UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

PROBLEMA DE RECOBRIMENTO DE CONJUNTOS COM AGRUPAMENTO: MODELOS E ALGORITMOS

DISSERTAÇÃO DE MESTRADO

Artur Ferreira Brum

Santa Maria, RS, Brasil

2015

PROBLEMA DE RECOBRIMENTO DE CONJUNTOS COM AGRUPAMENTO: MODELOS E ALGORITMOS

Artur Ferreira Brum

Dissertação apresentada ao Curso de Mestrado do Programa de Pós-Graduação em Informática (PPGI), Área de Concentração em Computação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de
Mestre em Ciência da Computação

Orientador: Prof. Dr. Felipe Martins Müller

Santa Maria, RS, Brasil

2015

Brum, Artur Ferreira

Problema de recobrimento de conjuntos com agrupamento: modelos e algoritmos / por Artur Ferreira Brum. – 2015.
65 f.: il.; 30 cm.

Orientador: Felipe Martins Müller
Dissertação (Mestrado) - Universidade Federal de Santa Maria, Centro de Tecnologia, Mestrado em Ciência da Computação, RS, 2015.

1. Otimização combinatória. 2. Modelo matemático. 3. Metaheurística. I. Müller, Felipe Martins. II. Título.

E-mail: arturfbrum@gmail.com

Universidade Federal de Santa Maria
Centro de Tecnologia
Programa de Pós-Graduação em Informática

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

PROBLEMA DE RECOBRIMENTO DE CONJUNTOS COM AGRUPAMENTO: MODELOS E ALGORITMOS

elaborada por
Artur Ferreira Brum

como requisito parcial para obtenção do grau de
Mestre em Ciência da Computação

COMISSÃO EXAMINADORA:

Felipe Martins Müller, Dr.
(Presidente/Orientador)

Haroldo Gambini Santos, Dr. (UFOP)

Viviane Cátia Köhler, Drª. (UFSM)

Santa Maria, 14 de Abril de 2015.

Aos meus pais

AGRADECIMENTOS

Agradeço à minha família por todo o apoio.

Aos Profs. Olinto, Felipe e Haroldo pelo acompanhamento e disponibilidade.

Aos colegas e amigos Aécio, Alberto e Renan, que contribuíram direta ou indiretamente para a conclusão desta dissertação.

RESUMO

Dissertação de Mestrado
Programa de Pós-Graduação em Informática
Universidade Federal de Santa Maria

PROBLEMA DE RECOBRIMENTO DE CONJUNTOS COM AGRUPAMENTO: MODELOS E ALGORITMOS

AUTOR: ARTUR FERREIRA BRUM
ORIENTADOR: FELIPE MARTINS MÜLLER
Local da Defesa e Data: Santa Maria, 14 de Abril de 2015.

Este trabalho aborda o problema de recobrimento de conjuntos com agrupamento, definido como uma variante do problema de recobrimento de conjuntos clássico, na qual os conjuntos estão dispostos em K agrupamentos disjuntos. São apresentadas duas aplicações práticas para o problema encontradas na literatura. A primeira delas tem origem na engenharia elétrica, especificamente no problema abordado por Fritzen et al (2012). Uma característica intrínseca ao problema, levou à generalização da atribuição dos conjuntos aos agrupamentos, de forma que a intersecção seja permitida. A segunda aplicação é oriunda da indústria de mineração e devida a Bilal et al. (2014), que propõem um conjunto de instâncias e uma meta-heurística baseada em *iterated tabu search* (ITS). Em se tratando de métodos de resolução, a primeira contribuição apresentada neste trabalho consiste em uma formulação matemática com geração de colunas para obtenção de limitantes superiores para as instâncias. Outras duas contribuições se referem à meta-heurística ITS, sendo que a primeira consiste em uma abordagem paralela para o componente de busca tabu, enquanto a segunda reside na adição de uma vizinhança resolvida com programação inteira e que faz uso de *hard variable fixing*. A ideia subjacente ao uso da vizinhança é ocupar o tempo economizado na paralelização com o aperfeiçoamento de soluções promissoras obtidas pela busca tabu. Os resultados computacionais obtidos mostram melhora na qualidade da solução para a ampla maioria das instâncias. Dada ainda a magnitude dos custos envolvidos no problema, é possível concluir que o aprimoramento da meta-heurística ITS com as sugestões deste trabalho pode ser de significativo benefício econômico.

Palavras-chave: Otimização combinatória. Modelo matemático. Meta-heurística.

ABSTRACT

Master's Dissertation
Post-Graduate Program in Informatics
Federal University of Santa Maria

CLUSTERED SET COVERING PROBLEM: MODELS AND ALGORITHMS

AUTHOR: ARTUR FERREIRA BRUM
ADVISOR: FELIPE MARTINS MÜLLER
Defense Place and Date: Santa Maria, April 14th, 2015.

This work addresses the clustered set covering problem, a variant of the classic set covering problem in which sets are arranged in K disjoint clusters. Two practical applications found in the literature are presented. The first originates from electrical engineering, specifically from the problem addressed by Fritzen et al. (2012). An intrinsic characteristic of this problem led to a generalization in set allocation, so that the intersection between clusters is allowed. The second application comes from the mining industry and is owed to Bilal et al. (2014). They proposed a set of instances and a metaheuristic based on iterated tabu search (ITS). Regarding solution methods, the first contribution of this work is a mathematical formulation with column generation, used in order to obtain upper bounds for the instances. Two other contributions refer to the ITS metaheuristic, with the former consisting in a parallel approach for the tabu search component, while the latter resides in the addition of a neighborhood solved with integer programming and based on hard variable fixing. The idea behind the addition of the neighborhood is to occupy the time saved by the parallel approach with the improvement of promising solutions obtained by tabu search. The computational results show improvements in solution quality for the vast majority of instances. Moreover, given the magnitude of the costs involved in the problem, it is possible to conclude that the improvement of ITS metaheuristic with the contributions presented in this work can provide significant economic benefit.

Keywords: Combinatorial optimization, Mathematical model, Metaheuristic.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE APÊNDICES

LISTA DE ABREVIATURAS E SIGLAS

API	*Application programming interface*
DRP	Desvio relativo percentual
GRASP	*Greedy randomized adaptive search procedure*
HVF	*Hard variable fixing*
ITS	*Iterated tabu search*
MIP	*Mixed integer programming*
PGRCA	Problema generalizado de recobrimento de conjuntos com agrupamento
PMR	Problema mestre restrito
PRCA	Problema de recobrimento de conjuntos com agrupamento
PRC	Problema de recobrimento de conjuntos

SUMÁRIO

1 INTRODUÇÃO

Problemas de otimização combinatória são problemas para os quais busca-se encontrar a melhor solução dentro de um extenso universo de soluções possíveis, geralmente finito e discreto. Esses problemas costumam ser de fácil descrição, mas de difícil resolução (OSMAN; KELLY, 1996). Dado que o conjunto de soluções é finito, sempre é possível resolvê-los na otimalidade através de uma técnica elementar como a enumeração exaustiva, no entanto existem classes de problemas cuja resolução é inviável em tempo hábil na prática, considerando o atual conhecimento sobre o assunto.

A teoria da complexidade computacional (COOK, 1971) classifica problemas de otimização em P e NP de acordo com a dificuldade inerente. Enquanto a classe P engloba os problemas de decisão para os quais existe um algoritmo de tempo polinomial, a classe NP representa os que podem ser resolvidos em tempo polinomial por algoritmos não determinísticos. Isso significa que é possível avaliar em tempo polinomial a qualidade de uma solução qualquer. Há também a classe de problemas NP-difíceis, na qual são incluídos os problemas para os quais existe redução a partir de qualquer outro problema da classe NP, ou seja, são problemas ao menos tão difíceis quanto os problemas em NP. Acredita-se ser improvável a existência de algoritmos eficientes (i.e., de tempo polinomial) para resolver esses problemas. São exemplos de problemas NP-difíceis o problema do caixeiro viajante (GAREY; JOHNSON, 1979) e o problema de coloração de grafos (KARP, 1972).

Este trabalho aborda o problema de recobrimento de conjunto com agrupamento (PRCA), recentemente definido por Alfandari e Monnot (2014), e sua generalização. O PRCA consiste em uma variante NP-difícil do problema de recobrimento de conjuntos (PRC) clássico, que é amplamente conhecido e abordado em Ciência da Computação. Nessa variante os conjuntos de recobrimentos ficam dispostos em K agrupamentos disjuntos e sob cada um incide um custo fixo a ser pago uma única vez sempre que pelo menos um conjunto nele contido é selecionado.

Neste trabalho é demonstrado como os problemas abordados por Fritzen et al. (2012) e Bilal et al. (2014) podem ser englobados como PRCA. Ainda, especificamente para o primeiro problema, uma característica observada fez necessária a generalização da atribuição dos conjuntos de recobrimento aos agrupamentos, de forma que a intersecção seja permitida.

Por fim, uma meta-heurística proposta no trabalho de Bilal et al. (2014), denominada *iterated tabu search* (ITS), é investigada. É proposto o uso simultâneo de uma abordagem paralela

e uma nova vizinhança a fim de melhorar a qualidade das soluções obtidas pelo método. O estudo se concentra no conjunto de instâncias encontrado na literatura, devido a Bilal et al. (2014).

1.1 Objetivo

1.1.1 Objetivo geral

O objetivo geral desta dissertação é estudar o PRCA, bem como sua generalização, apresentando aplicações práticas e técnicas eficientes para sua resolução.

1.1.2 Objetivos específicos

Como objetivos específicos, este trabalho visa:

- Estudar e apresentar aplicações práticas para o PRCA.
- Estudar técnicas de programação matemática.
- Estudar heurísticas, meta-heurísticas e *matheuristics*.
- Implementar, avaliar e aprimorar a meta-heurística ITS.
- Comparar resultados obtidos aos encontrados na literatura.

1.2 Motivação

O estudo do PRCA é motivado pela sua aplicabilidade em problemas práticos reais. A primeira aplicação apresentada é oriunda da engenharia elétrica, especificamente do problema de processamento de alarmes de proteção em sistemas elétricos de potência descrito por Fritzen et al. (2012). O problema consiste, resumidamente, em selecionar padrões de mensagens (comumente chamadas alarmes) contidos em uma base de conhecimento, de forma que a combinação dos padrões selecionados explique logicamente os alarmes gerados durante a ocorrência de distúrbios no sistema elétrico. O estudo deste problema tem relevância, pois como vivemos em época de crescimento de demanda por energia no Brasil, os sistemas elétricos de potência vêm trabalhando com maior proximidade em relação aos seus limites e requerem investimentos em expansão e modernização. Dessa forma, o refinamento dos métodos e ferramentas computacionais atuais auxilia os operadores dos sistemas elétricos com a automatização do processa-

mento de grandes volumes de dados gerados em tempo real, tornando mais eficientes processos altamente dependentes do desempenho humano. Dada ainda a escassez de profissionais especializados, ferramentas computacionais eficientes beneficiam não só as concessionárias de energia, que precisam atender metas, mas também os seus clientes, que têm reduzidos seus períodos de indisponibilidade de energia elétrica.

A segunda aplicação apresentada para o PRCA está na indústria de mineração. O problema de otimização combinatória abordado por Bilal et al. (2014) envolve a avaliação de potenciais pontos de extração de minérios e inclui uma etapa em que o solo é analisado através de perfurações de caráter exploratório, chamadas sondagens. As sondagens possuem um alto custo envolvido, devido à movimentação e operação dos equipamentos de perfuração. O planejamento cuidadoso de quantidade, extensão e localização das sondagens é, portanto, fundamental para a minimização dos custos. Nesse sentido, o aprimoramento das abordagens para solução do problema pode ser de grande benefício, pois mesmo reduções de pequenos percentuais no custo trazem substancial economia, dada a magnitude dos valores envolvidos.

Em suma, para ambas as aplicações apresentadas, há grande interesse prático envolvido, pois o aperfeiçoamento dos métodos de resolução se traduz em significativos ganhos do ponto de vista econômico, justificando assim o tempo e esforço empregados no estudo do problema subjacente.

1.3 Estrutura do trabalho

Esta dissertação está estruturada em cinco capítulos, incluindo o de introdução. Os capítulos seguintes estão organizados conforme segue. No Capítulo 2 o PRCA e sua generalização são descritos formalmente. São apresentadas ainda duas aplicações práticas distintas para o problema. O Capítulo 3 aborda métodos de resolução. Uma abordagem com geração de colunas é descrita. Na sequência, a meta-heurística ITS de Bilal et al. (2014) é exposta e duas contribuições são propostas. Os resultados obtidos são apresentados no Capítulo 4. Por fim, no Capítulo 5 são expostas as conclusões do trabalho.

2 PROBLEMA DE RECOBRIMENTO DE CONJUNTOS COM AGRUPAMENTO

O problema de recobrimento de conjuntos com agrupamento é definido por Alfandari e Monnot (2014) como uma variante do problema de recobrimento de conjuntos, na qual os conjuntos de recobrimento são particionados em K agrupamentos disjuntos. Para cada agrupamento há um custo fixo associado, que deve ser pago quando ao menos um conjunto de recobrimento nele contido é selecionado.

O PRCA é definido formalmente por Alfandari e Monnot (2014) da seguinte forma. Seja $C = \{1, \ldots, n\}$ um conjunto de elementos e $\mathcal{S} = \{S_1, \ldots, S_m\}$ uma coleção de subconjuntos de C. Cada subconjunto $S_j \in \mathcal{S}$ tem associado um custo positivo $c_j = c(S_j)$. Assume-se ainda que o conjunto de índices $J = \{1, \ldots, m\}$ é particionado em K subconjuntos disjuntos, de forma que $\mathcal{J} = \{J_k \colon k = 1, \ldots, K\}$, ou seja, $\bigcup_{k=1}^{K} J_k = J$ e $J_k \cap J_{k'} = \emptyset$ para todo $k \neq k'$. Cada agrupamento é definido por $\mathcal{F}_k = \{S_j \in \mathcal{S} \colon j \in J_k\}$ e um custo f_k é pago quando pelo menos um subconjunto contido em $\mathcal{F}_k$ é selecionado. O PRCA consiste em cobrir todos os elementos de C através da seleção de uma coleção de subconjuntos $\mathcal{S}' \subset \mathcal{S}$, minimizando $c(\mathcal{S}')$ e o custo dos agrupamentos utilizados, ou seja, $\overline{c}(\mathcal{S}') = c(\mathcal{S}') + \sum_{k \colon \mathcal{S}' \cap \mathcal{F}_k \neq \emptyset} f_k$. O seguinte modelo matemático é formalizado.

