MÁRIO HENRIQUE DE PAIVA PERCHÉ

Abordagens Exata e Heurística Para o Problema de Edição Não Automática de Clusters

NITERÓI-RJ

2012

MÁRIO HENRIQUE DE PAIVA PERCHÉ

Abordagens Exata e Heurística Para o Problema de Edição Não Automática de Clusters

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

Orientador:
Loana Tito Nogueira

Co-orientador:
Fábio Protti

Universidade Federal Fluminense

NITERÓI-RJ

2012

Abordagens Exata e Heurística Para o Problema de Edição Não Automática de *Clusters*

Mário Henrique de Paiva Perché

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

Aprovada por:

Profa. Loana Tito Nogueira, D.Sc. / IC-UFF
(Presidente) / (Orientadora)

Prof. Fábio Protti, D.Sc. / IC-UFF
(Co-orientador)

Prof. Marcone Jamilson Freitas Souza, D.Sc. / DECOM-UFOP

Prof. Luiz Satoru Ochi / IC-UFF

Niterói, 03 de Setembro de 2012.

À minha família, aos meus amigos.

Agradecimentos

Aos meus pais, Roberto e Wanda e aos meus irmãos Felipe e Priscila: sem vocês nada disso seria possível!

À minha namorada Mirelle, por todo o apoio e carinho durante o desenvolvimento deste trabalho.

Ao Igor, Pablo, Sabir e Marcos, pela amizade e momentos únicos vividos dentro e fora da OptHouse.

À Loana e ao Fábio por toda confiança depositada em mim e também pelos ensinamentos de Pesquisa e da Vida.

Ao Satoru pela receptividade e amizade.

Ao Marcone, pela amizade e exemplo de dedicação.

Ao CNPq, pela bolsa concedida.

Aos companheiros do IC-UFF e do projeto OptFrame.

Resumo

Este trabalho trata do Problema de Edição Não Automática de *Clusters* (PENAC). Neste problema, dados um grafo G e um inteiro k (número de *clusters*), objetiva-se realizar o menor número de edições (adições ou remoções de arestas) em G com intuito de torná-lo um grafo clusterizado, isto é, uma união disjunta de exatamente k subgrafos completos. O PENAC pertence à classe NP-Difícil, uma vez que ele pode ser reduzido do Problema de Edição de *Clusters*, quando o número de *clusters* não é fixado. Para resolver o PENAC, faz-se o uso de uma abordagem heurística, inspirada na combinação das metaheurísticas *Iterated Local Search* e *Variable Neighborhood Descent*. Além de tal abordagem heurística, um modelo matemático é proposto para resolução de forma exata do PENAC. O modelo matemático e o algoritmo heurístico foram testados com um conjunto de instâncias geradas de maneira aleatória. O algoritmo heurístico obteve um melhor desempenho que o modelo matemático proposto à medida em se aumentou o nível de complexidade das instâncias utilizadas.

Palavras-chave: Problema de Edição Não Automática de *Clusters*, Modelo Matemático, Busca Local Iterada, Descida em Vizinhança Variável

Abstract

This work addresses the Non-Automatic Cluster Edition Problem. In this problem, given a graph G and an integer k (the number of *clusters*), the objective is to make the least number of editions (additions or deletions of edges) in G in order to make it a clustered graph, i.e., a disjoint union of exactly k complete subgraphs. This problem belongs to the class NP-Hard, once it can be reduced from the Cluster Edition Problem, when the number of clusters is not fixed. To solve it, we use an heuristic algorithm, inspired by the combination of metaheuristics Iterated Local Search and Variable Neighborhood Descent. Besides this heuristic approach, a mathematical model is proposed for solving the exact form PENAC. The mathematical model and the heuristic algorithm was tested with a set of randomly generated instances. The heuristic algorithm achieved a better performance than the mathematical model to measure whether increased the level of complexity of the instances used.

Keywords: Non-Automatic Cluster Edition Problem, Mathematical Model, Iterated Local Search, Variable Neighborhood Descent

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

1 Introdução

Este trabalho descreve o estudo do Problema de Edição Não Automática de *Clusters* (PENAC). Neste problema, dados um grafo G e um inteiro k (número de *clusters*), objetiva-se realizar o menor número de edições (adições ou remoções de arestas) em G com intuito de torná-lo um grafo clusterizado, isto é, uma união disjunta de exatamente k subgrafos completos. O PENAC é uma variação do clássico Problema de Edição de *Clusters* (PEC). A idéia, em ambos os problemas, é realizar o particionamento de uma coleção de dados em *clusters*, de modo que os elementos pertencentes a cada *cluster* possuam maior semelhança entre si do que quando comparados com elementos de outros *clusters*. O Problema de Edição de *Clusters* foi inicialmente estudado por [1]. Desde então, têm surgido vários outros trabalhos com foco em tal problema. Enquanto alguns desses trabalhos aplicam técnicas heurísticas [2, 3, 4, 5, 6] na resolução do Problema de Edição de *Clusters*, outros usam de métodos exatos [2, 5, 7] ou, ainda, algoritmos parametrizados [8]. O grande número de trabalhos relacionados ao Problema de Edição de *Clusters* se deve à aplicabilidade de problemas desse tipo nas mais diversas áreas de conhecimento. Segundo os trabalhos de [2, 9, 10, 11, 5, 6, 12, 13] algumas dessas áreas são:

Engenharia : (Inteligência Computacional, Aprendizado de Máquina, Reconhecimento de Padrões, Engenharia Mecânica, Engenharia Elétrica). As aplicações típicas de clusterização em engenharia abordam, por exemplo, reconhecimento biométrico e de reconhecimento de fala, compressão de informação, remoção de ruído.

Ciência da Computação : (Mineração Web, Recuperação da Informação). Tem-se notado uma maioria de aplicações de clusterização em mineração web, análise de banco de dados, recuperação da informação.

Ciências Médicas : (Genética, Biologia, Microbiologia, Psiquiatria). Aplicações importantes incluem definição de taxonomia, identificação de genes e proteínas, diagnóstico e tratamento de doenças.

Astronomia : (Geografia, Geologia, Sensoriamento Remoto). Clusterização pode ser utilizada para classificar estrelas e planetas, investigar formações rochosas e estudar os mecanismos de funcionamento de sistemas de rios e montanhas.

Ciências Sociais : (Sociologia, Psicologia, Arqueologia, Antropologia, Educação). Aplicações interessantes podem ser encontradas na história evolutiva dos idiomas, na análise de redes sociais, na classificação de artefatos arqueológicos e no estudo da psicologia criminal.

Economia : (*Marketing*, Empresas). Algumas aplicações a partir da análise de *clusters* são: reconhecimento de padrões de compra, agrupamento de empresas e análise de tendências.

Além da sua importância prática, trata-se de um problema difícil de ser resolvido na otimalidade e em tempos computacionais aceitáveis, visto pertencer à classe NP-difícil [14]. Essa conjunção de aplicabilidade e dificuldade de resolução na otimalidade motiva, ainda mais, os pesquisadores a desenvolverem algoritmos eficientes para resolvê-lo.

1.1 Motivação

Problemas de Edição de *Clusters* são amplamente etudados na literatura devido a sua dificuldade de resolução e alta aplicabilidade nas mais diversas áreas. O PENAC, embora seja um problema com muitas aplicações potenciais, é um problema ainda pouco explorado na literatura quando comparado ao PEC clássico. Além do mais, apresenta certo grau de dificuldade para se encontrar soluções ótimas mesmo em grafos com poucos vértices.

1.2 Objetivos do trabalho

Este trabalho tem como objetivo geral o desenvolvimento de um algoritmo eficiente de otimização, baseado nas metaheurísticas *Iterated Local Search* e *Variable Neighborhood Descent*, para resolver o PENAC. Objetiva-se também fazer o uso de uma abordagem exata na tentativa de resolução do problema.

1.2.1 Objetivos específicos

São os seguintes os objetivos específicos:

1. Fazer uma revisão de literatura do PENAC e de suas técnicas de solução.
2. Propor um modelo matemático para sua resolução de forma exata.
3. Avaliar o desempenho do algoritmo heurístico desenvolvido com base nos resultados do modelo matemático proposto.

1.3 Estrutura do trabalho

O presente trabalho está dividido em 7 capítulos, incluindo esta introdução.

No Capítulo 2 são estabeleidas algumas definições e notações utilizadas ao longo deste trabalho.

O Capítulo 3 apresenta a definição e a literatura existente do problema abordado neste trabalho, o Problema de Edição Não Automática de *Clusters*, bem como problemas similares. Apresentam-se também, revisões dos métodos heurísticos utilizados no algoritmo proposto.

No Capítulo 4 é proposto um modelo de programação linear, bem como algumas variações do mesmo.

No Capítulo 5 é apresentado, com riqueza de detalhes, o algoritmo heurístico utilizado para resolver o PENAC. A representação de uma solução do problema, a função de avaliação, as estruturas de vizinhança utilizadas para percorrer o espaço de soluções e o método para geração de soluções iniciais também são detalhados.

No Capítulo 6 são apresentados e analisados os resultados computacionais e no Capítulo 7 são apresentadas as conclusões e apontadas as sugestões para trabalhos futuros.

2 Definições e Complexidade de Algoritmos

A seguir, são estabelecidas algumas definições e notações que serão utilizadas ao longo deste trabalho.

2.1 Grafos - Definições e Notações

Um *grafo simples* é um par ordenado $G = (V, A)$, sendo V um conjunto finito não-vazio de *vértices*, denotado por $V(G)$ e A um conjunto de pares não-ordenados de vértices distintos, chamados *arestas*, e denotado por $A(G)$. As notações $n = |V(G)|$ e $m = |A(G)|$ denotam a cardinalidade de $V(G)$ e $A(G)$, respectivamente. Um grafo completo de n vértices é aquele que possui

$$m \ = \ n \ \times (\ n \ - \ 1 \ ) \ / \ 2$$

arestas. Denotamos por K_n o grafo completo com n vértices.

Com intuito de simplificar a terminologia utilizada neste trabalho, uma aresta entre quaisquer dois vértices, v_i e v_j, é representada por v_iv_j; neste caso, dizemos que v_i e v_j são adjacentes. Note que um grafo G é completo se quaisquer dois vértices distintos de G são adjacentes.

O *complemento* de um grafo G, denotado por $\overline{G}$, é o grafo que possui o mesmo conjunto de vértices de G tal que dois vértices são adjacentes em $\overline{G}$ se e somente se não são adjacentes em G. O conjunto de arestas de $\overline{G}$, denotado por $\overline{A}(G)$, tem cardinalidade definida por

$$\left|\overline{A}(G)\right| \ = \ (\ n \ \times (\ n \ - \ 1 \ ) \ / \ 2) \ - \ m$$

Um grafo G é dito *trivial* se $|V(G)| = 1$, isto é G possui um único vértice. Um laço,

ou *loop*, é definido por uma aresta v_iv_j tal que $i = j$. Em um grafo direcionado uma aresta v_iv_j não equivale à aresta v_jv_i.

Vértices adjacentes v_i e v_j são também chamados *vizinhos*, e a aresta $a = v_iv_j$ é *incidente* a v_i e a v_j, ou tem *extremos* v_i e v_j. Denota-se por $N(v)$ o conjunto de vértices adjacentes a v em G sendo tal conjunto denominado *vizinhança* de v. Já o conjunto $\overline{N}(v)$ é composto pelos vértices que não fazem parte da vizinhança de v.

Uma edição em um grafo G corresponde à remoção de uma aresta pertencente a $A(G)$ ou à adição de uma aresta pertencente à $\overline{A}(G)$. O número de operações de adição e remoção de arestas são denotados por a^+ e a^-, respectivamente.

A densidade δ de um grafo G é dada por $\delta = m \;/\; \binom{n}{2}$.

O *grau* de um vértice $v_i \in V(G)$, denotado por $d(v_i)$, é o número de arestas incidentes ao vértice v_i.

Um grafo H é um *subgrafo* de um grafo G se $V(H) \subseteq V(G)$ e $A(H) \subseteq A(G)$. Dado um conjunto de vértices $Y \subseteq V(G)$, $Y \neq \emptyset$, o *subgrafo induzido* por Y, denotado por $G[Y]$ é o subgrafo H de G tal que $V(H) = Y$ e $A(H)$ é o conjunto das arestas de G que tem ambos os extremos em Y.

Um *passeio* em um grafo G é uma sequência de vértices, cada qual adjacente ao vértice que lhe sucede na sequência. Um *caminho* num grafo G é um passeio $P = v_1, v_2, \ldots, v_k$ onde os $v_i's$ são vértices (dois a dois distintos), e $(v_i, v_{i+1}) \in A(G)$, $1 \leq i \leq k-1$.

Um conjunto de vértices C de um grafo G é uma *clique* se $G[C]$ é um grafo completo. Denota-se por K_k uma clique de k vértices.

Um grafo clusterizado, ou um conjunto de vértices k de um grafo G, é um *cluster* se $G[k]$ é uma clique, ou ainda, um grafo completo. O número de *clusters* em um grafo é, no mínimo, igual a um e, no máximo igual a n, isto é $k \in \{1, 2, 3, ..., n\}$. Um grafo k-clusterizado possui k *clusters*. No caso em que k assume o menor valor possível o grafo G é um grafo completo. Já no caso em que k assume o maior valor possível o grafo G possui $m = 0$.

Sejam $v_i, v_j \in V(G)$. A *distância* entre v_i e v_j em G, denotada por $d_G(v_i, v_j)$ é o comprimento do menor caminho entre v_i e v_j, em G. Em outras palavras, a distância entre quaisquer dois vértices, v_i e $v_j \in V(G)$ é dada pelo menor número de arestas que, juntas, conectam os vértices v_i e v_j em G. Assume-se $d_G(v_i, v_j) = \infty$ quando não existe um caminho em G que conecta os vértices v_i e v_j.

Seja H um subgrafo induzido do grafo G com conjunto de vértices e arestas definidos por $V(H) = \{v_i,\ v_j,\ v_k\}$ e $A(H) = \{v_iv_j,\ v_jv_k\}$, respectivamente. Denomina-se C_n um grafo formado por um caminho com n vértices. Logo, o subgrafo H é um grafo C_3. Observa-se, nesse caso, que $d_H(v_i, v_k) = 2$.

2.2 Complexidade de Algoritmos

Um problema algorítmico ρ consiste de um conjunto I de entradas para o problema, chamado *conjunto de instâncias*, e de uma *questão* Q sobre estas instâncias. Resolver um problema algorítmico é desenvolver um algoritmo cuja entrada é uma instância do problema e cuja saída é uma resposta à questão de tal problema.

Um problema é dito de *decisão* quando a questão exige uma resposta do tipo SIM ou NÃO.

Um problema é dito de *otimização* quando a questão exige a melhor resposta dentre todas possíveis. Problemas de otimização podem ser associados a problemas de *decisão*, uma vez que em ambos os tipos de problema busca-se encontrar respostas do tipo SIM ou NÃO. A diferença é que, em problemas de otimização, deseja-se que o tipo de resposta SIM esteja acompanhado da melhor solução possível. Portanto, um problema de *otimização* ρ pode ser representado pela quádrupla $(I_\rho,\ S_\rho,\ m_\rho,\ g_\rho)$, onde:

- I_ρ é o conjunto de instâncias de ρ.
- S_ρ é a função que associa um conjunto de soluções a cada instância $i \in I_\rho$.
- m_ρ é uma função de medida, definida por pares (i, s) tais que $i \in I_\rho$ e $s \in S_\rho$. Para cada par (i, s), dada uma instância i, $m_\rho(i, s)$ associa um valor v referente ao custo de s.
- $g_\rho \in \{MIN, MAX\}$ especifica se ρ é um problema de *minimização* ou de *maximização*.

