UNIVERSIDADE FEDERAL DE OURO PRETO

Algoritmos Exatos e Heurísticos para a resolução do Problema da Descoberta de Cliques de Peso Máximo

Matheus Guedes Vilas Boas
Universidade Federal de Ouro Preto

Orientador: Haroldo Gambini Santos

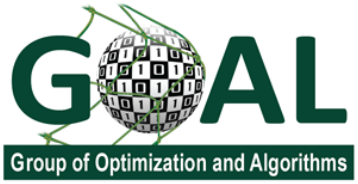
Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Ouro Preto, como parte dos requisitos exigidos para a obtenção do título de Mestre em Ciência da Computação.

Ouro Preto, Junho de 2015

Algoritmos Exatos e Heurísticos para a resolução do Problema da Descoberta de Cliques de Peso Máximo

Matheus Guedes Vilas Boas
Universidade Federal de Ouro Preto

Orientador: Haroldo Gambini Santos



V697a Vilas Boas, Matheus Guedes.
Algoritmos exatos e heurísticos para a resolução do problema da descoberta de cliques de peso máximo [manuscrito] / Matheus Guedes Vilas Boas. - 2015.
96f.: il.: tabs.

Orientador: Prof. Dr. Haroldo Gambini Santos.

Dissertação (Mestrado) - Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Biológicas. Departamento de Computação. Programa de Pós-Graduação em Ciência da Computação.
Área de Concentração: Ciência da Computação.

1. Otimização combinatoria. 2. Algoritmos de computador. 3. Programação heuristica. I. Santos, Haroldo Gambini. II. Universidade Federal de Ouro Preto. III. Titulo.

CDU: 004.023

Catalogação: www.sisbin.ufop.br

Ata da Defesa Pública de Dissertação de Mestrado

Aos 06 dias do mês de julho de 2015, às 10 horas na Sala de Seminários do DECOM no Instituto de Ciências Exatas e Biológicas (ICEB), reuniram-se os membros da banca examinadora composta pelos professores: **Prof. Dr. Haroldo Gambini Santos (presidente e orientador), Prof. Dr. Anderson Almeida Ferreira, Profa. Dra. Isabel Cristina Mello Rosseti e MSc. Samuel Souza Brito**, aprovada pelo Colegiado do Programa de Pós-Graduação em Ciência da Computação, a fim de arguirem o mestrando **Matheus Guedes Vilas Boas**, com o título **"Algoritmos Exatos e Heurísticos para a Resolução do Problema da Descoberta de Cliques de Peso Máximo"**. Aberta a sessão pelo presidente, coube ao candidato, na forma regimental, expor o tema de sua dissertação, dentro do tempo regulamentar, sendo em seguida questionado pelos membros da banca examinadora, tendo dado as explicações que foram necessárias.

Recomendações da Banca:

[X] Aprovada sem recomendações

[ ] Reprovada

[ ] Aprovada com recomendações: _____

Banca Examinadora:

Prof. Dr. Haroldo Gambini Santos

Prof. Dr. Anderson Almeida Ferreira

Profa. Dra. Isabel Cristina Mello Rosseti

MSc. Samuel Souza Brito

Prof. Dr. Luiz Henrique de Campos Merschmann
Coordenador do Programa de Pós-Graduação em Ciência da Computação
DECOM/ICEB/UFOP

Ouro Preto, 06 de julho de 2015.

A Deus

À minha família

Aos meus pais, Luiz e Inês

Ao meu irmão Lucas

À minha noiva Laís

Resumo

O presente trabalho trata do projeto, implementação e avaliação de algoritmos exatos e heurísticos, sequenciais e paralelos, para a resolução do problema da enumeração de cliques com peso acima de um limiar (PECPL). Esse problema considera um grafo com vértices ponderados, onde o objetivo é encontrar todos os cliques maximais com peso acima de um limiar. Os algoritmos estudados neste trabalho são aplicados na separação de cortes no contexto de Programação Inteira. Encontrar todos os cliques acima de um dado peso é equivalente ao problema de encontrar todas as desigualdades violadas de clique. Foram desenvolvidas adaptações em algoritmos conhecidos na literatura, para a resolução do problema. Para o algoritmo de Bron-Kerbosch, uma adaptação foi realizada para resolver o PECPL. Além disso, várias melhorias foram propostas a fim de melhorar a eficiência na resolução das instâncias do problema. Foram propostas uma versão iterativa do algoritmo, originalmente recursivo, e uma versão paralela. O algoritmo de Ostergard e a heurística busca tabu com multi-vizinhanças também foram implementados e modificados para refletir o problema abordado no presente trabalho. Por fim, a metaheurística *Simulated Annealing* foi proposta e desenvolvida utilizando-se das mesmas estruturas de vizinhança utilizadas na heurística busca tabu com multi-vizinhanças. A diferença das duas técnicas está na estratégia de resolução do problema: enquanto a primeira utiliza-se do conceito de lista tabu, a última simula o processo de recozimento de metais. Nos experimentos computacionais, foram utilizadas 7292 instâncias, oriundas de quatro conjuntos referentes à separação de cortes em problemas formulados por meio do uso de programação inteira. Os experimentos foram conduzidos em duas partes: em um primeiro momento, as instâncias foram utilizadas para resolução do PECPL. Posteriormente, o foco foi a resolução do problema do clique de peso máximo (PCPM). Quanto à resolução do PECPL, os resultados obtidos comprovam a eficiência do algoritmo de Bron-Kerbosch, quando comparado aos demais algoritmos, ao encontrar a solução ótima para todas as instâncias e em um tempo consideravelmente menor do

que as outras técnicas. Quando a análise dos resultados foi direcionada à resolução do PCPM, todas as técnicas implementadas obtiveram bons resultados, com destaque para a heurística busca tabu com multi-vizinhanças, a qual resolveu todas as instâncias de forma ótima, com o menor tempo computacional em relação às demais abordagens. Como trabalhos futuros, são sugeridos a adoção de operadores lógicos para a representação do grafo no algoritmo de Bron-Kerbosch, a melhoria da versão paralela do algoritmo e o estudo do projeto das metaheurísticas *Simulated Annealing* e busca tabu.

Palavras-chave: problema da descoberta de cliques de peso máximo, algoritmos exatos, metaheurística busca tabu, metaheurística *simulated annealing*, paralelismo, separação de cortes, programação inteira.

Abstract

This work deals with the design, implementation and evaluation of exact and heuristic algorithms, sequential and parallel to the resolution of clique enumeration problem with weight above a threshold (PECPL). This problem considers a graph with weighted vertices, where the goal is to find all maximal cliques with weight above a threshold. The algorithms studied in this work are applied in the separation cuts in the context of Integer Programming. Find all clique above a certain weight is equivalent to the problem of finding all the inequalities violated clique. Adaptations were developed algorithms known in the literature, to solve the problem. For the Bron-Kerbosch algorithm, an adaptation was made to solve the PECPL. In addition, several improvements were proposed in order to improve efficiency in the resolution of problem instances. It has been proposed an iterative version of the algorithm, recursive originally, and a parallel version. The Ostergard algorithm and multi-neighborhoods tabu search heuristic were also implemented and modified to reflect the problem addressed in this paper. Finally, the Simulated Annealing metaheuristic was proposed and developed using the same neighborhood structures used in multi-neighborhoods tabu search heuristic. The difference of the two techniques is in solving strategy problem: while the first is used the concept of tabu list, the last simulates the process of annealing of metals. In the computational experiments, we used 7292 instances, belonging to four sets related to the separation cuts in problems formulated by using integer programming. The experiments were conducted in two parts: at first, the instances were used for solving the PECPL. Later, the focus was on resolving the maximum weight clique problem (PCPM). As for the resolution of the PECPL, the results prove the efficiency of Bron-Kerbosch algorithm, when compared to other algorithms to find the optimal solution for all instances and in a considerably shorter time than the other techniques. When analyzing the results was directed to resolving the PCPM, all techniques implemented performed well, particularly the multi-neighborhoods tabu search heuristic, which solved all instances optimally with

less computational time compared to other approaches. As future work, it is suggested the adoption of logical operators for the representation of the graph in Bron-Kerbosch algorithm, improved parallel version of the algorithm and the study design of simulated annealing and tabu search metaheuristics.

Keywords: clique discovery problem maximum weight, exact algorithms, tabu search metaheuristic, simulated annealing metaheuristic, parallelism, cuts separation, integer programming.

Declaração

Esta dissertação é resultado de meu próprio trabalho, exceto onde referência explicitatória é feita ao trabalho de outros, e não foi submetida para outra qualificação nesta, nem em outra universidade.

Matheus Guedes Vilas Boas

Agradecimentos

Agradeço a Deus, por sempre ter me dado força para não desistir e principalmente, por me proporcionar o prazer de viver!

Agradeço ao professor Dr. Haroldo Gambini Santos, orientador do corrente trabalho, pela dedicação, compromisso e principalmente pela confiança depositada em mim.

Agradeço ao meu amigo Samuel Souza Brito, "coorientador deste trabalho", pela ajuda e disponibilidade sempre presentes durante o desenvolvimento do projeto.

Agradeço aos professores Dr. Anderson Almeida Ferreira e Dr. Gustavo Peixoto Silva pelos ensinamentos dados durante o período de realização deste trabalho.

Agradeço a todos os outros professores que contribuíram para a minha formação, sempre com o prazer e dedicação em ensinar.

Agradeço à minha mãe Inês Alves Guedes e meu pai Luiz Carlos Vilas Boas, pelo amor, carinho, apoio e pelos ensinamentos que me foram dados.

Agradeço ao meu irmão Lucas Guedes Vilas Boas, por toda a ajuda oferecida não só nesse trabalho, como em todo o mestrado. Além da ajuda, agradeço por sempre acreditar no meu sucesso.

Agradeço à minha noiva Laís Soares Caldeira, pelo amor, carinho e cumplicidade sempre presentes.

Agradeço a todos que de alguma forma contribuíram para que eu conseguisse alcançar esse objetivo de grande importância na minha vida.

Muito obrigado a todos!

Sumário

Lista de Figuras

Lista de Tabelas

Nomenclatura

BK	*Bron-Kerbosch*
BT	Busca Tabu
INRC	*International Nurse Rostering Competition*
MIPLIB	*Mixed Integer Problem Library*
PC	Problema do Clique
PCPM	Problema do Clique de Peso Máximo
PECPL	Problema da Enumeração de Cliques com Peso acima de um Limiar
PI	Programação Inteira
SA	*Simulated Annealing*

Capítulo 1

Introdução

O problema de clique (PC) (Smith (1994)) consiste em encontrar subgrafos completos em um grafo. Um subgrafo é completo quando para cada par de vértices há uma aresta de ligação. Formalmente, dado um grafo não-direcionado $G = (V, A)$, tal que V representa o conjunto de vértices e A representa o conjunto de arestas, então $G_1 = (V_1, A_1)$ denomina-se subgrafo de G se $V_1 \subseteq V$ e $A_1 \subseteq A$, onde cada aresta de A_1 incide nos vértices de V_1.

O problema de detecção de cliques modela diversas áreas de aplicações, tais como: Atribuição de Frequências em Comunicação a Rádio (Murphey et al. (1999)), Criptografia (Juels e Peinado (2000)), Determinação de Trilhas em Circuitos Impressos (Yu et al. (1992)), Emparelhamento de Sequências de Aminoácidos em Protéinas (Samudrala et al. (1998)), Redes Sociais (Luce e Perry (1949)), Bioinformática (Ben-Dor et al. (1999)), Química Computacional (Rhodes et al. (2003)), Reconhecimento de Padrões (Pavan e Pelillo (2003)), Robótica (Ludwig e Gini (2006)), Alocação de Registradores em Compiladores (Gupta et al. (1989)), entre outras.

Existem diversas variações do problema do clique. Como exemplo, pode-se citar os problemas que consideram grafos com vértices ponderados e aqueles nos quais procura-se apenas o clique de cardinalidade máxima (pesos unitários). Todos esses problemas serão descritos no próximo capítulo. O presente trabalho investiga o problema da enumeração de cliques com peso acima de um limiar (PECPL). Esse problema considera um grafo com vértices ponderados, em que o objetivo está em encontrar todos os cliques maximais com peso acima de um valor informado previamente, conhecido como limiar.

A aplicação de interesse deste trabalho é a separação de cortes (Borndorfer (1998)),

no contexto de programação inteira (PI) (Wolsey (1998)). Na PI, encontrar todos os cliques acima de um dado peso é equivalente ao problema de encontrar todas as desigualdades violadas de clique. Segundo Chvátal (1973), esses cortes desempenham um papel fundamental na geração de desigualdades fortes para problemas com estrutura de empacotamento de nós (Campêlo et al. (2004)) e (Coll et al. (2002)). É importante destacar uma particularidade sobre o problema de otimização envolvido na separação de desigualdades: Apesar da possibilidade do problema de separação de cortes ser formulado em termos de se encontrar a desigualdade mais violada, experimentalmente se verificou que mais importante do que a descoberta dessa desigualdade é a descoberta de um grande conjunto de cortes violados (Fischetti e Lodi (2005)), (Burke et al. (2012)) e (Cornuéjols (2007)).

Foram implementados algoritmos exatos e heurísticos para a resolução do PECPL. Para o algoritmo de Bron-Kerbosch, foram implementadas oito versões diferentes, cada qual se diferindo na forma de armazenamento do grafo G e na forma de seleção dos vértices pivôs. Além disso, também foram propostas uma versão sequencial do algoritmo, originalmente recursivo, e uma versão paralela. O algoritmo de Ostergard também foi implementado, o qual faz uso da noção de coloração de vértices para a resolução do problema. A heurística busca tabu com multi-vizinhanças, descrita no artigo de Wu et al. (2012), também foi implementada. Essa heurística se difere da busca tabu tradicional, ao utilizar três vizinhanças simultaneamente em cada iteração do procedimento. Além disso, há um mecanismo de busca tabu dedicado e uma estratégia de reinicialização aleatória. É válido ressaltar que todos esses algoritmos foram adaptados para refletir o problema da enumeração de cliques com peso acima de um limiar. Por fim, foi proposta e implementada a heurística *Simulated Annealing* para a resolução do problema. Foram utilizadas as mesmas estruturas de vizinhança utilizadas na heurística busca tabu com multi-vizinhanças, diferindo-se apenas na forma de exploração do espaço de busca e nos parâmetros utilizados. Uma descrição sucinta de cada método será dada nos próximos capítulos.

Os objetivos gerais e específicos do trabalho são descritos a seguir. Além disso, ao final deste capítulo, é sumarizada a organização dos demais capítulos deste trabalho.

1.1 Objetivos

O objetivo geral do trabalho é a resolução do problema da enumeração de cliques com peso acima de um limiar. Almeja-se desenvolver algoritmos exatos e heurísticos que produzam solução ótima ou próxima do ótimo para o PECPL, em tempo de computação considerado razoável.

1.1.1 Objetivos Específicos

Para a resolução do problema abordado no presente trabalho, várias tarefas foram consideradas para o alcance do objetivo geral. Essas tarefas são apresentadas a seguir como objetivos específicos.

- Projeto do algoritmo de Bron-Kerbosch: em sua forma original, o algoritmo de Bron-Kerbosch foi implementado para encontrar todos os cliques maximais de um grafo. Faz-se necessária a adaptação do algoritmo para refletir o problema abordado no atual trabalho. Além disso, o trabalho tem como intuito estudar o projeto do algoritmo, de forma a obter melhorias em sua estrutura, possibilitando aumentar a eficiência na resolução das instâncias de interesse.
- Adaptação do algoritmo de Ostergard: o autor desse algoritmo propôs uma versão do mesmo para encontrar o clique máximo em um grafo e, posteriormente, uma versão para obtenção do clique de peso máximo. Nesse sentido, uma adaptação do algoritmo também se faz necessária.
- Adaptação da heurística busca tabu com multi-vizinhanças: os autores desenvolveram essa heurística com o foco na resolução do problema do clique de peso máximo. Assim como os outros métodos supracitados, uma adaptação do algoritmo para a resolução do problema da enumeração de cliques com peso acima de um limiar é necessária.
- Projeto e implementação da heurística *Simulated Annealing*: esse trabalho tem como interesse o desenvolvimento da heurística *Simulated Annealing* e o estudo dos valores utilizados nos parâmetros pertinentes à mesma.
- Paralelização do algoritmo de Bron-Kerbosch: implementação de uma versão paralela do algoritmo de Bron-Kerbosch, a fim de se verificar qual a aceleração ao se utilizar várias *threads* para a resolução do PECPL.

- Comparação dos métodos implementados: utilização de conjuntos de instâncias oriundas de problemas de programação inteira para realização de experimentos computacionais, a fim de se verificar os resultados obtidos para cada método, assim como uma comparação entre eles.

- Resolução do problema do clique de peso máximo: utilização das técnicas implementadas no presente trabalho para a resolução de um problema semelhante ao PECPL - o problema do clique de peso máximo. Nesse problema, o objetivo é encontrar o clique com o maior peso do grafo, não existindo mais o conceito de limiar.

1.2 Organização do Trabalho

O restante do trabalho está organizado como se segue:

- Capítulo 2: apresentação do problema do clique e suas variações. Apresentação e descrição dos algoritmos exatos e heurísticos desenvolvidos para a resolução do problema da enumeração de cliques com peso acima de um limiar.

- Capítulo 3: apresentação da versão paralela do algoritmo Bron-Kerbosch, com todas as suas limitações e dificuldades.

- Capítulo 4: descrição e apresentação dos experimentos computacionais para a resolução do problema da enumeração de cliques com peso acima de um limiar e para a resolução do problema do clique de peso máximo. Além disso, uma descrição sumarizada das instâncias utilizadas é apresentada. Também são relatadas as principais conclusões obtidas em relação ao tempo computacional despendido para a execução de cada algoritmo implementado, bem como um comparativo dos mesmos.

- Capítulo 5: apresentação das principais conclusões obtidas, das dificuldades encontradas e do rumo de pesquisa futura.

Capítulo 2

Problema da Enumeração de Cliques com Peso acima de um Limiar

O problema da detecção de cliques modela diversas áreas de aplicações, destacando-se: atribuição de frequências em comunicação a rádio, criptografia, determinação de trilhas em circuitos impressos, emparelhamento de sequências de aminoácidos em proteínas, redes sociais, bioinformática, química computacional, reconhecimento de padrões, robótica, alocação de registradores em compiladores. Existem diversas variações do problema do clique, que serão descritas nas seções seguintes.

2.1 Problema do Clique com grafos não ponderados

Nessa seção, serão revisados os problemas da detecção de cliques para grafos em que os vértices não possuem peso associado. O problema do clique máximo consiste em determinar o clique com a maior cardinalidade em um grafo. Em seus trabalhos, os autores Östergård (2002) e Debroni et al. (2011) propõem a resolução desse problema. Já o problema do k-clique consiste em encontrar todos os cliques de tamanho k. Para resolução desse problema, são citados os trabalhos de Goldschmidt et al. (1996) e Vassilevska (2009). Por fim, o problema da enumeração de cliques maximais consiste em encontrar todos os cliques não dominados. Um clique C é maximal, se não existe um vértice $v \in G$, que não faça parte do clique, mas tenha ligação com todos os vértices do mesmo, levando a uma extensão do seu tamanho. Os autores Schmidt et al. (2009) e Bron e Kerbosch (1973) resolvem esse problema.

Considere a Figura 2.1, onde é apresentado um grafo com seis vértices (numerados de 1 a 6). Essa figura trata de um grafo não ponderado, onde não existem pesos associados a cada vértice. Logo após a Figura 2.1, são mostrados os cliques maximais encontrados no grafo, o clique máximo e os cliques de tamanho k.

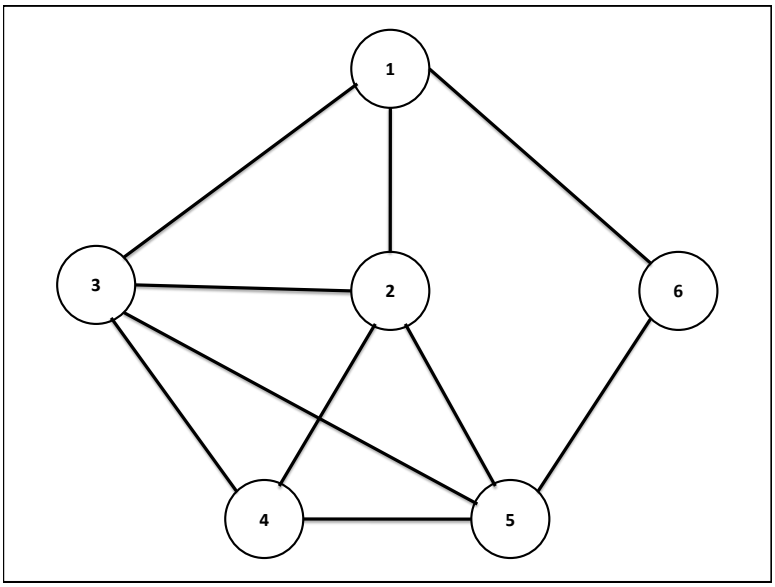


Figura 2.1: Grafo com vértices não ponderados

- Problema do clique máximo: {2, 3, 4, 5}.
- Problema do k-clique ($k = 2$): {1, 2}, {1, 3}, {1, 6}, {2, 3}, {2, 4}, {2, 5}, {3, 4}, {3, 5}, {4, 5}, {5, 6}.
- Problema da enumeração de cliques maximais: {1, 2, 3}, {2, 3, 4, 5}, {1, 6}, {5, 6}.

