LII Simpósio Brasileiro de Pesquisa Operacional
João Pessoa - PB, 29 de setembro a 02 de outubro de 2020

Algoritmo GRASP-VND para o problema da máxima biclique balanceada com peso no vértice

Lucas M. A. Assunção
Instituto de Computação, Universidade Federal de Alagoas
Av. Lourival Melo Mota, s/n, Tabuleiro do Martins CEP:57072-900, Maceió - AL
lmaa@ic.ufal.br

Rian G. S. Pinheiro
Instituto de Computação, Universidade Federal de Alagoas
Av. Lourival Melo Mota, s/n, Tabuleiro do Martins CEP:57072-900, Maceió - AL
rian@ic.ufal.br

RESUMO

Este trabalho trata do problema da máxima biclique balanceada com peso no vértice. Dado um grafo não direcionado com peso nos vértices, o objetivo do problema é encontrar uma biclique (subgrafo bipartite completo) balanceada que possua o máximo somatório dos pesos. Além de ser um problema $\mathcal{NP}$-difícil, possui diversas aplicações preeminentes. Este artigo propõe uma meta-heurística GRASP com uso de uma busca local VND com três estruturas de vizinhança. A heurística proposta foi avaliada nas instâncias da literatura (DIMACS e BHOSLIB) e os resultados indicam que o algoritmo proposto, em comparação com o algoritmo exato CPLEX, foi capaz de encontrar todas as soluções ótimas conhecidas em um baixo tempo computacional.

PALAVRAS CHAVE. Otimização em grafo, bicliques, meta-heurísticas.

Tópicos (Meta-Heurísticas, Grafo)

ABSTRACT

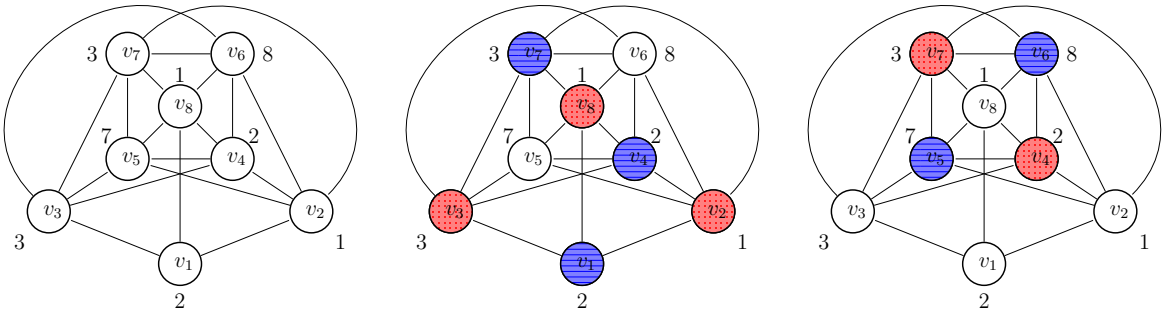
This work deals with the problem of maximum weight balanced biclique. Given a non-directional graph where each vertex has a weight, the problem consists in finding a balanced biclique that maximum total weight. This problem is $\mathcal{NP}$-hard and is a prominent model with numerous applications. We present a GRASP+VND procedure that explores three neighborhood structures for solving this problem. The proposed heuristic was evaluated using the literature benchmark instances (DIMACS and BHOSLIB) and the exact CPLEX algorithm, for all instances with known optimal, our method is able to determine the optimum solutions in practical runtime.

KEYWORDS. Graph Optimization, bicliques, metaheuristics.

Topics (Metaheuristics, Graph)

1. Introdução

Dado um grafo $G = (V, E)$ de entrada, o problema da biclique (subgrafo bipartite completo) máxima balanceada tem como objetivo encontrar uma biclique balanceada (partes com o mesmo número de vértice) que tenha a cardinalidade maximizada. Uma generalização deste problema é a MÁXIMA BICLIQUE BALANCEADA COM PESO NOS VÉRTICES (PMBBPV), nesta variante, além do grafo de entrada, existe uma função de peso $w : V \to \mathbb{R}$ que atribui a cada vértice $v \in V$ um peso $w(v)$. Dessa forma, busca-se encontrar uma biclique balanceada em que a soma de todos os pesos dos vértices pertencentes a solução seja a maximizada. A Figura 1a mostra um grafo G de entrada para o PMBBPV com 8 vértices. Uma solução viável $G' = (V', E')$ com peso 12 é apresentada na Figura 1b, note que G' possui 6 vértices os quais os seus vértices V' são particionados em V'_1 e V'_2 formados pelos vértices em azul tracejado e vermelho pontilhado, respectivamente. G' é uma biclique balanceada uma vez que: (i) para todo par de vértice $v \in V'_1$ e $u \in V'_2$ tem-se que $vu \in E$, (ii) V'_1 e V'_2 são conjuntos independentes e (iii) $|V'_1| = |V'_2|$. Finalmente, a Figura 1c apresenta a biclique G'' com peso 20 que é a solução ótima do problema, mesmo tendo uma cardinalidade menor que G'.