Em um problema de *minimização*, deseja-se encontrar uma solução s para uma dada instância i onde $m(i, s)$ apresenta o menor valor possível. Já em um problema de *maximização* o objetivo é encontrar uma solução s para uma dada instância i onde $m(i, s)$ apresenta o maior valor possível. Em ambos os casos, as soluções $s \in S_\rho$, denotadas por s^*, para as quais $m(i, s^*)$ assume o melhor valor possível são ditas ótimas.

Define-se um algoritimo *polinomial* quando sua complexidade de tempo (medida do número de passos que o algoritmo efetua) é uma função polinomial no tamanho da sua entrada. Os problemas de *decisão* para os quais existem algoritmos polinomiais constituem a classe P. Tais problemas são chamados *polinomiais*.

Um problema de *decisão* é *não-determinístico polinomial* quando qualquer instância que produz resposta SIM possui um *certificado* sucinto, isto é, uma comprovação de que a resposta SIM é de fato verificável em tempo polinomail no tamanho da instância. Esta classe de problemas de *decisão* é a classe *NP*.

A classe *Co-NP* é formada pelos problemas que possuem um certificado sucinto para as instâncias que produzem resposta NÃO.

Sejam $\rho_1(I_1, Q_1)$ e $\rho_2(I_2, Q_2)$ dois problemas de *decisão*. Uma *tranformação* ou *redução polinomial* de ρ_1 em ρ_2 é uma função $f : I_1 \longrightarrow I_2$ tal que as seguintes condições são satisfeitas:

1. f pode ser calculada em tempo polinomial.
2. para toda instância $i \in I_1$, tem-se que i produz resposta SIM para ρ_1 se e somente se $f(i)$ produz resposta SIM para ρ_2.

Um problema de *decisão* ρ pertence à classe *NP-completo* quando as seguintes condiçes são satisfeitas:

1. $\rho \in NP$.
2. para todo problema $\rho' \in NP$ existe uma transformação polinomial de ρ' em ρ.

Um problema pertencente à classe NP-completo é chamado NP-completo. Para provar que um certo problema ρ é NP-completo, basta mostrar que $\rho \in NP$ e que existe uma transformação de um problema NP-completo ρ' em ρ.

Analogamente, prova-se que um problema de *decisão* ρ pertence à classe *Co-NP-completo* (e, neste caso, ρ é dito Co-NP-completo) quando $\rho \in$ Co-NP e existe um problema ρ' (Co-)NP-completo tal que:

1. se ρ' é NP-completo, existe uma função f que pode ser calculada em tempo polinomial tal que para toda instância i' de ρ', tem-se que i' produz SIM para ρ' se e somente se $i = f(i')$ produz NÃO para ρ.

2. se ρ' é Co-NP-completo, existe uma função f que pode ser calculada em tempo polinomial tal que para toda instância i' de ρ', tem-se que i' produz NÃO para ρ' se e somente se $i = f(i')$ produz NÃO para ρ.

2.3 Otimização e Aproximação

De acordo com a seção 2.2, um problema de otimização ρ possui um conjunto de instâncias e outro de soluções, uma fução de medida a ser minimizada ou maximizada de acordo com o parâmetro que especifica o tipo do problema.

O subconjunto de soluções ótimas $S^*_\rho \subseteq S_\rho$ é composto por soluções $s^* \in S^*_\rho$ para as quais $m(i, s^*)$ assume o melhor valor possível. O valor de uma solução ótima é denotado por m^*_ρ.

A partir de uma solução ótima s^*_ρ, pode-se compará-la a qualquer outra solução s_ρ a fim de se estabelecer uma relação de desempenho para algum algoritmo que encontre s_ρ. Esta relação mostra o quão boa a solução s_ρ é.

Em um problema de *minimização* ρ, a relação de aproximação de uma solução s para uma dada instância i é definida por

$$R(s, i) \;=\; \frac{m(i, s)}{m^*_\rho}$$

Por outro lado, em um problema de *maximização* tal relação é definida por

$$R(s, i) \;=\; \frac{m^*_\rho}{m(i, s)}$$

Dado um problema de *otimização* $\rho = (I_\rho, S_\rho, m_\rho, g_\rho)$, um algoritmo A_ρ que retorna uma solução $A_\rho(i) \in S_\rho(i)$ para cada instância $i \in I_\rho$ é um algoritmo $f(i)-$aproximado para ρ se, para cada instância $i \in \rho$:

- $R(i, A_\rho(i)) \leq f(i)$ e
- A_ρ retorna uma solução s em tempo polinomial (com base no tamanho de i).

A função $f(i)$ estabelece o fator de aproximação de A_ρ.

Embora exista uma variedade de classes de aproximação, neste trabalho, apenas a classe APX é utilizada. Tal classe é utilizada a fim de obter um fator constante de

aproximação de problemas de *otimização*. Dito de outra forma, APX é a classe de todos os problemas otimização ρ pertencentes à NP, tal que, para alguma constante $r \leq 1$, existe um algoritmo de r-aproximação para ρ.

Para mais detalhes sobre teoria da aproximação, indicam-se os os trabalhos [15] e [16].

3 PENAC - Descrição e Estado da Arte

O Problema de Edição Não Automática de *Clusters* é uma variante do Problema de Edição de *Clusters* clássico. O principal objetivo do Problema de Edição de *Clusters* é realizar o menor número de edições (adições ou remoções de arestas) em um dado grafo G que resulte em uma união disjunta de subgrafos completos, isto é, um grafo clusterizado.

No PENAC, em sua versão mais simples, as arestas não possuem pesos associados. Isto é, a relevância de uma edição em um grafo G é a mesma que qualquer outra edição, seja uma adição ou remoção de aresta de G. Diferentemente do PEC, na versão não automática o total de *clusters* presentes em uma solução deve ser equivalente a um valor pré-determinado, denotado por k. Tal característica é a única diferença entre o PEC e o PENAC. O objetivo do Problema de Edição Não Automática de *Clusters* é encontrar uma solução que minimize os custos referentes às edições realizadas em um grafo base G, satisfazendo às seguintes restrições de partição:

- Devem existir, exatamente, k *clusters* não vazios.
- Em uma solução válida, cada vértice só pode estar associado a um *cluster*.
- Cada cluster é um subgrafo completo.
- Não existe aresta entre clusters diferentes.

Uma solução G', ou um grafo clusterizado, para o Problema de Edição Não Automática de *Clusters* de um dado grafo G deve satisfazer $V(G') = V(G)$.

Sejam $A(G')^+$ e $A(G')^-$ os conjuntos de arestas que foram, respectivamente, adicionadas e removidas do grafo G no processo de obtenção de uma solução G' válida. O custo associado a G' é obtido de acordo com a seguinte equação: $custo(G') = |A(G')^+| + |A(G')^-|$. Em outras palavras, o custo de uma solução equivale ao total de edições realizadas em G. Quando o $custo(G')$ assumir o menor valor possível, define-se tal solução como ótima.

Embora o número de *clusters* de uma solução seja pré-determinado no PENAC, persiste o problema de determinar o menor número de edições a serem realizadas a fim de se obter uma união disjunta de subgrafos completos. Enquanto no PEC busca-se encontrar uma solução com o menor $custo(G')$ possível (não importando o valor final de k), no PENAC deseja-se, também, que tal solução possua um determinado número de *clusters* k. Como o PENAC pode ser reduzido do PEC, onde o número de *clusters* não é pré-determinado, é correto afirmar que tal problema pertence à classe NP-Difícil.

Para ilustrar o PENAC, considere a Figura 1. Nela, um grafo G não clusterizado com número de arestas $m = 3$ e número de vértices $n = 4$ é apresentado. Ainda na Figura 1, são apresentados 4 subgrafos clusterizados de G, sendo cada qual diretamente associado a um possível valor de k (número de *clusters*). Cada subgrafo possui um custo referente ao número de edições realizadas em G de modo a torná-lo um grafo clusterizado G'.

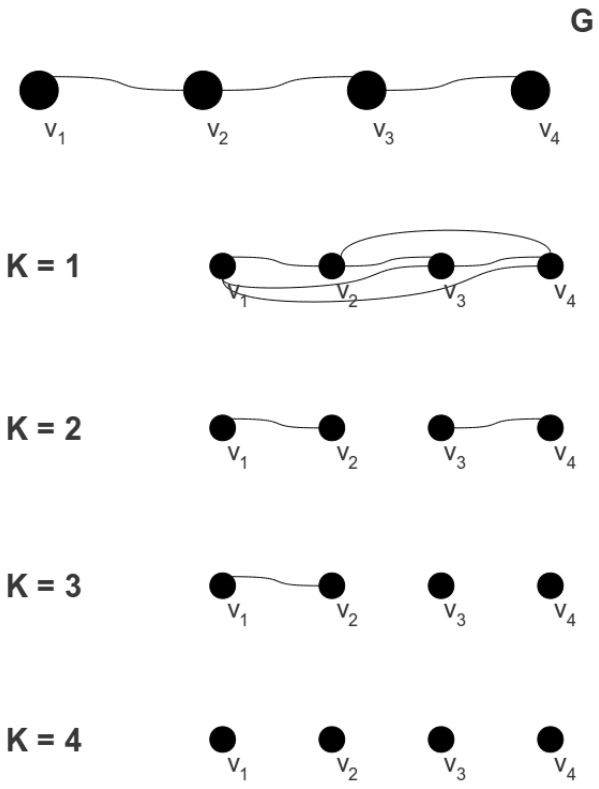

Figura 1: O Problema de Edição Não Automática de *Clusters*

Vale observar que o maior número possível de *clusters* $\max_k$ de um dado grafo G é definido pelo total de vértices n. Portanto, $\max_k = n$. Nesse caso, basta remover todas as arestas de G a fim de torná-lo um grafo clusterizado. Por sua vez, o menor número de *clusters* é sempre igual a 1. Já nesse caso, para tornar G um grafo clusterizado, basta adicionar todas as arestas faltantes. Os grafos à direita dos valores $k = 1$ e $k = 4$ na

Figura 1 ilustram, na devida ordem, as situações anteriormente descritas. Os grafos com $k = 2$ e $k = 3$ ilustram soluções clusterizadas para os devidos valores de k.

Considerando os vértices dos grafos da Figura 1 dispostos na ordem

$$V(G) = \{ v_1, v_2, v_3, v_4 \}$$

os conjuntos de arestas $A(G)$ e de não-arestas $\overline{A}(G)$ do grafo G são definidos por

$$A(G) = \{ v_1v_2, v_2v_3, v_3v_4 \}$$

$$\overline{A}(G) = \{ v_1v_3, v_1v_4, v_2v_4 \}$$

Conforme ilustrado na Figura 1, o menor número de edições que torna G um grafo clusterizado para $k = 1$ é $a^+ = 3$ e $a^- = 0$. Nesse caso, todas as arestas pertencentes a $\overline{A}(G)$ passam a pertencer a $A(G)$. Quando $k = 2$ a aresta v_2v_3 é removida de $A(G)$ passando a pertencer a $\overline{A}(G)$ e totalizando uma edição, isto é $a^- = 1$. Para $k = 3$ o número de edições corresponde a $a^- = 2$, uma vez que as arestas v_2v_3 e v_3v_4 foram removidas de $A(G)$. Por último, quando k assume seu valor máximo, $k = 4$, todas as arestas pertencentes a $A(G)$ são removidas, totalizando $a^- = 3$ edições.

Nos casos em que $k = 1$ ou $k = n$ basta um algoritmo de complexidade linear para clusterizar um dado grafo G. Note que quando $k = n$ a complexidade de tal algoritmo pode ser expressa como $O(A(G))$. No entanto, quando $k = 1$ sua complexidade é expressa como $O(n - A(G))$. Para qualquer outro número de *clusters* ($1 < k < n$) o PENAC apresenta uma complexidade de resolução de natureza combinatória. Tal característica implica em um elevado grau de dificuldade de resolução, uma vez que existem várias combinações de edições a serem avaliadas a fim de encontrar a melhor dentre todas.

No Problema de Edição Não Automática de *Clusters* tratado neste trabalho são considerados grafos simples e não nulos, isto é, grafos com conjunto de vértices não vazios, sem laços, não direcionados e com no máximo uma aresta entre quaisquer dois vértices. Além das características mencionadas anteriormente, assume-se que as arestas não possuem peso associado.

3.1 Definição formal do problema

De acordo com a terminologia utilizada, o problema abordado neste trabalho pode ser definido como: Dado um grafo simples e finito G, com conjunto de vértices $V(G)$ não

nulo e conjunto de arestas $A(G)$, objetiva-se encontrar, para um dado número de *clusters* k, o menor número de edições tais que, estas tornem o grafo G um grafo clusterizado G'. Em outras palavras, deseja-se encontrar uma solução ótima G' para o PENAC.

3.2 Literatura

Segundo [17], o problema de agrupar dados com algum grau de semelhança em *clusters* surge em vários contextos. Esse tipo de problema tem sido estudado na literatura de algoritmos e otimização combinatória, sendo que grande parte dessa literatura trabalha com a abstração do seguinte problema: Sejam x e y um par de dados obtidos de uma tabela de distâncias entre dados. A distância entre os dados x e y representa o grau de diferença entre eles. O objetivo é encontrar um agrupamento dos dados que otimize uma função das distâncias entre os dados, dentre ou entre os grupos formados, sob alguma restrição global, como por exemplo o conhecimento do número total de *clusters*.

Desde o seu estudo inicial, em [1], o Problema de Edição de *Clusters* tem apresentado grande relevância em outros trabalhos encontrados na literatura. Em [14] é demonstrado que tal problema pertence à classe NP-Difícil.

Enquanto alguns desses trabalhos aplicam técnicas heurísticas [2, 3, 4, 5, 12] na resolução do Problema de Edição de *Clusters*, outros, utilizam métodos exatos [7, 3, 4] ou, ainda, de algoritmos parametrizados [18] [19] [8].

Conforme [2, 9, 10, 11, 5, 6, 12], o Problema de Edição de *Clusters* é amplamente aplicável nas mais diversas áreas de conhecimento, tais como processamento de imagem e biologia computacional.

Segundo [13], a edição de *clusters* tem aplicabilidade nas seguintes áreas de conhecimento: Engenharia, Ciência da Computação, Ciências Médicas, Astronomia, Ciências Sociais, Economia, entre outras.

Uma formulação matemática para o Problema de Edição de *Clusters* foi proposta em [20]. Mais tarde, em [17], foi apresentada uma formulação simples e intuitiva para se resolver o problema $MinDisAgree$. Nesse problema, dado um grafo $G = (V, A)$ com rótulos "+" (acertos) e "-" (erros) nas arestas, objetiva-se encontrar um particionamento C, de modo que minimize os erros.

Parcialmente motivados por alguns problemas de aprendizagem de máquina relativos à classificação de documentos, [21] realizaram um estudo em busca de soluções aproximadas

para os problemas $MinDisAgree$ e $MaxAgree$. Dado um grafo $G = (V, A)$ com rótulos "+" (acertos) e "-" (erros) nas arestas, no problema $MaxAgree$ o objetivo é encontrar um particionamento C que maximize os acertos.

Note que estes dois problemas ($MinDisAgree$ e $MaxAgree$) são equivalentes quando comparadas as respectivas soluções ótimas, mas são diferentes no ponto de vista de aproximação.

Para o problema $MaxAgree$ em grafos completos existe um PTAS (*Polynomial Time Approximation Scheme* ou Esquema de Aproximação em Tempo Polinomial) apresentado por [22], já para o problema $MinDisAgree$ o melhor resultado é uma 2,5 aproximação, dada por [23].

Encontram-se na literatura algumas variações para os problemas $MaxAgree$ e $MinDisAgree$. Estas variações devem-se ao fato do grafo ser completo ou não, e na função de pesos das arestas. Nota-se que, segundo [24], no caso em que o grafo considerado não é um grafo completo, a resolução dos algoritmos $MaxAgree$ e $MinDisAgree$ torna-se mais difícil. Para o problema $MaxAgree$ o melhor resultado existente na literatura é um algoritmo 0,7664-Aproximado que utiliza programação semidefinida (subcampo de otimização convexa) apresentado em [25]. Já para o problema $MinDisAgree$, o melhor resultado é um algoritmo $\log n$-Aproximado apresentado simultaneamente por [26, 27, 25].

