UNIVERSIDADE FEDERAL FLUMINENSE

ANDRE BARROS CHAGAS DE OLIVEIRA

Heurísticas para o Problema de Quasi-Clique de Cardinalidade Máxima

NITERÓI

2013

UNIVERSIDADE FEDERAL FLUMINENSE

ANDRE BARROS CHAGAS DE OLIVEIRA

Heurísticas para o Problema de Quasi-Clique de Cardinalidade Máxima

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Otimização Combinatória.

Orientador:
Celso Carneiro Ribeiro

Co-orientador:
Alexandre Plastino

NITERÓI

2013

Heurísticas para o Problema de Quasi-Clique de Cardinalidade Máxima

Andre Barros Chagas de Oliveira

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre.

Aprovada por:

Prof. D.Sc. Celso Ribeiro / IC-UFF (Presidente)

Prof. D.Sc. Alexandre Plastino / IC-UFF

Prof. D.Sc. Fábio Protti / IC-UFF

Profa. D.Sc. Lilian Markenzon / UFRJ-NCE

Profa. D.Sc. Isabel Cristina Mello Rosseti / IC-UFF

Niterói, 12 de Abril de 2013.

Aos que farão do sistema educacional uma biblioteca de livros abertos.

Agradecimentos

Primeiramente, agradeço aos meus orientadores, Celso Ribeiro e Alexandre Plastino, pelos ensinamentos, apoio e confiança no meu trabalho. A troca de experiências foi realmente enriquecedora. Essa experiência foi fundamental para meu crescimento como pesquisador.

A meus pais Roberto e Cristina pelo carinho, suporte e compreensão neste período de ausência minha em diversos momentos familiares.

A meus irmãos, Flavia e Pedro, pelo apoio e torcida do seu irmão mais novo na conquista de um sonho.

A minha adorável e tão amável Débora, que me acompanhou durante este período de dedicação a um sonho. Seu companheirismo, carinho, dedicação, incentivo e, principalmente, sua paciência, foram decisivos neste momento de aprendizado. Seu suporte foi decisivo para conclusão dessa etapa. Sou imensamente grato por tudo que me ensinou e ainda me ensina.

A Ruby e Denise que muito me confortaram em momentos de cansaço e dedicação.

Aos meus amigos de faculdade, em especial, Pablo, Igor, Marcos, Julianny, Leonardo, Renatha, Gustavo, Pedro, Marcus e Ed, pelo apoio e ajuda em momentos de dúvidas nas matérias da faculdade.

Aos meus amigos Moulin, Willen, Diego, Mario, Thiago, Guilherme, Flávio, Israel, Turicas, Julinha, Jojo, Vanessa, Marta e Marcelo, pela força e estímulo a seguir em frente.

Agradeço o apoio dos professores da UFF, que colaboraram com suas experiências e feedbacks que agregaram muito a este trabalho. As meninas da secretaria, Teresa e Viviane, pelo carinho, dedicação e socorro em diversos momentos de prazos curtos.

Aos professores da banca, meu agradecimento especial por aceitarem o convite e participarem deste momento tão importante em minha carreira.

Resumo

O Problema de Quasi-Clique de Cardinalidade Máxima (PQM) é obtido por meio da relaxação da restrição de densidade de grafo igual a 1 no Problema de Clique Máximo (PCM), permitindo que grafos não completos sejam encontrados e garantindo uma quantidade relativa de arestas no grafo induzido pelos nós da solução. Na literatura, são encontradas três versões de problemas obtidos com a relaxação de restrições do PCM, visando obter: quasi-cliques de densidade mínima, quasi-cliques de grau mínimo e quasi-cliques de densidade e grau mínimos. Nesta dissertação, será abordado o problema de quasi-clique de densidade mínima. Considerando esse problema, poucas heurísticas eram conhecidas e tratavam de grafos muito grandes, massivos. Nesta dissertação, serão tratados grafos que podem ser armazenados em memória principal, levando ao uso de instâncias menores. Foram implementados e comparados três heurísticas construtivas e, a partir daí, outras heurísticas foram consideradas para melhoria das soluções obtidas. Para encontrar melhores soluções, foram utilizadas heurísticas empregadas em outros tipos de problemas como *Iterated Greedy* (IG) e *Restarted Iterated Greedy* (RIG) sem o uso de buscas locais. Novas heurísticas baseadas em IG e RIG foram propostas com novas estratégias, IG* e RIG*, respectivamente. Além disso, heurísticas híbridas que fazem uso de técnicas de mineração de dados e construção de vocabulário foram propostas em conjunto com RIG. Experimentos computacionais foram realizados em um total de 121 instâncias e os resultados mostram que as heurísticas empregadas — *Iterated Greedy** (IG*), *Restarted Iterated Greedy** (RIG*), *Restarted Iterated Greedy with Data Mining* (DMRIG*) e *Restarted Iterated Greedy with Vocabulary Building* (VBRIG*) — permitiram uma melhoria considerável quando comparada às heurísticas construtivas, principalmente quando se tratam de heurísticas guiadas por padrões frequentes. A melhoria apresentada é relativa tanto a tempo de execução, como qualidade da solução.

Palavras-chave: Problema de Quasi-Clique de Cardinalidade Máxima, Quasi-Clique, Heurísticas, Métodos Construtivos, Heurísticas Reconstrutivas, Heurísticas Híbridas, Mineração de Dados, Construção de Vocabulários

Abstract

The Maximum Cardinality Quasi-Clique Problem (MCQ) is obtained by a relaxation of the constraint of graph density on Maximum Clique Problem (MCP) allowing not complete graphs to be found and ensuring a relative amount of edges in an induced graph by the nodes of the solution. The literature considers three versions of this problem and they deal with the relaxation of different restrictions compared to the MCP. They aim to find quasi-cliques of minimum density, quasi-cliques with a minimal vertex degree and quasi-cliques with minimum density and vertex degree. On this thesis, we will address the minimum density quasi-clique problem. Considering this problem, few heuristics were known and dealt with very large or massive graphs. This dissertation will deal with graphs that can be stored in the main memory, leading to the use of smaller instances. It was developed and compared three constructive heuristics and other heuristics have been considered in order to improve the solutions obtained. In order to find better solutions it was considered heuristics employed in other kinds of problems such as *Iterated Greedy* (IG) and *Restarted Iterated Greedy* (RIG). None of them made use of local search procedures. New heuristics based on IG and RIG were proposed considering new strategies, IG* and RIG*, respectively. In addition, hybrid heuristic techniques which make use of data mining and vocabulary construction have been proposed in conjunction with RIG*. Computational experiments were performed on a total of 121 instances and the results show that the heuristics employed — *Iterated Greedy* (IG*), *Restarted Iterated Greedy* (RIG*), *Restarted Iterated Greedy with Data Mining* (DMRIG*) and *Restarted Iterated Greedy with Vocabulary Building* (VBRIG*) — allowed an improvement when compared to the constructive heuristics, particularly when dealing with heuristics guided by frequent patterns. The improvement is shown on both the runtime, such as the quality of the solution.

Keywords: Maximum Cardinality Quasi-Clique Problem, Quasi-Clique, Heuristics, Constructive Methods, Re-constructive Heuristics, Hybrid Heuristics, Data Mining, Vocabulary Building

Palavras-chave

1. Problema de Quasi-Clique de Cardinalidade Máxima
2, Quasi-Clique
3. Heurísticas
4. Métodos Construtivos
5. Heurísticas Reconstrutivas
6. Heurísticas Híbridas
7. Mineração de Dados
8. Construção de Vocabulários

Glossário

PQM	:	Problema de Quasi-Clique de Cardinalidade Máxima;
PCM	:	Problema de Clique Máximo;
HC1	:	Heurística Construtiva 1;
HC2	:	Heurística Construtiva 2;
HC3	:	Heurística Construtiva 3;
IG	:	Iterated Greedy;
IG+	:	Iterated Greedy +;
IG*	:	Iterated Greedy *;
RIG	:	Restarted Iterated Greedy;
RIG*	:	Restarted Iterated Greedy*;
GRASP	:	Greedy Randomized Adaptive Search Procedure;
DIMACS	:	Center for Discrete Mathematics & Theoretical Computer Science;
BHOSLIB	:	Benchmarks with Hidden Optimum Solutions for Graph Problems;
LRC	:	Lista Restrita de Candidatos;
DMRIG*	:	Heurística Restarted Iterated Greedy with Data Mining;
VBRIG*	:	Heurística Restarted Iterated Greedy with Vocabulary Building;
TTT-Plot	:	Time-To-Target Plot;

Sumário

Lista de Figuras

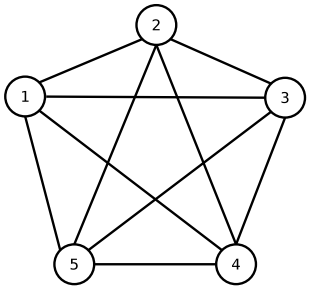
Lista de Tabelas

Capítulo 1

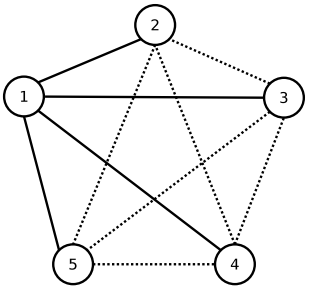
Introdução

1.1 O Problema

Um grafo não direcionado $G = (V(G), E(G))$ consiste em um conjunto finito de vértices $V(G) = \{1, 2, 3, ..., n\}$ e de um conjunto de arestas $E(G) \subseteq V(G) \times V(G)$ formado por pares de elementos de $V(G)$. Um laço em um grafo G é toda aresta que conecte um vértice a ele mesmo. Um grafo G é simples se não possui laços, nem mais de uma aresta ligando o mesmo par de vértices. Um grafo G é completo quando todo par de vértices é unido por uma aresta, ou seja, $(i, j) \in E(G), \forall i, j \in V(G)$. Um grafo H é dito subgrafo de G, o que é representado por $H \subseteq G$, se $V(H) \subseteq V(G)$ e $E(H) \subseteq E(G)$. A Figura 1.1a apresenta um exemplo de grafo simples e completo.



(a) Grafo G simples e completo



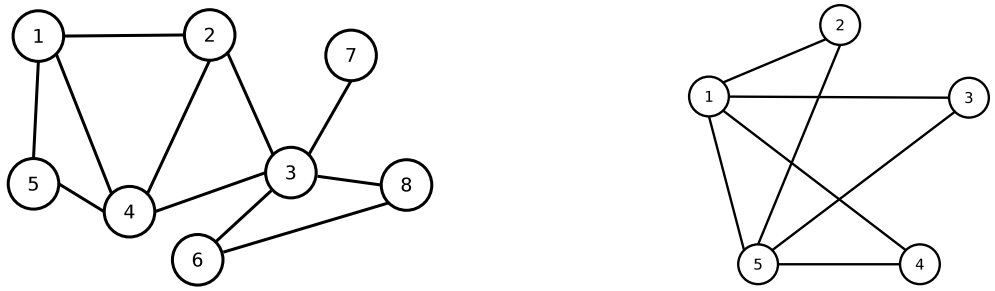
(b) Grau de vértice: $deg_G(1) = 4$

Figura 1.1: Exemplo de grafo simples e completo e grau de vértice.

Seja V' um subconjunto de vértices de $V(G)$. O subgrafo de G com conjunto de vértices V' e conjunto de arestas formado pelas arestas de G que possuem ambas extremidades em V' é chamado de subgrafo induzido em G por V' e é denotado por $G(V')$. O grau de um vértice v em um grafo G é representado por $deg_G(v) = |\mathcal{N}_G(v)|$, onde $\mathcal{N}_G(v) = \{u \in V(G) | (u, v) \in E(G)\}$. A Figura 1.1b apresenta a vizinhança $\mathcal{N}_G(1)$ conec-

tada por arestas contínuas.

Dado um subconjunto S de vértices do grafo G, sua vizinhanca $\mathcal{N}_G(S) = \bigcup_{x \in S} \mathcal{N}_G(x)$ é o conjunto dos vizinhos de todos vértices em S, como ilustra a Figura 1.2a. A densidade de G é determinada pela razão $dens(G) = \frac{|E(G)|}{|V(G)| \times (|V(G)|-1)/2}$, como mostra a Figura 1.2b.



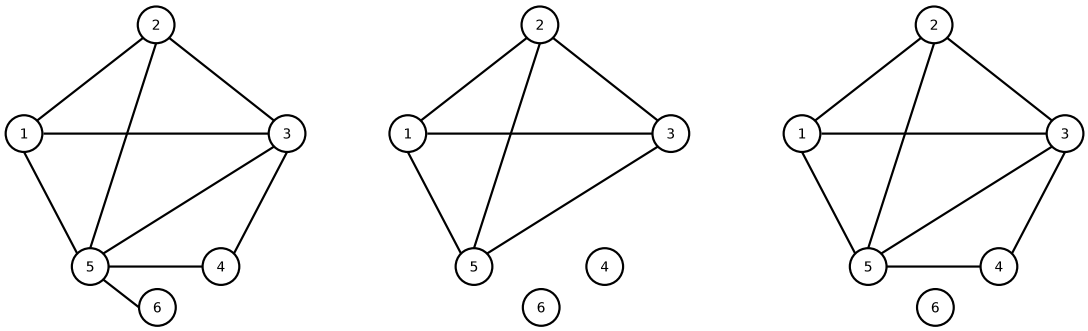
(a) $\mathcal{N}_G(\{2,3,8\}) = \{1,2,3,4,6,7,8\}$ (b) $dens(G) = \frac{7}{5 \times 4/2} = 0,7$

Figura 1.2: Exemplo de vizinhança e densidade de grafo.

Uma clique de G é um subconjunto C de vértices de V tal que o grafo induzido $G(C)$ é completo. Se o grafo G não possuir cliques de maior cardinalidade, então C é dita clique máxima. O tamanho máximo de uma clique em G é chamado de número clique do grafo e é denotado por $\omega(G)$.

O Problema de Clique Máxima (PCM) consiste na busca pelo maior subgrafo completo de um dado grafo [5, 6, 28]. Este problema é NP-difícil [15, 22]. Sendo assim, a menos que $P = NP$, algoritmos exatos para este problema retornam uma solução em tempo que cresce exponencialmente com o número de vértices do grafo.

O Problema de Quasi-Clique de Cardinalidade Máxima (PQM) é uma generalização do PCM que consiste na busca pelo maior subgrafo de um dado grafo G, tal que a densidade seja maior ou igual a um valor mínimo [7, 9, 26], conforme definido a seguir.



(a) Grafo G (b) H = G({1,2,3,5}) (c) I = G({1,2,3,4,5})

Figura 1.3: Grafo G e dois de seus subgrafos H e I.

Um grafo $G = (V(G), E(G))$ é γ-denso se sua densidade for maior ou igual a γ. Um

γ-quasi-clique S é um subconjunto de $V(G)$ tal que o subgrafo induzido por S em G é conexo e γ-denso.

A Tabela 1.1 apresenta exemplos de γ-quasi-cliques com diferentes cardinalidades do grafo G da Figura 1.3a. Observe que o subgrafo induzido por $V' = \{1, 2, 3, 4, 5\}$ em G tem 5 vértices, é conexo e é γ-denso, ou seja, $|E(G)| \geq \gamma\binom{|V(G)|}{2} = 0,8 \times \binom{5}{2} = 8$.

Tabela 1.1: γ-quasi-cliques para o grafo G (Figura 1.3a) com diferentes valores de γ .

γ	Cardinalidade Máxima	Quasi-Clique	Figura
1	4	{1,2,3,5}	1.3b
0,8	5	{1,2,3,4,5}	1.3c
0,6	6	{1,2,3,4,5,6}	1.3a

Em [1, 2], tendo como base uma heurística *GRASP* (metaheurística multi-start que se baseia em duas fases: construção e busca local) [13], estratégias propostas buscam por γ-quasi-cliques de cardinalidade máxima em grafos com uma quantidade de vértices na ordem de 10^5, explorando grafos que não cabem na memória principal disponível. No procedimento de busca por quasi-cliques, critérios de poda são utilizados considerando-se o grau do vértice a ser podado.

Note que quando $\gamma = 1$ tem-se o problema clássico de clique máximo. Consequentemente, o problema de γ-quasi-clique para γ arbitrário é ao menos tão difícil quanto o PCM [2, 8, 10, 20, 25]. Na seção a seguir, será apresentada a formulação de programação inteira para o problema de encontrar subgrafos de cardinalidade máxima cuja densidade é maior ou igual a γ.

1.2 Notação

Seja um grafo não-direcionado $G = (V(G), E(G))$, onde $V(G)$ é o conjunto de vértices e $E(G)$ é o conjunto de arestas $u = (i, j)$, com $i, j \in V(G)$.