Parâmetros:

$$a_{ij} = \begin{cases} 1 & \text{se } i \in S_j \\ 0 & \text{caso contrário} \end{cases}$$

$$f_k = \text{custo do agrupamento } k.$$

$$c_j = \text{custo do conjunto } j.$$

Variáveis:

$$x_j = \begin{cases} 1 & \text{se o conjunto } S_j \text{ é selecionado} \\ 0 & \text{caso contrário} \end{cases}$$

$$y_k = \begin{cases} 1 & \text{se o agrupamento } k \text{ é selecionado} \\ 0 & \text{caso contrário} \end{cases}$$

Modelo matemático:

$$\text{Minimizar } \sum_{k=1}^{K} f_k y_k + \sum_{j=1}^{m} c_j x_j \tag{2.1}$$

Sujeito a:

$$\sum_{j=1}^{m} a_{ij} x_j \geq 1 \qquad i = 1, \ldots, n \tag{2.2}$$

$$y_k \geq x_j \qquad k = 1, \ldots, K,\ j \in J_k \tag{2.3}$$

$$x_j \in \{0, 1\} \qquad j = 1, \ldots, m \tag{2.4}$$

$$y_k \in \{0, 1\} \qquad k = 1, \ldots, K \tag{2.5}$$

A função objetivo (2.1) considera a minimização dos custos pagos pelos conjuntos e agrupamentos selecionados. O conjunto de restrições (2.2) estabelece que cada elemento deve ser coberto por pelo menos um conjunto. Nas restrições em (2.3) é definido que se um agrupamento é selecionado, os conjuntos de cobertura contidos são liberados para seleção. Por fim, as restrições (2.4) e (2.5) definem o domínio das variáveis.

De acordo com os autores, muitos problemas de transporte podem ser formulados como PRCA, como o problema de determinação de viagens abordado por Desaulniers (1997) ou o problema de roteamento de veículos com frota heterogênea de Barnhart et al. (2003).

Durante o desenvolvimento deste trabalho, foram encontradas na literatura duas novas aplicações para o PRCA. A primeira delas, oriunda da engenharia elétrica, é abordada por Fritzen et al. (2012), enquanto a segunda, com origem na indústria de mineração, é encontrada no trabalho de Bilal et al. (2014).

No trabalho de Fritzen et al. (2012), no entanto, observou-se uma característica intrínseca ao problema tratado que tornou necessária a definição de uma generalização para o PRCA. Nele é comum que os conjuntos de recobrimento estejam em mais de um agrupamento simultaneamente, ou seja, os agrupamentos não são disjuntos. A generalização e as aplicações do PRCA são detalhadas nas seções seguintes.

2.1 Generalização

A generalização do PRCA é aqui denominada problema generalizado de recobrimento de conjuntos com agrupamento (PGRCA). O problema é definido formalmente da forma que segue. Seja $C = \{1, \ldots, n\}$ um conjunto de elementos a serem cobertos, $\mathcal{S} = \{S_1, \ldots, S_m\}$ uma coleção contendo subconjuntos de C, e $c(S_j)$ um custo positivo para cada $S_j \in \mathcal{S}$. Ainda, considere um conjunto de índices $J = \{1, \ldots, m\}$ e uma coleção $\mathcal{J} = \{J_1, \ldots, J_k\}$ contendo K subconjuntos de J de forma que $\bigcup_{k=1}^{k} J_k = J$. Cada agrupamento é definido por $\mathcal{F}_k =$

$\{S_j \in \mathcal{S}\colon j \in J_k,\ k = 1, \ldots, K\}$ e tem um custo fixo $f_k \geq 0$ a ser pago quando ao menos um conjunto contido é selecionado. É definida ainda uma coleção $\mathcal{Q} = \{Q_1, \ldots, Q_m\}$, na qual cada subconjunto Q_j é composto pelos índices dos agrupamentos que contêm S_j. O objetivo do PGRCA consiste em determinar uma coleção $\mathcal{S}' \subset \mathcal{S}$ que cubra todos os elementos em C enquanto é minimizado $c(\mathcal{S}')$ e os custos fixos relativos aos agrupamentos.

O modelo matemático para o PGRCA é formalizado conforme segue.

Parâmetros:

$$a_{ij} = \begin{cases} 1 & \text{se } i \in S_j \\ 0 & \text{caso contrário} \end{cases}$$

$$f_k = \text{custo do agrupamento } k.$$

$$c_j = \text{custo do conjunto } j.$$

Variáveis:

$$x_j = \begin{cases} 1 & \text{se o conjunto } S_j \text{ é selecionado} \\ 0 & \text{caso contrário} \end{cases}$$

$$y_k = \begin{cases} 1 & \text{se o agrupamento } k \text{ é selecionado} \\ 0 & \text{caso contrário} \end{cases}$$

Modelo matemático:

$$\text{Minimizar } \sum_{k=1}^{K} f_k y_k + \sum_{j=1}^{m} c_j x_j \tag{2.6}$$

Sujeito a:

$$(2.2), (2.4), (2.5)$$

$$\sum_{k \in Q_j} y_k \geq x_j \qquad\qquad j = 1, \ldots, m \tag{2.7}$$

A função objetivo e três conjuntos de restrições são os mesmo do modelo do PRCA. Nas restrições em (2.7) é definido que para um conjunto ser selecionado, ao menos um agrupamento que o contenha deve ser selecionado também.

Em comparação ao PRCA, a diferença reside na forma como os agrupamentos são organizados. No PRCA os agrupamentos são disjunto e formam, portanto, uma partição de $\mathcal{S}$,

enquanto no PGRCA a intersecção pode ocorrer. Conclui-se que resolver o PRCA é equivalente a resolver um PGRCA em que $J_k \cap J_{k'} = \emptyset$, para todo $k \neq k'$. Assim sendo, pode-se afirmar que o PGRCA também é NP-difícil.

2.2 Aplicação em engenharia elétrica

A fim de evitar danos materiais ou humanos, sistemas elétricos de potência têm esquemas de proteção projetados para detectar desordens e isolar áreas defeituosas enquanto evitam o desligamento incorreto de equipamentos que operam sem falta. Quando um dispositivo de proteção age, uma mensagem, comumente denominada alarme, é enviada ao Centro de Controle. O operador, com base em seu conhecimento e experiência, toma medidas para evitar propagação de faltas e apagões.

Quando um ou mais dispositivos de proteção são ativados, uma grande número de alarmes é reportado, invariavelmente dificultando que o operador chegue a um diagnóstico correto e tome providências rapidamente. No processo, também é possível que informações errôneas sejam geradas, geralmente chamadas alarmes falhos e falsos. Um alarme é considerado falho quando sua ativação era esperada mas não ocorreu, enquanto alarmes falsos são os que dispararam sem razão aparente.

Para auxiliar o processo de diagnóstico de falta, Fritzen et al. (2012) propõe uma base de dados de diagnósticos conhecidos para determinados grupos de alarmes. Esse diagnósticos são representados por padrões de alarmes denominados eventos. Quando um evento não explica todos os alarmes recebidos, os faltantes são considerados falhos. Se a base de dados armazena todos os eventos relativos ao funcionamento dos dispositivos de proteção, então, para um cenário sem alarmes falhos ou falsos, o problema de otimização é restrito a um PRC, ou seja, encontrar o menor número de eventos contendo todos os alarmes reportados. Visto que é inviável listar todos os eventos para cada situação factível, um modelo matemático é utilizado para identificar o menor conjunto de eventos que explica a falta no sistema elétrico. Quando múltiplas faltas ocorrem simultaneamente, é amplamente aceito o uso da Teoria da Cobertura Parcimoniosa (PENG; REGGIA, 1986), que afirma que a explicação factível mais simples deve ser escolhida. Dado que encontrar essa explicação ou diagnóstico não é uma tarefa trivial, ferramentas computacionais fornecem um valioso auxílio ao operador.

Fritzen et al. (2012) propõe o seguinte modelo matemático.

Conjuntos:

$$I = \text{conjunto de índices de eventos}$$

$$J = \text{conjunto de índices de alarmes}$$

$$E_j = \text{conjunto de índices de eventos associados ao alarme } j$$

$$A_i = \text{conjunto de índices de alarmes associados ao evento } i$$

Parâmetros:

$$a_j = \begin{cases} 1 & \text{se o alarme } j \text{ é disparado} \\ 0 & \text{caso contrário} \end{cases}$$

$$W_1 = \text{custo para considerar um alarme falso}$$

$$W_2 = \text{custo para considerar um alarme falho}$$

$$W_3 = \text{custo para selecionar um evento}$$

Variáveis:

$$e_i = \begin{cases} 1 & \text{se o evento } i \text{ é selecionado} \\ 0 & \text{caso contrário} \end{cases}$$

$$s_j = \begin{cases} 1 & \text{se o alarme } j \text{ é falso} \\ 0 & \text{caso contrário} \end{cases}$$

$$f_j = \begin{cases} 1 & \text{se o alarme } j \text{ é falho} \\ 0 & \text{caso contrário} \end{cases}$$

Modelo matemático:

$$\text{Minimizar } W_1 \sum_{j \in J} s_j + W_2 \sum_{j \in J} f_j + W_3 \sum_{i \in I} e_i \tag{2.8}$$

Sujeito a:

$$s_j + \sum_{i \in E_j} e_i \geq a_j \qquad \forall j \in J \tag{2.9}$$

$$e_i \leq a_j + f_j \qquad \forall i \in I, \forall j \in A_i \tag{2.10}$$

$$e_i \in \{0, 1\} \qquad \forall i \in I \tag{2.11}$$

$$s_j, f_j \in \{0, 1\} \qquad \forall j \in J \tag{2.12}$$

A função objetivo (2.8) considera a minimização dos custos pagos pela seleção de eventos, alarmes falsos e falhos. São utilizados três custos distintos para determinar a importância de cada componente da função objetivo, de forma que $W1 > W2 > W3$. O conjunto de restrições (2.9) estabelece que cada alarme disparado deve ser recoberto por pelo menos um evento, ou então deve ser considerado falso. Nas restrições em (2.10) é definido que, na ausência de um alarme esperado, ele é considerado falho. Essas restrições são trivialmente satisfeitas para alarmes recebidos, onde $a_j = 1$. Por fim, as restrições em (2.11) e (2.12) definem os domínios das variáveis.

O modelo consiste em uma formulação compacta para o PGRCA, onde os agrupamentos são construídos implicitamente de acordo com os alarmes falhos. Para cada alarme considerado falho um custo fixo é pago, permitindo que seja selecionado um novo grupo de eventos que consideram o dado alarme falho. Os alarmes falsos podem ser interpretados como conjuntos de recobrimento que contém um único elemento. Seguindo a metodologia de Fritzen et al. (2012), estes conjuntos devem ter um custo superior aos demais, de acordo com os valores definidos para W_1 e W_3. Posto isso, torna-se intuitiva a modelagem do problema como PGRCA, sendo que eventos correspondem a conjuntos de recobrimento, alarmes são equivalentes aos elementos e alarmes falhos definem os agrupamentos.

O problema de processamento de alarmes abordado por Fritzen et al. (2012) possui ainda uma característica interessante que motivou a adoção da formulação compacta. Quando um conjunto F de alarmes falhos é analisado, cada subconjunto do conjunto potência de F pode gerar um agrupamento. Como o número de alarmes que podem ser considerados falhos é geralmente grande, a explosão combinatorial torna-se um grande obstáculo para a enumeração dos agrupamentos. Além dessa complicação, é comum que eventos compartilhem muitos alarmes, tornando a intersecção entre agrupamentos muito frequente, aumentando significativamente o número de conjuntos contidos em cada agrupamento. Dessa forma, a formulação compacta de Fritzen et al. (2012), onde os agrupamentos são considerados implicitamente, torna-se essencial para problemas práticos. Note que a intersecção entre agrupamentos acarretou a definição do PGRCA neste trabalho.

Especificamente para esse problema, o Algoritmo 1 pode ser utilizado para listar todos os agrupamentos. Note que o símbolo $\mathcal{P}$ na linha 1 denota conjunto potência. Para todo subconjunto p em P e para cada par de alarmes falhos (f_h, f_k) em $p \times p$, se existe um evento e_i que contém f_h e f_k, então o par é armazenado em A e e_i é armazenado em C. Se um grafo G com

p como conjunto de vértices e A como conjunto de arestas é conexo, então um agrupamento contendo os subconjuntos em C é criado e adicionado a $\mathcal{C}$. O conjunto $\mathcal{C}$ é a saída do algoritmo e contém todos os agrupamentos gerados pelos alarmes falhos de entrada.

Algoritmo 1 Algoritmo para explicitar agrupamentos para a formulação compacta do PGRCA.

