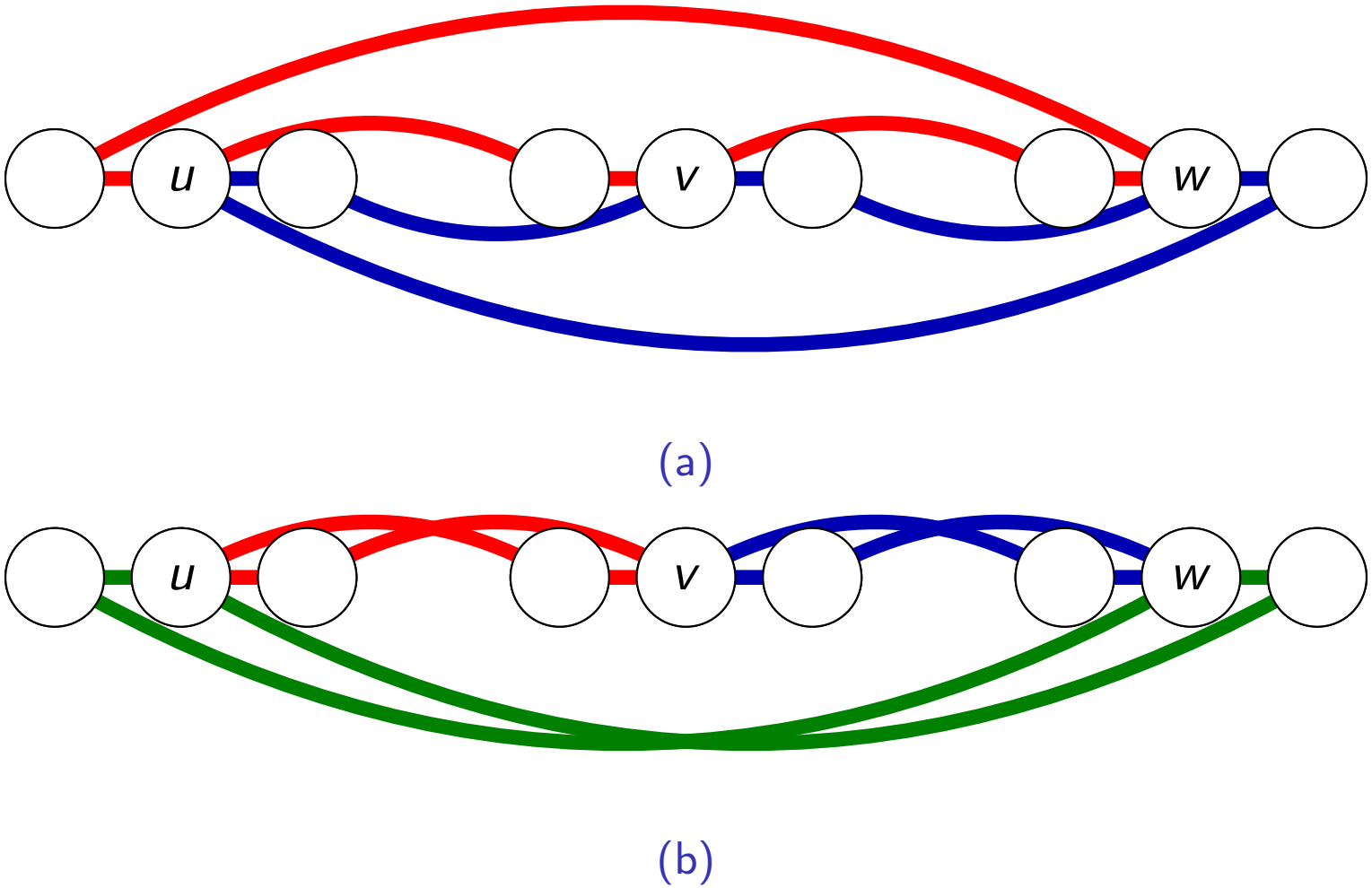
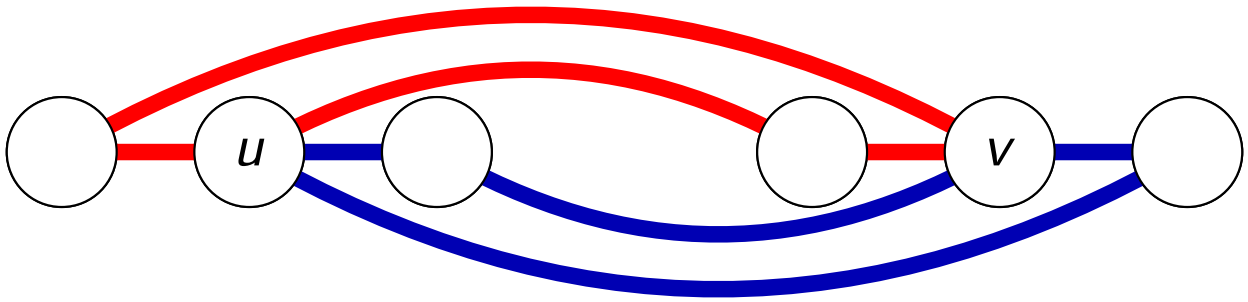
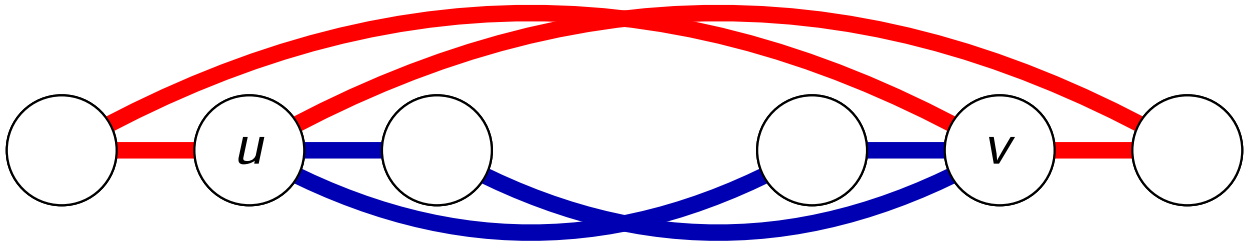