Em [17] demonstra-se que $MinDisAgree$ é APX-Difícil, e portanto não se pode esperar um PTAS para o problema de minimização semelhante ao PTAS para o $MaxAgree$.

Também, segundo [24], problemas de k-Clusterização introduzem novos desafios, não triviais, que requerem técnicas diferentes para as suas soluções. Nesse tipo de problema o número de *clusters* k é fixo. Denotam-se, respectivamente, $MaxAgree(k)$ e $MinDisAgree(k)$ os algoritmos $MaxAgree$ e $MinDisAgree$ que tratam de problemas k-Clusterizados.

3.3 Formulação Matemática para Problemas de Edição de *Clusters*

A formulação de Programação Linear Inteira (PLI) descrita abaixo é utilizada para resolver Problemas de Edição de *Clusters*. No Capítulo 4 um novo conjunto restrições é incorporado à formulação matemática descrita nessa seção. Tal conjunto tem como objetivo tornar possível a abordagem exata de Problemas de Clusterização onde o número

de *clusters* é predeterminado. Por ora, será apresentada uma breve descrição do modelo matemático utilizado em [17].

$$Minimizar \quad \sum_{i<j,\, ij \in A(G)} x_{ij} + \sum_{i<j,\, ij \in \overline{A}(G)} (1 - x_{ij}) \tag{3.1}$$

$$sa. \quad x_{ik} \leq x_{ij} + x_{jk} \quad \forall\, i,j,k \tag{3.2}$$

$$x_{ij} \in \{0,1\} \quad \forall\, i,j \tag{3.3}$$

Uma partição de *clusters* pode ser representada por um conjunto de variáveis binárias, onde cada variável está diretamente associada a uma respectiva aresta de um dado grafo G. Logo, para cada par de vértices $v_iv_j \in V(G)$ existe uma variável x_{ij}, binária, que assume valor igual a zero quando existe uma aresta entre os vértices v_i e v_j (indicando que ambos os vértices pertencem ao mesmo *cluster*) e, caso contrário, tal variável assume valor igual a um. Assumindo-se um terceiro vértice $v_k \in V(G)$ pertencente ao *cluster* que contém os vértices v_i e v_j observa-se que se $x_{ij} = 0$ e $x_{jk} = 0$ então $x_{ik} = 0$. Essa propriedade é representada pela Inequação (3.2). O objetivo é minimizar o número de edições, deseja-se encontrar uma solução em que o número de arestas removidas de $A(G)$ que tornam x_{ij} igual a um e o número de arestas adicionadas a $A(G)$ (ou removidas de $\overline{A}(G)$) que tornam x_{ij} igual a zero seja o menor possível. Ambos os objetivos são respectivamente representados pela Função (3.1). Ao remover uma ou mais arestas de $A(G)$ aumenta-se a primeira parcela da Função (3.1) e ao adicionar uma ou mais arestas a $A(G)$ aumenta-se a segunda parcela da respectiva função. A Equação (3.3) garante a integralidade e não negatividade das variáveis x_{ij}. Este modelo foi contemplado com o intuito de tratar grafos simétricos, ou seja, a variável x_{ij} representa a mesma aresta que a variável x_{ji}. Portanto, faz-se o uso apenas das variáveis x_{ij} onde $i < j$.

3.4 Heurísticas

As heurísticas são procedimentos empíricos que visam a obtenção de soluções de qualidade satisfatória em um tempo computacional aceitável. Tais procedimentos, no entanto, não garantem a obtenção da solução ótima para o problema nem são capazes de garantir o quão próximo a solução obtida está da ótima.

As heurísticas podem ser construtivas ou de refinamento. As construtivas têm por objetivo construir uma solução, elemento a elemento. A escolha de cada elemento está,

geralmente, relacionada a uma determinada função que o avalia de acordo com sua contribuição para a solução. Tal função é bastante relativa, pois varia conforme o tipo de problema abordado.

As heurísticas de refinamento, ou também chamadas de heurísticas de busca local, são técnicas baseadas na noção de vizinhança. Para definir o que é uma vizinhança, seja S o espaço de pesquisa de um problema de otimização e f a função objetivo a minimizar. O conjunto $N(s) \subseteq S$, o qual depende da estrutura do problema tratado, reune um número determinado de soluções s', denominado vizinhança de s. Cada solução $s' \in N(s)$ é chamada de *vizinho* de s e é obtido de s a partir de uma operação chamada de *movimento*.

Em linhas gerais, esses métodos partem de uma solução inicial s_0, percorrem o espaço de busca, através de um movimento, passando de uma solução para outra que seja sua vizinha.

A Subseção 3.4.1 descreve a heurística de refinamento utilizada neste trabalho, a saber, o Método Randômico de Descida.

3.4.1 Método Randômico de Descida

O Método Randômico de Descida (MRD) é uma heurística de refinamento que consiste em analisar um vizinho qualquer e o aceitar somente se ele for estritamente melhor que a solução corrente. Caso esse vizinho não seja melhor, a solução corrente permanece inalterada e outro vizinho é gerado. O método é finalizado quando se atinge um número máximo de iterações ($IterMax$) sem que haja a produção de melhoria na solução corrente.

O Algoritmo 1 apresenta o pseudocódigo do MRD aplicado ao refinamento de uma solução s em um problema de otimização (minimização ou maximização) de uma função $f(.)$, utilizando uma estrutura de vizinhança $N(.)$. Neste algoritmo, *IterMax* representa o número máximo de iterações sem melhora no valor da função de avaliação.

Algoritmo 1: Método Randômico de Descida (*MRD*)

Entrada: $f(.)$, N(.), *IterMax*, s

```
1  iter ← 0
2  enquanto iter < IterMax e critério de parada não satisfeito faça
3  |  iter ← iter + 1
4  |  Selecione aleatoriamente s' ∈ N(s)
5  |  se s' for melhor que s de acordo com a função f então
6  |  |  s ← s'
7  |  |  iter ← 0
8  |  fim
9  fim
10 retorna s
```

3.5 Metaheurísticas

As metaheurísticas são procedimentos destinados a encontrar uma boa solução, eventualmente a ótima, consistindo na aplicação, em cada passo, de uma heurística subordinada, a qual tem que ser modelada para cada problema específico [28].

Contrariamente às heurísticas convencionais, as metaheurísticas são de caráter geral e providas de mecanismos para tentar escapar de ótimos locais ainda distantes dos ótimos globais.

As metaheurísticas diferenciam-se entre si basicamente pelo mecanismo usado para sair das armadilhas dos ótimos locais. Elas se dividem em duas categorias, de acordo com o princípio usado para explorar o espaço de soluções: busca local e busca populacional.

Nas metaheurísticas baseadas em busca local, a exploração do espaço de soluções é feita por meio de movimentos, os quais são aplicados a cada passo sobre a solução corrente, gerando outra solução promissora em sua vizinhança. Busca Tabu, *Simulated Annealing*, Busca em Vizinhança Variável (*Variable Neighborhood Search*) e *Iterated Local Search* são exemplos de métodos que se enquadram nesta categoria.

Os métodos baseados em busca populacional, por sua vez, consistem em manter um conjunto de boas soluções e combiná-las de forma a tentar produzir soluções ainda melhores. Exemplos clássicos de procedimentos desta categoria são os Algoritmos Genéticos, os Algoritmos Meméticos e o Algoritmo Colônia de Formigas.

Nas Subseções 3.5.1 e 3.5.2 são apresentadas, respectivamente, as metaheurísticas *Variable Neighborhood Descent* e *Iterated Local Search*, referenciadas ao longo deste trabalho.

3.5.1 *Variable Neighborhood Descent*

O Método de Descida em Vizinhança Variável (*Variable Neighborhood Descent*, VND) [29], é um método de refinamento que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança, aceitando somente soluções de melhora da solução corrente e retornando à primeira estrutura quando uma solução melhor é encontrada.

Conforme os autores, o VND é baseado em três princípios básicos:

- Um ótimo local em relação a uma estrutura de vizinhança pode não ser um ótimo local relativo a outra estrutura de vizinhança;
- Um ótimo global é um ótimo local em relação a todas as possíveis estruturas de vizinhança;
- Para muitos problemas, ótimos locais com relação a uma ou mais estruturas de vizinhança são relativamente próximos.

Ainda segundo os autores, o último princípio, de natureza empírica, indica que um ótimo local frequentemente fornece algum tipo de informação sobre o ótimo global. Esse é o caso em que os ótimos local e global compartilham muitas variáveis com o mesmo valor, o que sugere uma investigação sistemática da vizinhança de um ótimo local até a obtenção de uma nova solução de melhor valor.

O pseudocódigo desse algoritmo, em que se considera o refinamento de uma solução s utilizando uma função de avaliação f, a ser minimizada, e um conjunto de r diferentes vizinhanças $N = \left\{N^{(1)}; N^{(2)}; ...; N^{(r)}\right\}$, é apresentado pelo Algoritmo 2.

Algoritmo 2: *Variable Neighborhood Descent* (*VND*)

Entrada: $f(.)$, N(.), r, s

```
1  iter ← 0
2  Seja r o número de estruturas diferentes de vizinhança
3  k ← 1        {Tipo de estrutura de vizinhança corrente}
4  enquanto  k ≤ r  e critério de parada não satisfeito faça
5  |  Encontre o melhor vizinho s' ∈ N^k(s)
6  |  se  s' for melhor que s de acordo com a função f então
7  |  |  s ← s'
8  |  |  k ← 1
9  |  fim
10 |  senão
11 |  |  k ← k + 1
12 |  fim
13 fim
14 retorna s
```

Dependendo do problema abordado, a busca pelo melhor vizinho (linha 5 do Algoritmo 2) pode ter um elevado custo computacional. Nessa situação é comum fazer a busca pela primeira solução de melhora. Outra alternativa é considerar a exploração apenas em um certo percentual da vizinhança.

3.5.2 *Iterated Local Search*

O *Iterated Local Search* - (ILS) [30], é um método de busca local focado em explorar um subespaço de soluções. Tal subespaço contém apenas soluções ótimas locais, dadas por um determinado mecanismo de otimização. De acordo com [30], justifica-se o sucesso do ILS com as escolhas do método de busca local, das perturbações e do critério de aceitação a serem utilizados. A busca baseada em soluções ótimas locais também é um fator determinante para o sucesso do método.

Mais especificamente, seja C a função custo do problema a ser minimizada, S o conjunto de todas as soluções onde $s \in S$ é uma solução candidata, deseja-se explorar S^*, um subconjunto de S, via um mecanismo de otimização que alterna entre as soluções $s^* \in S^*$.

Para tanto, uma vez determinada a solução corrente s^*, aplica-se uma perturbação a

essa solução. Como resultado desse procedimento obtem-se uma solução $s' \in S$, denominada solução intermediária. Vale ressaltar que a solução intermediária pode, ou não, ser um ótimo local. Após isso, o método de busca local é aplicado na solução s' encontrando-se uma solução $s^{*\prime} \in S^*$. O critério de aceitação determina se $s^{*\prime}$ deve, ou não, se tornar a nova solução corrente. Esse procedimento pode encontrar regiões promissoras no espaço de soluções.

Geralmente a caminhada do ILS não é reversível, isto é, se a caminhada for de s_1 para s_2, não se consegue vir de s_2 para s_1.

Como perturbações determinísticas podem conduzir a curtos ciclos, pode-se tornar o mecanismo de perturbação aleatório ou adaptativo afim de evitar esse tipo de ciclagem. Observa-se que tais perturbações devem ser fortes o suficiente para permitir o desvio do ótimo local corrente e, também, que o método de busca local explore diferentes regiões do espaço de soluções. Ao mesmo tempo, elas devem ser suficientemente fracas para preservar características do ótimo local corrente e evitar reinícios aleatórios.

O algoritmo 3 exibe o pseudocódigo do ILS básico.

Algoritmo 3: *Iterated Local Search* (*ILS*)

$s_0 \leftarrow GeraSolucaoInicial()$
$s \leftarrow BuscaLocal(s_0)$
enquanto *critério de parada não satisfeito* **faça**
  $s' \leftarrow Perturbacao(Historico,\ s)$
  $s'' \leftarrow BuscaLocal(s')$
  $s \leftarrow CriterioAceitacao(s',\ s'',\ Historico)$
fim
retorna s

4 Modelo Matemático

O modelo de Programação Linear Inteira (PLI) apresentado a seguir é baseado naquele descrito na Seção 3.3. Para sua representação, sejam os vértices i, $j \in V(G)$ tais que a variável x_{ij} indica a existência, ou não, de uma aresta entre os vértices i e j. Por sua vez, a variável y_j indica se o j-ésimo *cluster* se faz presente em uma dada solução para o PENAC. O modelo proposto é:

$$Minimizar \quad \sum_{i<j,\, ij \in A(G)} x_{ij} \; + \sum_{i<j,\, ij \in \overline{A}(G)} (1 - x_{ij}) \tag{4.1}$$

$$sa. \quad x_{ik} \;\leq\; x_{ij} \;+\; x_{jk} \qquad \forall\, i,j,k \tag{4.2}$$

$$y_j = 1 \qquad j = 1 \tag{4.3}$$

$$y_j = x_{ij} \qquad i = 1,\; j = 2 \tag{4.4}$$

$$y_j \leq \left(\tfrac{1}{j-1}\right) \times \sum_{i<j} x_{ij} \qquad \forall\, i,j \;\mid\; j \geq 3 \tag{4.5}$$

$$y_j \geq \Big(\sum_{i<j} x_{ij}\Big) \;-\; j \;+\; 2 \qquad \forall\, i,j \;\mid\; j \geq 3 \tag{4.6}$$

$$\sum_{j \in V(G)} y_j \;=\; k \tag{4.7}$$

$$x_{ij},\; y_j \in \{0,1\} \qquad \forall\, i,j \tag{4.8}$$

A função objetivo, representada pela Função (4.1), minimiza o número de arestas convertidas em não-arestas e o número de não-arestas convertidas em arestas. A Equação (4.8) garante a integralidade e não negatividade das novas variáveis x_{ij} e y_j. Nota-se que, de acordo com as restrições (4.4), (4.5) e (4.6), apenas $n-1$ variáveis y_j podem assumir os valores {0,1}. Como o número mínimo de *clusters* em uma solução é igual a um, assume-se que o primeiro *cluster* sempre integra a solução. A restrição (4.3) representa tal propriedade. De forma geral, cada *cluster* $y_j \mid j > 1$ está diretamente associado ao j-ésimo vértice. Um *cluster* y_j se faz presente em uma solução, assumindo valor igual a um, quando o conjunto de arestas, composto pelas variáveis $x_{ij} \mid i < j$, não apresenta

nenhuma adjacência ao j-ésimo vértice. Nesse caso, todas as variáveis $x_{ij} \mid i < j$ assumem valor igual a um. A restrição (4.7) estabelece a existência de exatamente k *clusters*.

Com intuito de melhor esclarecer o funcionamento do modelo proposto, mais especificamente das restrições (4.3), (4.4), (4.5) e (4.6), seja G' um grafo clusterizado com base no grafo G representados na Figura 2.

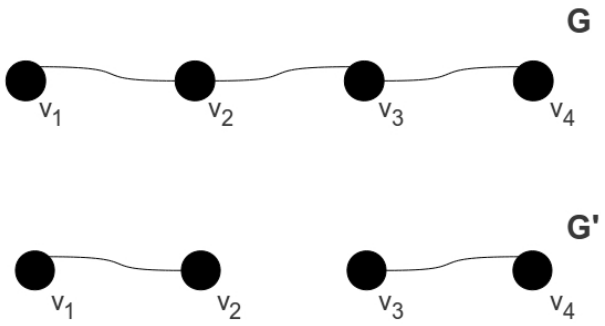

Figura 2: Grafo G' clusterizado com base no grafo G.