2.2 Problema do Clique com grafos ponderados

Nessa seção, serão revisados os problemas da detecção de cliques para grafos ponderados. Em tais grafos, cada vértice possui um peso associado. Esses problemas são uma generalização dos problemas descritos anteriormente - Seção 2.1 -, onde os pesos dos vértices são iguais a um. O problema do clique de peso máximo consiste em encontrar o clique de maior peso dentre todos os cliques do grafo. Em seus trabalhos, os autores Östergård (1999) e Wu et al. (2012) propõem a resolução desse problema. Já o problema

da enumeração de cliques com peso acima de um limiar consiste em encontrar todos os cliques não dominados do grafo cujo peso total está acima de um limiar, onde esse valor é informado previamente. Para resolução desse problema, é citado o trabalho de Brito e Santos (2011).

Considere a Figura 2.2, onde é apresentado um grafo com seis vértices (numerados de 1 a 6), onde cada vértice possui um peso associado. Cada peso é ilustrado entre chaves próximo ao vértice correspondente. Logo após a Figura 2.2, são apresentados os cliques com peso acima de um limiar e o clique de peso máximo.

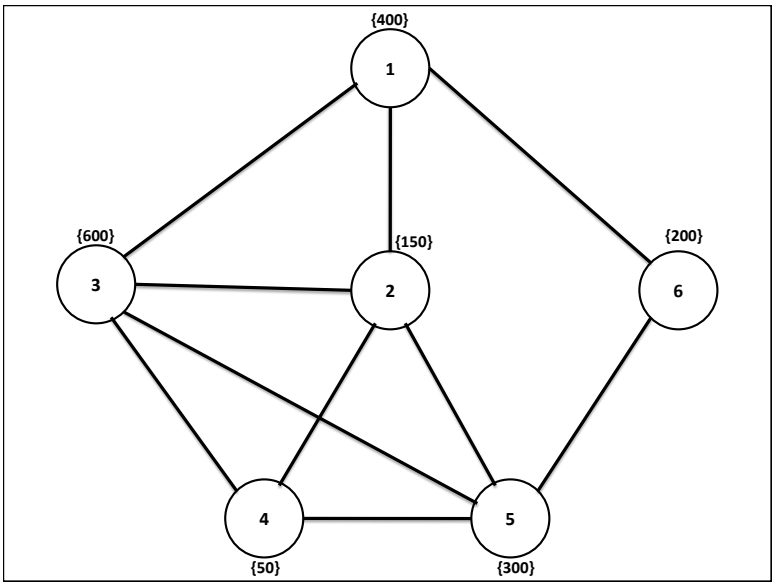


Figura 2.2: Grafo com vértices ponderados

- Problema do clique de peso máximo: {1, 2, 3}.
- Problema da enumeração de cliques com peso acima de um limiar (limiar = 1020): {1, 2, 3}, {2, 3, 4, 5}.

No presente trabalho, o objetivo é o estudo do problema da enumeração de cliques com peso acima de um limiar (PECPL). Tal problema considera um grafo com vértices ponderados, em que o objetivo está em encontrar todos os cliques maximais com peso acima de um valor informado previamente, conhecido como limiar. No contexto da programação inteira, encontrar todos os cliques acima de um dado peso é equivalente ao problema de encontrar todas as desigualdades violadas de clique. Em Brito (2015), algoritmos dessa natureza são utilizados para a separação de cortes de clique. Foram

utilizados algoritmos exatos e heurísticos para a resolução deste problema, que serão descritos minuciosamente nas seções seguintes.

A Seção 2.3 descreve o funcionamento do algoritmo de Bron-Kerbosch e como foi realizada a implementação das oito versões. Uma versão iterativa do algoritmo de Bron-Kerbosch, originalmente recursivo, foi desenvolvida e é apresentada na Seção 2.4. As seções seguintes descrevem o algoritmo exato de Ostergard (Seção 2.5), a heurística busca tabu com multi-vizinhanças (Seção 2.6) e a heurística *Simulated Annealing* (Seção 2.7).

2.3 Algoritmo de Bron-Kerbosch

O algoritmo de Bron-Kerbosch clássico (Bron e Kerbosch (1973)) tem como objetivo encontrar todos os cliques maximais de um grafo G. É utilizado o processo de *backtracking*, e apenas cliques maximais são gerados, evitando assim, uma boa parte da comparação de um determinado conjunto de cliques que já foram testados.

No trabalho de Brito e Santos (2011), os autores adaptaram o algoritmo de Bron-Kerbosch básico para refletir o problema da enumeração de cliques com peso acima de um limiar. Além disso, propuseram algumas melhorias no algoritmo, descritas sucintamente a seguir. Primeiramente, utilizaram-se da noção de vértice pivô (Tomita et al. (2006)), com o propósito de acelerar a produção de cliques com peso alto. O vértice de máximo grau modificado foi escolhido como vértice pivô. Cada vértice tem um grau modificado, o qual se refere ao somatório do seu grau com os graus de todos os vértices adjacentes a ele. Para reduzir o número de chamadas, os autores observaram que se um clique contém um vértice adjacente ao vértice pivô u, tal vértice também faz parte do clique. Logo, apenas u e os vértices não adjacentes a ele precisam ser testados. Essa é a ideia central do pivoteamento, desenvolvido pelos autores do algoritmo de Bron-Kerbosch. Por fim, fora utilizado uma técnica de estimativa de peso (vide linha 7 do Algoritmo 1), a qual calcula o limite superior do peso de um clique parcial por meio da soma dos pesos dos vértices contidos nesse clique com a soma de todos os pesos dos vértices candidatos a entrarem no clique. Nesse sentido, alguns cliques parciais podiam ser previamente eliminados. O algoritmo implementado por Brito e Santos (2011) é apresentado em Algoritmo 1.

O conjunto C representa os vértices que fazem parte do clique. O conjunto P representa os vértices que têm ligação com todos os vértices de C - vértices candidatos a entrar no clique. Por sua vez, o conjunto S contém todos os vértices que já foram anali-

Algoritmo 1 Algoritmo de Bron-Kerbosch adaptado

```
1: procedimento BKA(C, P, S)
2:     se P e S estão vazios então
3:         se w(C) ≥ limiar então
4:             Adiciona o clique C no conjunto solução;
5:         fim se
6:     fim se
7:     se w(C) + h(C) ≥ limiar então
8:         Escolha um vértice pivô u de P;
9:         para cada Vértice v em P\N(u) faça
10:            BKA(C ∪ {v}, P ∩ N(v), S ∩ N(v));
11:            P := P\{v};
12:            S := S ∪ {v};
13:        fim para cada
14:    fim se
15: fim procedimento
```

sados e não levam a uma extensão do conjunto P. O peso do clique C está armazenado em $w(C)$. O somatório dos pesos de todos os vértices do conjunto P está em $h(C)$, onde tal variável em conjunto com a variável $w(C)$ formam a estimativa de peso, a qual é usada como um limite superior para o peso do Clique C.

No presente trabalho, foram utilizadas oito implementações (versões) do Algoritmo de Bron-Kerbosch adaptado, apresentado em Brito e Santos (2011). Essas implementações são diferentes na forma de armazenamento do grafo e na forma de seleção dos vértices pivôs:

1. Uso de matriz: nesta implementação foi utilizada uma matriz binária m de dimensão $|V| \times |V|$ para representação das arestas existentes no grafo. Nesse sentido, caso exista uma aresta conectando os vértices v_1 e v_2, $m[v_1][v_2] = 1$ e $m[v_2][v_1] = 1$. Os valores zero na matriz binária representam que os vértices não estão conectados. Esta abordagem é vantajosa no processamento do Algoritmo de Bron-Kerbosch adaptado, no momento da verificação da existência ou não de aresta conectando um par de vértices qualquer porque é realizada em O(1). Como desvantagem, há um uso considerável de memória.

2. Uso de lista: com o uso desta implementação, os vértices vizinhos de um vértice v_1 qualquer são representados por uma lista de adjacência. Em relação à abordagem anterior (Versão 1), esta economiza o uso de memória. Além disso, a interseção realizada na linha 10 do Algoritmo 1, é feita de forma mais eficiente, visto que

não é necessário percorrer toda a matriz para procurar a interseção dos vértices do conjunto P ou S com os vértices vizinhos do vértice v. Como desvantagem, o tempo gasto para verificação da existência de arco conectando dois vértices qualquer pode ser dispendioso (O(n)), sendo n a quantidade de vértices.

3. Uso de lista e matriz: esta terceira abordagem considera uma implementação híbrida, com o uso de lista e matriz (Versões 1 e 2). Nesse sentido, quando ocorre a verificação de existência de aresta entre dois vértices, é utilizada a matriz para esse fim. Quando o objetivo é realizar a interseção do conjunto P ou S com os vértices vizinhos do vértice v, a lista de adjacência é utilizada.

4. Uso do conjunto P ordenado: na linha 8 do Algoritmo 1, o vértice pivô u é escolhido a partir do vértice pertencente ao conjunto P com o máximo grau modificado. Uma quarta implementação do Algoritmo de Bron-Kerbosch considera a otimização realizada com o uso de lista e matriz (Versão 3), acrescida da manutenção do conjunto P sempre ordenado, de modo a evitar percorrer todo esse conjunto quando é realizada a escolha do vértice pivô.

5. Uso do conjunto $P\backslash N(u)$ como vetor: esta versão modifica a forma de representação do conjunto $P\backslash N(u)$, usado na linha 9 do Algoritmo 1. Tal conjunto, que anteriormente era representado como uma lista encadeada na Versão 4, passa a ser representado como um vetor.

6. Pivoteamento 1: utiliza-se a ideia do máximo grau modificado, citado anteriormente, porém, o objetivo está em ordenar os vértices pelo máximo peso modificado. O conceito máximo peso modificado se refere ao somatório do peso do vértice com os pesos de todos os vértices adjacentes a ele.

7. Pivoteamento 2: faz uso da combinação entre peso e grau para seleção do vértice pivô. Nesse sentido, o Pivoteamento 2 é dado pela seguinte fórmula: (máximo grau modificado * maior peso) + (máximo peso modificado). Os conceitos máximo grau modificado e máximo peso modificado já foram explicados. Já o conceito maior peso refere-se ao peso do vértice de maior peso do grafo.

8. Pivoteamento 3: semelhante ao Pivoteamento 2, este pivoteamento é dado por (máximo grau modificado * menor peso) + (máximo peso modificado), onde menor peso refere-se ao peso do vértice de menor peso do grafo.

2.4 Algoritmo de Bron-Kerbosch Iterativo

Como mencionado anteriormente, o algoritmo de Bron-Kerbosch é implementado de forma recursiva. Nesta seção, é apresentada uma versão iterativa para esse algoritmo. Considerando as diversas versões descritas na seção anterior, a versão iterativa representa o grafo de interesse como na Versão 5 do algoritmo de Bron-Kerbosch recursivo. A ideia é adicionar a uma pilha as chamadas recursivas do algoritmo original, sejam elas expansão ou retrocesso na árvore de busca. A versão iterativa do algoritmo de Bron-Kerbosch adaptado é apresentada no Algoritmo 2.

Algoritmo 2 Versão Iterativa do Algoritmo de Bron-Kerbosch adaptado

```
1:  procedimento IBKA(P, PsemU)
2:      Pilha := ∅;
3:      Pilha.push({}, P, {}, PsemU);
4:      enquanto Pilha não está vazia faça
5:          C, P, S, PsemU := Pilha.pop();
6:          se P e S estão vazios então
7:              se w(C) ≥ limiar então
8:                  Adiciona o clique C no conjunto solução;
9:              fim se
10:         fim se
11:         se PsemU não está vazio então
12:             Escolha um vértice v de PsemU;
13:             Pilha.push(C, P\{v}, S ∪ {v}, PsemU\{v})
14:             Escolha um vértice pivô u de P;
15:             Pilha.push(C ∪ {v}, P ∩ N(v), S ∩ N(v), P\N(u))
16:         fim se
17:     fim enquanto
18: fim procedimento
```

A definição de cada conjunto C, P e S é exatamente a mesma do algoritmo de Bron-Kerbosch recursivo. A estratégia da versão iterativa é usar uma pilha para simular a recursão. A fim de se estabelecer uma comparação entre os algoritmos, pode-se perceber que a linha 13 da versão iterativa do algoritmo (Algoritmo 2) simula o retrocesso na árvore de busca da versão recursiva (vide linhas 11-12 do Algoritmo 1). De modo análogo, a linha 15 do Algoritmo 2 simula a expansão na árvore de busca (linha 10 do Algoritmo 1).

2.5 Algoritmo de Ostergard

O algoritmo de Ostergard foi desenvolvido para encontrar o clique máximo em um grafo (Östergård (2002)). O autor também desenvolveu uma versão do algoritmo para obtenção do clique de peso máximo em um grafo (Östergård (1999)). O algoritmo impõe uma ordenação aos vértices da seguinte forma: Colore-se o clique uma cor de cada vez, sempre adicionando os vértices pela ordem de importância, dada pelo vértice de peso mínimo no grafo restante, seguido pelo vértice vizinho que tem a maior soma de pesos. Vale ressaltar que em uma coloração de vértices, vértices adjacentes não podem receber cores iguais. Ao determinar uma nova classe de cor, o grafo induzido pelos vértices sem cor é construído e, em seguida, contanto que exista um vértice que pode ser adicionado à classe de cor, o vértice com maior grau é adicionado. Os vértices são rotulados como v_n, v_{n-1}, ..., v_1 na ordem em que foram adicionados a uma classe de cor.

Um conjunto de vértices denominado *working set*, adjacentes a todos os vértices fixos na busca pelo clique de peso máximo, é mantido. Em cada nível da árvore de busca, um vértice do conjunto citado anteriormente é adicionado aos vértices fixos e os conjuntos são atualizados. Algumas subárvores são podadas, quando a soma do peso do conjunto atual e do conjunto de trabalho (*working set*) é menor ou igual ao peso de um clique já construído. O algoritmo de Ostergard, adaptado para resolver o problema de interesse do presente trabalho, é apresentado em Algoritmo 3.

As adaptações necessárias para refletir o mesmo problema abordado no atual trabalho estão contidas nas linhas 6-10. Em suma, são adicionados ao conjunto solução, todos os cliques com o peso acima de um limiar, previamente informado. Uma verificação adicional é necessária para garantir que não existam cliques dominados por outros cliques no conjunto solução.

Dois trabalhos presentes na literatura estudaram a eficiência do Algoritmo de Ostergard para o Problema do Clique de Peso Máximo: No trabalho de Yamaguchi e Masuda (2008), os autores propõem um novo algoritmo exato que apresenta melhores resultados - quando comparados ao Algoritmo de Ostergard - para grafos com densidade acima de 0.8. O mesmo ocorre no trabalho de Kumlander (2008), onde é utilizada uma melhoria na ordenação dos vértices antes que ocorra a coloração do grafo. Este novo algoritmo é três vezes mais rápido que o Algoritmo de Ostergard.

Algoritmo 3 Algoritmo de Ostergard adaptado

```
1: para cada i := n até 1 faça
2:     WCLIQUE(S_i ∩ N(v_i), w[i])
3: fim para cada
4: função WCLIQUE(U, peso)
5:     se |U| = 0 então
6:         se peso ≥ limiar então
7:             se cliqueNaoDominado(C, CSol) então
8:                 Adicione o clique C no conjunto solução CSol.
9:             fim se
10:        fim se
11:    fim se
12:    enquanto U ≠ ∅ faça
13:        se peso + wt(U) ≤ limiar então
14:            return
15:        fim se
16:        i := min{j | v_j ∈ U}
17:        U := U\{v_i}
18:        WCLIQUE(U ∩ N(v_i), peso + w[i])
19:    fim enquanto
20:    return
21: fim função
```

2.6 Heurística Busca Tabu com Multi-Vizinhanças

No trabalho de Wu et al. (2012), os autores implementaram a heurística busca tabu com multi-vizinhanças para a resolução do problema do clique de peso máximo. A metaheurística utiliza-se de três vizinhanças, cada qual com movimentos diferentes. Ao contrário da busca tabu tradicional, a busca tabu nesse trabalho explora, em cada iteração, a união das três vizinhanças e seleciona a melhor solução admissível (não-tabu ou melhoria global) da vizinhança, ou seja, a solução que produz o maior ganho de peso. Além disso, os autores implementaram um mecanismo de busca tabu dedicado e a estratégia de reinicialização aleatória.

Em suma, a heurística funciona do seguinte modo: a construção da solução inicial é realizada de forma aleatória, onde um vértice v_i é adicionado ao clique C, inicialmente vazio. Logo após, outro vértice v_j é escolhido para ser incluído no clique, desde que esse já não faça parte do clique ($v_j \notin C$) e esteja ligado a todos os vértices pertencentes ao mesmo. Esse procedimento é repetido até que não exista mais nenhum vértice v_j com tal característica.

Os operadores básicos de movimento são denominados *ADD*, *SWAP* e *DROP*. Tais operadores são baseados em dois subconjuntos de vértices - *PA* e *OM* - relacionados a um dado clique *C*. O subconjunto *PA* é composto pelos vértices que não estão inclusos no clique *C*, porém, estão conectados a todos os vértices de *C*. Por sua vez, o subconjunto *OM* é composto dos vértices que não estão inclusos no clique *C*, porém, estão conectados a $|C|$ - 1 vértices do clique.

Para ilustrar a utilização dos subconjuntos *PA* e *OM*, considere o exemplo apresentado na Figura 2.3, onde há um clique *C*, contendo os vértices 1, 2, 3 e 4. Pode-se perceber que o único vértice que está conectado a todos os vértices pertencentes ao clique *C* e não está contido no mesmo, é o vértice 5. Portanto, este vértice fará parte do subconjunto *PA*. Quanto ao subconjunto *OM*, os vértices 6 e 7 farão parte dele, visto que eles estão conectados a todos os vértices contidos no clique *C*, com exceção do vértices 1 e 3, respectivamente.

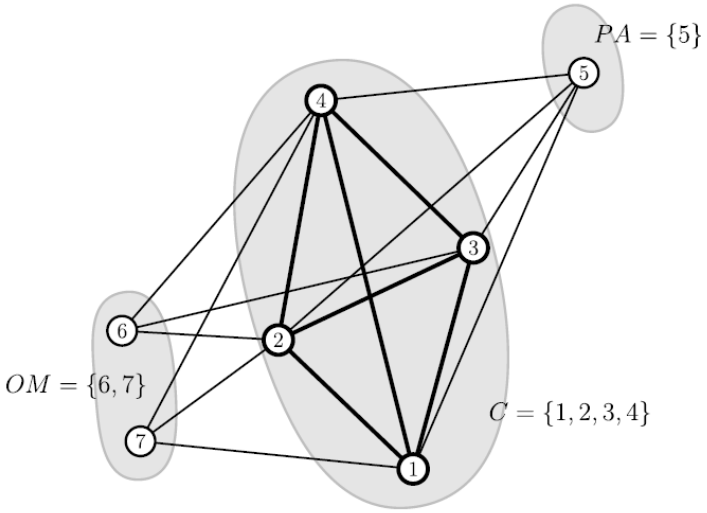


Figura 2.3: Representação dos subconjuntos *PA* e *OM* (Figura extraída de Wu et al. (2012))

O operador *ADD* adiciona um vértice v ($v \in PA$) ao atual clique *C*. É utilizado somente quando $PA \neq \emptyset$ e sempre representa uma melhora no peso do clique. O operador *SWAP* realiza a troca de um vértice v_i ($v_i \in OM$) com o vértice v_j ($v_j \in$ C) que não esteja ligado com o vértice v_i. Esse movimento pode representar uma piora na qualidade da solução corrente. Por sua vez, o operador *DROP* remove um vértice v do clique *C*.