Procura-se um γ-quasi-clique S de cardinalidade máxima de G, onde a densidade γ é um parâmetro e está no intervalo $(0, 1]$.

Serão utilizadas as seguintes variáveis:

$$x_i = \begin{cases} 1, \text{ se o vértice } i \in V(G) \text{ é selecionado} \\ 0, \text{ caso contrário} \end{cases}$$

$$y_{ij} = \begin{cases} 1, \text{ se a aresta } (i,j) \in E(G) \text{ faz parte do subgrafo induzido em } G \text{ pelos} \\ \text{vértices } i \text{ e } j. \\ 0, \text{ caso contrário} \end{cases}$$

1.2.1 Clique de cardinalidade máxima

O problema de clique de cardinalidade máxima pode ser formulado como:

$$\max \sum_{i \in V(G)} x_i \tag{1.1}$$

sujeito a:

$$x_i + x_j \leq 1, \qquad \forall 1 \leq i < j \leq |V(G)| : (i,j) \notin E(G) \tag{1.2}$$

$$x_i \in \{0,1\}, \qquad i \in V(G) \tag{1.3}$$

1.2.2 Formulação não-linear

Apresenta-se a seguir uma formulação não-linear original que tratar o problema de quasi-clique máxima como sendo a busca de um subconjunto de vértices de cardinalidade máxima cujo subgrafo por eles induzido em G satisfaz à restrição de densidade mínima, considerando ϵ suficientemente pequeno:

$$\max \sum_{i \in V(G)} x_i - \epsilon \sum_{(i,j) \in E(G)} y_{ij} \tag{1.4}$$

sujeito a:

$$1 + y_{ij} \geq x_i + x_j, \qquad \forall 1 \leq i < j \leq |V(G)| : (i,j) \in E(G) \tag{1.5}$$

$$\sum_{1 \leq i < j \leq |V(G)| : (i,j) \in E(G)} y_{ij} / \sum_{i \in V(G)} \sum_{1 \leq i < j \leq |V(G)|} x_i x_j \geq \gamma \tag{1.6}$$

$$x_i \in \{0,1\}, \qquad i \in V(G) \tag{1.7}$$

$$y_{ij} \in \{0,1\}, \qquad 1 \leq i < j \leq |V(G)| : (i,j) \in E(G). \tag{1.8}$$

A restrição (1.5) garante que se os vértices $i, j \in V(G)$ são selecionados e se existe uma aresta $u = (i,j) \in E(G)$ que os conecta, então ela fará parte do subgrafo gerado pelos vértices selecionados e y_{ij} valerá 1. Caso pelo menos um dos dois vértices $i, j \in V(G)$ não seja selecionado, então a função objetivo (1.4) forçará a variável y_{ij} a valer 0.

A restrição (1.6) garante que a densidade do subgrafo induzido em G pelos vértices selecionados deve ser maior ou igual a γ. O número de arestas no subgrafo é dado por $\sum_{1 \leq i < j \leq |V(G)| : (i,j) \in E(G)} y_{ij}$, enquanto o número máximo possível de arestas é igual a $\sum_{i \in V(G)} \sum_{1 \leq i < j \leq |V(G)|} x_i x_j$. A restrição (1.6) pode ser reescrita como:

$$\sum_{1 \leq i < j \leq |V(G)| : (i,j) \in E(G)} y_{ij} \geq \gamma \sum_{i \in V(G)} \sum_{1 \leq i < j \leq |V(G)|} x_i x_j. \tag{1.9}$$

Como o termo à direita na restrição acima envolve os produtos cruzados de variáveis $\sum_{i \in V(G)} x_i \sum_{j \in V(G) : i \neq j} x_j$, a formulação (1.4-1.5)(1.7-1.9) não é linear.

1.2.3 Linearização do problema de quasi-clique de cardinalidade máxima

A formulação anterior é tomada como base para uma nova formulação na qual as variáveis são linearizadas. Definem-se novas variáveis

$$t_{ij} \geq 0, \quad \forall 1 \leq i < j \leq |V(G)|.$$

Para cada vértice $i \in V(G)$, define-se ainda o conjunto $E(i)$ formado por todas as arestas que possuem i como uma de suas extremidades. Como não pode haver vértices isolados em um quasi-clique de cardinalidade máxima, necessariamente $x_i = 0$ se $y_u = 0, \forall u \in E(i)$. A restrição abaixo pode então ser incluída na formulação não-linear anterior, garantindo que um vértice somente será selecionado se alguma aresta que o tenha como extremidade for considerada:

$$\sum_{1\leq i<j\leq |V(G)|:(i,j)\in E(i)} y_{ij} \geq x_i, \qquad \forall i \in V(G). \tag{1.10}$$

Cada produto $x_i x_j$ na restrição (1.9) pode ser linearizado pelas equações abaixo, para $i, j \in V(G)$:

$$x_i x_j = t_{ij} \tag{1.11}$$

$$t_{ij} \geq 0 \tag{1.12}$$

$$x_i + x_j - 1 \leq t_{ij} \tag{1.13}$$

$$t_{ij} \leq x_i \tag{1.14}$$

$$t_{ij} \leq x_j. \tag{1.15}$$

A restrição (1.9) pode então ser substituída pelo conjunto de restrições abaixo:

$$\sum_{1\leq i<j\leq |V(G)|:(i,j)\in E(G)} y_{ij} \geq \gamma \sum_{i\in V(G)} \sum_{1\leq i<j\leq |V(G)|} t_{ij} \tag{1.16}$$

$$t_{ij} \geq 0, \qquad \forall 1 \leq i < j \leq |V(G)| \tag{1.17}$$

$$x_i + x_j - 1 \leq t_{ij}, \qquad \forall 1 \leq i < j \leq |V(G)| \tag{1.18}$$

$$t_{ij} \leq x_i, \qquad \forall 1 \leq i < j \leq |V(G)| \tag{1.19}$$

$$t_{ij} \leq x_j, \qquad \forall 1 \leq i < j \leq |V(G)|. \tag{1.20}$$

Chega-se então à formulação (1.4-1.5)(1.7-1.8)(1.16-1.20) por programação inteira do problema de determinar o quasi-clique de cardinalidade máxima com restrição de densi-

dade mínima.

1.3 Contribuições e Estrutura da Dissertação

O problema de quasi-clique possui aplicações em diversas áreas que envolvem a representação de informação em grafos, tais como redes sociais, redes de telecomunicações, estudos de biologia computacional, etc. A motivação inicial do estudo se deu a partir do trabalho apresentado em [2] que explora a busca por quasi-clique em a grafos muito grandes. A estrutura de algoritmos apresentados nessa dissertação é representada na Figura 1.4.

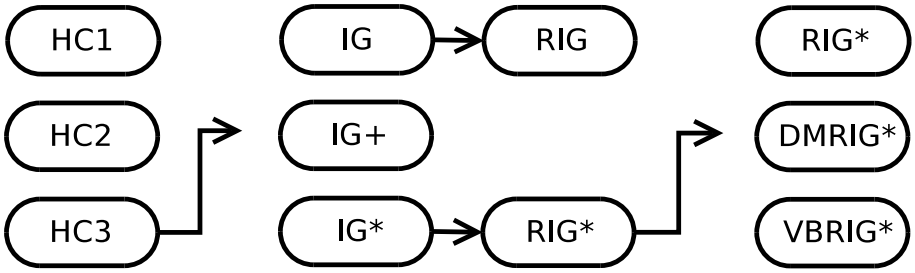


Figura 1.4: Ordem dos algoritmos abordados.

Nesta dissertação, inicialmente são implementadas três heurísticas construtivas para o PQM, apresentadas no Capítulo 2. As duas primeiras heurísticas são propostas nesta dissertação com o intuito de servir de comparação com a heurística *HC3*. A heurística *HC1* procura incorporar vértices à solução considerando a densidade resultante. A heurística *HC2* realiza a seleção de vértices considerando grau dos vértices e a densidade resultante. A heurística *HC3*, que é baseada na heurística construtiva apresentada em [2], revisada na subseção anterior, porém sem a utilização de métodos criados para tratamento de grafos massivos.

Em seguida, abordagens reconstrutivas — realizam a destruição parcial de uma solução e a reconstrução da solução restante — são exploradas, no Capítulo 3, com o intuito de melhorar o desempenho das heurísticas construtivas *HC1*, *HC2* e *HC3*. São elas: *Iterated Greedy* (IG), *Iterated Greedy+*, (IG+) e *Restarted Iterated Greedy* (RIG).

A heurística *Iterated Greedy* (IG) foi inicialmente proposta para problemas de escalonamento em [34]. Após a construção de uma solução inicial, duas etapas são realizadas. A primeira é chamada de *fase destrutiva* e a segunda de *fase construtiva*. Na primeira, o algoritmo destrói, de forma aleatória, parte da solução gerada, para que seja reconstruída na segunda fase, de acordo com uma função gulosa de reconstrução.

Visando uma melhoria na heurística *Iterated Greedy* (IG), em [12], foi proposta uma

alteração na fase destrutiva. Essa passaria a conter um algoritmo mais sofisticado, no lugar de uma simples seleção aleatória. Utilizou-se um método com funções gulosas, tanto na fase de destruição como na fase de reconstrução. Essa nova heurística foi chamada de *Iterated Greedy+* (IG+).

A heurística *Restarted Iterated Greedy* (RIG) foi proposta em [24], iterando a heurística *Iterated Greedy* (IG), até que um determinado critério de parada seja atingido.

Baseadas nas heurísticas IG+ e RIG, duas novas estratégias são propostas nesta dissertação: *Iterated Greedy** (IG*) e *Restarted Iterated Greedy** (RIG*), respectivamente.

A heurística *Iterated Greedy** (IG*) também possui as fases de destruição e reconstrução. Em sua fase de destruição, utiliza não somente uma função gulosa de destruição, como também de um parâmetro β que permite definir, em uma escala de 0 a 1, o quão guloso será o procedimento de destruição, onde 0 corresponde a uma destruição totalmente gulosa e 1 totalmente aleatória. Em sua fase de reconstrução, o método utilizado para reconstruir a solução será o mesmo utilizado pelos construtores da solução inicial. A outra abordagem proposta, *Restarted Iterated Greedy** (RIG*), ativa a heurística *Iterated Greedy** (IG*) dentro de um laço de repetição com um critério de parada. No entanto, é adicionado um procedimento de diversificação que interfere no tamanho da solução a ser destruída pela heurística interna.

Em seguida, no Capítulo 4, são propostas duas novas estratégias *RIG + Data Mining* (DMRIG*) e *RIG + Vocabulary Building* (VBRIG*), que se baseiam em duas técnicas de intensificação que têm sido utilizadas com sucesso em outros trabalhos.

A heurística híbrida proposta, denominada *RIG + Data Mining*, tem como base a heurística híbrida apresentada em [27], que incorpora um processo de mineração de dados a fim de identificar padrões de boas soluções. Uma vez obtidos, esses padrões são utilizados para guiar procedimentos heurísticos na busca por soluções melhores.

De forma similar, o procedimento de construção de vocabulário, apresentado em [17], foi utilizado na heurística híbrida proposta, denominada *RIG + Vocabulary Building*, que considerou um conjunto de boas soluções iniciais para buscar sequências ou partes comuns que são posteriormente usadas na reconstrução de soluções similares ou melhores. No Apêndice A, serão apresentadas as instâncias que serão utilizadas nesta dissertação.

Capítulo 2

Heurísticas Construtivas

Nesse capítulo, serão apresentadas as heurísticas construtivas HC1, HC2 e HC3, assim como os experimentos computacionais realizados para avaliar o desempenho dessas estratégias.

2.1 Heurística Construtiva 1 (HC1)

Em geral, uma heurística construtiva para o Problema de Quasi-Clique de Cardinalidade Máxima (PQM) constrói uma solução acrescentando um vértice a cada iteração, de acordo com um determinado critério de escolha, até que nenhum outro vértice possa ser escolhido ou até que a inclusão do vértice selecionado induza uma densidade menor que a buscada.

Em um algoritmo construtivo puramente guloso, o elemento a ser inserido ao conjunto solução a cada iteração é determinado segundo uma função gulosa. Algoritmos gulosos randomizados também utilizam uma função gulosa para selecionar elementos candidatos, no entanto, contam com um procedimento de seleção aleatória de um entre os diversos elementos candidatos.

Dessa forma, bons elementos, mas não necessariamente os melhores, são selecionados e inseridos na solução. Normalmente, são utilizados dois conjuntos de elementos. O Conjunto dos Elementos Candidatos C é formado pelos elementos que mantém a viabilidade da solução, uma vez adicionados. A Lista Restrita de Candidatos (LRC) é um subconjunto de C formado pelos $p = \lceil max\{1, \alpha \times |C|\} \rceil$ melhores elementos de C de acordo com o valor de uma função gulosa $g : C \rightarrow \mathbb{R}$ onde $\alpha \in [0, 1]$.

Após a definição da LRC, é realizada uma seleção aleatória do candidato a ser incluído na solução, com probabilidade uniforme. Dessa forma, tem-se $\alpha = 0$ para construções

puramente gulosas e $\alpha = 1$ para construções puramente aleatórias. Assim, uma LRC(C, α, $f()$) é uma Lista Restrita de Candidatos dos p primeiros elementos do conjunto C, ordenado pela função gulosa $f()$.

As heurísticas HC1 e HC2 foram criadas com lógicas simples para permitir que um comparativo com HC3 fosse realizado. Nesse comparativo, pretende-se avaliar o quão melhor HC3 é, em comparação às heurísticas menos elaboradas. O Algoritmo 2.1 descreve a Heurística Construtiva 1 (HC1), que é uma heurística gulosa randomizada, proposta por esta dissertação. O algoritmo recebe como parâmetros de entrada os valores de γ, $V(G)$, $E(G)$ e α. Inicialmente, é selecionado um vértice (linha 2) de acordo com uma LRC que considera como função de ordenação o grau de vértice, no grafo G. Esse vértice selecionado é adicionado à solução S (linha 3). Na linha 4, é criado um conjunto C formado por todo vértice $v \in \mathcal{N}(S)$ que quando adicionado a S, mantém a densidade mínima da solução. Enquanto C não for vazio (linha 5), um vértice será selecionado aleatoriamente a partir da LRC de C e será adicionado ao conjunto solução S (linhas 6 e 7). Na linha 8, o conjunto C é atualizado após a remoção do vértice x.

Algorithm 1 Heurística Construtiva 1

entrada: γ, $V(G)$, $E(G)$ e α
saída: S
1: $S \leftarrow \emptyset$
2: Selecionar $x \in LRC(V(G), \alpha, deg_G(x))$
3: $S \leftarrow S \cup \{x\}$
4: $C \leftarrow \{v \in \mathcal{N}(S) | dens(S \cup \{v\}) \geq \gamma\}$
5: **enquanto** $C \neq \emptyset$ **faça**
6: Selecionar $x \in LRC(C, \alpha, dens(S \cup \{x\}))$
7: $S \leftarrow S \cup \{x\}$
8: $C \leftarrow \{v \in \mathcal{N}(S) | dens(S \cup \{v\}) \geq \gamma\}$
9: **fim enquanto**
10: retorne S

2.2 Heurística Construtiva 2 (HC2)

Enquanto a estratégia HC1 considera a densidade do grafo induzido $G(S \cup \{v\})$ dos vizinhos v da solução S no critério de seleção dos vértices a serem incluídos, a estratégia HC2 considera o grau dos vértices em G. Essa alteração visa mudar o foco do algoritmo em escolher vértices que gerem quasi-cliques de maior densidade em prol de vértices que colaborem na geração de um γ-quasi-clique de maior cardinalidade.

O Algoritmo 2.2 descreve a Heurística Construtiva 2 (HC2). O algoritmo recebe como parâmetros de entrada os valores de γ, $V(G)$, $E(G)$ e α. O primeiro vértice é selecionado (linha 2) de acordo com uma LRC que considera como função de ordenação o grau de vértice, no grafo G. Esse vértice é adicionado à solução na linha 3. Na linha 4, um conjunto C de vértices é criado com todo vértice $v \in \mathcal{N}(S)$ que quando adicionado a S, mantém a densidade mínima da solução. Enquanto C não for vazio (linha 5), um vértice será selecionado aleatoriamente a partir da LRC de C e será adicionado ao conjunto solução S (linhas 6 e 7). Na linha 8, o conjunto C é atualizado após a remoção do vértice x.

