UNIVERSIDADE FEDERAL DE ALAGOAS
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
NÚCLEO DE INOVAÇÃO TECNOLOGICA

PROGRAMA INSTITUCIONAL DE BOLSAS DE INICIAÇÃO CIENTÍFICA PIBIC/UFAL/FAPEAL/CNPq

RELATÓRIO PARCIAL (2019–2020)

TÍTULO DO PROJETO:
Algoritmos exatos e heurísticos para problemas combinatórios em grafos

TÍTULO DO PLANO DE ATIVIDADES:
Algoritmos heurísticos para o problema da biclique máxima

Nome do Orientador/Unidade / Campus / Email	Rian Gabriel Santos Pinheiro / Instituto de Computação / Campus A. C. Simões/ rian@ic.ufal.br
Nome do Aluno:	Lucas Montenegro Andrade Assunção
Email/Fone (aluno)	lmaa@ic.ufal.br / (82) 991353939

	BOLSISTA CNPQ		BOLSISTA FAPEAL
X	BOLSISTA UFAL		COLABORADOR
	BOLSISTA CNPQ-AF		

Obs.: Marcar com um “X” o tipo de bolsa ou colaborador

RESUMO DO PROJETO

Palavras Chave: Biclique, Grafos, Algoritmos Heurísticos.

O foco deste projeto está voltado para algumas aplicações de Otimização Combinatória em problemas de Teoria dos Grafo. Muitos problemas reais de natureza combinatória podem ser modelados por modelos matemáticos discretos, como grafos por exemplo. A solução destes problemas frequentemente exige uma abordagem algorítmica, juntamente com a implementação destes algoritmos em algum sistema computacional. É importante ressaltar a necessidade de obter algoritmos eficientes, pois tipicamente tanto o volume de dados como a frequência de acessos ao sistema são grandes. Se o problema em questão possui complexidade intrinsecamente alta, uma possibilidade é recorrer a um algoritmo aproximativo ou heurístico em vez de um algoritmo exato.

Neste projeto serão abordados o problema da biclique máxima e suas variantes. Dado um grafo $G = (V, E)$, em que V é um conjunto não vazio de vértice e E é um conjunto de arestas. Um par de subconjuntos disjuntos A e B de V é dito biclique se $(a, b) \in E$ para todo $a \in A$ e $b \in B$. Em outras palavras, uma biclique corresponde a um subgrafo bipartite completo de G. Dado como entrada um grafo G, o objetivo do problema da BICLIQUE MÁXIMA é encontrar uma biclique de G que maximize o número de vértices.

OBJETIVOS DO PROJETO DE PESQUISA

Geral

O projeto tem como objetivo geral desenvolver soluções algorítmicas para problemas de Otimização em Teoria do Grafos utilizando abordagens baseadas em métodos consagrados de otimização como a programação matemática e meta-heurísticas.

Específicos

Como objetivos específicos:

- A construção de algoritmos eficientes para a resolução dos problemas abordados neste projeto;
- Desenvolvimento de *software* de código aberto que possa ser utilizado pela academia e indústria;
- Obtenções de novos resultados científicos e divulgação de pesquisas em veículos de impacto;
- Formação de novos pesquisadores. Este projeto pretende capacitar o aluno a escrever e ler de forma crítica, discutir temas emergentes de pesquisa e capacitá-lo para uma vida acadêmica em pós graduação em nível de mestrado e doutorado, além de incentivá-lo a buscar autonomia em estudar e apresentar soluções a problemas complexos.

OBJETIVO ESPECÍFICO DO PLANO DE ATIVIDADES DO ALUNO

O projeto tem como objetivo geral desenvolver soluções heurísticas para o problema da biclique máxima. O presente plano de trabalho tem os seguintes objetivos específicos: (i) implementar uma meta-heurística para resolver o problema da biclique máxima; (ii) desenvolver novos algoritmos de busca local a fim de de refinar ainda mais os resultados já obtidos pela meta-heurística implementada.

DETALHAR ETAPAS DO PLANO DE ATIVIDADE ALUNO

Durante os três primeiros meses, foram realizadas buscas na literatura sobre problemas relacionadas a biclique e reuniões sobre o rumo da pesquisa. Após essa etapa, o problema da biclique máxima balanceada com peso no vértice foi escolhido dentre os principais problema de biclique e o seu desenvolvimento inicial ocorreu no mês de novembro.