Figura: Exemplo de aplicação do primeiro movimento da Busca Tabu.



(a)



(b)

Figura: Exemplo de aplicação do segundo movimento da Busca Tabu.

Modelo PLI

- Modelo semelhante ao do `MAX-ECD`

$$\max \sum_{C \in \mathcal{C}^0} x_C \tag{1}$$

sujeito a:

$$\sum_{C \in \mathcal{C}_e^0} x_C \leq 1, \qquad \forall e \in E \tag{2}$$

$$x_C \in \{0, 1\}, \qquad \forall C \in \mathcal{C} \tag{3}$$

- Pequena alteração no algoritmo de *pricing*

Experimentos Computacionais

- As instâncias são grafos de *breakpoint* baseados em permutações aleatórios com o número máximo de *breakpoints*
  - 100 instâncias para cada valor em $\{10, 20, 30, \ldots, 100\}$
- Mesmo ambiente computacional

Resultados dos Experimentos

- Comparamos o algoritmo de Lin e Jiang (L&J) para o `MAX-ACD` com as heurísticas gulosas e Busca Tabu propostas
- Executamos os experimentos em quatro cenários:
  1. Comparando as variações do algoritmo guloso com o algoritmo de L&J
  2. Indicando ao algoritmo guloso que deve conter os circuitos curtos que estão na solução do algoritmo de L&J
  3. Executando Busca Tabu nos resultados obtidos do experimento 1
  4. Executando Busca Tabu nos resultados obtidos do experimento 2