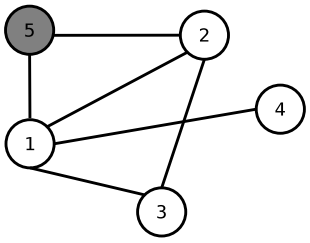
Algorithm 2 Heurística Construtiva 2

entrada: γ, $V(G)$, $E(G)$ e α
saída: S
1: $S \leftarrow \emptyset$
2: Selecionar $x \in LRC(V(G), \alpha, deg_G(x))$
3: $S \leftarrow S \cup \{x\}$
4: $C \leftarrow \{v \in \mathcal{N}(S) | dens(S \cup \{v\}) \geq \gamma\}$
5: **enquanto** $C \neq \emptyset$ **faça**
6: Selecionar $x \in LRC(C, \alpha, deg_G(x))$
7: $S \leftarrow S \cup \{x\}$
8: $C \leftarrow \{v \in \mathcal{N}(S) | dens(S \cup \{v\}) \geq \gamma\}$
9: **fim enquanto**
10: retorne S

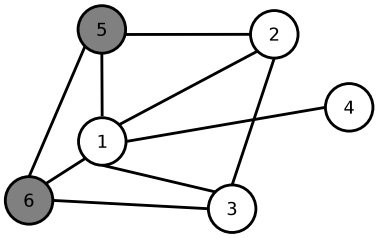
2.3 Heurística Construtiva 3 (HC3)

A Heurística Construtiva 3 (HC3) é uma adaptação da fase de construção da heurística apresentada em [2] sem o tratamento específico para grafos massivos, que não cabem em memória principal, e sem os critérios de poda envolvidos na busca por quasi-cliques nesse tipo de grafo. Utilizam-se os conceitos de γ-vértice, γ-conjunto e $\mathcal{N}_\gamma(S)$. Um γ-vértice de um conjunto S de vértices é um vértice $v \in \overline{S}$ tal que $G(S \cup \{v\})$ é um γ-quasi-clique. Um γ-conjunto de um conjunto S de vértices é um subconjunto T de $\overline{S}$ se $S \cup T$ é um γ-quasi-clique. $\mathcal{N}_\gamma(S)$ é o conjunto de γ-vértices do conjunto S de vértices, ou seja, é o conjunto de vértices vizinhos ao conjunto S que mantém individualmente a densidade γ, se adicionados ao conjunto S. Todo elemento de $\mathcal{N}_\gamma(S)$ é chamado de γ-vizinho. Esses conceitos podem ser observados na Figura 2.1.

A estratégia HC3 seleciona vértices de três maneiras: pelo grau de vértice $deg_G(v)$ em G (linha 2), pelo grau de vértice $deg_S(v)$ na solução (linha 12) e por meio do conceito de



(a) Vértice 5 é um γ-vértice.



(b) {5,6} é um γ-conjunto contido em $\mathcal{N}_\gamma(S)$.

Figura 2.1: Conceitos de γ-vértice, γ-conjunto e $\mathcal{N}_\gamma(S)$, com $S = \{1, 2, 3\}$ e $\gamma = 0,8$.

diferença de potencial (linha 7)[2]. Para todos os casos, uma LRC é gerada e cada um desses critérios é utilizado na função que ordenará os vértices a serem selecionados.

A noção de potencial de um conjunto avalia o número de arestas do grafo induzido pelos vértices da solução corrente, além das arestas necessárias para atingir a densidade mínima. A diferença de potencial de vértice soma o potencial dos γ-vizinhos em relação a um determinado vértice v, no conjunto $\mathcal{N}_\gamma(S)$. O vértice v que maximiza essa diferença é selecionado. Esse procedimento visa medir o efeito provocado na diferença de potencial de outros vértices na vizinhança $\mathcal{N}_\gamma(S)$ na solução corrente S.

Representa-se por γ^* a densidade da solução em construção que deverá se manter maior que a densidade buscada γ, que é inicialmente definida como $\gamma^* = 1$. Conforme a seleção dos vértices é feita, o algoritmo permite que o valor de γ^* seja reduzido até o valor γ de entrada. O algoritmo busca vértices a serem adicionados na solução corrente S^* até que a densidade de S^* seja inferior à pedida. Nesse momento, o algoritmo não considera esse último vértice adicionado e retorna uma solução com densidade satisfatória.

O Algoritmo 2.3 apresenta a heurística HC3 que recebe como parâmetros de entrada os valores de γ, $V(G)$, $E(G)$ e α. Inicialmente, define-se o valor de γ^* (densidade corrente) como 1 (linha 1). Esse valor será gradualmente reduzido conforme a heurística for encontrando vértices, até que seja atingido um valor de γ^* inferior ao valor buscado γ. Na linha 2, utiliza-se a função gulosa $deg_G()$ para definir uma LRC e então selecionar o vértice a ser adicionado à solução corrente S^* (linha 3). A partir dessa adição, enquanto o valor de γ^* não for inferior ao valor de γ (linha 4), o algoritmo busca vértices que estejam na vizinhança $\mathcal{N}_{\gamma*}(S)$, buscando por γ-vértices vizinhos à solução corrente S^* (linhas 5, 6 e 7). Essa busca é realizada por meio de uma LRC, considerando a função gulosa de diferença de potencial.

Caso não encontre algum vértice nessa LRC, o algoritmo faz a busca nos vizinhos da solução, excluídos os já presentes na solução, $\mathcal{N}(S)\backslash S$ (linhas 8 e 9). Se nessa vizinhança

$\mathcal{N}(S)\backslash S$ não for encontrado nenhum vértice, o algoritmo retorna a solução S construída até esse ponto (linha 10). Caso essa vizinhança contenha algum vértice, a seleção é feita utilizando-se uma LRC que utiliza a função $deg_S()$ como função gulosa, na linha 12. Ao final de cada iteração, o algoritmo adiciona o vértice selecionado e recalcula a densidade obtida, linhas 14 e 15. Enquanto a densidade encontrada não for menor que a densidade mínima, o algoritmo voltará a iterar até retornar a solução S construída, nas linhas 10 e 17.

Vale ressaltar que o algoritmo retorna a solução em dois casos: quando não encontrar vértice na vizinhança $\mathcal{N}(S)\backslash S$, ou quando o vértice selecionado tornar a densidade da solução inferior a γ.

Algorithm 3 Heurística Construtiva 3

entrada: γ, $V(G)$, $E(G)$ e α
saída: S
1: $\gamma* \leftarrow 1$
2: Selecionar $x \in LRC(V(G), \alpha, deg_G(x))$
3: $S^* \leftarrow \{x\}$
4: **enquanto** $\gamma* \geq \gamma$ **faça**
5: $S \leftarrow S^*$
6: **se** $(\mathcal{N}_{\gamma*}(S) \neq \emptyset)$ **então**
7: Selecionar $x \in LRC(\mathcal{N}_{\gamma*}(S), \alpha$, diferença de potencial)
8: **senão**
9: **se** $(\mathcal{N}(S)\backslash S = \emptyset)$ **então**
10: retorne S
11: **fim se**
12: Selecionar $x \in LRC(\mathcal{N}(S)\backslash S, \alpha, deg_S(x))$
13: **fim se**
14: $S^* \leftarrow S \cup \{x\}$
15: $\gamma* \leftarrow |E(S^*)|/\binom{|S^*|}{2}$
16: **fim enquanto**
17: retorne S

2.4 Experimentos Computacionais

Nesta seção, serão reportados os experimentos computacionais realizados para avaliar as heurísticas construtivas apresentadas HC1, HC2 e HC3.

2.4.1 Ambiente Computacional

Os experimentos foram realizados em um computador HP Pavilion PC Intel Core 2 Duo T6600 de 2.20GHz com 4,0 GB de Memória RAM, usando Ubuntu 10.04 LTS - Lucid Lynx, Kernel 2.6.32.21, com as execuções limitadas a apenas um processador. Para cada algoritmo, 10 execuções foram realizadas. O código foi implementado em linguagem C++ e compilado com o compilador C/C++ do projeto GNU (GCC) versão 4.4.3.

2.4.2 Instâncias

Não se encontram disponibilizadas instâncias específicas para o Problema de Quasi-clique de Cardinalidade Máxima (PQM). Em função disso, para testar os algoritmos serão utilizadas instâncias para o Problema de Clique Máximo (PCM). Essas instâncias encontram-se divididas em dois grupos: DIMACS e BHOSLIB [29]. Por serem instâncias do PCM, não possuem o parâmetro da densidade mínima. Logo, para cada instância desses conjuntos foi definido um conjunto de instâncias para o PQM com densidades mínimas distintas. Como critério para escolha dessas densidades mínimas, definiu-se que densidades maiores que as densidades dos grafos das instâncias seriam buscadas. Assim, todas instâncias alvo serão maiores que as densidades das instâncias.

2.4.2.1 DIMACS

As instâncias DIMACS foram geradas para a segunda edição do evento *DIMACS Implementation Challenge* [21]. As Tabelas A.1 e A.2 apresentam o nome de cada instância, a quantidade de vértices, a quantidade de arestas, a densidade do grafo, a cardinalidade de clique máximo e a densidade alvo.

2.4.2.2 BHOSLIB

As instâncias BHOSLIB — *Benchmarks with Hidden Optimum Solutions for Graph Problems* — foram geradas a partir de instâncias do Problema de Satisfatibilidade Booleana (SAT) [29].

Mais detalhes são fornecidos na Tabela A.3, que apresenta o nome da instância, a quantidade de vértices, a quantidade de arestas, a densidade do grafo, a cardinalidade de clique máximo e a densidade alvo.

2.4.3 Medidas de Qualidade

Para realizar as comparações entre os algoritmos, foram utilizadas as medidas de qualidade apresentadas em [11, 30, 32]. Essas medidas servirão como base de comparação tanto na fase de ajustes de parâmetros como nos experimentos finais da dissertação.

- *Best* e *#Best*: Para cada instância, *Best* é o valor da melhor solução obtida, tomando como base todas as execuções dos algoritmos sendo comparados. *#Best* é o número de instâncias em que um determinado algoritmo alcançou o valor *Best*. Quanto maior o valor de *#Best*, melhor é o algoritmo.

- NScore e *Score*: Para cada instância, $NScore$ de um algoritmo é a quantidade de algoritmos que encontraram uma solução estritamente melhor do que ele. Para cada algoritmo, *Score* é a soma dos valores de $NScore$ sobre todas as instâncias. Quanto menor o valor de $NScore$, melhor é o algoritmo.

- Dif e MDif: Para cada combinação de algoritmo e instância, Dif é a diferença percentual entre o valor de *Best* da instância e o valor obtido pelo algoritmo. $MDif$ é a média de todos os valores de Dif de um algoritmo, considerando todas as instâncias. Quanto mais próximo de 0 for o MDif, melhor é o algoritmo.

- Tempo: Para cada algoritmo, Tempo é a média dos seus tempos de execução sobre todas as instâncias do problema e são apresentados em segundos.

2.4.4 Ajuste de Parâmetros

Para analisar os algoritmos construtivos, foi necessário realizar o ajuste do parâmetro α, que define o tamanho da Lista Restrita de Candidatos, separadamente para cada algoritmo, de forma que cada um apresentasse o seu melhor desempenho.

Para esta etapa, foram consideradas 24 instâncias a fim de analisar o comportamento dos algoritmos em termos da variação de α. Desse total, 12 instâncias são selecionadas, conforme Tabelas A.1, A.2 e A.3. Outras 12 instâncias são cópias, com o valor de densidade alvo alterada. Isso foi feito para analisar o comportamento das heurísticas em duas situações: (a) quando buscam por γ-quasi-cliques com densidade menor do que a do grafo original e (b) quando buscam por γ-quasi-cliques com densidade maior do que a do grafo original. As instâncias selecionadas são apresentadas na Tabela 2.1.

Tabela 2.1: Instâncias para ajuste de parâmetros das heurísticas construtivas

Instância	|V(G)|	|E(G)|	Densidade (%)	$\omega(G)$	densidade alvo
brock400_4 (a)	400	59765	74,89	33	0,500
brock400_4 (b)	400	59765	74,89	33	0,900
c-fat500-5 (a)	500	23191	18,59	64	0,100
c-fat500-5 (b)	500	23191	18,59	64	0,500
hamming6-4 (a)	64	704	34,92	4	0,200
hamming6-4 (b)	64	704	34,92	4	0,500
johnson8-2-4 (a)	28	210	55,56	4	0,400
johnson8-2-4 (b)	28	210	55,56	4	0,800
keller5 (a)	776	225990	75,15	27	0,600
keller5 (b)	776	225990	75,15	27	0,950
MANN_a45 (a)	1035	533115	99,63	345	0,800
MANN_a45 (b)	1035	533115	99,63	345	0,999
p_hat700-1 (a)	700	60999	24,93	11	0,100
p_hat700-1 (b)	700	60999	24,93	11	0,500
san400_0.5_1 (a)	400	39900	50,00	13	0,300
san400_0.5_1 (b)	400	39900	50,00	13	0,800
C2000.5 (a)	2000	999836	50,02	>= 16	0,300
C2000.5 (b)	2000	999836	50,02	>= 16	0,800
DSJC500.5 (a)	500	62624	50,20	>= 13	0,300
DSJC500.5 (b)	500	62624	50,20	>= 13	0,800
gen400_p0.9_75 (a)	400	71820	90,00	75	0,800
gen400_p0.9_75 (b)	400	71820	90,00	75	0,990
frb35-17-1 (a)	595	148859	84,24	35	0,700
frb35-17-1 (b)	595	148859	84,24	35	0,950

As Tabelas 2.2, 2.3 e 2.4 apresentam os resultados obtidos por cada uma das heurísticas construtivas. Na Tabela 2.2, pode-se notar que a heurística HC1 com $\alpha = 0,05$, apesar de obter um valor de *Score* maior que o algoritmo $\alpha = 0,10$, obteve um valor de $\#Best$ superior a todas as outras execuções desse algoritmo. Seus valores de $MDif$ e $Tempo$ também foram melhores que a maioria dos demais valores de α. Assim sendo, para esse algoritmo foi escolhido $\alpha = 0,05$.

Na Tabela 2.3, referente à heurística HC2, nota-se que todos os valores de α geraram valores muito próximos para as medidas $\#Best$ e $Tempo$. No entanto, o algoritmo com valor de $\alpha = 0,05$ obteve desempenho superior ou igual a todos os demais em todas as medidas. Dessa forma, foi escolhido $\alpha = 0,05$ para a estratégia HC2.

Tabela 2.2: Ajuste de parâmetros de HC1

Medidas	$\alpha = 0,00$	$\alpha = 0,01$	$\alpha = 0,05$	$\alpha = 0,10$	$\alpha = 0,20$
#Best	7	13	**17**	16	15
Score	57	30	**19**	14	19
MDif	0,11	0,07	**0,02**	0,02	0,03
Tempo (s)	0,06	0,05	**0,05**	0,05	0,06

Tabela 2.3: Ajuste de parâmetros de HC2

Medidas	$\alpha = 0,00$	$\alpha = 0,01$	$\alpha = 0,05$	$\alpha = 0,10$	$\alpha = 0,20$
#Best	17	17	**18**	17	16
Score	23	11	**10**	16	18
MDif	0,04	0,02	**0,02**	0,02	0,03
Tempo (s)	0,22	0,23	**0,22**	0,22	0,23

Tabela 2.4: Ajuste de parâmetros de HC3

Medidas	$\alpha = 0,00$	$\alpha = 0,01$	$\alpha = 0,05$	$\alpha = 0,10$	$\alpha = 0,20$
#Best	18	**19**	15	16	14
Score	11	**6**	15	21	33
MDif	0,02	**0,01**	0,03	0,05	0,07
Tempo (s)	0,11	**0,11**	0,10	0,10	0,09

Na Tabela 2.4, referente à heurística HC3, observa-se que as medidas das execuções com o parâmetro $\alpha = 0,01$ foram relativamente melhores que as demais execuções, justificando sua escolha.

2.4.5 Comparação das Heurísticas Construtivas

Os algoritmos construtivos foram comparados considerando-se os parâmetros α ajustados, em todas as instâncias da Tabela 2.1. A heurística HC3 obteve desempenho superior às demais heurísticas, como observado na Tabela 2.5. A heurística HC1, apesar de executar em menos tempo, apresentou desempenho inferior ao obtido pela heurística HC3. De forma similar, a heurística HC2 obteve desempenho um pouco mais próximo do desempenho da heurística HC3, no entanto, apresentou um custo computacional — *Tempo* — mais elevado.

Com os resultados obtidos, pode-se observar que a heurística HC3 atingiu melhores soluções em mais instâncias do que as demais, ao mesmo tempo que obteve um valor de *Score* reduzido. Sendo assim, as heurísticas descritas nos capítulos seguintes serão desenvolvidas tomando por base a heurística construtiva HC3.