Dado um grafo de entrada $G = (V, E)$, o objetivo de uma biclique máxima balanceada é encontrar uma bipartição no grafo de entrada onde a parte A e a parte B (partes da bipartição) possuem o mesmo número de vértices e tenham a cardinalidade maximizada.

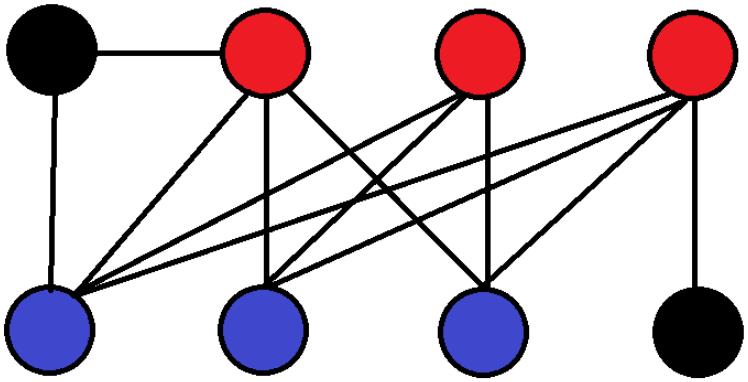

Figura 1: Exemplo de uma Biclique Máxima Balanceada, onde os vértices azuis pertencem a parte A, os vértices em vermelho pertencem a parte B e os vértices pretos não fazem parte da solução

A Figura 1 mostra um grafo de 8 vértices, os quais os vértices pretos pertencem ao grafo, mas não pertencem a solução, pois não fazem parte da biclique máxima balanceada. Os vértices azuis pertencem a parte A da solução e os vértices vermelhos pertencem a parte B da solução.

No problema da biclique máxima balanceada com peso no vértice, além do grafo, uma função w de pesos nos vértices é dada como entrada. Nesta variante, busca-se encontrar uma biclique balanceada em que a soma de todos os pesos dos vértices pertencentes a solução (solução é composta pelas duas partições) seja a maximizada.

Existe uma discussão sobre a complexidade de encontrar uma biclique em um grafo bipartido, já que possuem duas variantes principais de problemas relacionados. Uma delas se caracteriza pela propriedade de balanceamento, ou seja, se $|A| = |B| = K$, sendo K um inteiro, então o problema é $\mathcal{NP}$-completo. A outra variante é quando a biclique não é balanceada, ou seja, o objetivo é maximizar $|A| + |B|$, neste caso o problema pode ser resolvido em tempo polinomial (Garey & Johnson, 1979; Feige & Kogan, 2004; Peeters, 2003). Logo, a complexidade da biclique máxima irá depender da definição utilizada, já que terá uma complexidade para a versão balanceada e outra para a não balanceada. Dessa forma, o problema abordado — biclique balanceada com peso no vértice — sendo uma generalização da versão balanceada sem peso, também é $\mathcal{NP}$-completo. Outros problemas relacionados remoção de arestas/vértices para formação de bicliques são abordados em Yannakakis (1981) e Hochbaum (1998).

Durante o planejamento e escolha do tema, houve um levantamento bibliográfico, onde foram encontrados diversos artigos sobre a biclique máxima balanceada (sem peso) como o Wang et al. (2018), que desenvolveu quatro heurísticas com buscas locais tendo um bom resultado nos testes realizados. Outro artigo encontrado foi o Zhou et al. (2018) que trata sobre um algoritmo

exato para resolver a biclique máxima balanceada (sem peso) usando um algoritmo *upper bound propagation* integrado com um *branch-and-bound* que, em alguns casos, ele consegue reduzir o tempo computacional em 4 ordens de magnitude comparando com um algoritmo original *branch-and-bound*.

Uma outra abordagem sobre a biclique máxima balanceada foi feita pelo Zhou & Hao (2017) que utilizou um algoritmo de Constraint-Based Tabu Search (CBTS) e duas técnicas de redução de grafo que atuam em conjunto com a busca tabu apresentada.

Durante o planejamento, foram estudadas várias meta-heurísticas e foi optado por utlizar o Iterated Local Search — ILS (Lourenço et al., 2010) com o VND (Hansen et al., 2010) utilizando duas estruturas de vizinhança: `one-improvement` e o `swap`(1,1). Além de um mecanismo de reinicialização de solução e uma pertubação baseada na remoção dos vértices. Tal escolha se baseia em bons resultados obtidos pela literatura em problemas similares em grafos (Martins et al., 2015; Bastos et al., 2016; Pinheiro et al., 2018; Nogueira et al., 2018; Nogueira & Pinheiro, 2018, 2019).