Observa-se que o número de *clusters* k no grafo G' é igual a dois. Sejam os vértices $\{1,\ 2,\ 3,\ 4\} \in V(G')$ apresentados, na devida ordem (da esquerda para a direita), na Figura 2. De acordo as restrições (4.3), (4.4), (4.5) e (4.6) tem-se:

$$y_1 = 1 \tag{4.9}$$

$$y_2 = x_{12} \tag{4.10}$$

$$y_3 \leq (x_{13} + x_{23})/2 \tag{4.11}$$

$$y_3 \geq x_{13} + x_{23} - 1 \tag{4.12}$$

$$y_4 \leq (x_{14} + x_{24} + x_{34})/3 \tag{4.13}$$

$$y_4 \geq x_{14} + x_{24} + x_{34} - 2 \tag{4.14}$$

A Equação (4.9) garante a ativação do *cluster* y_1, isto é, $y_1 = 1$. Os demais *clusters* são ativados conforme a Equação (4.10) e as Inequações (4.11), (4.12), (4.13) e (4.14). De acordo com a equação (4.10), o *cluster* y_2 é ativado se não existe aresta entre os vértices 1 e 2, isto é, $x_{12} = 1$. Contudo, o grafo G' apresentado na Figura 2 possui uma aresta tal que $x_{12} = 0$. Logo, $y_2 = 0$. O *cluster* y_3 é ativado conforme as Inequações (4.11) e (4.12). Como não existe aresta entre o terceiro vértice e qualquer outro vértice que o antecede ($x_{13} = 1$ e $x_{23} = 1$) ativa-se o *cluster* y_3. Enquanto a Inequação (4.11) estabelece um limite superior para y_3 tal que $y_3 \leq 1$ a Inequação (4.12) estabelece um

limite inferior tal que $y_3 \geq 1$. Logo, $y_3 = 1$. Por último, o *cluster* y_4 assume valor igual a zero de acordo com as Inequações (4.13) e (4.14). Como existe uma aresta entre os vértices 3 e 4 o *cluster* y_4 não é ativado. De acordo com a Inequação (4.13) o limite superior de y_4 é igual a 0,66. Já o limite inferior de y_4, estabelecido pela Inequação (4.14), é igual a -1. Dessa forma, $-1 \leq y_4 \leq 0,66$. Como o domínio das variáveis que compoem o modelo é definido com base nos valores {0,1}, o único valor possível que y_4 pode assumir é zero.

A Restrição (4.6) garante que apenas os *clusters* que fazem parte da solução podem ser ativados. Na ausência de tal Restrição, se o grafo G (Figura 2) possuísse apenas a aresta entre os vértices 1 e 2 e o número de *clusters* desejado fosse igual a dois, o próprio grafo G poderia ser apresentado como solução. Nota-se porém, que nesse caso, G possuiria três *clusters* o que torna inválida a solução apresentada. Como o limite inferior das variáveis y_3 e y_4 são iguais a zero o modelo assumiria o grafo G como um grafo {1,2,3}-clusterizado.

De forma resumida, uma vez estabelecida uma ordem para o conjunto de vértices de um dado grafo G basta analisá-los, um a um, a fim de identificar quais deles ativam os *clusters* de uma solução. Os vértices não adjacentes a seus antecessores (de acordo com a ordem estabelecida) ativam tais *clusters*. Como o primeiro vértice da sequência não possui nenhum antecessor assume-se que este sempre ativa o primeiro *cluster*.

O conjunto de restrições apresentadas até então garantem uma formulação matemática capaz de encontrar soluções ótimas para PENAC. Com intuito de reduzir o espaço de busca, facilitando assim, a obtenção de uma solução ótima para um dado grafo G, propõem-se as seguintes restrições:

$$\sum_{i<j}(1 - x_{ij}) \geq MinEd \qquad \forall\, i, j \tag{4.15}$$

$$\sum_{i<j}(1 - x_{ij}) \leq MaxEd \qquad \forall\, i, j \tag{4.16}$$

$$\sum_{i<j}(1 - x_{ij}) + \sum_{j<k}(1 - x_{jk}) \leq n - k \qquad \forall\, i, j, k \tag{4.17}$$

Os pares de vértices das Restrições (4.15), (4.16) e (4.17) pertencem ao conjunto de arestas $\{A(G) \cup \overline{A}(G)\}$.

As Restrições (4.15) e (4.16) estabelecem, respectivamente, o número mínimo ($MinEd$) e máximo ($MaxEd$) de arestas em um grafo clusterizado G. Mais especificamente, as Res-

trições (4.15) e (4.16) auxiliam o algoritmo de programação linear, utilizado pelo otimizador adotado neste trabalho, a encontrar limites, inferior e superior, de melhor qualidade ao iniciar uma otimização do PENAC.

Para um melhor entendimento sobre algoritmos de programação linear, bem como a relevância dos limites inferior e superior em um problema de otimização indica-se o trabalho de [31].

Os valores $MinEd$ e $MaxEd$ podem ser obtidos através do cálculo de todas as possibilidades de combinações dos n vértices de um grafo G em k *clusters*. Vale ressaltar que todo *cluster* deve possuir, pelo menos, um vértice e que ao enumerar todas as combinações, estamos também, especificando todas as possíveis soluções para o grafo a ser clusterizado. A Tabela 1 exibe todas as possibilidades de combinações para um grafo G com $n = 25$ e $k = 2$. Como cada *cluster* representa um grafo completo, faz-se o cálculo do número de arestas por *cluster* e, em seguida, somam-se os valores resultantes a fim de se obter o total de arestas para a combinação em questão. As colunas k_1, k_2, $|A_{k_1}|$, $|A_{k_2}|$ e *Total* representam, respectivamente, o número de vértices do *cluster* k_1, o número de vértices do *cluster* k_2, o total de arestas do *cluster* k_1, o total de arestas do *cluster* k_2 e a soma dos dois últimos valores. As combinações em que um *cluster* possui 24 vértices e o outro *cluster* possui um vértice resultam no maior número de arestas. Já as combinações em que um *cluster* possui 13 vértices e o outro *cluster* possui 12 vértices resultam no menor número arestas. Logo, os valores de $MinEd$ e $MaxEd$ são, respectivamente, 144 e 256.

Tabela 1: Possibilidades de combinações para um grafo G com $n = 25$ e $k = 2$.

k_1	k_2	$\lvert A_{k_1}\rvert$	$\lvert A_{k_2}\rvert$	Total
24	1	276	0	276
23	2	253	1	254
22	3	231	3	234
21	4	210	6	216
20	5	190	10	200
19	6	171	15	186
18	7	153	21	174
17	8	136	28	164
16	9	120	36	156
15	10	105	45	150
14	11	91	55	146
13	12	78	66	144
12	13	66	78	144
11	14	55	91	146
10	15	45	105	150
9	16	36	120	156
8	17	28	136	164
7	18	21	153	174
6	19	15	171	186
5	20	10	190	200
4	21	6	210	216
3	22	3	231	234
2	23	1	253	254
1	24	0	276	276

A Restrição (4.17) define o grau máximo dos vértices de um grafo clusterizado G'. Seja G um grafo, com conjunto de arestas $A(G) = \emptyset$, o qual deseja-se editar a fim de obter um grafo G' com número de *clusters* $k < n$. O maior grau possível de um vértice em G' é igual a $n - k$. Nesse caso, os vértices com grau $n - k$ formam um *cluster* e os demais vértices formam, individualmente, os demais $k - 1$ *clusters*.

5 Metodologia Proposta

Neste capítulo é apresentada a metodologia proposta para resolver o PENAC. A utilização de procedimentos heurísticos para resolvê-lo se deve à sua complexidade combinatória. O algoritmo utilizado neste trabalho combina as combina as metaheurísticas *Iterated Local Search* e *Variable Neighborhood Descent*.

Na Seção 5.1 mostra-se como uma solução do problema é representada, enquanto na Seção 5.2 são apresentados os movimentos utilizados para explorar o espaço de soluções do problema. Na Seção 5.3 mostra-se como uma solução é avaliada. O método de geração de uma solução inicial é apresentado na Seção 5.4. Por fim, na Seção 5.5, o algoritmo proposto para resolver o PENAC é descrito.

5.1 Representação de uma solução

Uma solução para o PENAC é representada por um vetor s de tamanho n. Cada posição do vetor s é representada pelos índices $i = \{0, 1, 2, ..., n\text{-}1\}$. O valor armazenado em cada posição de s indica a qual *cluster* pertence o i-ésimo vértice. Logo, seja uma solução $s = \{1, 1, 2, 3, 4\}$. Pode-se observar que os dois primeiros vértices pertencem ao *cluster* 1. Já os vértices 3, 4 e 5 estão associados aos *clusters* 2, 3 e 4, respectivamente.

5.2 Estruturas de Vizinhança

Para explorar o espaço de soluções são utilizados 5 tipos de movimento, a saber: *Swap*, *Replace* e *kOr-Opt* (com $k \in \{1,\ 2,\ 3\}$). Estes movimentos definem, na devida ordem, as estruturas de vizinhança N^{Swap}, $N^{Replace}$ e N^{OrOpt}.

Dado um conjunto com n vértices, há $(n \times (n\ -\ 1))/2$ possibilidades de trocas de *cluster* entre dois vértices selecionados. Na estrutura de vizinhança N^{Swap} seleciona-se

aleatoriamente dois vértices e realiza-se a troca do *cluster* de um pelo do outro. Quando os *clusters* dos vértices selecionados são iguais, faz-se uma nova seleção de dois novos vértices. Este procedimento é executado até que se obtenham vértices de *clusters* diferentes, tornando possível a aplicação de tal movimento.

Já na estrutura de vizinhança $N^{Replace}$, um vértice é realocado de um *cluster* k_1 para outro *cluster* k_2 tal que $k_1 \neq k_2$. Mais especificamente, seja uma solução $s' = \{1, 2, 3, 3, 2\}$ vizinha de v na vizinhança $N^{Replace}$. Observa-se que, dada a atual configuração da solução descrita, não se pode alocar o primeiro vértice a qualquer outro *cluster*. Um movimento factível (que não torna a solução inviável) seria alocar o vértice 4, pertencente ao *cluster* k_3 para o *cluster* k_2. Desse modo, o número de clusters $k = 3$ permanece inalterado.

Por fim, na estrutura de vizinhança N^{OrOpt}, um conjunto consecutivo de *vértices* é selecionado para, em seguida, serem realocados em outros *clusters*. Mais especificamente, seja uma solução $s' = \{1, 2, 3, 3, 2\}$ vizinha de v na vizinhança N^{OrOpt} e o conjunto de *vértices* $\{2, 3\}$ alocados aos *clusters* $\{1, 2\}$, respectivamente. Ao escolher, por exemplo, a última posição da solução corrente para realocar os *clusters* selecionados obtém-se uma nova solução $s'' = \{1, 3, 2, 2, 3\}$. Neste trabalho podem ser realocados, no máximo, três vértices.

5.3 Função de avaliação

Uma solução s' é avaliada por uma função f, devendo ser minimizada. Essa função contabiliza todas as alterações realizadas ao aplicar um, ou mais, movimentos à solução s a fim de se obter s'. Quanto menor o número de alterações, melhor é o resultado obtido. Essas alterações implicam em adições de arestas ou na remoção de arestas existentes.

Com intuito de melhor detalhar as etapas envolvidas no processo de avaliação de uma solução considera-se o grafo G com vértices $v \in \{1, 2, 3, 4\}$ representados na Figura 3. Seja o número de *clusters* $k = 2$. Deseja-se, então, obter um grafo clusterizado G' a partir do grafo G, de modo que G' possua exatamente dois *clusters*. Para tanto, basta remover a aresta v_2v_3 e adicionar a aresta v_2v_4 ao grafo G. O grafo clusterizado G', com $k = 2$, resultante da aplicação dos movimentos descritos anteriormente pode ser observado na Figura 3. De acordo com a função $f(s')$, onde $s' = \{1, 1, 2, 1\}$ a partir do grafo G', o número de alterações é igual a dois ($f(s') = 2$).

Outra possível solução, dado o grafo G, é remover a aresta v_1v_2. Assim, obtém-se um

novo grafo clusterizado G'' também com $k = 2$. Novamente, seja $s'' = \{1, 2, 2, 1\}$ o vetor solução do grafo G''. Logo $f(s'') = 1$.

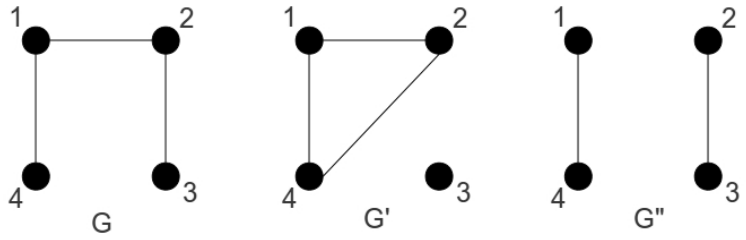

Figura 3: Grafos G, G' e G''

Com base nas avaliações dos grafos G' e G'' conclui-se que a solução apresentada pelo grafo G'' é melhor, quando comparada à solução de G'.

5.4 Geração de uma solução inicial

Nesta seção é apresentado um procedimento heurístico para a geração de uma solução inicial para o PENAC.

5.4.1 Construção Aleatória

Seja s um vetor solução para o o PENAC. Este método consiste em criar uma solução inicial s_i para o problema abordado. Com s_i inicialmente vazia, associa-se um *cluster* diferente a cada vértice até que a solução s_i contenha, uma vez, cada *cluster*. Ao final dessa etapa, devem existir exatamente k *clusters* contendo apenas um vértice cada. Os demais n - k vértices são, um a um, aleatoriamente associados aos *clusters*. Ao final desse processo, tem-se uma solução s_i clusterizada.

Como exemplo, considere uma solução inicial $s_i = \{1, 2, 2, 1\}$ para o grafo clusterizado G, onde $k = 2$. A Figura 4 exibe tal solução. Com s_i inicialmente vazia associam-se os vértices a e b aos *clusters* 1 e 2, respectivamente. O próximo passo a ser executado é, de maneira aleatória, associar um *cluster* aos vértices restantes. Nesse casso, aloca-se o vértice c no *cluster* 2 enquanto o vértice d é alocado no *cluster* 1.

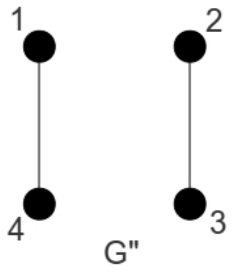

Figura 4: Solução Inicial

5.5 Algoritmo MILSRVND

O algoritmo adotado, denominado *Multistart Iterated Local Search with Random Variabale Neigbohrhood Descent* (*MILSRVND*) consiste na combinação de um algoritmo *Multistart*, que a cada iteração, gera uma solução inicial e, em seguida, a repassa à metaheurística *Iterated Local Search* (`ILS`).

O pseudocódigo do `MILSRVND` está esquematizado no Algoritmo 4. Nele, *MaxExecs* indica o número máximo de iterações que o algoritmo poderá ser executado enquanto *IterMax* indica o número máximo de iterações realizadas em um dado nível de perturbação.

O algoritmo é executado por, no máximo, *MaxExecs* vezes. Em cada execução, a construção da solução inicial s (linha 4 do Algoritmo 4) é realizada com base no método descrito na Seção 5.4.1.

Tal solução é submetida ao algoritmo `ILSRVND` (iniciado na linha 5 e concluído na linha 23 do Algoritmo 4).

A busca local (linha 14 do Algoritmo 4) faz o uso do procedimento `VND` com as estruturas de vizinhança descritas na Subseção 5.2. A cada etapa do `VND` busca-se encontrar o melhor vizinho em relação à vizinhança corrente. O `VND` utilizado pelo `MILSRVND` escolhe, aleatoriamente, a ordem em que as estruturas de vizinhança serão aplicadas. Tal procedimento de busca local é denominado `RVND`.

