UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Revisitando o Método de Ranking de Pontos Extremos para o Problema da Mochila Linear

Julio Cesar Mosquera Gutierres

Itajubá, dezembro de 2015

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Julio Cesar Mosquera Gutierres

Revisitando o Método de Ranking de Pontos Extremos para o Problema da Mochila Linear

Dissertaçãosubmetida ao Programa de Pós-Graduação em Engenharia de Produção como parte dos requisitos para obtenção do Título de Mestre em Ciências em Engenharia de Produção.

Área de Concentração: Otimização

Orientador: Prof. Dr. Rafael Coradi Leme

Dezembro de 2015

Itajubá

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Julio Cesar Mosquera Gutierres

Revisitando o Método de Ranking de Pontos Extremos para o Problema da Mochila Linear

Dissertaçãoaprovada por banca examinadora em 11 de dezembro de 2015, conferindo ao autor o título de Mestre em Ciências em Engenharia de Produção.

Banca Examinadora:
Prof. Dr. Rafael Coradi Leme (Orientador)
Prof. Dr. Heitor Silvério Lopes
Prof. Dr. Antonio Carlos Zambroni de Souza

Itajubá
2015

Especialmente para

meus pais Julio e Ligia, pelo apoio constante sem importar a distância; Meus irmãos: Juan José, María Caridad e María Paz, pelas constantes palavras de motivação; Minha avó Eduviges, pela preocupação que sempre teve por mim; E para Pamela por nunca me deixar desistir a nada.

AGRADECIMENTOS

Quero iniciar agradecendo ao meu orientador Prof. Dr. Rafael Coradi Leme por todos os conhecimentos ensinados e por sempre ficar pendente da pesquisa e ter feito que o resultado dessa dissertação seja ótimo. Além de ter se transformado em um bom amigo.

Agradecer também aos pesquisadores Heitor Silvério Lopes e Antonio Carlos Zambroni de Souza, por terem aceitado fazer parte da banca avaliadora, o que acrescentará, e muito, o conteúdo dessa dissertação.

Quero fazer um agradecimento a todos os professores da UNIFEI que ajudaram a enriquecer os meus conhecimentos.

É importante fazer um agradecimento especial à FAPEMIG por ter fornecido a bolsa que me permitiu fazer essa pesquisa.

Não posso deixar de agradecer aos amigos/amigas que fiz durante esses mais de dois anos, especialmente a: Patricia, Taynara, Elisa, Diogo, David e Harlenn por sempre terem mostrado boa vontade na hora de me ajudar com problemas.

Quero agradecer também a Guilherme Augusto Barucke Marcondes, por sempre me salvar quando tinha algum problema com o documento da dissertação.

Um agradecimento especial a todos os membros da República V8: Marcelo, LuÃs Gustavo (Léo), José Eduardo, Edgar, Lucas (Vinte), Luis Eduardo (Tropica) e Francisca (Chiquinha) por terem sido minha família no tempo que morei em Itajubá.

E Ã© impossível não agradecer aos membros do F-O-C: David, Harlenn, Ibrahim e Rashid, por terem me convidado a fazer parte dessa grande parceria.

RESUMO

O problema da mochila linear visa encontrar um subconjunto de itens que otimize uma função objetivo sem exceder uma capacidade de mochila dada. É um dos problemas mais estudados em otimização combinatória e que nas últimas décadas vem sendo muito utilizado nas áreas de produção e administração. Na literatura existem vários métodos para resolver efetivamente esse problema. Porém, o método de ranking de pontos extremos busca a solução do problema analisando os vértices adjacentes ao vértice que resolve o problema relaxado para encontrar soluções alternativas. Quando foi apresentado em 1973, mostrou resultados interessantes, mas não tem sido mais utilizado pelos pesquisadores há aproximadamente 40 anos. Nessa dissertação será retomado o conceito de ranking de pontos extremos com o objetivo de determinar se foi acertada a decisão dos pesquisadores de não utilizar mais esse método. Para tal propósito o desempenho do ranking de pontos extremos foi comparado com o desempenho de dois métodos *branch-and-bound*. Um utiliza o método *simplex* para resolver os problemas, *branch-and-bound-simplex*(BBS), enquanto o segundo utiliza o método proposto por Danztig para achar a solução do problema da mochila contínuo, *branch-and-bound-Dantzig*(BBD). Os resultados obtidos mostraram que o método BBD é o melhor dos três tanto em eficácia como em rapidez, já o ranking de pontos extremos se apresentou competitivo ao BBD em problemas com até 500 variáveis, piorando rapidamente à medida que o tamanho dos problemas aumentava. Os métodos BBD e de ranking de pontos extremos obtiveram sempre as respostas ótimas. O BBS, dependendo das características de alguns problemas, não atingiu o ótimo, sendo por esse fato considerado como o pior de todos. O que faz concluir que sim, é justificado ter deixado de usar o método de ranking de pontos extremos para resolver o problema da mochila linear já que existem outros métodos com desempenho melhor.

ABSTRACT

The knapsack problem aims at finding a subset of items that maximize (minimize) an objective function without exceeding a given knapsack capacity. This is one of the most studied problems in combinatorial optimization and has been widely used in the areas of production and business and operation research in decades. There are various methods to solve effectively this problem in the literature. One of such methods is to the ranking of extreme points. In this approach, the problem is solved by analyzing the vertices adjacent to the vertex that optimize the relaxed problems as to find alternative solutions. When this method was proposed, in 1973, it showed promissing results. However, it has not been used by researchers for approximately 40 years. This dissertation will revisit the concept of ranking of extreme points in order to determine whether or not was valid the decision of researchers to abandon this method. For this purpose, the performance of ranking of extreme points was compared with the performance of two branch-and-bound approaches. One of them uses the simplex method to solve the problems, branch-and-bound-simplex (BBS); while the other solves the relaxed subproblems of branch-and-bound by the method proposed by Danztig, branch-and-bound-Dantzig (BBD). Results showed that the BBD method is the best of the three in both efficiency and speed. The ranking of extreme points was competitive to BBD in problems with up to 500 variables, worsening quickly as the size of the problems increased. The BBD and ranking of extreme points methods always obtained good solutions, while the BBS, depending on the characteristics of some problems, did not reach the optimum. Thus, BBS is considered as the worst of all methods. We conclude that it was justified to stop using the method of ranking of extreme points to solve the knapsack problem since there are other methods with better performance.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

O *problema da mochila clássico* (PM) pode ser definido da seguinte maneira: dados n elementos, cada elemento tem um lucro p e um peso w, e uma capacidade da mochila c; o objetivo é escolher um subconjunto de elementos que serão colocados na mochila para gerar o maior lucro/menor custo (para problemas de maximização/minimização) sem exceder c.

Este problema é um problema de decisão bem conhecido em que a probabilidade de uma seleção particular de alternativas pode ser avaliada por uma combinação linear de coeficientes para cada decisão contínua, inteira ou binária (ISLAM et al., 2009).

O PM é uma boa maneira de representar decisões que têm o foco sobre uma única restrição. Porém, na vida real não existem muitas decisões com uma única restrição mas, sim, muitas onde uma restrição tem mais importância do que as outras. Por isso o modelo de mochila pode ser mais amplamente aplicado do que na simples ideia de ter uma única restrição. Esta é a razão pela qual muitos dos métodos de solução dos problemas mais complexos empregam o problema da mochila como um subproblema. Por isso o problema da mochila vem sendo muito estudado nas últimas décadas (BARTHOLDI, 2008). Esses problemas mais complexos podem, às vezes, ser resolvidos mediante a resolução de uma sequência de PM. Uma abordagem geral para esses problemas é identificar a restrição que mais limita, ignorar as outras, resolver o PM, e ajustar a solução para satisfazer as restrições ignoradas (KELLERER et al., 2004).

Em um modelo de programação linear o resultado de um processo de decisão é avaliado por uma combinação linear dos valores associados a cada uma das decisões contínuas, inteiras ou binárias (dependendo das caraterísticas do problema) (KELLERER et al., 2004). Por isso o *problema da mochila* é um dos problemas mais estudados em otimização combinatória. Existem diversas variantes do problema (GAO et al., 2014), entre elas: Problema da mochila $0-1$, Problema da mochila quadrático, Problema da mochila de múltipla escolha, etc.[1]. O problema da mochila, além de ser fundamental é amplamente estudado nos modelos de pesquisa operacional que fornecem informações sobre as soluções dos problemas mais complexos de atribuição de capacidade de recursos discretos (JACKO, 2013).

Entre os diferentes tipos de problemas da mochila, o *problema da mochila linear binário* $(0-1)$ é, talvez, o mais importante e um dos mais estudados de programação discreta. A razão para tal interesse está basicamente ligada a três fatores: a) pode ser visto como o problema mais simples de programação linear inteira; b) aparece como um subproblema em muitos outros problemas mais complexos; c) pode representar uma gama muito grande de situações práticas

[1] Para a revisão dos diferentes tipos de problemas da mochila revisar (PISINGER, 1995)

(com várias aplicações na vida cotidiana) (MARQUES, 2000).

A diferença entre o *problema da mochila clássico* e o *problema da mochila* $0-1$ é que no *problema da mochila* $0-1$ somente pode ser escolhido um elemento de cada classe para ser colocado dentro da mochila (MARTELLO et al., 1999). Contudo, qualquer problema da mochila clássico pode ser reescrito como um problema da mochila 0-1.

Outro problema muito importante e estudado é o *Problema da Mochila Quadrático (PMQ)* que visa maximizar a função objetivo quadrática sujeita a uma restrição de mochila onde todos os coeficientes são considerados não negativos e todas as variáveis de decisão são binárias e/ou inteiras (CAPRARA et al., 1999) e (PISINGER et al., 2007). O PMQ é um problema no qual um item tem um lucro correspondente e um lucro adicional é resgatado se o item é selecionado junto a outro item (KELLERER et al., 2004).

É bem conhecido que os problemas da mochila são *NP-difícil*. Um problema é dito polinomial (pertencente à classe P) se existe um algoritmo conhecido capaz de resolver o mesmo, cuja complexidade, no pior dos casos, seja polinomial em relação ao tamanho do problema. Um problema pertence à *classe NP* quando é possível verificar se uma solução candidata é de fato uma solução do problema. Já o problema é dito *NP-difícil* se ele for tão difícil quanto qualquer problema em NP. Um problema *C* é *NP-difícil* se: *C* está em *NP* (problema de tempo polinomial); e se todo problema em *NP-difícil* é redutível para *C* em tempo polinomial (MATHUR; VENKATESHAN, 2007), (ZHAO; LI, 2013). Por isso para resolvê-los, os pesquisadores têm utilizado diversas técnicas de otimização das quais algumas serão abordadas nos próximos capítulos.

Em geral, os algoritmos de solução utilizados para resolver o problema da mochila (em todas suas formas) são baseados em técnicas de otimização combinatória e/ou inteira. Sendo assim, as técnicas de otimização combinatória se mostram como as melhores para resolver este tipo de problemas, pois exigem frequentemente resolver uma sequência de subproblemas ou relaxamentos de programação linear (PL)(mais simples) (ENGAU et al., 2010). Esses subproblemas normalmente surgem de novas formulações de programação inteira, podendo serem resolvidos através de diversos métodos de otimização que têm sido apresentados ao longo do tempo (MUNARI et al., 2011). Entre os mais utilizados estão: Métodos *simplex* e *simplex* revisado, método de pontos interiores, método de *branch-and-bound*, método de planos de corte, método de inteligência artificial.

Métodos *simplex* e *simplex* revisado, são utilizados para resolver problemas lineares. Analisam somente a fronteira do espaço de solução e fornecem uma resposta contínua. Método de pontos interiores, são geralmente utilizados para resolver problemas lineares de grande porte e, fornecem uma resposta contínua após analisar a parte central do espaço de solução. Método de

branch-and-bound é utilizado para resolver problemas inteiros ou mistos dividindo o problema original em subproblemas lineares mais fáceis de resolver até encontrar a resposta inteira ótima. Métodos de planos de corte permitem encontrar uma resposta contínua estreitando o espaço de solução do problema. Meta-heurísticas que permitem resolver problemas difíceis mais rapidamente, geralmente utilizando algoritmos evolutivos. A motivação por trás desta dissertação está em retomar uma técnica que não tem sido utilizada pelos pesquisadores faz 40 anos o **método de ranking de pontos extremos**. O método é enumerativo e examina os vértices adjacentes ao vértice que está minimizando a função objetivo a fim de encontrar soluções alternativas para o problema original. Quando Taha (1972) apresentou o método, este se mostrou muito efetivo para resolver o problema da mochila clássico. Segundo os resultados apresentados pelo autor, o ranking de pontos extremos foi mais eficiente do que o método de *branch-and-bound* tradicional. Taha também observou que quando se analisavam novos pontos no método de ranking de pontos extremos o tempo de convergência incrementava de uma maneira quase que linear, já no método de *branch-and-bound* o fazia de maneira exponencial. Desde então, o método de ranking de pontos extremos não tem mais sido explorada na literatura. No trabalho de Taha foram avaliados somente problemas com até 100 variáveis (neste trabalho serão analisados problemas com até 30.000 variáveis).

1.1 Objetivo

O objetivo dessa dissertação é determinar a eficiência do método de ranking de pontos extremos para resolver o problema da mochila linear, para poder responder a seguinte pergunta: justifica não utilizar mais o método de ranking de pontos extremos? Além de determinar se o método oferece melhores resultados do que o método de *branch-and-bound* (tradicional) para problemas com instâncias maiores.

1.2 Organização da Dissertação

Para alcançar o objetivo planejado, esta dissertação será dividida da seguinte forma:

No *capítulo 2* será apresentado o Problema da Mochila Linear, tomando em consideração a formulação contínua, inteira e binária.

No *capítulo 3* serão apresentadas as metodologias mais utilizadas na literatura para a resolução do problema da mochila. Além de analisar o método de ranking de pontos extremos de Taha para resolver o problema da mochila linear, bem como apresentar uma proposta para melhorar a sua performance na hora de procurar uma solução para problemas de instâncias maiores.

No *capítulo 4* serão apresentados os resultados obtidos e sua análise.

Finalmente o *capítulo 5* apresentará as conclusões da dissertação, além de indicar algumas recomendações.

2 O PROBLEMA DA MOCHILA (PM)

O "problema da mochila"aparece em muitas formas na economia, engenharia, e administração: qualquer local onde tenha que ser atribuído um único recurso escasso entre vários candidatos para esse recurso. Tem adquirido o nome de *"problema da mochila"* porque a experiência de embalar bagagem dá uma noção do problema: O que deve ser escolhido quando o espaço é limitado? (BARTHOLDI, 2008).

A maioria dos eventos são consequências de decisões, e é por isso que o *PM* vem sendo muito estudado nas últimas décadas já que tem várias aplicações em diversas áreas (ZHONG; YOUNG, 2010). Encontrar o valor ótimo pode ser difícil, pois o conjunto de opções disponíveis pode ser extremamente grande e/ou não explicitamente conhecido. Este problema é bem conhecido como uma decisão onde a escolha particular de alternativas pode ser avaliada por uma combinação linear de coeficientes para cada decisão contínua, inteira ou binária. Uma decisão pode incluir um item ou excluir o item da seleção (ISLAM et al., 2009).