A atividade de implementação foi feita em C++ e buscou juntar todas as ideias discutidas e planejadas. A implementação começou com a estruturação para a leitura da entrada e logo após foi desenvolvida a função de inicialização de uma solução inicial randômica. Após a realização de testes, foi implementada a meta-heurística já citada com as suas respectivas estruturas de vizinhança.

Em janeiro de 2020, após várias mudanças e correções no código, saiu a primeira versão do algoritmo heurístico e foi realizada uma bateria de testes, fornecendo resultados promissores.

APRESENTAÇÃO E DISCUSSÃO DOS PRINCIPAIS RESULTADOS OBTIDOS DURANTE O PRIMEIRO SEMESTRE DA PESQUISA

O algoritmo desenvolvido consiste em um ILS-VND com Multi-Start. A parte do ILS é um loop que irá iterar n vezes onde cada iteração ocorre uma busca local (no caso um VND) na solução local para tentar melhorá-la. O Algoritmo 1 apresenta o pseudocódigo da meta-heurística proposta implementada para o problema da biclique máxima balanceada com peso no vértice.

Input : Grafo de entrada $G = (V, E)$, Função de peso nas arastas w

```
1  ILS-VND(G)
2      S_0 = Initialize(G)                          // solução inicial
3      S* = S                                        // melhor solução
4      local_best_p = Peso(S)                        // biclique máxima
5      i = 1 while critério de parada do
           // aplica a busca local --- VND
6          S' = LocalSearch(S', G)
           // Critério de aceitação
7          (S, S*, i, local_best_p) = Accept (S, S*, S', i, local_best_p, G)
8          if Não melhorou em x iterações then
                                          // Caso não melhore a solução
9              S' = RestartSolution (S')              // Reinicia a solução
10             S' = Initialize(G)                // Gera solução randômica
11         else
                                      // Caso a solução tenha melhorado
12             S' = Perturb(S, G)                          // perturbação
13
14     return S*
```

Algorithm 1: Pseudocódigo do ILS para o problema da biclique máxima.

No ILS-VND:

- Linha 2: É criada uma solução inicial aleatória que atenda as propriedades da biclique balanceada
- Linha 5: É feita uma quantidade n de iterações
- Linha 6: Ocorre a busca local na solução atual onde cada partição é maximizada e pode ocorrer um aumento na solução
- Linha 8: É feita uma validação da solução local e caso seja a melhor solução encontrada, S^* é atualizado de acordo com S'
- Linha 8: Verifica se a solução não conseguiu melhorar durante x iterações, S' é reiniciado e é gerada outra nova solução randômica
- Linha 11: Caso contrário, ocorre uma perturbação em S' onde é feita uma troca na solução atual, preservando as propriedades da biclique, a fim de encontrar outros máximos locais

```
1  LocalSearch(S', G)
2      k ← 1 // neighborhood structure selector
3      while k ≤ |𝒩| do
           // find a improving neighbor S'' of S' in 𝒩_k(S')
4          S'' ← BestImprovement(𝒩_k, S')
5          if Peso(S'') > Peso(S') then
6              k ← 1
7              S' ← S''
8          else
9              k ← k + 1
10     return S'
```

Algorithm 2: Busca local VND.

O Algoritmo 2 representa a busca local (VND) que é chamada durante o algoritmo do ILS (ver Algoritmo 1). Essa busca local utiliza um *bestImprovements* — busca pelo melhor vizinho — nas duas estruturas de vizinhança: $\mathcal{N}_1$ e $\mathcal{N}_2$, que correspondem a `oneImprovement` e `swap`, respectivamente.

- `oneImprovement` — busca o par de vértices v, u para serem adicionados às partes A e B, respectivamente, de modo a maximizar o peso da solução (sem a remoção de vértice já contido na solução);
- `swap` — busca o par de vértices $v \in A$ (ou $v \in B$) e $w \notin A \cup B$, para trocarem de lugar, ou seja, v sairá da solução e w fará parte da solução, de modo a maximizar o peso da solução.

Inicialmente o VND utiliza o `oneImprovement` para encontrar o melhor par a ser incluído na solução. Caso seja possível, é feita essa adição, caso contrário, utilizaremos a estrutura de vizinhança `swap`. Nesta estrutura, inicialmente, é chamada a função `evalSwap()` para cada parte com o objetivo de encontrar o melhor par para realizar a troca. Somente se o par promover um melhora no peso da solução, será realizada o `swap` e o algoritmo volta para a estrutura `oneImprovement`. Cajo não haja um par que melhore a solução, então a busca local é finalizada e é retornada a solução local atualizada.