Quando um certo número de iterações sem melhora é alcançado, o `MILSRVND` aplica P vezes o procedimento `Shake` utilizando uma vizinhança escolhida previamente. O procedimento `SelectNeighborhood` (linha 11 do Algoritmo 4) consiste em escolher aleatoriamente uma vizinhança k entre N^{Swap} e $N^{Replace}$.

Cada chamada de $Shake(s', k)$ (linha 12 do Algoritmo 4) executa um movimento aleatório da vizinhança k da solução perturbada s'. Após *IterMax* iterações sem melhora,

p é incrementada de forma a gerar soluções cada vez mais distantes da solução corrente no espaço de busca.

Se o `RVND` encontra uma solução melhor, a variável p retorna ao menor valor, ou seja, $p = 0$.

Ao extrapolar o número máximo de iterações $IterMax$ ou satisfazer qualquer outra condição de parada, o algoritmo `MILSRVND` armazena a solução corrente, caso ela seja melhor que s^* (linhas 24 e 25 do Algoritmo 4). Em seguida, inicia-se um novo ciclo, onde uma nova solução inicial é construída e submetida ao `ILSRVND`.

Algoritmo 4: *MILSRVND*

Entrada: Função $f(.)$, *MaxExec*, *IterMax*, Nível P, r Vizinhanças: N^{Swap}, N^{Rplc} e N^{OrOpt}

```
1  s* ← ∅
2  exec ← 0
3  enquanto exec < MaxExecs e critério de parada não satisfeito faça
4      s ← generateInitialSolution()
5      p ← 0
6      enquanto critério de parada não satisfeito faça
7          iter ← 0
8          enquanto iter < IterMax e critério de parada não satisfeito faça
9              s' ← s
10             para i = 1 to P faça
11                 k ← SelectNeighborhood()
12                 s' ← Shake(s', k)
13             fim
14             s'' ← RVND(s')
15             se s'' for melhor que s de acordo com a função f então
16                 s ← s''; p ← 0; iter ← 0
17             fim
18             senão
19                 iter ← iter + 1
20             fim
21         fim
22         p ← p + 1
23     fim
24     se s for melhor que s* de acordo com a função f então
25         s* ← s
26     fim
27     exec ← exec + 1
28 fim
29 retorna s*
```

6 Resultados Computacionais

Neste capítulo são apresentados os resultados obtidos bem como os problemas-teste utilizados para avaliar os modelos exato e heurístico descritos nos capítulos 4 e 5, respectivamente.

6.1 Problemas-Teste

Os problemas-teste utilizados neste trabalho foram gerados com base no Algoritmo 5.

Algoritmo 5: *Geração Aleatória de um Problema-Teste*

Entrada: n, d

1 $ARESTAS \leftarrow generateAllEdges(n)$
2 $A \leftarrow \emptyset,\ \overline{A} \leftarrow \emptyset$
3 $d' \leftarrow 0$
4 **enquanto** *d' não for igual a d* **faça**
5 $\quad A \leftarrow A \cup pickSomeEdge(ARESTAS)$
6 $\quad d' \leftarrow \mid A \mid /n$
7 **fim**
8 $\overline{A} \leftarrow (ARESTAS - A)$
9 **retorna** $(A, \overline{A})$

São considerados o número de vértices n e a densidade d para gerar um grafo G. O método *generateAllEdges* (linha 1 do Algoritmo 5) retorna o conjunto de todas as possíveis arestas dado um número de vértices n. Por sua vez, o método *pickSomeEdge* escolhe, de maneira aleatória, uma aresta pertencente ao conjunto $ARESTAS$ para compor o grafo desejado (linha 5 do Algoritmo 5). Essa última operação é repetida até que seja alcançada a densidade desejada para o grafo (linha 4 do Algoritmo 5). As demais arestas compoem

o conjunto de não arestas $\overline{A}$ (linha 8 do Algoritmo 5).

6.2 Modelo Matemático

O modelo matemático proposto foi implementado na linguagem C++ utilizando a tecnologia CPLEX/Concert. O compilador utilizado foi o *gcc 4.4* e a máquina utilizada foi um Intel Core i7 2,93GHz, 8GB de memória, sistema operacional Ubuntu 10.04 kernel 2.6.32-28-generic. Foi utilizado um conjunto reduzido de instâncias a fim de, apenas, justificar a utilização das Restrições (4.15), (4.16) e (4.17) descritas no Capítulo 4.

Os testes envolvendo o modelo matemático foram divididos em 4 etapas, a saber:

I : Modelo Simples (MS): Compreende as Restrições (4.2), (4.3), (4.4), (4.5), (4.6), (4.7) e (4.8) do Capítulo 4.

II : MS + MinMaxArestas: Além das restrições do MS possui as Restrições (4.15) e (4.16) do Capítulo 4.

III : MS + GrauMax: Além das restrições do MS possui a Restrição (4.17) do Capítulo 4.

IV : Modelo Completo: Contempla os casos [I], [II] e [III] descritos anteriormente.

Os problemas-teste utilizados nesta etapa possuem $n = 25$ e $d = \{30, 40, 50, 60, 70\}\%$ e, para cada um deles, foi encontrada uma solução ótima. As Tabelas 2, 3 e 4 mostram os tempos (em segundos) despendidos, por cada abordagem exata (I, II, III e IV), para encontrar uma solução ótima.

Tabela 2: Tempos totais despendidos (em segundos), pelos modelos I, II, III e IV, para encontrar uma solução ótima para cada problema-teste com $n = 25$ e $d = \{30, 70\}\%$.

k	Densidade							
	30%				70%			
	I	II	III	IV	I	II	III	IV
1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
2	4119,69	10,92	**6,1**	6,54	**0,07**	0,11	0,18	0,17
3	4335,96	11,01	7,98	**7,97**	**0,21**	0,48	0,5	0,43
4	810,87	168,71	**41,8**	42,03	1,59	1,41	**0,95**	0,96
5	42,93	55,12	**29,81**	30,2	6,81	3,03	1,94	**1,93**
6	18,5	27,11	6,93	**6,92**	13,49	11,21	**8,67**	9,1
7	**6,15**	17,57	6,45	6,56	14,59	**10,91**	15,18	15,11
8	**3**	7,72	5,6	5,61	19,6	18,83	**17,45**	17,86
9	5,56	14	**4,9**	4,96	14,56	**13,29**	17,97	17,61
10	**4,73**	10,6	5,36	5,34	**20,35**	24,53	24,59	24,9
11	**3,41**	13,32	7,18	7,27	50,31	25,8	**24,33**	25,3
12	**4,67**	11	5,93	5,94	55,58	**31,49**	44,93	44,98
13	**4,28**	17,35	5,35	5,29	66,58	44,36	46,03	**43,64**
14	**4,44**	27,9	5,62	5,62	67,93	70,27	41,8	**40,88**
15	5,28	10,13	3,84	**3,83**	116,82	**65,07**	67,26	65,77
16	**2,35**	5,05	2,39	2,39	115,81	**48,31**	100,17	98,51
17	2,99	9,43	2,49	**2,47**	184,33	85,48	**58,37**	61,05
18	**2,02**	3,15	3,47	3,46	305,31	**17,83**	95,32	96,28
19	2,1	21,54	1,73	**1,71**	467,61	**3,62**	9,62	9,81
20	**1,27**	14,76	5,92	5,9	627,71	**5,48**	11,91	11,66
21	0,51	7,22	0,47	**0,46**	511,99	0,94	0,39	**0,38**
22	0,77	0,6	**0,26**	0,27	254,17	**0,36**	0,38	0,39
23	1,65	0,55	**0,24**	0,27	75,51	**0,34**	0,37	0,35
24	0,58	**0,02**	**0,02**	**0,02**	3,93	**0,02**	**0,02**	**0,02**
25	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Tabela 3: Tempos totais despendidos (em segundos), pelos modelos I, II, III e IV, para encontrar uma solução ótima para cada problema-teste com $n = 25$ e $d = \{40, 60\}\%$.

k	Densidade							
	40%				60%			
	I	II	III	IV	I	II	III	IV
1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
2	6619,91	8,2	5,69	**5,38**	7,54	**5,89**	6,76	6,51
3	264,82	**18,31**	39,49	36,3	**8,14**	8,9	11,15	11,12
4	58,27	39,86	**20,42**	20,53	10,9	**8,82**	10,34	10,59
5	**12,27**	12,84	16,23	15,85	16,95	**14,5**	15,28	15,46
6	**13,98**	17,04	16,55	16,55	**12,8**	16,56	16,25	16,76
7	17,88	16,94	**14,63**	15,18	22,65	19,25	**16,99**	17,09
8	**15,62**	18,26	41,11	38,4	22,83	**20,28**	20,77	20,84
9	18,86	17,31	17,17	**16,77**	44,97	**40,1**	47,26	44,35
10	46,06	**40,2**	41,78	41,69	64,48	68,38	56,42	**54,41**
11	**36,28**	43,67	40,56	38,89	**55,98**	58,73	79,51	79,36
12	41,01	43,72	35,2	**35,03**	**67,99**	70,87	173,37	171,33
13	**41,48**	42,7	46,68	46,04	119,23	**102,01**	121,87	121,92
14	37,22	**32,04**	34,35	34,53	75,94	**60,1**	118,86	119,46
15	33,31	**33,04**	36,98	37,11	122,51	75,72	70,88	**69,73**
16	33,04	33,58	**26,26**	26,3	190,78	77,38	**72,96**	73,89
17	26,51	27,31	**22,06**	22,27	131,17	**50,77**	68,19	68,51
18	28,17	22,79	**20,25**	20,66	59,21	**35,61**	65,62	64,63
19	**17**	24,26	18,56	18,99	149,63	**24,39**	43,85	43,06
20	14,89	10,82	10,5	**10,2**	90,8	**3,48**	6,16	6,6
21	13,95	2,63	2,34	**2,3**	191,79	13,24	**0,43**	0,53
22	14,2	0,36	**0,35**	0,37	99,01	**0,51**	0,56	0,56
23	12,5	0,32	**0,27**	**0,27**	55,31	**0,25**	**0,25**	**0,25**
24	0,81	**0,02**	**0,02**	**0,02**	1,92	**0,02**	**0,02**	**0,02**
25	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Tabela 4: Tempos totais despendidos (em segundos), pelos modelos I, II, III e IV, para encontrar uma solução ótima para cada problema-teste com $n = 25$ e $d = 50\%$.

k	Densidade = 50%			
	I	II	III	IV
1	**0**	**0**	**0**	**0**
2	310,1	**11,22**	12,31	13,11
3	**55,41**	87,78	65,65	66,26
4	17,77	**17,49**	20,72	20,84
5	**17,45**	24,06	18,02	18,46
6	**22**	42,73	44,99	44,49
7	59,27	**23,07**	38,94	40,07
8	**45,73**	50,92	49,73	50,5
9	67,18	77,63	54,51	**53,25**
10	59,36	66,92	**53,68**	55,27
11	72,05	**53,55**	60,8	59,08
12	94,07	95,35	**82,16**	82,98
13	111,97	77,59	77,02	**76,11**
14	**84,2**	102,86	125,16	126,9
15	145,22	**107,12**	117,02	119,38
16	82,68	95,12	**77,95**	79,12
17	93,01	**87,89**	103,26	102,47
18	58,35	70,05	53,45	**53,07**
19	76,09	30,33	**25,97**	26,02
20	59,27	29,61	17,18	**17,01**
21	33,33	3,03	**2,32**	**2,32**
22	35,31	0,42	**0,35**	0,4
23	29,61	**0,22**	0,26	0,24
24	1,66	**0,02**	**0,02**	**0,02**
25	**0**	**0**	**0**	**0**

Observa-se que no caso em que o número de *clusters* é igual a 1 basta adicionar todas as arestas faltantes e no caso em que o número de *clusters* é igual à n basta remover todas as arestas existentes. Logo, o tempo de execução é ignorado, conforme exibido nas Tabelas 2, 3 e 4. Os valores em negrito indicam o menor tempo despendido ao resolver a instância em questão. Ao todo, desconsiderando os casos citados anteriormente, foram realizados 460 testes (4 modelos $\times$ 23 vértices $\times$ 5 níveis de densidade). O modelo matemático (II) foi o que mais obteve sucesso em provar a otimalidade de uma solução em um menor intervalo de tempo. Ao todo, foram 39 provas contra 33 do modelo (III), 29 do modelo

(IV) e 28 do modelo (I). Se comparados em relação ao tempo total despendido para resolver as 23 instâncias nos 5 níveis de densidade os modelos matemáticos (II), (III) e (IV) se apresentam mais eficientes do que o modelo (I). Enquanto o modelo matemático (I) demandou 23.050,23 segundos para resolver as instâncias, os modelos (II), (III) e (IV) demandaram, respectivamente, 3.384,91 seg, 3.380,85 seg e 3.372,05 seg. Excluindo-se o modelo (I), os demais modelos são bem competitivos entre si. Embora o desempenho dos modelos (II), (III) e (IV) tenham sido bem semelhantes, adotou-se o modelo (IV) como principal. Isso se deve ao fato de que tal modelo considera mais características do problema. Portanto, nos demais testes, o modelo (IV) será comparado ao algoritmo heurístico utilizado nesse trabalho.

A Tabela 5 apresenta os valores ótimos referentes aos totais de edições a fim de se obter um grafo clusterizado para cada problema-teste proposto.

Tabela 5: Número de edições realizadas para encontrar uma solução ótima para cada problema-teste com $n = 25$ e $d = \{30, 40, 50, 60, 70\}\%$.

k	Densidade				
	30%	40%	50%	60%	70%
1	210	180	150	120	90
2	110	106	108	104	88
3	78	87	100	102	89
4	70	83	97	103	93
5	66	82	98	105	98
6	63	82	99	108	104
7	62	83	100	109	109
8	61	84	101	111	115
9	62	85	104	116	120
10	63	87	106	120	126
11	64	88	109	123	133
12	65	90	113	127	139
13	67	93	115	132	146
14	68	94	118	136	154
15	70	96	122	141	161
16	71	99	125	145	169
17	73	101	126	150	176
18	74	104	130	154	182
19	77	105	133	159	189
20	79	107	135	165	195
21	80	11	140	170	200
22	84	114	144	174	204
23	87	117	147	177	207
24	89	119	149	179	209
25	90	120	150	180	210

Com intuito de melhor analisar o desempenho do modelo (IV), em relação ao número de vértices e variação de densidade de um grafo, foram realizados novos testes com grafos de $n = \{25, 50\}$ vértices e densidades $d = \{5, 25, 50, 75, 95\}\%$. Salienta-se que os resultados obtidos, no caso em que $n = 25$ e $d = 50\%$, são os mesmos que os exibidos nas Tabelas 4 e 5. A Tabela 6 exibe os resultados obtidos para as respectivas densidades e $n = 25$.