Esse operador de movimento sempre diminui o peso do clique.

Para estabelecer uma forma mais global de diversificação, com o objetivo de visitar áreas ainda inexploradas no espaço de busca, os autores utilizaram a estratégia de reinicialização aleatória. Esse processo é acionado sempre que a busca atual estiver presa em um ótimo local profundo. Formalmente, isso acontece quando o número máximo permitido de iterações consecutivas sem melhora, representado por L, é excedido: $NI > L$.

A heurística busca tabu com multi-vizinhanças - adaptada para resolver o problema da enumeração de cliques com peso acima de um limiar - é apresentada no Algoritmo 4.

Algoritmo 4 Heurística Busca Tabu com Multi-Vizinhanças adaptado

Entrada: Um grafo ponderado $G = (V, E, w)$, inteiro L (profundidade da busca), inteiro $QtdCliques$ (solução ótima)
Saída: Conjunto $Csol$ de cliques com peso acima de um limiar ($PesoMin$)

enquanto ($tempoExecucao < tempoLimite$ e $|Csol| < QtdCliques$) **faça**
    $C = Inicializar()$
    Iniciar $lista\_tabu$
    $NI = 0$
    se ($w(C) \geq PesoMin$ e $PA = \emptyset$) **então**
        se ($cliqueNaoRepetido(C, Csol)$) **então**
            Adiciona o clique C no conjunto solução $Csol$.
        fim se
        $NI = L$
    fim se
    enquanto ($NI < L$ e $|Csol| < QtdCliques$) **faça**
        Construção das vizinhanças N_1, N_2 e N_3 de C
        Escolha o melhor vizinho global permitido $C' \in N_1 \cup N_2 \cup N_3$
        $C = C'$ {Mover para a nova solução}
        se ($w(C) \geq PesoMin$ e $PA = \emptyset$) **então**
            se ($cliqueNaoRepetido(C, Csol)$) **então**
                Adiciona o clique C no conjunto solução $Csol$.
            fim se
            $NI = L$
        fim se
        $NI = NI + 1$
        Atualize $lista\_tabu$
    fim enquanto
fim enquanto
return Conjunto $Csol$

As seguintes modificações foram realizadas no algoritmo original proposto por Wu

et al. (2012): o inteiro L, que representa a profundidade da busca, foi definido como a quantidade de vértices do grafo ($L = |V|$). Na construção da solução inicial, a cada iteração é escolhido um vértice v_i diferente, iniciando-se de $v_i = 1$ até $v_i = |V|$. No algoritmo original de Wu et al. (2012), tal vértice era escolhido sempre de forma aleatória, de modo que essa abordagem não produziu bons resultados no problema da enumeração de cliques com peso acima de um limiar.

A execução é encerrada quando o tempo limite de execução é excedido ou a solução ótima é encontrada, ou seja, todos os cliques acima de um limiar foram detectados. Os cliques são adicionados ao conjunto solução *CSol* quando o peso dos mesmos forem maior que um dado limiar e o conjunto *PA* estiver vazio, a fim de evitar que cliques dominados (cliques não maximais) sejam adicionados ao conjunto solução. Além disso, por se tratar de uma heurística, vários cliques repetidos - cliques que possuem exatamente os mesmos vértices - podem ser encontrados. A heurística verifica isso e considera apenas um dos cliques repetidos.

2.7 Heurística Simulated Annealing

A metaheurística *Simulated Annealing* (SA), proposta por Kirkpatrick et al. (1983), trata-se de uma técnica de busca local probabilística, onde busca-se a simulação do processo de recozimento de metais. A ideia dessa simulação é que o resfriamento rápido conduz a produtos meta-estáveis, de maior energia interna. Por outro lado, resfriamento lento conduz a produtos mais estáveis, estruturalmente fortes, de menor energia. Durante o recozimento, o material passa por vários estados possíveis. Ao se fazer uma analogia a um problema de otimização combinatória, os estados possíveis de um metal correspondem a soluções do espaço de busca. A energia em cada estado corresponde ao valor da função objetivo e a energia mínima ou máxima, corresponde ao valor de uma solução ótima local, possivelmente global. Em suma, utilizando-se dessa metaheurística, no início de sua execução, há uma chance maior de escapar de ótimos locais e à medida que a temperatura se aproxima de zero, a probabilidade de se aceitar movimentos de piora é reduzida.

A heurística *Simulated Annealing* considerou as mesmas estruturas de vizinhança utilizadas na heurística busca tabu com multi-vizinhanças, descritas na Seção 2.6. A fim de ressaltar a diferença entre essas duas heurísticas, vale lembrar que a heurística *Simulated Annealing* gera apenas um vizinho, ao passo que a heurística busca tabu gera um

conjunto de vizinhos. Além disso, a última heurística mantém uma lista de movimentos tabus, a fim de evitar o processo de ciclagem, ao passo que a primeira heurística utiliza-se do conceito de temperatura, onde vizinhos piores são aceitos probabilisticamente, contendo maiores chances de serem aceitos a uma temperatura alta.

Os parâmetros utilizados na heurística SA são: α, T_0 e $SAmax$. Tais parâmetros se referem respectivamente à razão de resfriamento, à temperatura inicial e ao número máximo de iterações em uma dada temperatura. Os valores dos parâmetros utilizados na heurística SA foram setados por meio de experimentos computacionais apresentados no Apêndice B.

Capítulo 3

Versão Paralela do Algoritmo de Bron-Kerbosch

Neste capítulo, serão apresentadas e discutidas técnicas de paralelização para o algoritmo de Bron-Kerbosch. Mais especificamente, a paralelização será realizada na versão 4 (Uso do conjunto P ordenado) implementada do algoritmo acima mencionado. Antes de apresentar o que fora desenvolvido no presente trabalho, é válido destacar alguns trabalhos disponíveis na literatura sobre paralelização em problemas de clique. No trabalho de McCreesh e Prosser (2012), os autores propõem a distribuição de um algoritmo exato para a resolução do problema do clique máximo. O foco dos autores era maximizar o *costup*, ou seja, aumentar o desempenho com pouco aumento em custos, sejam eles relacionados a dinheiro ou energia gastos com equipamentos (computadores).

Os autores propõem a divisão do problema em n partes, sendo n o número de vértices do grafo. Cada processador expande uma árvore enraizada em um nó do nível 1, onde o clique corrente contém um único vértice. A ideia é que, inicialmente, cada processador contenha um clique diferente de tamanho um. Tal estratégia não resolve o problema do tamanho desigual das árvores de pesquisa. Nesse sentido, os autores propuseram expandir uma árvore de busca por todos os triângulos. Em suma, cada processador terá um clique (triângulo) diferente de tamanho três, no início da execução.

No trabalho de Schmidt et al. (2009), os autores utilizam o conceito de roubo de trabalho para minimizar o tempo ocioso de elementos. Nesse contexto, os autores propõem uma representação da árvore de busca, por meio da utilização da estrutura de caminho de candidato, para permitir paralelização e garantir um procedimento de balanceamento

dinâmico de carga, a fim de manter todos os processos ocupados, sem atrasos significantes de comunicação. Em suma, a ideia é manter a estrutura que faz parte de cada elemento candidato a entrar no clique, para que seja possível o roubo de trabalho entre processos. Como desvantagem, um processo não pode terminar, mesmo quando nenhum trabalho adicional está disponível. Isso ocorre porque cada processo deve lidar com os pedidos de balanceamento de carga até que possa determinar que todos os outros processos tenham atingido o mesmo ponto.

Em todos trabalhos supracitados, pode-se perceber alguns problemas na paralelização de um algoritmo voltado à resolução do problema do clique. A divisão de um problema em subproblemas de mesmo tamanho não é trivial, visto que subárvores podem diferir consideravelmente em tamanho. Além disso, a árvore de pesquisa tem um retrocesso altamente irregular e é praticamente impossível predizer a estrutura de uma árvore. Como solução para esses problemas, os autores usam os termos roubo de trabalho/doação de trabalho. A ideia é evitar que *threads* fiquem ociosas. Nesse sentido, quando uma *thread* termina seu trabalho, ela pode roubar ou receber uma doação de trabalho de outra *thread* qualquer.

A próxima seção (Seção 3.1) ilustra uma tentativa de paralelizar o algoritmo de Bron-Kerbosch, por meio da divisão da carga de trabalho em subgrafos conexos. Por sua vez, a Seção 3.2 mostra a versão paralela implementada do algoritmo supracitado para a resolução do problema da enumeração de cliques com peso acima de um limiar.

3.1 Paralelismo utilizando subgrafos conexos

Como discutido na introdução deste capítulo, a divisão da carga de trabalho entre processadores é extremamente complicada, devido à exigência de sincronismo entre os conjuntos pertinentes ao algoritmo implementado para que se obtenha um resultado final correto, ou seja, para que se consiga resolver o PECPL de forma ótima e que não sejam geradas soluções inválidas por meio da troca de informações não sincronizadas. Nesse contexto, uma primeira tentativa de se paralelizar o algoritmo de Bron-Kerbosch considerou a divisão do trabalho em subgrafos conexos, de modo que não seria necessária a utilização de conjuntos em paralelo, ou seja, que exigissem sincronismo.

Para ilustrar o que fora informado, considere o grafo mostrado na Figura 3.1. Nesse grafo, existem 14 vértices, numerados de 1 a 14 e 14 arestas conectando os mesmos.

Como pode-se perceber no grafo, existem quatro conjuntos ({1, 2, 3}, {4, 5, 6}, {7, 8, 9, 10, 11}, {12, 13, 14}) totalmente independentes entre si. Nesse sentido, paralelizar o algoritmo para resolução do PECPL seria trivial, visto que não seriam necessários mecanismos de sincronização entre *threads*.

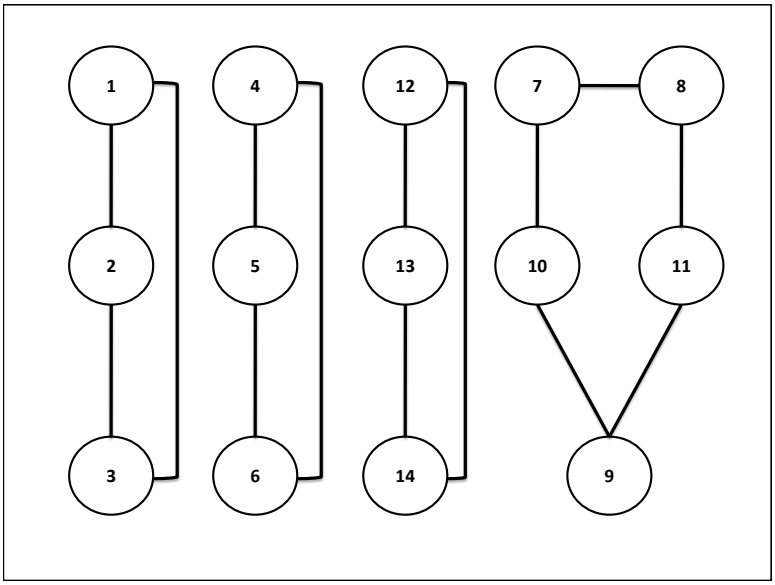


Figura 3.1: Grafo contendo 14 vértices e 4 subconjuntos conexos

Para as instâncias mais críticas, ou seja, aquelas instâncias que demandam um maior tempo computacional para sua execução, essa paralelização não tornou a resolução do problema mais rápida, em virtude das características de tais instâncias. Ao se considerar grafos (ou instâncias) conexos, não é possível a divisão do problema em vários subproblemas (ou conjuntos) independentes e nesse sentido, apenas uma *thread* ficaria responsável por todo o trabalho. Outra estratégia de paralelização foi desenvolvida e é apresentada na próxima seção.

3.2 Algoritmo Paralelo de Bron-Kerbosch

O algoritmo de Bron-Kerbosch foi paralelizado do seguinte modo: observou-se que no laço contido na linha 9 do Algoritmo 1 haveria a possibilidade da divisão das iterações entre *threads*, ressalva algumas modificações que serão descritas a seguir. A Figura 3.2 apresenta um bloco do pseudocódigo do algoritmo de Bron-Kerbosch. Em suma, esse bloco é responsável pela iteração dos vértices contidos no conjunto P, excluídos os

vizinhos do vértice u. Tal vértice fora escolhido como sendo o vértice de máximo grau modificado.

```
 9.   para cada Vértice v em P \ N(u)
10.         BKA (C ∪ {v}, P ∩ N(v), S ∩ N(v));
11.         P := P \ {v};
12.         S := S ∪ {v};
13.   fim para cada
```

Figura 3.2: Paralelização no algoritmo de Bron-Kerbosch (Código extraído do Algoritmo 1)

Nesse sentido, com o uso da paralelização, cada *thread* ficaria responsável por percorrer a árvore de busca dado um vértice do conjunto $P \backslash N(u)$. Com isso, cada *thread* passa a ter seus próprios conjuntos C, P e S e as linhas 11 e 12 são executadas antes da chamada recursiva.

Capítulo 4

Experimentos Computacionais

Os experimentos computacionais foram realizados em um computador com processador Intel(R) Core(TM) i7-4790, 3.60Hz, 16GB de Memória RAM. As implementações foram realizadas na linguagem de programação C e podem ser disponibilizadas mediante solicitação. Foi estabelecido como critério de parada um tempo limite de execução (TLE) de 60 segundos para a resolução de cada instância. Para aquelas instâncias onde o tempo de execução foi inferior a 0,001 segundos, foi atribuído o tempo padrão de 0,001 segundos. Na execução das heurísticas, busca tabu com multi-vizinhanças e *Simulated Annealing*, para a resolução do problema da enumeração de cliques com peso acima de um limiar, a heurística é encerrada quando é encontrado 60% do peso total da solução ótima. Tal critério é justificado pela dificuldade de se encontrar todos os cliques pertencentes ao problema, visto que heurísticas provavelmente não explorarão todo o espaço de busca e a verificação contínua para não permitir a contabilidade de cliques repetidos é extremamente custosa.

Ao todo, foram utilizadas 7292 instâncias para a realização dos experimentos, oriundas de quatro conjuntos de problemas formulados por meio do uso de programação inteira, que serão descritos na Seção 4.1. É válido ressaltar que, devido ao total elevado de instâncias, tornou-se impossível descrever minuciosamente os resultados encontrados para todas as instâncias. Nesse sentido, as seções seguintes descrevem como foram geradas as instâncias e os resultados obtidos para cada conjunto de problemas. Os experimentos foram conduzidos em duas partes: em um primeiro momento, as instâncias foram utilizadas para a resolução do PECPL (Seção 4.2). Posteriormente, o foco foi a resolução do problema do clique de peso máximo (Seção 4.3).

A título de padronização, as Seções 4.2.1, 4.2.2, 4.2.3 e 4.2.4 apresentam os resultados obtidos, em cada conjunto de problemas formulados por meio do uso de programação inteira, da seguinte maneira: primeiramente, uma breve descrição de cada conjunto é dada, ao informar quantas instâncias pertencem àquele conjunto, o número máximo e mínimo de vértices e arestas dentre as instâncias. Posteriormente, cada seção apresenta três tabelas, sendo que a primeira tabela apresenta o tempo total gasto na execução das instâncias do conjunto, quando da utilização das implementações desenvolvidas no presente trabalho. A segunda tabela ilustra as características de um subgrupo de instâncias do conjunto e a última tabela apresenta os resultados obtidos para as instâncias descritas na terceira tabela. Como em alguns casos não foi possível a obtenção da solução ótima para uma determinada instância, será atribuído o valor TLE na respectiva coluna da tabela. Além disso, ocorreram alguns estouros de memória, ao se utilizar a versão iterativa do algoritmo de Bron-Kerbosch (ver Seção 2.4). O símbolo (*) foi utilizado na coluna da tabela para ilustrar o estouro.

4.1 Caracterização das Instâncias

As instâncias utilizadas neste trabalho foram geradas a partir de quatro conjuntos de problemas formuladas por meio do uso da técnica de programação inteira. O primeiro conjunto (MIPLIB) apresenta instâncias de *benckmark* da MIPLIB 2010 (Koch et al. (2011)), contendo 87 instâncias. Desde a sua introdução em 1992, a MIPLIB tornou-se uma biblioteca padrão de testes, usada para comparar o desempenho dos resolvedores de PI. Ela contém uma coleção de problemas reais, sendo na maior parte, aplicações industriais.

O segundo conjunto de instâncias (INRC) foi obtido da formulação usada por Santos et al. (2014) para resolver problemas da *international nurse rostering competition* (Haspeslagh et al. (2014)), contendo 60 instâncias. Trata-se de uma competição realizada para incentivar e comparar pesquisas relacionadas ao problema de escalonamento de enfermeiras.

Por sua vez, o terceiro conjunto consiste em problemas de planejamento de rotas para o Telebus (Borndörfer et al. (1999)), um serviço de transporte de pessoas com deficiência física localizado em Berlim, contendo 28 instâncias. Esses problemas apresentam formulações de PI baseadas no problema de particionamento de conjuntos.

O quarto conjunto (Uchoa) consiste em apenas uma instância do problema de P-dispersão, reduzido a uma sequência de problemas de conjunto independente. Tal instância possui muitas restrições esparsas, de modo que a separação de cortes encontra muitos cliques violados.

Para cada problema formulado por meio da técnica de programação inteira dos conjuntos supracitados, foram gerados vários grafos de conflitos. Esses grafos foram construídos a partir da utilização da rotina de separação de cortes desenvolvida no trabalho de Brito (2015). Nessa rotina, um grafo inicial é construído a partir da análise das restrições impostas pelo programa inteiro. Os demais grafos são construídos com base na análise do problema gerado pela aplicação iterativa de planos de corte. Com esse procedimento, foram geradas um total de 7292 instâncias, sendo 399 instâncias geradas a partir do conjunto MIPLIB, 3804 do INRC, 3085 do Telebus e 4 do Uchoa.

A Tabela 4.1 apresenta as características relacionadas à densidade de cada um dos quatro conjuntos de problemas de Programação Inteira, descritos anteriormente. Um grafo é denso quando possui muitas arestas para uma determinada quantidade de vértices. Quando o grafo possui poucas arestas, ele é dito esparso.

Tabela 4.1: Densidade dos conjuntos de problemas de Programação Inteira

Densidade/Conjunto	**MIPLIB**	**INRC**	**Telebus**	**Uchoa**
Densidade Média	0,14	0,01	0,04	0,16
Densidade Mínima	0,01	0,01	0,01	0,11
Densidade Máxima	0,84	0,07	0,37	0,21

Ao se analisar a Tabela 4.1, pode-se perceber que os conjuntos **INRC**, **Telebus** e **Uchoa** apresentam grafos esparsos. Tal conclusão se dá ao observar os valores da densidade média, mínima e máxima de cada um desses conjuntos. O conjunto **MIPLIB** apresenta algumas instâncias com uma quantidade grande de arestas (grafo denso). A densidade máxima para este conjunto é de 0,84.

4.2 Análise e Apresentação dos Experimentos Computacionais para o PECPL

4.2.1 Conjunto MIPLIB

O conjunto MIPLIB apresenta um total de 399 instâncias, as quais variam o número de vértices entre 7 e 8055, enquanto que o número de arestas varia entre 6 e 91459. A Tabela 4.2 apresenta um resumo comparativo do tempo total (em segundos) despendido para a execução de todas as instâncias do conjunto, utilizando cada uma das técnicas de resolução implementadas. A tabela ilustra também qual foi o menor (Tmin) e o maior (Tmax) tempo despendido na execução das instâncias do conjunto. Por fim, a última coluna da tabela apresenta a quantidade de instâncias que excederam o tempo limite de execução (TLE).