Finalizando a busca local, é verificado se a solução local é melhor que a *best solution* e, caso seja, ocorre uma validação. Sendo aceita a validação, a solução atual passa a ser a melhor solução.

Se a solução local não for melhor que a *best solution* durante **x** iterações seguidas, então a solução local é reiniciada. Caso contrário ocorre uma perturbação. Dessa forma, são feitas trocas aleatórias nos vértices da solução por vértices que não estão presentes na solução (mantendo as propriedades da biclique) não calculando se haverá uma perda ou ganho na solução local, pois o objetivo é modificá-la para encontrar outro máximo local.

Durante o seu desenvolvimento, foram implementadas duas estruturas de dados auxiliares para cada parte, com finalidade de reduzir a complexidade dos operadores de busca local do algoritmo, que foram as estruturas $\texttt{mu}_A$, $\texttt{mu}_B$, $\texttt{tightness}_A$ e $\texttt{tightness}_B$.

O vetor `mu` serve para verificar o ganho/perda ao inserir o vértice na solução da partição verificada. Esse cálculo é feito ao inserir ou remover um vértice. Assim, $\forall v \in V$ o valor $\texttt{mu}_A[v]$ quantifica a melhoria na qualidade da solução com a inserção de v na parte A. Em outras palavras, o valor do peso de v subtraído dos vizinhos de v na parte A. Com isso, após a inserção de um vértice, não é necessário calcular o `mu` de todos os vértices do grafo, apenas dos vizinhos do vértice inserido/removido, diminuindo, assim, a complexidade.

Já o valor $\texttt{tightness}_A[u]$ serve para indicar quantos vizinhos do vértice u estão na solução da parte A e, a partir disso, verificar se u é um possível candidato para a solução. Isso permite que não seja necessária percorrer todos os vizinhos dos vértices presentes na solução a todo instante, o que irá reduzir a complexidade do algoritmo. De forma similar para o $\texttt{tightness}_B$.

Note que sabe-se que um vértice fora da solução poderá ser trocado com um da solução em O(1) utilizando as estruturas mu e tightness. Para isso o mu de ve ser maior que 0 e o tightness da sua parte deve valer 1 e o tightness da outra parte deve valer $|S|$ (tamanho da solução da outra parte). Assim, o best improvment do swap(1,1) pode ser feito em O(V).

Tabela 1: Resultados do algoritmo heurístico e de um algoritmo exato para o mesmo problema.

Instância				ILS_{VND}			B&B	
Nome	Densidade	$\|V\|$	$\|E\|$	Best	Média	Tempo (s)	Ótimo	GAP
C125-9	0.898452	125	1343	1142	1118	0.006023	1171	2,48%
C250-9	0.899084	250	2940	2550	2437	0.009264	2840	10,21%
MANN-a9	0.927273	45	351	300	300	0.002946	321	6,54%
MANN-a27	0.990148	378	3636	2865	2655	0.01989	3540	19,07%
MANN-a45	0.9963	1035	10377	9420	8073	0.06275	10293	8,52%
c-fat200-1	0.0770854	200	1312	730	676	0.01466	730	0,00%
c-fat200-2	0.162563	200	1370	745	713	0.02023	745	0,00%
c-fat200-5	0.425779	200	1382	743	730	0.03599	743	0,00%
c-fat500-1	0.0357435	500	3338	1861	1740	0.03696	1861	0,00%
c-fat500-2	0.0732585	500	3522	1901	1849	0.05177	1901	0,00%
c-fat500-5	0.1859	500	3590	1885	1860	0.0903	1885	0,00%
c-fat500-10	0.373764	500	3648	1916	1834	0.1283	1916	0,00%
johnson8-2-4	0.555556	28	208	224	224	0.002305	224	0,00%
san200-0-7-1	0.7	200	3348	3492	2796	0.006755	3492	0,00%
san200-0-9-1	0.9	200	2096	1784	1732	0.006959	1856	3,88%
san200-0-9-2	0.9	200	2456	2024	2024	0.007073	2312	12,46%
san200-0-9-3	0.9	200	2344	2024	1927	0.008794	2552	20,69%
gen200-p0-9-44	0.9	200	2528	2080	1760	0.01207	2584	19,50%
gen200-p0-9-55	0.9	200	2296	2144	1767	0.008727	2152	0,37%

A Tabela 1 mostra os resultados obtidos com a versão inicial do algoritmo feito. A coluna `Nome` representa as instâncias utilizadas para a comparação, $|V|$ representa a quantidade de vértices, $|E|$ representa a quantidade de arestas da entrada e a coluna `Densidade` representa a seguinte fórmula: $(2 \times |E|) / (|V| \times (|V| - 1))$ para cada instância.