Tabela 6: Modelo Matemático Completo: Tempo total despendido (em segundos) e número de edições realizadas para encontrar uma solução ótima para cada problema-teste com $n = 25$ e $d = \{5, 25, 50, 75, 95\}\%$.

k	Densidade									
	5%		25%		50%		75%		95%	
	FO	t(s)	FO	t(s)	FO	t(s)	FO	t(s)	FO	t(s)
1	285	0,00	225	0,00	180	0,00	75	0,00	15	0,00
2	129	0,51	113	51,69	106	5,38	77	0,14	31	0,10
3	79	6,72	78	34,55	87	36,30	80	0,55	48	0,33
4	55	1551,99	65	92,18	83	20,53	86	1,52	66	2,90
5	41*	3171,48	59	69,69	82	15,85	95	5,96	83	1,34
6	31	421,58	55	30,28	82	16,55	103	8,83	101	1,42
7	25	3020,81	53	5,09	83	15,18	112	12,05	118	6,16
8	20	67,15	52	3,60	84	38,40	120	16,58	134	6,49
9	16	6,81	51	1,29	85	16,77	127	20,05	149	8,50
10	13	0,59	51	1,88	87	41,69	135	56,81	165	1,21
11	12	1,02	52	1,33	88	38,89	142	51,86	180	5,02
12	11	0,54	53	1,84	90	35,03	150	67,76	194	3,54
13	10	0,33	54	1,28	93	46,04	159	61,67	207	23,07
14	9	0,30	55	1,04	94	34,53	167	82,75	219	17,40
15	8	0,29	56	1,11	96	37,11	174	225,85	230	25,48
16	7	0,27	58	1,34	99	26,30	182	98,05	240	22,88
17	7	0,26	59	1,03	101	22,27	189	19,13	249	2,00
18	8	0,27	61	0,74	104	20,66	197	14,43	257	2,60
19	9	0,29	62	0,65	105	18,99	204	6,24	264	2,94
20	10	0,29	64	1,33	107	10,20	210	5,37	270	15,65
21	11	0,29	65	0,49	110	2,30	215	0,54	275	1,76
22	12	0,26	69	0,38	114	0,37	219	0,55	279	0,72
23	13	0,25	72	0,26	117	0,27	222	0,41	282	0,53
24	14	0,02	74	0,02	119	0,02	224	0,02	284	0,02
25	15	0,00	75	0,00	120	0,00	225	0,00	285	0,00

De acordo com a Tabela 6, o modelo proposto só não foi capaz de encontrar uma solução ótima para a instância onde $k = 5$ e $d = 5\%$. Nesse caso, foram despendidos 3171,48 segundos até que toda a memória (RAM (8GB) + *Swap* (32GB)) disponível fosse utilizada. O melhor resultado obtido para tal instância foi de 41 edições, sendo este valor um limite superior. Já o melhor limite inferior encontrado para tal instância foi igual a 39. Portanto, o número ótimo de edições para tal instância corresponde a um dos seguintes valores $\{39, 40, 41\}$.

O modelo demandou um maior intervalo de tempo para resolver algumas instâncias com $d = 5\%$ quando comparado aos intervalos de tempos despendidos na resolução de instâncias com densidades mais elevadas. Os resultados referentes às instâncias com $n = 50$ vértices são melhor descritos na Seção 6.5.

6.3 Considerações sobre a medida *gap* utilizada

A medida *gap* é comumente utilizada para avaliar a qualidade de um limite superior ou inferior para um problema.

Assim como no trabalho de [32], utiliza-se o seguinte cálculo do *gap*:

$$gap = \left| \frac{ls - li}{li} \right| \tag{6.1}$$

onde: ls é um limite superior, li é um limite inferior e $|x|$ é o valor absoluto de x.

6.4 Algoritmo `MILSRVND`

O algoritmo heurístico utilizado neste trabalho, o `MILSRVND`, foi implementado na linguagem C++ no *framework* computacional OptFrame[1]. O compilador utilizado foi *gcc 4.4* e a máquina utilizada foi um Intel Core i7 2,93GHz, 8GB de memória, sistema operacional Ubuntu 10.04 kernel 2.6.32-28-generic. O conjunto de instâncias utilizadas compreende as instâncias com $n = 25$ e $d = \{5, 25, 50, 75, 95\}\%$. Esse algoritmo foi executado 10 vezes para cada problema-teste de forma a comprovar sua eficiência, mesmo que de forma empírica.

A calibração dos parâmetros do `MILSRVND` foi empírica e resultou nos seguintes valores: número de execuções do algoritmo por teste $ExecMax = \lfloor k/2 \rfloor$, número de iterações $IterMax = 5 \times n$; nível máximo de perturbação $p = 0,4 \times n$, sendo n o número de vértices e k o número de *clusters* da instância em questão. Tal calibração busca oferecer ao algoritmo um número de iterações suficiente para encontrar bons resultados nos problemas menores e também controlar o crescimento do tempo computacional do algoritmo nos problemas de dimensões mais elevadas.

Os resultados para 10 execuções do método `MILSRVND` são apresentados nas Tabelas 7, 8 e 9. As colunas **Instância**, **Sol. Int.**, **Tempo (s)**, **Melhor**, **Média**, ***gap*(%)**, **Var.**

[1] OptFrame está disponível na página *http://sourceforge.net/projects/optframe/* sob licença LGPLv3.

(%) e **Tempo Médio (s)** representam, respectivamente, a instância utilizada, a melhor solução inteira e o tempo despendido pelo modelo matemático para provar a otimalidade da solução encontrada, a melhor solução encontrada em 10 execuções do método `MILSRVND`, a média da soluções encontradas em 10 execuções do método `MILSRVND`, o *gap* percentual entre a melhor solução encontrada pelo método `MILSRVND` e o limite inferior estimado, a variabilidade percentual do método calculada de acordo com a Equação (6.2) e o tempo médio, em segundos, de 10 execuções do método. A coluna **Instância** está dividida em duas outras colunas, sendo que a primeira faz referência ao número de *clusters* k contidos na instância e a segunda à sua densidade d.

Assim como nos testes do modelo matemático, não foram testados os casos especiais em que $k = n$ e $k = 1$.

$$Var.(\%) = \frac{Media - Melhor}{Melhor} \times 100 \tag{6.2}$$

Tabela 7: Resultados MILSRVND (Instâncias com $n = 25$ e $d = \{5, 25\}\%$)

Instância		**Sol. Int.**	**Tempo (s)**	**Melhor**	**Média**	***gap*(%)**	**Var. (%)**	**Tempo Médio (s)**
k	**d**							
2	5%	129	**0,51**	129	129,00	0	0	1,07
3	5%	79	6,72	79	79,00	0	0	**6,2**
4	5%	55	1551,99	55	55,00	0	0	**0,23**
5	5%	39	3171,48	**41**	41,00	0,05	0	**88,43**
6	5%	31	421,58	31	31,00	0	0	**0,43**
7	5%	25	3020,81	25	25,00	0	0	**0,41**
8	5%	20	67,15	20	20,00	0	0	**0,25**
9	5%	16	6,81	16	16,00	0	0	**0,35**
10	5%	13	**0,59**	13	13,00	0	0	0,66
11	5%	12	1,02	12	12,00	0	0	**0,12**
12	5%	11	0,54	11	11,00	0	0	**0,09**
13	5%	10	0,33	10	10,00	0	0	**0,2**
14	5%	9	0,3	9	9,00	0	0	**0,18**
15	5%	8	**0,29**	8	8,00	0	0	1,13
16	5%	7	**0,27**	7	7,00	0	0	1,29
17	5%	7	**0,26**	7	7,00	0	0	1,33
18	5%	8	**0,27**	8	8,00	0	0	0,32
19	5%	9	0,29	9	9,00	0	0	**0,1**
20	5%	10	0,29	10	10,00	0	0	**0,07**
21	5%	11	0,29	11	11,00	0	0	**0,08**
22	5%	12	0,26	12	12,00	0	0	**0,04**
23	5%	13	0,25	13	13,00	0	0	**0,02**
24	5%	14	0,02	14	14,00	0	0	**0,01**
2	25%	113	51,69	113	113,00	0	0	**1,82**
3	25%	78	34,55	78	78,00	0	0	**0,81**
4	25%	65	92,18	65	65,00	0	0	**0,76**
5	25%	59	69,69	59	59,00	0	0	**7,36**
6	25%	55	30,28	55	55,00	0	0	**5,4**
7	25%	53	5,09	53	53,00	0	0	**2,11**
8	25%	52	3,6	52	52,00	0	0	**0,6**
9	25%	51	**1,29**	51	51,00	0	0	3,57
10	25%	51	1,88	51	51,00	0	0	**0,95**
11	25%	52	1,33	52	52,00	0	0	**0,63**
12	25%	53	1,84	53	53,00	0	0	**1,32**
13	25%	54	1,28	54	54,00	0	0	**0,71**
14	25%	55	**1,04**	55	55,00	0	0	5,77
15	25%	56	**1,11**	56	56,00	0	0	7,89

Tabela 8: Resultados MILSRVND (Instâncias com $n = 25$ e $d = \{25, 50, 75\}\%$)

Instância		**Sol. Int.**	**Tempo (s)**	**Melhor**	**Média**	***gap*(%)**	**Var. (%)**	**Tempo Médio (s)**
k	**d**							
16	25%	58	1,34	58	58,00	0	0	**0,98**
17	25%	59	**1,03**	59	59,00	0	0	3,92
18	25%	61	0,74	61	61,00	0	0	**0,58**
19	25%	62	**0,65**	62	62,00	0	0	0,99
20	25%	64	1,33	64	64,00	0	0	**0,79**
21	25%	65	**0,49**	65	65,00	0	0	2,71
22	25%	69	**0,38**	69	69,00	0	0	4,06
23	25%	72	0,26	72	72,00	0	0	**0,19**
24	25%	74	0,02	74	74,00	0	0	**0,01**
2	50%	108	13,11	108	108,00	0	0	**0,27**
3	50%	100	66,26	100	100,00	0	0	**0,42**
4	50%	97	20,84	97	97,00	0	0	**6,28**
5	50%	98	18,46	98	98,00	0	0	**7,64**
6	50%	99	44,49	99	99,00	0	0	**1,73**
7	50%	100	40,07	100	100,00	0	0	**6,27**
8	50%	101	50,5	101	101,00	0	0	**11,88**
9	50%	104	53,25	104	104,00	0	0	**3,05**
10	50%	106	55,27	106	106,00	0	0	**3,68**
11	50%	109	59,08	109	109,00	0	0	**5,92**
12	50%	113	82,98	113	113,00	0	0	**1,67**
13	50%	115	76,11	115	115,00	0	0	**3,04**
14	50%	118	126,9	118	118,00	0	0	**1,6**
15	50%	122	119,38	122	122,00	0	0	**10,57**
16	50%	125	79,12	125	125,00	0	0	**7,38**
17	50%	126	102,47	126	126,00	0	0	**42,53**
18	50%	130	53,07	130	130,00	0	0	**2,57**
19	50%	133	26,02	133	133,00	0	0	**1,97**
20	50%	135	17,01	135	135,00	0	0	**4,9**
21	50%	140	2,32	140	140,00	0	0	**0,37**
22	50%	144	0,4	144	144,00	0	0	**0,28**
23	50%	147	0,24	147	147,00	0	0	**0,04**
24	50%	149	0,02	149	149,00	0	0	**0**
2	75%	77	0,14	77	77,00	0	0	**0,11**
3	75%	80	0,55	80	80,00	0	0	**0,09**
4	75%	86	1,52	86	86,00	0	0	**0,08**
5	75%	95	5,96	95	95,00	0	0	**0,08**
6	75%	103	8,83	103	103,00	0	0	**0,15**
7	75%	112	12,05	112	112,00	0	0	**0,17**
8	75%	120	16,58	120	120,00	0	0	**0,3**

Tabela 9: Resultados MILSRVND (Instâncias com $n = 25$ e $d = \{75, 95\}\%$)

Instância		**Sol. Int.**	**Tempo (s)**	**Melhor**	**Média**	***gap*(%)**	**Var. (%)**	**Tempo Médio (s)**
k	**d**							
9	75%	127	20,05	127	127,00	0	0	**1,68**
10	75%	135	56,81	135	135,00	0	0	**0,74**
11	75%	142	51,86	142	142,00	0	0	**3,99**
12	75%	150	67,76	150	150,00	0	0	**0,13**
13	75%	159	61,67	159	159,00	0	0	**0,15**
14	75%	167	82,75	167	167,00	0	0	**0,18**
15	75%	174	225,85	174	174,00	0	0	**3,27**
16	75%	182	98,05	182	182,00	0	0	**4,49**
17	75%	189	19,13	189	189,00	0	0	**2,08**
18	75%	197	14,43	197	197,00	0	0	**0,13**
19	75%	204	6,24	204	204,00	0	0	**0,07**
20	75%	210	5,37	210	210,00	0	0	**0,06**
21	75%	215	0,54	215	215,00	0	0	**0,04**
22	75%	219	0,55	219	219,00	0	0	**0,06**
23	75%	222	0,41	222	222,00	0	0	**0,02**
24	75%	224	0,02	224	224,00	0	0	**0**
2	95%	31	0,1	31	31,00	0	0	**0,08**
3	95%	48	0,33	48	48,00	0	0	**0,08**
4	95%	66	2,9	66	66,00	0	0	**0,06**
5	95%	83	1,34	83	83,00	0	0	**0,07**
6	95%	101	1,42	101	101,00	0	0	**0,06**
7	95%	118	6,16	118	118,00	0	0	**0,07**
8	95%	134	6,49	134	134,00	0	0	**0,08**
9	95%	149	8,5	149	149,00	0	0	**0,14**
10	95%	165	1,21	165	165,00	0	0	**0,06**
11	95%	180	5,02	180	180,00	0	0	**0,05**
12	95%	194	3,54	194	194,00	0	0	**0,05**
13	95%	207	23,07	207	207,00	0	0	**0,06**
14	95%	219	17,4	219	219,00	0	0	**0,05**
15	95%	230	25,48	230	230,00	0	0	**0,05**
16	95%	240	22,88	240	240,00	0	0	**0,04**
17	95%	249	2	249	249,00	0	0	**0,04**
18	95%	257	2,6	257	257,00	0	0	**0,03**
19	95%	264	2,94	264	264,00	0	0	**0,03**
20	95%	270	15,65	270	270,00	0	0	**0,02**
21	95%	275	1,76	275	275,00	0	0	**0,02**
22	95%	279	0,72	279	279,00	0	0	**0,02**
23	95%	282	0,53	282	282,00	0	0	**0,01**
24	95%	284	0,02	284	284,00	0	0	**0**

De acordo com as Tabelas 7, 8 e 9, percebe-se que os *gaps* são nulos. Isto é, as soluções encontradas pelo método proposto MILSRVND são equivalentes ao limite inferior encontrado pelo modelo matemático.

Um resultado bem interessante pode ser visto na instância com $n = 5$ e $d = 5\%$, onde o método proposto conseguiu alcançar a mesma solução em 10 execuções, indicando a potencial qualidade da solução. A variabilidade do método também foi nula, indicando a robustez do método.

Como pode ser visto nas Tabelas 7, 8 e 9, o método MILSRVND proposto encontrou todos os ótimos provados pelo modelo matemático sendo que na maioria das vezes, o tempo despedindo pela abordagem heurística foi relativamente inferior ao tempo de execução do modelo.

6.5 Algoritmo MILSRVND e Modelo Matemático

Com intuito de melhor analisar o desempenho do modelo matemático, bem como do método heurístico, ambos foram submetidos a novos testes com instâncias de 50 vértices e densidades {5, 50, 95}%. Como, na maioria dos casos, o modelo matemático não foi capaz de provar a otimalidade da solução final, realizou-se um novo conjunto de testes onde as mesmas instâncias foram resolvidas em um processo de duas etapas. Na primeira etapa, executou-se o método MILSRVND por um tempo máximo de 600 segundos. Nos casos em que o modelo matemático encontrou uma solução ótima, o critério de parada do MILSRVND foi buscar por tal valor em um tempo máximo de 600 segundos. Já na segunda etapa, a solução proposta pelo método MILSRVND foi introduzida no modelo matemático como uma solução inicial.

Em todos os testes realizados, o principal critério de parada foi um valor predeterminado para o tempo limite de execução. Nos testes em que apenas o modelo matemático foi executado, foi estabelecido um tempo limite de 1800 segundos. Já o método MILSRVND e a metodologia que combinou a solução do MILSRVND com o modelo matemático foram executados, cada um, em um tempo limite de 600 segundos. É importante ressaltar que o método MILSRVND foi executado apenas uma vez para cada instância.