(a) Instância. (b) Sol. viável com peso 12. (c) Sol. ótima com peso 20.

Figura 1: Biclique máxima ponderada.

Existe uma discussão sobre a complexidade de encontrar uma biclique em um grafo bipartido, já que existem diversas variantes de problemas relacionados. Na variante com a restrição de balanceamento, ou seja $|V_1| = |V_2| = K$, sendo K um inteiro, o problema é $\mathcal{NP}$-completo. Em outro problema, quando a biclique não precisa ser balanceada, ou seja, o objetivo é maximizar $|V_1| + |V_2|$, a solução ótima pode ser encontrada em tempo polinomial caso o grafo de entrada já esteja biparticionado [Garey e Johnson, 1979], caso contrário o problema é $\mathcal{NP}$-completo [Garey e Johnson, 1979; Feige e Kogan, 2004; Peeters, 2003]. Como o problema abordado é uma generalização da versão balanceada sem peso, então ele é $\mathcal{NP}$-completo. Dentre as diversas aplicações destacam-se: biologia computacional, sistemas nanoeletrônicos e VLSI [Wang et al., 2018].

Tratando do problema da biclique máxima balanceada sem peso, várias soluções algorítmicas foram propostas na literatura. Wang et al. [2018] propuseram quatro heurísticas baseadas em buscas locais. Já Zhou e Hao [2017] utilizaram um algoritmo *constraint-based tabu search* (CBTS) e duas técnicas de redução do grafo. Em outra abordagem, Zhou et al. [2018] desenvolveram um algoritmo exato para o problema. Recentemente, Li et al. [2020] propuseram o algoritmo *general swap-based multiple neighborhood adaptive search* (SBMNAS). No entanto, em todos algoritmos, o grafo de entrada já é

biparticionado além de não ser ponderado, o que torna o problema diferente do abordado neste trabalho.

Apesar de ser semelhante, os autores desconhecem soluções especificamente projetadas para a versão ponderada com o grafo de entrada livre. Dessa forma, para resolver o PMBBPV, este trabalho propõe um algoritmo GRASP — *Greedy Randomized Adaptive Search Procedure* [Feo e Resende, 1995] — com o VND [Mladenovic e Hansen, 1997]. A hibridização GRASP e VND foi utilizada com êxito na resolução de diversos problemas combinatórios, Resende e Ribeiro [2019, Seção 6.6] apresentam uma bibliografia detalhada sobre essa extensão.

Este trabalho é estruturado da seguinte forma: na Seção 2 é apresentado o algoritmo GRASP$_{\text{VND}}$. A Seção 3 apresenta uma formulação matemática para o problema. A Seção 4 detalha os resultados do experimentos computacionais e, por fim, a Seção 5 apresenta as conclusões e trabalhos futuros.

2. Proposta GRASP com VND

A meta-heurística GRASP, proposta por Feo e Resende [1995], consiste em um método guloso aleatório seguido de uma busca local. Dentre as meta-heurísticas existentes, o GRASP tem se mostrado uma das mais competitivas [Resende e Ribeiro, 2019]. No GRASP$_{\text{VND}}$ proposto, a fase construtiva (Seção 2.1) cria uma solução viável considerando o peso do vértice como critério guloso. Já na etapa da busca local (Seção 2.2), foi utilizado um método VND com 3 estruturas de vizinhança.