Tabela 2.5: Comparativo das heurísticas construtivas

Medidas	HC1 ($\alpha = 0,05$)	HC2 ($\alpha = 0,05$)	HC3 ($\alpha = 0,01$)
#Best	12	17	**22**
Score	23	8	**3**
MDif	0,07	0,03	**0,00**
Tempo (s)	0,05	0,22	**0,11**

Capítulo 3

Heurísticas Reconstrutivas

Neste capítulo, serão descritas as heurísticas reconstrutivas IG, IG+ e RIG para o PQM. Além disso, serão propostas as variantes IG* e RIG*. São apresentados também os experimentos computacionais realizados para avaliar o desempenho dessas estratégias.

3.1 Iterated Greedy

Proposta em [34] para o problema de escalonamento, a heurística *Iterated Greedy* (IG) foi apresentada como uma alternativa às heurísticas comumente utilizadas, como *Simulated Annealing* e *Tabu Search*, entre outras. Trata-se de uma heurística simples e eficiente comparada com as existentes no estado da arte, já tendo sido utilizada com sucesso também para o problema de cobertura em [19, 23]. O método se baseia em dois procedimentos centrais: destruição (fase destrutiva) e reconstrução (fase construtiva). Dada uma solução inicialmente criada por um método construtivo, submete-se a solução a esses dois procedimentos ou fases. Na primeira fase, o algoritmo destrói, de forma aleatória, parte da solução inicial. Com essa solução incompleta, a segunda fase reconstrói uma solução tomando como base uma função gulosa [34, 12]. Essas fases são executadas alternadamente até a satisfação de um critério de parada. No caso das heurísticas *Iterated Greedy*, *Iterated Greedy+* e *Iterated Greedy**, implementadas nesta dissertação, as iterações ocorrem até que não seja mais possível gerar uma solução de tamanho maior que a anteriormente obtida. A heurística inclui a utilização opcional de uma fase de busca local após as duas fases principais.

O Algoritmo 3.1 apresenta a heurística *Iterated Greedy* para o PQM, que recebe como entrada os parâmetros γ, $V(G)$, $E(G)$, S e d. O parâmetro S é a solução inicial e d é a quantidade de elementos que serão removidos da solução inicial. Inicialmente, define-se

Algorithm 4 Iterated Greedy

entrada: γ, $V(G)$, $E(G)$, S e d
saída: $MelhorS$
1: $MelhorS \leftarrow \emptyset$
2: **enquanto** Critério de Parada Não Satisfeito **faça**
3: **para** $k = 1$ to d **faça** {Destruição}
4: Selecionar $v \in S$ aleatoriamente
5: $S \leftarrow S \backslash \{v\}$
6: **fim para**
7: **enquanto** $\exists v \in (V(G) \backslash S)$ tal que $|E(S \cup \{v\})| \geq \gamma\binom{|S|+1}{2}$ **faça** {Reconstrução}
8: Selecionar $v \in (V(G) \backslash S)$ que maximiza a função gulosa $f()$
9: $S \leftarrow S \cup \{v\}$
10: **fim enquanto**
11: **se** Qualidade(S) $>$ Qualidade($MelhorS$) **então**
12: $MelhorS \leftarrow S$
13: **fim se**
14: **fim enquanto**
15: retorne $MelhorS$

o conjunto $MelhorS$ vazio (linha 1). Nas linhas de 3 a 6, ocorre a seleção aleatória dos d elementos para remoção. Em seguida, na fase de reconstrução (linhas 7 a 10), ocorre a seleção dos elementos a serem inseridos nessa solução corrente enquanto existir algum vértice $v \in (V(G) \backslash S)$ tal que $|E(S \cup \{v\})| \geq \gamma\binom{|S|+1}{2}$. Essa seleção ocorre de acordo com a função gulosa $f()$. Uma nova solução é obtida ao final das duas fases.

3.2 Iterated Greedy+

O método *Iterated Greedy+* (IG+) foi proposto como uma melhoria do método *Iterated Greedy* em [12]. Os autores propõem a utilização de uma heurística na fase de remoção (linhas de 3 a 6). Diferente do método previamente proposto, que utilizava uma seleção aleatória, este propõe que na fase de destruição seja utilizada uma função gulosa para guiar a destruição da solução corrente.

Essa alteração permite que as duas fases utilizem funções para avaliar, de forma gulosa, qual os melhores itens a serem removidos e inseridos. O Algoritmo 3.2 apresenta essa alteração na Linha 4. Essa alteração evita, por exemplo, que em uma execução da primeira fase, itens bons sejam removidos. Assim, mantém na solução corrente os candidatos mais promissores para se obter uma solução boa.

Algorithm 5 Iterated Greedy+

entrada: γ, $V(G)$, $E(G)$, S e d
saída: $MelhorS$
1: $MelhorS \leftarrow \emptyset$
2: **enquanto** Critério de Parada Não Satisfeito **faça**
3: **para** $k = 1$ to d **faça** {Destruição}
4: Selecionar $v \in S$ que maximiza função gulosa $f'()$
5: $S \leftarrow S \backslash \{v\}$
6: **fim para**
7: **enquanto** $\exists v \in (V(G) \backslash S)$ tal que $|E(S \cup \{v\})| \geq \gamma\binom{|S|+1}{2}$ **faça** {Reconstrução}
8: Selecionar $v \in (V(G) \backslash S)$ que maximiza função gulosa $f()$
9: $S \leftarrow S \cup \{v\}$
10: **fim enquanto**
11: **se** Qualidade(S) $>$ Qualidade($MelhorS$) **então**
12: $MelhorS \leftarrow S$
13: **fim se**
14: **fim enquanto**
15: retorne $MelhorS$

3.3 Iterated Greedy*

O método *Iterated Greedy** (IG*), proposto nesta dissertação, incorpora alterações que visam um melhor ajuste dos métodos ao problema em questão. Como o PQM possui soluções com tamanhos diferentes, a quantidade de elementos removidos será definida com base em um percentual sobre o tamanho da solução, definido como parâmetro de entrada. Assim, remove-se um percentual δ de elementos de uma solução ao invés de d elementos (linha 3).

A heurística IG* é apresentada no Algoritmo 3.3 e recebe como entrada os parâmetros γ, $V(G)$, $E(G)$, S, δ e β.

Inicialmente, define-se o conjunto $MelhorS$ vazio (linha 1). A função gulosa na fase de destruição do método (linhas de 3 a 6) foi alterada. Uma heurística foi criada para realizar a seleção dos elementos a serem removidos. A ideia utilizada é similar a de uma LRC, onde os candidatos são ordenados de acordo com a função gulosa, e um sorteio decide qual candidato será removido. O parâmetro β define o tamanho da LRC e permite que a função, que escolhe os candidatos a serem removidos, o realize de forma não gulosa. Isso abre a possibilidade de um avaliação do melhor parâmetro β para geração de soluções boas. Esse parâmetro será avalido na parte dos experimentos, ao final do capítulo. Nessa fase de remoção, um percentual δ de vértices é removido da solução S de entrada.

Na segunda fase do método (linhas 7 a 10) foi adotada a mesma função utilizada

pelo algo algoritmo construtivo que gerou a solução inicial para o método, reconstruindo a solução com a mesma ideia dos construtores de solução inicial. Esses construtores utilizam uma LRC para realizar a seleção dos vértices. Nesta dissertação, esse método construtivo utilizou a ideia de seleção dos vértices apresentada na heurística HC3. O parâmetro α, que define o quão guloso ou aleatório será a função na LRC, é mantido e também será utilizado na fase construtiva das soluções.

Como critério de parada (linha 2), utilizou-se a ideia de melhoria contínua, ou seja, uma vez que a solução encontrada após as duas fases do método não apresenta melhor qualidade do que a solução corrente, o laço pode ser finalizado.

Algorithm 6 Iterated Greedy*

entrada: γ, $V(G)$, $E(G)$, S, δ e β
saída: $MelhorS$
1: $MelhorS \leftarrow \emptyset$
2: **enquanto** Critério de Parada Não Satisfeito **faça**
3: **para** $k = 1$ to $(|S| \times \delta)$ **faça** {Destruição}
4: Selecionar $v \in LRC(S, \beta, deg_G(v))$
5: $S \leftarrow S \backslash \{v\}$
6: **fim para**
7: **enquanto** $\exists v \in (V(G) \backslash S)$ tal que $|E(S \cup \{v\})| \geq \gamma\binom{|S|+1}{2}$ **faça** {Reconstrução}
8: Selecionar $v \in LRC(V(G) \backslash S, \alpha, construtivo)$
9: $S \leftarrow S \cup \{v\}$
10: **fim enquanto**
11: **se** Qualidade(S) > Qualidade$(MelhorS)$ **então**
12: $MelhorS \leftarrow S$
13: **fim se**
14: **fim enquanto**
15: retorne $MelhorS$

3.4 Restarted Iterated Greedy

Proposta originalmente para o problema de escalonamento em [24], a heurística *Restarted Iterated Greedy* (RIG) utiliza o método *Iterated Greedy* dentro de um laço. Nesse laço, inicia-se uma nova solução a cada iteração , conforme apresentado no Algoritmo 3.4.

Inicialmente, define-se o conjunto $MelhorS$ vazio (linha 1) . No interior do laço de repetição (linhas 3 a 10), uma solução inicial é construída (linha 4) e então, submetida a execução do método *Iterated Greedy* (IG), na linha 5. Nesta dissertação, a solução inicial foi construída com a heurística HC3. Nesta heurística pode-se incluir uma fase de busca local logo após a execução do método IG, opcionalmente.

No artigo original, o critério de parada se baseava na noção de estabilidade da solução. Ou seja, o processo era parado quando era detectada a estagnação na geração de novas soluções. Como critério de parada para o Algoritmo 3.4 utilizou-se o conceito de janela de execução. A ideia é que o processo pare de ocorrer caso as últimas execuções não produzam nenhuma solução melhor que as anteriores. Sendo assim, o processo se repete até que não seja produzida, dentro de um determinado número de iterações (janela), uma solução que aprimore a melhor solução encontrada. Será definida uma janela de execuções de tamanho W, tal que uma vez executado W iterações sem melhoria, o processo é interrompido.

Algorithm 7 Restarted Iterated Greedy

entrada: γ, $V(G)$, $E(G)$, d e W
saída: $MelhorS$
1: $MelhorS \leftarrow \emptyset$
2: **repetir**
3: Construir S
4: $S = IteratedGreedy(\gamma, V(G), E(G), S, d)$
5: **se** Qualidade(S) $>$ Qualidade($MelhorS$) **então**
6: $MelhorS \leftarrow S$
7: **fim se**
8: **até** Critério de Parada Satisfeito
9: retorne $MelhorS$

3.5 Restarted Iterated Greedy*

A heurística *Restarted Iterated Greedy** (RIG*) equivale à heurística RIG, porém executa internamente a estratégia IG* e não a estratégia IG. Como critério de parada, utilizou-se o mesmo conceito de janela de execução considerado na estratégia RIG.

A heurística RIG*, apresentado no Algoritmo 3.5 recebe como parâmetros de entrada γ, $V(G)$, $E(G)$, δ e W. Na linha 2, deve ser gerado um conjunto P de valores percentuais que serão passados como parâmetros às execuções do método IG* interno, na linha 6. Nesta dissertação, esse conjunto foi gerado com valores próximos ao melhor valor de δ encontrado para o algoritmo IG*, com o intuito de gerar uma variação no percentual da destruição da solução, porém com esse valor próximo ao melhor valor encontrado experimentalmente.

Assim, a cada iteração (linhas 3 a 10), seleciona-se um valor $p \in P$ que será passado como parâmetro à execução de IG* (linha 4). O valor define o percentual de destruição que a solução inicial construída deverá sofrer. Nesse caso, experimentos serão realizados para encontrar um valor de parâmetro de destruição ideal para IG*. Dado esse valor, um

conjunto P de valores considerados para o sorteio será composto por valores próximos desse valor calibrado. O valor β considerado será o encontrado nos testes de ajuste de parâmetros de IG*. Na linha 5, é gerada uma solução de acordo com a heurística construtiva utilizada. Neste algoritmo, a heurística HC3 será utilizada.

Algorithm 8 Restarted Iterated Greedy*

entrada: γ, $V(G)$, $E(G)$, δ e W
saída: $MelhorS$
1: $MelhorS \leftarrow \emptyset$
2: Gerar conjunto P com valores de percentuais
3: **repetir**
4: $\delta \leftarrow p \in P$ aleatóriamente escolhido
5: Gerar S em heurística construtiva
6: $S = IteratedGreedy*(\gamma, V(G), E(G), S, \delta, \beta)$
7: **se** Qualidade(S) $>$ Qualidade($MelhorS$) **então**
8: $MelhorS \leftarrow S$
9: **fim se**
10: **até** Critério de Parada Satisfeito
11: retorne $MelhorS$

3.6 Experimentos Computacionais

Nesta seção, serão reportados os experimentos computacionais para avaliar as heurísticas reconstrutivas IG*, RIG e RIG*.

3.6.1 Ajuste de Parâmetros

Para a etapa de ajuste de parâmetros foram selecionadas 12 instâncias, apresentadas na Tabela 3.1.

Os primeiros experimentos de ajuste de parâmetros foram realizados para definir os valores do parâmetro β e do percentual de destruição de solução (δ) da heurística *Iterated Greedy**. Os valores testados de β são apresentados na primeira coluna da Tabela 3.2 e os valores de percentual de destruição (δ) são apresentados na primeira linha.

Vale ressaltar que IG* com valor $\beta = 1.0$ equivale à heurística IG e com valor $\beta = 0,0$ equivale à heurística IG+.

A Tabela 3.2 mostra que para cada valor de $\beta \in \{0,0; 0,1; 0,5; 1,0\}$ executaram-se os experimentos com variação nos valores de percentual de destruição

Tabela 3.1: Instâncias para ajuste de parâmetros das heurísticas reconstrutivas

Instância	|V(G)|	|E(G)|	Densidade (%)	$\omega(G)$	densidade alvo
brock400_4	400	59765	74,89	33	0,900
c-fat500-5	500	23191	18,59	64	0,500
hamming6-4	64	704	34,92	4	0,500
johnson8-2-4	28	210	55,56	4	0,800
keller5	776	225990	75,15	27	0,950
MANN_a45	1035	533115	99,63	345	0,999
p_hat700-1	700	60999	24,93	11	0,500
san400_0.5_1	400	39900	50,00	13	0,800
C2000.5	2000	999836	50,02	>= 16	0,800
DSJC500.5	500	62624	50,20	>= 13	0,800
gen400_p0.9_75	400	71820	90,00	75	0,990
frb35-17-1	595	148859	84,24	35	0,950

$\delta \in \{0,01; 0,05; 0,10; 0,20; 0,30; 0,40; 0,50; 0,60; 0,70\}$. Para cada linha da tabela, podem ser observados os valores de percentual analisados e suas medidas de qualidade.

Nessa Tabela 3.2, pode ser observado que foi realizada uma tentativa de analisar um grande número de cruzamento de parâmetros (β e δ). A cada valor de β, pode-se observar o comportamento da qualidade das soluções encontradas, avaliando-se as medidas de qualidade. Para cada cada valor de β, um valor de δ foi definido com sendo o melhor para compor o par β, δ.

Uma vez obtidos os valores de δ para cada β, o resultados dessas execuções foram comparados na Tabela 3.3. Os resultados das execuções de cada par β, δ foram comparados. Observa-se que os valores $\beta = 0,5$ e $\delta = 0,10$ se destacam nas duas principais medidas de qualidade — $\#Best$ e $Score$ — e por tal, serão utilizados pelas heurísticas que serão consideradas mais adiante.