Tabela 4.2: Sumário dos resultados obtidos para as instâncias MIPLIB

Algoritmo	**Tempo total**	**TMax**	**TMin**	**TLE**
BK Versão 1	354,27	TLE	0,001	1
BK Versão 2	385,45	TLE	0,001	2
BK Versão 3	106,50	18,122	0,001	0
BK Versão 4	65,25	11,763	0,001	0
BK Versão 5	69,76	11,577	0,001	0
BK Versão 6[1]	48,54	8,679	0,001	0
BK Pivoteamento 1	68,96	11,238	0,001	0
BK Pivoteamento 2	50,42	9,008	0,001	0
BK Pivoteamento 3	53,79	9,158	0,001	0
BK Iterativo	160,98	52,976	0,001	94*
BK Paralelo	78,90	11,900	0,001	0
Ostergard	10259,84	TLE	0,001	169
Busca Tabu	2744,89	TLE	0,001	22
Simulated Annealing	2661,41	TLE	0,001	20

Os resultados obtidos para o conjunto MIPLIB comprovam a eficiência do algoritmo de Bron-Kerbosch, principalmente depois das melhorias propostas: a título de com-

[1]Essa versão utilizou o parâmetro de otimização **-03** na compilação do algoritmo.

paração, o algoritmo de Bron-Kerbosch Versão 1 despendeu 354,27 segundos para a execução das 399 instâncias do conjunto, ao passo que o algoritmo de Bron-Kerbosch Versão 6[1] despendeu apenas 48,54 segundos. Os métodos de pivoteamento implementados para a escolha do vértice pivô do algoritmo supracitado não contribuíram para uma melhoria no tempo computacional gasto. Além disso, as versões iterativa e paralela do algoritmo obtiveram resultados inferiores ao algoritmo de Bron-Kerbosch Versão 6[1]. Quando a análise é feita analisando o desempenho do algoritmo de Ostergard, pode-se perceber que a adaptação do mesmo para refletir o PECPL levou a uma queda considerável de desempenho. Na próxima seção, o algoritmo original de Ostergard será utilizado para a resolução do PCPM. As heurísticas utilizadas - heurística busca tabu com multi-vizinhanças e heurística *Simulated Annealing* - também tiveram um ruim desempenho para a resolução do problema, mesmo ao relaxar o problema, ao considerar como objetivo encontrar 60% do peso total (solução ótima).

A Tabela 4.3 apresenta as características de um subgrupo de instâncias do conjunto MIPLIB. Esse subgrupo contém apenas as instâncias que levaram um tempo superior a um segundo para serem executadas, quando da utilização do Algoritmo de Bron-Kerbosch Versão 4 para a resolução do problema. Tais instâncias são consideradas as mais críticas do conjunto MIPLIB. A primeira coluna da tabela apresenta o nome da instância. As informações do grafo (instância) são representadas pela quantidade de vértices (|V|) e pela quantidade de arestas (|A|). Outras informações pertinentes ao problema são mostradas na tabela, tais como o Menor Grau, Maior Grau, Menor Peso e o Maior Peso dentre os vértices do grafo.

Tabela 4.3: Características das instâncias do conjunto MIPLIB

Instância	|V|	|A|	Menor Grau	Maior Grau	Menor Peso	Maior Peso
eilB101_cut_21 (**Inst1**)	290	9844	36	102	1	530
eilB101_cut_27 (**Inst2**)	332	13266	40	124	1	458
eilB101_cut_28 (**Inst3**)	339	13934	42	130	1	450
eilB101_cut_29 (**Inst4**)	329	13022	44	124	1	418
eilB101_cut_30 (**Inst5**)	352	14968	41	145	1	412
eilB101_cut_31 (**Inst6**)	370	16534	51	137	1	487
eilB101_cut_32 (**Inst7**)	360	15833	42	157	1	473
eilB101_cut_33 (**Inst8**)	370	16557	48	149	1	436
eilB101_cut_34 (**Inst9**)	380	17798	45	172	1	494
eilB101_cut_35 (**Inst10**)	393	19075	55	153	1	458
eilB101_cut_36 (**Inst11**)	391	18861	48	173	1	444

A Tabela 4.4 apresenta um comparativo dos resultados obtidos pelas técnicas na

execução de um subgrupo de 11 instâncias do conjunto MIPLIB. O tempo computacional gasto está expresso em segundos. Valores TLE na tabela indicam que o algoritmo não encontrou a solução ótima para a instância dentro do tempo limite de execução. Por sua vez, valores (*) indicam estouro de memória na execução da instância.

Tabela 4.4: Resultados obtidos para as instâncias MIPLIB

Algoritmo	Inst1	Inst2	Inst3	Inst4	Inst5	Inst6	Inst7	Inst8	Inst9	Inst10	Inst11
BK Versão 1	1,865	3,996	4,484	4,363	8,055	14,310	13,773	19,230	37,177	22,349	16,333
BK Versão 2	4,462	10,216	8,939	11,592	16,607	35,269	23,205	40,400	TLE	59,037	50,042
BK Versão 3	1,473	2,757	2,645	3,510	4,812	9,318	6,918	11,157	18,122	14,993	13,482
BK Versão 4	1,035	1,716	2,190	1,370	4,163	4,024	3,426	11,032	6,134	5,469	11,763
BK Versão 5	1,009	1,716	2,175	1,353	4,129	3,930	3,336	10,639	7,188	5,401	11,577
BK Versão 6[1]	0,758	1,256	1,624	1,007	3,049	2,913	2,525	8,050	4,422	3,974	8,679
BK Iterativo	1,633	3,747	13,279	8,232	7,193	*	*	11,481	13,484	*	52,976
BK Paralelo	1,301	2,321	3,591	2,392	4,501	3,964	3,262	11,718	9,922	8,182	11,900
BK Pivoteamento 1	1,339	1,039	1,936	1,028	3,703	2,460	3,615	7,394	7,098	11,238	9,715
BK Pivoteamento 2	0,815	1,327	1,708	1,095	3,165	3,06	2,672	8,282	4,542	4,062	9,008
BK Pivoteamento 3	1,007	1,559	1,804	1,259	3,648	3,048	2,775	9,158	4,482	3,979	8,743
Ostergard	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE
Busca Tabu	38,055	45,443	36,063	39,866	TLE	TLE	TLE	TLE	TLE	TLE	TLE
Simulated Annealing	24,888	46,669	28,936	29,146	TLE	TLE	TLE	TLE	TLE	TLE	TLE

4.2.2 Conjunto INRC

O conjunto INRC apresenta um total de 3804 instâncias, as quais variam o número de vértices entre 174 e 3660, enquanto que o número de arestas varia entre 255 e 19421. A Tabela 4.5 apresenta um resumo comparativo do tempo total (em segundos) despendido para a execução de todas as instâncias do conjunto, utilizando cada uma das técnicas de resolução implementadas. A tabela informa também qual foi o menor (Tmin) e o maior (Tmax) tempo despendido na execução das instâncias do conjunto. Por fim, a última coluna da tabela apresenta a quantidade de instâncias que excederam o tempo limite de execução (TLE).

Os resultados obtidos para o conjunto INRC comprovam a eficiência do algoritmo de Bron-Kerbosch, principalmente depois das melhorias propostas: a título de comparação, o algoritmo de Bron-Kerbosch Versão 1 despendeu 257,00 segundos para a execução das 3804 instâncias do conjunto, ao passo que o algoritmo de Bron-Kerbosch Versão 6[1] despendeu apenas 39,06 segundos. O algoritmo de Bron-Kerbosch Pivoteamento 1 e o algoritmo de Bron-kerbosch Pivoteamento 3 não contribuíram para uma melhoria no

[1]Essa versão utilizou o parâmetro de otimização **-03** na compilação do algoritmo.

[1]Essa versão utilizou o parâmetro de otimização **-03** na compilação do algoritmo.

Tabela 4.5: Sumário dos resultados obtidos para as instâncias INRC

Algoritmo	**Tempo total**	**TMax**	**TMin**	**TLE**
BK Versão 1	257,00	1,155	0,001	0
BK Versão 2	160,13	0,517	0,001	0
BK Versão 3	58,02	0,189	0,001	0
BK Versão 4	54,17	0,196	0,001	0
BK Versão 5	65,63	0,228	0,001	0
BK Versão 6[1]	39,06	0,132	0,001	0
BK Pivoteamento 1	46,86	0,147	0,001	0
BK Pivoteamento 2	39,04	0,132	0,001	0
BK Pivoteamento 3	45,12	0,149	0,001	0
BK Iterativo	282,06	0,853	0,002	0
BK Paralelo	46,91	0,135	0,001	0
Ostergard	61535,29	TLE	0,001	921
Busca Tabu	43338,81	TLE	0,001	399
Simulated Annealing	4422,62	21,232	0,001	0

tempo computacional gasto, ao passo que o algoritmo de Bron-Kerbosch Pivoteamento 2 obteve desempenho similar ao algoritmo de Bron-Kerbosch Versão 6[1]. Além disso, as versões iterativa e paralela do algoritmo obtiveram resultados inferiores ao algoritmo de Bron-Kerbosch Versão 6[1]. Quando a análise é feita em relação ao desempenho do algoritmo de Ostergard, pode-se perceber que a adaptação do mesmo para refletir o PECPL levou a uma queda considerável de desempenho, sendo que cerca de 25% do total de instâncias não foram resolvidas em sua otimalidade durante o tempo limite de execução (TLE). Quanto às heurísticas utilizadas - heurística busca tabu com multi-vizinhanças e heurística *Simulated Annealing*, a última obteve um desempenho consideravelmente melhor que a primeira, ao encontrar a solução ótima relaxada para todas as instâncias em um tempo consideravelmente menor.

A Tabela 4.6 apresenta as características de um subgrupo de instâncias do conjunto INRC. Esse subgrupo contém apenas as instâncias que levaram um tempo superior a 0,170 segundos para serem executadas, quando da utilização do Algoritmo de Bron-Kerbosch versão 4 para a resolução do problema. Tais instâncias são consideradas as mais críticas do conjunto INRC. A primeira coluna da tabela apresenta o nome da instância. As informações do grafo (instância) são representadas pela quantidade de

vértices (|V|) e pela quantidade de arestas (|A|). Outras informações pertinentes ao problema são mostradas na tabela, tais como o Menor Grau, Maior Grau, Menor Peso e o Maior Peso dentre os vértices do grafo.

Tabela 4.6: Características das instâncias do conjunto INRC

Instância	|V|	|A|	Menor Grau	Maior Grau	Menor Peso	Maior Peso
long_hidden_02_cut_15 (**Inst1**)	3323	17426	1	32	1	993
long_hidden_02_cut_16 (**Inst2**)	3415	18337	1	32	1	995
long_hidden_02_cut_18 (**Inst3**)	3443	19084	1	34	1	988
long_hidden_03_cut_15 (**Inst4**)	3660	19209	1	32	1	997
long_hidden_03_cut_16 (**Inst5**)	3645	18719	1	31	1	998
long_hidden_03_cut_17 (**Inst6**)	3644	18890	1	34	1	997
long_hidden_03_cut_18 (**Inst7**)	3657	19120	1	35	1	991
long_hidden_05_cut_16 (**Inst8**)	3428	18570	1	35	1	998
long_hidden_05_cut_17 (**Inst9**)	3380	17668	1	33	1	993
long_hidden_05_cut_18 (**Inst10**)	3507	19421	1	35	1	991
long_late_03_cut_13 (**Inst11**)	3594	16443	1	29	1	998
long_late_03_cut_17 (**Inst12**)	3648	17725	1	30	1	997
long_late_03_cut_18 (**Inst13**)	3598	17676	1	30	1	996

A Tabela 4.7 apresenta um comparativo dos resultados obtidos pelas técnicas na execução de um subgrupo de 13 instâncias do conjunto INRC. O tempo computacional gasto estão expresso em segundos. Valores TLE na tabela indicam que o algoritmo não encontrou a solução ótima para a instância dentro do tempo limite de execução.

Tabela 4.7: Resultados obtidos para as instâncias INRC

Algoritmo	Inst1	Inst2	Inst3	Inst4	Inst5	Inst6	Inst7	Inst8	Inst9	Inst10	Inst11	Inst12	Inst13
BK Versão 1	0,786	0,832	0,449	0,755	0,742	0,767	0,757	0,828	0,765	0,801	0,688	0,752	0,754
BK Versão 2	0,380	0,414	0,446	0,454	0,452	0,456	0,456	0,425	0,401	0,446	0,397	0,433	0,430
BK Versão 3	0,158	0,173	0,189	0,166	0,163	0,167	0,165	0,178	0,160	0,176	0,148	0,162	0,161
BK Versão 4	0,173	0,187	0,196	0,187	0,182	0,185	0,184	0,182	0,172	0,192	0,173	0,182	0,178
BK Versão 5	0,185	0,199	0,204	0,228	0,220	0,222	0,223	0,202	0,193	0,214	0,202	0,214	0,207
BK Versão 6[1]	0,109	0,116	0,119	0,132	0,128	0,129	0,129	0,118	0,113	0,124	0,122	0,127	0,122
BK Iterativo	0,803	0,782	0,739	0,830	0,817	0,816	0,819	0,713	0,687	0,741	0,917	0,880	0,842
BK Paralelo	0,116	0,122	0,128	0,135	0,132	0,133	0,134	0,123	0,117	0,130	0,122	0,128	0,124
BK Pivoteamento 1	0,124	0,133	0,142	0,147	0,142	0,145	0,146	0,136	0,128	0,142	0,132	0,140	0,136
BK Pivoteamento 2	0,109	0,116	0,119	0,132	0,127	0,129	0,129	0,118	0,113	0,124	0,123	0,127	0,123
BK Pivoteamento 3	0,126	0,136	0,143	0,149	0,143	0,146	0,147	0,138	0,130	0,142	0,134	0,142	0,138
Ostergard	0,386	0,312	1,067	0,152	0,146	0,180	0,171	0,303	0,231	0,305	0,123	0,166	0,166
Busca Tabu	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE	TLE
Simulated Annealing	16,285	13,589	13,829	15,295	18,393	18,025	15,670	16,320	15,737	4,239	19,536	17,039	17,845

[1]Essa versão utilizou o parâmetro de otimização **-03** na compilação do algoritmo.

4.2.3 Conjunto Telebus

O conjunto Telebus apresenta um total de 3085 instâncias, as quais variam o número de vértices entre 13 e 1082, enquanto que o número de arestas varia entre 14 e 18323. A Tabela 4.8 apresenta um resumo comparativo do tempo total (em segundos) despendido para a execução de todas as instâncias do conjunto, utilizando cada uma das técnicas de resolução implementadas. A tabela informa também qual foi o menor (Tmin) e o maior (Tmax) tempo despendido na execução das instâncias do conjunto. Por fim, a última coluna da tabela apresenta a quantidade de instâncias que excederam o tempo limite de execução (TLE).

Tabela 4.8: Sumário dos resultados obtidos para as instâncias Telebus

Algoritmo	**Tempo total**	**TMax**	**TMin**	**TLE**
BK Versão 1	193,46	0,218	0,001	0
BK Versão 2	224,59	0,254	0,001	0
BK Versão 3	44,39	0,044	0,001	0
BK Versão 4	39,17	0,047	0,001	0
BK Versão 5	61,05	0,060	0,001	0
BK Versão 6[1]	34,34	0,033	0,001	0
BK Pivoteamento 1	40,63	0,049	0,001	0
BK Pivoteamento 2	34,41	0,047	0,001	0
BK Pivoteamento 3	37,41	0,044	0,001	0
BK Iterativo	145,81	0,140	0,001	0
BK Paralelo	44,69	0,045	0,001	0
Ostergard	69886,86	TLE	0,001	888
Busca Tabu	11867,61	TLE	0,001	8
Simulated Annealing	1305,41	14,635	0,001	0

Os resultados obtidos para o conjunto Telebus comprovam a eficiência do algoritmo de Bron-Kerbosch, principalmente depois das melhorias propostas: a título de comparação, o algoritmo de Bron-Kerbosch Versão 1 despendeu 193,46 segundos para a execução das 3085 instâncias do conjunto, ao passo que o algoritmo de Bron-Kerbosch Versão 6[1] despendeu apenas 34,34 segundos. O algoritmo de Bron-Kerbosch Pivoteamento 1 e o

[1]Essa versão utilizou o parâmetro de otimização **-03** na compilação do algoritmo.

algoritmo de Bron-kerbosch Pivoteamento 3 não contribuíram para uma melhoria no tempo computacional gasto, ao passo que o algoritmo de Bron-Kerbosch Pivoteamento 2 obteve desempenho similar ao algoritmo de Bron-Kerbosch Versão 6[1]. Além disso, as versões iterativa e paralela do algoritmo obtiveram resultados inferiores ao algoritmo de Bron-Kerbosch Versão 6[1]. Quando a análise é feita considerando o desempenho do algoritmo de Ostergard, pode-se perceber que a adaptação do mesmo para refletir o PECPL levou a uma queda considerável de desempenho. Quanto às heurísticas utilizadas - heurística busca tabu com multi-vizinhanças e heurística *Simulated Annealing*, a última encontrou a solução ótima relaxada para todas as instâncias, ao passo que a primeira excedeu o tempo limite de execução para oito dessas instâncias. O tempo computacional gasto para a resolução das instâncias utilizando a heurística *Simulated Annealing* foi consideravelmente menor que a heurística busca tabu com multi-vizinhanças (1305,41 segundos $\times$ 11867,61 segundos).

A Tabela 4.9 apresenta as características de um subgrupo de instâncias do conjunto Telebus. Esse subgrupo contém apenas as instâncias que levaram um tempo superior a 0,037 segundos para serem executadas, quando foi usado o Algoritmo de Bron-Kerbosch Versão 4 para a resolução do problema. Tais instâncias são consideradas as mais críticas do conjunto Telebus. A primeira coluna da tabela apresenta o nome da instância. As informações do grafo (instância) são representadas pela quantidade de vértices (|V|) e pela quantidade de arestas (|A|). Outras informações pertinentes ao problema são mostradas na tabela, tais como o Menor Grau, Maior Grau, Menor Peso e o Maior Peso dentre os vértices do grafo.

Tabela 4.9: Características das instâncias do conjunto Telebus

Instância	**|V|**	**|A|**	**Menor Grau**	**Maior Grau**	**Menor Peso**	**Maior Peso**
t0415_cut_1 (**Inst1**)	776	8572	1	55	1	998
t0418_cut_29 (**Inst2**)	1069	17754	1	75	1	996
t0418_cut_36 (**Inst3**)	1075	18111	1	79	1	996
t0418_cut_37 (**Inst4**)	1074	18026	1	77	1	998
t0418_cut_66 (**Inst5**)	1073	18000	1	79	1	995

A Tabela 4.10 apresenta um comparativo dos resultados obtidos pelas técnicas na execução de um subgrupo de cinco instâncias do conjunto Telebus. O tempo computacional gasto estão expresso em segundos. Valores TLE na tabela indicam que o algoritmo não encontrou a solução ótima para a instância dentro do tempo limite de execução.

[1]Essa versão utilizou o parâmetro de otimização **-03** na compilação do algoritmo.

Tabela 4.10: Resultados obtidos para as instâncias Telebus

Algoritmo	**Inst1**	**Inst2**	**Inst3**	**Inst4**	**Inst5**
BK Versão 1	0,079	0,211	0,213	0,214	0,210
BK Versão 2	0,071	0,246	0,251	0,248	0,248
BK Versão 3	0,017	0,044	0,044	0,043	0,043
BK Versão 4	0,047	0,038	0,038	0,038	0,038
BK Versão 5	0,023	0,129	0,059	0,059	0,059
BK Versão 6[1]	0,015	0,093	0,033	0,033	0,033
BK Iterativo	0,140	0,126	0,129	0,126	0,125
BK Paralelo	0,020	0,041	0,043	0,044	0,041
BK Pivoteamento 1	0,047	0,037	0,041	0,039	0,042
BK Pivoteamento 2	0,046	0,033	0,033	0,033	0,033
BK Pivoteamento 3	0,044	0,035	0,037	0,036	0,036
Ostergard	0,014	0,585	TLE	15,937	TLE
Busca Tabu	5,795	9,265	4,317	18,167	3,857
Simulated Annealing	1,342	0,275	0,305	0,119	0,084

4.2.4 Conjunto Uchoa

O conjunto Uchoa apresenta um total de quatro instâncias. As características de cada instância são apresentadas na Tabela 4.11. A primeira coluna da tabela apresenta o nome da instância. As informações do grafo (instância) são representadas pela quantidade de vértices (|V|) e pela quantidade de arestas (|A|). Outras informações pertinentes ao problema são mostradas na tabela, tais como o Menor Grau, Maior Grau, Menor Peso e o Maior Peso dentre os vértices do grafo.

Tabela 4.11: Características das instâncias do conjunto Uchoa

Instância	**|V|**	**|A|**	**Menor Grau**	**Maior Grau**	**Menor Peso**	**Maior Peso**
pdistuchoa_cut_1 (**Inst1**)	500	26314	14	320	500	500
pdistuchoa_cut_2 (**Inst2**)	310	5534	5	81	1	746
pdistuchoa_cut_3 (**Inst3**)	371	9929	7	114	4	788
pdistuchoa_cut_4 (**Inst4**)	428	15430	7	172	2	797

A Tabela 4.12 apresenta um comparativo dos resultados obtidos pelas técnicas implementadas no trabalho, na execução das quatro instâncias do conjunto Uchoa. O tempo

computacional gasto está expresso em segundos. Valores TLE na tabela indicam que o algoritmo não encontrou a solução ótima para a instância dentro do tempo limite de execução. Por sua vez, valores (*) indicam estouro de memória na execução da instância.