O Algoritmo 1 apresenta o pseudocódigo do GRASP$_{\text{VND}}$ desenvolvido. Inicialmente, utilizando o construtivo do GRASP$_{\text{VND}}$ é criada uma solução inicial, caso necessário, ocorre um balanceamento na solução seguido de uma busca local (linhas 2–5). Na linha 7, a variável x (inicialmente com valor β) indicará por mais quantas iterações o algoritmo poderá encontrar a atual melhor solução. Isso serve para que o programa não perca tempo tentando melhorar uma solução que supostamente já é ótimo global.

O começo do *loop* do GRASP$_{\text{VND}}$ é dado na linha 8 por um critério de parada. As linhas 9–12 repetem o mesmo processo das linhas 2–5. Após a busca local (VND), é verificado se a solução local (S') é melhor que a solução atual (S^*) através da função `custo` que calcula o peso total das soluções (linha 13). Caso ela seja a melhor, S^* é atualizado (linha 14) e o valor de x é reinicializado para o valor do parâmetro de encerramento β (linha 15). Caso $S' = S^*$ — utilizando a função `custo` para a verificação —, o valor de x é decrementado (linha 17). Finalmente, as linhas 19–20 representam a finalização do algoritmo retornando a melhor solução encontrada até o momento.

2.1. Fase de construção

O método construtivo do GRASP$_{\text{VND}}$ desenvolvido utiliza uma lista restrita de candidatos (RCL) baseada em qualidade (*quality-based RCL*) de acordo com o peso dos vértices. O parâmetro α é utilizado para definir a qualidade da RCL, seu valor varia de 0 (totalmente aleatório) a 1 (puramente guloso). Após a criação da RCL, é escolhido aleatoriamente um dos vértices para pertencer à solução.

O Algoritmo 2 descreve o pseudocódigo do algoritmo construtivo. Inicialmente, a variável i é inicializada com o valor 1 (linha 2), esta variável indica a parte corrente da biclique. Em seguida, a criação da solução irá ocorrer enquanto houver vértices disponíveis (linha 3). É criada a cada iteração uma nova RCL com vértices disponíveis e que atendam aos critérios. Após isso, será escolhido aleatoriamente um dos vértices da RCL para entrar na solução. Nas linhas 4 e 5, as variável c_min e c_max recebem o menor e maior peso dos vértices aptos a entrar na parte V_i', respectivamente. Após a inicialização da RCL, na linha 6, ocorre a criação do intervalo de candidatos aptos a entrar na RCL (na linha

Input : Grafo de entrada $G = (V, E)$, função de peso nos vértices w, parâmetros α e β
Output: Biclique balanceada S^*

```
1  GRASP_VND (G, w, α, β)
2      S = greedyRandomizedConstructive(α) // solução inicial
3      if S não é balanceada then
4          balanceBiclique(S)
5      S = LocalSearch(S)                                  // aplica VND
6      S* = S                                              // melhor solução
7      x = β                          // parâmetro de encerramento do algoritmo
8      while critério de parada do
9          S' = greedyRandomizedConstructive(α) // nova solução
10         if S' não é balanceada then
11             balanceBiclique(S')
12         S' = LocalSearch(S')                            // aplica VND
13         if custo(S*) < custo(S') then
14             S* = S'                          // atualiza a melhor solução
15             x = β                  // reinicia o parâmetro de encerramento
16         else if custo(S*) == custo(S') then
17             x = x − 1                        // caso a solução seja a mesma
18         if x == 0 then
19             return S* // finaliza algoritmo (β iterações sem melhora)
20     return S*
```

Algoritmo 1: Pseudocódigo do $\text{GRASP}_{\text{VND}}$

8). Após a criação do intervalo, a linha 9 irá percorrer todos os vértices livres (vértice que podem ser adicionados em V_i) e irá analisar se o peso do vértice em análise pertence ao $intervalo$. Caso pertença, na linha 11 ocorre a adição do vértice à RCL. Na linha 12, com a criação da RCL, um vértice é escolhido aleatoriamente e é inserido na solução (linha 13). Após a adição de novo vértice à solução, a parte da bipartição é alterada (linha 14). Concluindo, na linha 15 ocorre o retorno da solução.

2.2. Busca Local

Após o método construtivo, inicia-se o método de busca local baseado no VND [Mladenovic e Hansen, 1997]. Tendo como entrada uma biclique inicial já balanceada, a busca local tenta melhorar a solução com a exploração do espaço de solução. Dado um conjunto de estruturas de vizinhança, o VND explora o espaço de solução trocando sistematicamente as estruturas de vizinhança. O método aceita apenas soluções que melhorem a solução corrente e, nesses casos, o método volta à primeira estrutura de vizinhança e inicia uma nova busca.