3.6.2 Comparação das Heurísticas Reconstrutivas

A comparação entre as heurística RIG e RIG* visa avaliar o método que utiliza o IG* com parâmetros β e δ fixos (RIG) e o método que varia o parâmetro percentual δ (RIG*). Na heurística RIG utilizaram-se os parâmetros obtidos nos testes de ajustes para a chamada ao método IG*. Esse parâmetros são $\beta = 0,5$ e $\delta = 0,10$. A heurística RIG* apresenta o mesmo procedimento, no entanto ao executar a chamada IG*, altera o parâmetro de destruição da solução. Sendo assim, as execuções foram realizadas com valor $\beta = 0,5$ fixo e valores de δ próximos ao encontrado no ajuste de parâmetro, ou seja, $\delta \in \{0,01; 0,05; 0,10; 0,20; 0,30\}$. Para entender o efeito gerado na qualidade das

Tabela 3.2: Ajuste de parâmetros da heurística reconstrutiva IG*

	Medidas / δ	0,01	0,05	0,10	0,20	0,30	0,40	0,50	0,60	0,70
$\beta = 0{,}0$ (IG)	#Best	6	**7**	6	5	6	6	6	5	3
	Score	39	**23**	25	19	12	18	20	26	45
	MDif	0,04	**0,02**	0,03	0,02	0,01	0,02	0,02	0,03	0,11
	Tempo (s)	0,03	**0,04**	0,04	0,04	0,05	0,06	0,07	0,07	0,08
$\beta = 0{,}1$	#Best	5	5	5	**7**	5	5	5	4	5
	Score	42	31	13	**10**	12	20	17	39	46
	MDif	0,04	0,03	0,02	**0,01**	0,02	0,02	0,02	0,03	0,11
	Tempo (s)	0,04	0,04	0,05	**0,05**	0,06	0,06	0,07	0,08	0,09
$\beta = 0{,}5$	#Best	4	6	**9**	9	6	6	7	4	5
	Score	45	30	**10**	13	14	19	17	38	46
	MDif	0,04	0,03	**0,01**	0,01	0,01	0,01	0,01	0,03	0,10
	Tempo (s)	0,04	0,04	**0,05**	0,06	0,06	0,07	0,08	0,08	0,09
$\beta = 1.0$ (IG+)	#Best	5	6	5	7	**8**	5	4	7	5
	Score	45	28	23	28	**9**	28	30	12	40
	MDif	0,07	0,05	0,04	0,03	**0,03**	0,04	0,04	0,03	0,11
	Tempo (s)	0,04	0,04	0,05	0,06	**0,07**	0,07	0,08	0,09	0,10

Tabela 3.3: Comparação da heurística reconstrutiva IG*

Medidas	$\beta(0,0);\delta(0,05)$	$\beta(0,1);\delta(0,20)$	$\beta(0,5);\delta(0,10)$	$\beta(1,0);\delta(0,30)$
#Best	6	7	**8**	6
Score	13	8	**7**	9
MDif	0,03	0,01	**0,02**	0,01
Tempo (s)	0,04	0,05	**0,05**	0,07

soluções, será apresentado um comparativo entre RIG e RIG*. As instâncias serão as mesmas utilizadas nos ajustes de parâmetros anteriormente apresentados.

Como o critério de parada em ambas as heurísticas é o tamanho W da janela, espera-se que quanto maior o valor dessa janela, melhor será a qualidade da solução apresentada, uma vez que permite ao algoritmo um maior tempo de execução e, consequentemente, atingir uma maior fração do espaço de busca. Esse comportamento pode ser observado nas Tabelas 3.4 e 3.5.

Tabela 3.4: Ajuste do parâmetro W da heurística reconstrutiva RIG

Medidas / (W)	50	100	200
#Best	9	11	**12**
Score	5	1	**0**
MDif	0,01	0,00	**0,00**
Tempo (s)	2,23	4,55	**8,41**

Para entender a diferença na qualidade das soluções com as diferentes abordagens,

Tabela 3.5: Ajuste do parâmetro W da heurística reconstrutiva RIG*

Medidas	50	100	200
#Best	9	9	**10**
Score	6	4	**0**
MDif	0,02	0,01	**0,00**
Tempo (s)	2,31	4,43	**9,49**

um comparativo com o parâmetro W de janela fixo foi realizado e apresentado na Tabela 3.6. As heurísticas RIG e RIG* serão comparados e consideram o valor $W = 200$.

A Tabela 3.6 apresenta essa comparação. Pode ser observado que a heurística RIG* melhora a qualidade da solução quando comparada a RIG, apesar de consumir um pouco mais de tempo.

Tabela 3.6: Comparação das heurística reconstrutivas RIG e RIG* com $W = 200$

Medidas	*RIG*	*RIG∗*
#Best	8	**10**
Score	4	**2**
MDif	0,03	**0,00**
Tempo (s)	8,41	**9,49**

Nota-se que o parâmetro β da heurística IG* permite uma destruição que não é totalmente gulosa, nem totalmente aleatória. Assim, observa-se que as execuções que consideram esse parâmetro diferente de 0 ou 1 levam a um melhor resultado, devido ao aumento da perturbação nas soluções geradas. Os parâmetros $\beta = 0,5$ e $\delta = 0,10$ serão utilizados nos próximos testes e heurísticas.

Com a comparação entre as heurísticas RIG e RIG*, percebe-se que a perturbação gerada pela seleção aleatória de um dos parâmetros (δ) melhora a qualidade das soluções encontradas. A heurística RIG*, devido ao seu melhor desempenho, será utilizada na avaliação das heurísticas propostas no próximo capítulo.

Capítulo 4

Heurísticas Híbridas

Neste capítulo, serão descritas e propostas as heurísticas híbridas DMRIG* e VBRIG* para o PQM, considerando como base a heurística RIG* e incorporando, respectivamente, conceitos de mineração de dados e construção de vocabulários. Serão apresentados também os experimentos computacionais realizados para avaliar o desempenhos dessas estratégias.

4.1 GRASP com Mineração de Dados

Uma hibridização da metaheurística *GRASP* (*Greedy Randomized Adaptive Search Procedure*) [14, 13, 31] com técnicas de mineração de dados foi apresentada em [33] para o problema de empacotamento de conjuntos e para o problema das p-Medianas em [27]. A partir de um subconjunto das melhores soluções geradas pelo GRASP, chamado conjunto elite de soluções, o procedimento de mineração de dados identifica padrões dessas boas soluções, que são utilizados para guiar a busca por melhores soluções. Isso levou a buscas mais eficazes no espaço de soluções, com melhorias significativas em questões de tempo e qualidade de solução. Nesta dissertação, essa ideia é utilizada e um procedimento de mineração é inserido no RIG*. De mesmo modo que em [27], neste estudo utilizou-se o algoritmo FPMax* [18] para realizar o processo de mineração de dados. Para execução desse algoritmo, definiu-se um *Suporte Mínimo* de tamanho 2 e foram extraídos 10 padrões desse conjunto.

De maneira geral, essa hibridização coleta boas soluções e as deposita em um conjunto elite. Esse conjunto contém soluções que servem de base para geração de padrões no procedimento de mineração de dados. Uma vez encerrado o primeiro bloco de iterações, responsável pela geração de soluções do conjunto elite, executa-se um procedimento de

mineração para identificar os padrões que guiam as buscas por melhores soluções. No segundo bloco de iterações, esses padrões são utilizados para guiar a busca por melhores soluções de acordo com um procedimento adaptado dos métodos construtivos. Obtém-se então as soluções que voltarão a ser utilizados pela heurística IG* interna.

Algorithm 9 Restarted Iterated Greedy with Data Mining (DMRIG*)

entrada: γ, $V(G)$ e $E(G)$
saída: $MelhorS$
1: $MelhorS \leftarrow \emptyset$
2: Conjunto elite $E \leftarrow \emptyset$
3: Gerar conjunto P com valores de percentuais
4: **repetir**
5: $\delta \leftarrow p \in P$ aleatório
6: Contruir S
7: $S \leftarrow IteratedGreedy * (\gamma, V(G), E(G), S, \delta, \beta)$
8: AtualizaElite(E, S, d)
9: **se** Qualidade(S) $>$ Qualidade($MelhorS$) **então**
10: $MelhorS \leftarrow S$
11: **fim se**
12: **até** Critério de Parada Satisfeito
13: Conjunto de Padrões $Ptts \leftarrow$ Minerar(E)
14: **repetir**
15: $\delta \leftarrow p \in P$ aleatório
16: Buscar S com base em padrão de $Ptts$
17: $S \leftarrow IteratedGreedy * (\gamma, V(G), E(G), S, \delta, \beta)$
18: **se** Qualidade(S) $>$ Qualidade($MelhorS$) **então**
19: $MelhorS \leftarrow S$
20: **fim se**
21: **até** Critério de Parada Satisfeito
22: retorne $MelhorS$

O Algoritmo 4.1 apresenta a hibridização de RIG* com DM. Dois laços principais podem ser observados (linhas de 4 a 12 e de 14 a 21). No primeiro, são geradas as soluções que farão parte do conjunto elite, definido em um tamanho de 10 soluções. Essas soluções são geradas em um laço de repetição igual ao apresentado em RIG* com uma alteração somente: na linha 8, têm-se a coleta e atualização de soluções no conjunto elite. A linha 13 apresenta a chamada do procedimento que minera o conjunto das melhores soluções gerando os padrões que serão utilizados no bloco de iterações seguintes. No segundo laço, é apresentado um procedimento de reconstrução de solução que recebe como entrada um padrão gerado pelo procedimento de mineração, na linha 16. Em comparação ao RIG*, na linha 16, tem-se um procedimento de reconstrução de solução com base nos padrões encontrados. Esse procedimento utiliza a ideia de construção de soluções da heurística HC3. A partir de então, essa solução reconstruída é utilizada no procedimento *Iterated*

*Greedy** interno. Para que essa reconstrução seja possível, o procedimento de construção inicial foi adaptado para gerar soluções a partir de padrões. O algoritmo construtivo adaptado recebe uma solução parcial (padrão) e completa a solução, a partir dessa solução recebida.

4.2 Construção de Vocabulário

Construção de vocabulário é um processo de intensificação baseado na identificação de atributos que aparecem com frequência em boas soluções e criação de novos atributos a partir desses. Essa abordagem é focada na identificação de fragmentos relevantes de soluções para geração de novas soluções. Por meio da combinação dos fragmentos identificados, o procedimento gera novos fragmentos maiores que darão origem a novas soluções. De maneira geral, pode ser entendido como uma análise de texto, onde os vocabulários existentes são analisados para que suas partes frequentes sejam identificadas, possibilitando a geração de novos vocabulários.

Algorithm 10 Restarted Iterated Greedy with Vocabulary Building (VBRIG*)

entrada: γ, $V(G)$ e $E(G)$
saída: $MelhorS$
1: $MelhorS \leftarrow \emptyset$
2: Conjunto de Soluções $CS \leftarrow \emptyset$
3: Gerar conjunto P com valores de percentuais
4: **repetir**
5: $\delta \leftarrow p \in P$ aleatório
6: Construir S
7: $S \leftarrow IteratedGreedy*(\gamma, V(G), E(G), S, \delta, \beta)$
8: Atualizar conjunto elite CS com S
9: **se** Qualidade(S) $>$ Qualidade($MelhorS$) **então**
10: $MelhorS \leftarrow S$
11: **fim se**
12: **até** Critério de parada
13: Gerar Vocabulários $V \leftarrow$ Vocab(CS)
14: **repetir**
15: $\delta \leftarrow p \in P$ aleatório
16: Reconstruir S em heurística construtiva
17: $S \leftarrow IteratedGreedy*(\gamma, V(G), E(G), S, \delta, \beta)$
18: **se** Qualidade(S) $>$ Qualidade($MelhorS$) **então**
19: $MelhorS \leftarrow S$
20: **fim se**
21: **até** Critério de parada
22: retorne $MelhorS$

A construção de vocabulário apresentada nesta dissertação tem como base o procedimento apresentado em [17]. Baseia-se em dois operadores que identificam as interseções de pares de soluções no conjunto elite, também de tamanho 10, e combinam os pares de fragmentos para gerar novas soluções. O primeiro operador $Int(x', x'')$ é aplicado a cada duas soluções do grupo de soluções elite x' e x'' para identificar fragmentos relevantes e armazená-lo em um conjunto Z de fragmentos. O segundo operador $EInt(z', z'')$ é utilizado para combinar dois fragmentos z' e z'' e apresentar uma solução ou parte de uma solução. As soluções obtidas podem estar incompletas, sendo necessário o uso de heurísticas de reconstrução de solução [4, 16, 17]. Nesta proposta, a melhor heurística construtiva, HC3 será utilizada para completar essas soluções parciais.

De maneira geral, a construção de vocabulários é uma abordagem similar ao processo de mineração de dados, no entanto, este permite que padrões possam ser gerados com base em mais de duas soluções e aquele considera somente duas soluções.

4.3 Experimentos Computacionais

Uma vez ajustadas e calibradas, as heurísticas testadas nos capítulos anteriores serviram de base para os experimentos computacionais deste capítulo. A heurística RIG* será comparada com as heurísticas híbridas DMRIG* e VBRIG* e serão consideradas todas as 121 instâncias mencionadas no Apêndice A.

4.3.1 Comparação das Heurísticas Híbridas

Nesta etapa, utilizou-se uma abordagem mais criteriosa para comparar os resultados das heurísticas híbridas. De acordo com o critério de parada do RIG*, cada heurística somente pararia de executar quando as últimas execuções não produzissem nenhuma solução melhor que as anteriores. Sendo assim, dependendo da ordem em que as novas melhores soluções fossem encontradas, uma heurística poderia ser executada mais tempo do que outra. Considerando essa possibilidade, o critério de tempo limite de execução foi adotado. Para definir o tempo limite que cada heurística será executada, a heurística RIG* será executada com o parâmetro de janela $W = 100$ e será coletado o tempo médio em que esta esteve executando para cada instância. Essas execuções podem ser observadas nas Tabelas 4.2, 4.3 e 4.4. Em seguida, as heurísticas RIG*, DMRIG* e VBRIG* foram executadas com esses tempos definidos como tempo limite e os resultados apresentados nas Tabelas 4.5, 4.6 e 4.7.

As heurísticas DMRIG* e VBRIG* possuem duas etapas bem definidas. A primeira trata de buscar soluções utilizando heurísticas construtivas e reconstrutivas para servir de base para os procedimentos de busca de padrões ou de geração de vocabulários. A segunda trata da busca por soluções guiadas por esses padrões ou vocabulários. Assim sendo, para essa etapa de experimentação, definiu-se que 50% do tempo limite é utilizado na primeira etapa e 50% é utilizado na segunda etapa. As Tabelas 4.5, 4.6 e 4.7 apresentam os resultados obtidos por cada heurística com cada uma das 121 instâncias. A Tabela 4.1 apresenta um resultado comparativo final entre as heurísticas RIG*, DMRIG* e VBRIG* seguindo as mesmas medidas utilizadas nos capítulos anteriores.

Nas Tabelas 4.5, 4.6 e 4.7, observa-se o desempenho de cada heurística por instância. A Tabela 4.1 apresenta os dados de forma sumarizada, como realizado nos capítulos anteriores. Nota-se que as heurísticas DMRIG* e VBRIG* apresentam melhorias na quantidade de melhores soluções apresentadas quando comparadas com RIG*. Quando comparadas as heurísticas DMRIG* e VBRIG*, nota-se que a heurística DMRIG* obteve um desempenho superior. Nota-se que DMRIG* atingiu o melhor valor de solução, considerando as três heurísticas comparadas, em 94 instâncias, enquanto VBRIG* atingiu em 85 instâncias.

Tabela 4.1: Medidas de qualidade entre RIG*, DMRIG* e VBRIG*

Medidas	RIG*	DMRIG*	VBRIG*
#Best	45	**94**	85
Score	139	**32**	40
Mdif	0,021	**0,005**	0,007
Tempo	24,28	**24,23**	24,20

Visando-se avaliar o comportamento das heurísticas na busca por um alvo, tomou-se por base a instância frb50-23-3 e dois alvos, um fácil e um difícil, foram definidos com os valores de qualidade de solução 115 e 130, respectivamente. Nas Figuras 4.1 e 4.2, pode-se observar os gráficos *Time To Target* (TTT-Plots) [3] referente a esses alvos. Para este experimento, definiu-se o tempo máximo de execução em 60 segundos. Para as heurísticas DMRIG* e VBRIG*, definiu-se que a etapa de coleta de boas soluções seria realizada até 50% do tempo limite, e somente a partir de então seria realizada a geração dos padrões ou vocabulários seguida da busca por novas soluções com base nesses padrões e vocabulários.

Ao analisar a Figura 4.1, observa-se que, com o alvo definido em 115, nenhuma heurística atingiu a fase de geração de padrões ou vocabulários. Assim, tem-se a curva das três abordagens muito parecidas e praticamente sobrepostas. Isso ocorre por não ter sido necessária a segunda etapa das heurísticas DMRIG* e VBRIG*.