Entrada: Conjunto F de alarmes falhos. Conjunto I de índices para eventos.
Saída: Conjunto $\mathcal{C}$ de agrupamentos.

```
1:  P ← 𝒫(F)
2:  𝒞 ← ∅
3:  Para todo p ∈ P faça
4:      C ← ∅
5:      A ← ∅
6:      Para todo (f_h, f_k) ∈ p × p faça
7:          Se ∃e_i, i ∈ I : f_h e f_k habilitam e_i então
8:              A ← A ∪ {(f_h, f_k)}
9:              C ← C ∪ {e_i}
10:         Fim Se
11:     Fim Para
12:     Se G = (p, A) é um grafo conexo então
13:         𝒞 ← 𝒞 ∪ {C}
14:     Fim Se
15: Fim Para
```

Para ilustrar o algoritmo, a Figura 2.1 apresenta uma instância exemplo com três eventos (e_1, e_2 e e_3) e três alarmes falhos (f_1, f_2 e f_3). Para esse exemplo tem-se:

$$F = \{f_1, f_2, f_3\}$$

$$P = [\{\}, \{f_1\}, \{f_2\}, \{f_3\}, \{f_1, f_2\}, \{f_1, f_3\}, \{f_2, f_3\}, \{f_1, f_2, f_3\}]$$

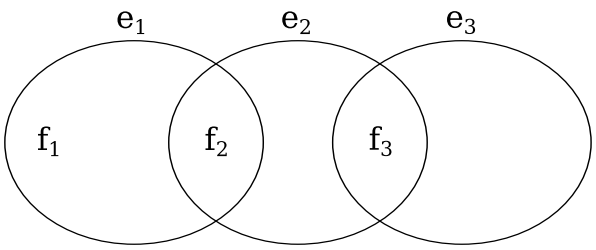

Figura 2.1 – Representação gráfica para uma instância com 3 eventos e 3 alarmes falhos

Adiante, escolhemos arbitrariamente $p = \{f_1, f_2, f_3\}$ para prosseguir no exemplo. Nesse caso:

- O par (f_1, f_2) habilita e_1

- O par (f_2, f_3) habilita e_2 e e_3
- O par (f_3, f_3) habilita e_3

Logo, $A = \{(f_1, f_2), (f_2, f_3), (f_3, f_3)\}$ e $C = \{e_1, e_2, e_3\}$.

O grafo G resultante tem, portanto, três arestas. Ele é ilustrado na Figura 2.2. Como G é conexo, o agrupamento $\{e_1, e_2, e_3\}$ é identificado.

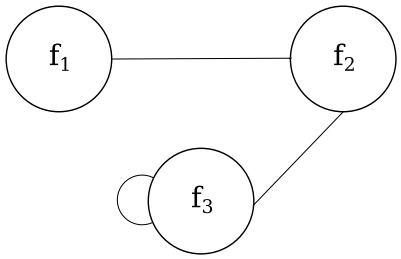

Figura 2.2 – Grafo conexo gerado para o subconjunto $p = \{f_1, f_2, f_3\}$

Ao fim da execução do algoritmo da Figura 1, é possível identificar quatro agrupamentos:

$$y_1 = \{e_1\}$$
$$y_2 = \{e_2, e_3\}$$
$$y_3 = \{e_3\}$$
$$y_4 = \{e_1, e_2, e_3\}$$

Por fim, sendo $c(f_j)$ o custo para considerar o alarme j falho, os agrupamento têm os seguintes custos.

$$c(y_1) = c(f_1) + c(f_2)$$
$$c(y_2) = c(f_2) + c(f_3)$$
$$c(y_3) = c(f_3)$$
$$c(y_4) = c(f_1) + c(f_2) + c(f_3)$$

2.3 Aplicação na indústria de mineração

Na indústria de mineração, a primeira etapa para o estabelecimento de minas de extração de minérios geralmente consiste em um estudo geológico com a finalidade de avaliar a

viabilidade da região. Na etapa seguinte, são feitas longas perfurações de caráter exploratório, denominadas sondagens. Através das sondagens são obtidas amostras do solo, que servem para validar ou ajustar as estimativas iniciais da localização dos minérios. Como a perfuração do solo envolve alto custo, é importante otimizar a quantidade, localização, orientação e extensão das sondagens. Há ainda um custo associado à movimentação e posicionamento dos equipamentos que deve ser levado em consideração.

Os resultados do estudo costumam ser apresentados em um diagrama de blocos, no qual o solo é dividido em porções de formato cúbico e a cada porção é associada uma estimativa da quantidade de minerais contida. Ao representar no diagrama os locais onde os equipamentos de perfuração podem ser instalados, é possível determinar todas as sondagens factíveis. Um bloco é considerado coberto por uma sondagem se a distância ortogonal do seu centro até a perfuração é menor que um determinado valor. Um exemplo bidimensional do diagrama é apresentado na Figura 2.3.

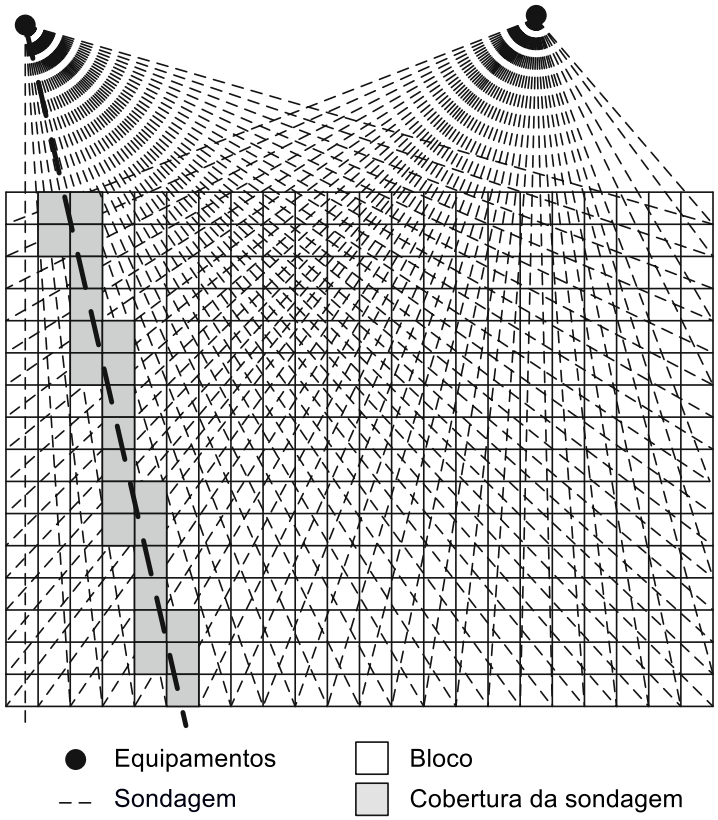

Figura 2.3 – Representação de potenciais sondagens. Figura extraída de Bilal et al. (2014)

A modelagem do problema subjacente como PRCA é bastante intuitiva. Nesse caso, os blocos correspondem aos elementos a serem cobertos, enquanto as sondagens são equivalentes

aos conjuntos de recobrimento, uma vez que cada uma cobre um subconjunto de elementos. O custo da movimentação dos equipamentos é análogo ao custo fixo de um agrupamento, pois, ao ser pago, libera um grupo de conjuntos de recobrimento. A Figura 2.3 ilustra possíveis sondagens a partir de dois posicionamentos distintos para os equipamentos e auxilia na visualização da equivalência entre problemas.

O seguinte modelo matemático é formalizado:

Parâmetros:

$$a_{ij} = \begin{cases} 1 & \text{se o elemento } i \text{ é recoberto pelo conjunto } j \\ 0 & \text{caso contrário} \end{cases}$$

$$b_{kj} = \begin{cases} 1 & \text{se o conjunto } j \text{ está contido no agrupamento } k \\ 0 & \text{caso contrário} \end{cases}$$

$$g_i = \text{ganho associado ao elemento } i.$$

$$c_j = \text{custo para selecionar o conjunto } j.$$

$$f_k = \text{custo fixo do agrupamento } k.$$

Variáveis:

$$y_i = \begin{cases} 1 & \text{se o elemento } i \text{ é recoberto} \\ 0 & \text{caso contrário} \end{cases}$$

$$x_j = \begin{cases} 1 & \text{se o conjunto } j \text{ é selecionado} \\ 0 & \text{caso contrário} \end{cases}$$

$$gr_k = \begin{cases} 1 & \text{se o agrupamento } k \text{ é selecionado} \\ 0 & \text{caso contrário} \end{cases}$$

Modelo matemático:

$$\text{Maximizar } \sum_{i=1}^{m} g_i y_i - \sum_{j=1}^{n} c_j x_j - \sum_{k=1}^{l} f_k gr_k \tag{2.13}$$

Sujeito a:

$$y_i \leq \sum_{j=1}^{n} a_{ij} x_j \qquad i = 1, \ldots, m \tag{2.14}$$

$$b_{kj} x_j \leq gr_k \qquad j = 1, \ldots, n,\ k = 1, \ldots, l \tag{2.15}$$

$$y_i, x_j, gr_k \in \{0, 1\} \tag{2.16}$$

A função objetivo (2.13) considera a maximização do ganho gerado pela cobertura dos elementos, enquanto são minimizados os custos referentes à seleção de conjuntos de recobrimento e agrupamentos. As restrições em (2.14) definem que um elemento é coberto se pelo menos um conjunto que o cubra é selecionado. Em (2.15) fica definido que um agrupamento deve ser selecionado sempre que ao menos um conjunto nele contido for selecionado. Por fim, (2.16) determina o domínio das variáveis.

Para demonstrar a equivalência deste modelo ao do PRCA, aplica-se uma sequência de operações simples ao modelo matemático. Começando pela função objetivo (2.13), tem-se

1. O primeiro passo é a substituição da variável y_i por $(1 - \overline{y_i})$, sendo $\overline{y_i}$ o complemento de y_i, i.e., $\overline{y_i}$ é a negação de y_i. A substituição resulta na seguinte expressão:

$$\text{Maximizar } \sum_{i=1}^{m} g_i(1 - \overline{y_i}) - \sum_{j=1}^{n} c_j x_j - \sum_{k=1}^{l} f_k g r_k$$

2. Aplicando a propriedade distributiva à multiplicação do primeiro termo da expressão obtém-se:

$$\text{Maximizar } \sum_{i=1}^{m} g_i - \sum_{i=1}^{m} g_i \overline{y_i} - \sum_{j=1}^{n} c_j x_j - \sum_{k=1}^{l} f_k g r_k$$

3. O primeiro termo pode ser omitido por ser um somatório de constantes. Restam, portanto, três termos com sinal negativo. Ao multiplicar a expressão por -1 tem-se:

$$\text{Minimizar } \sum_{i=1}^{m} g_i \overline{y_i} + \sum_{j=1}^{n} c_j x_j + \sum_{k=1}^{l} f_k g r_k$$

4. Como $\overline{y_i}$ é o complemento de y_i, pode-se interpretar $\overline{y_i}$ como a desistência de recobrir o elemento i. Dessa forma, haverá um custo g_i associado à decisão de não cobrir o elemento i, o que é equivalente a considerar um conjunto de recobrimento contendo apenas i e com custo g_i. Assim sendo, passa-se a considerar m novos conjuntos, que podem ser representados junto aos do segundo termo. Ainda, deve ser criado um agrupamento artificial, com custo zero, para estes conjuntos.

$$\text{Minimizar } \sum_{j=1}^{n+m} c_j x_j + \sum_{k=1}^{l+1} f_k g r_k$$

O resultado é equivalente à função objetivo em (2.1).

Para o conjunto de restrições em (2.14), a equivalência pode ser verificada com os seguintes passos:

1. Ao substituir y_i por $(1 - \overline{y_i})$ tem-se:

$$1 - \overline{y_i} \leq \sum_{j=1}^{n} a_{ij} x_j$$

2. Por fim, conforme descrito anteriormente, $\overline{y_i}$ pode ser considerado junto aos demais conjuntos de cobertura. Desta forma, podemos representar a restrição por:

$$\sum_{j=1}^{n+m} a_{ij} x_j \geq 1$$

O resultado é equivalente a (2.2).

O conjunto de restrições (2.15) é equivalente ao conjunto de restrições (2.3). Neste caso, note que o parâmetro b_{kj} tem a mesma finalidade de J_k, definido na Seção 2.1, que é indicar quais conjuntos de recobrimento cada agrupamento contém.

3 MÉTODOS DE RESOLUÇÃO

Métodos de otimização combinatórias normalmente são divididos em duas categorias: exatos e heurísticos. A principal característica de interesse em algoritmos exatos é a sua garantia de encontrar e provar a solução ótima para qualquer instância de problemas de otimização. Abordagens exatas, entretanto, costumam ser eficientes apenas para instâncias de pequeno e médio porte, visto que o tempo de execução costuma aumentar exponencialmente conforme a dimensão da instância, restringindo assim a aplicação prática destes métodos. São exemplos de abordagens exatas *branch-and-bound*, programação dinâmica e métodos baseados em programação linear e inteira, como *branch-and-cut*, *branch-and-price* e *branch-and-cut-and-price*.

Diferentemente dos algoritmos exatos, métodos heurísticos buscam apenas fornecer uma boa solução, não necessariamente ótima, em tempo reduzido. Como problemas de otimização reais podem ter grande dimensão, frequentemente, heurísticas ou meta-heurísticas são as únicas opções viáveis. De maneira geral, espera-se que métodos heurísticos:

- encontrem soluções próximas ao ótimo com alta frequência;
- tenham baixa probabilidade de obter soluções distantes do ótimo;
- demandem tempo moderado.

Sob a designação de meta-heurísticas pode-se citar como exemplo: *simulated annealing*, busca tabu, busca em vizinhança variável, algoritmos genéticos e GRASP (OSMAN; KELLY, 1996), (BLUM; ROLI, 2003).

Como o PRCA foi definido recentemente na literatura, foi encontrada apenas uma meta-heurística projetada especificamente para o problema. O método é devido a Bilal et al. (2014) e consiste em uma combinação de *iterated local search* com busca tabu. *Iterated local search* é uma meta-heurística que realiza sucessivas iterações de busca local e que conta com um operador de perturbação que define um novo ponto inicial para a busca a cada iteração. Em trabalhos recentes, *iterated local search* tem sido aplicada a uma variedade de problemas como: problema de roteamento de veículos e variantes (PENNA; SUBRAMANIAN; OCHI, 2013), (WALKER et al., 2012), (MICHALLET et al., 2014), *multi-capacity bin packing problem* (MASSON et al., 2013) e *permutation flow shop scheduling problem* (DONG; HUANG; CHEN, 2009), (MARMION et al., 2011).

A busca tabu é um algoritmo que estende busca local, fazendo uso de estruturas de memória para evitar ciclos e escapar de ótimos locais. A memória consiste em uma lista de movimentos proibidos, chamada lista tabu, que é mantida de tal forma que, sempre que um movimento é executado, seu inverso é adicionado à lista. Um movimento fica proibido enquanto estiver na lista tabu e é liberado após um determinado número de iterações. Glover (1989) apresenta os princípios fundamentais subjacentes ao método.