2.1 Problema da Mochila Linear (PML)

Em um modelo de decisão linear o resultado de um processo de decisão completa é avaliado por uma combinação linear dos valores associados a cada uma das decisões binárias (inteiras ou contínuas) (KELLERER et al., 2004). Por isso é bem conhecido que o problema da mochila linear é *NP-difícil* (MATHUR; VENKATESHAN, 2007), (ZHAO; LI, 2013). Segundo Zhao e Li (2013) o PML pode ser facilmente entendido da seguinte forma: Dados n itens e uma mochila com capacidade c, cada item tem um peso w_i e um valor p_i, o objetivo é encontrar um subconjunto S de todos os itens de forma que o peso total dos itens em S é no máximo c, e o valor total é o máximo. Já Islam et al. (2009) indicam que para um problema de minimização dados n itens com um custo p_i e um peso w_i e uma demanda D, o objetivo é encontrar um subconjunto S de itens para os quais o tamanho total seja pelo menos a demanda D e que o custo total de todos os itens seja o mínimo possível.

A estrutura comum do PML é sintetizada da seguinte maneira:

1. Depara-se com um número de decisões sim/não.
2. As decisões são independentes, exceto que cada decisão "sim"consome uma quantidade conhecida de um único recurso escasso comum.
3. Nenhum dado mudará durante o tempo no qual as decisões serão tomadas.

Baseados na interpretação prévia, o PML *"contínuo"* no qual busca-se minimizar a função objetivo, pode ser formulado da seguinte maneira:

$$\begin{aligned} \text{maximizar} \quad & \sum_{j=1}^{n} p_j x_j \\ \text{sujeito a} \quad & \sum_{j=1}^{n} w_j x_j \leq c \\ & x_j \geq 0 \end{aligned} \tag{2.1}$$

Kellerer et al. (2004) e Pisinger (2005) indicam que o PML é um simples modelo de programação inteira com uma restrição única e com coeficientes positivos. Desta maneira os PML frequentemente são resolvidos por um relaxamento de vários programas inteiros (PISINGER, 1995). Por isso geralmente as quantidades dos itens escolhidos precisam ser inteiras e o PML pode ser reformulado como um problema inteiro, como se mostra a seguir:

$$\begin{aligned} \text{maximizar} \quad & \sum_{j=1}^{n} p_j x_j \\ \text{sujeito a} \quad & \sum_{j=1}^{n} w_j x_j \leq c \\ & x_j \in \mathbb{Z}, \forall_j e w_j, D \in \mathbb{Z}^*, \forall_j \end{aligned} \tag{2.2}$$

Para Bartholdi (2008), o PML é uma boa maneira de representar decisões que têm o foco sobre uma única restrição. Porém, na vida real, não existem muitas decisões com uma única restrição, mas muitas onde uma restrição tem maior importância do que as outras. Por isso o modelo de mochila pode ser amplamente aplicado do que na simples ideia de ter uma única restrição, sendo essa a razão de que muitos dos métodos de solução dos problemas mais complexos empregam o PML como um subproblema. Esses problemas mais complexos podem, às vezes, ser resolvidos mediante à resolução de uma sequência de PML. Uma abordagem geral para esses problemas é identificar a restrição que mais limita, ignorar as outras, resolver o PML, e ajustar a solução para satisfazer as restrições ignoradas (KELLERER et al., 2004).

O PML é estático e determinístico pois todos os itens são considerados em um ponto no tempo, e seus tamanhos e valores são conhecidos a priori (KLEYWEGT; PAPASTAVROU, 1998), motivo pelo qual é bem conhecido que pode ser resolvido exatamente por meio de programação dinâmica: decompõe-se um problema complexo em subproblemas mais simples; em um tempo *O(nc)* onde *n* é o número de itens e *c* é a capacidade da mochila (FOMENI; LETCHFORD,

2013). O PML varia quando novas restrições são adicionadas ou as restrições existentes são modificadas (ZHONG; YOUNG, 2010).

Marques (2000) indica que o PML, pelo fato de ser considerado um problema *NP-difícil*, tem sido exaustivamente estudado nas últimas décadas e vários algoritmos exatos de solução podem ser encontrados na literatura, o que fez considerar ao PML como um dos problemas *"fáceis"* dentre os problemas *NP-difícil*. Mas Pisinger (2005) faz uma revisão das características de todas as soluções exatas recentes e mostra que o PML ainda continua sendo difícil de se resolver com esses algoritmos para uma variedade de novos problemas testados. A maioria dos algoritmos utilizados para tratar o PML encontrados na literatura utilizam as seguintes técnicas: *branch-and-bound*, programação dinâmica, relaxamento do estado espacial e pré-processamento[1]. Existem outras técnicas que também ajudam a resolver o PML, por exemplo: balanceamento (PISINGER, 1999), particionamento (HOROWITZ; SAHNI, 1974), o que indica que existem várias técnicas que resolvem efetivamente o PM (MARTELLO et al., 2000).

Existem casos em que as decisões precisam da escolha de uma, entre duas alternativas, onde $x = 1$ indica que a alternativa analisada foi escolhida para ser colocada dentro da mochila. Por outro lado, se $x = 0$, indica que a alternativa analisada foi rejeitada como uma opção para ser colocada na mochila. Quando isso ocorre o problema é conhecido como: *Problema da Mochila Binário* ou também conhecido como *Problema da Mochila 0-1 (PM 0-1)* (pode ser linear ou quadrático). Esse problema é considerado como um dos problemas clássicos em otimização combinatória (GORSKI et al., 2012) e pode ser formulado como se segue:

$$\begin{aligned} \text{maximizar} \quad & \sum_{j=1}^{n} p_j x_j \\ \text{sujeito a} \quad & \sum_{j=1}^{n} w_j x_j \leq c \\ & x_j \in \{0,1\} \quad j \in \{1,\ldots,n\} \end{aligned} \tag{2.3}$$

Martello et al. (1999) indicam que nos casos em que existe pouca ou nenhuma correlação entre o custo c e o peso s de cada item i dos problemas podem ser facilmente resolvidos com otimalidade, inclusive para grandes valores de n. Por outro lado, nos casos onde existe uma forte correlação, bem como casos envolvendo quantidades muito grandes de valor p_i e de peso w_i, pode ser muito difícil encontrar a solução ótima.

Tanto o PML quanto o PML 0-1 são usualmente utilizados para modelar situações da indústria ou decisões financeiras (GAIVORONSKI et al., 2011). Exemplos podem ser encontrados

[1] para entender melhor estas técnicas e alguns dos algoritmos que as utilizam para resolver o PML revisar (PISINGER, 1995).

em problemas de investimento, programação de dieta para pessoas, problemas de corte de estoque, problemas de orçamento (PISINGER, 1995), problemas na indústria do transporte, programação de lotes de produção, área de vendas, investimentos, etc., (KLEYWEGT; PAPASTAVROU, 1998). O PML e o PML 0-1 também são muito utilizados em problemas de carregamento de contêineres em navios de carga; problemas de otimização de portfólio são comumente abordados como problemas de mochila onde o orçamento é a restrição mais importante; inclusive problemas de programação de produção podem ser resolvidos por um enfoque do problema da mochila, especialmente quando as máquinas representam um grande investimento e elas possuem o tempo como o recurso escasso mais importante (LINHARES, 2009). Nessa dissertação a análise de algoritmos de solução será aplicada exclusivamente para o PML 0-1.

3 MÉTODOS DE SOLUÇÃO PARA O PROBLEMA DA MOCHILA

Os algoritmos de solução utilizados para resolver o PM são, em geral, baseados em técnicas de otimização combinatória e/ou inteira. Estas técnicas são as melhores para resolver este tipo de problemas, já que exige frequentemente resolver uma sequência de subproblemas ou relaxamentos de programação linear (PL) sucessivos (ENGAU et al., 2010). Esses subproblemas surgem de novas formulações de programação inteira e podem ser resolvidos com diversos métodos de otimização que têm sido apresentadas ao longo do tempo (MUNARI et al., 2011). Neste capítulo será realizada uma pequena introdução de alguns dos métodos mais utilizados para resolver o PML 0-1.

3.1 Programação Dinâmica (PD)

Na maioria dos problemas práticos, as decisões têm que ser feitas sequencialmente em diferentes pontos no tempo, em diferentes pontos no espaço e em diferentes níveis, por um componente, um subsistema ou por um sistema. A PD é uma abordagem de otimização que decompõe um problema complexo em uma sequência de subproblemas mais simples. Isso pode ser considerado como um processo recursivo que interpreta um problema de otimização como um processo de decisão multiestágio. Associado a cada estágio existem estados do processo, cada estado é uma maneira de descrever a solução do subproblema mesmo que contenha suficiente informação para fazer futuras decisões sobre o problema original (RONG; FIGUEIRA, 2014). Também é sabido que a PD é uma técnica matemática bem estudada na otimização de problemas de decisão multiestágio, que quando aplicada, representa ou decompõe um problema de decisão multiestágio como uma sequência de problemas de decisão de estágio único. Desse modo, um problema de $N-$variáveis é representado como uma sequência de problemas de $N-$variáveis únicas que são resolvidas sucessivamente e, na maioria dos casos. Esses N subproblemas são mais fáceis de resolver do que o problema original (RAO, 2009).

Segundo Dasgupta et al. (2006), a PD é um poderoso paradigma algorítmico no qual um problema é resolvido mediante a identificação de uma coleção de subproblemas e combatendo eles um de cada vez, os menores primeiro, para depois utilizar as respostas desses problemas como ajuda na resolução dos maiores, se repetindo até todos os problemas sejam resolvidos. Já para Huang e Sagae (2010) a principal observação para a PD é misturar *"estados equivalentes"* na mesma etapa se eles têm os mesmos valores de recurso, já que eles vão ter os mesmos

custos. Zhang et al. (2011) utilizaram técnicas de PD para desenvolver um método computacional simples para buscar um modelo de ramificação-específica ótimo em um tempo prático; Kraft e Steffensen (2013) utilizaram a PD para resolver o problema de portfólio restrito, já que a PD dá fácil acesso à função de valor e os controles do problema e, portanto, desempenha um papel importante para a resolução de problemas de controle estocásticos em finanças. Rong e Figueira (2014) utilizaram algoritmos de PD para encontrar a fronteira de Pareto exata para o problema da mochila inteira bi-objetivo. Schuetz e Kolisch (2012) utilizaram uma aproximação da PD para determinar um modelo que permita achar soluções ótimas no problema de alocação de capacidade na indústria de serviços.

O PML 0-1 pode ser resolvido através de PD da seguinte maneira:

Algoritmo 3.1: Programação Dinâmica

Dados: p, w, c, n

Resultado: Solução do problema

Inicio; $k = 0$;
para $k = 0$ *até* $k = n$ **faça**
    $f_k(c) = Max f_{k-1}(c), p_k + f_{k-1}(c - w_k)$
fim
se $f_k(c) = f_{k-1}(c)$ **então**
    $P_k(c) = 0$;
senão
    $P_k(c) = 1$;
fim
se $P_k(c) = 0$ **então**
    $x_k = 0$;
    $f_k(c) = f_{k-1}(c - w_k)$;
senão
    $x_k = 1$;
    $f_k(c) = p_k + f_{k-1}(c - w_k)$;
fim
retorna x

Onde p_k representa o valor do item k, w_k representa o peso do item k, c representa a capacidade da mochila. A seguir será resolvido um exemplo para ilustrar a maneira que a PD atua. Seja:

$$\begin{aligned} \text{maximizar} \quad & 3x_1 + x_2 + 2x_3 \\ \text{sujeito a} \quad & 2x_1 + 4x_2 + 3x_3 \leq 8 \\ & x \in \{0,1\} \end{aligned} \tag{3.1}$$

A Tabela 3.1 mostra a resolução usando PD.

Tabela 3.1 – Exemplo resolvido por PD

c	f_0	f_1	f_2	f_3	P_1	P_2	P_3
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	3	3	3	1	0	0
3	0	3	3	3	1	0	0
4	0	3	3	3	1	0	0
5	0	3	3	5	1	0	1
6	0	3	4	5	1	1	1
7	0	3	4	5	1	1	1
8	0	3	4	5	1	1	1

$$\begin{aligned} &P_3(8) = 1 \rightarrow x_3^* = 1 \\ &P_2(8-3) = P_2(5) = 0 \rightarrow x_2^* = 0 \\ &P-1(5) = 1 \rightarrow x_1^* = 1 \end{aligned}$$

A solução ótima é $x^*(1,0,1)$ e $f(x^*) = 5$.

3.2 Método de Dantzig para o PML

O método apresentado por Dantzig (1957) permite encontrar um bom limite superior (inferior) inicial e boas soluções iniciais, porém não necessariamente soluções ótimas. Além disso, esse método também produz rapidamente a solução do PML contínuo.

O método de Dantzig indica que, todos os valores dos pesos a_j e dos custos c_j são positivos e, assumindo que os índices das variáveis x_j são arranjadas de modo que $c_1/a_1 \geq c_2/a_2 \geq \ldots \geq c_n/a_n$, é possível definir p como o menor inteiro ($0 \leq p \leq n$) tal que $\sum_{j \leq p} a_j \geq b$ (p indica o índice da variável limite que pode ser selecionada para ser colocada dentro da mochila). Dessa maneira a solução fracional ótima para o PM é dada como:

$$x_j = \begin{cases} 1 & \text{se } j < p \\ (b - \sum_{j<k} a_k)/a_p & \text{se } j = p \\ 0 & \text{se } j > p \end{cases} \tag{3.2}$$

Se não existe nenhuma variável p, então todas as variáveis $x_j = 1$. E se $x_p = 0$, então a solução inteira resultante é ótima para o PM.

O método de Dantzig enxerga o PM como um programa linear com variáveis limitadas, $0 \leq x_j \leq 1, j = 1,2,\ldots n$ onde uma única variável aparece na base enquanto as variáveis restantes são não base em nível 0 ou nível 1. Deixando x_r ser a variável básica, os custos reduzidos $z_j - c_j$ para $x_j, j \neq r$ devem satisfazer as seguintes condições para alcançar a solução ótima:

$$z_j - c_j = \begin{cases} (-c_r)(1/a_r)/a_j - (-c_j) & \text{se } x_j = 0 \\ -(-c_r)(1/a_r)/a_j - (-c_j) & \text{se } x_j = 1 \end{cases} \tag{3.3}$$

Isso significa que $c_r/a_r \geq c_j/a_j$ para toda j tal que, $x_j = 0$ e não básica pelo limite inferior, e $c_r/a_r \leq c_j/a_j$ para todo j tal que, $x_j = 1$ e não básica pelo limite superior. A escolha de $r = p$ resulta na viabilidade da restrição. Isso resulta em uma aproximação que permite obter uma solução inteira inicial e, consequentemente, um bom limite superior inicial. A ideia da aproximação é adicionar a maior quantidade de novas variáveis que for possível ao nível 1 provando que a viabilidade é mantida enquanto vai chegando até o menor limite superior. Isso faz com que mais variáveis se mostrem atrativas para serem elevadas ao nível um antes de ocorrer a inviabilidade. A solução aproximada é dada por

$$x_j{}^* = \begin{cases} 1 & \text{se } j < p \\ 0 & \text{caso contrário} \end{cases} \tag{3.4}$$

E o limite superior inicial é definido por

$$\bar{z} = \sum_{j=1}^{n} (-c_j) x_j{}^* \tag{3.5}$$

A seguir, a Figura 3.1 mostra o exemplo 3.2 sendo resolvido através do método proposto por Dantzig.