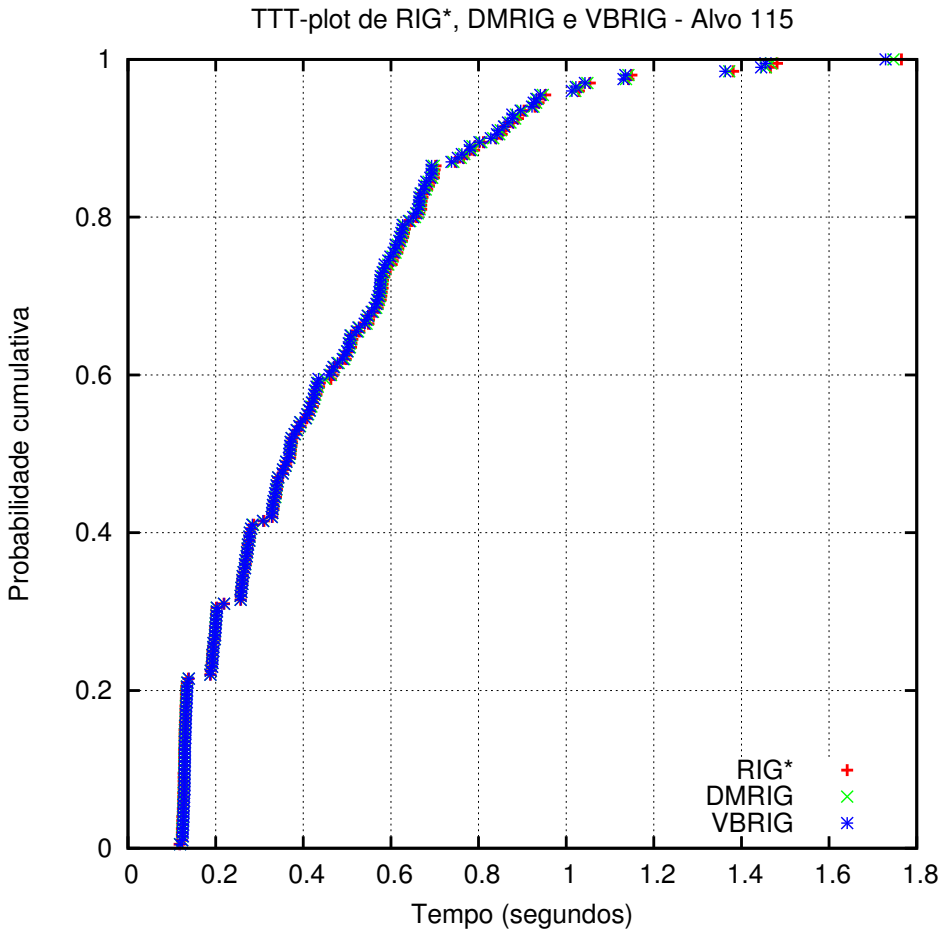


Figura 4.1: TTT-plot com Alvo fácil.

Quando trata-se de um alvo difícil (Figura 4.2), observa-se que após metade do tempo percorrido, as heurísticas apresentam curvas bem diferentes. Na Figura 4.2, observa-se que as heurísticas DMRIG* e VBRIG* retornaram soluções alvo mais rapidamente que a heurística RIG*. Comparativamente, DMRIG* e VBRIG* possuem comportamentos similares, no entanto, a heurística que fez uso de técnicas de mineração de dados se sobressaiu em relação àquela que utilizou construção de vocabulários.

Com o gráfico TTT-Plot, pode-se observar uma relação entre tempo percorrido e probabilidade de encontrar a solução no alvo determinado. No caso da Figura 4.2, pode-se observar que, com 40 segundos de execução, por exemplo, as três heurísticas apresentam probabilidades bem distintas em atingir o alvo definido. A heurística RIG* apresenta uma probabilidade de 40% em atingir o alvo nesse tempo, enquanto as heurísticas DMRIG* e VBRIG* apresentam probabilidades de 95% e 85%, respectivamente.

Com esses resultados, pode-se concluir que as heurísticas apresentadas com técnicas de intensificação auxiliaram a heurística RIG* a atingir melhores resultados mais rapi-

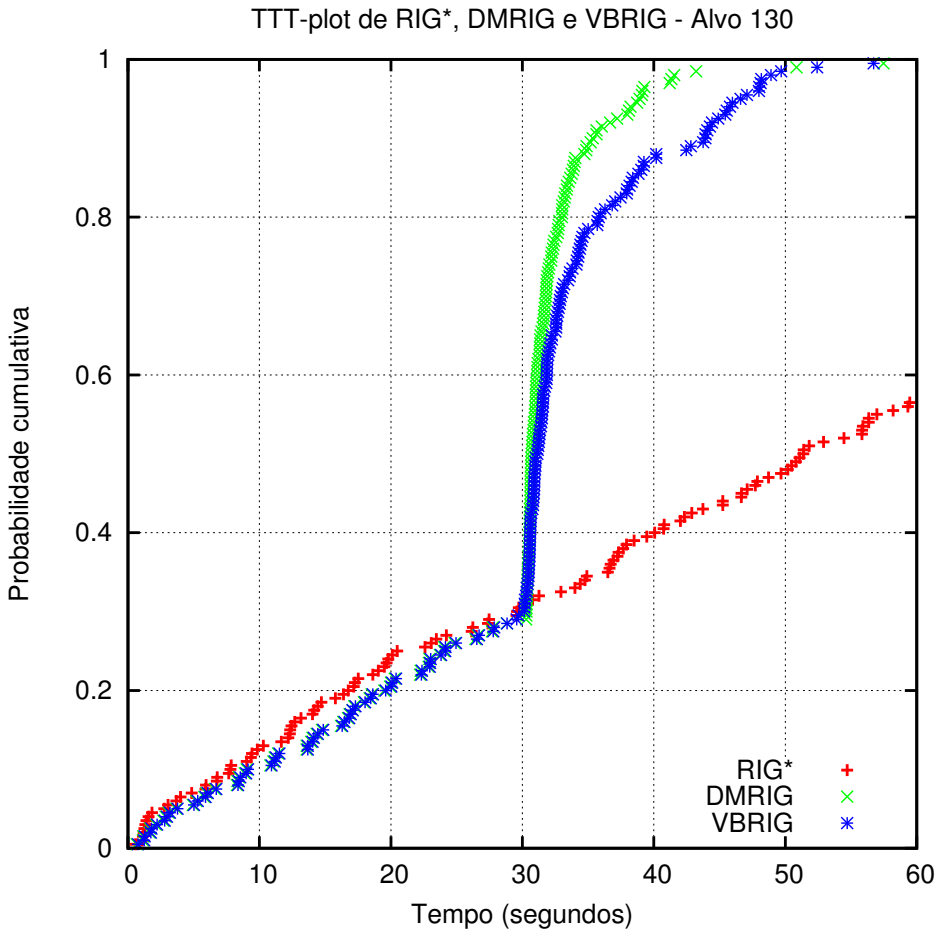


Figura 4.2: TTT-plot com Alvo difícil.

damente. Observou-se também que dentro do mesmo limite de tempo, essas heurísticas encontraram melhores soluções quando comparadas a abordagem sem técnica de intensificação.

Ao analisar os resultados apresentados na Tabela 4.1, observa-se que a estratégia de utilizar técnicas de intensificação no auxílio da busca por melhores soluções do PQM produz bons resultados. Tanto a heurística DMRIG* como a VBRIG* apresentaram melhora significativa sobre os resultados apresentados pela heurística RIG*. É possível avaliar que dado o mesmo tempo de execução para as heurísticas comparadas, aquelas que apresentaram técnicas de intensificação se destacaram. Na Figura 4.3, pode-se observar o desempenho das abordagens em um diagrama de conjuntos. Cada valor representa a quantidade de vezes em que aquela heurística obteve a melhor solução isoladamente. Considera-se que as interseções dos conjuntos representam heurísticas obtendo o melhor valor de qualidade de solução concomitantemente. Assim tem-se que, em 38 instâncias, as três heurísticas encontraram uma solução de mesmo tamanho. As abordagens DMRIG* e VBRIG*, em conjunto, encontraram melhores do que a RIG* em 21 instâncias. Sep-

aradamente avaliadas, DMRIG* encontrou 33 melhores soluções que as demais heurísticas, enquanto VBRIG* encontrou 22 melhores soluções, e RIG* encontrou somente uma solução melhor que DMRIG* e VBRIG*.

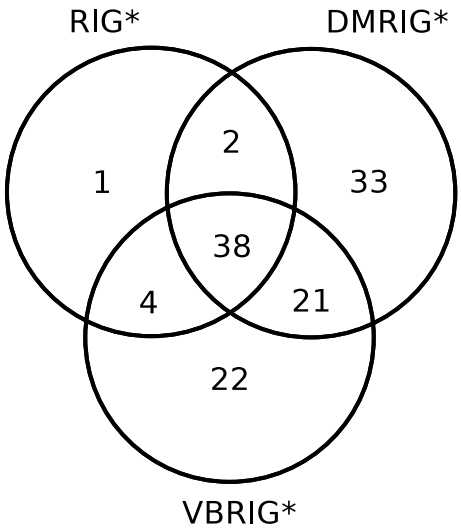


Figura 4.3: Quantidade de melhores soluções encontradas pelas heurísticas híbridas.

Dessa forma, ficou evidenciada a melhoria causada pelo uso de técnicas de intensificação sobre a heurística RIG*. Nesta dissertação, seguiu-se uma ideia de melhoria gradual das abordagens utilizadas na busca por boas soluções para o PQM. A dissertação inicia a investigação ao problema com as heurísticas construtivas HC1, HC2 e HC3. Quando comparadas estas três abordagens, seleciona-se HC3 pelo fato dela ser a melhor heurística. A partir dessa abordagem, consideram-se as heurísticas reconstrutivas IG, IG+ e RIG e implementam-se as heurísticas reconstrutivas IG* e RIG* visando uma melhoria na qualidade das soluções encontradas, sem a utilização de procedimentos de busca local. No intuito de melhorar, ainda mais, as soluções tanto em qualidade, como em questão de tempo de execução, adotam-se heurísticas híbridas DMRIG* e VBRIG*, que se demonstram eficazes, principalmente quando comparadas à heurística RIG*.

Tabela 4.2: RIG*, W = 100 - Instâncias DIMACS parte 1

	Qualidade da solução				Tempo		
Instância	Min.	Máx.	Méd.	Desvio Padrão	Min.	Máx.	Méd.
brock200_1	113,00	113,00	113,00	0,00	1,55	3,10	2,19
brock200_2	22,00	23,00	22,20	0,40	0,27	0,55	0,34
brock200_3	37,00	40,00	38,90	0,70	0,54	1,21	0,85
brock200_4	52,00	54,00	53,30	0,64	0,72	1,57	1,16
brock400_1	185,00	188,00	186,60	0,80	6,81	10,69	8,19
brock400_2	182,00	184,00	183,40	0,66	5,70	12,27	8,45
brock400_3	184,00	185,00	184,70	0,46	5,71	10,17	7,69
brock400_4	184,00	187,00	185,10	0,94	5,91	11,48	7,99
brock800_1	81,00	86,00	83,50	1,63	4,61	8,46	6,07
brock800_2	82,00	87,00	84,20	1,40	4,95	15,83	7,38
brock800_3	82,00	87,00	83,30	1,42	4,57	10,77	8,10
brock800_4	79,00	84,00	82,10	1,30	5,11	9,27	6,89
c-fat200-1	30,00	30,00	30,00	0,00	0,06	0,10	0,07
c-fat200-2	58,00	58,00	58,00	0,00	0,17	0,18	0,18
c-fat200-5	148,00	148,00	148,00	0,00	0,94	0,97	0,95
c-fat500-1	34,00	35,00	34,20	0,40	0,09	0,14	0,10
c-fat500-10	323,00	324,00	323,90	0,30	4,35	7,69	5,33
c-fat500-2	65,00	66,00	65,10	0,30	0,24	0,26	0,25
c-fat500-5	163,00	164,00	163,60	0,49	1,23	2,16	1,52
hamming10-2	525,00	525,00	525,00	0,00	32,79	62,48	44,34
hamming10-4	78,00	80,00	78,30	0,64	4,88	10,45	7,20
hamming6-2	37,00	37,00	37,00	0,00	0,13	0,13	0,13
hamming6-4	32,00	32,00	32,00	0,00	0,12	0,14	0,13
hamming8-2	129,00	129,00	129,00	0,00	1,20	1,75	1,36
hamming8-4	71,00	71,00	71,00	0,00	1,07	1,86	1,32
johnson16-2-4	34,00	34,00	34,00	0,00	0,25	0,26	0,26
johnson32-2-4	21,00	21,00	21,00	0,00	0,62	0,63	0,63
johnson8-2-4	5,00	5,00	5,00	0,00	0,01	0,01	0,01
johnson8-4-4	43,00	43,00	43,00	0,00	0,19	0,36	0,24
keller4	54,00	54,00	54,00	0,00	0,61	0,70	0,67
keller5	486,00	486,00	486,00	0,00	24,87	28,42	26,44
keller6	257,00	267,00	264,00	3,03	75,96	151,50	100,79
MANN_a27	125,00	128,00	126,90	0,70	2,84	5,51	4,07
MANN_a45	416,00	418,00	416,80	0,98	25,00	44,63	28,80
MANN_a81	3294,00	3294,00	3294,00	0,00	472,99	479,74	476,90
MANN_a9	16,00	16,00	16,00	0,00	0,05	0,05	0,05

Tabela 4.3: RIG*, W = 100 - Instâncias DIMACS parte 2

	Qualidade da solução				Tempo		
Instância	Min.	Máx.	Méd.	Desvio Padrão	Min.	Máx.	Méd.
p_hat300-1	61,00	63,00	62,30	0,64	1,31	3,41	2,18
p_hat300-2	114,00	114,00	114,00	0,00	2,26	2,67	2,44
p_hat300-3	69,00	71,00	70,20	0,60	1,68	2,95	2,20
p_hat500-1	94,00	95,00	94,50	0,50	3,87	8,19	5,13
p_hat500-2	211,00	211,00	211,00	0,00	7,12	8,03	7,51
p_hat500-3	123,00	123,00	123,00	0,00	4,76	8,06	6,04
p_hat700-1	116,00	117,00	116,30	0,46	6,83	13,86	9,98
p_hat700-2	288,00	288,00	288,00	0,00	13,08	14,63	13,80
p_hat700-3	174,00	175,00	174,60	0,49	9,64	18,45	12,81
p_hat1000-1	140,00	142,00	141,40	0,66	11,72	25,22	18,23
p_hat1000-2	384,00	384,00	384,00	0,00	25,53	28,79	27,17
p_hat1000-3	208,00	209,00	208,50	0,50	16,49	27,91	21,02
p_hat1500-1	213,00	215,00	214,10	0,54	27,94	54,35	41,31
p_hat1500-2	642,00	642,00	642,00	0,00	64,39	88,02	71,54
p_hat1500-3	349,00	350,00	349,10	0,30	43,05	88,29	54,66
san1000	562,00	562,00	562,00	0,00	25,48	26,43	25,93
san200_0.7_1	55,00	57,00	55,90	0,83	0,58	1,13	0,78
san200_0.7_2	29,00	33,00	30,10	1,70	0,35	0,55	0,40
san200_0.9_1	48,00	68,00	51,90	6,12	0,35	1,11	0,58
san200_0.9_2	38,00	40,00	39,20	0,75	0,29	0,66	0,40
san200_0.9_3	32,00	36,00	33,90	1,14	0,27	0,58	0,39
san400_0.5_1	400,00	400,00	400,00	0,00	6,67	6,87	6,73
san400_0.7_1	201,00	201,00	201,00	0,00	4,22	6,19	4,71
san400_0.7_2	62,00	62,00	62,00	0,00	1,33	1,55	1,43
san400_0.7_3	34,00	34,00	34,00	0,00	0,66	0,85	0,75
san400_0.9_1	54,00	100,00	63,20	13,19	0,65	1,79	1,03
sanr200_0.7	71,00	72,00	71,60	0,49	0,99	1,77	1,29
sanr200_0.9	89,00	91,00	90,20	0,60	1,25	2,24	1,60
sanr400_0.5	28,00	30,00	28,80	0,60	0,69	1,71	1,02
sanr400_0.7	26,00	27,00	26,50	0,50	0,56	1,31	0,75
C125.9	30,00	34,00	31,30	1,35	0,17	0,36	0,24
C250.9	38,00	40,00	38,40	0,66	0,36	0,91	0,54
C500.9	44,00	47,00	46,10	1,14	0,73	1,44	1,00
C1000.9	52,00	56,00	53,70	1,10	1,63	3,62	2,41
C2000.5	34,00	36,00	35,30	0,78	3,64	7,44	4,80
C2000.9	59,00	64,00	61,10	1,58	3,93	8,56	5,50
C4000.5	37,00	39,00	37,50	0,67	7,61	14,23	11,35
DSJC500.5	29,00	30,00	29,20	0,40	0,86	1,73	1,28
DSJC1000.5	32,00	34,00	32,60	0,66	1,86	3,58	2,80
gen200_p0.9_44	35,00	39,00	36,30	1,27	0,33	0,77	0,41
gen200_p0.9_55	35,00	51,00	38,30	4,45	0,29	0,64	0,43
gen400_p0.9_55	42,00	46,00	43,60	1,02	0,62	1,61	1,05
gen400_p0.9_65	42,00	44,00	43,40	0,66	0,64	1,62	0,96
gen400_p0.9_75	44,00	47,00	44,90	1,14	0,62	1,18	0,85