Resultados dos Experimentos

n	Experimento 1					Experimento 2			
	L&J	FIRST	ALL	RANDOM	MAX	FIRST	ALL	RANDOM	MAX
10	**4.14**	3.99	4.09	3.98	**4.14**	**4.14**	**4.14**	**4.14**	**4.14**
20	6.58	6.23	6.63	6.35	**6.89**	6.75	6.75	6.75	6.75
30	8.01	8.04	8.75	8.27	**9.22**	8.89	8.90	8.88	8.91
40	9.12	9.98	10.70	10.06	**11.39**	10.81	10.89	10.84	10.97
50	9.70	11.39	12.75	11.67	**13.45**	12.61	12.74	12.62	12.97
60	10.20	13.06	14.38	13.20	**15.38**	14.25	14.47	14.27	14.88
70	10.97	14.45	16.36	14.73	**17.13**	16.09	16.33	16.02	16.84
80	11.52	15.69	18.16	16.30	**19.01**	17.77	18.24	17.77	18.95
90	11.73	17.27	19.51	17.60	20.44	19.14	19.63	19.13	**20.72**
100	12.31	18.35	21.35	19.03	22.17	20.67	21.27	20.71	**22.54**

Tabela: Número médio de circuitos nas partições retornadas.

Resultados dos Experimentos

n	Experimento 3					Experimento 4			
	L&J	FIRST	ALL	RANDOM	MAX	FIRST	ALL	RANDOM	MAX
10	**4.14**	4.09	4.12	**4.14**	**4.14**	**4.14**	**4.14**	**4.14**	**4.14**
20	6.76	6.50	6.80	**6.89**	**6.89**	6.77	6.77	6.81	6.81
30	8.87	8.54	9.00	**9.22**	**9.22**	9.03	9.02	9.13	9.13
40	10.85	10.50	11.12	11.39	**11.40**	11.08	11.14	11.33	11.33
50	12.55	12.28	13.06	13.45	**13.58**	12.99	13.08	13.40	13.42
60	14.39	13.97	14.90	15.40	**15.61**	14.75	14.85	15.34	15.41
70	16.16	15.72	16.78	17.15	**17.48**	16.60	16.79	17.35	17.41
80	17.90	17.18	18.67	19.11	**19.63**	18.39	18.77	19.42	19.54
90	19.24	18.65	20.17	20.58	**21.33**	19.87	20.24	21.14	21.26
100	20.72	20.29	21.96	22.33	**23.27**	21.56	21.99	22.94	23.14

Tabela: Número médio de circuitos nas partições retornadas.

Resultados dos Experimentos com Ordenação por Rearranjos

n	L&J	FIRST	ALL	RANDOM	MAX
10	**6.86**	**6.86**	**6.86**	**6.86**	**6.86**
20	14.42	14.23	14.23	**14.19**	**14.19**
30	22.99	21.97	21.98	**21.87**	**21.87**
40	31.88	29.92	29.86	**29.67**	**29.67**
50	41.30	38.01	37.92	37.60	**37.58**
60	50.80	46.25	46.15	45.66	**45.59**
70	60.03	54.40	54.21	53.65	**53.59**
80	69.48	62.61	62.23	61.58	**61.46**
90	79.27	71.13	70.76	69.86	**69.74**
100	88.69	79.44	79.01	78.06	**77.86**

Tabela: Média das distâncias usando a operação de DCJ.

Resultados dos Experimentos com Ordenação por Rearranjos

n	L&J	FIRST	ALL	RANDOM	MAX
10	**6.02**	6.23	6.23	6.20	6.21
20	12.16	12.17	12.15	12.06	**11.91**
30	18.64	**17.70**	17.79	17.83	**17.70**
40	25.11	**23.17**	23.22	23.26	23.24
50	32.86	29.03	28.89	28.83	**28.79**
60	39.91	33.98	34.26	34.24	**33.68**
70	46.83	39.48	39.44	39.48	**39.04**
80	54.69	44.45	44.56	44.74	**44.29**
90	61.12	**49.55**	50.10	50.11	49.56
100	68.59	55.04	54.91	55.20	**54.13**

Tabela: Média das distâncias usando as operações de reversão e transposição.