A coluna `Best` mostra o melhor resultado obtido em cada instância, a coluna `Média` mostra a média dos resultados obtidos em cada instância e a coluna `Tempo(s)` representa a média do tempo de execução de cada caso. Essas três colunas representam resultados do algoritmo heurístico, que foi executado 10 vezes para cada instância com critério de parada de 100 iterações para cada execução de instância.

A coluna `Ótimo` mostra o resultado exato obtido pelo algoritmo Branch-and-Bound — proposto pelo aluno Thiago Santos no outro projeto PIBIC (Santos, 2020) — em cada instância.

A coluna `GAP` representa o quão perto o resultado do algoritmo heurístico está do algoritmo

exato, sendo calculado pela fórmula:

$$GAP = \frac{\texttt{exato} - \texttt{melhor}_{\texttt{ILS}}}{\texttt{exato}}$$

Um GAP de 0% indica que o algoritmo heurístico chegou na resposta exata.

As instâncias utilizadas fazem parte do desafio DIMACS. Como elas não possuem peso, então foi feito um gerador que coloca peso nos vértices de acordo com a seguinte fórmula:

$Weight_{\ i}$ = [(i $\times$ f) % $|V|$] + ($|V|$ - 1), onde **i** é o índice do vértice, **f** é a quantidade de arestas ($|E|$) dividida pela quantidade de vértices ($|V|$) e **%** é a operação resto.

Note que o algoritmo heurístico obteve a solução ótima em 9 instâncias de 19. O GAP médio obtido nas 19 instâncias foi de 5,45%, além disso, a métrica GAP ficou limitada a 20,69%, ou seja, o pior resultado obtido está 20,69% próximo da solução exata, o que indica que o algoritmo ILS (Algoritmo 1) atinge um valor próximo em pouco tempo.

CRONOGRAMA DE ATIVIDADES

Atividade 1: Levantamento do estado da arte a respeito do problema nas bases científicas descritas;

Atividade 2: Planejamento e análise (teórica) dos algoritmos propostos;

Atividade 3: Implementação heurística, bem como propor melhorias para soluções já existentes;

Atividade 4: Comparação e análise dos resultados obtidos com os resultados já disponíveis na literatura;

Atividade 5: Escrita do relatório final e artigo a serem submetidos a conferências ou revistas.

Tabela 2: Cronograma de atividades do Aluno 2

ATIV.	**Meses**											
	2019					**2020**						
	AGO	SET	OUT	NOV	DEZ	JAN	FEV	MAR	ABR	MAI	JUN	JUL
1	OK	OK	OK	OK	OK	OK	■	■	■	■	■	■
2		OK	OK	OK	OK							
3				OK	OK	OK	■	■				
4					X	OK	■	■	■	■		
5										■	■	■

O levantamento bibliográfico foi uma atividade realizada durante todo esse tempo de pesquisa para verificar se novos artigos estão sendo escritos sobre o tema e é uma atividade que irá durar até o final da pesquisa. A fase de planejamento dos algoritmos e meta-heurísticas aconteceu durante os meses iniciais, após um levantamento da bibliografia já existente, tendo o seu término em meados do mês de novembro onde começou a implementação.

Na atividade 3 ocorreu a implementação do algoritmo ILS-VND com Multi-Start e teve a sua primeira versão em janeiro de 2020, sendo usado para testes e obtenção dos resultados parciais. A implementação ainda ocorrerá durante os próximos meses, pois será necessária a otimização do código e calibragem dos parâmetros.

Como não foi encontrada nenhuma bibliografia relacionada ao problema escolhido (Biclique Máxima Balanceada com pesos nos vértices), então a atividade 4 foi realizada em janeiro, após a obtenção da primeira versão do algoritmo, comparando o resultado parcial obtido com o resultado do algoritmo exato que trata o mesmo problema feito pelo Santos (2020).

A escrita do relatório final será a última parte da pesquisa, sendo realizada nos últimos meses e terá um artigo submetido para o SBPO (Simpósio Brasileiro de Pesquisa Operacional).