No algoritmo resultante, que é denominado *iterated tabu search*, é feita a substituição do operador de busca bocal da *iterated local search* por busca babu. Para diversificação da solução são propostas duas funções de perturbação, sendo a que primeira diversifica a busca ao redor do ótimo local atual, enquanto a segunda tem por objetivo forçar grandes saltos no espaço de busca a fim de explorar novas vizinhanças.

Para avaliar o método apresentado, Bilal et al. (2014) adaptaram o algoritmo memético de Beasley e Chu (1996), originalmente proposto para resolução do PRC clássico. Os experimentos computacionais consideraram um conjunto de 30 instâncias, obtidas a partir de dados reais da indústria de mineração.

Ainda, Bilal et al. (2014) usaram o resolvedor CPLEX para resolver na otimalidade, ou próximo disso, 20 instâncias consideradas pequenas ou médias. As soluções obtidas serviram para avaliar a eficiência das meta-heurísticas abordadas. Para as 10 instâncias consideradas grandes, no entanto, os autores reportam que o CPLEX não foi capaz de resolver nem mesmo o nó raiz do problema em um intervalo de tempo aceitável, inviabilizando assim a obtenção de soluções para essas instâncias. Uma das contribuições deste trabalho consiste em um método com geração de colunas, para a obtenção de limitantes superiores para as instâncias grandes.

Outras duas contribuições se referem à meta-heurística ITS. A primeira delas consiste em uma abordagem paralela para o componente de busca tabu, fundamentada no fato de a avaliação do melhor movimento não possuir dependência de dados e ser, portanto, facilmente paralelizável. A segunda contribuição compreende a hibridização da ITS, através da adição de uma vizinhança resolvida com programação inteira.

Nas seções seguintes são apresentadas as três referidas contribuições para o trabalho de Bilal et al. (2014).

3.1 Método com geração de colunas

Geração de colunas é uma técnica usualmente aplicada a problemas com um número muito grande de variáveis. A ideia subjacente à técnica é que, visto que a maior parte das variáveis tem valor zero na solução ótima, apenas um subconjunto das variáveis precisa ser considerado durante a resolução do problema. Nesse subconjunto são adicionadas iterativamente apenas as variáveis que possuem potencial para melhorar a função objetivo, ou seja, aquelas com custo reduzido positivo (para problemas de maximização). Isso é feito através da decomposição do problema em um problema mestre e um subproblema (DANTZIG; WOLFE, 1960). O problema mestre é o problema original restrito a um subconjunto das variáveis, enquanto o subproblema é utilizado para identificar variáveis a serem adicionadas ao problema mestre. O Algoritmo 2 auxilia no detalhamento do processo.

Algoritmo 2 Geração de colunas para um problema linear

Entrada: Problema mestre restrito (PMR), subproblema
Saída: Solução ótima para o problema mestre
1: Crie colunas iniciais
2: **Repita**
3: Resolva o PMR, encontre π
4: Atualize os coeficientes do subproblema de acordo com π
5: Resolva o subproblema
6: Adicione a nova coluna ao PMR
7: **enquanto** há colunas com custo reduzido positivo ▷ Problema de maximização
8: **Retorna** Solução ótima para o problema mestre

Partindo-se do problema mestre restrito (PMR) com um conjunto inicial de colunas, resolve-se o PMR e os valores duais referentes às restrições do problema são armazenados em um vetor π. O vetor é utilizado para atualizar os coeficientes das variáveis da função objetivo do subproblema. Na sequência, o subproblema é resolvido. Se o valor da função objetivo é maior que zero, então há uma coluna que deve ser adicionada ao PMR. Após adicionar a nova coluna, o PMR é resolvido novamente e o vetor π é atualizado conforme os novos valores duais encontrados. Quando não há mais colunas a serem adicionadas, o algoritmo é encerrado e a solução do PMR é a solução ótima do problema mestre.

São exemplos de trabalhos recentes que utilizam geração de colunas: (DAYARIAN et al., 2015), (OOSTRUM et al., 2008) e (TAŞ et al., 2014).

Neste trabalho é apresentada uma nova modelagem para o problema de Bilal et al. (2014), que tem como finalidade a geração de limitantes superiores para as instâncias consideradas nos

testes computacionais. Nessa modelagem são consideradas todas as configurações de seleção de conjuntos de recobrimento possíveis para cada agrupamento. Seja J_k uma coleção contendo todos o conjuntos de recobrimento do agrupamento k, as configurações possíveis de k são dadas pelo conjunto potência de J_k, isto é, $\mathcal{P}(J_k)$. O custo de cada configuração é a soma dos custos dos conjuntos de recobrimento selecionados e do custo fixo referente ao agrupamento.

Como o número de variáveis torna-se exponencial, faz-se uso de geração de colunas para resolver o problema. O modelo matemático para o problema mestre é definido conforme segue.

Conjuntos:

$$I = \text{conjunto de elementos a serem recobertos}$$

$$J = \text{coleção contendo todas as configurações possíveis para cada agrupamento}$$

$$K = \text{coleção de agrupamentos.}$$

Parâmetros:

$$a_{ij} = \begin{cases} 1 & \text{se o elemento } i \text{ é recoberto por } j \\ 0 & \text{caso contrário} \end{cases}$$

$$b_{kj} = \begin{cases} 1 & \text{se o agrupamento de } j \text{ é } k \\ 0 & \text{caso contrário} \end{cases}$$

$$g_i = \text{ganho associado ao elemento } i.$$

$$\widetilde{c}_j = \text{custo da configuração } j.$$

Variáveis:

$$y_i = \begin{cases} 1 & \text{se o elemento } i \text{ é recoberto} \\ 0 & \text{caso contrário} \end{cases}$$

$$\lambda_j = \begin{cases} 1 & \text{se a configuração } j \text{ é selecionada} \\ 0 & \text{caso contrário} \end{cases}$$

Modelo matemático:

$$\text{Maximizar} \sum_{i \in I} g_i y_i - \sum_{j \in J} \widetilde{c}_j \lambda_j \tag{3.1}$$

Sujeito a:

$$\sum_{j \in J} a_{ij} \lambda_j \geq y_i \qquad \forall i \in I \tag{3.2}$$

$$\sum_{j \in J} b_{kj} \lambda_j \leq 1 \qquad \forall k \in K \tag{3.3}$$

$$y_i \in \{0, 1\} \qquad \forall i \in I \tag{3.4}$$

$$\lambda_j \in \{0, 1\} \qquad \forall j \in J \tag{3.5}$$

A função objetivo (3.1) considera a maximização do ganho referente aos elementos cobertos, enquanto é minimizado o custo das configurações selecionadas. O conjunto de restrições em (3.2) estabelece que um elemento é coberto se é selecionada pelo menos uma configuração que o cubra. As restrições (3.3) definem apenas uma configuração pode ser selecionada para cada agrupamento. Por fim, em (3.4) e (3.5) é definido o domínio das variáveis y_i e λ_j.

Considerando o problema mestre restrito, apenas uma coluna é explicitada inicialmente. A coluna corresponde a uma configuração vazia, onde nenhum conjunto de recobrimento é selecionado. Na relaxação, as restrições em (3.4) e (3.5) são substituídas por:

$$0 \leq y_i \leq 1, \qquad \forall i \in I \tag{3.6}$$

$$0 \leq \lambda_j \leq 1, \qquad \forall j \in J \tag{3.7}$$

Ainda, na metodologia proposta o subproblema é resolvido para cada agrupamento k em cada iteração e as cinco melhores soluções encontradas são utilizadas. Dessa forma, até $5k$ colunas podem ser adicionadas por iteração. O modelo matemático para o subproblema referente ao agrupamento k é definido conforme segue.

Conjuntos:

$I =$ conjunto de elementos a serem recobertos

$J_k =$ coleção contendo os conjuntos de recobrimento do agrupamento k

Parâmetros:

$$\theta_k = \text{valor dual referente ao agrupamento } k$$

$$f_k = \text{custo para selecionar o agrupamento } k$$

$$c_j = \text{custo para selecionar o conjunto de recobrimento } j$$

$$\pi_i = \text{valor dual referente ao elemento } i$$

Variáveis:

$$x_j = \begin{cases} 1 & \text{se o conjunto de recobrimento } j \text{ é selecionado} \\ 0 & \text{caso contrário} \end{cases}$$

$$y_i = \begin{cases} 1 & \text{se o elemento } i \text{ é coberto} \\ 0 & \text{caso contrário} \end{cases}$$

Modelo matemático:

$$\text{Maximizar } -(\theta_k + f_k) - \sum_{j \in J_k} c_j x_j - \sum_{i \in I} \pi_i y_i \tag{3.8}$$

Sujeito a:

$$y_i \leq \sum_{j \in J} a_{ij} x_j \qquad \forall i \in I \tag{3.9}$$

$$y_i, x_j \in \{0, 1\} \tag{3.10}$$

A função objetivo (3.8) considera a maximização do custo reduzido das colunas passíveis de serem incluídas no problema mestre restrito relaxado. Os dois primeiros termos correspondem aos custos do agrupamento e conjuntos de recobrimento selecionados, enquanto o último termo se refere aos elementos cobertos. O valor de θ_k é definido pelo valor dual da restrição correspondente ao agrupamento k em (3.3). Da mesma forma π corresponde aos duais de (3.2). Em (3.9) é definido que para um elemento ser selecionado, ele deve ser coberto por pelo menos um conjunto. Os domínios das variáveis são definidos em (3.10).

O Algoritmo 3 resume a forma final do método com geração de colunas proposto nesse trabalho. Note que, como o intuito do algoritmo é apenas a geração de limitantes superiores, não é necessário encontrar uma solução inteira.

Algoritmo 3 Geração de colunas para o problema de Bilal et al. (2014)

Entrada: Problema mestre restrito (PMR), subproblema
Saída: Limitante superior para o problema mestre

```
1: Crie uma coluna inicial com uma configuração vazia
2: Repita
3:     Resolva o PMR, encontre θ e π
4:     Atualize θ e π no subproblema
5:     Resolva o subproblema para cada agrupamento
6:     Adicione as novas colunas ao PMR
7: enquanto há colunas com custo reduzido positivo        ▷ Problema de maximização
8: Retorna Limitante superior para o problema mestre
```

3.2 Iterated tabu search

3.2.1 Representação de solução

Uma solução é representada por um vetor binário com tamanho igual ao número de conjuntos de recobrimento da instância. Assim sendo, a posição j do vetor tem valor igual a 1 se o j-ésimo conjunto está selecionado ou zero caso contrário. O valor da função objetivo de uma solução é igual a soma dos ganhos referentes aos elementos cobertos, menos os custos pagos pela seleção de conjuntos de recobrimento e agrupamentos. O vetor v na Figura 3.1 exemplifica a representação de uma solução para uma instância com 6 conjuntos de recobrimento através. Nesse exemplo estão selecionados os conjuntos de índice 0, 3 e 4.

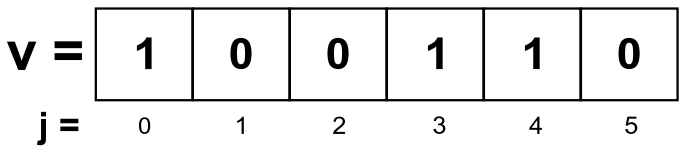

Figura 3.1 – Exemplo de representação da solução

3.2.2 Funções de perturbação

No algoritmo de Bilal et al. (2014) são utilizadas duas funções de perturbação com a finalidade de diversificar as soluções obtidas. As duas funções se diferem quanto a intensidade da perturbação gerada.

A primeira função de perturbação ($perturb1$), apresentada no Algoritmo 4, tem a inten-

ção de diversificar a busca ao redor da solução corrente. Isso é feito da forma que segue. Cada conjunto de recobrimento selecionado tem uma probabilidade p (intensidade da perturbação) de ser removido. Em seguida, se n conjuntos foram removidos, então n novos conjuntos aleatórios são adicionados. Se um dos conjuntos aleatórios já está na solução, ele continua selecionado e o seguinte é sorteado. Segundo os autores, ter um número de conjuntos adicionados menor ou igual ao de removidos é importante para a manter baixa a redundância de cobertura e evitar que o operador de busca tabu gaste tempo removendo conjuntos redundantes.

Algoritmo 4 $perturb_1$

Entrada: S, p
Saída: Solução perturbada S

```
1:  n ← 0
2:  Para cada conjunto de recobrimento j faça
3:      a ← número aleatório no intervalo [0, 1]
4:      Se a < p então
5:          S ← removeConjunto(S, j)
6:          n ← n + 1
7:      Fim Se
8:      Para i = 1 até n faça
9:          j ← conjunto de recobrimento aleatório
10:         S ← adicionaConjunto(S, j)
11:     Fim Para
12: Fim Para
13: Retorna S
```

Segundo Bilal et al. (2014), foi observada dificuldade de escapar de ótimos locais devido ao custo fixo associado aos agrupamentos. A segunda função de perturbação ($perturb2$), portanto, foi criada com o propósito de guiar a busca para uma nova região do espaço de soluções. A estratégia consiste em forçar agrupamentos a serem selecionados ou removidos da solução. Isso é feito através da manipulação dos custos fixos, da seguinte forma: seja M um número suficientemente grande, para forçar a exploração de um agrupamento que não está na solução subtrai-se M do seu custo, tornando-o negativo e fazendo com que o agrupamento fique muito atrativo. Da mesma forma, para forçar um agrupamento a sair da solução, remove-se todos os seus conjuntos e soma-se M ao seu custo, tornando-o inacessível e facilitando a evasão do ótimo local. Para liberar um agrupamento, simplesmente restaura-se o seu custo original.