As Tabelas 10, 11, 12, 13, 14, 15 e 16 apresentam os resultados obtidos. As colunas **Instância**, **Modelo IV**, **MILSRVND**, **gap**(%), **MILSRVND + Modelo IV** e **gap**(%) representam, respectivamente, a instância utilizada, a solução encontrada pelo modelo matemático, a solução encontrada em uma execução do método MILSRVND, o *gap*

percentual entre a soluções encontradas pelo modelo matemático e pelo método MILSRVND, a solução encontrada pela metologia que combina o método MILSRVND com o modelo matemático e o *gap* percentual entre a soluções encontradas pelo método MILSRVND e tal metodologia. A coluna **Instância** está dividida em duas outras colunas, sendo que a primeira faz referência ao número de *clusters* k contidos na instância e a segunda à sua densidade d. A próximas três colunas fazem referência aos resultados obtidos pela execução do modelo matemático. A coluna **Sol. Int.** representa o valor da solução inteira encontrada pelo modelo matemático. Em outras palavras, esses valores representam o número mínimo de edições a serem realizadas tais que estas tornem o grafo proposto pela instância num grafo clusterizado. Os valores em negrito assinalam quando a solução ótima foi encontrada. A segunda coluna indica o valor dar melhor solução relaxada encontrada nos casos em que não se conseguiu provar o ótimo. Em todos os casos, tal valor atua com um limite inferior encontrado pelo modelo matemático. Observa-se que quanto mais próximo este valor estiver da solução inteira proposta, maior será a probabilidade de se encontrar o valor ótimo para a instância proposta. Nesta coluna, os valores em negrito indicam quando a solução relaxada encontrada pelo modelo matemático foi melhor que a solução relaxada encontrada pela metodologia que combina o método MILSRVND com o modelo matemático. A terceira coluna indica o tempo total (em segundos) despendido pelo modelo matemático para encontrar uma solução inteira. Os valores em negrito nesta coluna e nas demais colunas referentes a tempo de execução indicam um menor tempo despendido em relação às outras. As duas próximas colunas fazem referência à execução do método MILSRVND, sendo que a primeira coluna indica o valor da solução encontrada e a segunda coluna o respectivo tempo despendido. Os valores em negrito na primeira coluna indicam quando a solução ótima foi encotrada ou, nos casos em que não se obteve uma solução ótima, os valores menores que a solução encontrada pelo modelo matemático. A próxima coluna, já citada, apresenta os valores de *gap* percentuais. As três próximas colunas tem, respectivamente, o mesmo sentido que as três colunas descritas no caso do modelo matemático IV. Por fim, a última coluna indica o, já citado, *gap* percentual.

Tabela 10: Resultados Algoritmo MILSRVND e Modelo Matemático (Instâncias com $n = 50$ e $d = \{5, 50, 95\}\%$)

Instância		**Modelo IV**			**MILSRVND**		**gap**(%)	MILSRVND + **Modelo IV**			**gap(%)**
k	**d**	Sol. Int.	Sol. Rel.	Tempo (s)	Sol. Int.	Tempo (s)		Sol. Int.	Sol. Rel.	Tempo (s)	
2	5%	556	544,35	1800,21	**550**	600,39	**-0,01**	550	544,35	600,28	0,00
3	5%	353	**342,55**	1800,16	**349**	600,26	**-0,01**	349	342,36	600,13	0,00
4	5%	259	243,06	1800,12	**247**	600,67	**-0,05**	247	**243,08**	600,85	0,00
5	5%	195	**184,23**	1803,26	**192**	600,26	**-0,02**	192	183,64	600,94	0,00
6	5%	153	**146,00**	1803,26	**151**	600,02	**-0,01**	151	145,87	600,02	0,00
7	5%	130	**119,44**	1804,70	**125**	600,68	**-0,04**	125	119,09	601,38	0,00
8	5%	107	99,95	1805,63	**105**	600,27	**-0,02**	105	**100,21**	601,76	0,00
9	5%	90	**85,71**	1810,91	**88**	600,01	**-0,02**	88	85,58	602,04	0,00
10	5%	78	**73,58**	1804,93	**78**	600,69	0,00	78	72,88	603,20	0,00
11	5%	68	**66,67**	1802,19	**68**	600,21	0,00	68	65,89	603,00	0,00
12	5%	62	**59,00**	1805,90	**62**	600,01	0,00	62	58,69	604,52	0,00
13	5%	57	**55,11**	1805,25	58	600,02	**0,02**	57	55,00	606,48	**0,02**
14	5%	54	**51,97**	1822,58	**54**	600,9	0,00	54	51,67	607,72	0,00
15	5%	51	**49,00**	1806,98	**51**	600,07	0,00	51	48,50	609,63	0,00
16	5%	48	**47,00**	1801,83	**48**	600,79	0,00	48	46,33	607,81	0,00
17	5%	**45**		494,12	46	600,17	**0,02**	**45**		**45,68**	**0,02**
18	5%	**44**		**23,08**	**44**	69,06	0,00	**44**		101,57	0,00
19	5%	**43**		**19,44**	**43**	**10,09**	0,00	**43**		44,20	0,00
20	5%	**42**		7,29	**42**	28,39	0,00	**42**		**5,82**	0,00
21	5%	**41**		7,86	**41**	**1,67**	0,00	**41**		**6,75**	0,00
22	5%	**40**		8,03	**40**	37,41	0,00	**40**		**4,87**	0,00
23	5%	**39**		4,98	**39**	118,45	0,00	**39**		**2,98**	0,00
24	5%	**38**		2,98	**38**	50,91	0,00	**38**		**1,39**	0,00

Tabela 11: Resultados Algoritmo MILSRVND e Modelo Matemático (Instâncias com $n = 50$ e $d = \{5, 50, 95\}\%$)

Instância		**Modelo IV**			**MILSRVND**		**gap**(%)	MILSRVND + **Modelo IV**			**gap(%)**
k	**d**	Sol. Int.	Sol. Rel.	Tempo (s)	Sol. Int.	Tempo (s)		Sol. Int.	Sol. Rel.	Tempo (s)	
25	5%	**37**		2,74	**37**	10,37	0,00	**37**		**1,26**	0,00
26	5%	**37**		2,18	**37**	15,24	0,00	**37**		**1,11**	0,00
27	5%	**37**		2,02	**37**	13,00	0,00	**37**		**1,10**	0,00
28	5%	**37**		2,59	**37**	50,23	0,00	**37**		**1,20**	0,00
29	5%	**38**		2,77	**38**	20,65	0,00	**38**		**2,23**	0,00
30	5%	**39**		5,82	40	600,23	**0,03**	**39**		**4,87**	**0,03**
31	5%	**40**		3,62	**40**	24,87	0,00	**40**		**2,78**	0,00
32	5%	**41**		3,15	**41**	103,23	0,00	**41**		**2,81**	0,00
33	5%	**42**		3,37	**42**	15,82	0,00	**42**		**2,96**	0,00
34	5%	**43**		3,04	**43**	16,22	0,00	**43**		**2,57**	0,00
35	5%	**44**		3,01	**44**	**0,87**	0,00	**44**		**2,59**	0,00
36	5%	**45**		4,30	**45**	85,35	0,00	**45**		**2,83**	0,00
37	5%	**46**		3,75	**46**	71,72	0,00	**46**		**2,66**	0,00
38	5%	**47**		2,95	**47**	16,53	0,00	**47**		**2,42**	0,00
39	5%	**48**		2,79	**48**	75,70	0,00	**48**		**2,44**	0,00
40	5%	**49**		3,32	50	600,27	**0,02**	**49**		**2,43**	**0,02**
41	5%	**50**		2,39	**50**	480,48	0,00	**50**		**1,98**	0,00
42	5%	**51**		2,39	**51**	247,53	0,00	**51**		**1,91**	0,00
43	5%	**52**		2,31	**52**	66,13	0,00	**52**		**1,97**	0,00
44	5%	**53**		2,87	**53**	82,28	0,00	**53**		**1,96**	0,00
45	5%	**54**		9,26	**54**	9,30	0,00	**54**		**8,15**	0,00
46	5%	**55**		9,35	56	600,06	**0,02**	**55**		**5,28**	**0,02**
47	5%	**57**		8,07	**57**	32,67	0,00	**57**		**0,81**	0,00

Tabela 12: Resultados Algoritmo MILSRVND e Modelo Matemático (Instâncias com $n = 50$ e $d = \{5, 50, 95\}\%$)

Instância		**Modelo IV**			**MILSRVND**		**gap**(%)	**MILSRVND + Modelo IV**			**gap(%)**
k	**d**	Sol. Int.	Sol. Rel.	Tempo (s)	Sol. Int.	Tempo (s)		Sol. Int.	Sol. Rel.	Tempo (s)	
48	5%	**58**		0,89	**58**	122,58	0,00	**58**		**0,60**	0,00
49	5%	**60**		0,22	**60**	**0,14**	0,00	**60**		**0,20**	0,00
2	50%	559	**412,40**	1800,88	**476**	600,03	**-0,15**	476	410,78	600,14	0,00
3	50%	616	**356,87**	1800,13	**452**	600,15	**-0,27**	452	356,50	600,23	0,00
4	50%	602	**401,27**	1800,34	**444**	600,63	**-0,26**	444	306,00	612,56	0,00
5	50%	586	**396,43**	1800,38	**442**	600,34	**-0,25**	442	306,00	608,67	0,00
6	50%	564	**398,89**	1800,14	**447**	600,91	**-0,21**	447	306,00	603,51	0,00
7	50%	533	**397,39**	1800,30	**448**	600,06	**-0,16**	448	306,00	605,17	0,00
8	50%	559	**398,89**	1800,61	**449**	600,98	**-0,2**	449	306,00	600,15	0,00
9	50%	538	**400,77**	1800,27	**448**	600,98	**-0,17**	448	306,00	600,21	0,00
10	50%	530	**394,55**	1800,20	**451**	600,57	**-0,15**	451	306,00	600,11	0,00
11	50%	554	**397,71**	1800,12	**456**	600,12	**-0,18**	456	306,00	601,09	0,00
12	50%	546	**402,19**	1818,85	**456**	600,25	**-0,16**	456	306,00	606,90	0,00
13	50%	517	**400,00**	1800,34	**463**	600,98	**-0,1**	463	306,00	606,21	0,00
14	50%	522	**408,36**	1800,21	**465**	600,98	**-0,11**	465	306,00	600,24	0,00
15	50%	520	**411,02**	1800,17	**467**	600,71	**-0,1**	467	306,00	600,20	0,00
16	50%	533	**412,87**	1800,10	**472**	600,26	**-0,11**	472	306,00	611,74	0,00
17	50%	529	**415,79**	1800,29	**478**	600,42	**-0,1**	478	306,00	600,06	0,00
18	50%	534	**418,04**	1800,23	**477**	600,55	**-0,11**	477	306,00	600,82	0,00
19	50%	531	**419,06**	1805,45	**478**	600,47	**-0,1**	478	306,00	602,11	0,00
20	50%	536	**424,48**	1800,20	**484**	600,50	**-0,1**	484	306,00	614,59	0,00
21	50%	538	**425,66**	1800,22	**485**	600,88	**-0,1**	485	306,00	600,06	0,00
22	50%	529	**427,91**	1801,63	**491**	600,08	**-0,07**	491	306,00	615,05	0,00

Tabela 13: Resultados Algoritmo MILSRVND e Modelo Matemático (Instâncias com $n = 50$ e $d = \{5, 50, 95\}\%$)

Instância		**Modelo IV**			**MILSRVND**		**gap**(%)	**MILSRVND + Modelo IV**			**gap(%)**
k	**d**	Sol. Int.	Sol. Rel.	Tempo (s)	Sol. Int.	Tempo (s)		Sol. Int.	Sol. Rel.	Tempo (s)	
23	50%	543	**426,09**	1817,29	**500**	600,72	**-0,08**	500	306,00	607,69	0,00
24	50%	546	**399,86**	1832,50	**499**	600,25	**-0,09**	499	306,00	601,93	0,00
25	50%	545	**439,38**	1812,41	**501**	600,37	**-0,08**	501	306,00	601,05	0,00
26	50%	548	**445,89**	1800,28	**503**	600,39	**-0,08**	503	312,00	617,37	0,00
27	50%	553	**441,64**	1901,84	**508**	600,18	**-0,08**	508	336,00	600,10	0,00
28	50%	558	**400,59**	1800,30	**515**	600,47	**-0,08**	515	359,00	688,83	0,00
29	50%	558	**455,34**	1814,69	**517**	600,26	**-0,07**	517	381,00	622,38	0,00
30	50%	554	**461,69**	1800,38	**520**	600,19	**-0,06**	520	402,00	600,09	0,00
31	50%	552	**467,16**	1800,23	**526**	600,00	**-0,05**	526	422,00	606,10	0,00
32	50%	558	**474,86**	1800,19	**530**	600,98	**-0,05**	530	441,00	600,02	0,00
33	50%	551	**481,46**	1800,13	**535**	600,78	**-0,03**	535	459,00	601,46	0,00
34	50%	559	**492,84**	1800,13	**540**	600,19	**-0,03**	540	476,00	600,02	0,00
35	50%	570	492,00	1800,32	**546**	600,80	**-0,04**	546	492,00	600,14	0,00
36	50%	572	507,00	1800,55	**545**	600,16	**-0,05**	545	507,00	600,22	0,00
37	50%	570	521,00	1802,66	**551**	600,19	**-0,03**	551	521,00	600,16	0,00
38	50%	577	534,00	1801,92	**554**	600,54	**-0,04**	554	534,00	600,20	0,00
39	50%	578	546,00	1802,06	**556**	600,22	**-0,04**	556	546,00	600,14	0,00
40	50%	583	557,00	1804,75	**561**	600,38	**-0,04**	561	557,00	600,20	0,00
41	50%	583	567,00	1805,29	**569**	600,27	**-0,02**	569	567,00	600,08	0,00
42	50%	587	**576,00**	1802,90	**576**	600,30	**-0,02**	**576**		**1,62**	0,00
43	50%	594	**584,00**	1805,38	**584**	600,75	**-0,02**	**584**		**1,31**	0,00
44	50%	596	**591,00**	1803,60	**591**	600,00	**-0,01**	**591**		**1,42**	0,00
45	50%	**597**		871,99	**597**	**3,99**	0,00	**597**		**1,19**	0,00

Tabela 14: Resultados Algoritmo MILSRVND e Modelo Matemático (Instâncias com $n = 50$ e $d = \{5, 50, 95\}\%$)

Instância		**Modelo IV**			**MILSRVND**		**gap**(%)	MILSRVND + **Modelo IV**			**gap(%)**
k	**d**	Sol. Int.	Sol. Rel.	Tempo (s)	Sol. Int.	Tempo (s)		Sol. Int.	Sol. Rel.	Tempo (s)	
46	50%	**602**		904,69	**602**	**0,54**	0,00	**602**		**1,09**	0,00
47	50%	**606**		9,86	**606**	**4,08**	0,00	**606**		**1,10**	0,00
48	50%	**609**		2,56	**609**	**0,34**	0,00	**609**		**0,89**	0,00
49	50%	**611**		**0,18**	**611**	**0,09**	0,00	**611**		0,22	0,00
2	95%	**101**		10,35	**101**	**1,09**	0,00	**101**		**8,74**	0,00
3	95%	**141**		252,56	**141**	**0,7**	0,00	**141**		**188,76**	0,00
4	95%	**180**		1200,14	**180**	**0,75**	0,00	**180**		**524,36**	0,00
5	95%	218	**215,39**	1800,41	**218**	600,51	0,00	218	214,69	600,20	0,00
6	95%	255	251,28	1800,28	**255**	600,00	0,00	255	251,28	600,17	0,00
7	95%	291	287,08	1800,35	**291**	600,65	0,00	291	287,08	600,22	0,00
8	95%	**326**	323,52	1312,85	**326**	**0,85**	0,00	326	323,52	600,19	0,00
9	95%	360	356,55	1800,15	**360**	600,21	0,00	360	356,55	600,27	0,00
10	95%	393	389,40	1800,24	**393**	600,22	0,00	393	389,40	600,21	0,00
11	95%	427	422,85	1800,10	**427**	600,62	0,00	427	422,85	600,25	0,00
12	95%	557	422,85	1800,17	**460**	600,14	**-0,17**	460	**453,99**	600,25	0,00
13	95%	492	**485,80**	1800,13	**492**	600,18	0,00	492	484,64	600,22	0,00
14	95%	529	515,66	1800,16	**523**	600,76	**-0,01**	523	515,66	600,11	0,00
15	95%	559	546,74	1800,23	**555**	600,20	**-0,01**	555	546,74	600,17	0,00
16	95%	608	**573,44**	1800,17	**586**	600,79	**-0,04**	586	573,29	600,22	0,00
17	95%	624	**604,34**	1800,20	**616**	600,16	**-0,01**	616	604,06	600,28	0,00
18	95%	848	635,00	1800,26	**645**	600,78	**-0,24**	645	635,00	600,14	0,00
19	95%	846	667,00	1800,21	**673**	600,18	**-0,2**	673	667,00	600,16	0,00
20	95%	732	698,00	1800,27	**702**	600,86	**-0,04**	702	698,00	600,13	0,00