Tabela 4.12: Resultados obtidos para as instâncias Uchoa

Algoritmo	**Inst1**	**Inst2**	**Inst3**	**Inst4**	**Tempo total**
BK Versão 1	TLE	0,109	0,436	1,217	61,763
BK Versão 2	TLE	0,073	0,412	1,633	62,137
BK Versão 3	23,977	0,033	0,129	0,360	24,499
BK Versão 4	18,379	0,029	0,104	0,245	18,757
BK Versão 5	16,980	0,027	0,098	0,243	17,348
BK Versão 6[1]	14,109	0,021	0,076	0,187	14,393
BK Pivoteamento 1	14,999	0,022	0,080	0,200	15,301
BK Pivoteamento 2	14,836	0,021	0,076	0,191	15,124
BK Pivoteamento 3	15,023	0,022	0,077	0,192	15,314
BK Iterativo	45,104	0,060	0,245	0,672	46,081
BK Paralelo	19,612	0,029	0,105	0,251	19,997
Ostergard	TLE	0,598	2,930	9,334	72,862
Busca Tabu	TLE	15,909	29,297	45,168	150,374
Heurística *Simulated Annealing*	TLE	8,560	23,771	TLE	152,331

Os resultados obtidos para o conjunto Uchoa comprovam a eficiência do algoritmo de Bron-Kerbosch, principalmente depois das melhorias propostas: a título de comparação, o algoritmo de Bron-Kerbosch Versão 1 despendeu 61,763 segundos para a execução das quatro instâncias do conjunto, ao passo que o algoritmo de Bron-Kerbosch Versão 6[1] despendeu apenas 14,393 segundos. Os métodos de pivoteamento implementados para a escolha do vértice pivô do algoritmo supracitado apresentaram desempenho similar ao desempenho do algoritmo de Bron-Kerbosch Versão 6[1]. Além disso, as versões iterativa e paralela do algoritmo obtiveram resultados inferiores ao algoritmo de Bron-Kerbosch Versão 6[1]. A instância pdistuchoa_cut_1, que é constituída de um grafo contendo 500 vértices e 26314 arestas, foi considerada a instância mais crítica dentre o total de 7292 instâncias utilizadas nos experimentos computacionais, provenientes dos quatro conjuntos de instâncias MIPLIB, INRC, Telebus e Uchoa. Para essa instância, o algoritmo de Ostergard e as heurísticas implementadas no presente trabalho não encontraram a solução ótima dentro do tempo limite de execução. Quanto ao conjunto Uchoa como um todo, as heurísticas tiveram desempenhos similares (150,374 segundos $\times$ 152,331 segundos).

[1]Essa versão utilizou o parâmetro de otimização **-03** na compilação do algoritmo.

4.3 Análise e Apresentação dos Experimentos Computacionais para o PCPM

Nessa seção, os experimentos computacionais realizados para a resolução do problema do clique de peso máximo são apresentados. Para o PCPM, foram utilizados os seguintes algoritmos: algoritmo de Bron-Kerbosch Versão 6 - a qual obteve os melhores resultados na resolução do problema da enumeração de cliques com peso acima de um limiar; o algoritmo de Bron-Kerbosch com os três métodos de pivoteamento; o algoritmo de Ostergard e a heurística busca tabu com multi-vizinhanças, onde as implementações foram gentilmente disponibilizadas pelos autores e; a heurística *Simulated Annealing*.

A Tabela 4.13 apresenta o tempo computacional gasto (em segundos) para a execução das instâncias do conjunto MIPLIB, o tempo de execução da instância mais crítica (TMax) e da instância menos crítica (TMin). Além disso, na coluna TLE, é apresentada a quantidade de instâncias que não obtiveram a solução ótima dentro do tempo limite de execução. Cada linha da tabela apresenta o desempenho de um dos algoritmos implementados.

Tabela 4.13: Sumário dos resultados obtidos para as instâncias MIPLIB

Algoritmo	**Tempo total**	**TMax**	**TMin**	**TLE**
BK Versão 6[1]	6,122	0,342	0,001	0
BK Pivoteamento 1	8,146	0,768	0,001	0
BK Pivoteamento 2	6,128	0,361	0,001	0
BK Pivoteamento 3	6,371	0,362	0,001	0
Ostergard	2,751	0,191	0,001	0
Busca Tabu	0,174	0,063	0,001	0
Simulated Annealing	1657,532	TLE	0,001	16

Os resultados obtidos para o conjunto MIPLIB comprovam a eficiência das técnicas implementadas, visto que para um total de 399 instâncias, com exceção da heurística *Simulated Annealing*, todas as outras técnicas encontraram a solução ótima para todas as instâncias em um tempo inferior a nove segundos. A heurística *Simulated Annealing* teve um desempenho bem pior quando comparada às outras técnicas, mas, ainda sim, encontrou a solução ótima para quase todas as instâncias: apenas 16 delas não foram resolvidas

[1]Essa versão utilizou o parâmetro de otimização **-03** na compilação do algoritmo.

até a sua otimalidade. Quanto às versões utilizadas do algoritmo de Bron-Kerbosch, todas obtiveram um desempenho similar. O algoritmo de Ostergard e a heurística busca tabu com multi-vizinhanças foram considerados os algoritmos mais eficientes, ao usar respectivamente, 2,751 segundos e 0,174 segundos de tempo de computação. Cabe ressaltar que, para as heurísticas, a solução ótima (clique de peso máximo) é utilizada no processamento das mesmas, evitando processamentos adicionais, como pode ser o caso dos algoritmos exatos.

A Tabela 4.14 apresenta o tempo computacional gasto (em segundos) para a execução das instâncias do conjunto INRC, o tempo de execução da instância mais crítica (TMax) e da instância menos crítica (TMin). Além disso, na coluna TLE, é apresentada a quantidade de instâncias que não obtiveram a solução ótima dentro do tempo limite de execução. Cada linha da tabela apresenta o desempenho de um dos algoritmos implementados.

Tabela 4.14: Sumário dos resultados obtidos para as instâncias INRC

Algoritmo	**Tempo total**	**TMax**	**TMin**	**TLE**
BK Versão 6[1]	29,180	0,097	0,001	0
BK Pivoteamento 1	36,239	0,128	0,001	0
BK Pivoteamento 2	29,303	0,100	0,001	0
BK Pivoteamento 3	29,310	0,096	0,001	0
Ostergard	22,816	0,081	0,001	0
Busca Tabu	4,528	0,216	0,001	0
Simulated Annealing	4802,684	56,282	0,001	0

Os resultados obtidos para o conjunto INRC comprovam a eficiência das técnicas implementadas, visto que para um total de 3804 instâncias, todas as técnicas encontraram a solução ótima para todas as instâncias dentro do tempo limite de execução (TLE). Pode-se observar que a utilização do algoritmo de Bron-Kerbosch Pivoteamento 1 teve uma perda de desempenho considerável ao se comparar com o tempo computacional de execução das outras versões do mesmo algoritmo. A heurística busca tabu com multi-vizinhanças foi consideravelmente mais eficiente do que as outras técnicas, ao encontrar a solução ótima para todas as instâncias em um tempo inferior a cinco segundos. Cabe ressaltar que, para as heurísticas, a solução ótima (clique de peso máximo) é

[1]Essa versão utilizou o parâmetro de otimização **-03** na compilação do algoritmo.

utilizada no processamento das mesmas, evitando processamentos adicionais, como pode ser o caso dos algoritmos exatos. Considerando a quantidade de instâncias do conjunto INRC, é totalmente justificável a diferença de desempenho da heurística busca tabu com multi-vizinhanças e dos algoritmos exatos.

A Tabela 4.15 apresenta o tempo computacional gasto (em segundos) para a execução das instâncias do conjunto Telebus, o tempo de execução da instância mais crítica (TMax) e da instância menos crítica (TMin). Além disso, na coluna TLE, é apresentada a quantidade de instâncias que não obtiveram a solução ótima dentro do tempo limite de execução. Cada linha da tabela apresenta o desempenho de um dos algoritmos implementados.

Tabela 4.15: Sumário dos resultados obtidos para as instâncias Telebus

Algoritmo	**Tempo total**	**TMax**	**TMin**	**TLE**
BK Versão 6[1]	33,131	0,033	0,001	0
BK Pivoteamento 1	39,690	0,050	0,001	0
BK Pivoteamento 2	33,491	0,044	0,001	0
BK Pivoteamento 3	33,227	0,044	0,001	0
Ostergard	8,787	0,018	0,001	0
Busca Tabu	2,721	0,038	0,001	0
Simulated Annealing	994,846	18,003	0,001	0

Os resultados obtidos para o conjunto Telebus comprovam a eficiência das técnicas implementadas, visto que para um total de 3085 instâncias, todas as técnicas encontraram a solução ótima para todas as instâncias dentro do tempo limite de execução (TLE). Pode-se observar que a utilização do algoritmo de Bron-Kerbosch Pivoteamento 1 teve uma perda de desempenho considerável ao se comparar com o tempo computacional de execução das outras versões do mesmo algoritmo. A heurística busca tabu com multi-vizinhanças foi consideravelmente mais eficiente do que as outras técnicas, ao encontrar a solução ótima para todas as instâncias em um tempo inferior a três segundos, ao passo que o algoritmo de Ostergard gastou aproximados nove segundos e o algoritmo de Bron-Kerbosch despendeu 33 segundos. A heurística *Simulated Annealing* teve um desempenho bem pior quando comparada às outras técnicas, mas, ainda sim, encontrou a solução ótima para todas as instâncias, sendo que a instância mais crítica foi resolvida

[1]Essa versão utilizou o parâmetro de otimização **-03** na compilação do algoritmo.

em 18 segundos.

A Tabela 4.16 apresenta o tempo computacional gasto (em segundos) para a execução das instâncias do conjunto Uchoa, o tempo de execução da instância mais crítica (TMax) e da instância menos crítica (TMin). Além disso, na coluna TLE, é apresentada a quantidade de instâncias que não obtiveram a solução ótima dentro do tempo limite de execução. Cada linha da tabela apresenta o desempenho de um dos algoritmos implementados.

Tabela 4.16: Sumário dos resultados obtidos para as instâncias Uchoa

Algoritmo	**Tempo total**	**TMax**	**TMin**	**TLE**
BK Versão 6[1]	0,812	0,750	0,005	0
BK Pivoteamento 1	0,833	0,768	0,006	0
BK Pivoteamento 2	0,801	0,740	0,005	0
BK Pivoteamento 3	0,821	0,759	0,005	0
Ostergard	0,067	0,055	0,002	0
Busca Tabu	0,004	0,001	0,001	0
Simulated Annealing	17,013	8,960	0,063	0

Os resultados obtidos para as quatro instâncias do conjunto Uchoa comprovam a eficiência das técnicas implementadas, visto que todas as técnicas encontraram a solução ótima para as instâncias dentro do tempo limite de execução (**TLE**). Com exceção da heurística *Simulated Annealing*, todas as outras técnicas despenderam menos de um segundo para a execução de todas as instâncias. O algoritmo de Ostergard e a heurística busca tabu com multi-vizinhanças foram considerados os algoritmos mais eficientes, ao usar respectivamente, 0,067 segundos e 0,004 segundos de tempo de computação. A heurística *Simulated Annealing* teve um desempenho bem pior quando comparada às outras técnicas, mas, ainda sim, encontrou a solução ótima para todas as instâncias - em um tempo total de 17,013 segundos, sendo que a instância mais crítica foi resolvida em 8,960 segundos.

[1]Essa versão utilizou o parâmetro de otimização **-03** na compilação do algoritmo.

Capítulo 5

Conclusões

O problema da enumeração de cliques com peso acima de um limiar foi apresentado e discutido no presente trabalho. Neste problema, o objetivo é encontrar todos os cliques maximais com peso acima de um limiar, valor este previamente informado. Para a resolução do PECPL foram utilizados três algoritmos presentes na literatura, com algumas adaptações propostas e, um algoritmo inédito na literatura para a resolução desse problema. Esses algoritmos são aplicados na separação de cortes no contexto de programação inteira. Encontrar todos os cliques acima de um dado peso é equivalente ao problema de encontrar todas as desigualdades violadas de clique. Para o algoritmo de Bron-Kerbosch foram implementadas oito versões, as quais se diferem na forma de armazenamento do grafo e na forma de seleção dos vértices pivôs. Além disso, também foi proposta uma versão iterativa desse algoritmo, originalmente recursivo, e por fim, uma versão paralela. O algoritmo exato de Ostergard, a heurística busca tabu com multi-vizinhanças (Wu et al. (2012)) e a heurística *Simulated Annealing* também foram implementados.

A fim de se obter uma comparação entre a eficiência das técnicas implementadas, um total de 7292 instâncias, provenientes de quatro conjuntos de problemas de otimização combinatória, foram usadas. Um tempo limite de um minuto foi estipulado como critério de parada para a execução de cada instância. Ao se fazer um comparativo da eficiência dos quatro algoritmos implementados, percebe-se que o algoritmo de Bron-Kerbosch obteve o melhor desempenho, ao encontrar a solução ótima para todas as instâncias dentro do tempo limite de execução. Credita-se esse bom desempenho à utilização adequada de estruturas e pivoteamento para a representação dos conjuntos pertencentes ao algoritmo. Além disso, dentre as técnicas implementadas, o algoritmo de Bron-Kerbosch

é o único que não necessita de verificações adicionais para a resolução do problema. No algoritmo de Ostergard, uma verificação adicional é necessária para garantir que não existam cliques dominados (não maximais) por outros cliques no conjunto solução e, para as heurísticas busca tabu com multi-vizinhanças e *Simulated Annealing*, é necessário verificar se não existem cliques repetidos, haja visto que por se tratar de uma heurística, uma subárvore pode ser explorada várias vezes.

Quando a comparação é direcionada às versões implementadas do algoritmo de Bron-Kerbosch, é importante observar que cada otimização de uma das operações, seja ela a verificação de aresta entre dois vértices, a união ou interseção, levou a uma melhoria considerável no tempo de execução das instâncias. Além disso, como uma estratégia para redução do tempo computacional gasto, um parâmetro de otimização (**-O3**) foi utilizado na compilação do algoritmo e obteve uma redução de até 30% no tempo total de execução de cada conjunto de instância.

Uma versão iterativa do algoritmo de Bron-Kerbosch também foi proposta, porém, não obteve bons resultados quando comparado à versão recursiva. Além disso, houve estouro de memória para algumas instâncias, haja visto que a abordagem iterativa do algoritmo mantém uma pilha de chamadas, o que não acontece na versão recursiva. Por fim, uma versão paralela do algoritmo de Bron-Kerbosch foi implementada e não obteve uma aceleração linear, como era esperado, quando utilizado na resolução do problema da enumeração de cliques com peso acima de um limiar.

Por fim, foram realizados experimentos computacionais para a resolução do problema do clique de peso máximo. Todas as técnicas implementadas obtiveram bons resultados, sendo considerado a heurística busca tabu com multi-vizinhanças a de melhor desempenho dentre as técnicas. Para o PCPM, heurísticas podem funcionar de modo mais eficiente que algoritmos exatos, visto que não exploram todo o espaço de busca e, com o conhecimento da solução ótima, podem encerrar suas execuções assim que encontrarem o clique de peso máximo.

Durante o desenvolvimento do trabalho, principalmente em relação à resolução do problema da enumeração de cliques com peso acima de um limiar utilizando o algoritmo de Bron-Kerbosch, pode-se perceber que é de suma importância otimizar cada uma das operações, tais como a verificação de aresta entre vértices, união e interseção de conjuntos. Nesse sentido, algumas melhorias foram propostas e obtiveram melhoras consideráveis no tempo de execução. Como trabalho futuro, é sugerida a adoção de operadores lógicos para a representação do grafo e realização das operações de união e

interseção. Sugere-se também, como rumo de pesquisa futura, as seguintes estratégias:

- Implementação de outros algoritmos disponíveis na literatura, a fim de se obter uma comparação mais ampla dos métodos utilizados na literatura para a resolução do problema do clique e suas variações;
- Melhorar a versão paralela do algoritmo de Bron-Kerbosch. Haja visto a dificuldade encontrada em paralelizar um método voltado para a resolução do problema da enumeração de cliques com peso acima de um limiar, principalmente pelas questões levantadas no Capítulo 3, como árvores de pesquisa com retrocesso altamente irregular e/ou com uma diferença de tamanho considerável, sugere-se uma melhoria na versão paralela do Algoritmo de Bron-Kerbosch, implementada neste trabalho;
- Geração de instâncias maiores que as utilizadas no presente trabalho. As instâncias utilizadas neste trabalho foram geradas a partir de quatro conjuntos de problemas de otimização combinatória, sendo esses utilizados no problema da separação de cortes. Nesse sentido, as instâncias supracitadas foram utilizadas para permitir uma integração do problema resolvido neste trabalho e o problema da separação de cortes. Em trabalhos relacionados da literatura, alguns autores utilizam instâncias bem maiores, haja visto que pode não haver interesse na resolução do problema da separação de cortes e sim, na eficiência dos algoritmos de resolução do problema do clique e;
- Utilização da mineração de dados para a descoberta de padrões entre os cliques contidos na solução do problema. A solução ótima para cada instância do PECPL pode conter um clique ou milhões de cliques. Nesse sentido, a utilização da mineração de dados pode ajudar na descoberta de padrões interessantes entre os vértices que compõem os cliques, facilitando a resolução rápida do problema.

Referências Bibliográficas

Ben-Dor, A., Shamir, R. e Yakhini, Z.: 1999, Clustering gene expression patterns.

Borndorfer, R.: 1998, *Aspects of Set Packing, Partitioning, and Covering*, PhD thesis, Technische Universitat Berlin.

Borndörfer, R., Grötschel, M., Klostermeier, F. e Küttner, C.: 1999, Telebus berlin: Vehicle scheduling in a dial-a-ride system, *in* N. Wilson (ed.), *Computer-Aided Transit Scheduling*, Vol. 471 of *Lecture Notes in Economics and Mathematical Systems*, Springer Berlin Heidelberg, pp. 391–422.
URL: *http://dx.doi.org/10.1007/978-3-642-85970-0*$_1$9

Brito, S. e Santos, H. G.: 2011, Pivoteamento no Algoritmo Bron-Kerbosch para a Detecção de Cliques com Peso Máximo., *Anais do XLIII Simpósio Brasileiro de Pesquisa Operacional*, pp. 3052–3059.

Brito, S. S.: 2015, *Grafo de conflitos: Construção de aplicações em problemas de programação inteira*, Master's thesis, Universidade Federal de Ouro Preto.

Bron, C. e Kerbosch, J.: 1973, Algorithm 457: finding all cliques of an undirected graph, *Commun. ACM* **16**, 575–577.

Burke, E., Marecek, J., Parkes, A. e Rudová, H.: 2012, A branch-and-cut procedure for the udine course timetabling problem, *Annals of Operations Research* **194**, 71–87.
URL: *http://dx.doi.org/10.1007/s10479-010-0828-5*

Campêlo, M., Corrêa, R. e Frota, Y.: 2004, Cliques, holes and the vertex coloring polytope, *Information Processing Letters* **89**, 159 – 164.
URL: *http://www.sciencedirect.com/science/article/pii/S002001900300512X*

Chvátal, V.: 1973, Edmonds polytopes and a hierarchy of combinatorial problems, *Discrete mathematics* **4**, 305–337.

Coll, P., Marenco, J., Méndez Díaz, I. e Zabala, P.: 2002, Facets of the graph coloring polytope, *Annals of Operations Research* **116**, 79–90.

Cornuéjols, G.: 2007, Revival of the gomory cuts in the 1990's, *Annals of Operations Research* **149**, 63–66.

Debroni, J., Eblen, J. D., Langston, M. A., Myrvold, W., Shor, P. e Weerapurage, D.: 2011, A complete resolution of the keller maximum clique problem, *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, SIAM, pp. 129–135.
URL: *http://dl.acm.org/citation.cfm?id=2133036.2133047*

Fischetti, M. e Lodi, A.: 2005, Optimizing over the first chvàtal closure, *in* M. Jünger e V. Kaibel (eds), *Integer Programming and Combinatorial Optimization*, Vol. 3509 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 12–22.

Goldschmidt, O., Hochbaum, D. S., Hurkens, C. e Yu, G.: 1996, Approximation algorithms for the k-clique covering problem, *SIAM J. Discret. Math.* **9**, 492–509.
URL: *http://dx.doi.org/10.1137/S089548019325232X*

Gupta, R., Soffa, M. L. e Steele, T.: 1989, Register allocation via clique separators, *In Proceedings of the SIGPLAN '89 Conference on Programming Language Design and Implementation*, pp. 264–274.