A função $perturb2$ é chamada sempre que é detectada estagnação na busca, ou seja, quando não há melhora na solução incumbente por E_{its} iterações. Os agrupamentos ficam dispostos em uma fila e são forçados a entrar ou sair da solução, um a um, por até F_{max} iterações. Quando há melhora, o agrupamento forçado é liberado, o contador de estagnação e a fila são

reiniciados e o algoritmo retoma seu fluxo normal. Como os autores não especificam a forma como a fila é organizada, neste trabalho os agrupamentos são postos em ordem crescente de acordo com seus índices. O Algoritmo 5 ilustra a segunda função de perturbação.

Algoritmo 5 $perturb_2$

Entrada: $S, agr, estagn, contaF, F_{max}$
Saída: Solução perturbada S

1: **Se** $contaF == 0$ **então**
2: Força o agrupamento agr em S
3: **Senão**
4: **Se** $contaF == F_{max}$ **então**
5: Libera o agrupamento forçado em S
6: $agr \leftarrow$ próximo agrupamento na fila
7: $contaF \leftarrow 0$
8: **Se** agr é inválido **então** ▷ Todos agrupamentos já foram forçados
9: $agr \leftarrow$ primeiro agrupamento da fila
10: $estagn \leftarrow 0$
11: **Retorna** S
12: **Fim Se**
13: Força o agrupamento agr em S
14: **Fim Se**
15: **Fim Se**
16: $contaF \leftarrow contaF + 1$
17: **Retorna** S

3.2.3 Busca tabu

A busca tabu, ilustrada no Algoritmo 6, considera dois movimentos: adição e remoção. Dada a representação binária da solução, o movimento de adição do conjunto j consiste simplesmente em alterar o valor da posição j de zero para um. Da mesma forma, para remover j, seu respectivo valor é alterado de um para zero.

A cada iteração da busca tabu, todos os movimentos possíveis são avaliados, o melhor movimento não-tabu é executado e o seu inverso é adicionado à lista tabu. O melhor movimento é o que traz a melhor variação ao valor da função objetivo. O Algoritmo 7 exemplifica a função de avaliação de movimentos. Note que é utilizada uma estrutura de memória que armazena quantos conjuntos de cada agrupamento estão selecionados e quantos conjuntos ativos estão cobrindo cada elemento. A manutenção dessa estrutura simplifica os testes condicionais da linhas 4, 8, 14 e 18, evitando que seja necessário fazer uma recontagem a cada iteração. A escolha do melhor movimento reside em um laço de repetição, onde a função $avaliaMovimento$ é chamada para todo conjunto de recobrimento j. O conjunto que gera o maior valor de re-

Algoritmo 6 TS

Entrada: S, E_{ts}
Saída: Melhor solução encontrada S^*

Limpa listas tabu
$S^* \leftarrow S$
$estagn \leftarrow 0$
Enquanto $estagn < E_{ts}$ **faça**
    $m \leftarrow melhorMovimento(S)$
    $S \leftarrow executaMovimento(S, m)$
    Marca o inverso de m como tabu
    Se $avalia(S) > avalia(S^*)$ **então**
        $S^* \leftarrow S$
        $estagn \leftarrow 0$
    Fim Se
    $estagn \leftarrow estagn + 1$
Fim Enquanto
Retorna S^*

Algoritmo 7 $avaliaMovimento$

Entrada: S, j
Saída: Variação no valor da função objetivo de S provocada por adicionar/remover j

$variacao \leftarrow 0$
Se j está selecionado em S **então** ▷ Movimento de remoção
    $variacao \leftarrow variacao +$ custo do conjunto j
    Se j é o único conjunto selecionado de seu agrupamento **então**
        $variacao \leftarrow variacao +$ custo do agrupamento de j
    Fim Se
    Para cada elemento i coberto por j **faça**
        Se j é o único conjunto cobrindo i **então**
            $variacao \leftarrow variacao -$ ganho associado a i
        Fim Se
    Fim Para
Senão ▷ Movimento de adição
    $variacao \leftarrow variacao -$ custo do conjunto j
    Se não há nenhum conjunto do mesmo agrupamento de j selecionado **então**
        $variacao \leftarrow variacao -$ custo do agrupamento de j
    Fim Se
    Para cada elemento i coberto por j **faça**
        Se i não está sendo coberto por nenhum conjunto **então**
            $variacao \leftarrow variacao +$ ganho associado a i
        Fim Se
    Fim Para
Fim Se
Retorna $variacao$

torno (função objetivo de maximização) é escolhido. Se o conjunto está selecionado na solução corrente, então o movimento executado efetuará sua remoção. Caso contrário, o conjunto será adicionado à solução.

Ainda sobre a busca tabu, Bilal et al. (2014) optaram pela manutenção de duas listas tabu simultaneamente, sendo uma para cada tipo de movimento. Essa opção foi justificada pelo fato de a proibição de movimentos de remoção atrasar a convergência ao ótimo, pois, conforme citado anteriormente, movimentos de remoção costumam auxiliar na redução da redundância de cobertura. Assim sendo, a lista tabu para remoções tem tamanho menor do que a de adições. É considerado ainda um critério de aspiração, que permite que movimentos tabu sejam executados desde que levem a uma solução melhor do que todas visitadas até então.

Por fim, a busca tabu é encerrada quando a solução não é melhorada por E_{ts} iterações.

3.2.4 Algoritmo ITS

O Algoritmo 8 apresenta a organização geral da ITS. O primeiro passo é a obtenção de uma solução inicial, através de uma rodada de busca tabu. Em seguida, enquanto o limite de tempo T não é atingido, aplica-se a função de perturbação $perturb1$ seguida de uma rodada de busca tabu. Sempre que é detectada estagnação, a segunda função de perturbação também passa a ser utilizada. Note que a função $avalia$ deve considerar o custo original dos agrupamentos, caso algum esteja forçado na solução.

O Algoritmo 8 diferencia-se por incluir um critério de parada bem definido e por manter a solução incumbente armazenada. No algoritmo para a ITS apresentado por Bilal et al. (2014) não é especificado nenhum critério de parada para o laço de repetição equivalente ao da linha 7 do Algoritmo 8. A informação de que esse o critério é o tempo de execução, entretanto, pode ser encontrada em outro ponto do trabalho.

Quanto ao armazenamento da solução incumbente, a descrição original do algoritmo feita por possui uma falha que faz com que a referência à melhor solução encontrada até então seja sobrescrita na primeira chamada à função $perturb2$. A correção é feita no Algoritmo 8 através da adição do trecho entre as linhas 18 e 20 e com a inclusão de S^*, que tem como único propósito armazenar a solução incumbente.

Algoritmo 8 ITS

Entrada: $p, F_{max}, E_{ts}, E_{its}, T$
Saída: Melhor solução encontrada S^*

```
1: S ← solução vazia
2: S ← TS(S, E_ts)
3: S* ← S
4: estagn ← 0
5: agr ← primeiro agrupamento da fila
6: contaF ← 0
7: Enquanto tempoDecorrido < T faça
8:     S' ← perturb_1(S, p)
9:     S' ← TS(S', E_ts)
10:    Se avalia(S') > avalia(S) então
11:        S ← S'
12:        Se S tem agrupamento forçado então
13:            Libera o agrupamento forçado em S
14:        Fim Se
15:        estagn ← 0
16:        agr ← primeiro agrupamento da fila
17:        iterF ← 0
18:        Se avalia(S) > avalia(S*) então
19:            S* ← S                          ▷ Atualiza incumbente
20:        Fim Se
21:    Fim Se
22:    Se estagn > E_its então                 ▷ Estagnação detectada
23:        S ← perturb_2(S, agr, estagn, contaF, F_max)
24:    Fim Se
25:    estagn ← estagn + 1
26: Fim Enquanto
27: Retorna S*
```

3.3 Abordagem paralela para a ITS

Com a atual alta disponibilidade de *hardware multicore*, a computação paralela tem recebido crescente atenção. Quando utilizadas corretamente, técnicas de paralelismo podem gerar significativa economia de tempo ou permitir a abordagem de grandes problemas até então considerados inviáveis. São exemplos de trabalhos recentes que relacionam otimização combinatória e computação paralela: busca tabu cooperativa multivizinhança paralela (JIN; CRAINIC; LØKKETANGEN, 2012) e *simulated annealing* paralelo (WANG et al., 2015), ambos para problemas de roteamento de veículos, e um algoritmo genético de chaves aleatórias para um problema de carregamento de *containers* (GONÇALVES; RESENDE, 2012). Ainda, Crainic e Toulouse (2010) apresentam uma visão geral sobre estratégias e princípios de implementação

paralelos aplicáveis a meta-heurísticas.

Durante os testes de análise de desempenho conduzidos neste trabalho, foi observado que a maior parte do tempo de execução era dispendida na escolha do melhor movimento na busca tabu. Conforme mencionado anteriormente, essa escolha consiste em um laço de repetição onde é chamada a função de avaliação de movimentos (Algoritmo 7) para cada conjunto de recobrimento e é escolhido aquele que gerar o maior ganho na função objetivo. Se esse conjunto estiver selecionado na solução corrente, então o movimento executado efetuará sua remoção. Caso contrário, o conjunto será adicionado à solução.

Ainda, no Algoritmo 7 é possível observar que a avaliação de adição ou remoção de um conjunto de recobrimento j depende apenas da leitura de informações imutáveis como o custo do próprio conjunto, o de seu agrupamento e os ganhos relativos aos elementos por ele cobertos. Da mesma forma, a estrutura de memória mencionada anteriormente nunca é modificada durante a etapa de avaliação de movimentos. Portanto, como a avaliação de um movimento é completamente independente da avaliação dos demais, foi identificada uma oportunidade para paralelização da seleção do melhor movimento.

A abordagem implementada neste trabalho consistiu na paralelização da função *melhorMovimento* (linha 5 do Algoritmo 6), de forma que múltiplas *threads* avaliem simultaneamente uma parcela dos conjuntos cada uma. Após todas as *threads* finalizarem, os resultados individuais são comparados e o melhor deles é escolhido. A Figura 3.2 exemplifica a divisão da avaliação da solução em 2 *threads* paralelas para uma instância com 6 conjuntos. Nesse exemplo, a primeira *thread* avalia os movimentos relativos aos conjuntos de índices 0, 1 e 2, enquanto a segunda avalia 3, 4 e 5.

Por não haver nenhuma dependência de dados e envolver um gargalo no algoritmo, a paralelização trouxe amplo ganho de desempenho. Com o uso de 12 *threads*, em algumas das instâncias de maiores dimensões foram observadas reduções superiores a 80% no tempo necessário para atingir o mesmo número de iterações da abordagem sem paralelismo. A ferramenta escolhida foi a API OpenMP.

3.4 Vizinhança HVF para a ITS

Dado o modelo matemático apresentado na Seção 2.3, neste trabalho é definida uma nova vizinhança, que faz uso de programação inteira, e é proposta sua integração à meta-heurística ITS. Nessa vizinhança é incorporada uma estratégia heurística baseada em *hard vari-*

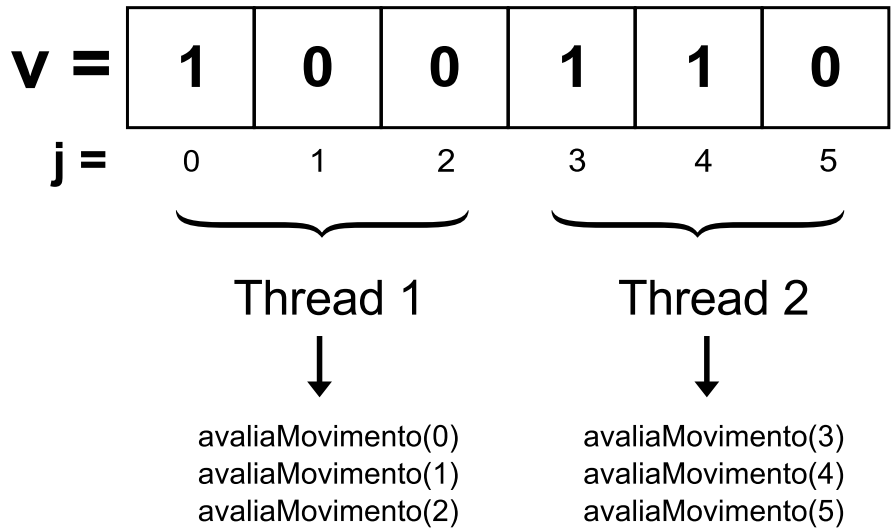

Figura 3.2 – Avaliação da solução com paralelismo

able fixing (HVF) ou *diving* (BIXBY et al., 2000). A estratégia resume-se a resolver o modelo sucessivas vezes fixando parte das variáveis que correspondem aos conjuntos de recobrimento ativos, i.e., cujo valor na solução corrente é não nulo.

Conforme visto na seção anterior, o uso de paralelismo trouxe significativo ganho de desempenho. A ideia subjacente ao uso da vizinhança HVF é ocupar o tempo economizado com o polimento de soluções promissoras obtidas pela busca tabu. Uma solução é considerada promissora se o valor de sua função objetivo está dentro de um determinado *gap* em relação à melhor solução encontrada até então. A escolha de não empregar para todas as soluções se deve ao custo computacional relativamente alto da vizinhança HVF. O *gap* é calculado conforme segue.

$$gap = \frac{S_{incumbente} - S_{corrente}}{S_{incumbente}} \times 100$$

O Algoritmo 9 ilustra a vizinhança HVF, que é formalizada conforme segue.