O método proposto utiliza três estruturas de vizinhança: "add"; "$swap(1,1)$" e "$swap(2,2)$", com mecanismo de *best-improvement* para a segunda estrutura e *first-improvement* para as demais. A primeira estrutura é o add que tenta adicionar um par aleatório de vértices na solução. Inicialmente, é escolhido aleatoriamente o primeiro vértice livre (para a parte V_1) e, a partir desse vértice, ocorre uma busca pelos seus vizinhos a fim de encontrar um vértice livre para a parte V_2 que, juntos, atendam às

Input : Grafo $G = (V, E)$ e parâmetro α da *quality-based* RCL
Output: Conjunto de vértice $V' \subseteq V$ particionado em V'_1 e V'_2 de forma a induzir uma biclique

```
1  greedyRandomizedConstructive(G = (V, E), α)
2      i = 1
3      while existe vértice disponível em V_i do
4          c_min = minWeight(V_i)            // menor peso dos vértices disp.
5          c_max = maxWeight(V_i)            // maior peso dos vértices disp.
6          RCL = {}                          // cria um conjunto vazio
7          // cria um intervalo da RCL
8          intervalo = [c_min + α * (c_max − c_min), c_max]
9          for u in vértices livres em V_i do
10             if weight(u) está no intervalo then
11                 RCL = RCL ∪ {u}           // adiciona u ao vetor RCL
12         t = random(RCL)                   // escolhe vértice aleatório da RCL
13         addVertex(t, V'_i)                // adiciona t na solução V'
14         i = (i mod 2) + 1
15     return V'
```

Algoritmo 2: Pseudocódigo da fase construtiva do grasp.

propriedades da biclique e apresentem uma melhora na solução. Caso o vértice escolhido aleatoriamente para V_1 não encontre um vizinho disponível para a parte V_2, então ele é descartado e é escolhido um outro vértice (entre os vértices livres para V_1) e o procedimento se repete. O processo acaba quando for encontrado o primeiro par que atenda às condições ou quando não é encontrado nenhum par disponível.

A segunda estrutura é o $swap(1, 1)$. Esta estrutura irá realizar a melhor troca, ou seja, a que irá gerar o maior acréscimo no peso da solução, entre um vértice que está na solução e um vértice não-livre (não está na solução e não está disponível) desde que mantenha as propriedades de biclique balanceada. Por fim, a estrutura $swap(2, 2)$ é bem parecida com a segunda estrutura de vizinhança, entretanto ela realiza uma troca entre dois vértices da solução e dois vértices não-livres, desde que essa troca realize uma melhora no peso da solução e mantenha as propriedades da biclique balanceada. Outro fator importante é que a $swap(2, 2)$ realiza a primeira troca possível.

2.3. Estruturas auxiliares utilizadas

No desenvolvimento do algoritmo foram criadas três estruturas auxiliares. A primeira é o vetor `solution`, dividida em 3 partes, a parte da *solução*, que representa os vértices pertencentes à solução, a parte dos *vértices livres*, que estão livres para entrar na solução e a parte dos *vértices não-livres*, que representa os vértices impossibilitados de entrarem na solução. Essa estrutura é utilizada na literatura nos trabalhos de Andrade et al. [2012] e Nogueira et al. [2018].

A segunda estrutura é o vetor `mu` que indica o quanto a solução irá perder ou ganhar caso um vértice entre na solução. A terceira estrutura utilizada é o vetor `tightness` que determina a quantidade de vizinhos, pertencentes à solução, de um vértice. Para cada uma das três estruturas auxiliares utilizadas foram criadas duas, uma para cada parte da biclique. Por exemplo, no lugar de único vetor `solution`, foram utilizados dois vetores $\texttt{solution}_{V_1}$ e $\texttt{solution}_{V_2}$ para as partes V_1 e V_2, respectivamente.