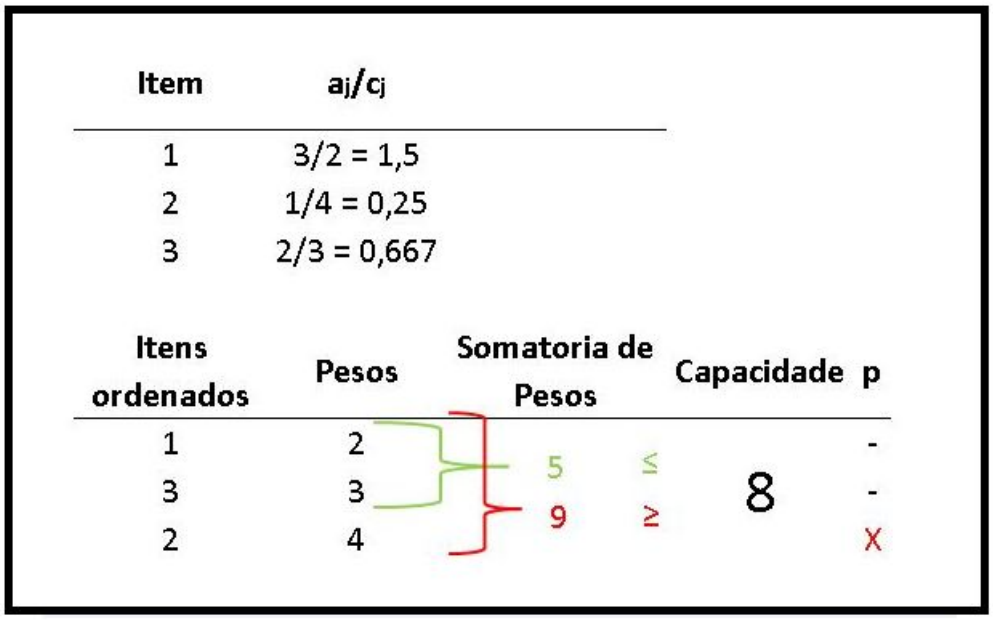

Figura 3.1 – Exemplo 3.2 resolvido pelo método de Dantzig

Coincidentemente a solução ótima é $x^*(1,0,1)$ e $f(x^*) = 5$ igual que na PD.

3.3 Planos de Corte (PC)

Os planos de corte são uma parte integral da programação inteira. Por anos, vários tipos de planos de corte para problemas inteiros e misto-inteiros têm sido apresentados e estudados na literatura (DUNKEL; SCHULZ, 2012).

Em geometria, uma equação com duas variáveis é chamada de *plano*, uma equação em n variáveis é chamada de *hiperplano* e uma restrição de desigualdade com n variáveis é chamada de *semi-espaço*. O termo *plano de corte* é, às vezes, usado para designar uma restrição de igualdade ou de desigualdade que pode cortar uma parte fracionária de uma região factível de um problema de PL, sem excluir nenhuma solução inteira factível. Os métodos de planos de corte restringem o problema relaxado de PL para aproximá-lo a um problema de programação inteira (PI) (MITCHELL, 2002).

Na aproximação pelo plano de corte, um ou mais cortes são adicionados à tabela simplex do problema PL atual que por sua vez são resolvidos por um novo problema de PL ótimo, este processo se repete até satisfazer os requerimentos de integralidade prescritos (CHEN et al., 2010). O procedimento para resolver iterativamente programas lineares e adicionar restrições violadas é comumente chamado como algoritmo de plano de corte (FERREIRA et al., 1996).

A ideia principal dos algoritmos de planos de corte é eliminar a necessidade das restrições de integridade através da introdução de mais e novas desigualdades. Também se chama

plano de corte uma restrição linear adicionada a um problema inteiro para remover uma solução linear, uma vez que a solução do problema linear é cortada a partir da região viável (JUKNA; SCHNITGER, 2012). As técnicas de planos de corte permitem que o modelo utilize cálculos realizados por iterações anteriores para resolver o próximo problema inteiro (ALVES; CLÃMACO, 1999). A Figura 3.2 indica como o plano de corte faz o corte no espaço de solução.

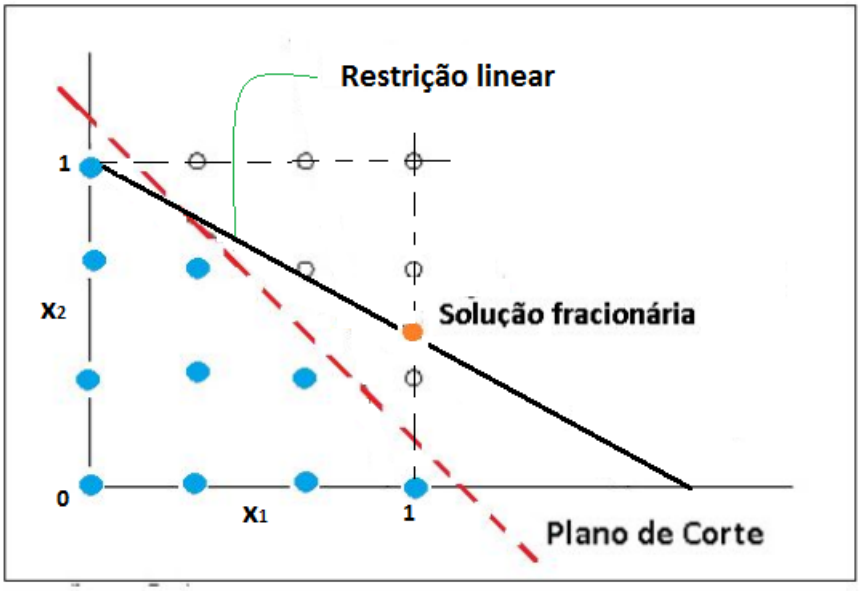

Figura 3.2 – Um corte da região de solução do problema original

Nota-se que a região viável para o relaxamento linear será um politopo convexo e a solução do problema linear será o vértice desse politopo (adicionando um plano de corte *"recortamos"* este vértice)(CUSSENS, 2012).

A questão fundamental é como encontrar rapidamente bons planos de corte. Para isso Achterberg (2007) indica três critérios de qualidade para escolher uma coleção de planos de corte: a eficácia dos cortes (a distância dos seus hiperplanos correspondentes à solução de PL atual); a ortogonalidade dos cortes com respeito aos outros; e o paralelismo dos cortes com respeito à função objetivo.

Com tudo que foi discutido anteriormente, pode-se concluir que o método de planos de corte é adequado para resolver problemas lineares inteiros através de derivadas das restrições até que o problema seja reduzido a um programa linear geral (DANTCHEV; MARTIN, 2012).

Agora o problema 3.2 será resolvido utilizando o método de planos de corte, para o qual

será utilizado o algoritmo a seguir:

Algoritmo 3.2: Plano de Corte

Dados: p, W, c

Resultado: Solução do problema

```
Inicio;
k = 0 ;
cond = 0;
enquanto cond = 0 faça
    [P_relax, x] = p'_x, W_x ≤ c, 0 ≤ x ≤ 1;
    se x = inteiro então
        cond = 1;
        parar;
    senão
        R = R + 1 [P_relax, x] = p'_x, W_x ≤ c, R_r, 0 ≤ x ≤ 1;
    fim
fim
retorna x
```

Onde p equivale a matriz dos valores dos itens, W representa a matriz dos pesos dos itens, c é a capacidade da mochila, R_r representa a restrição que deve ser acrescentada. Ao resolver o problema relaxado, a resposta obtida é $x(1, 0.75, 1)$ com $z = 5,75$. Como x_2 não é inteiro, isso indica que podem ser selecionados unicamente dois itens para ser colocados na mochila. O seguinte plano de corte é gerado

$$x_1 + x_2 + x_3 \leq 2 \tag{3.6}$$

Adiciona-se o plano de corte ao problema relaxado e resolve-se o novo problema gerado:

$$\begin{array}{ll} \text{maximizar} & 3x_1 + x_2 + 2x_3 \\ \text{sujeito a} & 2x_1 + 4x_2 + 3x_3 \leq 8 \\ & x_1 + x_2 + x_3 \leq 2 \\ & 0 \leq x \leq 1 \end{array} \tag{3.7}$$

A solução encontrada é $x(1, 0, 1)$ com $z = 5$ que é a solução ótima para o problema original.

3.4 Método de Branch-and-Bound (B&B)

O método de *branch-and-bound* (ramificação e poda) é uma abordagem de uso geral capaz de resolver Problemas Inteiros (PI) puros, mistos (somente para a parte inteira), binários (CHEN et al., 2010) e problemas de programação não- linear (RAO, 2009).

Kellerer et al. (2004) indicam que o método de B&B está baseado em dois princípios fundamentais: *branching* (ramificar), que divide o problema original em subproblemas menores, repetindo o processo até cada subproblema conter somente uma solução simples factível; e *bounding* (limitar ou podar), que deriva tanto o limite superior quanto o limite inferior para diminuir o espaço de solução. Dessa forma toda resposta encontrada que não estiver dentro dos novos espaços achados será retirada (podada) da árvore.

Outra estratégia para utilizar o método de B&B, similar à indicada por Kellerer et al. (2004), é a estratégia conhecida como "dividir e conquistar"que para Chen et al. (2010) basicamente divide o problema dado em uma série de subproblemas mais fáceis de se resolver que são sistematicamente gerados e resolvidos (conquistados).

Considerando que o método mais simples para resolver um problema de otimização inteira envolve enumerar todos os pontos inteiros, eliminando os inviáveis. O método de B&B pode ser considerado como um método de enumeração refinado no qual a maioria dos pontos inteiros não promissórios são descartados sem testar (RAO, 2009).

Uma forma comum de representar o método é mediante uma árvore conhecida como a árvore branch-and-bound que é uma árvore de enumeração especializada para fazer o seguimento de como os subproblemas de programação linear são resolvidos. A árvore se desenha de cima para abaixo com o nó raiz (representa o PL relaxado) no topo. Para escolher o próximo nó a ser analisado existem duas opções: *depth-first*, resolve-se primeiro o subproblema mais recente gerado; e o *best-bound-first*, minimiza o número total de nós avaliados ramificando o nó ativo com melhor valor objetivo (CHEN et al., 2010). Para entender melhor o método B&B, a continuação será indicado mostrado o algoritmo apresentado por (CHEN et al., 2010) que facilitará seu entendimento. A seguinte notação ajudara no entendimento do algoritmo.

$S =$ problema dado de PI
$S_{LP} =$ relaxamento linear de S
$x_{LP} =$ solução de $S_(LP)$
$\bar{z} =$ menor (melhor) limite superior em z^* do problema dado
$\underline{z} =$ maior (melhor) limite superior em z^* do problema dado
$S^k =$ subproblema k do problema S
$S^k_{LP} =$ relaxamento linear do subproblema k

z^k = valor objetivo ótimo de S^k

$\bar{z}^k$ = melhor (menor) limite superior do subproblema S^k (mostrado sobre o nó k)

$\underline{z}^k$ = melhor (maior) limite inferior do subproblema S^k (mostrado sobre o nó k)

x^k_{LP} = solução ótima do subproblema S^k_{LP}

$\bar{x}_j$ = valor não inteiro da variável inteira x_j (valor numérico atual de x_j)

$\lfloor a \rfloor$ = o maior inteiro $\leq$ a (ou arredondado para abaixo de a)

$\lceil a \rceil$ = o maior inteiro $\geq$ a (ou arredondado para abaixo de a)

L_n = lista de nós ativos

O algoritmo será descrito a seguir:

Algoritmo 3.3: Branch-and-Bound

Dados: S

Resultado: Solução do problema

Inicio;
$k = 0$; $cond1 = 0$; $cond2 = 0$;
enquanto $cond1 = 0$ **faça**
    $[f_{SL}, x_{LP}] =$ resolver (S);
    se $x_{LP} = inteira$ **então**
        $cond1 = 1$; parar;
    fim
    se $x_{LP} = não\ viável$ **então**
        $cond1 = 1$; parar;
    fim
    $\bar{z}^k = f_{SL}$; $\underline{z}^k = -\inf$;
    colocar S^k_{LP} em L_n;
    enquanto $cond2 = 0$ **faça**
        se $L_n = vazia$ **então**
            $cond2 = 1$; $cond1 = 1$; parar;
        senão
            escolher um S^k com S^k_{LP} por *deph-first* ou *best-bound-first*;
        fim
        $[f_{S^k_{LP}}, x^k_{LP}] =$ resolver (S^k_{LP});
        $\bar{z}^k = f_{S^k_{LP}}$; $x^* = x^k_{LP}$;
        se x^k_{LP} *ou* $\bar{z}^k = \underline{z}$ **então**
            podar nó;
        senão
            $\underline{z}^k = x^k_{LP}$;
            se $z^k > \underline{z}$ **então**
                $\underline{z} = \underline{z}^k$; podar nó;
                gerar $S^k_1 = S^k \bigcap \{y : y_j \leq \lfloor \bar{y}_j \rfloor\}$ e $S^k_s = S^k \bigcap \{y : y_j \geq \lceil \bar{y}_j \rceil\}$;
            fim
        fim
    fim
fim
retorna x^*

Uma parte importante no método B&B é a maneira como determinar qual o próximo nó a ser explorado. As estratégias mais comumente utilizadas são: *depth-first* e *best-bound-first* (CHEN et al., 2010).

A estratégia *depth-first* resolve primeiro o subproblema gerado mais recente. O objetivo dessa estratégia é obter rapidamente uma solução inteira viável cuja função objetivo tenha um valor z^k menor do que o limite inferior dado pelo problema inteiro e que pode ser utilizado para podar nós por otimalidade. Já a estratégia *best-bound-first* analisa o nó que possua o maior valor de z. O objetivo é minimizar o número total de nós avaliados na árvore de B&B. Essa estratégia demanda uma estrutura de dados explícita para armazenar os subproblemas considerados, e o espaço consumido pode ser exponencialmente grande.

A seguir será resolvido o exemplo 3.2 pelo método B&B. Não será utilizado nenhum critério de parada, ou seja, a árvore inteira será elaborada para melhor compreensão do método.

A árvore indica a quantidade total de nós que teriam que ser analisados se o método B&B não podasse nenhum deles enquanto procura a melhor solução. A árvore também mostra quais nós são viáveis e podem ser analisados (nós em cor azul) e os nós que não são viáveis (nós vermelhos) que consequentemente serão descartados pelo método sem ser analisados; além de mostrar o nó que contém a melhor solução.

Já que o método B&B pede para resolver o problema original como um problema linear relaxado, a alternativa mais comum para realizar isso é utilizar os métodos simplex e simplex revisado (esses métodos são abordados no apêndice da dissertação). No entanto, para o caso do PML 0-1 contínuo, o método proposto por Dantzig e discutido na seção 3.2 é uma alternativa bastante atrativa. O método de Dantzig é mais rápido e simples comparado com os métodos simplex e simplex revisado.