Tabela 15: Resultados Algoritmo MILSRVND e Modelo Matemático (Instâncias com $n = 50$ e $d = \{5, 50, 95\}\%$)

Instância		**Modelo IV**			**MILSRVND**		**gap**(%)	MILSRVND + **Modelo IV**			**gap(%)**
k	**d**	Sol. Int.	Sol. Rel.	Tempo (s)	Sol. Int.	Tempo (s)		Sol. Int.	Sol. Rel.	Tempo (s)	
21	95%	833	728,00	1800,17	**730**	600,09	**-0,12**	730	728,00	600,22	0,00
22	95%	900	757,00	1800,22	**759**	600,67	**-0,16**	759	757,00	600,23	0,00
23	95%	815	**785,00**	1800,18	**785**	600,30	**-0,04**	**785**		**0,50**	0,00
24	95%	828	**812,00**	1800,29	**812**	600,73	**-0,02**	**812**		**0,51**	0,00
25	95%	866	**838,00**	1800,26	**838**	600,24	**-0,03**	**838**		**0,49**	0,00
26	95%	885	**863,00**	1800,28	**863**	600,02	**-0,02**	**863**		**0,48**	0,00
27	95%	911	**887,00**	1800,17	**887**	600,71	**-0,03**	**887**		**0,51**	0,00
28	95%	951	**910,00**	1800,36	**910**	600,22	**-0,04**	**910**		**0,55**	0,00
29	95%	950	**910,00**	1800,25	**932**	600,89	**-0,02**	**932**		**0,55**	0,00
30	95%	977	**953,00**	1800,19	**953**	600,06	**-0,02**	**953**		**0,60**	0,00
31	95%	979	**973,00**	1800,24	**973**	600,60	**-0,01**	**973**		**0,56**	0,00
32	95%	1030	**992,00**	1801,10	**992**	600,38	**-0,04**	**992**		**0,59**	0,00
33	95%	**1010**		1178,85	**1010**	**0,87**	0,00	**1010**		**0,52**	0,00
34	95%	1033	**1027,00**	1800,34	**1027**	600,10	**-0,01**	**1027**		**0,63**	0,00
35	95%	1049	**1043,00**	1800,10	**1043**	600,33	**-0,01**	**1043**		**0,54**	0,00
36	95%	1066	**1058,00**	1800,01	**1058**	600,11	**-0,01**	**1058**		**0,59**	0,00
37	95%	**1072**		75,73	**1072**	**0,50**	0,00	**1072**		**0,58**	0,00
38	95%	1087	**1085,00**	1800,21	**1085**	600,00	0,00	**1085**		**0,63**	0,00
39	95%	1107	**1097,00**	1800,19	**1097**	600,42	**-0,01**	**1097**		**0,63**	0,00
40	95%	**1108**		14,19	**1108**	**0,24**	0,00	**1108**		**0,64**	0,00
41	95%	**1118**		4,93	**1118**	**0,69**	0,00	**1118**		**0,69**	0,00
42	95%	1129	**1127,00**	1800,20	**1127**	600,05	0,00	**1127**		**0,61**	0,00
43	95%	**1135**		3,57	**1135**	**0,27**	0,00	**1135**		**0,61**	0,00

Tabela 16: Resultados Algoritmo MILSRVND e Modelo Matemático (Instâncias com $n = 50$ e $d = \{5, 50, 95\}\%$)

Instância		**Modelo IV**			**MILSRVND**		**gap**(%)	MILSRVND + **Modelo IV**			**gap(%)**
k	**d**	Sol. Int.	Sol. Rel.	Tempo (s)	Sol. Int.	Tempo (s)		Sol. Int.	Sol. Rel.	Tempo (s)	
44	95%	**1142**		2,30	**1142**	**0,26**	0,00	**1142**		**0,61**	0,00
45	95%	**1148**		1264,73	**1148**	**0,37**	0,00	**1148**		**0,63**	0,00
46	95%	**1153**		5,54	**1153**	**0,16**	0,00	**1153**		**2,09**	0,00
47	95%	**1157**		4,28	**1157**	**0,08**	0,00	**1157**		**1,29**	0,00
48	95%	**1160**		1,51	**1160**	**0,07**	0,00	**1160**		**0,81**	0,00
49	95%	**1162**		**0,15**	**1162**	**0,00**	0,00	**1162**		0,20	0,00

Nos testes em que a densidade dos grafos era de 5%, o modelo matemático conseguiu provar a otimalidade de 33 das 48 instâncias propostas. O método MILSRVND, em apenas uma execução, encontrou a maioria dos valores ótimos, sendo o *gap* máximo igual a 3% considerando os casos onde o valor ótimo foi encontrado. Observa-se também, uma melhoria de qualidade das soluções encontradas em alguns casos. Além do mais, para esse conjunto de instâncias, o método MILSRVND demandou um maior tempo de execução quando comparado ao modelo matemático (em ambas as abordagens). Ao aplicar a metodoligia mista, conseguiu-se provar a otimalidade de mais algumas instâncias, bem como reudizir o tempo despendido em cada prova. É importante ressaltar que a metodologia mista não foi capaz de melhorar nenhuma das soluções iniciais propostas pelo método MILSRVND. Portanto, afirma-se que o método MILSRVND foi eficaz ao auxiliar o modelo matemático no processo de busca de uma solução ótima. De acordo com os *gaps* entre o método MILSRVND e o modelo matemático para as instâncias com densidades iguais a 50% e 95% o método MILSRVND apresenta um melhor desempenho, sendo que em alguns desses casos a metodologia mista foi capaz de provar a otimalidade das soluções propostas pelo respectivo método.

7 Conclusões e Trabalhos Futuros

Este trabalho abordou o Problema de Edição Não Automática de *Clusters* (PENAC). Dada a sua dificuldade de resolução na otimalidade, em virtude de o mesmo pertencer à classe NP-difícil, foi utilizado um algoritmo heurístico híbrido, denominado MILSRVND. Este algoritmo consiste em duas etapas, onde na primeira uma solução inicial é gerada a partir de uma heurística construtiva. Na segunda etapa um refinamento é feito por uma heurística inspirada na metaheurística *Iterated Local Search*, tendo como busca local o método VND com duas estruturas de vizinhança diferentes.

Para validar o algoritmo heurístico, foi utilizado um conjunto de instâncias gerado de maneira aleatória, contendo ao todo 115 problemas-teste.

De acordo com os resultados empíricos obtidos, verifica-se que o MILSRVND é um algoritmo robusto, com baixa variabilidade média, e com soluções aderentes ao limite inferior.

O modelo de programação matemática proposto foi implementado em 4 versões. Embora a versão simples do modelo, contendo apenas a modelagem necessária para garantir a viabilidade de uma solução, tenha obtido os valores ótimos para as instâncias em questão, o tempo despendido para encontrá-las foi elevado se comparado às demais modelagens.

A metodologia mista (combinação do método `MILSRVND` com o modelo matemático) se mostrou mais eficiente ao encontrar um maior número de soluções ótimas do que o modelo matemático simples (sem o auxílio do método `MILSRVND`).

Como trabalho futuro, pretende-se explorar melhor as vizinhanças, tentando melhorar a qualidade das soluções finais geradas pelo método `MILSRVND`. Também, como sugestão para trabalhos futuros, podemos incluir o desenvolvimento de novas heurísticas utilizando conceitos de outras metaheurísticas, tais como os Algoritmos Evolutivos e o *Iterated Local Search* (ILS) que tem tido bom desempenho em problemas similares ao PENAC. Talvez, o

principal trabalho futuro seja investir no desenvolvimento de uma modelagem matemática mais eficiente que, com o auxílio de uma boa solução inicial, consiga provar a otimalidade de todas as instâncias propostas neste trabalho.

Referências

[1] GUPTA, A. S.; PALIT, A. On clique generation using boolean equations. In: *Proceedings of the IEEE Press Series on Computational Intelligence*. 345 East 47 Street, New York. NY 10017.: The lnstitute of Electrical and Electronics Engineers, Inc., 1979. v. 67, n. 1, p. 178–180.

[2] BEN-DOR, A.; SHAMIR, R.; YAKHINI, Z. Clustering gene expression patterns. *Journal of Computational Biology*, Mary Ann Liebert, Inc., v. 6, n. 3/4, p. 281–297, 1999.

[3] DEHNE, F. et al. The cluster editing problem: Implementations and experiments. *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, v. 4169, p. 13–24, 2006.

[4] RAHMANN, S. et al. Exact and heuristic algorithms for weighted cluster editing. In: MARKSTEIN, P.; XU, Y. (Ed.). *Computational systems bioinformatics: CSB 2007 Conference Proceedings*. 57 Shelton Street, Covent Garden, London WC2H 9HE: Imperial College Press, 2007. v. 6, p. 391–400.

[5] SHARAN, R.; MARON-KATZ, A.; SHAMIR, R. Click and expander: a system for clustering and visualizing gene expression data. *Bioinformatics*, v. 19, n. 14, p. 1787–1799, 2003.

[6] TATUSOV, R. et al. The cog database: an updated version includes eukaryotes. *BMC Bioinformatics*, v. 4, n. 1, p. 41, 2003. ISSN 1471-2105. Disponível em: <http://www.biomedcentral.com/1471-2105/4/41>.

[7] BÖCKER, S.; BRIESEMEISTER, S.; KLAU, G. Exact algorithms for cluster editing: Evaluation and experiments. *Algorithmica*, Springer New York, p. 1–19, 2009. ISSN 0178-4617. 10.1007/s00453-009-9339-7. Disponível em: <http://dx.doi.org/10.1007/s00453-009-9339-7>.

[8] PROTTI, F.; SILVA, M. Dantas da; SZWARCFITER, J. L. Applying modular decomposition to parameterized cluster editing problems. *Theor. Comp. Sys.*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 44, n. 1, p. 91–104, dez. 2008. ISSN 1432-4350. Disponível em: <http://dx.doi.org/10.1007/s00224-007-9032-7>.

[9] HARTUV, E. et al. An algorithm for clustering cdna fingerprints. *Genomics*, v. 66, n. 3, p. 249–256, 2000.

[10] JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: A review. *ACM Computing Surveys*, v. 31, n. 3, p. 264–323, 1999.

[11] MILOSAVLJEVIC, A. et al. Clone clustering by hybridization. *Genomics*, v. 27, n. 1, p. 83–89, 1995.

[12] WITTKOP, T. et al. Large scale clustering of protein sequences with FORCE – a layout based heuristic for weighted cluster editing. *BMC Bioinformatics*, v. 8, n. 1, 2007. Disponível em: <http://dx.doi.org/10.1186/1471-2105-8-396>.

[13] XU, R.; WUNSCH, D. *Clustering (IEEE Press Series on Computational Intelligence)*. illustrated edition. [S.l.]: Wiley-IEEE Press, 2008. Hardcover. ISBN 0470276800.

[14] SHAMIR, R.; SHARAN, R.; TSUR, D. Cluster graph modication problems. *Discrete Applied Mathematics*, v. 144, p. 173–182, 2004.

[15] AUSIELLO, G. et al. Complexity and approximation. Springer-Verlag, 1999.

[16] VAZIRANI, V. Approximation algorithms. Springer-Verlag, 2001.

[17] CHARIKAR, M.; GURUSWAMI, V.; WIRTH, A. Clustering with qualitative information. *Journal of Computer and System Sciences*, v. 71, p. 360–383, 2005.

[18] J., M. J. C. A 2k kernel for the cluster editing problem. *Journal of Computer and System Sciences*, 2011.

[19] GRAMM, J. et al. Graph-modeled data clustering: Exact algorithms for clique generation. *Theory of Computing Systems*, Springer New York, v. 38, p. 373–392, 2005. ISSN 1432-4350. 10.1007/s00224-004-1178-y. Disponível em: <http://dx.doi.org/10.1007/s00224-004-1178-y>.

[20] GRöTSCHEL, M.; WAKABAYASHI, Y. A cutting plane algorithm for a clustering problem. *Math. Program.*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 45, n. 1, p. 59–96, 1989. ISSN 0025-5610.

[21] BANSAL, N.; BLUM, A.; CHAWLA, S. *Correlation clustering*. Springer, jul. 2004. 89–113 p. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.3857>.

[22] BANSAL, N.; BLUM, A.; CHAWLA, S. *Correlation clustering*. 2002. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.3857>.

[23] AILON, N.; CHARIKAR, M.; NEWMAN, A. Aggregating inconsistent information: ranking and clustering. In: *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*. New York, NY, USA: ACM Press, 2005. p. 684–693. ISBN 1581139608. Disponível em: <http://dx.doi.org/10.1145/1060590.1060692>.

[24] GIOTIS, I.; GURUSWAMI, V. *Correlation Clustering with a Fixed Number of Clusters*. abr. 2005. Disponível em: <http://arxiv.org/abs/cs/0504023>.

[25] CHARIKAR, M.; GURUSWAMI, V.; WIRTH, A. Clustering with qualitative information. In: *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 2003. (FOCS '03), p. 524+. ISBN 0-7695-2040-5. Disponível em: <http://portal.acm.org/citation.cfm?id=946306>.

[26] DEMAINE, E. D.; IMMORLICA, N. Correlation clustering with partial information. In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. [s.n.], 2003. p. 71–80. Disponível em: <http://www.springerlink.com/content/mpyhale5nbwtbbjh>.

[27] EMANUEL, D.; FIAT., A. Correlation clustering - minimizing disagreements on arbitrary weighted graphs. In: *Proceedings of the 11th Annual European Symposium*. [S.l.: s.n.], 2003. v. 2832, p. 208–220.

[28] RIBEIRO, C. Metaheuristics and applications. 1996.

[29] HANSEN, P.; MLADENOVIĆ, N. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, v. 130, n. 3, p. 449–467, May 2001.

[30] LOURENÇO, H. R.; MARTIN, O.; STÜTZLE, T. Iterated local search. In: GLOVER, F.; KOCHENBERGER, G. (Ed.). *Handbook of Metaheuristics*. [S.l.]: Kluwer Academic Publishers, Norwell, MA, 2003, (International Series in Operations Research & Management Science, v. 57). p. 321–353.

[31] LAND, A. H.; DOIG, A. G. An automatic method of solving discrete programming problems. *Econometrica*, v. 28, n. 3, p. 497–520, 1960. Disponível em: <http://jmvidal.cse.sc.edu/library/land60a.pdf>.

[32] GRIBKOVSKAIA, I.; LAPORTE, G.; SHYSHOU, A. The single vehicle routing problem with deliveries and selective pickups. *Computers and Operations Research*, Elsevier Science Ltd., Oxford, UK, UK, v. 35, p. 2908–2924, Setembro 2008. ISSN 0305-0548. Disponível em: <http://portal.acm.org/citation.cfm?id=1343112.1343206>.