Haspeslagh, S., De Causmaecker, P., Schaerf, A. e Stølevik, M.: 2014, The first international nurse rostering competition 2010, *Annals of Operations Research* **218**, 221–236.
URL: *http://dx.doi.org/10.1007/s10479-012-1062-0*

Juels, A. e Peinado, M.: 2000, Hiding cliques for cryptographic security, *Designs, Codes and Cryptography* **20**, 269–280.

Kirkpatrick, S., Gelatt, C. D. e Vecchi, M. P.: 1983, Optimization by simulated annealing, *Science* **220**, 671–680.
URL: *http://www.sciencemag.org/content/220/4598/671.abstract*

Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R., Danna, E., Gamrath, G., Gleixner, A., Heinz, S., Lodi, A., Mittelmann, H., Ralphs, T., Salvagnin, D., Steffy, D. e Wolter, K.: 2011, Miplib 2010, *Mathematical Programming Computation* **3**, 103–163.

Kumlander, D.: 2008, On importance of a special sorting in the maximum-weight clique algorithm based on colour classes, *in* H. Le Thi, P. Bouvry e T. Pham Dinh (eds), *Modelling, Computation and Optimization in Information Systems and Management Sciences*, Vol. 14 of *Communications in Computer and Information Science*, Springer Berlin Heidelberg, pp. 165–174.
URL: *http://dx.doi.org/10.1007/978-3-540-87477-5*$_1$8

Luce, R. e Perry, A.: 1949, A method of matrix analysis of group structure, *Psychometrika* **14**, 95–116.
URL: *http://dx.doi.org/10.1007/BF02289146*

Ludwig, L. e Gini, M.: 2006, Robotic swarm dispersion using wireless intensity signals, *in* M. Gini e R. Voyles (eds), *Distributed Autonomous Robotic Systems 7*, Springer Japan, pp. 135–144.
URL: *http://dx.doi.org/10.1007/4-431-35881-1*$_1$4

McCreesh, C. e Prosser, P.: 2012, Distributing an exact algorithm for maximum clique: maximising the costup, *CoRR* **abs/1209.4560**.
URL: *http://arxiv.org/abs/1209.4560*

Murphey, R. A., Pardalos, P. M. e Resende, M. G.: 1999, Frequency assignment problems, *Handbook of Combinatorial Optimization*, Kluwer Academic Publishers, pp. 295–377.

Pavan, M. e Pelillo, M.: 2003, A new graph-theoretic approach to clustering and segmentation, *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, Vol. 1, pp. I–145–I–152 vol.1.

Rhodes, N., Willett, P., Calvet, A., Dunbar, J. B. e Humblet, C.: 2003, Clip: Similarity searching of 3d databases using clique detection, *Journal of Chemical Information and Computer Sciences* **43**, 443–448. PMID: 12653507.
URL: *http://dx.doi.org/10.1021/ci025605o*

Samudrala, R., Moult, J. e Biology, C.: 1998, A graph-theoretic algorithm for comparative modeling of protein structure, *J Mol Biol* **279**, 279–287.

Santos, H. G., Toffolo, T. A., Gomes, R. A. e Ribas, S.: 2014, Integer programming techniques for the nurse rostering problem, *Annals of Operations Research* pp. 1–27.

Schmidt, M. C., Samatova, N. F., Thomas, K. e Park, B.-H.: 2009, A scalable, parallel algorithm for maximal clique enumeration, *Journal of Parallel and Distributed Computing* **69**, 417 – 428.
URL: *http://www.sciencedirect.com/science/article/pii/S0743731509000082*

Smith, C.: 1994, *A Recursive Introduction to the Theory of Computation*, 1st edn, Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Tomita, E., Tanaka, A. e Takahashi, H.: 2006, The worst-case time complexity for generating all maximal cliques and computational experiments, *Theoretical Computer Science* **363**, 28 – 42. Computing and Combinatorics 10th Annual International Conference on Computing and Combinatorics (COCOON 2004).
URL: *http://www.sciencedirect.com/science/article/pii/S0304397506003586*

Vassilevska, V.: 2009, Efficient algorithms for clique problems, *Information Processing Letters* **109**, 254 – 257.
URL: *http://www.sciencedirect.com/science/article/pii/S0020019008003293*

Wolsey, L. A.: 1998, *Integer Programming*, Wiley Series in Discrete Mathematics and Optimization, Wiley.

Wu, Q., Hao, J.-K. e Glover, F.: 2012, Multi-neighborhood tabu search for the maximum weight clique problem, *Annals of Operations Research* **196**, 611–634.
URL: *http://dx.doi.org/10.1007/s10479-012-1124-3*

Yamaguchi, K. e Masuda, S.: 2008, A New Exact Algorithm for the Maximum Weight Clique Problem, Vol. , THE INSTITUTE OF ELECTRONICS ENGINEERS OF KOREA, pp. 317–320.
URL: *http://www.dbpia.co.kr/Article/1577880*

Yu, G., Kouvelis, P. e Luo: 1992, A weighted vertex packing problem for specially structured geometric graphs arising in the design of electronic testing fixtures, ORSA/TIMS Presentation, San Francisco, CA.

Östergård, P. R.: 1999, A new algorithm for the maximum-weight clique problem, *Electronic Notes in Discrete Mathematics* **3**, 153 – 156. 6th Twente Workshop on Graphs and Combinatorial Optimization.
URL: *http://www.sciencedirect.com/science/article/pii/S1571065305800459*

Östergård, P. R.: 2002, A fast algorithm for the maximum clique problem, *Discrete Applied Mathematics* **120**, 197 – 207. Special Issue devoted to the 6th Twente Workshop on Graphs and Combinatorial Optimization.
URL: *http://www.sciencedirect.com/science/article/pii/S0166218X01002906*

Apêndice A

Publicações

O apêndice a seguir lista um artigo, desenvolvido durante a realização deste trabalho, aceito em um evento científico.

VILAS BOAS, M. G.; SANTOS, H. G.; BRITO, S. S. **Algoritmos Exatos e Heurísticos para o Problema da Detecção de Cliques com Peso acima de um Limiar**. *XLVII Simpósio Brasileiro de Pesquisa Operacional*, 2015, Porto de Galinhas, Brasil.

Apêndice B

Avaliação dos parâmetros utilizados na Heurística Simulated Annealing

Neste apêndice, são detalhados todos os experimentos realizados para a definição dos valores de cada parâmetro utilizado na heurística *Simulated Annealing*. Os parâmetros pertinentes à essa heurística são α, T_0 e $SAmax$, relacionados respectivamente à razão de resfriamento, à temperatura inicial e o número máximo de iterações em uma dada temperatura. Foram considerados três diferentes valores para cada parâmetro e todas as possíveis combinações entre eles foram testadas.

Nas próximas seções, são apresentados os experimentos computacionais realizados para cada um dos quatro conjuntos de instâncias (MIPLIB, INRC, Telebus e Uchoa). Tais experimentos consideraram apenas as instâncias consideradas mais críticas para cada conjunto, sendo essas descritas nas Tabelas 4.3, 4.6, 4.9 e 4.11. Conforme mencionado durante todo o projeto, o objetivo na execução da heurística *Simulated Annealing* é encontrar 60% do peso total (solução ótima) dentro de um tempo limite de execução (TLE) de 60 segundos. Nesse sentido, para as seções seguintes, são apresentadas tabelas com os resultados encontrados para cada instância de um dado conjunto supracitado e ao final do capítulo é apresentada uma tabela, sendo esta caracterizada por um sumário dos resultados obtidos, onde é realizada uma análise dos resultados e as principais conclusões são relatadas.

B.1 Parametrização para o conjunto MIPLIB

Para avaliar os parâmetros da heurística *Simulated Annealing* que serão utilizados na execução das 399 instâncias do conjunto MIPLIB, foram realizados experimentos com um subgrupo de instâncias desse conjunto, caracterizado na Tabela 4.3. Nas tabelas a seguir (Tabelas B.1-B.11), são apresentados, para cada instância do subgrupo, o tempo necessário para encontrar 60% do peso total (solução ótima), a porcentagem de peso e de cliques encontrada. Foi considerado como critério de parada um tempo limite de execução (TLE) de 60 segundos. Na primeira linha das tabelas, a solução ótima de cada instância é apresentada, constituída pela quantidade de cliques (|Cliques|) e do peso total (|Peso|).

Tabela B.1: Parâmetros SA: Instância eilB101_cut_21

Parâmetros	**TLE = 60s**	**|Cliques| = 2418**	**|Peso| = 2668146**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	39,626	59,39%	60,03%
(0,1; 10; |Cliques|)	60,000	47,06%	47,67%
(0,1; 100; |V|)	29,947	59,31%	60,03%
(0,1; 100; |Cliques|)	60,000	45,74%	46,44%
(0,1; 1020; |V|)	**23,504**	**59,39%**	**60,01%**
(0,1; 1020; |Cliques|)	60,000	43,38%	43,92%
(0,5; 10; |V|)	45,510	59,47%	60,03%
(0,5; 10; |Cliques|)	60,000	27,83%	28,32%
(0,5; 100; |V|)	35,613	59,31%	60,01%
(0,5; 100; |Cliques|)	60,000	27,71%	28,33%
(0,5; 1020; |V|)	37,066	59,26%	60,01%
(0,5; 1020; |Cliques|)	60,000	29,69%	30,38%
(0,9; 10; |V|)	60,000	33,09%	33,61%
(0,9; 10; |Cliques|)	60,000	7,24%	7,43%
(0,9; 100; |V|)	60,000	36,93%	37,60%
(0,9; 100; |Cliques|)	60,000	10,46%	10,87%
(0,9; 1020; |V|)	60,000	34,53%	35,25%
(0,9; 1020; |Cliques|)	60,000	9,88%	10,21%

Tabela B.2: Parâmetros SA: Instância eilB101_cut_27

Parâmetros	**TLE = 60s**	**|Cliques| = 2257**	**|Peso| = 2446649**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	54,221	59,24%	60,00%
(0,1; 10; |Cliques|)	60,000	37,22%	37,97%
(0,1; 100; |V|)	42,689	59,28%	60,00%
(0,1; 100; |Cliques|)	60,000	35,40%	36,14%
(0,1; 1020; |V|)	**40,260**	**59,19%**	**60,00%**
(0,1; 1020; |Cliques|)	60,000	33,63%	34,36%
(0,5; 10; |V|)	60,000	54,10%	54,89%
(0,5; 10; |Cliques|)	60,000	22,37%	22,87%
(0,5; 100; |V|)	60,000	52,06%	52,92%
(0,5; 100; |Cliques|)	60,000	23,26%	23,83%
(0,5; 1020; |V|)	60,000	52,15%	53,02%
(0,5; 1020; |Cliques|)	60,000	21,05%	21,56%
(0,9; 10; |V|)	60,000	24,28%	24,94%
(0,9; 10; |Cliques|)	60,000	4,65%	4,75%
(0,9; 100; |V|)	60,000	23,22%	23,75%
(0,9; 100; |Cliques|)	60,000	7,53%	7,76%
(0,9; 1020; |V|)	60,000	24,06%	24,69%
(0,9; 1020; |Cliques|)	60,000	6,29%	6,53%

Tabela B.3: Parâmetros SA: Instância eilB101_cut_28

Parâmetros	**TLE = 60s**	**|Cliques| = 2679**	**|Peso| = 2942326**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	46,923	59,65%	60,01%
(0,1; 10; |Cliques|)	60,000	36,13%	36,40%
(0,1; 100; |V|)	32,424	59,65%	60,03%
(0,1; 100; |Cliques|)	60,000	33,48%	33,79%
(0,1; 1020; |V|)	**28,764**	**59,72%**	**60,00%**
(0,1; 1020; |Cliques|)	60,000	31,47%	31,84%
(0,5; 10; |V|)	60,000	57,82%	58,19%
(0,5; 10; |Cliques|)	60,000	19,75%	20,10%
(0,5; 100; |V|)	58,962	59,65%	60,03%
(0,5; 100; |Cliques|)	60,000	21,09%	21,46%
(0,5; 1020; |V|)	60,000	57,11%	57,47%
(0,5; 1020; |Cliques|)	60,000	19,78%	20,15%
(0,9; 10; |V|)	60,000	23,96%	24,37%
(0,9; 10; |Cliques|)	60,000	5,86%	5,97%
(0,9; 100; |V|)	60,000	29,53%	29,87%
(0,9; 100; |Cliques|)	60,000	6,64%	6,83%
(0,9; 1020; |V|)	60,000	28,48%	28,83%
(0,9; 1020; |Cliques|)	60,000	5,97%	6,13%

Tabela B.4: Parâmetros SA: Instância eilB101_cut_29

Parâmetros	**TLE = 60s**	**|Cliques| = 2998**	**|Peso| = 3304625**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	41,987	59,91%	60,02%
(0,1; 10; |Cliques|)	60,000	34,52%	34,64%
(0,1; 100; |V|)	31,185	59,87%	60,00%
(0,1; 100; |Cliques|)	60,000	33,99%	34,13%
(0,1; 1020; |V|)	**25,027**	**59,81%**	**60,00%**
(0,1; 1020; |Cliques|)	60,000	33,76%	33,95%
(0,5; 10; |V|)	55,289	59,94%	60,01%
(0,5; 10; |Cliques|)	60,000	19,98%	20,08%
(0,5; 100; |V|)	50,643	59,77%	60,02%
(0,5; 100; |Cliques|)	60,000	21,75%	21,94%
(0,5; 1020; |V|)	58,531	59,67%	60,02%
(0,5; 1020; |Cliques|)	60,000	23,48%	23,65%
(0,9; 10; |V|)	60,000	27,12%	27,25%
(0,9; 10; |Cliques|)	60,000	4,40%	4,45%
(0,9; 100; |V|)	60,000	29,79%	30,06%
(0,9; 100; |Cliques|)	60,000	5,97%	6,02%
(0,9; 1020; |V|)	60,000	28,49%	28,79%
(0,9; 1020; |Cliques|)	60,000	5,80%	5,91%

Tabela B.5: Parâmetros SA: Instância eilB101_cut_30

Parâmetros	**TLE = 60s**	**|Cliques| = 5326**	**|Peso| = 5789722**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	60,000	45,64%	46,09%
(0,1; 10; |Cliques|)	60,000	13,48%	13,70%
(0,1; 100; |V|)	**60,000**	**49,74%**	**50,20%**
(0,1; 100; |Cliques|)	60,000	12,73%	12,96%
(0,1; 1020; |V|)	60,000	48,65%	49,11%
(0,1; 1020; |Cliques|)	60,000	12,09%	12,29%
(0,5; 10; |V|)	60,000	37,48%	37,89%
(0,5; 10; |Cliques|)	60,000	6,50%	6,61%
(0,5; 100; |V|)	60,000	37,93%	38,36%
(0,5; 100; |Cliques|)	60,000	6,46%	6,56%
(0,5; 1020; |V|)	60,000	36,12%	36,58%
(0,5; 1020; |Cliques|)	60,000	6,97%	7,10%
(0,9; 10; |V|)	60,000	13,67%	13,91%
(0,9; 10; |Cliques|)	60,000	1,16%	1,19%
(0,9; 100; |V|)	60,000	14,38%	14,61%
(0,9; 100; |Cliques|)	60,000	1,92%	1,97%
(0,9; 1020; |V|)	60,000	14,19%	14,48%
(0,9; 1020; |Cliques|)	60,000	2,25%	2,32%

B.2 Parametrização para o conjunto INRC

Para avaliar os parâmetros da heurística *Simulated Annealing* que serão utilizados na execução das 3804 instâncias do conjunto INRC, foram realizados experimentos com

Tabela B.6: Parâmetros SA: Instância eilB101_cut_31

Parâmetros	**TLE = 60s**	**|Cliques| = 3764**	**|Peso| = 4064542**
(α; T_0; *SAmax*)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	60,000	52,42%	53,11%
(0,1; 10; |Cliques|)	60,000	19,95%	20,35%
(0,1; 100; |V|)	**60,000**	**55,55%**	**56,25%**
(0,1; 100; |Cliques|)	60,000	20,48%	20,90%
(0,1; 1020; |V|)	60,000	52,79%	53,51%
(0,1; 1020; |Cliques|)	60,000	18,36%	18,74%
(0,5; 10; |V|)	60,000	41,63%	42,27%
(0,5; 10; |Cliques|)	60,000	10,18%	10,41%
(0,5; 100; |V|)	60,000	40,70%	41,37%
(0,5; 100; |Cliques|)	60,000	10,47%	10,72%
(0,5; 1020; |V|)	60,000	38,87%	39,52%
(0,5; 1020; |Cliques|)	60,000	11,50%	11,73%
(0,9; 10; |V|)	60,000	16,42%	16,81%
(0,9; 10; |Cliques|)	60,000	1,89%	1,92%
(0,9; 100; |V|)	60,000	17,85%	18,25%
(0,9; 100; |Cliques|)	60,000	2,52%	2,58%
(0,9; 1020; |V|)	60,000	18,89%	19,33%
(0,9; 1020; |Cliques|)	60,000	2,74%	2,84%

Tabela B.7: Parâmetros SA: Instância eilB101_cut_32

Parâmetros	**TLE = 60s**	**|Cliques| = 3783**	**|Peso| = 4067291**
(α; T_0; *SAmax*)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	60,000	50,09%	50,75%
(0,1; 10; |Cliques|)	60,000	18,56%	18,92%
(0,1; 100; |V|)	**60,000**	**51,52%**	**52,17%**
(0,1; 100; |Cliques|)	60,000	16,79%	17,13%
(0,1; 1020; |V|)	60,000	50,07%	50,71%
(0,1; 1020; |Cliques|)	60,000	15,97%	16,26%
(0,5; 10; |V|)	60,000	37,22%	37,78%
(0,5; 10; |Cliques|)	60,000	9,60%	9,83%
(0,5; 100; |V|)	60,000	35,84%	36,40%
(0,5; 100; |Cliques|)	60,000	9,75%	9,96%
(0,5; 1020; |V|)	37,066	35,08%	35,69%
(0,5; 1020; |Cliques|)	60,000	7,59%	7,81%
(0,9; 10; |V|)	60,000	14,17%	14,46%
(0,9; 10; |Cliques|)	60,000	1,80%	1,84%
(0,9; 100; |V|)	60,000	15,75%	16,12%
(0,9; 100; |Cliques|)	60,000	2,88%	2,99%
(0,9; 1020; |V|)	60,000	13,67%	14,01%
(0,9; 1020; |Cliques|)	60,000	2,33%	2,41%

um subgrupo de instâncias desse conjunto, caracterizado na Tabela 4.6. Nas tabelas a seguir (Tabelas B.12-B.24), são apresentados, para cada instância do subgrupo, o tempo necessário para encontrar 60% do peso total (solução ótima), a porcentagem de peso

Tabela B.8: Parâmetros SA: Instância eilB101_cut_33

Parâmetros	**TLE = 60s**	**|Cliques| = 4745**	**|Peso| = 5142458**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	60,000	47,65%	48,16%
(0,1; 10; |Cliques|)	60,000	14,50%	14,75%
(0,1; 100; |V|)	**60,000**	**49,04%**	**49,52%**
(0,1; 100; |Cliques|)	60,000	13,42%	13,64%
(0,1; 1020; |V|)	60,000	46,11%	46,63%
(0,1; 1020; |Cliques|)	60,000	11,65%	11,86%
(0,5; 10; |V|)	60,000	37,07%	37,52%
(0,5; 10; |Cliques|)	60,000	7,21%	7,33%
(0,5; 100; |V|)	60,000	36,38%	36,84%
(0,5; 100; |Cliques|)	60,000	7,86%	8,02%
(0,5; 1020; |V|)	60,000	33,57%	34,04%
(0,5; 1020; |Cliques|)	60,000	8,30%	8,45%
(0,9; 10; |V|)	60,000	12,43%	12,64%
(0,9; 10; |Cliques|)	60,000	1,20%	1,22%
(0,9; 100; |V|)	60,000	14,25%	14,49%
(0,9; 100; |Cliques|)	60,000	1,48%	1,52%
(0,9; 1020; |V|)	60,000	11,59%	11,77%
(0,9; 1020; |Cliques|)	60,000	1,90%	1,96%