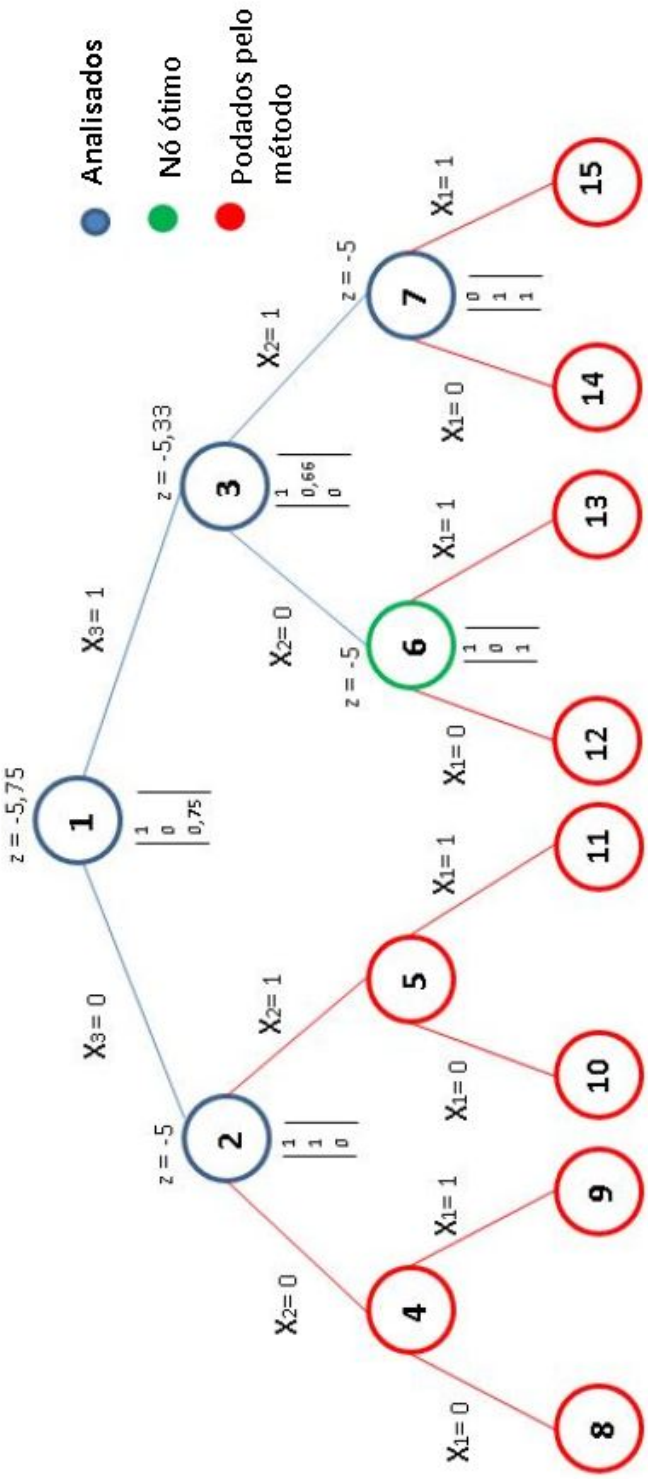

Figura 3.3 – Árvore completa para o exemplo 3.2

3.5 Ranking de pontos extremos (vértices)

Esta dissertação aborda o conceito de ranking de vértices para problemas binários apresentado no livro de Taha (1974) (baseado no trabalho original de (TAHA, 1972)), o qual é utilizado

para resolver o problema da mochila linear binário.

A primeira abordagem enumerativa foi desenvolvida por (MURTY, 1968), que elaborou uma ideia criativa para enumerar sistematicamente os pontos extremos do espaço de solução E até que o mínimo global da função seja encontrado. Essa ideia é chamada como *ranking de pontos extremos*(vértices). A ideia do método de ranking de pontos extremos é a de examinar os vértices adjacentes ao vértice que está minimizando a função objetivo a fim de encontrar soluções alternativas.

Os passos do ranking de pontos extremos podem ser executados de várias maneiras, mas a forma mais comum assume que, através de algum método avaliado é possível encontrar uma função linear $l : \mathbf{R}^n \rightarrow \mathbf{R}$ tal que $l(x) \leq f(x)$ para todo $x \in E$. Dada a função l, o passo inicial começa definindo qualquer solução ótima x^0 (nos pontos extremos) para o programa linear que visa minimizar l através de E. Então $l(x) \leq v \leq UB$, onde v é o valor ótimo do problema original, e $UB = f(x^0)$ é o limite superior de v. O vetor x^0 é então armazenado para ser atribuído a $\bar{y} \in \mathbf{R}^n$. O vetor $\bar{x}$ sempre representa o melhor ponto extremo encontrado até então no processo de ranking e é chamada de resposta incumbente (HORST; PARDALOS, 1995).

É sabido que os problemas lineares zero-um (binários) têm uma única propriedade que indica que a solução ótima para esses problemas ocorre em um dos pontos extremos do espaço de solução. A demonstração para essa afirmação pode ser encontrada em (TAHA, 1974) e está transcrita no apêndice.

O procedimento utilizado por Taha é inspirado no trabalho de Murty (1968), no qual foi utilizado o conceito de ranking de vértices como uma extensão do método simplex para desenvolver um algoritmo ótimo que serve para resolver diferentes modelos do problema de carga fixa na área de transporte. Comparado com o método B&B tradicional o método apresentado por Murty determina a sequência dos próximos vértices mediante uma simples mudança nas bases. Isto permite evitar a análise de ótimos alternativos que não estiverem nos pontos extremos do espaço de solução reduzindo, dessa maneira, o tempo computacional requerido. Outros procedimentos e algoritmos de ranking de pontos extremos são bem discutidos em (TAHA, 1972).

O algoritmo apresentado por Taha é simples. Para entender melhor o algoritmo a seguinte notação sera utilizada:

- x^i. Ponto extremo na posição i.
- z. Valor da função objetivo.
- Δ. É o menor múltiplo comum de todos os p_j.

O algoritmo é apresentado a seguir:

Algoritmo 3.4: Ranking de Pontos Extremos

Dados: p, w, c, n

Resultado: Solução do problema

```
1  Inicio;
2  [z,x] = resolver {p'x, wx ≤ c, 0 ≤ x ≤ 1};
3  se x = binario então
4  |   para;
5  senão
6  |   i = 1;
7  fim
8  para k = 0 até k = n faça
9  |   x_i = pontoextremoquegeraomelhorz;
10 |   z_i = limiteinferior;
11 |   se i = n então
12 |   |   parar, resposta não viável;
13 |   senão
14 |   |   se z_i > z − Δ então
15 |   |   |   se e_i = binario então
16 |   |   |   |   e* = e_i;
17 |   |   |   |   z = z_i;
18 |   |   |   fim
19 |   |   fim
20 |   fim
21 fim
22 retorna x
```

Para que o método seja efetivo, Taha indica que o algoritmo depende de duas ideias básicas: *(i)* determinar um bom limite superior inicial, e *(ii)* determinar qual será o próximo ponto extremo a ser ranqueado e^i dado e^{i-1}.

A ideia para obter um bom limite superior inicial é a de estreitar o espaço de busca para poder, dessa maneira, obter melhores respostas. Já para determinar o próximo vértice a ser ranqueado se procura aquele ponto que proporcione em problemas de minimização o menor valor objetivo entre os pontos extremos adjacentes. Cada ponto adjacente escolhido será armazenado em ordem ascendente. Em outras palavras, o método de ranking olha todos os vértices adjacentes do politopo do espaço de resposta e guarda aqueles que geram um melhor valor objetivo do

que a solução incumbente, e coloca todos os vértices achados de forma ascendente. Na iteração seguinte o método repete o procedimento, mas desta vez deve-se evitar os vértices já visitados nas iterações prévias.

Taha indica que para utilizar o procedimento de ranking de vértices, uma variável inteira tem que ser substituída em termos de variáveis $0-1$ equivalentes. Por isso no problema da mochila com uma única restrição cada vértice deve necessariamente satisfazer pelo menos uma restrição binaria, o que indica que no procedimento de ranking. Todos os vértices adjacentes são numerados envolvendo um amplo campo de busca entre todos os vértices contidos no espaço de solução.

Um fato que deve ser considerado é quando as variáveis inteiras são substituídas em termos de variáveis $0-1$ equivalentes. É que isso incrementará o número de variáveis, só que esse incremento será unicamente um múltiplo de aproximadamente $\ln\prod_{j=1}^{n}(\mu_j-1)/\ln 2$, onde μ_j é o limite superior da variável j. Isso mostra que a razão de incremento das variáveis não é exponencial, o que permite uma resolução mais rápida. O método de ranking de pontos extremos tem sido muito utilizado em problemas de programação de dois níveis.

O método de ranking de pontos extremos, pode ser facilmente interpretado como uma árvore de nós integrados sem ciclo onde um ponto não pode ser criado mais de uma única vez. Cada ponto extremo é representado por um único nó. A maneira de percorrer a árvore é analisar os nós pelo valor da função objetivo (o melhor primeiro). O processo se repete até encontrar uma resposta inteira que satisfaça a restrição e que forneça o melhor valor objetivo para o problema original. Para criar os novos pontos, são analisadas as variáveis básicas e as variáveis não básicas. Cada nó possui uma única variável básica, as outras variáveis que não são consideradas como básicas são chamadas de variáveis não básicas. Se ao trocar uma variável não básica pela variável básica o novo conjunto de variáveis não básicas é igual a algum conjunto anterior, Então, o ponto não pode ser criado a partir do nó analisado, caso contrário o novo ponto pode ser criado.

Uma vez que o primeiro passo do ranking de pontos extremos pede para resolver o problema original de forma relaxada, o método de Dantzig (**??**) será utilizado para tal propósito. Outras alternativas para resolver o problema relaxado são os métodos simplex e simplex revisado.

A Figura 3.4 mostra a árvore do ranking de todos os pontos extremos para o exemplo 3.2. A árvore indica a quantidade total de nós que teriam que ser analisados pelo método se todos os pontos extremos forem viáveis. A árvore também mostra quais nós são viáveis e podem ser analisados (nós em cor azul) e os nós que não são viáveis (nós vermelhos) que consequentemente serão descartados pelo método sem ser analisados; além de mostrar o nó que contém a melhor solução (nó verde). As variáveis base B e não base N para cada nó (viável) também são

apresentadas.

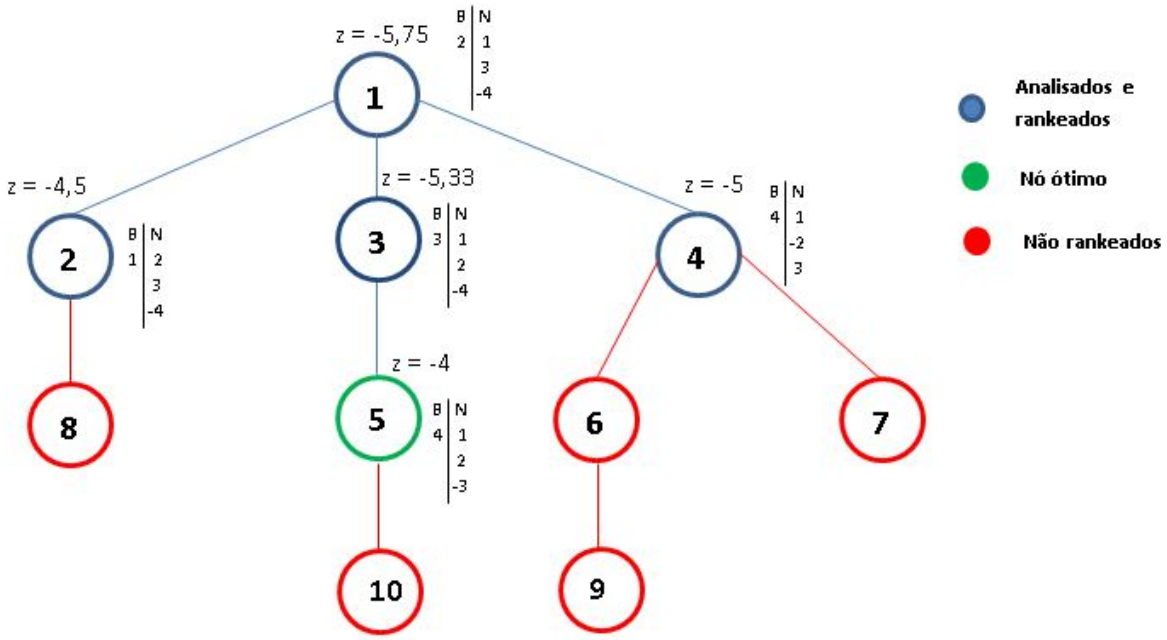

Figura 3.4 – Ranking completo para o exemplo 3.2

Uma vantagem de utilizar o método de ranking de pontos extremos comparado com o método B&B é que na árvore do ranking de pontos extremos o número de nós a ser gerados geralmente poderá ser menor do que no B&B.

3.6 Meta Heurísticas

Os métodos apresentados anteriormente são conhecidos como métodos exatos, porque eles fornecem uma resposta exata do problema que está sendo resolvido. Porém, existem outros métodos que também são utilizados para trabalhar o PML 0-1, os conhecidos métodos inexatos. Os métodos inexatos ou conhecidos também como meta-heurísticas são geralmente aplicados em problemas para os quais não existe um algoritmo que fornece uma solução exata do problema. As meta-heurísticas são comumente utilizadas como métodos de aproximação que utilizam a combinação de conhecimentos históricos dos resultados anteriores para realizar uma busca no espaço de solução para evitar paradas prematuras em ótimos locais. À medida que o tamanho dos problemas aumenta as meta-heurísticas perdem qualidade no seu desempenho, a seguir serão apresentadas algumas meta-heurísticas que tem sido utilizadas para trabalhar o PML 0-1.

3.6.1 Inteligência Artificial (IA)

A habilidade para aprender é geralmente considerada como um atributo fundamental da inteligência (GLOVER, 1986).

A IA começou na década de 1950 como um inquérito sobre a natureza da inteligência. Usando computadores como uma ferramenta revolucionária para simular de fato a utilização de inteligência. Proporcionando um meio para examiná-la com maiores detalhes. Sua fundamentação foi a capacidade de construir sistemas que apresentem inteligência, simulações da teoria da inteligência humana, ou analise dos sistemas que poderiam executar tarefas práticas que exijam inteligência (SIMON, 1995).

Ibrahim e Morcos (2002) sugerem que uma boa definição da IA seria a automação de atividades que estão associadas ao pensamento humano, tais como tomada de decisão, resolução de problemas, aprendizagem, percepção e raciocínio.

Segundo Neuhauser et al. (2013), a IA procura imitar ou melhorar as capacidades da inteligência humana e tem permitido o desenvolvimento de uma comunicação altamente sofisticada, como interfaces homem-máquina fáceis de usar. Para Luger (2005) a IA pode ser definida como um ramo da ciência da computação que se preocupa com a automação do comportamento inteligente e que tem prestado mais atenção em expandir as capacidades das ciências da computação do que definir seus limites.

Já para Negnevitsky (2005) a IA se foca no desenvolvimento de soluções de valor autônomas (sistemas inteligentes) para problemas que poderiam requerer o uso de inteligência se forem feitos por humanos. Para o que Alan et al. (2004) indicam que a IA é a parte computacional da capacidade de alcançar metas no mundo. Luger (2005) enfatiza que a IA, como toda ciência, é um esforço humano, sendo por isso melhor compreendida nesse contexto.

Pesquisadores de IA, há muito tempo, têm se interessado na construção de aplicações que analisam os dados não estruturados, e, de alguma forma, categorizam e estruturam os dados de modo que a informação resultante possa ser usada diretamente para ficar sob um processo ou na interface com outros aplicativos.

Luger (2005) assegura que as duas principais preocupações dos pesquisadores da IA são a *representação do conhecimento* e a *pesquisa*. A primeira aborda o problema da captura em uma linguagem adequada de todos os parâmetros do conhecimento para um comportamento inteligente na manipulação do computador, já a segunda é uma técnica de resolução de problemas que sistematicamente explora um espaço dos estados do problema, ou seja, etapas sucessivas e alternativas no processo de resolução de problemas.