Dada uma solução de entrada S, os conjuntos de recobrimento que estão selecionados são organizados em uma fila ordenada aleatoriamente. A seguir são fixados todos, exceto os primeiros n, conjuntos da fila. O modelo matemático é resolvido e a solução obtida é avaliada. Se houver melhora em relação à solução inicial S, todos os conjuntos fixados são liberados, a fila é atualizada com os conjuntos de recobrimento da nova solução e eles são novamente

Algoritmo 9 $vizinhancaHVF$

Entrada: S, n, r
Saída: Melhor solução encontrada S^*

1: $S^* \leftarrow S$
2: $fila \leftarrow$ uma fila contendo todos os conjuntos selecionados em S
3: $fila \leftarrow ordenaAleatorio(fila)$
4: $estagn \leftarrow 0$
5: Fixa todos os conjuntos da fila, exceto os n primeiros
6: **Para** $j = 1$ até r **e** $estagn < r/2$ **faça**
7: $S \leftarrow resolveModelo()$
8: **Se** $avalia(S) > avalia(S^*)$ **então**
9: $S^* \leftarrow S$
10: Libera todos os conjuntos fixados
11: $fila \leftarrow$ fila atualizada contendo os conjuntos selecionados em S
12: $fila \leftarrow ordenaAleatorio(fila)$
13: Fixa todos os conjuntos da fila, exceto os n primeiros
14: $estagn \leftarrow 0$
15: **Senão**
16: Fixa os n primeiros conjuntos da fila
17: Move os n primeiros para o fim da fila
18: Libera os novos n primeiros da fila
19: $estagn \leftarrow estagn + 1$
20: **Fim Se**
21: **Fim Para**
22: Libera todos os conjuntos fixados
23: **Retorna** S^*

fixados segundo o mesmo critério considerado anteriormente. Caso não haja melhora, os primeiros n elementos são fixados e movidos para o fim da fila, enquanto os novos primeiros n são liberados. Esse processo é repetido r vezes e a melhor solução encontrada é retornada. Dado o significativo custo computacional deste algoritmo, é considerado ainda um critério de parada adicional, que determina o encerramento do mesmo caso não haja melhora na solução por $r/2$ iterações consecutivas.

O Algoritmo 10 apresenta a organização da ITS com a vizinhança HVF.

Algoritmo 10 $ITS\ HVF$

Entrada: $p, F_{max}, E_{ts}, E_{its}, T, gap, n, r$
Saída: Melhor solução encontrada S^*

1: $S \leftarrow$ solução vazia
2: $S \leftarrow TS(S, E_{ts})$
3: $S^* \leftarrow S$
4: $estagn \leftarrow 0$
5: $agr \leftarrow$ primeiro agrupamento da fila
6: $contaF \leftarrow 0$
7: **Enquanto** $tempoDecorrido < T$ **faça**
8: $S' \leftarrow perturb_1(S, p)$
9: $S' \leftarrow TS(S', E_{ts})$
10: **Se** $avalia(S')/avalia(S^*) > (1 - gap)$ **então**
11: $S' \leftarrow vizinhancaHVF(S', n, r)$
12: **Fim Se**
13: **Se** $avalia(S') > avalia(S)$ **então**
14: $S \leftarrow S'$
15: **Se** S tem agrupamento forçado **então**
16: Libera o agrupamento forçado em S
17: **Fim Se**
18: $estagn \leftarrow 0$
19: $agr \leftarrow$ primeiro agrupamento da fila
20: $iterF \leftarrow 0$
21: **Se** $avalia(S) > avalia(S^*)$ **então**
22: $S^* \leftarrow S$
23: **Fim Se**
24: **Fim Se**
25: **Se** $estagn > E_{its}$ **então**
26: $S \leftarrow perturb_2(S, agr, estagn, contaF, F_{max})$
27: **Fim Se**
28: $estagn \leftarrow estagn + 1$
29: **Fim Enquanto**
30: **Retorna** S^*

4 RESULTADOS COMPUTACIONAIS

Para facilitar a comparação de resultados, todos os testes computacionais descritos nesta seção utilizam o modelo matemático proposto por Bilal et al. (2014), ou seja, a função objetivo é de maximização.

Os testes foram realizados em um computador com dois processadores Intel Xeon E5-2697 v2 e 64 GB de memória principal. As implementações foram feitas em linguagem C++ e compiladas com o G++ v4.7. O paralelismo da busca tabu e do resolvedor CPLEX (versão 12.5) foi limitado em 12 *threads*.

4.1 Instâncias

As instâncias consideradas neste trabalho foram obtidas diretamente com Nehmé Bilal, um dos autores da meta-heurística ITS. O autor informa ainda que em seu trabalho as instâncias passaram por um pré-processamento durante a leitura, de forma que conjuntos de recobrimento com custo maior que a soma dos ganhos de seus elementos são eliminados.

A Tabela 4.1 apresenta as dimensões das 30 instâncias após o pré-processamento. De acordo com o número de elementos, as instâncias são aqui divididas em três grupos: A e B têm 1000 elementos e são consideradas pequenas, C e D são médias e têm 5192 elementos, enquanto E e F são grandes e têm 15000 elementos.

4.2 Limitantes superiores

Bilal et al. (2014) relatam dificuldade para tratar as instâncias consideradas grandes através de abordagens exatas, com isto, nenhuma solução factível foi obtida com o resolvedor CPLEX para estas instâncias. Das 20 instâncias pequenas ou médias, para apenas 7 instâncias foi provada a otimalidade da solução, sendo que o *gap* reportado para as demais varia entre 0,43% e 6,26%. Dessa forma, principalmente para as instâncias grandes, pouco é possível concluir sobre a qualidade das soluções obtidas pelos autores.

O modelo da Seção 3.1, que faz uso de geração de colunas, foi implementado para a obtenção de limitantes superiores, a fim de permitir uma melhor avaliação das soluções conhecidas. Os resultados, assim como os tempos individuais necessários para obtê-los, são apresentados na Tabela 4.2. As instâncias F4 e F5 não têm limitante superior apresentado devido

Tabela 4.1 – Dimensões das instâncias pré-processadas

Instância	Elementos	Conjuntos	Agrupamentos
A1	1000	3212	10
A2	1000	4060	10
A3	1000	6015	10
A4	1000	11900	10
A5	1000	76932	10
B1	1000	6362	20
B2	1000	9638	20
B3	1000	23654	20
B4	1000	36601	20
B5	1000	64087	20
C1	5192	5891	31
C2	5192	10651	31
C3	5192	21312	31
C4	5192	32516	31
C5	5192	56449	31
D1	5192	13303	62
D2	5192	24694	62
D3	5192	49034	62
D4	5192	75264	62
D5	5192	103760	62
E1	15000	8521	100
E2	15000	16264	100
E3	15000	30673	100
E4	15000	57508	100
E5	15000	67368	100
F1	15000	86308	100
F2	15000	105959	100
F3	15000	148104	100
F4	15000	239331	100
F5	15000	408690	100

a restrições de memória encontradas durante os testes computacionais. Na mesma tabela, é apresentada ainda uma coluna com um limitante superior obtido pela resolução da relaxação do modelo matemático original de Bilal et al. (2014). Para cada instância, ambos os limitantes superiores apresentados são comparados à melhor solução conhecida e os respectivos *gaps* são expostos. Em negrito estão destacadados os limitantes obtidos pelo método com geração de colunas que são pelo menos 50% menores que os obtidos pelo modelo relaxado. Algumas das melhores soluções foram obtidas durante os testes com diferentes configurações de parâmetros e podem, portanto, ser diferentes dos resultados definitivos apresentados na Seção 4.4.

Observa-se que o limitante superior obtido com o método com geração de colunas foi melhor para todas as instâncias testadas. Analisando os *gaps* reportados, nota-se uma diferença maior nas as instâncias pequenas e médias, enquanto os valores são mais próximos nas as grandes. Ao analisar os valores absolutos, entretanto, a diferença nos limitantes é substancial nas instâncias E e F, chegando a 11817 para F3.

4.3 Definição dos parâmetros

Para as instâncias médias observou-se que as soluções continham menor quantidade de conjuntos selecionados em relação às demais. Dessa forma, um número menor de conjuntos é fixado durante a vizinhança HVF, ocasionando um aumento no tempo necessário para sua resolução. Para essas instâncias, notou-se ainda a ocorrência de soluções muito diferentes, mas com valor da função objetivo muito próximo. Por consequência, mais soluções dentro do *gap* para execução da vizinhança HVF são identificadas e mais tempo é dispendido com a vizinhança, fazendo com o número total de iterações da ITS fosse drasticamente reduzido. Posto isso, para manter uma configuração de parâmetros consistente para instâncias pequenas e médias, optou-se pelo acréscimo de 1800 segundos ao tempo utilizado originalmente por Bilal et al. (2014).

Ainda, durante os testes computacionais com as instâncias grandes, foi observado que uma quantia significativa de tempo era consumida na aplicação da vizinhança HVF às soluções iniciais, geralmente de baixa qualidade. Como o polimento dessas soluções não é imprescindível e, de maneira geral, atrasou a convergência do algoritmo, adotou-se para essas instâncias um número mínimo de iterações ($iniIters$) antes que a vizinhança passe a ser aplicada. O valor escolhido para $iniIters$ é igual a 500.

Quanto aos demais parâmetros, a mesma configuração de Bilal et al. (2014) foi utilizada

Tabela 4.2 – Limitantes superiores

Inst.	Limitante superior GC	Tempo (s)	Melhor solução conhecida	Gap lim. superior GC (%)	Modelo relaxado	Gap lim. superior modelo relaxado(%)
A1	156106	24	150386	3,8035	160769	6,9042
A2	181954	11	179973	**1,1007**	185623	3,1394
A3	160944	52	155266	3,6570	166291	7,1007
A4	162141	109	156559	3,5654	167526	7,0050
A5	166143	1339	160767	**3,3440**	171843	6,8895
B1	159300	36	152335	4,5722	165053	8,3487
B2	162291	71	155752	4,1983	167915	7,8092
B3	163854	209	158176	**3,5897**	170538	7,8153
B4	164902	369	158411	**4,0976**	171912	8,5228
B5	166082	790	159581	**4,0738**	172975	8,3932
C1	247057	1459	246121	**0,3803**	248582	0,9999
C2	248816	2677	247455	**0,5500**	250369	1,1776
C3	249483	9628	249070	**0,1658**	251252	0,8761
C4	249801	39848	249124	**0,2718**	251627	1,0047
C5	250335	162418	249935	**0,1600**	252237	0,9210
D1	249652	5933	247250	0,9715	251872	1,8694
D2	250612	12689	248424	**0,8808**	252930	1,8138
D3	251145	38730	249165	**0,7947**	253584	1,7735
D4	251275	68830	249107	**0,8703**	253799	1,8835
D5	252246	129347	250577	**0,6661**	254666	1,6318
E1	659351	10985	623606	5,7320	664109	6,4950
E2	526610	81686	449288	17,2099	533357	18,7116
E3	619142	285133	551531	12,2588	628970	14,0407
E4	667919	30986	637825	4,7182	671402	5,2643
E5	560360	1978191	474222	18,1641	570563	20,3156
F1	575672	361917	501319	14,8315	582597	16,2128
F2	565596	403911	495887	14,0574	571768	15,3021
F3	566832	538077	485865	16,6645	578649	19,0967
F4	-	-	485126	-	587882	21,1822
F5	-	-	498559	-	601381	20,6238

para as instâncias pequenas e médias. O valor de p (intensidade da perturbação 1), que não é especificado no trabalho original, foi obtido em contato com Nehmé Bilal e é igual a 0,3. Para as instâncias grandes, p com valor igual a 0,2 mostrou-se mais adequado. No tempo limite não

houve alteração e são utilizados, portanto, 36000 segundos. Da mesma forma, F_{max}, E_{ts} e E_{its} mantém os valores originais.

No que concerne a vizinhança HVF, n e r têm valor igual a 5 e 10, respectivamente, para todas as instâncias. O *gap* máximo para a execução da vizinhança é igual a 1% para instâncias pequenas e médias, e 0,2% para as grandes.

A Tabela 4.3 apresenta todos os parâmetros e respectivos valores utilizados neste trabalho. Note que T é variável para as instâncias pequenas e médias.

Tabela 4.3 – Valores utilizados para os parâmetros

Parâmetro	Inst. pequenas e médias	Instâncias grandes
E_{ts}	1000	1000
E_{its}	200	200
p	0,3	0,2
F_{max}	50	1
n	5	5
r	10	10
gap	0,01	0,002
$iniIters$	0	500
T	Entre 3600 e 9000 s	36000 s

4.4 Resultados da abordagem heurística

A metodologia seguida para os testes computacionais é a mesma de Bilal et al. (2014). A ITS com vizinhança HVF foi executada cinco vezes para cada instância e a melhor, a pior e a solução média são apresentadas. Os resultados obtidos para as instâncias pequenas e médias são apresentados na Tabela 4.4.

Em negrito são destacadas soluções encontradas que são melhores do que as melhores soluções conhecidas apresentadas por Bilal et al. (2014). A coluna t_m informa o tempo médio necessário para obtenção da melhor solução. It_m e M_m apresentam as médias para, respectivamente, a quantia total de iterações do algoritmo e o número de vezes que a solução incumbente foi melhorada. Já HVF_m especifica o número médio de vezes que a vizinhança HVF foi capaz de melhorar a solução incumbente. P_m indica o número médio de vezes que foi detectada estagnação e a segunda função de perturbação foi acionada, enquanto PM_m apresenta o número médio de vezes em que o uso dessa função levou a uma melhora na solução. Por fim, T mostra

Tabela 4.4 – Resultados para as instâncias pequenas e médias

Inst.	Pior	Melhor	Média	t_m (s)	It_m	M_m	HVF_m	P_m	PM_m	T (s)
A1	150368	150386	150378,8	393,0	26573,0	15,6	14,6	38,2	1,6	3600
A2	179973	179973	179973,0	1374,0	1731,6	4,6	4,6	3,0	0,8	3600
A3	155266	155266	155266,0	628,6	47157,4	19,6	19,0	67,6	1,2	3600
A4	156395	156540	156453,0	2378,0	13603,2	20,8	20,4	20,0	2,6	3600
A5	160235	**160767**	160438,8	3548,2	2972,2	20,4	20,2	4,2	1,0	5400
B1	152223	152335	152306,2	296,6	26114,2	9,0	8,6	22,2	0,4	3600
B2	155554	**155752**	155593,6	1445,0	10783,6	16,4	15,2	9,4	1,8	3600
B3	157337	**158176**	157556,0	2339,6	8113,8	19,0	19,0	7,2	1,6	3600
B4	158104	**158411**	158254,2	2922,0	3175,6	17,0	16,8	2,6	0,8	3600
B5	158379	**159581**	158966,4	1637,4	1129,6	12,6	12,4	1,0	0,0	3600
C1	246121	246121	246121,0	61,4	1336,2	5,4	5,4	1,0	0,0	3600
C2	247447	247455	247450,2	1503,6	506,0	7,4	7,2	1,0	0,4	3600
C3	249070	249070	249070,0	1091,4	221,2	7,0	7,0	0,2	0,0	3600
C4	249088	249124	249110,8	2375,2	150,8	7,6	7,4	0,0	0,0	3600
C5	249838	249935	249881,2	2811,6	119,4	9,6	9,6	0,0	0,0	5400
D1	247113	**247250**	247185,4	1274,8	1011,6	13,6	13,6	0,8	0,0	3600
D2	247935	248389	248087,4	1553,8	249,2	14,2	14,0	0,2	0,0	3600
D3	248483	**249165**	248818,4	3269,0	138,0	10,8	10,8	0,0	0,0	5400
D4	248353	248949	248691,0	7257,0	150,2	14,8	14,8	0,0	0,0	9000
D5	249860	**250288**	250019,2	7514,2	79,0	13,0	12,8	0,0	0,0	9000

o tempo limite utilizado para cada instância.