Tabela 4.4: RIG*, W = 100 - Instâncias BHOSLIB

	Qualidade da solução				Tempo		
Instância	Min.	Máx.	Méd.	Desvio Padrão	Min.	Máx.	Méd.
frb30-15-1	52,00	56,00	53,00	1,34	1,43	3,74	2,48
frb30-15-2	50,00	53,00	51,60	0,92	1,53	2,51	1,86
frb30-15-3	50,00	55,00	52,90	1,45	1,39	4,15	2,16
frb30-15-4	50,00	56,00	51,90	1,58	1,35	2,57	1,91
frb30-15-5	51,00	56,00	52,90	1,45	1,62	3,66	2,24
frb35-17-1	66,00	71,00	69,00	1,61	3,05	6,74	4,44
frb35-17-2	66,00	70,00	66,80	1,40	2,58	5,58	3,99
frb35-17-3	71,00	78,00	73,30	1,95	3,10	5,10	4,09
frb35-17-4	69,00	74,00	71,50	1,63	2,89	7,77	4,60
frb35-17-5	66,00	69,00	68,40	1,02	2,66	5,65	3,96
frb40-19-1	96,00	103,00	98,40	2,54	5,29	12,62	8,88
frb40-19-2	87,00	91,00	89,00	1,48	4,41	9,48	7,05
frb40-19-3	82,00	89,00	85,50	2,46	4,07	10,08	7,11
frb40-19-4	83,00	87,00	84,40	1,50	4,67	8,72	7,03
frb40-19-5	84,00	91,00	86,90	2,21	4,55	9,75	6,39
frb45-21-1	106,00	111,00	108,60	1,56	7,21	14,37	9,97
frb45-21-2	105,00	109,00	106,80	1,33	6,88	13,94	9,95
frb45-21-3	108,00	118,00	112,70	3,32	6,91	18,74	10,99
frb45-21-4	111,00	119,00	115,20	2,96	6,78	24,60	9,86
frb45-21-5	104,00	113,00	107,50	2,38	6,78	14,61	9,98
frb50-23-1	136,00	148,00	143,70	3,55	13,10	30,01	18,73
frb50-23-2	140,00	145,00	142,00	1,79	11,87	27,26	18,63
frb50-23-3	123,00	128,00	125,90	1,76	10,44	33,40	18,18
frb50-23-4	137,00	145,00	139,20	2,32	12,43	21,94	16,97
frb50-23-5	141,00	147,00	143,60	1,69	15,31	30,15	22,97
frb53-24-1	180,00	185,00	182,50	1,69	20,05	46,62	33,57
frb53-24-2	150,00	159,00	154,00	2,93	16,28	32,79	24,86
frb53-24-3	160,00	166,00	163,20	2,09	15,43	56,82	28,12
frb53-24-4	160,00	166,00	163,60	1,91	18,65	38,62	27,14
frb53-24-5	149,00	158,00	152,10	3,08	14,25	36,08	22,94
frb56-25-1	207,00	215,00	210,30	2,41	27,79	58,12	35,00
frb56-25-2	191,00	201,00	197,20	3,46	22,76	83,11	38,11
frb56-25-3	178,00	186,00	183,10	2,62	23,94	57,02	38,82
frb56-25-4	172,00	185,00	178,20	3,60	18,73	50,59	32,92
frb56-25-5	180,00	188,00	184,10	2,51	20,67	62,66	37,32
frb59-26-1	218,00	227,00	222,40	2,76	31,18	103,06	56,77
frb59-26-2	216,00	227,00	221,00	3,71	29,40	59,03	41,51
frb59-26-3	223,00	231,00	228,30	2,65	30,40	81,62	51,84
frb59-26-4	208,00	216,00	212,50	2,16	25,41	69,33	38,82
frb59-26-5	201,00	207,00	203,70	2,15	24,38	54,47	40,30
frb100-40	1835,00	1837,00	1835,80	0,60	1003,51	1015,64	1007,18

Tabela 4.5: Resultados RIG*, DMRIG* e VBRIG*, instâncias DIMACS parte 1

	RIG*		DMRIG*		VBRIG*	
Instância	Melhor Solução	Solução Média	Melhor Solução	Solução Média	Melhor Solução	Solução Média
brock200_1	113,00	113,00	**114,00**	113,80	**114,00**	114,00
brock200_2	22,00	22,00	**23,00**	22,50	**23,00**	22,60
brock200_3	**40,00**	39,10	**40,00**	39,50	**40,00**	39,60
brock200_4	54,00	53,00	**55,00**	53,60	**55,00**	54,30
brock400_1	187,00	186,30	188,00	186,50	**189,00**	188,30
brock400_2	**186,00**	184,40	**186,00**	183,40	**186,00**	185,10
brock400_3	185,00	185,00	185,00	184,30	**186,00**	185,90
brock400_4	186,00	185,60	187,00	185,40	**188,00**	187,20
brock800_1	86,00	83,30	**89,00**	86,70	88,00	86,00
brock800_2	88,00	84,80	**89,00**	86,90	**89,00**	86,50
brock800_3	86,00	83,20	**89,00**	86,20	**89,00**	85,50
brock800_4	85,00	83,10	86,00	84,70	**88,00**	84,80
c-fat200-1	**30,00**	30,00	**30,00**	29,90	**30,00**	29,90
c-fat200-2	**58,00**	58,00	**58,00**	58,00	**58,00**	58,00
c-fat200-5	**148,00**	148,00	**148,00**	148,00	**148,00**	148,00
c-fat500-1	**35,00**	34,10	34,00	34,00	**35,00**	34,10
c-fat500-10	**324,00**	324,00	**324,00**	323,80	**324,00**	323,90
c-fat500-2	**66,00**	65,40	**66,00**	65,20	**66,00**	65,30
c-fat500-5	**164,00**	163,50	**164,00**	163,50	**164,00**	163,50
hamming10-2	**525,00**	525,00	**525,00**	525,00	**525,00**	525,00
hamming10-4	79,00	78,50	79,00	78,60	**81,00**	78,80
hamming6-2	**37,00**	37,00	**37,00**	37,00	**37,00**	37,00
hamming6-4	**32,00**	32,00	**32,00**	32,00	**32,00**	32,00
hamming8-2	**129,00**	129,00	**129,00**	129,00	**129,00**	129,00
hamming8-4	**71,00**	71,00	**71,00**	71,00	**71,00**	71,00
johnson16-2-4	**34,00**	34,00	**34,00**	34,00	**34,00**	34,00
johnson32-2-4	**21,00**	21,00	**21,00**	21,00	**21,00**	21,00
johnson8-2-4	**5,00**	5,00	**5,00**	5,00	**5,00**	5,00
johnson8-4-4	**43,00**	42,90	**43,00**	43,00	**43,00**	43,00
keller4	**54,00**	54,00	**54,00**	54,00	**54,00**	54,00
keller5	**486,00**	486,00	**486,00**	486,00	**486,00**	486,00
keller6	270,00	267,00	273,00	268,00	**274,00**	267,60
MANN_a27	127,00	127,00	**128,00**	127,40	**128,00**	127,30
MANN_a45	**418,00**	417,80	**418,00**	416,80	**418,00**	417,00
MANN_a81	**3294,00**	3294,00	**3294,00**	3294,00	**3294,00**	3294,00
MANN_a9	**16,00**	16,00	**16,00**	16,00	**16,00**	16,00

Tabela 4.6: Resultados RIG*, DMRIG* e VBRIG*, instâncias DIMACS parte 2

	RIG*		DMRIG*		VBRIG*	
Instância	Melhor Solução	Solução Média	Melhor Solução	Solução Média	Melhor Solução	Solução Média
p_hat300-1	63,00	62,30	**64,00**	63,30	**64,00**	63,30
p_hat300-2	**114,00**	114,00	**114,00**	114,00	**114,00**	114,00
p_hat300-3	**71,00**	70,20	**71,00**	70,90	**71,00**	70,90
p_hat500-1	**96,00**	94,90	**96,00**	95,60	**96,00**	95,80
p_hat500-2	**211,00**	211,00	**211,00**	211,00	**211,00**	211,00
p_hat500-3	123,00	122,80	**124,00**	123,20	**124,00**	123,30
p_hat700-1	117,00	116,60	**118,00**	117,80	**118,00**	117,70
p_hat700-2	**288,00**	288,00	**288,00**	288,00	**288,00**	288,00
p_hat700-3	**175,00**	174,80	**175,00**	175,00	**175,00**	175,00
p_hat1000-1	142,00	141,60	**143,00**	142,70	**143,00**	142,90
p_hat1000-2	**384,00**	384,00	**384,00**	384,00	**384,00**	384,00
p_hat1000-3	209,00	208,60	209,00	208,80	**210,00**	209,10
p_hat1500-1	215,00	214,00	215,00	213,60	**216,00**	215,40
p_hat1500-2	**642,00**	642,00	**642,00**	642,00	**642,00**	642,00
p_hat1500-3	349,00	349,00	349,00	349,00	**350,00**	349,50
san1000	**562,00**	562,00	**562,00**	562,00	**562,00**	562,00
san200_0.7_1	**57,00**	55,80	**57,00**	56,10	**57,00**	55,80
san200_0.7_2	32,00	29,60	29,00	29,00	**33,00**	30,30
san200_0.9_1	**62,00**	50,30	57,00	49,80	52,00	49,20
san200_0.9_2	40,00	39,30	**41,00**	39,80	**41,00**	40,10
san200_0.9_3	**35,00**	33,70	**35,00**	34,00	**35,00**	34,20
san400_0.5_1	**400,00**	400,00	**400,00**	400,00	**400,00**	400,00
san400_0.7_1	**201,00**	201,00	**201,00**	201,00	**201,00**	201,00
san400_0.7_2	**62,00**	62,00	**62,00**	62,00	**62,00**	62,00
san400_0.7_3	34,00	34,00	**35,00**	34,10	**35,00**	34,10
san400_0.9_1	**100,00**	73,70	**100,00**	86,30	**100,00**	86,50
sanr200_0.7	72,00	71,80	**73,00**	72,50	**73,00**	72,40
sanr200_0.9	91,00	90,30	**92,00**	91,30	**92,00**	91,60
sanr400_0.5	30,00	28,60	**31,00**	30,00	**31,00**	29,70
sanr400_0.7	28,00	26,90	**30,00**	28,00	29,00	27,40
C125.9	32,00	31,20	**34,00**	32,60	**34,00**	33,40
C250.9	40,00	38,50	41,00	39,80	**42,00**	39,80
C500.9	49,00	46,50	**51,00**	47,10	48,00	47,00
C1000.9	**56,00**	53,90	55,00	54,10	**56,00**	54,10
C2000.5	**37,00**	35,70	**37,00**	35,60	36,00	35,20
C2000.9	**64,00**	60,90	62,00	61,00	**64,00**	61,70
C4000.5	38,00	37,40	**39,00**	37,80	38,00	37,30
DSJC500.5	30,00	29,50	31,00	30,30	**34,00**	30,40
DSJC1000.5	34,00	33,10	**35,00**	32,80	34,00	32,50
gen200_p0.9_44	**39,00**	35,90	**39,00**	37,50	**39,00**	36,90
gen200_p0.9_55	42,00	38,40	**52,00**	40,30	51,00	44,70
gen400_p0.9_55	44,00	43,10	**47,00**	44,30	46,00	44,20
gen400_p0.9_65	44,00	43,30	**47,00**	45,10	46,00	44,00
gen400_p0.9_75	**48,00**	45,30	**48,00**	46,10	47,00	45,60

Tabela 4.7: Resultados RIG*, DMRIG* e VBRIG*, instâncias BHOSLIB

	RIG*		DMRIG*		VBRIG*	
Instância	Melhor Solução	Solução Média	Melhor Solução	Solução Média	Melhor Solução	Solução Média
frb30-15-1	56,00	53,40	**57,00**	55,20	56,00	54,10
frb30-15-2	**54,00**	51,10	53,00	52,70	**54,00**	52,80
frb30-15-3	56,00	52,80	**57,00**	55,10	56,00	55,10
frb30-15-4	53,00	51,70	55,00	53,20	**56,00**	52,30
frb30-15-5	55,00	52,50	**57,00**	55,50	**57,00**	54,70
frb35-17-1	72,00	68,40	**74,00**	70,40	73,00	70,00
frb35-17-2	68,00	66,70	**71,00**	68,10	69,00	67,40
frb35-17-3	74,00	73,20	**78,00**	76,10	76,00	74,70
frb35-17-4	72,00	71,10	**76,00**	73,60	75,00	73,40
frb35-17-5	72,00	68,80	**74,00**	71,30	73,00	70,50
frb40-19-1	103,00	99,00	**111,00**	105,50	106,00	103,30
frb40-19-2	92,00	89,90	95,00	93,20	**98,00**	92,90
frb40-19-3	89,00	87,00	**92,00**	89,40	91,00	88,40
frb40-19-4	87,00	84,40	**91,00**	87,30	89,00	86,50
frb40-19-5	90,00	86,20	92,00	91,10	**95,00**	90,30
frb45-21-1	114,00	108,80	**116,00**	113,40	114,00	111,90
frb45-21-2	110,00	106,50	**114,00**	109,60	111,00	107,90
frb45-21-3	115,00	112,50	**120,00**	116,80	117,00	114,50
frb45-21-4	118,00	114,40	**121,00**	117,80	119,00	116,90
frb45-21-5	112,00	107,60	**114,00**	110,90	113,00	110,80
frb50-23-1	147,00	144,20	**154,00**	148,80	149,00	146,10
frb50-23-2	142,00	140,20	**149,00**	146,40	**149,00**	145,90
frb50-23-3	130,00	126,30	**131,00**	129,90	133,00	129,90
frb50-23-4	144,00	140,30	147,00	144,30	**149,00**	143,90
frb50-23-5	148,00	144,40	**154,00**	148,60	152,00	148,50
frb53-24-1	185,00	182,90	**192,00**	190,30	**192,00**	189,30
frb53-24-2	160,00	154,10	**163,00**	160,10	161,00	158,60
frb53-24-3	167,00	164,00	**175,00**	169,60	**175,00**	170,10
frb53-24-4	167,00	162,90	**175,00**	169,70	173,00	168,10
frb53-24-5	157,00	151,50	**161,00**	156,60	159,00	156,20
frb56-25-1	213,00	210,30	222,00	218,70	**223,00**	217,00
frb56-25-2	201,00	197,10	**204,00**	201,80	**204,00**	202,10
frb56-25-3	186,00	182,80	**191,00**	187,00	190,00	186,50
frb56-25-4	182,00	178,50	**188,00**	184,90	185,00	182,80
frb56-25-5	188,00	184,60	**194,00**	190,00	193,00	189,60
frb59-26-1	226,00	222,10	**234,00**	229,40	231,00	229,30
frb59-26-2	227,00	219,50	229,00	225,70	**231,00**	224,60
frb59-26-3	233,00	227,80	236,00	232,90	**238,00**	233,00
frb59-26-4	217,00	211,80	222,00	217,80	**223,00**	219,10
frb59-26-5	209,00	205,00	**214,00**	210,40	212,00	209,20
frb100-40	1836,00	1835,80	1836,00	1835,70	**1838,00**	1837,20

Capítulo 5

Conclusão e Trabalhos Futuros

A presente dissertação abordou o Problema de Quasi-Clique de Cardinalidade Máxima (PQM) que é derivado do Problema de Clique Máximo (PCM). O PQM é uma variação do PCM que considera a densidade do subgrafo menor do que 1. No PCM, buscam-se subgrafos com a densidade mínima igual a 1, enquanto no PQM essa densidade pode ser menor.

Ao realizar o levantamento do problema na literatura, observou-se que este era definido de distintas formas. Alguns trabalhos consideram a definição de quasi-clique de grau mínimo e outros, a definição de quasi-clique de densidade mínima. Cada uma dessas variações apresenta propriedades matemáticas que podem auxiliar na busca por subgrafos. Nesta dissertação, considerou-se a busca por quasi-cliques de densidade mínima.

As heurísticas construtivas HC1 e HC2 foram definidos como algoritmos gulosos randomizados e permitiram um comparativo com a implementação da abordagem construtiva HC3, baseada na heurística apresentada em [2].

Em seguida, foram propostas heurísticas baseadas nos algoritmos *Iterated Greedy* e *Restarted Iterated Greedy* que apresentam uma estrutura de repetidas destruições e reconstruções da solução corrente em busca da melhor solução possível. A abordagem *Iterated Greedy** foi proposta como uma melhoria sobre a heurística *Iterated Greedy* incorporando uma fase reconstrutiva que utiliza, para construção de soluções iniciais, algoritmos implementados nas heurísticas construtivas. A abordagem *Restarted Iterated Greedy** foi proposta tomando como base a heurística *Restarted Iterated Greedy*, porém foi acrescentada a possibilidade de utilização de parâmetros nas chamadas internas às heurísticas *Iterated Greedy**. Essa alteração permitiu uma maior diversificação nas soluções encontradas.