Os programas de IA podem ser vistos como experimentos. Um projeto se concretiza em um programa e o programa é executado como um experimento, depois os designers do programa

observam os resultados, redesenham e executam de novo o programa (LUGER, 2005). Algumas áreas onde a IA são utilizadas são: Neuhauser et al. (2013) e Kreps e Neuhauser (2013) utilizam IA para melhorar os programas de comunicação na área da saúde; O'Leary (2013) utiliza a IA para tratar com dados de grande volume; Ibrahim e Morcos (2002) desenvolveram uma pesquisa para mostrar a aplicação de IA em problemas elétricos principalmente em problemas de qualidade da energia. Outras áreas em que a IA pode ser utilizada incluem: Pesquisa e Planejamento, Raciocínio probabilístico e Planejamento, Representação do Conhecimento e Planejamento, Aprendizado de Máquinas, Linguagem Natural de Processamento, Sistemas Multiagentes e Robótica (GEFFNER, 2014). O trabalho em muitas dessas áreas não é muito focado em escrever programas para problemas mal definidos, mas se preocupa em escrever solucionadores para modelos matemáticos perfeitamente bem definidos. Para observar mais áreas de aplicação da IA revisar (LUGER, 2005).

3.6.2 Algoritmos Genéticos (AG)

Os AG é técnica de otimização meta-heurísticas que imitam o processo evolutivo natural. Os AG foram desenvolvidos na década de 60 vem sendo muito utilizados devido a sua simplicidade, facilidade de operação, requerimentos mínimos e por terem uma perspectiva global (KUMAR, 2011).

Os AG foram inspirados na teoria evolutiva de Darwin que indica que a sobrevivência de um organismo é afetada pela regra "o indivíduo mais bem adaptado de uma espécie tem mais chances de sobreviver". A sobrevivência de um organismo também pode ser mantida através de reprodução, cruzamento e mutação. A teoria de Darwin pode ser adaptada em um algoritmo computacional para encontrar uma solução chamando a função objetivo de uma maneira natural. Uma solução é codificada em um ou mais cromossomos. Um indivíduo pode ser formado por um ou mais cromossomos. Uma população é um conjunto de indivíduos (HERMAWANTO, 2013).

Os AG codificam uma potencial solução para um problema específico em um ou mais cromossomos, como uma estrutura de dados, e aplica operadores genéticos nestas estruturas para gerar novos cromossomos. Uma implementação dos AG inicia com uma população (tipicamente aleatória) de cromossomos. Então a estrutura é avaliada e são atribuídas mais oportunidades reprodutivas para cada um dos cromossomos que representam boas soluções para o problema alvo, obtendo mais chances de se "reproduzir"do que aqueles cromossomos que são soluções de pior qualidade (WHITLEY, 1994).

Para cada cromossomo, diferentes posições são codificadas como bits, caracteres ou números. Essas posições podem ser referenciadas como genes. Uma função de avaliação é usada

para calcular a qualidade de cada cromossomo de acordo com a solução desejada; esta função é conhecida como "função de fitness". Durante o processo de avaliação, o "cruzamento"é usado para simular uma reprodução natural e a "mutação"é usado para a mutação de espécies (HOQUE et al., 2012).

Uma vez que o problema é codificado como um cromossomo e uma função de fitness é usada para discriminar boas soluções de ruins, pode-se começar a evoluir soluções para o problema de busca usando os seguintes passos. Tomado de (SASTRY et al., 2005).

1. **Inicialização:** A população inicial de soluções candidatas é usualmente gerada aleatoriamente.

2. **Avaliação:** Uma vez a população é iniciada, ou uma população de descendência é criado, os valores de *fitness* das soluções candidatas são avaliadas.

3. **Seleção:** Atribui probabilisticamente mais cópias para as soluções com melhores valores de *fitness* e, dessa maneira, impõe o mecanismo de ajuste-de-sobrevivência sobre as soluções candidatas.

4. **Recombinação:** Combina partes de duas ou mais soluções parentes para criar uma nova.

5. **Mutação:** Enquanto a recombinação opera sobre dois ou mais cromossomos paralelamente, a mutação modifica a solução local, mas aleatoriamente.

6. **Substituição:** A população de descendência criada por seleção, recombinação e mutação substitui a população parente original.

7. Repetir os passos 2 a 6 até chegar num critério de parada.

Na literatura podem ser encontrados vários trabalhos nos quais o AG é utilizado para resolver diferentes tipos de PM, por exemplo: Khuri et al. (1994) utilizam o AG para criar um algoritmo que resolva o problema da mochila 0-1 múltiplo mediante a reprodução e participação de soluções não viáveis da população. Chu e Beasley (1998) desenvolveram uma heurística baseada no AG para resolver o problema da mochila multidimensional, incorporando um conhecimento especifico do problema. Raidl (1998) resolve o problema da mochila 0-1 multi-restrito utilizando um AG hibrido. Cotta e Troya (1998) também utilizaram uma versão hibrida do AG para resolver o problema da mochila 0-1 múltiplo.

3.6.3 *Particle Swarm Optimization* (otimização por enxame de partículas)

PSO é um método heurístico de otimização global, que tem por duas ideais fundamentais: psicologia social e evolução computacional (PARSOPOULOS; VRAHATIS, 2007). Introduzido por Kennedy e Eberhart em 1995 (KENNEDY; MENDES, 2002). Foi desenvolvido baseado em inteligência coletiva (comportamento social), observando o comportamento do movimento de bandos de pássaros e cardumes de peixes à procura de alimento (BAI, 2010). O PSO vem ganhando popularidade devido a sua habilidade para resolver eficazmente e eficientemente uma grande variedade de problemas na ciência e na engenharia (PARSOPOULOS; VRAHATIS, 2007), especialmente pela sua velocidade de convergência e o fato de ser muito simples de se usar (MENDES et al., 2004). Além de se basear em um conceito muito simples, o paradigma é implementado em poucas linhas de código de programação e requer apenas operadores matemáticos primitivos que são computacionalmente baratos em termos de capacidade de memória e velocidade (EBERCHART; KENNEDY, 1995).

O PSO é uma técnica de busca em paralelo multi-agente, na qual as partículas são entidades conceituai que viajam através do espaço de busca multi-dimensional. A cada instante particular, cada partícula tem uma posição e uma velocidade. O vetor de posição da partícula com respeito à origem de busca representa uma solução candidata do problema de pesquisa (DAS et al., 2008).

Na implementação mais comum do PSO, as partículas se movimentam através do espaço de busca utilizando uma combinação de uma atração para a melhor solução que elas encontraram individualmente, e uma atração para a melhor solução que qualquer partícula em sua vizinhança tenha encontrado, e uma atração para a direção do vetor de velocidade (BRATTON; KENNEDY, 2007). Um exame de partículas é mais do que simplesmente uma coleção de partículas. Uma partícula por si só não tem o poder para resolver nenhum problema; progressos somente ocorrem quando as partículas interagem entre si (POLI et al., 2007).

O procedimento dos algoritmos baseados de busca populacionais como o PSO envolve duas etapas principais: *exploração*, que é responsável pela detecção da região mais promissoras dentro do espaço de busca, e *aproveitamento*, que promove a convergência das partículas na vizinhança da melhor solução encontrada (PARSOPOULOS; VRAHATIS, 2007). A *figura 3.5* mostra o comportamento das partículas após um número N de iterações.

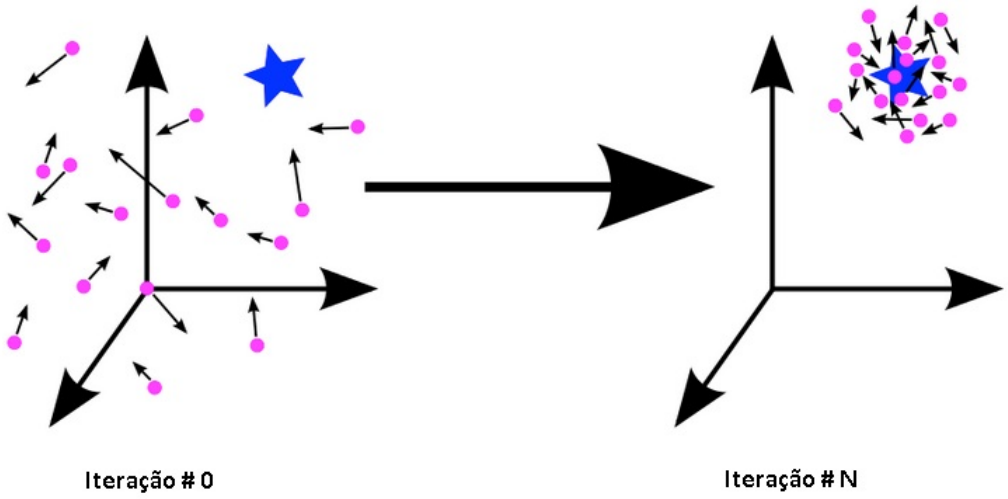

Figura 3.5 – Posição das partículas após *N* iterações

As aplicações do PSO são tão numerosas e diversas que fica muito difícil mencioná-las já que segundo Poli et al. (2007) o PSO tem mostrado grandes resultados em problemas multimodais e problemas para os quais não existem métodos especializados de avaliação, ou ainda onde todos os métodos especializados têm dado resultados não satisfatórios. Por isso os mesmos autores analisaram ao redor de 1100 publicações sobre PSO e dividiram as aplicações dentro de 26 categorias, indicando a quantidade de artigos referentes a cada aplicação e a porcentagem que representa dentro da literatura pesquisada.

Deep e Bansal (2008) desenvolveram uma versão do PSO para resolver o problema da mochila multidimensional, introduzindo uma distância entre a o melhor ponto da vizinhança local e o melhor ponto da vizinhança global. Chih et al. () também resolveram o problema da mochila multidimensional utilizando o PSO, mas eles introduziram coeficientes de aceleração de tempo variável. Bansal e Deep (2012) apresentam uma modificação do PSO binário para resolver o problema da mochila 0-1, introduzindo uma nova função de probabilidade que permite maior exploração do enxame.

3.6.4 Busca Tabu

A busca Tabu tem suas origens em procedimentos combinatórios aplicados a problemas não lineares nos anos 70, e é uma estratégia para resolver problemas de otimização combinatória cujas aplicações vão desde teoria de grafos até problemas gerais de programação pura, inteira ou mista (GLOVER, 1989).

A busca Tabu consiste em um algoritmo iterativo que utiliza o conceito de memória para controlar a execução do algoritmo através de uma lista dinâmica de movimentos proibidos (HENDERSON et al., 2003). Inspirado nos princípios da IA que explora o espaço de solução sem a necessidade de ser confinado a um ótimo local, para este propósito os movimentos que pioram a solução são permitidos e os últimos movimentos são armazenados em uma lista chamada "lista Tabu". Esses movimentos não podem ser aplicados na iteração seguinte para evitar ciclos durante o processo de otimização, desta maneira a busca Tabu é uma combinação de busca local com uma memória de curto prazo (NAVARRO; RUDNICK, 2009). A lista Tabu é uma das maneiras de utilizar a memória mediante o armazenamento das soluções exploradas através da busca ou, mais comumente, alguns atributos relevantes dessas soluções. A lista Tabu tem dois propósitos: prevenir o retorno à última solução visitada a fim de evitar ciclos; e conduzir a busca até regiões ainda não exploradas que possam ter alto potencial de conter boas soluções (BRANDÃO, 2011).

Nos últimos 20 anos centenas de trabalhos têm apresentado diferentes aplicações da busca Tabu para vários problemas combinatórios encontrados na literatura. Em vários casos a busca Tabu fornece soluções muito próximas da solução ótima e está entre as técnicas mais eficazes para enfrentar alguns problemas difíceis. Esses sucessos tornaram a busca Tabu extremamente popular entre os pesquisadores interessados em encontrar boas soluções para os problemas combinatórios de grande porte encontrados em várias definições práticas (GENDREAU; POTVIN, 2010).

Já procedimentos mais avançados incorporam estruturas adicionais de memória de curto e longo prazo, incluindo as estruturas baseadas em frequência e análise lógica. Esses autores utilizaram a busca Tabu para resolver o problema de atribuição quadrática.

Uma ótima aplicação da busca Tabu requer um bom balanço entre intensificação (exploração detalhada de alguma região do espaço de solução) e diversificação (dirigir a busca até regiões promissoras ainda não exploradas no espaço de solução e afastadas umas das outras) (BRANDÃO, 2011). A busca Tabu começa definindo a solução do espaço de trabalho para sua melhor solução inicial obtida a partir do conjunto de referências. Cada iteração examina todos os possíveis movimentos da vizinhança da permutação de trabalho escolhendo a melhor solução que ainda não se encontra na lista Tabu e que se mostra adequada para trocar uma das permutações já realizadas. Esta troca é realizada e a permutação modificada se torna a nova solução de trabalho (JAMES et al., 2009). A vizinhança de uma solução é um conjunto de outras soluções que podem ser obtidas a partir de uma ou algumas mudanças simples. A melhor solução da vizinhança é escolhida como uma nova solução incumbente. Movimentos que não levaram a melhorias são aceitos para tentar fugir de um mínimo local (PEDERSEN et al., 2009).

Uma área importante na qual a busca Tabu é muito utilizada é a área de transporte, por exemplo: Brandão (2011) e Cordeau e Maischberger (2012) utilizam busca Tabu para tratar o problema de roteamento de veículos.

Exemplos de PM resolvidos utilizando Busca Tabu podem ser encontrados em: Hanafi e Freville (1998) que usaram a Busca Tabu baseada numa estratégia de oscilação e substituição da informação das restrições para resolver o problema da mochila 0-1 multidimensional. Chocolaad (1998) utilizou a Busca Tabu para resolver o problema da mochila geométrica. Gandibleux e Freville (2000) basearam sua pesquisa na Busca Tabu para resolver o problema da mochila 0-1 multiobjetivo através da redução da estrutura do espaço de decisão.

3.6.5 *Simulated Annealing* (Recozimento Simulado (RS))

Simulated Annealing ou em português Recozimento Simulado, surgiu de uma analogia com certos processos termodinâmicos (CRAMA; SCHYNS, 2003) e pertence a uma classe de algoritmos de busca local que são conhecidos como algoritmos limiar. Os algoritmos limiar desempenham um papel especial na busca local por duas razões: a primeira, eles parecem ser muito bem-sucedidos quando são aplicados a uma ampla gama de problemas práticos, o que lhes proporcionou uma boa reputação entre os profissionais. A segunda, alguns algoritmos limiares como o RS têm componentes estocásticos, o que facilita uma análise teórica de sua convergência assintótica, e isso tem feito deles muito populares em matemática (AARTS et al., 2005).

Henderson et al. (2003) indicam que RS é uma meta-heurística popular usada para resolver de uma maneira discreta e menos estendida problemas de otimização contínua. A capacidade de fugir de um ótimo local, a simplicidade de sua implementação e suas propriedades de convergência e a utilização de movimentos de subida para escapar de ótimos locais fizeram da RS uma técnica popular ao longo das últimas décadas.

O princípio básico da heurística RS pode ser descrito como: a partir uma solução inicial de x, outra solução y é gerada após dar um passo estocástico em alguma vizinhança de x. Se a nova resposta melhorar o valor da função objetivo, então y substitui x como a nova solução atual; caso contrário, a nova solução y é aceita com uma probabilidade que diminui com o decorrer das iterações. Para Crama e Schyns (2003), a maior vantagem do RS sobre os métodos de busca clássicos é sua capacidade de evitar ficar preso em mínimos locais enquanto procura um mínimo global.

O RS é particularmente bem adaptado para vários problemas de otimização combinatória, uma vez que pode evitar mínimos locais, aceitando melhorias no custo (JEON et al., 2002).