Considerações Finais

Considerações Finais

- Para o `MAX-ECD`, desenvolvemos um algoritmo exato, que resolve um modelo `PLI`, uma heurística gulosa e uma heurística baseada no mesmo modelo `PLI`
  - Para o algoritmo exato foi necessária a criação de um algoritmo de *pricing*
  - Usando os algoritmos propostos, provamos solução ótima em 69.7% das instâncias testadas
  - Criamos um conjunto de instâncias de testes para o problema
- Artigo: "*Algorithms for the Maximum Eulerian Cycle Decomposition Problem*" (Pedro O. Pinheiro, Alexsandro O. Alexandrino, André R. Oliveira, Cid C. de Souza, e Zanoni Dias), *LIII Simpósio Brasileiro de Pesquisa Operacional* (SBPO), novembro de 2021.

Considerações Finais

- Para o `MAX-ACD`, desenvolvemos um algoritmo exato com `PLI`, heurísticas gulosas e uma heurística Busca Tabu
  - Os algoritmos gulosos propostos apresentaram resultados melhores que os do algoritmo de L&J e a Busca Tabu foi capaz de melhorar consistentemente esses resultados
  - Provamos a solução ótima em 96.4% das instâncias testadas
  - Quando aplicados para o problema da Ordenação por Rearranjos, os resultados dos algoritmos propostos foram melhores que L&J
- Artigo: "*Heuristics for Breakpoint Graph Decomposition with Applications in Genome Rearrangement Problems*" (Pedro O. Pinheiro, Alexsandro O. Alexandrino, André R. Oliveira, Cid C. de Souza, e Zanoni Dias) [8], *XIII Brazilian Symposium on Bioinformatics (BSB)*, novembro de 2020.

Trabalhos Futuros

- Desenvolver um algoritmo de *pricing* mais eficiente para o *B&P*
- Desenvolver algoritmos baseados na meta-heurística Algoritmos Genéticos para o `MAX-ECD`
- Desenvolver algoritmos baseados na meta-heurística `GRASP` para o `MAX-ACD`

Referências I

[1] T. Achterberg. SCIP: Solving Constraint Integer Programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[2] A. Caprara. Sorting permutations by reversals and Eulerian cycle decompositions. *SIAM Journal on Discrete Mathematics*, 12(1): 91–110, 1999.

[3] A. Caprara, A. Panconesi, and R. Rizzi. Packing cycles in undirected graphs. *Journal of Algorithms*, 48(1):239–256, 2003.

[4] X. Chen. On sorting unsigned permutations by double-cut-and-joins. *Journal of Combinatorial Optimization*, 25(3):339–351, 2013.

[5] I. Gurobi Optimization. Gurobi Optimizer Reference Manual. *URL* `http://www.gurobi.com`, 2015.

[6] S. L. Hakimi. On Realizability of a Set of Integers as Degrees of the Vertices of a Linear Graph. I. *Journal of the Society for Industrial and Applied Mathematics*, 10(3):496–506, 1962.

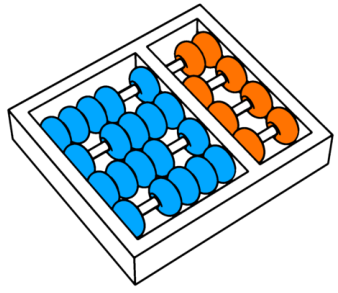
Referências II

[7] G. Lin and T. Jiang. A further improved approximation algorithm for breakpoint graph decomposition. *Journal of Combinatorial Optimization*, 8(2):183–194, 2004.

[8] P. O. Pinheiro, A. O. Alexandrino, A. R. Oliveira, C. C. de Souza, and Z. Dias. Heuristics for breakpoint graph decomposition with applications in genome rearrangement problems. In *Brazilian Symposium on Bioinformatics (BSB)*, pages 129–140. Springer, 2020.

Partição de Grafos Eulerianos em Circuitos

Pedro Olímpio N. de O. Pinheiro



Orientador: Prof. Dr. Zanoni Dias
Coorientador: Prof. Dr. Cid C. de Souza