3. Formulação Matemática

Neste trabalho será proposta uma formulação matemática com o objetivo de comparar e avaliar a performance da heurística GRASP$_{\text{VND}}$. A formulação é definida como:

$$\max \quad \sum_{v_i \in V} w_i x_i + \sum_{v_i \in V} w_i y_i \tag{1}$$

$$\text{s. a} \quad x_i + y_i \leq 1 \qquad \forall v_i \in V \tag{2}$$

$$x_i + x_j \leq 1 \text{ e } y_i + y_j \leq 1 \qquad \forall v_i v_j \in E \tag{3}$$

$$x_i + y_j \leq 1 \text{ e } y_i + x_j \leq 1 \qquad \forall v_i v_j \notin E \tag{4}$$

$$\sum_{v_i \in V} x_i = \sum_{v_i \in V} y_i \tag{5}$$

$$x_i, y_i \in \{0, 1\} \qquad \forall v_i \in V. \tag{6}$$

A formulação utiliza variáveis decisão binárias x_i e y_i para cada vértice $v_i \in V$. A variável x_i vale 1 se e somente se o vértice v_i pertencerá à parte V_1 na solução final. Da mesma forma, y_i determina se v_i pertencerá à parte V_2. A função objetivo (1) contabiliza o peso da biclique. As restrições (2) determinam que um vértice só pode pertencer a uma parte. As restrições (3) determinam que vértices adjacentes não podem pertencer a mesma parte. As restrições (4) determinam que vértices não-adjacentes não podem pertencer a partes distintas. A restrição (5) força a biclique ser balanceada. Finalmente, as restrições (6) determinam os domínios das variáveis.

4. Testes Computacionais

O GRASP$_{\text{VND}}$ foi implementado na linguagem C++ e compilado com g++ v7.4.0 com a opção de otimização -O3 em um computador com Windows 10 (versão 1909), 64 bits, 8 GB de RAM e processador Intel Core i5-8250U 1.60GHz. O algoritmo exato foi implementado com o CPLEX 19.9.

A Tabela 1 representa os resultados obtidos com os grupos de instâncias DIMACS e BHOSLIB. Neste experimento, o GRASP$_{\text{VND}}$ foi executados 10 vezes cada instância para obter o `Best` que representa o melhor resultado obtido das 10 execuções, a `Média` das 10 execuções e o `Tempo(s)` médio, em segundos. Os resultados obtidos utilizando o algoritmo exato para cada instância são apresentados pela coluna `Ótimo` (indicando o resultado ótimo da biclique) e pela coluna `Tempo(s)`, em segundos. A coluna `Gap` mostra a diferença relativa entre o `Best` da heurística e o `Ótimo` encontrado no exato. Quanto mais próximo de 0%, melhor o resultado, pois indica que a resposta da heurística está muito próxima do ótimo. Para as instâncias que possuem '—' o algoritmo exato não conseguiu encontrar a resposta dentro do tempo limite de 1 hora.

Nas instâncias DIMACS, das 69 instâncias testadas, 45 obtiveram um GAP de 0%, chegando no resultado ótimo. Já com as instâncias BHOSLIB, das 36 instâncias testadas, 35 obtiveram um GAP de 0%. Para as demais instâncias, em ambas as tabelas, o CPLEX não encontrou o ótimo dentro do tempo limite. Em todas as instâncias com ótimo conhecido, o GRASP$_{\text{VND}}$ foi capaz de encontrar a solução ótima com um tempo muito abaixo do algoritmo exato.

5. Conclusões e Trabalhos Futuros

Neste trabalho foi desenvolvido uma meta-heurística GRASP com o VND na busca local para tratar a PMBBPV. Formas eficientes de implementá-la foram estudadas e bons resultados foram obtidos. Foi feita uma comparação do GRASP$_{\text{VND}}$ com o algoritmo exato e foram utilizadas instâncias da

literatura. Nos testes realizado, foi possível constatar que o $GRASP_{VND}$ teve um ótimo desempenho tanto em termos de qualidade das soluções (Gap) e tempo computacional. Como trabalhos futuros, serão estudadas novas meta-heurísticas, estruturas de vizinhança e formas de acelerar a busca local.

Tabela 1: Resultados do $GRASP_{VND}$ para as instâncias do DIMACS e BHOSLIB.