A maioria dos desenvolvimentos teóricos e aplicações que trabalham com RS tem sido

para problemas de otimização discreta. Porém, também pode ser utilizada como uma ferramenta para resolver problemas no domínio contínuo, por isso existe um considerável interesse em utilizar RS para otimização global em regiões contendo vários mínimos locais e globais (HENDERSON et al., 2003).

Vários exemplos de utilização de RS podem ser encontrados na literatura por exemplo: Vincent et al. (2010) utilizaram RS para abordar o problema de roteamento de veículos capacitados; Dupanloup et al. (2002) usaram um enfoque de RS para definir a estrutura genética de diferentes grupos de populações; Jeon et al. (2002) propuseram um algoritmo de RS para a reconfiguração da rede de sistemas de distribuição a grande escala; Crama e Schyns (2003) descrevem a aplicação de RS para a solução de um modelo de seleção de portfólio complexo; outras aplicações do RS podem ser encontradas em (HENDERSON et al., 2003). Já no que diz a problemas da mochila vários são os exemplos achados na literatura, por exemplo: Drexl (1988) utilizou o RS para fazer uma boa aproximação na solução do problema da mochila 0-1 muti-restrito, Leung et al. (2012) desenvolveram um algoritmo meta-heurístico baseado no RS para resolver o problema do empacotamento da mochila bidimensional. Qian e Ding (2007) presentaram um algoritmo de RS para resolver o problema da mochila multidimensional incorporando conhecimento específico do problema. Liu et al. (2006) desenvolveram um algoritmo de RS melhorado para resolver eficientemente o problema da mochila linear 0-1.

4 EXPERIMENTOS COMPUTACIONAIS

Para determinar a eficiência do método de ranking de pontos extremos, foram criados 12 casos de problemas com o seguinte número de variáveis: 25, 50, 100, 250, 500, 1.000, 2.500, 5.000, 10.000, 15.000, 20.000 e 30.000. Para cada caso foram resolvidas 100 instâncias. Todos os resultados foram obtidos utilizando Matlab R2013a rodados em um computador DELL PRECISION M4500 com processador Core i7, memória RAM de 8,00GB e sistema operativo Windows 7.

O código desenvolvido para o método de ranking de pontos extremos utiliza a metodologia proposta por Dantzig (1957) para encontrar a solução inicial do problema relaxado e posteriormente utiliza as características normais do método de ranking de pontos extremos para analisar os próximos pontos. Para testar a validade código desenvolvido, o desempenho foi comparado com dois modelos diferentes de *branch-and-bound* que utilizam a técnica *best-first* para escolher o subproblema a ser resolvido. O primeiro utiliza o método *simplex* para resolver os problemas e será chamado como *branch-and-bound-simplex* ou BBS, e pode ser encontrado na função "bintprog"do Matlab R2013a. Para este modelo o tempo máximo de convergência foi aumentado em 100 vezes ao tempo padrão dado pelo Matlab, para melhorar a qualidade das respostas. O segundo modelo utiliza o método de Dantzig para resolver os problemas e será chamado como *branch-and-bound-Dantzig* ou BBD.

Os problemas gerados foram criados aleatoriamente pelo programa Matlab R2013a. Para que cada problema atendesse às condições do problema da mochila (uma função objetivo sujeita a uma única restrição de mochila) foi considerada a seguinte formulação para facilitar a manipulação dos dados:

$$\begin{aligned} \text{minimizar} \quad & p^T x \\ \text{sujeito a} \quad & Wx \geq c \\ & x \in \{0,1\} \end{aligned} \tag{4.1}$$

Onde p é um vetor coluna de dimensão n ($n =$ número de variáveis). Cada coluna de p representa uma função objetivo diferente, e cada linha contém os valores dos coeficientes para cada uma das variáveis do problema analisado. Os valores dos coeficientes foram gerados aleatoriamente no intervalo $\{1 : 100000\}$ com o intuito de evitar obter pontos iguais na hora de resolver cada problema. W é uma matriz linha de dimensão n onde cada coluna de W

representa uma restrição diferente, e cada linha contém os valores dos pesos para cada uma das variáveis do problema analisado. Os diferentes pesos foram gerados aleatoriamente no intervalo $\{1 : 200\}$ com a finalidade de ficar o mais perto possível a problemas reais. Já c é um vetor coluna de dimensão $100x1$ no qual cada linha representa uma capacidade de mochila diferente para cada restrição em W. As diferentes capacidades foram geradas utilizando a expressão $\min(W) - \sum_1^n (W - 1)$ (*min* representa o menor valor da matriz W).

Cada problema gerado foi testado com cada um dos métodos e as medidas de desempenho selecionadas para determinar a eficiência de cada método foram: média de tempos utilizados para resolver cada instância (em segundos computacionais), desvio padrão da média dos tempos, mediana dos tempos. Também foi determinado o tempo máximo e mínimo gasto para resolver um problema, para determinar em quais problemas o ranking de pontos extremos se apresenta melhor e pior. Os valores do primeiro e terceiro interquartil de tempos que serão apresentados abaixo servem para indicar em qual intervalo se localizam a maioria dos dados para cada método. O índice de assimetria ajudará a determinar qual dos métodos tem os dados melhor distribuídos. Todas essas análises ajudaram a determinar qual método é o melhor.

A Tabela 4.1 apresenta as seguintes medidas de desempenho: médias do tempo gasto por cada método para resolver as instâncias dos problemas para cada caso; o desvio padrão das médias de tempo e as médias dos tempos para cada caso. Na tabela, pode-se observar que o BBD é o método mais rápido e com menos desvio padrão dentre os três. Já o ranking de pontos extremos apresenta um desempenho aceitável até o caso com 500 variáveis, mas para os problemas maiores sua velocidade assim como o desvio padrão pioram rapidamente.

Tabela 4.1 – Medidas de desempenho de cada método

Medidas	**25**	**50**	**100**	**250**	**500**	**1.000**	**2.500**	**5.000**	**10.000**	**15.000**	**20.000**	**30.000**
Média												
BBS	0,064	0,269	1,569	3,959	7,054	8,688	36,566	59,092	153,773	23,221	259,109	412,316
BBD	0,011	0,022	0,051	0,146	0,366	0,538	1,402	1,369	1,591	1,367	2,140	3,263
Ranking	0,023	0,080	0,305	2,069	7,149	15,439	62,866	128,317	257,195	339,288	572,266	923,261
Desvio Padrão												
BBS	0,053	0,285	1,519	4,231	7,828	7,262	37,795	52,368	186,078	5,473	245,239	389,168
BBD	0,011	0,026	0,052	0,198	0,624	1,145	5,986	2,854	2,327	2,041	3,538	5,072
Ranking	0,032	0,109	0,360	2,695	10,488	22,118	147,694	171,024	305,464	508,326	909,166	1.295,628
Mediana												
BBS	0,052	0,203	1,255	2,608	4,427	6,849	22,620	44,442	100,073	25,519	187,326	309,418
BBD	0,008	0,016	0,036	0,071	0,108	0,162	0,216	0,585	0,574	0,576	0,821	1,441
Ranking	0,013	0,036	0,197	1,144	3,071	7,983	20,981	64,401	135,925	128,287	190,402	412,016

Analisando as medianas dos tempos, pode-se observar que a quantidade de respostas distantes em cada caso não é tão grande entre o BBS e o ranking de pontos extremos, como indicavam os resultados das médias e do desvio-padrão.

Ao analisar os tempos máximos e mínimos gastos pelos métodos para cada caso analisado

apresentados na Tabela 4.2, pode-se observar que o método do ranking de pontos extremos tem um comportamento aceitável para algumas estruturas, perdendo somente para o BBD. Mas quando o desempenho se apresenta ruim, as respostas são péssimas.

Tabela 4.2 – Tempos máximos e mínimos gasto por cada método

Tempos	**25**	**50**	**100**	**250**	**500**	**1.000**	**2.500**	**5.000**	**10.000**	**15.000**	**20.000**	**30.000**
Máximo												
BBS	0,323	1,409	8,068	18,666	39,260	34,884	148,487	248,531	1.079,950	32,281	1.200,280	2.501,169
BBD	0,066	0,131	0,297	1,200	3,219	7,415	49,532	21,243	14,130	13,178	19,550	40,551
Ranking	0,219	0,533	1,941	18,743	65,438	147,529	1.144,973	1.033,588	1.387,440	3.304,650	4.651,920	9.080,339
Mínimo												
BBS	0,010	0,011	0,046	0,080	0,166	0,354	0,932	1,761	10,735	3,286	6,412	47,915
BBD	0,001	0,0003	0,001	0,001	0,002	0,002	0,002	0,003	0,004	0,006	0,007	0,007
Ranking	0,001	0,001	0,002	0,002	0,002	0,002	0,003	0,003	0,006	0,007	0,007	0,010

Também foi analisada a assimetria dos dados para cada método com o objetivo de determinar o quão distribuídas foram as respostas à medida que o tamanho dos problemas crescia. Na Tabela 4.3 observa-se que para quase todos os casos o ranking de pontos extremos foi mais assimétrico do que o BBS.

Tabela 4.3 – Índice de assimetria para cada método

Assimetria	**25**	**50**	**100**	**250**	**500**	**1.000**	**2.500**	**5.000**	**10.000**	**15.000**	**20.000**	**30.000**
BBS	2,591	2,007	1,47	1,583	1,662	1,398	1,467	1,407	2,596	-2,276	1,782	2,585
BBD	2,573	2,255	1,854	2,664	2,637	4,254	6,989	4,580	2,958	3,235	3,083	4,532
Ranking	3,578	2,312	2,147	3,018	2,810	3,068	5,686	2,402	1,669	3,066	2,787	3,195

Para entender melhor o comportamento de cada método nos diferentes casos a figura 4.1, mostra por meio de gráficos box-plot (mediana-quartis) como cada método resolveu as instâncias para cada caso. Box-plot é uma maneira padrão para mostrar a concentração da distribuição de dados, o 50% dos dados da distribuição estão concentrados dentro da caixa do box-plot.

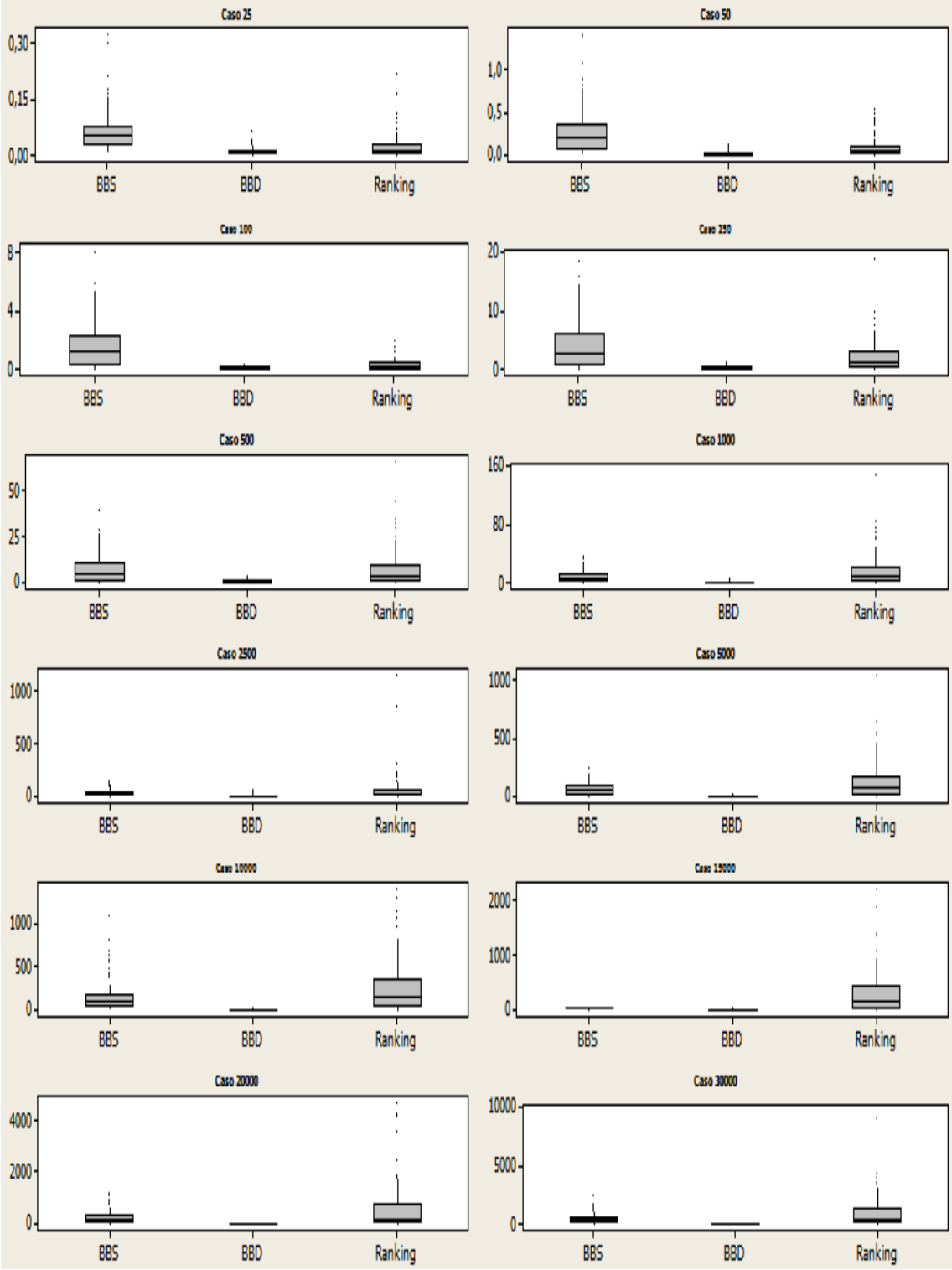

Figura 4.1 – Box-plot para cada caso

A Figura 4.2, mostra as melhores regressões para cada método, sendo a regressão de potência a que apresentou um melhor R^2 para os três métodos, mas o valor do R^2 foi baixo para cada um deles isso porque como se pode notar na figura4.2 os valores apresentam um comportamento heterocedástico. Para uma melhor compreensão visual se utilizou a mesma escala para todos os métodos.

Da Figura 4.2, obtêm-se a Tabela 4.4 que resume qual foi a melhor regressão encontrada para cada método junto com sua equação e respetivo R^2. É importante destacar que para o BBD a formula da regressão indica que, à medida que o tamanho do problema aumenta o tempo vá crescendo de maneira mais lenta do que os outros dois métodos.

Assim, ao obter e analisar as respostas, o valor R^2 para os modelos BBS y Ranking de pontos extremos foram bons, já o valor do R^2 do modelo de BBD não foi bom, mas foi o melhor que valor encontrado após analisar várias transformações dos valores originais.