A Tabela 4.5 apresenta uma comparação aos resultados de Bilal et al. (2014). A medida utilizada é o desvio relativo percentual (DRP), que é calculado da seguinte forma.

$$DRP = \frac{S_{obtida} - S_{original}}{S_{original}} \times 100$$

Tabela 4.5 – Desvio relativo percentual em relação aos resultados de Bilal et al. (2014) para instâncias pequenas e médias

Inst.	Pior(%)	Melhor(%)	Média(%)	$t_m(\%)$	$It_m(\%)$	$M_m(\%)$	$P_m(\%)$	$PM_m(\%)$	$T(\%)$
A1	**0,140**	0,000	**0,089**	147,170	30,100	6,849	27,333	-27,273	100
A2	0,000	0,000	0,000	1026,230	-81,482	-51,064	-78,571	-50,000	100
A3	**0,724**	0,000	**0,230**	1,584	384,660	-30,000	376,056	-33,333	100
A4	**0,097**	**0,075**	**0,073**	395,210	180,652	-11,111	194,118	160,000	100
A5	**0,352**	**0,306**	**0,297**	210,864	69,743	-35,443	90,909	66,667	50
B1	**0,074**	0,000	**0,011**	272,613	165,658	-42,308	141,304	-75,000	100
B2	**0,093**	**0,127**	**0,045**	419,410	62,968	-8,889	56,667	80,000	100
B3	**0,056**	**0,397**	**0,064**	158,634	235,697	-20,168	157,143	-11,111	100
B4	**0,147**	**0,005**	**0,025**	197,072	85,599	-21,296	8,333	-42,857	100
B5	**0,051**	**0,097**	**0,088**	101,949	9,246	-37,624	0,000	-100,000	100
C1	0,000	0,000	0,000	490,385	-75,333	-71,875	-66,667	0,000	100
C2	0,000	0,000	0,000	323,311	-81,759	-54,321	-70,588	-81,000	100
C3	0,000	0,000	0,000	1066,026	-86,762	-71,774	-90,000	-100,000	100
C4	**0,011**	**0,012**	**0,015**	336,778	-89,542	-65,766	-100,000	-100,000	100
C5	**0,135**	**0,022**	**0,050**	59,478	-93,936	-53,398	-100,000	-100,000	50
D1	**0,002**	**0,021**	**0,006**	117,840	-63,414	-50,365	-76,471	-100,000	100
D2	**0,086**	0,000	**0,048**	57,778	-81,093	-54,777	-80,000	-100,000	100
D3	**0,004**	**0,097**	**0,062**	32,445	-88,725	-57,813	-100,000	-100,000	50
D4	**0,099**	-0,063	**0,008**	34,989	-89,302	-57,471	-100,000	-100,000	25
D5	**0,102**	**0,052**	**0,025**	118,461	-94,429	-58,861	-100,000	-100,000	25

A comparação mostra uma consistente melhora na qualidade das soluções obtidas. A solução média é melhorada em 16 das 20 instâncias, sendo que para três das quatro restantes o algoritmo já era capaz de chegar ao ótimo em todas as execuções (A2, C1 e C3). O mesmo comportamento é observado nas piores soluções. A melhor solução obtida foi melhorada em 11 instâncias, sendo que para 8 obteve-se uma nova melhor solução conhecida. Em D4 nota-se uma exceção, onde a melhor solução é 0,063% pior do que a de Bilal et al. (2014). Em negrito são destacados os casos onde houve melhora na qualidade da solução.

Na coluna t_m observa-se que, para as instâncias pequenas e médias, a adição da vizinhança HVF leva a um aumento significativo, em geral, no tempo para obter a melhor solução. O número de iterações é superior para as instâncias A e B (A2 é exceção) e inferior para C e

D. Conforme mencionado anteriormente, características intrínsecas a C e D fazem com que a resolução da vizinhança HVF tome mais tempo, reduzindo o total de iterações. Contudo, apesar dessas significativas reduções, foi possível melhorar a qualidade das soluções. Ainda, para as instâncias onde houve redução do total de iterações, observa-se, conforme esperado, uma redução no número de chamadas à segunda função de perturbação. Para as demais nota-se que a perturbação foi ativada mais vezes. Em 5 instâncias, a função nunca foi utilizada. Na coluna PM_m é possível observar que, quando utilizada, houve grande variação quanto ao número de vezes que a perturbação foi capaz de melhorar a solução. Já em M_m é visto que, em geral, o uso da vizinhança HVF reduziu o número de vezes que a solução incumbente é atualizada. Por fim, T apresenta valores maiores que zero, pois o tempo limite utilizado é maior.

Na Tabela 4.6 são apresentados os resultados para as instâncias grandes.

Tabela 4.6 – Resultados para as instâncias grandes

Inst.	Pior	Melhor	Média	t_m (s)	It_m	M_m	MA_m	MV_m	P_m	PM_m
E1	620895	**623606**	621962,0	17716,6	59752,0	55,4	23,8	14,4	195,8	2,8
E2	448420	449257	448898,8	23195,2	70043,8	61,6	28,0	12,6	229,8	2,4
E3	547653	**551531**	549323,0	28508,2	40480,8	63,0	37,2	22,6	129,2	3,0
E4	635868	**637825**	636640,4	24749,8	4967,6	63,4	26,4	25,6	11,6	1,2
E5	470445	**474222**	471652,8	27756,6	43478,8	62,6	25,8	19,2	139,0	1,6
F1	499462	**501319**	500452,8	27000,2	13180,6	79,8	39,4	35,8	37,6	1,8
F2	493710	**495887**	494511,6	29901,0	23419,8	82,4	43,6	42,4	70,8	3,0
F3	483929	**485865**	484909,0	25523,6	12011,8	57,8	22,8	21,2	34,6	1,6
F4	482359	**485126**	484126,4	18434,8	8305,8	44,0	10,8	10,0	24,6	0,6
F5	495644	**498559**	496947,2	16953,8	3563,0	36,0	5,2	5,2	9,4	0,0

Para 9 das 10 instâncias foram conhecidas novas melhores soluções e elas são destacadas em negrito.

Conforme mencionado anteriormente, para as instâncias grandes foi adotado um número mínimo de iterações necessárias para que a vizinhança HVF passe a ser aplicada. Dessa forma, na Tabela 4.6 as colunas MA_m e MV_m indicam, respectivamente, o número médio de vezes que a solução incumbente foi atualizada após as iterações iniciais, e o número médio de vezes em que a vizinhança foi capaz de melhorar a solução incumbente. As demais colunas apresentam as mesmas informações descritas para as tabelas anteriores.

A comparação aos resultados de Bilal et al. (2014) é apresentada na Tabela 4.7.

Em negrito são destacados os resultados para os quais houve melhora na qualidade da solução. A pior solução e solução média são melhores para todas as instâncias, enquanto a

Tabela 4.7 – Desvio relativo percentual em relação aos resultados de Bilal et al. (2014) para instâncias grandes

Inst.	Pior(%)	Melhor(%)	Média(%)	$t_m(\%)$	$It_m(\%)$	$M_m(\%)$	$P_m(\%)$	$PM_m(\%)$
E1	**0,591**	**0,614**	**0,503**	38,337	5,177	60,116	5,043	133,333
E2	**0,639**	-0,007	**0,430**	45,282	13,625	-2,532	16,768	-14,286
E3	**0,701**	**0,624**	**0,745**	93,271	73,121	42,534	81,461	150,000
E4	**0,090**	**0,128**	**0,090**	-13,417	-46,705	37,826	-57,664	-85,714
E5	**0,436**	**0,437**	**0,148**	-3,603	258,825	32,068	323,780	0,000
F1	**0,656**	**0,104**	**0,294**	-13,026	51,192	78,125	50,400	-73,529
F2	**1,245**	**0,314**	**0,797**	-8,487	184,221	92,523	195,000	-59,459
F3	**0,465**	**0,115**	**0,358**	-26,266	125,871	45,960	147,143	-76,471
F4	**0,486**	**0,060**	**0,489**	-29,940	273,798	69,231	355,556	-82,353
F5	**0,636**	**0,706**	**0,595**	-51,137	150,211	80,000	193,750	-100,000

melhor solução é pior apenas para E2 (e apenas 0,007% pior).

Na coluna t_m é observado um aumento para as 3 primeiras instâncias, enquanto há redução para as demais. Quanto ao total de iterações, apenas E4 teve redução. Da mesma forma, o número de chamadas à segunda função de perturbação é menor apenas para E4. Embora a função, em geral, tenha sido utilizada mais vezes, foi observada redução em PM_m para 7 instâncias. Na coluna M_m é visto que, à exceção de E2, o número de atualizações da função incumbente foi superior. É possível que isso seja consequência da opção pela redução do valor do parâmetro p, que gera mudanças mais sutis nas soluções a cada iteração e faz com que a convergência se dê através de passos menores. Por fim, o mesmo tempo de Bilal et al. (2014) foi utilizado e o DRP, portanto, é igual a zero para as instâncias grandes.

5 CONCLUSÕES

Este trabalho abordou o PRCA, uma variante do PRC clássico na qual os conjuntos de recobrimento são organizados em K agrupamentos disjuntos. Sobre cada agrupamento incide um custo fixo a ser pago uma única vez sempre que ao menos um conjunto contido é selecionado.

Foram apresentados ainda, dois outros problemas encontrados na literatura que são aplicações práticas para o PRCA. O primeiro deles, de Fritzen et al. (2012), tem origem na engenharia elétrica, especificamente no processamento de alarmes de proteção em sistemas elétricos de potência. Uma característica intrínseca ao problema levou à definição de uma generalização do PRCA, de forma que os agrupamentos deixam de ser disjuntos e a intersecção é permitida. O segundo problema é devido a Bilal et al. (2014) e é oriundo da indústria de mineração. São propostos um conjunto de instâncias e uma meta-heurística baseada em ITS, que guiaram os experimentos computacionais desenvolvidos neste trabalho.

A primeira contribuição apresentada neste trabalho consistiu na proposição de uma formulação com geração de colunas, que foi utilizada para geração de limitantes superiores para as instâncias. Essa contribuição tem grande importância devido ao fato de não haver até então qualquer estimativa semelhante para uma parte das instâncias, o que impedia a avaliação da qualidade das soluções conhecidas. Outras duas importantes contribuições se referem ao método ITS. A primeira consistiu na paralelização do componente de busca tabu, enquanto a segunda residiu na adição de uma nova vizinhança, resolvida com programação inteira, e que baseia-se em *hard variable fixing*.

Nos testes computacionais, a meta-heurística foi executada 5 vezes para cada instância e a pior solução, melhor solução e solução média foram apresentadas. Os resultados mostraram que as contribuições ao método ITS proporcionaram melhora na qualidade das soluções para a ampla maioria das instâncias. Observando a solução média e a pior solução, notou-se melhora em 26 das 30 instâncias, sendo que para 3 das demais o algoritmo já era capaz de sempre chegar ao ótimo. Quanto à melhor solução, foi possível observar melhoria para 20 instâncias, sendo que os resultados de 17 são as novas melhores soluções conhecidas. É importante ressaltar que, embora os percentuais de melhora pareçam pequenos, quando os valores são tomados em absoluto, as melhoras são bastante significativas, principalmente para as instâncias grandes. Visto que os custos envolvidos no problema são de grande magnitude, a utilização das estratégias

propostas neste trabalho podem ser de grande benefício econômico.

De maneira geral, conclui-se que os resultados obtidos foram satisfatórios e os objetivos do trabalho foram atingidos.

REFERÊNCIAS

ALFANDARI, L.; MONNOT, J. A note on the Clustered Set Covering Problem. **Discrete Applied Mathematics**, [S.l.], v.164, p.13–19, Feb. 2014.

BARNHART, C.; BELOBABA, P.; ODONI, A. R. Applications of Operations Research in the Air Transport Industry. **Transportation Science**, [S.l.], v.37, n.4, p.368–391, 2003.

BEASLEY, J.; CHU, P. A genetic algorithm for the set covering problem. **European Journal of Operational Research**, [S.l.], v.94, n.2, p.392 – 404, 1996.

BILAL, N.; GALINIER, P.; GUIBAULT, F. An iterated-tabu-search heuristic for a variant of the partial set covering problem. **Journal of Heuristics**, [S.l.], v.20, n.2, p.143–164, Jan. 2014.

BIXBY, E. et al. MIP: theory and practice — closing the gap. In: POWELL, M.; SCHOLTES, S. (Ed.). **System Modelling and Optimization**. [S.l.]: Springer US, 2000. p.19–49. (IFIP — The International Federation for Information Processing, v.46).

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: overview and conceptual comparison. **ACM Computing Surveys (CSUR)**, [S.l.], v.35, n.3, p.268–308, 2003.

COOK, S. A. The Complexity of Theorem-proving Procedures. In: THIRD ANNUAL ACM SYMPOSIUM ON THEORY OF COMPUTING, New York, NY, USA. **Proceedings...** ACM, 1971. p.151–158. (STOC '71).