Instância	$GRASP_{VND}$				CPLEX		Instância	$GRASP_{VND}$				CPLEX	
	Best	Média	Tempo (s)	Gap (%)	Ótimo	Tempo (s)		Best	Média	Tempo (s)	Gap (%)	Ótimo	Tempo (s)
Instâncias DIMACS													
C125-9	673	673	0.07	0.00	673	4.82	hamming6-2	242	242	0.04	0.00	242	0.06
C250-9	1180	1157	0.11	0.00	1180	10.83	hamming6-4	455	455	0.03	0.00	455	2.13
C500-9	1373	1349	0.04	0.00	1373	606.53	hamming8-2	786	786	0.11	0.00	786	0.87
C1000-9	1596	1526	0.51	—	—	—	hamming8-4	4048	4048	0.07	0.00	4048	285.47
C2000-5	2524	2427	2.66	—	—	—	hamming10-2	798	798	0.81	0.00	798	27.64
C2000-9	1777	1750	3.83	—	—	—	hamming10-4	5636	5600	0.50	—	—	—
C4000-5	2703	2599	11.75	—	—	—	p-hat300-3	1602	1598	0.16	0.00	1602	2220.44
MANN-a9	255	255	0.02	0.00	255	3.89	p-hat500-2	1766	1741	0.27	—	—	—
MANN-a27	1173	1160	0.05	0.00	1173	1.52	p-hat500-3	1887	1820	0.26	—	—	—
MANN-a45	1191	1190	0.30	0.00	1191	30.42	p-hat1000-1	1892	1802	0.26	—	—	—
MANN-a81	1194	1194	3.52	0.00	1194	369.21	p-hat1000-2	2063	2019	0.45	—	—	—
keller4	2166	2166	0.05	0.00	2166	221.53	p-hat1000-3	265	260	0.18	0.00	265	65.79
keller5	4779	4704	0.22	—	—	—	sanr200-0-7	1661	1661	0.12	0.00	1661	1106.80
keller6	10128	9724	9.00	—	—	—	sanr200-0-9	1231	1220	0.10	0.00	1231	80.64
DSJC500-5	1898	1819	0.07	—	—	—	sanr400-0-5	2013	1947	0.17	—	—	—
DSJC1000-5	2353	2195	0.96	—	—	—	sanr400-0-7	1839	1804	0.19	—	—	—
brock200-1	1542	1532	0.11	0.00	1542	688.59	johnson8-2-4	129	129	0.02	0.00	129	0.27
brock200-2	1644	1608	0.14	0.00	1644	3037.92	johnson16-2-4	1477	1477	0.02	0.00	1477	104.88
brock200-3	1681	1680	0.09	0.00	1681	1674.18	johnson32-2-4	5050	5018	0.05	—	—	—
brock200-4	1786	1786	0.06	0.00	1786	1522.89	san200-0-7-1	1911	1906	0.10	0.00	1911	192.06
brock400-1	1767	1742	0.20	—	—	—	san200-0-7-2	2764	2764	0.06	0.00	2764	308.64
brock400-2	1821	1811	0.17	—	—	—	san200-0-9-1	1155	1146	0.06	0.00	1155	94.62
brock400-3	1952	1843	0.13	—	—	—	san200-0-9-2	1362	1362	0.10	0.00	1362	30.71
brock400-4	1813	1800	0.21	—	—	—	san200-0-9-3	1279	1279	0.11	0.00	1279	77.41
brock800-1	2157	2056	0.17	—	—	—	san400-0-5-1	7274	7274	0.17	0.00	7274	20.99
brock800-2	2143	2085	0.29	—	—	—	san400-0-7-1	2666	2666	0.12	0.00	2666	1165.40
brock800-3	2132	2073	0.18	—	—	—	san400-0-7-2	3520	3520	0.18	0.00	3520	1266.08
brock800-4	2184	2104	0.26	—	—	—	san400-0-7-3	4188	4188	0.14	0.00	4188	1828.57
c-fat200-1	399	399	0.02	0.00	399	3.20	san400-0-9-1	1427	1426	0.11	0.00	1427	83.60
c-fat200-2	399	399	0.03	0.00	399	3.36	gen200-p0-9-44	1388	1381	0.09	0.00	1388	117.18
c-fat200-5	399	399	0.10	0.00	399	0.70	gen200-p0-9-55	1278	1271	0.09	0.00	1278	55.71
c-fat500-1	399	399	0.10	0.00	399	23.44	gen400-p0-9-55	2040	2029	0.13	0.00	2040	124.25
c-fat500-2	400	400	0.23	0.00	400	6.17	gen400-p0-9-65	1811	1811	0.08	0.00	1811	167.69
c-fat500-5	399	399	0.16	0.00	399	7.53	gen400-p0-9-75	1620	1534	0.09	0.00	1620	177.69
c-fat500-10	400	400	0.18	0.00	400	1.42							
Instâncias BHOSLIB													
frb30-15-1	5340	5340	0.11	0.00	5340	47.76	frb45-21-4	7728	7728	0.47	0.00	7728	291.71
frb30-15-2	5565	5565	0.11	0.00	5565	32.39	frb45-21-5	7707	7707	0.52	0.00	7707	222.76
frb30-15-3	5565	5565	0.13	0.00	5565	21.98	frb50-23-1	8441	8441	0.94	0.00	8441	378.73
frb30-15-4	5415	5415	0.21	0.00	5415	31.19	frb50-23-2	8441	8441	0.94	0.00	8441	429.18
frb30-15-5	5415	5415	0.11	0.00	5415	33.13	frb50-23-3	8234	8234	0.95	0.00	8234	617.89
frb35-17-1	6222	6222	0.07	0.00	6222	56.15	frb50-23-4	8441	8441	1.00	0.00	8441	338.90
frb35-17-2	6290	6290	0.16	0.00	6290	46.69	frb50-23-5	8441	8441	0.97	0.00	8441	461.89
frb35-17-3	6222	6222	0.16	0.00	6222	57.59	frb53-24-1	9048	9048	1.37	0.00	9048	475.86