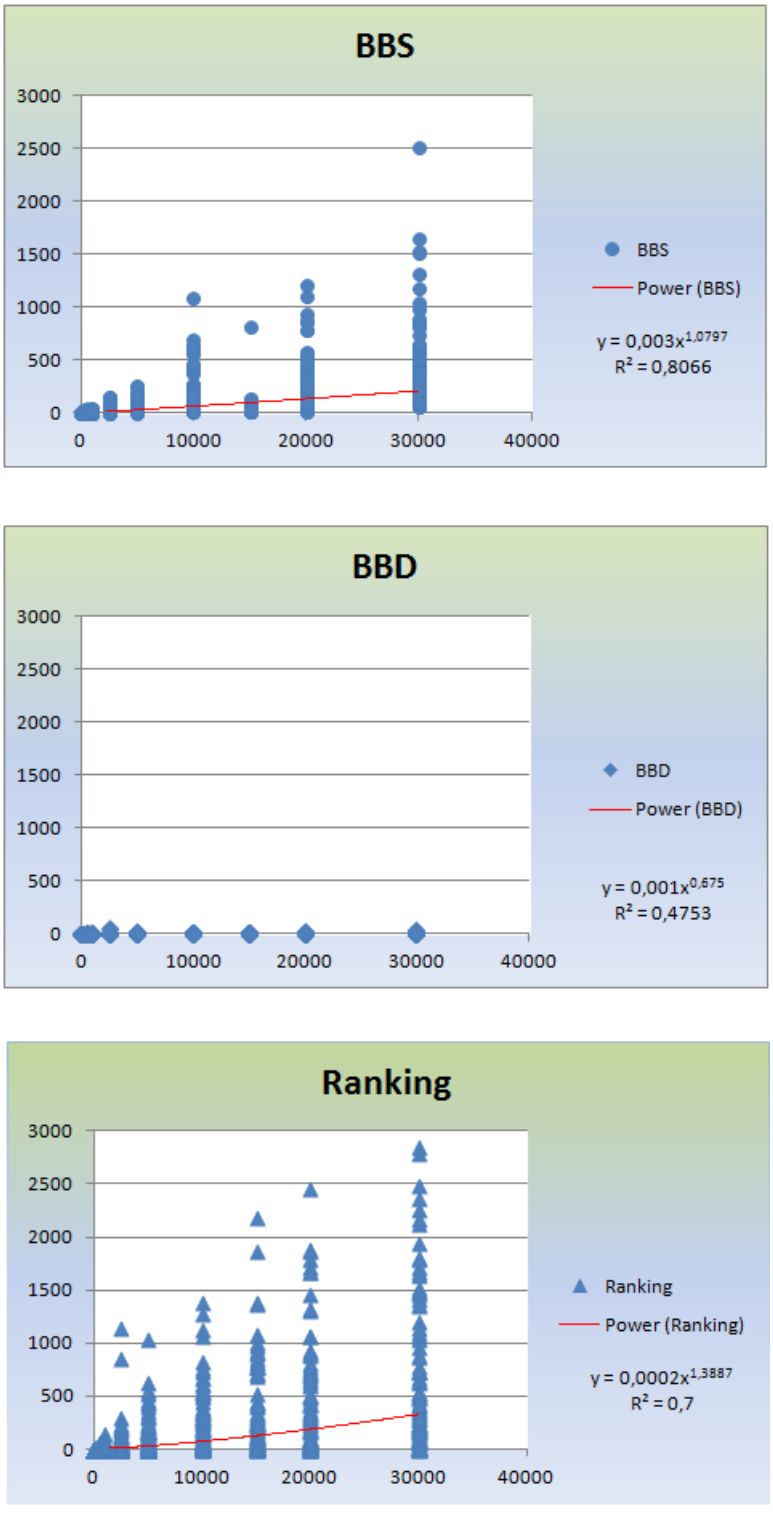

Figura 4.2 – Melhor regressão para cada caso

Tabela 4.4 – Dados da regressão de cada método

Método	**Regressão**	**Fórmula**	R^2
BBS	Potência	$y = 0,003x^{1,0797}$	0,8066
BBD	Potência	$y = 0,001x^{0,675}$	0,4753
Ranking	Potência	$y = 0,0002x^{1,3887}$	0,7

O método de ranking de pontos extremos mostrou um desempenho aceitável comparado com o BBD em problemas até com 500 variáveis, mas à medida que o tamanho dos problemas aumenta este método perde desempenho comparado aos outros dois métodos. Isso acontece porque, na medida que o tamanho dos problemas aumenta, a estrutura dos problemas ramificados pelos modelos de B&B são menores e mais simples de resolver; enquanto o método de ranking de pontos extremos mantém a mesma estrutura para todos os problemas que analisa. O método BBS aparentemente se apresenta como o pior de todos os métodos devido ao fato de ter atingindo os valores ótimos em poucas instâncias para os casos analisados, a diferença dos outros dois métodos que alcançaram 100% das respostas ótimas como mostra a Tabela 4.5.

Tabela 4.5 – Porcentagem de instancias onde o ótimo foi encontrado

%	**25**	**50**	**100**	**250**	**500**	**1.000**	**2.500**	**5.000**	**10.000**	**15.000**	**20.000**	**30.000**
BBS	100	98	90	53	41	30	36	38	32	4	39	35
BBD	100	100	100	100	100	100	100	100	100	100	100	100
Ranking	100	100	100	100	100	100	100	100	100	100	100	100

5 CONCLUSÕES

Esse trabalho abordou o *Problema da Mochila Linear (PML)*, um problema de otimização combinatória que é muito utilizado na indústria e na administração.

Nesse estudo foi retomado o conceito do método de *ranking de pontos extremos* que, quando foi apresentado na década de 70, mostrou bons resultados para resolver o PML. Mas nos últimos 40 anos os pesquisadores têm preferido utilizar outras técnicas apresentadas após o ranking de pontos extremos. Por isso, essa dissertação buscou responder a seguinte pergunta: justifica não utilizar mais o método de ranking de pontos extremos para resolver o PML?

Para cumprir o objetivo da dissertação, no Capítulo 4 foram elaborados, resolvidos e analisados vários problemas de mochila com diferentes tamanhos e instâncias. Nesses problemas buscou-se comparar o desempenho do método de ranking de pontos extremos com os desempenhos de dois enfoques de *branch-and-bound* que utilizam a técnica *best-first* para elaborar os subproblemas. O primeiro resolve os subproblemas relaxados com a ajuda do método *simplex* e será chamado como *branch-and-bound-simplex* ou BBS, enquanto o segundo utiliza o método proposto por Dantzig para resolver os subproblemas relaxados, e será chamado como *branch-and-bound-Dantzig* ou BBD

Os resultados obtidos no Capítulo 4 mostraram que o melhor desempenho tanto em rapidez quanto em qualidade de respostas foi o obtido pelo método BBD, pelo menos na implementação em série, o que **justifica** ter deixado de utilizar o método de ranking de pontos extremos, porque existem métodos melhores.

No que diz respeito à rapidez, em problemas até com 500 variáveis o ranking de pontos extremos se apresentou competitivo com o BBD, mas à medida que o tamanho dos problemas aumentava seu desempenho piorava rapidamente. Isto acontece porque, à medida que as ramificações do B&B aumentam, os subproblemas a serem resolvidos são menores e mais simples, já no ranking de pontos extremos são resolvidos o mesmo número de problemas a cada instância.

No que diz respeito à qualidade das respostas, os métodos de ranking de pontos extremos e o BBD atingiram as respostas ótimas 100% das vezes. Já o BBS, dependendo das características dos problemas não conseguiu atingir o ótimo em vários deles mesmo tendo incrementado o tempo para a sua convergência. Isso nos levou a indicar que o BBS é o pior método dos aqui analisados.

Cabe destacar que em nenhum momento foi considerada a paralelização computacional dos algoritmos, algo que é muito comum hoje em dia e que poderia mudar os desempenhos de cada método aqui apresentado devido a existir computadores que podem fazer isso de maneira mais

rápida. Com a paralelização o ranking de pontos extremos provavelmente teria um desempenho mais competitivo comparado com o BBD em problemas com tamanhos grandes.

5.1 Sugestões para trabalhos futuros

É conveniente estudar o desempenho do ranking de pontos extremos e do BBD mediante uma implementação computacional paralela, já que se acredita que o ranking de pontos extremos se apresentará mais competitivo.

Também é conveniente estudar o comportamento do método de ranking de pontos extremos em problemas de mochila que apresentem capacidades pequenas, já que as soluções para problemas relativamente medianos foram bons. Seria interessante ver o desempenho do método ao resolver problemas com capacidades mais restritas.

Seria muito importante utilizar o BBD para resolver outro tipo de problemas contendo maior número de restrições, ou inclusive problemas quadráticos, para observar seu comportamento nesse tipo de problemas já que os problemas quadráticos podem ser tratados como lineares, e o método BBD demostrou ser muito bom para resolver problemas do tipo linear.

Torna-se interessante estudar o comportamento dos métodos BBD e ranking de pontos externos em problemas com uma correlação grande, já que esses tipos de problemas são mais difíceis de serem resolvidos.

Uma ideia interessante que pode ser estudada é a de utilizar o ranking de pontos extremos tomando em conta unicamente os pontos adjacentes mais promissores. Dessa maneira a velocidade do método poderia aumentar muito.

A APÊNDICES

A.1 Programação Linear (PL) e métodos de solução

A Programação Linear é um método aplicável para a solução de problemas nos quais a função objetivo e as restrições aparecem como funções lineares de variáveis de decisão, as que permitem fazer decisões ótimas em situações complexas (RAO, 2009).

Os programas lineares podem ser vistos de duas maneiras um tanto complementares. Por um lado, eles são uma classe de problemas de otimização contínua cada um com variáveis contínuas definidas em uma região viável convexa e com uma função objetivo comum. Por outro lado, os programas lineares podem ser considerados como uma classe de problemas combinatórios, porque é sabido que as soluções podem ser encontradas ao se restringir os vértices do poliedro convexo definido pelas restrições lineares (LUENBERGER; YE, 2008).

A formulação padrão dos problemas de PL tem a seguinte forma:

$$\begin{aligned} \text{maximizar} \quad & c^T x \\ \text{sujeito a} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Onde c (frequentemente se refere a custos) e x são vetores em $\mathbb{R}^n$, b é um vetor em $\mathbb{R}^m$, e A (conjunto de restrições) é uma matriz $mxn(m \leq n, usualmentem < n)$, e onde a função objetivo tem que ser a menor possível.

Muitas situações reais podem ser formuladas ou aproximadas como problemas de programação linear, nos quais as soluções ótimas são relativamente fáceis de se calcular e os códigos computacionais para resolver problemas de grandes instâncias são comercialmente aceitos. Mas às vezes a importância da PL não é observada já que frequentemente os algoritmos de PL são utilizados como sub-rotinas para resolver problemas de otimização mais difíceis em programação não linear e em programação inteira (BARD, 2013).

A seguir serão apresentados os métodos simplex e simplex revisado que são os métodos mais comumente utilizados para resolver problemas de PL.

A.2 Método Simplex (MS)

Atualmente a PL e o método *simplex* continuam se mantendo como as ferramentas mais utilizadas em otimização (NOCEDAL; WRIGHT, 2006). Introduzido por Dantzig no final dos

1940s, o método *simplex* tem marcado o início de uma era moderna de otimização (WRIGHT; NOCEDAL, 1999), sendo o método mais utilizado em PL, e um dos mais utilizados por todos os algoritmos numéricos (GRIVA et al., 2009). O MS fez possível formular e analisar modelos cumpridos de uma maneira sistemática e eficiente (NOCEDAL; WRIGHT, 2006).

A ideia básica é a de percorrer os vértices do politopo subjacente de uma forma sistemática que evita investigar pontos inviáveis. O MS percorre os vértices da região factível até encontrar a solução ótima, se baseia no fato de que o valor ideal de um programa linear (A.1), se finito, é sempre conseguida com uma solução básica viável (LUENBERGER; YE, 2008).

Diferentes combinações de variáveis são examinadas uma de cada vez sem ter que voltar a uma combinação (vértice) previamente explorada (BARD, 2013).

Oliveira (1991) indica que: *"o método simplex busca uma solução ótima através de deslocamentos entre os pontos extremos (vértices) da fronteira da região de factibilidade usando a informação de que um desses pontos é a solução ótima do problema"* (OLIVEIRA, 1991). O método simplex fornece a base para a otimização de uma longa sequência de relaxamentos de PL (CHEN et al., 2010), por isso Fletcher (2013) indica que o MS em sua forma padrão gera uma sequência de pontos $x_1, x_2, \ldots$ que terminam em uma solução, cada iteração x_k é um vértice da região de solução. A cada iteração o método observa se a base atual é ótima, se não, o método escolhe uma direção factível ao longo da qual melhora a função objetivo movimentando-se até uma solução factível adjacente ao longo daquela direção (GRIVA et al., 2009).

Luenberger e Ye (2008) indicam que a ideia do método simplex é proceder de uma solução básica viável (isto é, um ponto extremo) do conjunto de restrições de um problema na forma padrão (A.1) para outro, de tal forma que diminuem continuamente, até chegar em um ponto ótimo onde o mínimo da função seja alcançado.

Forsgren et al. (2002) indicam que o método simplex tem um procedimento no qual a iteração se movimenta desde um vértice até outro mudando o conjunto de restrições que tem (uma de cada vez), diminuindo o objetivo para onde ele vá, até achar um vértice ótimo. O MS não pode visitar o mesmo ponto factível x em duas diferentes iterações, porque consegue uma diminuição rigorosa em cada iteração. Já que o número de possíveis soluções é finito, o número de iterações também será finito, então o método deve terminar em uma solução ótima (NOCEDAL; WRIGHT, 2006).

A *figura A.1*, adaptada de Wright e Nocedal (1999), mostra o caminho que o método simplex segue a cada iteração para um problema de duas dimensões.

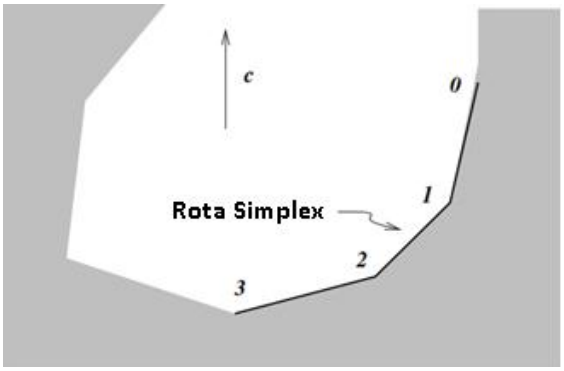

Figura A.1 – Interações simplex para um problema de duas dimensões.

Segundo Chen et al. (2010), o método simplex pode ser considerado como um algoritmo iterativo o mesmo que consiste dos seguintes passos:

1. *Inicialização*: Encontra uma solução básica inicial que é viável.
2. *Iteração*: Encontra uma solução básica melhor, adjacente e viável.
3. *Teste de otimalidade*: Testa se a atual solução é ótima. Se não, repete o passo 2.

Embora a programação linear e o método simplex continuem sendo as ferramentas de otimização mais utilizadas e tenham atuado bem na prática, na maioria de problemas de grande porte o método simplex não consegue visitar todos os vértices ou para fazê-lo requer um número de iterações muito grande (LUENBERGER; YE, 2008). E o método simplex não serve para problemas não lineares (diretamente).
Para um estudo mais profundo sobre o método simplex revisar: (FLETCHER, 2013), (LUENBERGER; YE, 2008), (WRIGHT; NOCEDAL, 1999), (GRIVA et al., 2009), (CHEN et al., 2010) ou (NOCEDAL; WRIGH 2006).

A.3 Método Simplex Revisado (MSR)

As oportunidades e limitações das modernas arquiteturas computacionais, junto com o desejo de resolver problemas com instâncias maiores, continua levando ao desenvolvimento de técnicas com alto rendimento computacional para o método *simplex* já que ele é muito ineficiente quando é aplicado a problemas de PL esparsados. Por isso, para esse tipo de problemas, o método *simplex* revisado é o preferido sempre que permita que a esparsidade dos problemas seja explorada. A esparsidade dos problemas é conseguida utilizando técnicas de fatoração para matrizes esparsas e resolvendo sistemas hiper-espersos (HUANGFU; HALL, 2015a). Para classes particulares de problemas de PL, o método simplex revisado é geralmente a técnica de solução preferida (HUANGFU; HALL, 2015b).