CRAINIC, T. G.; TOULOUSE, M. Parallel Meta-heuristics. In: GENDREAU, M.; POTVIN, J.-Y. (Ed.). **Handbook of Metaheuristics**. [S.l.]: Springer US, 2010. p.497–541. (International Series in Operations Research & Management Science, v.146).

DANTZIG, G. B.; WOLFE, P. Decomposition Principle for Linear Programs. **Operations Research**, [S.l.], v.8, n.1, p.101–111, 1960.

DAYARIAN, I. et al. A column generation approach for a multi-attribute vehicle routing problem. **European Journal of Operational Research**, [S.l.], v.241, n.3, p.888 – 906, 2015.

DESAULNIERS, G. et al. Crew pairing at Air France. **European Journal of Operational Research**, [S.l.], v.97, n.2, p.245 – 259, 1997.

DONG, X.; HUANG, H.; CHEN, P. An iterated local search algorithm for the permutation flowshop problem with total flowtime criterion. **Computers & Operations Research**, [S.l.], v.36, n.5, p.1664 – 1669, 2009. Selected papers presented at the Tenth International Symposium on Locational Decisions (ISOLDE X).

FRITZEN, P. C. et al. Hybrid system based on constructive heuristic and integer programming for the solution of problems of fault section estimation and alarm processing in power systems. **Electric Power Systems Research**, [S.l.], v.90, p.55–66, Sept. 2012.

GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability**: a guide to the theory of np-completeness. New York, NY, USA: W. H. Freeman & Co., 1979.

GLOVER, F. Tabu Search—Part I. **ORSA Journal on Computing**, [S.l.], v.1, n.3, p.190–206, 1989.

GONÇALVES, J. F.; RESENDE, M. G. A parallel multi-population biased random-key genetic algorithm for a container loading problem. **Computers & Operations Research**, [S.l.], v.39, n.2, p.179 – 190, 2012.

JIN, J.; CRAINIC, T. G.; LØKKETANGEN, A. A parallel multi-neighborhood cooperative tabu search for capacitated vehicle routing problems. **European Journal of Operational Research**, [S.l.], v.222, n.3, p.441 – 451, 2012.

KARP, R. M. Reducibility among Combinatorial Problems. In: MILLER, R.; THATCHER, J.; BOHLINGER, J. (Ed.). **Complexity of Computer Computations**. [S.l.]: Springer US, 1972. p.85–103. (The IBM Research Symposia Series).

MARMION, M.-E. et al. NILS: a neutrality-based iterated local search and its application to flowshop scheduling. In: MERZ, P.; HAO, J.-K. (Ed.). **Evolutionary Computation in Combinatorial Optimization**. [S.l.]: Springer Berlin Heidelberg, 2011. p.191–202. (Lecture Notes in Computer Science, v.6622).

MASSON, R. et al. An iterated local search heuristic for multi-capacity bin packing and machine reassignment problems. **Expert Systems with Applications**, [S.l.], v.40, n.13, p.5266 – 5275, 2013.

MICHALLET, J. et al. Multi-start iterated local search for the periodic vehicle routing problem with time windows and time spread constraints on services. **Computers & Operations Research**, [S.l.], v.41, n.0, p.196 – 207, 2014.

OOSTRUM, J. M. van et al. A master surgical scheduling approach for cyclic scheduling in operating room departments. **OR Spectrum**, [S.l.], v.30, n.2, p.355–374, 2008.

OSMAN, I. H.; KELLY, J. P. **Meta-Heuristics**: theory and applications. Norwell, MA, USA: Kluwer Academic Publishers, 1996.

PENG, Y.; REGGIA, J. A. Plausibility of Diagnostic Hypotheses: the nature of simplicity. In: AAAI. **Anais...** Morgan Kaufmann, 1986. p.140–147.

PENNA, P.; SUBRAMANIAN, A.; OCHI, L. An Iterated Local Search heuristic for the Heterogeneous Fleet Vehicle Routing Problem. **Journal of Heuristics**, [S.l.], v.19, n.2, p.201–232, 2013.

TAŞ, D. et al. Vehicle routing with soft time windows and stochastic travel times: a column generation and branch-and-price solution approach. **European Journal of Operational Research**, [S.l.], v.236, n.3, p.789 – 799, 2014. Vehicle Routing and Distribution Logistics.

WALKER, J. D. et al. Vehicle Routing and Adaptive Iterated Local Search within the HyFlex Hyper-heuristic Framework. In: HAMADI, Y.; SCHOENAUER, M. (Ed.). **Learning and Intelligent Optimization**. [S.l.]: Springer Berlin Heidelberg, 2012. p.265–276. (Lecture Notes in Computer Science).

WANG, C. et al. A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup–delivery and time windows. **Computers & Industrial Engineering**, [S.l.], v.83, n.0, p.111 – 122, 2015.

APÊNDICES

APÊNDICE A – Resultados individuais da ITS com vizinhança HVF

A.1 Instâncias pequenas

Tabela A.1 – Resultados individuais para instâncias A

Execução	Solução	t (s)	It	M	HVF	P	PM	T (s)
			A1					
1	150368	180	27349	10	8	39	1	3600
2	150386	330	27125	17	16	39	1	3600
3	150386	598	26828	19	18	39	3	3600
4	150368	708	22971	20	19	33	2	3600
5	150386	149	28592	12	12	41	1	3600
			A2					
1	179973	974	1597	7	7	3	1	3600
2	179973	3388	1787	5	5	3	1	3600
3	179973	979	1550	6	6	3	1	3600
4	179973	3	1929	1	1	3	0	3600
5	179973	1526	1795	4	4	3	1	3600
			A3					
1	155266	967	41632	17	17	59	1	3600
2	155266	676	47659	20	20	69	2	3600
3	155266	414	50143	21	21	72	1	3600
4	155266	545	52068	17	15	75	1	3600
5	155266	541	44285	23	22	63	1	3600
			A4					
1	156395	3319	10226	24	23	15	3	3600
2	156449	3051	9583	19	19	16	6	3600
3	156449	2449	14854	17	17	21	2	3600
4	156540	176	20564	18	18	29	0	3600
5	156432	2895	12789	26	25	19	2	3600
			A5					
1	160618	3184	3418	15	15	5	1	5400
2	160324	2709	4524	21	20	7	1	5400
3	160767	3541	3654	22	22	6	2	5400
4	160250	4525	1311	21	21	1	1	5400
5	160235	3782	1954	23	23	2	0	5400

Tabela A.2 – Resultados individuais para instâncias B

Execução	Solução	t (s)	It	M	HVF	P	PM	T (s)
			B1					
1	152335	198	24563	9	9	21	0	3600
2	152335	1150	26993	14	13	23	1	3600
3	152303	20	27018	8	8	23	1	3600
4	152223	61	23529	9	9	20	0	3600
5	152335	54	28468	5	4	24	0	3600
			B2					
1	155752	1645	12005	18	17	10	2	3600
2	155554	672	11038	14	12	10	1	3600
3	155554	2364	8729	21	21	8	3	3600
4	155554	589	11563	12	12	10	1	3600
5	155554	1955	10583	17	14	9	2	3600
			B3					
1	157337	2448	7683	13	13	7	2	3600
2	158176	1212	10506	15	15	9	1	3600
3	157415	3078	6514	27	27	6	2	3600
4	157339	3009	6927	13	13	7	2	3600
5	157513	1951	8939	27	27	7	1	3600
			B4					
1	158251	2058	3437	15	14	3	1	3600
2	158109	2503	2197	16	16	2	0	3600
3	158411	3118	1636	18	18	1	1	3600
4	158104	3502	3664	18	18	3	1	3600
5	158396	3429	4944	18	18	4	1	3600
			B5					
1	158878	3273	384	17	16	0	0	3600
2	159581	1646	1090	14	14	1	0	3600
3	158796	532	2627	6	6	2	0	3600
4	158379	1340	703	16	16	1	0	3600
5	159198	1396	844	10	10	1	0	3600

A.2 Intâncias médias

Tabela A.3 – Resultados individuais para instâncias C

Execução	Solução	t (s)	It	M	HVF	P	PM	T (s)
			C1					
1	246121	30	1455	4	4	1	0	3600
2	246121	52	1155	3	3	1	0	3600
3	246121	84	1347	6	6	1	0	3600
4	246121	60	1441	5	5	1	0	3600
5	246121	81	1283	9	9	1	0	3600
			C2					
1	247447	318	546	7	7	1	0	3600
2	247447	2960	447	6	6	1	1	3600
3	247455	1039	395	12	12	1	0	3600
4	247447	3108	534	5	4	1	1	3600
5	247455	93	608	7	7	1	0	3600
			C3					
1	249070	910	223	9	9	0	0	3600
2	249070	183	235	6	6	1	0	3600
3	249070	1477	210	7	7	0	0	3600
4	249070	894	221	4	4	0	0	3600
5	249070	1993	217	9	9	0	0	3600
			C4					
1	249124	696	154	8	7	0	0	3600
2	249124	3406	156	8	8	0	0	3600
3	249088	3104	156	10	10	0	0	3600
4	249094	3108	149	4	4	0	0	3600
5	249124	1562	139	8	8	0	0	3600
			C5					
1	249838	1993	117	10	10	0	0	5400
2	249849	3539	119	9	9	0	0	5400
3	249935	1915	123	9	9	0	0	5400
4	249849	4064	108	10	10	0	0	5400
5	249935	2547	130	10	10	0	0	5400

Tabela A.4 – Resultados individuais para instâncias D

Execução	Solução	t (s)	It	M	HVF	P	PM	T (s)
			D1					
1	247166	251	643	9	9	1	0	3600
2	247199	1148	553	17	17	1	0	3600
3	247199	1943	487	15	15	1	0	3600
4	247113	2402	476	11	11	0	0	3600
5	247250	630	2899	16	16	1	0	3600
			D2					
1	248071	1796	241	13	13	0	0	3600
2	248389	648	264	13	13	1	0	3600
3	248079	765	239	15	14	0	0	3600
4	247963	3576	248	18	18	0	0	3600
5	247935	984	254	12	12	0	0	3600
			D3					
1	248484	1476	125	8	8	0	0	5400
2	248804	5385	145	13	13	0	0	5400
3	249156	3644	142	12	12	0	0	5400
4	248483	1830	127	8	8	0	0	5400
5	249165	4010	151	13	13	0	0	5400
			D4					
1	248781	7118	183	17	17	0	0	9000
2	248756	7928	132	15	15	0	0	9000
3	248949	6435	180	16	16	0	0	9000
4	248353	8638	107	13	13	0	0	9000
5	248616	6166	149	13	13	0	0	9000
			D5					
1	250052	8332	82	15	15	0	0	9000
2	249860	5179	88	9	9	0	0	9000
3	249953	7495	57	13	13	0	0	9000
4	249943	7875	76	15	15	0	0	9000
5	250288	8690	92	13	12	0	0	9000

A.3 Instâncias grandes

Tabela A.5 – Resultados individuais para instâncias E

Execução	Solução	t (s)	It	M	MA	HVF	P	PM	T (s)
				E1					
1	620895	9821	50004	46	20	15	164	3	36000
2	621871	91	87664	49	0	0	291	1	36000
3	623606	24062	68300	82	41	22	223	3	36000
4	622020	34022	49749	45	24	14	162	4	36000
5	621418	20587	43043	55	34	21	139	3	36000
				E2					
1	449257	32098	42029	72	40	21	135	3	36000
2	448420	26833	17706	73	36	17	56	3	36000
3	449254	35852	64749	60	28	13	212	3	36000
4	448691	18368	84159	56	28	12	277	2	36000
5	448872	2825	141576	47	8	0	469	1	36000
				E3					
1	549258	35996	43755	67	36	20	140	3	36000
2	551531	23088	46745	57	39	22	151	2	36000
3	547653	34667	19664	71	42	27	58	4	36000
4	548115	22536	42287	54	28	17	136	3	36000
5	550058	26254	49953	66	41	27	161	3	36000
				E4					
1	636792	2570	10492	49	3	3	33	0	36000
2	636258	35102	3257	82	47	46	3	1	36000
3	636459	20039	5079	55	26	25	13	2	36000
4	637825	35334	3070	70	28	28	4	2	36000
5	635868	30704	2940	61	28	26	5	1	36000
				E5					
1	470558	15932	41735	58	24	14	134	1	36000
2	471457	35443	54067	45	7	6	177	1	36000
3	471582	34600	50235	70	28	24	162	3	36000
4	470445	21309	33977	73	36	26	104	2	36000
5	474222	31499	37380	67	34	26	118	1	36000

Tabela A.6 – Resultados individuais para instâncias F

Execução	Solução	t (s)	It	M	MA	HVF	P	PM	T (s)
				F1					
1	499462	35969	2986	94	46	39	1	1	36000
2	501305	19004	20473	88	44	43	64	1	36000
3	500006	29107	6712	79	47	45	15	2	36000
4	500172	35015	9127	70	31	24	23	3	36000
5	501319	15906	26605	68	29	28	85	2	36000
				F2					
1	493823	24953	17375	85	60	59	48	2	36000
2	495887	30506	14291	100	55	54	39	4	36000
3	493908	26661	26953	87	42	40	83	3	36000
4	495230	33239	38682	58	25	25	125	2	36000
5	493710	34146	19798	82	36	34	59	4	36000
				F3					
1	484625	35429	6567	70	32	28	15	2	36000
2	483929	29295	6321	53	26	24	15	2	36000
3	485865	32728	4496	61	28	28	8	1	36000
4	484912	18354	19025	60	17	16	60	1	36000
5	485214	11812	23650	45	11	10	75	2	36000
				F4					
1	483692	7468	12035	43	8	8	38	0	36000
2	482359	35494	4362	36	15	14	11	2	36000
3	485126	14645	8077	50	12	11	24	0	36000
4	484725	1833	15355	31	1	1	49	0	36000
5	484730	32734	1700	60	18	16	1	1	36000
				F5					
1	498559	1845	8924	39	1	1	28	0	36000
2	498429	29672	826	40	11	11	0	0	36000
3	496303	3072	6203	32	2	2	19	0	36000
4	495801	23267	976	33	5	5	0	0	36000
5	495644	26913	886	36	7	7	0	0	36000