Continua na próxima página

frb35-17-4	6069	6069	0.07	0.00	6069	88.53	frb53-24-2	8856	8856	1.25	0.00	8856	682.81
frb35-17-5	6222	6222	0.07	0.00	6222	67.86	frb53-24-3	8856	8856	1.43	0.00	8856	912.74
frb40-19-1	6859	6859	0.16	0.00	6859	190.16	frb53-24-4	9048	9048	1.39	0.00	9048	582.74
frb40-19-2	7049	7049	0.33	0.00	7049	105.66	frb53-24-5	9048	9048	1.39	0.00	9048	443.78
frb40-19-3	7049	7049	0.17	0.00	7049	107.76	frb59-26-1	9594	9594	2.53	0.00	9594	998.25
frb40-19-4	7049	7049	0.33	0.00	7049	97.27	frb59-26-2	9594	9594	2.40	0.00	9594	1007.03
frb40-19-5	7030	7030	0.16	0.00	7030	112.97	frb59-26-3	9438	9438	2.47	0.00	9438	2552.86
frb45-21-1	7707	7707	0.50	0.00	7707	342.83	frb59-26-4	9594	9594	2.49	0.00	9594	1009.55
frb45-21-2	7917	7917	0.51	0.00	7917	167.40	frb59-26-5	9594	9594	2.55	0.00	9594	1172.09
frb45-21-3	7917	7917	0.51	0.00	7917	165.84	frb100-40-1	14440	14440	14.44	—	—	—

Referências

Andrade, D. V., Resende, M. G., e Werneck, R. F. (2012). Fast local search for the maximum independent set problem. *J. Heuristics*, 18(4):525–547.

Feige, U. e Kogan, S. (2004). Hardness of approximation of the balanced complete bipartite subgraph problem. Technical report.

Feo, T. e Resende, M. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133.

Garey, M. R. e Johnson, D. S. (1979). *Computers and intractability : a guide to the theory of NP-completeness*. San Francisco : W.H. Freeman. ISBN 9780716710455.

Li, M., Hao, J.-K., e Wu, Q. (2020). General swap-based multiple neighborhood adaptive search for the maximum balanced biclique problem. *Computers & Operations Research*, 119:104922.

Mladenovic, N. e Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research*, 24(11):1097–1100.

Nogueira, B. C. e. S., Pinheiro, R. G. S., e 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. ISSN 0166-218X.

Resende, M. G. C. e Ribeiro, C. C. (2019). *Greedy Randomized Adaptive Search Procedures: Advances and Extensions*, p. 169–220. Springer International Publishing, Cham. ISBN 978-3-319-91086-4.

Wang, Y., Cai, S., e Yin, M. (2018). New heuristic approaches for maximum balanced biclique problem. *Information Sciences*, 432:362 – 375. ISSN 0020-0255.

Zhou, Y. e Hao, J.-K. (2017). Combining tabu search and graph reduction to solve the maximum balanced biclique problem. *ArXiv*, abs/1705.07339.

Zhou, Y., Rossi, A., e 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.