RELACIONE OS PRINCIPAIS FATORES POSITIVOS E NEGATIVOS QUE INTERFERIRAM NA CONDUÇÃO DO PROJETO E PLANO DE ATIVIDADES

Um fator essencial para a evolução da pesquisa foi a assistência fornecida pelo orientador Rian Pinheiro, que auxiliou durante todas as etapas do desenvolvimento da pesquisa tanto ao mostrar e explicar o(s) problema(s) a ser(em) solucionado(s) quanto no indicativo do rumo que a pesquisa deveria levar.

Entretanto, a escassez de artigos relacionados ao problema da biclique máxima balanceada com peso no vértice e a dificuldade de manipular e solucionar um problema em um grafo bipartido foram os dois principais fatores negativos que interferiram na condução do projeto.

Referências

Bastos, L. d. O., Ochi, L. S., Protti, F., Subramanian, A., Martins, I. C. & Pinheiro, R. G. S. (2016), 'Efficient algorithms for cluster editing', *Journal of Combinatorial Optimization* **31**, 347–371.

Feige, U. & Kogan, S. (2004), 'Hardness of approximation of the balanced complete bipartite subgraph problem'.

Garey, M. R. & Johnson, David S., .-a. (1979), *Computers and intractability : a guide to the theory of NP-completeness*, San Francisco : W.H. Freeman. Includes indexes.

Hansen, P., Mladenović, N., Brimberg, J. & Pérez, J. A. M. (2010), Variable neighborhood search, *in* M. Gendreau & J.-Y. Potvin, eds, 'Handbook of Metaheuristics', Springer US, Boston, MA, pp. 61–86. URL https://doi.org/10.1007/978-1-4419-1665-5_3.

Hochbaum, D. S. (1998), 'Approximating clique and biclique problems', *Journal of Algorithms* **29**(1), 174 – 200. URL http://www.sciencedirect.com/science/article/pii/S0196677498909646.

Lourenço, H. R., Martin, O. C. & Stützle, T. (2010), Iterated local search: Framework and applications, *in* M. Gendreau & J.-Y. Potvin, eds, 'Handbook of Metaheuristics', Springer US, Boston, MA, pp. 363–397. URL https://doi.org/10.1007/978-1-4419-1665-5_12.

Martins, I. C., Pinheiro, R. G. S., Protti, F. & Ochi, L. S. (2015), 'A hybrid iterated local search and variable neighborhood descent heuristic applied to the cell formation problem', *Expert Syst. Appl.* **42**, 8947–8955.

Nogueira, B. C. e. S. & Pinheiro, R. G. S. (2018), 'A cpu-gpu local search heuristic for the maximum weight clique problem on massive graphs', *Computers & OR* **90**, 232–248.

Nogueira, B. C. e. S. & Pinheiro, R. G. S. (2019), 'A gpu based local search algorithm for the unweighted and weighted maximum s-plex problems', *Annals of Operations Research* **284**, 367–400.

Nogueira, B. C. e. S., Pinheiro, R. G. S. & Subramania, A. (2018), 'A hybrid iterated local search heuristic for the maximum weight independent set problem', *Optimization Letters* **12**, 567–583.

Peeters, R. (2003), 'The maximum edge biclique problem is np-complete', *Discrete Applied Mathematics* **131**(3), 651 – 654. URL http://www.sciencedirect.com/science/article/pii/S0166218X03003330.

Pinheiro, R. G. S., Martins, I. C., Protti, F. & Ochi, L. S. (2018), 'A matheuristic for the cell formation problem', *Optimization Letters* **12**, 335–346.

Santos, T. J. S. (2020), 'Comunicação pessoal'.

Wang, Y., Cai, S. & Yin, M. (2018), 'New heuristic approaches for maximum balanced biclique problem', *Information Sciences* **432**, 362 – 375. URL http://www.sciencedirect.com/science/article/pii/S0020025517311337.

Yannakakis, M. (1981), 'Edge-deletion problems', *SIAM Journal on Computing* **10**(2), 297–309. URL https://doi.org/10.1137/0210021.

Zhou, Y. & Hao, J.-K. (2017), 'Combining tabu search and graph reduction to solve the maximum balanced biclique problem', *ArXiv*.

Zhou, Y., Rossi, A. & Hao, J.-K. (2018), 'Towards effective exact methods for the maximum balanced biclique problem in bipartite graphs', *European Journal of Operational Research* **269**(3), 834 – 843. URL http://www.sciencedirect.com/science/article/pii/S0377221718302194.