Tabela B.9: Parâmetros SA: Instância eilB101_cut_34

Parâmetros	**TLE = 60s**	**|Cliques| = 6370**	**|Peso| = 6855410**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	60,000	43,78%	44,32%
(0,1; 10; |Cliques|)	60,000	9,26%	9,46%
(0,1; 100; |V|)	**60,000**	**44,38%**	**44,89%**
(0,1; 100; |Cliques|)	60,000	9,34%	9,51%
(0,1; 1020; |V|)	60,000	41,85%	42,36%
(0,1; 1020; |Cliques|)	60,000	8,01%	8,19%
(0,5; 10; |V|)	60,000	31,93%	32,40%
(0,5; 10; |Cliques|)	60,000	4,52%	4,62%
(0,5; 100; |V|)	60,000	32,46%	32,92%
(0,5; 100; |Cliques|)	60,000	3,97%	4,06%
(0,5; 1020; |V|)	60,000	29,37%	29,81%
(0,5; 1020; |Cliques|)	60,000	4,77%	4,86%
(0,9; 10; |V|)	60,000	10,03%	10,24%
(0,9; 10; |Cliques|)	60,000	0,80%	0,83%
(0,9; 100; |V|)	60,000	12,26%	12,47%
(0,9; 100; |Cliques|)	60,000	0,94%	0,96%
(0,9; 1020; |V|)	60,000	9,64%	9,84%
(0,9; 1020; |Cliques|)	60,000	1,15%	1,18%

e de cliques encontrada. Foi considerado como critério de parada um tempo limite de execução (TLE) de 60 segundos. Na primeira linha das tabelas, a solução ótima de cada instância é apresentada, constituída pela quantidade de cliques (|Cliques|) e do peso

Tabela B.10: Parâmetros SA: Instância eilB101_cut_35

Parâmetros	**TLE = 60s**	**|Cliques| = 4845**	**|Peso| = 5252881**
(α; T_0; *SAmax*)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	**60,000**	**48,19%**	**48,83%**
(0,1; 10; |Cliques|)	60,000	12,07%	12,38%
(0,1; 100; |V|)	60,000	45,99%	46,60%
(0,1; 100; |Cliques|)	60,000	10,66%	10,93%
(0,1; 1020; |V|)	60,000	44,70%	45,32%
(0,1; 1020; |Cliques|)	60,000	9,81%	10,02%
(0,5; 10; |V|)	60,000	33,89%	34,49%
(0,5; 10; |Cliques|)	60,000	5,90%	6,09%
(0,5; 100; |V|)	60,000	33,50%	34,09%
(0,5; 100; |Cliques|)	60,000	5,88%	6,05%
(0,5; 1020; |V|)	60,000	31,20%	31,76%
(0,5; 1020; |Cliques|)	60,000	6,97%	7,23%
(0,9; 10; |V|)	60,000	11,60%	11,92%
(0,9; 10; |Cliques|)	60,000	0,90%	0,92%
(0,9; 100; |V|)	60,000	12,65%	13,01%
(0,9; 100; |Cliques|)	60,000	1,33%	1,36%
(0,9; 1020; |V|)	60,000	10,89%	11,19%
(0,9; 1020; |Cliques|)	60,000	1,15%	1,18%

Tabela B.11: Parâmetros SA: Instância eilB101_cut_36

Parâmetros	**TLE = 60s**	**|Cliques| = 6655**	**|Peso| = 7095837**
(α; T_0; *SAmax*)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	**60,000**	**35,46%**	**35,87%**
(0,1; 10; |Cliques|)	60,000	7,18%	7,31%
(0,1; 100; |V|)	60,000	34,80%	35,22%
(0,1; 100; |Cliques|)	60,000	7,24%	7,39%
(0,1; 1020; |V|)	60,000	33,79%	34,25%
(0,1; 1020; |Cliques|)	60,000	6,42%	6,55%
(0,5; 10; |V|)	60,000	24,27%	24,61%
(0,5; 10; |Cliques|)	60,000	3,25%	3,32%
(0,5; 100; |V|)	60,000	24,21%	24,57%
(0,5; 100; |Cliques|)	60,000	3,16%	3,22%
(0,5; 1020; |V|)	60,000	22,90%	23,27%
(0,5; 1020; |Cliques|)	60,000	4,19%	4,29%
(0,9; 10; |V|)	60,000	7,83%	7,98%
(0,9; 10; |Cliques|)	60,000	0,53%	0,54%
(0,9; 100; |V|)	60,000	9,15%	9,36%
(0,9; 100; |Cliques|)	60,000	0,78%	0,81%
(0,9; 1020; |V|)	60,000	7,84%	8,03%
(0,9; 1020; |Cliques|)	60,000	0,45%	0,47%

total (|Peso|).

Tabela B.12: Parâmetros SA: Instância long_hidden_02_cut_15

Parâmetros	**TLE = 60s**	**|Cliques| = 814**	**|Peso| = 909251**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	53,358	59,46%	60,11%
(0,1; 10; |Cliques|)	**13,746**	**59,34%**	**60,10%**
(0,1; 100; |V|)	60,000	2,46%	2,39%
(0,1; 100; |Cliques|)	19,911	59,34%	60,11%
(0,1; 1020; |V|)	60,000	1,84%	1,92%
(0,1; 1020; |Cliques|)	56,484	59,46%	60,12%
(0,5; 10; |V|)	60,000	0,00%	0,00%
(0,5; 10; |Cliques|)	38,656	59,58%	60,07%
(0,5; 100; |V|)	60,000	0,00%	0,00%
(0,5; 100; |Cliques|)	59,858	59,58%	60,11%
(0,5; 1020; |V|)	60,000	1,72%	1,78%
(0,5; 1020; |Cliques|)	60,000	6,27%	6,28%
(0,9; 10; |V|)	60,000	0,00%	0,00%
(0,9; 10; |Cliques|)	60,000	0,12%	0,13%
(0,9; 100; |V|)	60,000	0,86%	0,87%
(0,9; 100; |Cliques|)	60,000	0,00%	0,00%
(0,9; 1020; |V|)	60,000	0,86%	0,84%
(0,9; 1020; |Cliques|)	60,000	1,23%	1,28%

Tabela B.13: Parâmetros SA: Instância long_hidden_02_cut_16

Parâmetros	**TLE = 60s**	**|Cliques| = 826**	**|Peso| = 916282**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	55,688	59,56%	60,04%
(0,1; 10; |Cliques|)	**13,892**	**59,69%**	**60,01%**
(0,1; 100; |V|)	60,000	4,12%	4,25%
(0,1; 100; |Cliques|)	23,602	59,44%	60,05%
(0,1; 1020; |V|)	60,000	2,30%	2,38%
(0,1; 1020; |Cliques|)	59,270	59,44%	60,04%
(0,5; 10; |V|)	60,000	0,24%	0,23%
(0,5; 10; |Cliques|)	36,396	59,56%	60,07%
(0,5; 100; |V|)	60,000	0,00%	0,00%
(0,5; 100; |Cliques|)	60,000	56,54%	57,20%
(0,5; 1020; |V|)	60,000	0,97%	1,04%
(0,5; 1020; |Cliques|)	60,000	6,66%	6,69%
(0,9; 10; |V|)	60,000	0,00%	0,00%
(0,9; 10; |Cliques|)	60,000	0,00%	0,00%
(0,9; 100; |V|)	60,000	0,00%	0,00%
(0,9; 100; |Cliques|)	60,000	0,00%	0,00%
(0,9; 1020; |V|)	60,000	0,73%	0,72%
(0,9; 1020; |Cliques|)	60,000	0,48%	0,50%

Tabela B.14: Parâmetros SA: Instância long_hidden_02_cut_18

Parâmetros	**TLE = 60s**	**|Cliques| = 824**	**|Peso| = 919061**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	51,205	59,59%	60,09%
(0,1; 10; |Cliques|)	**13,568**	**59,71%**	**60,02%**
(0,1; 100; |V|)	60,000	2,18%	2,22%
(0,1; 100; |Cliques|)	20,351	59,34%	59,34%
(0,1; 1020; |V|)	60,000	1,46%	1,46%
(0,1; 1020; |Cliques|)	60,000	58,62%	59,40%
(0,5; 10; |V|)	60,000	0,00%	0,00%
(0,5; 10; |Cliques|)	35,092	59,34%	60,00%
(0,5; 100; |V|)	60,000	0,36%	0,35%
(0,5; 100; |Cliques|)	52,583	59,71%	60,11%
(0,5; 1020; |V|)	60,000	1,82%	1,88%
(0,5; 1020; |Cliques|)	60,000	3,40%	3,34%
(0,9; 10; |V|)	60,000	0,00%	0,00%
(0,9; 10; |Cliques|)	60,000	0,00%	0,00%
(0,9; 100; |V|)	60,000	0,00%	0,00%
(0,9; 100; |Cliques|)	60,000	0,00%	0,00%
(0,9; 1020; |V|)	60,000	0,97%	0,92%
(0,9; 1020; |Cliques|)	60,000	1,09%	1,09%

Tabela B.15: Parâmetros SA: Instância long_hidden_03_cut_15

Parâmetros	**TLE = 60s**	**|Cliques| = 916**	**|Peso| = 1000479**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	60,000	47,27%	47,64%
(0,1; 10; |Cliques|)	31,466	59,61%	60,05%
(0,1; 100; |V|)	60,000	1,53%	1,52%
(0,1; 100; |Cliques|)	**26,692**	**59,72%**	**60,05%**
(0,1; 1020; |V|)	60,000	6,22%	6,43%
(0,1; 1020; |Cliques|)	60,000	15,17%	15,20%
(0,5; 10; |V|)	60,000	0,00%	0,00%
(0,5; 10; |Cliques|)	46,959	59,72%	60,09%
(0,5; 100; |V|)	60,000	0,11%	0,11%
(0,5; 100; |Cliques|)	60,000	34,93%	35,14%
(0,5; 1020; |V|)	60,000	2,07%	2,13%
(0,5; 1020; |Cliques|)	60,000	8,84%	8,89%
(0,9; 10; |V|)	60,000	0,00%	0,00%
(0,9; 10; |Cliques|)	60,000	0,44%	0,46%
(0,9; 100; |V|)	60,000	0,00%	0,00%
(0,9; 100; |Cliques|)	60,000	0,00%	0,00%
(0,9; 1020; |V|)	60,000	1,53%	1,58%
(0,9; 1020; |Cliques|)	60,000	1,42%	1,53%

B.3 Parametrização para o conjunto Telebus

Para avaliar os parâmetros da heurística *Simulated Annealing* que serão utilizados na execução das 3085 instâncias do conjunto Telebus, foram realizados experimentos com

Tabela B.16: Parâmetros SA: Instância long_hidden_03_cut_16

Parâmetros	**TLE = 60s**	**|Cliques| = 881**	**|Peso| = 965183**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	60,000	44,15%	44,26%
(0,1; 10; |Cliques|)	**18,326**	**59,82%**	**60,04%**
(0,1; 100; |V|)	60,000	3,52%	3,60%
(0,1; 100; |Cliques|)	27,334	59,82%	60,02%
(0,1; 1020; |V|)	60,000	8,51%	8,46%
(0,1; 1020; |Cliques|)	60,000	59,25%	59,58%
(0,5; 10; |V|)	60,000	0,23%	0,23%
(0,5; 10; |Cliques|)	44,466	59,70%	60,02%
(0,5; 100; |V|)	60,000	0,57%	0,56%
(0,5; 100; |Cliques|)	60,000	34,05%	34,22%
(0,5; 1020; |V|)	60,000	2,16%	2,21%
(0,5; 1020; |Cliques|)	60,000	8,40%	8,44%
(0,9; 10; |V|)	60,000	0,00%	0,00%
(0,9; 10; |Cliques|)	60,000	0,11%	0,11%
(0,9; 100; |V|)	60,000	0,00%	0,00%
(0,9; 100; |Cliques|)	60,000	0,23%	0,22%
(0,9; 1020; |V|)	60,000	0,23%	0,22%
(0,9; 1020; |Cliques|)	60,000	1,02%	1,05%

Tabela B.17: Parâmetros SA: Instância long_hidden_03_cut_17

Parâmetros	**TLE = 60s**	**|Cliques| = 909**	**|Peso| = 990359**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	60,000	36,96%	37,36%
(0,1; 10; |Cliques|)	**16,842**	**59,74%**	**59,74%**
(0,1; 100; |V|)	60,000	1,43%	1,43%
(0,1; 100; |Cliques|)	28,837	59,74%	60,02%
(0,1; 1020; |V|)	60,000	6,93%	6,93%
(0,1; 1020; |Cliques|)	51,572	59,85%	60,10%
(0,5; 10; |V|)	60,000	0,00%	0,00%
(0,5; 10; |Cliques|)	45,610	59,96%	60,09%
(0,5; 100; |V|)	60,000	0,33%	0,35%
(0,5; 100; |Cliques|)	60,000	44,99%	45,31%
(0,5; 1020; |V|)	60,000	1,98%	2,08%
(0,5; 1020; |Cliques|)	60,000	8,91%	8,98%
(0,9; 10; |V|)	60,000	0,00%	0,00%
(0,9; 10; |Cliques|)	60,000	0,00%	0,00%
(0,9; 100; |V|)	60,000	0,00%	0,00%
(0,9; 100; |Cliques|)	60,000	0,11%	0,11%
(0,9; 1020; |V|)	60,000	1,21%	1,27%
(0,9; 1020; |Cliques|)	60,000	1,10%	1,20%

um subgrupo de instâncias desse conjunto, caracterizado na Tabela 4.9. Nas tabelas a seguir (Tabelas B.25-B.29), são apresentados, para cada instância do subgrupo, o tempo necessário para encontrar 60% do peso total (solução ótima), a porcentagem de peso

Tabela B.18: Parâmetros SA: Instância long_hidden_03_cut_18

Parâmetros	**TLE = 60s**	**|Cliques| = 850**	**|Peso| = 929264**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	60,000	38,82%	39,22%
(0,1; 10; |Cliques|)	**15,509**	**59,65%**	**60,10%**
(0,1; 100; |V|)	60,000	3,18%	3,08%
(0,1; 100; |Cliques|)	23,683	59,76%	60,05%
(0,1; 1020; |V|)	60,000	4,82%	4,78%
(0,1; 1020; |Cliques|)	46,437	59,76%	60,09%
(0,5; 10; |V|)	60,000	0,35%	0,38%
(0,5; 10; |Cliques|)	42,306	59,76%	60,01%
(0,5; 100; |V|)	60,000	0,47%	0,46%
(0,5; 100; |Cliques|)	60,000	49,41%	49,58%
(0,5; 1020; |V|)	60,000	2,24%	2,26%
(0,5; 1020; |Cliques|)	60,000	18,82%	18,81%
(0,9; 10; |V|)	60,000	0,24%	0,26%
(0,9; 10; |Cliques|)	60,000	0,00%	0,00%
(0,9; 100; |V|)	60,000	0,00%	0,00%
(0,9; 100; |Cliques|)	60,000	0,12%	0,11%
(0,9; 1020; |V|)	60,000	0,94%	0,95%
(0,9; 1020; |Cliques|)	60,000	0,82%	0,85%

Tabela B.19: Parâmetros SA: Instância long_hidden_05_cut_16

Parâmetros	**TLE = 60s**	**|Cliques| = 856**	**|Peso| = 945686**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	52,971	59,58%	60,02%
(0,1; 10; |Cliques|)	**13,155**	**59,81%**	**60,13%**
(0,1; 100; |V|)	60,000	35,51%	35,80%
(0,1; 100; |Cliques|)	21,173	59,58%	60,09%
(0,1; 1020; |V|)	60,000	2,57%	2,61%
(0,1; 1020; |Cliques|)	44,864	59,81%	60,11%
(0,5; 10; |V|)	60,000	0,00%	0,00%
(0,5; 10; |Cliques|)	38,759	59,58%	60,01%
(0,5; 100; |V|)	60,000	0,35%	0,36%
(0,5; 100; |Cliques|)	55,959	59,58%	60,05%
(0,5; 1020; |V|)	60,000	3,27%	3,29%
(0,5; 1020; |Cliques|)	60,000	17,06%	17,07%
(0,9; 10; |V|)	60,000	0,00%	0,00%
(0,9; 10; |Cliques|)	60,000	0,00%	0,00%
(0,9; 100; |V|)	60,000	0,00%	0,00%
(0,9; 100; |Cliques|)	60,000	0,00%	0,00%
(0,9; 1020; |V|)	60,000	0,70%	0,66%
(0,9; 1020; |Cliques|)	60,000	2,92%	2,91%

e de cliques encontrada. Foi considerado como critério de parada um tempo limite de execução (TLE) de 60 segundos. Na primeira linha das tabelas, a solução ótima de cada instância é apresentada, constituída pela quantidade de cliques (|Cliques|) e do peso

Tabela B.20: Parâmetros SA: Instância long_hidden_05_cut_17

Parâmetros	**TLE = 60s**	**|Cliques| = 807**	**|Peso| = 898279**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	54,798	59,48%	60,10%
(0,1; 10; |Cliques|)	**14,476**	**59,48%**	**60,08%**
(0,1; 100; |V|)	60,000	33,46%	33,99%
(0,1; 100; |Cliques|)	22,522	59,48%	60,10%
(0,1; 1020; |V|)	60,000	5,95%	6,10%
(0,1; 1020; |Cliques|)	34,536	59,60%	60,02%
(0,5; 10; |V|)	60,000	0,12%	0,12%
(0,5; 10; |Cliques|)	36,266	59,60%	60,09%
(0,5; 100; |V|)	60,000	0,37%	0,36%
(0,5; 100; |Cliques|)	54,283	59,48%	60,01%
(0,5; 1020; |V|)	60,000	4,21%	4,29%
(0,5; 1020; |Cliques|)	60,000	18,22%	18,25%
(0,9; 10; |V|)	60,000	0,00%	0,00%
(0,9; 10; |Cliques|)	60,000	0,12%	0,12%
(0,9; 100; |V|)	60,000	0,00%	0,00%
(0,9; 100; |Cliques|)	60,000	0,00%	0,00%
(0,9; 1020; |V|)	60,000	1,36%	1,35%
(0,9; 1020; |Cliques|)	60,000	4,34%	4,43%

Tabela B.21: Parâmetros SA: Instância long_hidden_05_cut_18

Parâmetros	**TLE = 60s**	**|Cliques| = 878**	**|Peso| = 971089**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	57,336	59,45%	60,09%
(0,1; 10; |Cliques|)	**14,941**	**59,68%**	**60,02%**
(0,1; 100; |V|)	60,000	19,59%	20,25%
(0,1; 100; |Cliques|)	24,760	59,79%	60,11%
(0,1; 1020; |V|)	60,000	2,62%	2,63%
(0,1; 1020; |Cliques|)	51,917	59,45%	60,04%
(0,5; 10; |V|)	60,000	0,00%	0,00%
(0,5; 10; |Cliques|)	36,192	59,57%	60,03%
(0,5; 100; |V|)	60,000	0,00%	0,00%
(0,5; 100; |Cliques|)	60,000	48,63%	49,49%
(0,5; 1020; |V|)	60,000	2,62%	2,74%
(0,5; 1020; |Cliques|)	60,000	12,87%	12,87%
(0,9; 10; |V|)	60,000	0,00%	0,00%
(0,9; 10; |Cliques|)	60,000	0,00%	0,00%
(0,9; 100; |V|)	60,000	0,00%	0,00%
(0,9; 100; |Cliques|)	60,000	0,00%	0,00%
(0,9; 1020; |V|)	60,000	0,34%	0,35%
(0,9; 1020; |Cliques|)	60,000	1,71%	1,77%

total (|Peso|).