Outra contribuição desta dissertação foi o desenvolvimento de duas heurísticas híbri-

das que identificam padrões ou partes das boas soluções e os utilizam para guiar a busca por novas e melhores soluções. Estas abordagens — *Restarted Iterated Greedy with Data Mining* (DMRIG*) e *Restarted Iterated Greed with Vocabulary Building* (VBRIG*) —, quando comparadas com o mesmo método de busca sem o procedimento de identificação de padrões, apresentaram melhores resultados na busca de uma solução boa. Também foi possível identificar um desempenho superior da heurística que faz uso de técnicas de mineração de dados quando comparada com a abordagem de construção de vocabulários.

Uma continuação natural do estudo realizado nesta dissertação seria a incorporação de busca local atrelada às heurísticas apresentadas, bem como outros métodos de intensificação em uma fase subsequente a fase de reconstrução das heurísticas do gênero *Iterated Greedy* e *Restarted Iterated Greedy*. Um método que poderia ser acrescentado a esta fase subsequente é o procedimento de path relinking.

Uma outra investigação interessante é considerar uma divisão de tempo de execução entre os dois laços principais utilizados na DMRIG* e VBRIG* diferente da realizada: metade do tempo para cada laço. Além disso a execução do processo de mineração de dados e de construção de vocabulários mais de uma vez poderá proporcionar melhores resultados.

Outra oportunidade de melhoria está na criação de uma heurística similar à apresentada em *Restarted Iterated Greedy**, porém apresentando uma variação não só no percentual de exclusão de solução (δ), bem como no tamanho da LRC (β). Esta alteração proporcionaria uma diversificação ainda maior nas soluções encontradas.

Por fim, uma outra modificação interessante seria a criação de uma heurística que utilizasse métodos exatos para subgrafos pequenos que não onerassem o tempo de execução, porém pudessem realizar parte da busca por soluções boas de forma exata. Essa alteração pode apresentar ganhos significativos na qualidade da solução encontrada e poderia ser inserida antes da fase de destruição, onde a solução em questão ainda é reduzida. Poderia ser feita de duas formas: a primeira seria inserida na parte da construção da solução e a outra forma seria no passo da destruição. Ao construir uma nova solução, deveria usar um algoritmo exato para terminar a construção até um tamanho determinado e deveria retomar a heurística a partir desse ponto. Na parte de destruição, deveria destruir frações pequenas para com o restante completar a solução de forma exata.

APÊNDICE A – Descrição das Instâncias

A.1 Instâncias

Neste apêndice, são apresentadas as 121 instâncias utilizadas nos experimentos computacionais realizados nesta dissertação. As Tabelas A.1, A.2 e A.3 apresentam o nome de cada instância, a quantidade de vértices, a quantidade de arestas, a densidade do grafo, a cardinalidade de clique máximo e a densidade alvo.

Instância	|V|	|E|	Densidade (%)	$\omega(G)$	densidade alvo
brock200_1	200	14834	74,54	21	0,80
brock200_2	200	9876	49,63	12	0,80
brock200_3	200	12048	60,54	15	0,80
brock200_4	200	13089	65,77	17	0,80
brock400_1	400	59723	74,84	27	0,80
brock400_2	400	59786	74,92	29	0,80
brock400_3	400	59681	74,79	31	0,80
brock400_4	400	59765	74,89	33	0,80
brock800_1	800	207505	64,93	23	0,80
brock800_2	800	208166	65,13	24	0,80
brock800_3	800	207333	64,87	25	0,80
brock800_4	800	207643	64,97	26	0,80
c-fat200-1	200	1534	7,71	12	0,50
c-fat200-2	200	3235	16,26	24	0,50
c-fat200-5	200	8473	42,58	58	0,50
c-fat500-1	500	4459	3,57	14	0,50
c-fat500-10	500	46627	37,38	126	0,50
c-fat500-2	500	9139	7,33	26	0,50
c-fat500-5	500	23191	18,59	64	0,50
hamming10-2	1024	518656	99,02	512	0,999
hamming10-4	1024	434176	82,89	40	0,95
hamming6-2	64	1824	90,48	32	0,95
hamming6-4	64	704	34,92	4	0,50
hamming8-2	256	31616	96,86	128	0,999
hamming8-4	256	20864	63,92	16	0,80
johnson16-2-4	120	5460	76,47	8	0,80
johnson32-2-4	496	107880	87,88	16	0,95
johnson8-2-4	28	210	55,56	4	0,80
johnson8-4-4	70	1855	76,81	14	0,80
keller4	171	9435	64,91	11	0,80
keller5	776	225990	75,15	27	0,80
keller6	3361	4619898	81,82	>= 59	0,95
MANN_a27	378	70551	99,01	126	0,999
MANN_a45	1035	533115	99,63	345	0,999
MANN_a81	3321	5506380	99,88	>= 1100	0,999
MANN_a9	45	918	92,73	16	0,999

Tabela A.1: Instâncias DIMACS (parte 1)

Instância	|V|	|E|	Densidade (%)	$\omega(G)$	densidade alvo
p_hat300-1	300	10933	24,38	8	0,50
p_hat300-2	300	21928	48,89	25	0,80
p_hat300-3	300	33390	74,45	36	0,95
p_hat500-1	500	31569	25,31	9	0,50
p_hat500-2	500	62946	50,46	36	0,80
p_hat500-3	500	93800	75,19	>= 50	0,95
p_hat700-1	700	60999	24,93	11	0,50
p_hat700-2	700	121728	49,76	>= 44	0,80
p_hat700-3	700	183010	74,80	>= 62	0,95
p_hat1000-1	1000	122253	24,48	>= 10	0,50
p_hat1000-2	1000	244799	49,01	>= 46	0,80
p_hat1000-3	1000	371746	74,42	>= 68	0,95
p_hat1500-1	1500	284923	25,34	>= 12	0,50
p_hat1500-2	1500	568960	50,61	>= 65	0,80
p_hat1500-3	1500	847244	75,36	>= 94	0,95
san1000	1000	250500	50,15	15	0,80
san200_0.7_1	200	13930	70,00	30	0,95
san200_0.7_2	200	13930	70,00	18	0,95
san200_0.9_1	200	17910	90,00	70	0,999
san200_0.9_2	200	17910	90,00	60	0,999
san200_0.9_3	200	17910	90,00	44	0,999
san400_0.5_1	400	39900	50,00	13	0,80
san400_0.7_1	400	55860	70,00	40	0,95
san400_0.7_2	400	55860	70,00	30	0,95
san400_0.7_3	400	55860	70,00	22	0,95
san400_0.9_1	400	71820	90,00	100	0,999
sanr200_0.7	200	13868	69,69	18	0,80
sanr200_0.9	200	17863	89,76	42	0,95
sanr400_0.5	400	39984	50,11	13	0,80
sanr400_0.7	400	55869	70,01	21	0,95
C125.9	125	6963	89,85	>= 34	0,999
C250.9	250	27984	89,91	>= 44	0,999
C500.9	500	112332	90,05	>= 57	0,999
C1000.9	1000	450079	90,11	>= 68	0,999
C2000.5	2000	999836	50,02	>= 16	0,80
C2000.9	2000	1799532	90,02	>= 77	0,999
C4000.5	4000	4000268	50,02	>= 18	0,80
DSJC500.5	500	62624	50,19	>= 13	0,80
DSJC1000.5	1000	249826	50,01	>= 15	0,80
gen200_p0.9_44	200	17910	90,00	44	0,999
gen200_p0.9_55	200	17910	90,00	55	0,999
gen400_p0.9_55	400	71820	90,00	55	0,999
gen400_p0.9_65	400	71820	90,00	65	0,999
gen400_p0.9_75	400	71820	90,00	75	0,999

Tabela A.2: Instâncias DIMACS (parte 2)

Instância	|V|	|E|	Densidade (%)	$\omega(G)$	densidade alvo
frb30-15-1	450	83198	82,35	30	0,95
frb30-15-2	450	83151	82,31	30	0,95
frb30-15-3	450	83216	82,37	30	0,95
frb30-15-4	450	83194	82,35	30	0,95
frb30-15-5	450	83231	82,39	30	0,95
frb35-17-1	595	148859	84,24	35	0,95
frb35-17-2	595	148868	84,24	35	0,95
frb35-17-3	595	148784	84,19	35	0,95
frb35-17-4	595	148873	84,24	35	0,95
frb35-17-5	595	148572	84,07	35	0,95
frb40-19-1	760	247106	85,68	40	0,95
frb40-19-2	760	247157	85,69	40	0,95
frb40-19-3	760	247325	85,75	40	0,95
frb40-19-4	760	246815	85,57	40	0,95
frb40-19-5	760	246801	85,57	40	0,95
frb45-21-1	945	386854	86,73	45	0,95
frb45-21-2	945	387416	86,86	45	0,95
frb45-21-3	945	387795	86,94	45	0,95
frb45-21-4	945	387491	86,87	45	0,95
frb45-21-5	945	387461	86,87	45	0,95
frb50-23-1	1150	580603	87,88	50	0,95
frb50-23-2	1150	579824	87,76	50	0,95
frb50-23-3	1150	579607	87,73	50	0,95
frb50-23-4	1150	580417	87,85	50	0,95
frb50-23-5	1150	580640	87,89	50	0,95
frb53-24-1	1272	714129	88,34	53	0,95
frb53-24-2	1272	714067	88,34	53	0,95
frb53-24-3	1272	714229	88,36	53	0,95
frb53-24-4	1272	714048	88,33	53	0,95
frb53-24-5	1272	714130	88,34	53	0,95
frb56-25-1	1400	869624	88,80	56	0,95
frb56-25-2	1400	869899	88,83	56	0,95
frb56-25-3	1400	869921	88,83	56	0,95
frb56-25-4	1400	869262	88,76	56	0,95
frb56-25-5	1400	869699	88,81	56	0,95
frb59-26-1	1534	1049256	89,24	59	0,95
frb59-26-2	1534	1049648	89,27	59	0,95
frb59-26-3	1534	1049729	89,28	59	0,95
frb59-26-4	1534	1048800	89,20	59	0,95
frb59-26-5	1534	1049829	89,29	59	0,95
frb100-40	4000	7425226	92,84	100	0,95

Tabela A.3: Tabela de instâncias de BHOSLIB

Referências

[1] Abello, J.; Pardalos, P. M.; Resende, M. G. C. On maximum clique problems in very large graphs. Em *External Memory Algorithms* (1999), James M. Abello e Jeffrey Scott Vitter, Ed., American Mathematical Society, pp. 119–130.

[2] Abello, J.; Resende, M. G. C.; Sudarsky, S. Massive quasi-clique detection. Em *Proceedings of the 5th Latin American Symposium on Theoretical Informatics* (2002), J. M. Abello e J. S. Vitter, Eds., Springer-Verlag, pp. 598–612.

[3] Aiex, R. M.; Resende, M. G. C.; Ribeiro, C. C. Tttplots: A perl program to create time-to-target plots. *Optimization Letters 1* (2006), pp. 10–1007.

[4] Aloise, D.; Ribeiro, C. C. Adaptive memory in multistart heuristics for multi-commodity network design. *Journal of Heuristics 17* (2011), pp. 153–179.

[5] Battiti, R.; Mascia, F. Reactive local search for the maximum clique problem: A new implementation. Relatório técnico, Università Degli Studi di Trento, 2007.

[6] Battiti, R.; Protasi, M. Reactive local search for the maximum clique problem. *Algorithmica 29* (2001), pp. 610–637.

[7] Beineke, L. A survey of packings and coverings of graphs. Em *The Many Facets of Graph Theory*, G. Chartrand e S. Kapoor, Eds., vol. 110 de *Lecture Notes in Mathematics*. Springer, 1969, pp. 45–53.

[8] Bhattacharyya, M.; Bandyopadhyay, S. Mining the largest quasi-clique in human protein interactome. Em *Proceedings of the International Conference on Adaptive and Intelligent Systems* (Washington, DC, USA, 2009), IEEE Computer Society, pp. 194–199.

[9] Bomze, I. M.; Budinich, M.; Pardalos, P. M.; Pelillo, M. The Maximum Clique Problem. Em *Handbook of Combinatorial Optimization* (1999), P. M. Pardalos e D.-Z. Du, Eds., vol. 4, pp. 1–74.

[10] Brunato, M.; Hoos, H.; Battiti, R. On effectively finding maximal quasi-cliques in graphs. Em *Learning and Intelligent Optimization* (2008), V. Maniezzo, R. Battiti, e J.-P. Watson, Eds., Springer, pp. 41–55.

[11] de Souza Pessôa, L. *Heurísticas para o problema de k-cobertura de conjuntos.* Tese de Doutorado, Instituto de Computação, Universidade Federal Fluminense, (2009).

[12] Fanjul-Peyro, L.; Ruiz, R. Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research 207* (2010), pp. 55 – 69.

[13] Feo, T. A.; Resende, M. G. C. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters 8* (1989), pp. 67–71.

[14] Feo, T. A.; Resende, M. G. C. Greedy Randomized Adaptive Search Procedures. *Journal of Global Optimization 6* (1995), pp. 109–133.

[15] Garey, M. R.; Johnson, D. S. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., (1990).

[16] Glover, F. Ejection chains, reference structures and alternating path methods for traveling salesman problems. *Discrete Applied Mathematics 65*, 1–3 (1996), pp. 223 – 253.

[17] Glover, F.; Laguna, M. *Tabu Search*. Kluwer Academic Publishers, (1997).

[18] Grahne, G.; Zhu, J. Efficiently using prefix-trees in mining frequent itemsets. CEUR-WS, (2003).

[19] Jacobs, L. W.; Brusco, M. J. A local-search heuristic for large set-covering problems. *Naval Research Logistics NRL 42* (1995), pp. 1129–1140.

[20] Jiang, D.; Pei, J. Mining frequent cross-graph quasi-cliques. *ACM Transactions Knowledge Discovery Data* (2009), pp. 16:1–16:42.

[21] Johnson, D. S.; Trick, M. A., Eds. *Second DIMACS implementation challenge: cliques, coloring and satisfiability*, vol. 26 de *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, (1996).

[22] Karp, R. M. Reducibility among combinatorial problems. *Complexity of Computer Computations 40* (1972), pp. 85–103.

[23] Marchiori, E.; Steenbeek, A. An evolutionary algorithm for large scale set covering problems with application to airline crew scheduling. Em *Real-World Applications of Evolutionary Computing*, S. Cagnoni, Ed., vol. 1803 de *Lecture Notes in Computer Science*. (2000), pp. 370–384.

[24] Minella, G.; Ruiz, R.; Ciavotta, M. Restarted iterated pareto greedy algorithm for multi-objective flowshop scheduling problems. *Computers & Operational Research*, 11 (2011), pp. 1521–1533.

[25] Pardalos, P.; Rebennack, S. Computational challenges with cliques, quasi-cliques and clique partitions in graphs. Em *Experimental Algorithms*, P. Festa, Ed., vol. 6049 de *Lecture Notes in Computer Science*. (2010), pp. 13–22.

[26] Pardalos, P. M.; Xue, J. The maximum clique problem. *Journal of Global Optimization 4* (1994), pp. 301–328.

[27] Plastino, A.; Fonseca, E.; Fuchshuber, R.; Martins, S.; Freitas, A.; Luis, M.; Salhi, S. A hybrid data mining metaheuristic for the p-median problem. Em *Proceedings of the 9th SIAM International Conference on Data Mining* (2009), H. Park, S. Parthasarathy, H. Liu, e Z. Obradovic, Eds., SIAM, pp. 305–316.

[28] Pullan, W.; Hoos, H. H. Dynamic local search for the maximum clique problem. *Journal of Artificial Intelligence Research* (2006), pp. 159–185.

[29] Pullan, W.; Mascia, F.; Brunato, M. Cooperating local search for the maximum clique problem. *Journal of Heuristics 17* (2011), pp. 181–199.

[30] Resende, M. G. C.; Martí, R.; Gallego, M.; Duarte, A. GRASP and path relinking for the max-min diversity problem. *Computers & Operational Research 37* (2010), pp. 498–508.

[31] Resende, M. G. C.; Ribeiro, C. C. Greedy randomized adaptive search procedures. Em *Handbook of Metaheuristics*, F. Glover e G. Kochenberger, Eds., vol. 57. Springer New York, (2003), pp. 219–249.

[32] Ribeiro, C. C.; Uchoa, E.; Werneck, R. F. A hybrid GRASP with perturbations for the steiner problem in graphs. *INFORMS Journal on Computing 14* (2002), pp. 228–246.

[33] Ribeiro, M.; Plastino, A.; Martins, S. Hybridization of GRASP metaheuristic with data mining techniques. *Journal of Mathematical Modelling and Algorithms 5* (2006), pp. 23–41.

[34] Ruiz, R.; Stützle, T. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research 177* (2006), pp. 2033–2049.