Tabela B.22: Parâmetros SA: Instância long_late_03_cut_13

Parâmetros	**TLE = 60s**	**|Cliques| = 867**	**|Peso| = 949106**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	60,000	50,29%	50,63%
(0,1; 10; |Cliques|)	**19,212**	**59,98%**	**60,08%**
(0,1; 100; |V|)	60,000	2,08%	2,06%
(0,1; 100; |Cliques|)	27,126	59,75%	60,10%
(0,1; 1020; |V|)	60,000	4,61%	4,58%
(0,1; 1020; |Cliques|)	51,305	59,75%	60,12%
(0,5; 10; |V|)	60,000	0,00%	0,00%
(0,5; 10; |Cliques|)	46,096	59,98%	60,10%
(0,5; 100; |V|)	60,000	0,00%	0,00%
(0,5; 100; |Cliques|)	60,000	48,63%	49,49%
(0,5; 1020; |V|)	60,000	2,19%	2,15%
(0,5; 1020; |Cliques|)	60,000	11,76%	11,85%
(0,9; 10; |V|)	60,000	0,00%	0,00%
(0,9; 10; |Cliques|)	60,000	0,00%	0,00%
(0,9; 100; |V|)	60,000	0,00%	0,00%
(0,9; 100; |Cliques|)	60,000	0,23%	0,23%
(0,9; 1020; |V|)	60,000	0,35%	0,34%
(0,9; 1020; |Cliques|)	60,000	2,88%	2,86%

Tabela B.23: Parâmetros SA: Instância long_late_03_cut_17

Parâmetros	**TLE = 60s**	**|Cliques| = 859**	**|Peso| = 944149**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	60,000	54,25%	54,33%
(0,1; 10; |Cliques|)	**15,174**	**59,84%**	**60,12%**
(0,1; 100; |V|)	60,000	1,16%	1,17%
(0,1; 100; |Cliques|)	25,966	59,95%	60,04%
(0,1; 1020; |V|)	60,000	5,70%	5,77%
(0,1; 1020; |Cliques|)	60,000	59,25%	59,57%
(0,5; 10; |V|)	60,000	0,00%	0,00%
(0,5; 10; |Cliques|)	41,012	59,84%	60,04%
(0,5; 100; |V|)	60,000	0,00%	0,00%
(0,5; 100; |Cliques|)	60,000	46,94%	47,30%
(0,5; 1020; |V|)	60,000	5,01%	5,08%
(0,5; 1020; |Cliques|)	60,000	11,87%	11,91%
(0,9; 10; |V|)	60,000	0,00%	0,00%
(0,9; 10; |Cliques|)	60,000	0,00%	0,00%
(0,9; 100; |V|)	60,000	0,00%	0,00%
(0,9; 100; |Cliques|)	60,000	0,00%	0,00%
(0,9; 1020; |V|)	60,000	1,28%	1,26%
(0,9; 1020; |Cliques|)	60,000	1,86%	1,84%

Tabela B.24: Parâmetros SA: Instância long_late_03_cut_18

Parâmetros	**TLE = 60s**	**|Cliques| = 845**	**|Peso| = 926472**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	60,000	53,37%	53,64%
(0,1; 10; |Cliques|)	**16,109**	**59,76%**	**60,04%**
(0,1; 100; |V|)	60,000	1,66%	1,69%
(0,1; 100; |Cliques|)	24,511	59,64%	60,03%
(0,1; 1020; |V|)	60,000	2,84%	2,91%
(0,1; 1020; |Cliques|)	59,035	59,76%	60,08%
(0,5; 10; |V|)	60,000	0,00%	0,00%
(0,5; 10; |Cliques|)	41,562	59,76%	60,12%
(0,5; 100; |V|)	60,000	0,00%	0,00%
(0,5; 100; |Cliques|)	60,000	46,27%	46,66%
(0,5; 1020; |V|)	60,000	4,14%	4,23%
(0,5; 1020; |Cliques|)	60,000	8,05%	8,07%
(0,9; 10; |V|)	60,000	0,12%	0,12%
(0,9; 10; |Cliques|)	60,000	0,00%	0,00%
(0,9; 100; |V|)	60,000	0,00%	0,00%
(0,9; 100; |Cliques|)	60,000	0,00%	0,00%
(0,9; 1020; |V|)	60,000	0,36%	0,34%
(0,9; 1020; |Cliques|)	60,000	2,60%	2,68%

Tabela B.25: Parâmetros SA: Instância t0415_cut_1

Parâmetros	**TLE = 60s**	**|Cliques| = 72**	**|Peso| = 79664**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	9,801	59,72%	60,38%
(0,1; 10; |Cliques|)	2,388	59,72%	60,06%
(0,1; 100; |V|)	14,680	61,11%	61,29%
(0,1; 100; |Cliques|)	2,327	61,11%	61,22%
(0,1; 1020; |V|)	12,900	59,72%	60,65%
(0,1; 1020; |Cliques|)	**0,703**	**59,72%**	**60,24%**
(0,5; 10; |V|)	30,940	59,72%	60,24%
(0,5; 10; |Cliques|)	4,059	59,72%	60,51%
(0,5; 100; |V|)	26,970	61,11%	60,98%
(0,5; 100; |Cliques|)	2,650	59,72%	60,18%
(0,5; 1020; |V|)	25,220	59,72%	60,28%
(0,5; 1020; |Cliques|)	3,588	59,72%	60,74%
(0,9; 10; |V|)	60,000	13,89%	14,30%
(0,9; 10; |Cliques|)	21,670	59,72%	60,28%
(0,9; 100; |V|)	60,000	26,39%	26,38%
(0,9; 100; |Cliques|)	16,630	59,72%	60,00%
(0,9; 1020; |V|)	60,000	12,50%	12,79%
(0,9; 1020; |Cliques|)	14,100	59,72%	60,85%

B.4 Parametrização para o conjunto Uchoa

Para avaliar os parâmetros da heurística *Simulated Annealing* que serão utilizados na execução das quatro instâncias do conjunto Uchoa, considere as características de cada

Tabela B.26: Parâmetros SA: Instância t0418_cut_29

Parâmetros	**TLE = 60s**	**|Cliques| = 11**	**|Peso| = 11466**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	5,369	63,64%	64,30%
(0,1; 10; |Cliques|)	0,986	63,64%	64,24%
(0,1; 100; |V|)	8,725	63,64%	63,17%
(0,1; 100; |Cliques|)	0,816	63,64%	64,34%
(0,1; 1020; |V|)	16,770	63,64%	63,95%
(0,1; 1020; |Cliques|)	0,300	63,64%	63,91%
(0,5; 10; |V|)	28,570	63,64%	64,29%
(0,5; 10; |Cliques|)	0,963	63,64%	64,29%
(0,5; 100; |V|)	25,500	63,64%	63,47%
(0,5; 100; |Cliques|)	**0,246**	**63,64%**	**63,82%**
(0,5; 1020; |V|)	29,200	63,64%	63,55%
(0,5; 1020; |Cliques|)	0,360	63,64%	64,28%
(0,9; 10; |V|)	60,000	27,27%	26,77%
(0,9; 10; |Cliques|)	1,219	63,64%	63,17%
(0,9; 100; |V|)	60,000	9,09%	9,58%
(0,9; 100; |Cliques|)	1,577	63,64%	64,22%
(0,9; 1020; |V|)	60,000	9,09%	8,90%
(0,9; 1020; |Cliques|)	1,426	63,64%	64,26%

Tabela B.27: Parâmetros SA: Instância t0418_cut_36

Parâmetros	**TLE = 60s**	**|Cliques| = 5**	**|Peso| = 5253**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	5,548	60,00%	60,65%
(0,1; 10; |Cliques|)	**0,041**	**60,00%**	**60,65%**
(0,1; 100; |V|)	8,188	60,00%	60,10%
(0,1; 100; |Cliques|)	0,058	80,00%	80,07%
(0,1; 1020; |V|)	6,427	60,00%	60,65%
(0,1; 1020; |Cliques|)	0,149	60,00%	60,00%
(0,5; 10; |V|)	18,430	80,00%	80,49%
(0,5; 10; |Cliques|)	0,063	60,00%	60,56%
(0,5; 100; |V|)	27,970	80,00%	79,52%
(0,5; 100; |Cliques|)	0,095	80,00%	79,52%
(0,5; 1020; |V|)	23,800	60,00%	60,00%
(0,5; 1020; |Cliques|)	0,147	60,00%	61,07%
(0,9; 10; |V|)	60,000	0,00%	0,00%
(0,9; 10; |Cliques|)	0,248	60,00%	60,10%
(0,9; 100; |V|)	60,000	20,00%	20,65%
(0,9; 100; |Cliques|)	0,544	80,00%	80,58%
(0,9; 1020; |V|)	60,000	40,00%	40,00%
(0,9; 1020; |Cliques|)	1,079	80,00%	79,35%

instância, apresentadas na Tabela 4.11. Nas tabelas a seguir (Tabelas B.30-B.33), são apresentados, para cada instância do conjunto, o tempo necessário para encontrar 60% do peso total (solução ótima), a porcentagem de peso e de cliques encontrada. Foi consi-

Tabela B.28: Parâmetros SA: Instância t0418_cut_37

Parâmetros	**TLE = 60s**	**|Cliques| = 4**	**|Peso| = 4223**
(α; T_0; *SAmax*)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	6,401	75,00%	75,37%
(0,1; 10; |Cliques|)	**0,030**	**75,00%**	**75,37%**
(0,1; 100; |V|)	4,213	75,00%	75,37%
(0,1; 100; |Cliques|)	0,050	75,00%	75,37%
(0,1; 1020; |V|)	15,170	75,00%	74,50%
(0,1; 1020; |Cliques|)	0,127	75,00%	74,50%
(0,5; 10; |V|)	26,030	75,00%	74,40%
(0,5; 10; |Cliques|)	0,874	75,00%	75,37%
(0,5; 100; |V|)	24,760	75,00%	75,37%
(0,5; 100; |Cliques|)	0,081	75,00%	75,37%
(0,5; 1020; |V|)	16,740	75,00%	74,50%
(0,5; 1020; |Cliques|)	0,122	75,00%	75,73%
(0,9; 10; |V|)	60,000	25,00%	25,60%
(0,9; 10; |Cliques|)	0,389	75,00%	74,50%
(0,9; 100; |V|)	60,000	25,00%	25,50%
(0,9; 100; |Cliques|)	0,596	75,00%	74,50%
(0,9; 1020; |V|)	60,000	25,00%	24,27%
(0,9; 1020; |Cliques|)	1,080	75,00%	75,37%

Tabela B.29: Parâmetros SA: Instância t0418_cut_66

Parâmetros	**TLE = 60s**	**|Cliques| = 2**	**|Peso| = 2090**
(α; T_0; *SAmax*)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	4,353	100,00%	100,00%
(0,1; 10; |Cliques|)	1,891	100,00%	100,00%
(0,1; 100; |V|)	9,670	100,00%	100,00%
(0,1; 100; |Cliques|)	**0,041**	**100,00%**	**100,00%**
(0,1; 1020; |V|)	18,070	100,00%	100,00%
(0,1; 1020; |Cliques|)	0,076	100,00%	100,00%
(0,5; 10; |V|)	17,450	100,00%	100,00%
(0,5; 10; |Cliques|)	0,048	100,00%	100,00%
(0,5; 100; |V|)	26,780	100,00%	100,00%
(0,5; 100; |Cliques|)	0,076	100,00%	100,00%
(0,5; 1020; |V|)	42,720	100,00%	100,00%
(0,5; 1020; |Cliques|)	0,073	100,00%	100,00%
(0,9; 10; |V|)	60,000	0,00%	0,00%
(0,9; 10; |Cliques|)	0,198	100,00%	100,00%
(0,9; 100; |V|)	60,000	50,00%	49,71%
(0,9; 100; |Cliques|)	0,340	100,00%	100,00%
(0,9; 1020; |V|)	60,000	0,00%	0,00%
(0,9; 1020; |Cliques|)	0,633	100,00%	100,00%

derado como critério de parada um tempo limite de execução (TLE) de 60 segundos. Na primeira linha das tabelas, a solução ótima de cada instância é apresentada, constituída pela quantidade de cliques (|Cliques|) e do peso total (|Peso|).

Tabela B.30: Parâmetros SA: Instância pdistuchoa_cut_1

Parâmetros	**TLE = 60s**	**|Cliques| = 1131719**	**|Peso| = 7663794500**
(α; T_0; *SAmax*)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	60,000	2,38%	2,27%
(0,1; 10; |Cliques|)	60,000	2,40%	2,28%
(0,1; 100; |V|)	60,000	2,37%	2,26%
(0,1; 100; |Cliques|)	60,000	2,36%	2,25%
(0,1; 1020; |V|)	60,000	2,73%	2,61%
(0,1; 1020; |Cliques|)	60,000	2,69%	2,57%
(0,5; 10; |V|)	60,000	2,45%	2,34%
(0,5; 10; |Cliques|)	60,000	2,44%	2,34%
(0,5; 100; |V|)	60,000	2,35%	2,24%
(0,5; 100; |Cliques|)	60,000	2,68%	2,63%
(0,5; 1020; |V|)	60,000	2,86%	2,77%
(0,5; 1020; |Cliques|)	**60,000**	**2,93%**	**2,84%**
(0,9; 10; |V|)	60,000	1,47%	1,42%
(0,9; 10; |Cliques|)	60,000	1,38%	1,31%
(0,9; 100; |V|)	60,000	1,36%	1,33%
(0,9; 100; |Cliques|)	60,000	1,41%	1,39%
(0,9; 1020; |V|)	60,000	1,98%	2,10%
(0,9; 1020; |Cliques|)	60,000	1,81%	1,89%

Tabela B.31: Parâmetros SA: Instância pdistuchoa_cut_2

Parâmetros	**TLE = 60s**	**|Cliques| = 3836**	**|Peso| = 5243298**
(α; T_0; *SAmax*)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	15,620	58,31%	60,00%
(0,1; 10; |Cliques|)	33,080	58,08%	60,01%
(0,1; 100; |V|)	12,830	58,02%	60,00%
(0,1; 100; |Cliques|)	28,440	58,21%	60,00%
(0,1; 1020; |V|)	8,560	58,02%	60,01%
(0,1; 1020; |Cliques|)	17,510	57,87%	60,00%
(0,5; 10; |V|)	15,710	58,10%	60,02%
(0,5; 10; |Cliques|)	60,000	48,61%	50,53%
(0,5; 100; |V|)	12,880	58,13%	60,00%
(0,5; 100; |Cliques|)	60,000	49,58%	51,64%
(0,5; 1020; |V|)	**7,790**	**57,82%**	**60,02%**
(0,5; 1020; |Cliques|)	36,280	57,90%	60,01%
(0,9; 10; |V|)	45,930	58,19%	60,01%
(0,9; 10; |Cliques|)	60,000	12,49%	13,38%
(0,9; 100; |V|)	41,810	58,00%	60,00%
(0,9; 100; |Cliques|)	60,000	13,95%	14,75%
(0,9; 1020; |V|)	22,120	57,61%	60,00%
(0,9; 1020; |Cliques|)	60,000	34,46%	36,61%

Tabela B.32: Parâmetros SA: Instância pdistuchoa_cut_3

Parâmetros	**TLE = 60s**	**|Cliques| = 5605**	**|Peso| = 6605680**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	44,220	59,30%	60,01%
(0,1; 10; |Cliques|)	60,000	17,05%	6,42%
(0,1; 100; |V|)	33,230	59,32%	60,01%
(0,1; 100; |Cliques|)	60,000	18,55%	19,05%
(0,1; 1020; |V|)	**23,770**	**59,25%**	**60,00%**
(0,1; 1020; |Cliques|)	60,000	23,92%	24,47%
(0,5; 10; |V|)	60,000	52,52%	53,12%
(0,5; 10; |Cliques|)	60,000	6,86%	6,98%
(0,5; 100; |V|)	60,000	54,02%	54,68%
(0,5; 100; |Cliques|)	60,000	7,44%	7,61%
(0,5; 1020; |V|)	40,560	59,13%	60,01%
(0,5; 1020; |Cliques|)	60,000	10,95%	11,33%
(0,9; 10; |V|)	60,000	14,77%	15,07%
(0,9; 10; |Cliques|)	60,000	1,27%	1,30%
(0,9; 100; |V|)	60,000	16,38%	16,90%
(0,9; 100; |Cliques|)	60,000	1,52%	1,61%
(0,9; 1020; |V|)	60,000	25,51%	26,34%
(0,9; 1020; |Cliques|)	60,000	5,25%	5,47%

Tabela B.33: Parâmetros SA: Instância pdistuchoa_cut_4

Parâmetros	**TLE = 60s**	**|Cliques| = 6266**	**|Peso| = 6957631**
(α; T_0; $SAmax$)	**Tempo (s)**	**% Cliques**	**% Peso**
(0,1; 10; |V|)	60,000	44,89%	45,38%
(0,1; 10; |Cliques|)	60,000	5,96%	6,42%
(0,1; 100; |V|)	60,000	44,20%	44,64%
(0,1; 100; |Cliques|)	60,000	5,47%	5,56%
(0,1; 1020; |V|)	**60,000**	**46,90%**	**47,40%**
(0,1; 1020; |Cliques|)	60,000	5,69%	5,82%
(0,5; 10; |V|)	60,000	23,57%	23,94%
(0,5; 10; |Cliques|)	60,000	2,25%	2,26%
(0,5; 100; |V|)	60,000	22,26%	22,61%
(0,5; 100; |Cliques|)	60,000	2,28%	2,32%
(0,5; 1020; |V|)	60,000	30,82%	31,32%
(0,5; 1020; |Cliques|)	60,000	2,59%	2,66%
(0,9; 10; |V|)	60,000	4,36%	4,43%
(0,9; 10; |Cliques|)	60,000	0,46%	0,47%
(0,9; 100; |V|)	60,000	5,23%	5,32%
(0,9; 100; |Cliques|)	60,000	0,53%	0,52%
(0,9; 1020; |V|)	60,000	8,57%	8,77%
(0,9; 1020; |Cliques|)	60,000	1,13%	1,15%

B.5 Análise dos resultados obtidos quanto aos parâmetros utilizados na heurística Simulated Annealing

Nesta seção, será realizada uma análise dos resultados obtidos para os quatro conjuntos de instâncias, dado a parametrização apresentada nas seções anteriores. Vale lembrar que foram testados alguns valores para os parâmetros pertinentes à heurística *Simulated Annealing* e todos os testes realizados foram apresentados nas tabelas das seções anteriores. A Tabela B.34 apresenta o tempo total gasto para cada um dos conjuntos, considerando a combinação dos valores utilizados para α, T_0 e $SAmax$.

Uma conclusão evidente apresentada pela tabela é o fato de que para valores altos de α, a heurística não apresenta um bom desempenho na resolução do problema. Considerando $\alpha = 0{,}9$, a execução das instâncias dos conjuntos MIPLIB e INRC excederam o tempo limite de execução e, para o conjunto Uchoa, o desempenho também foi ruim.

Já para o conjunto Telebus, pôde-se perceber que quando o número máximo de iterações ($SAmax$) em uma dada temperatura é definido como sendo igual a quantidade de cliques (|Cliques|), o desempenho é consideravelmente melhor que quando definido como igual a quantidade de vértices (|V|). A quantidade de cliques (solução ótima) desse conjunto é bem inferior à quantidade de cliques dos outros conjuntos. Nesse sentido, pode-se concluir que valores baixos para $SAmax$ apresentam melhores resultados que valores altos.

Como conclusão final, a tabela mostra que os melhores resultados obtidos foram utilizando $\alpha = 0{,}1$ e adotando a temperatura inicial $T_0 = 10$ ou $T_0 = 1020$. Além disso, o desempenho da heurística é sempre melhor quando define-se o número de iterações $SAmax$ como sendo o menor valor entre a quantidade de cliques (|Cliques|) e a quantidade de vértices (|V|). A partir dessa tabela, foram setados os valores dos parâmetros para a resolução do problema da enumeração de cliques com peso acima de um limiar e para o problema do clique de peso máximo. Para cada conjunto - MIPLIB, INRC, Telebus e Uchoa - foram utilizados os valores dos parâmetros que apresentaram os melhores resultados, segundo a tabela supracitada.

Tabela B.34: Parâmetros SA: Sumário dos resultados obtidos

Parâmetros	**MIPLIB**	**INRC**	**Telebus**	**Uchoa**
(α; T_0; $SAmax$)	**Tempo (s)**	**Tempo (s)**	**Tempo (s)**	**Tempo (s)**
(0,1; 10; |V|)	602,757	745,356	31,472	179,839
(0,1; 10; |Cliques|)	660,000	**216,416**	5,336	213,080
(0,1; 100; |V|)	556,245	780,000	45,479	166,055
(0,1; 100; |Cliques|)	660,000	316,468	3,292	208,440
(0,1; 1020; |V|)	**537,555**	780,000	69,330	**152,331**
(0,1; 1020; |Cliques|)	660,000	695,420	**1,355**	197,510
(0,5; 10; |V|)	640,799	780,000	121,416	195,710
(0,5; 10; |Cliques|)	660,000	529,372	6,007	240,000
(0,5; 100; |V|)	625,218	780,000	131,992	168,352
(0,5; 100; |Cliques|)	660,000	762,683	3,148	216,280
(0,5; 1020; |V|)	635,597	780,000	137,670	225,930
(0,5; 1020; |Cliques|)	660,000	780,000	4,290	240,000
(0,9; 10; |V|)	660,000	780,000	300,000	221,810
(0,9; 10; |Cliques|)	660,000	780,000	23,721	240,000
(0,9; 100; |V|)	660,000	780,000	300,000	202,120
(0,9; 100; |Cliques|)	660,000	780,000	19,684	240,000
(0,9; 1020; |V|)	660,000	780,000	300,000	202,120
(0,9; 1020; |Cliques|)	660,000	780,000	18,317	240,000