O método *simplex* revisado é uma representação algébrica sucinta e eficientemente aplicável utilizada para descrever o método *simplex* tradicional, no qual unicamente uma pequena parte da tabela condensada é calculada, a dizer, as entradas da tabela correspondentes à última coluna, a fila inferior, a coluna pivô e os níveis (essas entradas são suficientes para determinar o passo pivô), o que é necessário para decidir a direção a seguir até uma nova solução factível básica; em vez de representar toda a tabela explicitamente. Unicamente são manipuladas duas matrizes B (matriz de variáveis base) e N (matriz de variáveis não base), e mediante o armazenamento de B e N em essência se está armazenando uma representação implícita da tabela completa do método *simplex* tradicional, isso permite ter uma economia computacional importante (FERRIS et al., 2007).

O método *simplex* revisado pertence a uma classe geral de algoritmos para otimização restrita conhecida como "método do conjunto ativo", que mantém explicitamente as estimativas dos índices dos conjuntos ativos e não ativos que são atualizados a cada passo do algoritmo. Como a maioria dos métodos de conjuntos ativos, o método *simplex* revisado faz mudanças para os índices desses conjuntos a cada passo, um único índice é trocado entre B e N (B é a matriz que contém as variáveis base e N é a matriz que contem as variáveis não base) (NOCEDAL; WRIGHT, 2006).

O principal desafio computacional quando se implementa o MSR é achar a solução efetiva de um sistema de equações lineares cuja matriz de coeficientes é a matriz de bases do simplex ou sua transposta (HUANGFU; HALL, 2015b). Mais informações podem ser encontradas em (NOCEDAL; WRIGHT, 2006) e (RAO, 2009).

A.4 Teorema para problemas inteiros

O teorema A.4.1 apresentado no livro de (TAHA, 1974) indica que para problemas lineares zero-um a solucao ótima está em um dos vértices do politopo de solução.

A.4.1 Teorema Taha

Teorema Em um problema linear zero-um, misto ou puro, assume-se que a restrição inteira é substituída pelo intervalo continuo $0 \leq x_j \leq 1$; então *(i)* o espaço de solução continuo resultante contem pontos não viáveis (isso é, pontos que não satisfazem as condições de integralidade) no seu *interior*, e *(ii)* a solução viável ótima do problema inteiro ocorre em um ponto *extremo* do espaço (continuo) convexo. *Prova* Parte *(i)* seguindo o fato de que um ponto viável deve satisfazer a restrição $x_j = 0$ ou $x_j = 1$. Já que estes são planos limitantes do espaço de solução continuo, é impossível, por definição, que nenhum ponto pode ser um ponto interior. Parte *(ii)* é obvio para um problema zero-um puro. No caso misto, suponha que $x_j = x_j^*$, onde $x_j^* = 0$ ou 1, $j \in I$, são os valores inteiros viáveis ótimos. Ao fixar esses valores, o problema misto zero-um resultante torna-se um programa linear regular nas variáveis continuas restantes. Assim, pela teoria de programação linear, a solução ótima do problema zero-um original deve ocorrer em um ponto extremo do espaço (continuo) convexo de solução, especialmente porque $x_j = x_j^*$ são planos limitantes só conjunto convexo.

REFERÊNCIAS

AARTS, E.; KORST, J.; MICHIELS, W. Simulated annealing. In: *Search methodologies*. [S.l.]: Springer, 2005. p. 187–210.

ACHTERBERG, T. Constraint integer programming [phd thesis]. *Technical University of Berlin*, 2007.

ALAN, R. H. von; MARCH, S. T.; PARK, J.; RAM, S. Design science in information systems research. *MIS quarterly*, Springer, v. 28, n. 1, p. 75–105, 2004.

ALVES, M. J.; CLÃMACO, J. Using cutting planes in an interactive reference point approach for multiobjective integer linear programming problems. *European Journal of Operational Research*, Elsevier, v. 117, n. 3, p. 565–577, 1999.

BAI, Q. Analysis of particle swarm optimization algorithm. *Computer and information science*, v. 3, n. 1, p. p180, 2010.

BANSAL, J. C.; DEEP, K. A modified binary particle swarm optimization for knapsack problems. *Applied Mathematics and Computation*, Elsevier, v. 218, n. 22, p. 11042–11061, 2012.

BARD, J. F. *Practical bilevel optimization: algorithms and applications*. [S.l.]: Springer Science & Business Media, 2013. v. 30.

BARTHOLDI, J. J. The knapsack problem. In: *Building Intuition*. [S.l.]: Springer, 2008. p. 19–31.

BRANDÃO, J. A tabu search algorithm for the heterogeneous fixed fleet vehicle routing problem. *Computers & Operations Research*, Elsevier, v. 38, n. 1, p. 140–151, 2011.

BRATTON, D.; KENNEDY, J. Defining a standard for particle swarm optimization. In: IEEE. *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*. [S.l.], 2007. p. 120–127.

CAPRARA, A.; PISINGER, D.; TOTH, P. Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing*, INFORMS, v. 11, n. 2, p. 125–137, 1999.

CHEN, D.-S.; BATSON, R. G.; DANG, Y. *Applied integer programming: modeling and solution*. [S.l.]: John Wiley & Sons, 2010.

CHIH, M.; LIN, C.-J.; CHERN, M.-S.; OU, T.-Y. Particle swarm optimization with time-varying acceleration coefficients for the multidimensional knapsack problem. *Applied Mathematical Modelling*, v. 38.

CHOCOLAAD, C. A. *Solving Geometric Knapsack Problems using Tabu Search Heuristics.* [S.l.], 1998.

CHU, P. C.; BEASLEY, J. E. A genetic algorithm for the multidimensional knapsack problem. *Journal of heuristics*, Springer, v. 4, n. 1, p. 63–86, 1998.

CORDEAU, J.-F.; MAISCHBERGER, M. A parallel iterated tabu search heuristic for vehicle routing problems. *Computers & Operations Research*, Elsevier, v. 39, n. 9, p. 2033–2050, 2012.

COTTA, C.; TROYA, J. M. A hybrid genetic algorithm for the 0–1 multiple knapsack problem. In: SPRINGER. *Artificial neural nets and genetic algorithms*. [S.l.], 1998. p. 250–254.

CRAMA, Y.; SCHYNS, M. Simulated annealing for complex portfolio selection problems. *European Journal of operational research*, Elsevier, v. 150, n. 3, p. 546–571, 2003.

CUSSENS, J. Bayesian network learning with cutting planes. *arXiv preprint arXiv:1202.3713*, 2012.

DANTCHEV, S.; MARTIN, B. Cutting planes and the parameter cutwidth. *Theory of Computing Systems*, Springer, v. 51, n. 1, p. 50–64, 2012.

DANTZIG, G. B. Discrete-variable extremum problems. *Operations research*, INFORMS, v. 5, n. 2, p. 266–288, 1957.

DAS, S.; ABRAHAM, A.; KONAR, A. Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives. In: *Advances of Computational Intelligence in Industrial Systems*. [S.l.]: Springer, 2008. p. 1–38.

DASGUPTA, S.; PAPADIMITRIOU, C.; VAZIRANI, U. *Chapter 6 dynamic programming'. Algorithms*. 2006.

DEEP, K.; BANSAL, J. C. A socio-cognitive particle swarm optimization for multi-dimensional knapsack problem. In: IEEE. *Emerging Trends in Engineering and Technology, 2008. ICETET'08. First International Conference on*. [S.l.], 2008. p. 355–360.

DREXL, A. A simulated annealing approach to the multiconstraint zero-one knapsack problem. *Computing*, Springer, v. 40, n. 1, p. 1–8, 1988.

DUNKEL, J.; SCHULZ, A. S. A refined gomory-chvátal closure for polytopes in the unit cube. 2012.

DUPANLOUP, I.; SCHNEIDER, S.; EXCOFFIER, L. A simulated annealing approach to define the genetic structure of populations. *Molecular Ecology*, Wiley Online Library, v. 11, n. 12, p. 2571–2581, 2002.

EBERCHART, R.; KENNEDY, J. Particle swarm optimization. In: *Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia*. [S.l.: s.n.], 1995. p. 1942–1948.

ENGAU, A.; ANJOS, M. F.; VANNELLI, A. On interior-point warmstarts for linear and combinatorial optimization. *SIAM Journal on Optimization*, SIAM, v. 20, n. 4, p. 1828–1861, 2010.

FERREIRA, C.; MARTIN, A.; WEISMANTEL, R. Solving multiple knapsack problems by cutting planes. *SIAM Journal on Optimization*, SIAM, v. 6, n. 3, p. 858–877, 1996.

FERRIS, M. C.; MANGASARIAN, O. L.; WRIGHT, S. J. *Linear programming with MATLAB*. [S.l.]: SIAM, 2007. v. 7.

FLETCHER, R. *Practical methods of optimization*. [S.l.]: John Wiley & Sons, 2013.

FOMENI, F. D.; LETCHFORD, A. N. A dynamic programming heuristic for the quadratic knapsack problem. *INFORMS Journal on Computing*, INFORMS, v. 26, n. 1, p. 173–182, 2013.

FORSGREN, A.; GILL, P. E.; WRIGHT, M. H. Interior methods for nonlinear optimization. *SIAM review*, SIAM, v. 44, n. 4, p. 525–597, 2002.

GAIVORONSKI, A. A.; LISSER, A.; LOPEZ, R.; XU, H. Knapsack problem with probability constraints. *Journal of Global Optimization*, Springer, v. 49, n. 3, p. 397–413, 2011.

GANDIBLEUX, X.; FREVILLE, A. Tabu search based procedure for solving the 0-1 multiobjective knapsack problem: the two objectives case. *Journal of Heuristics*, Springer, v. 6, n. 3, p. 361–383, 2000.

GAO, S.; QIU, L.; CAO, C. Estimation of distribution algorithms for knapsack problem. *Journal of Software*, v. 9, n. 1, p. 104–110, 2014.

GEFFNER, H. Artificial intelligence: From programs to solvers. *AI Communications*, IOS Press, v. 27, n. 1, p. 45–51, 2014.

GENDREAU, M.; POTVIN, J.-Y. *Handbook of metaheuristics*. [S.l.]: Springer, 2010. v. 2.

GLOVER, F. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, Elsevier, v. 13, n. 5, p. 533–549, 1986.

____. Tabu search-part i. *ORSA Journal on computing*, INFORMS, v. 1, n. 3, p. 190–206, 1989.

GORSKI, J.; PAQUETE, L.; PEDROSA, F. Greedy algorithms for a class of knapsack problems with binary weights. *Computers & Operations Research*, Elsevier, v. 39, n. 3, p. 498–511, 2012.

GRIVA, I.; NASH, S. G.; SOFER, A. *Linear and nonlinear optimization*. [S.l.]: Siam, 2009.

HANAFI, S.; FREVILLE, A. An efficient tabu search approach for the 0-1 multidimensional knapsack problem. *European Journal of Operational Research*, v. 106, n. 2-3, p. 659 – 675, 1998. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221797002968>.

HENDERSON, D.; JACOBSON, S. H.; JOHNSON, A. W. The theory and practice of simulated annealing. In: *Handbook of metaheuristics*. [S.l.]: Springer, 2003. p. 287–319.

HERMAWANTO, D. Genetic algorithm for solving simple mathematical equality problem. *arXiv preprint arXiv:1308.4675*, 2013.

HOQUE, M. S.; MUKIT, M.; BIKAS, M.; NASER, A. et al. An implementation of intrusion detection system using genetic algorithm. *arXiv preprint arXiv:1204.1336*, 2012.

HOROWITZ, E.; SAHNI, S. Computing partitions with applications to the knapsack problem. *Journal of the ACM (JACM)*, ACM, v. 21, n. 2, p. 277–292, 1974.

HORST, R.; PARDALOS, P. M. *Handbook of global optimization, Nonconvex Optimization and its Applications*. [S.l.]: Kluwer Academic Publishers, Dordrecht, 1995.

HUANG, L.; SAGAE, K. Dynamic programming for linear-time incremental parsing. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. [S.l.], 2010. p. 1077–1086.

HUANGFU, Q.; HALL, J. Parallelizing the dual revised simplex method. *arXiv preprint arXiv:1503.01889*, 2015.

HUANGFU, Q.; HALL, J. J. Novel update techniques for the revised simplex method. *Computational Optimization and Applications*, Springer, v. 60, n. 3, p. 587–608, 2015.

IBRAHIM, W. A.; MORCOS, M. M. Artificial intelligence and advanced mathematical tools for power quality applications: a survey. *Power Delivery, IEEE Transactions on*, IEEE, v. 17, n. 2, p. 668–673, 2002.

ISLAM, M. T. et al. *Approximation algorithms for minimum knapsack problem*. Tese (Doutorado) — Lethbridge, Alta.: University of Lethbridge, Dept. of Mathematics and Computer Science, c2009, 2009.

JACKO, P. Resource capacity allocation to stochastic dynamic competitors: knapsack problem for perishable items and index-knapsack heuristic. *Annals of Operations Research*, Springer, p. 1–25, 2013.

JAMES, T.; REGO, C.; GLOVER, F. A cooperative parallel tabu search algorithm for the quadratic assignment problem. *European Journal of Operational Research*, Elsevier, v. 195, n. 3, p. 810–826, 2009.

JEON, Y.-J.; KIM, J.-C.; KIM, J.-O.; SHIN, J.-R.; LEE, K. Y. An efficient simulated annealing algorithm for network reconfiguration in large-scale distribution systems. *Power Delivery, IEEE Transactions on*, IEEE, v. 17, n. 4, p. 1070–1078, 2002.

JUKNA, S.; SCHNITGER, G. Cutting planes cannot approximate some integer programs. *Operations Research Letters*, Elsevier, v. 40, n. 4, p. 272–275, 2012.

KELLERER, H.; PFERSCHY, U.; PISINGER, D. *Knapsack problems*. [S.l.]: Springer, 2004.

KENNEDY, J.; MENDES, R. Population structure and particle swarm performance. IEEE computer Society, 2002.

KHURI, S.; BäCK, T.; HEITKöTTER, J. The zero/one multiple knapsack problem and genetic algorithms. In: *Proceedings of the 1994 ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 1994. (SAC '94), p. 188–193. ISBN 0-89791-647-6. Disponível em: <http://doi.acm.org/10.1145/326619.326694>.

KLEYWEGT, A. J.; PAPASTAVROU, J. D. The dynamic and stochastic knapsack problem. *Operations Research*, INFORMS, v. 46, n. 1, p. 17–35, 1998.

KRAFT, H.; STEFFENSEN, M. A dynamic programming approach to constrained portfolios. *European Journal of Operational Research*, Elsevier, v. 229, n. 2, p. 453–461, 2013.

KREPS, G. L.; NEUHAUSER, L. Artificial intelligence and immediacy: designing health communication to personally engage consumers and providers. *Patient education and counseling*, Elsevier, v. 92, n. 2, p. 205–210, 2013.

KUMAR, E. R. Intelligent optimization techniques for industrial applications. *IJCSI*, Citeseer, 2011.

LEUNG, S. C.; ZHANG, D.; ZHOU, C.; WU, T. A hybrid simulated annealing metaheuristic algorithm for the two-dimensional knapsack packing problem. *Computers & Operations Research*, Elsevier, v. 39, n. 1, p. 64–73, 2012.

LINHARES, A. Theory of constraints and the combinatorial complexity of the product-mix decision. *International Journal of Production Economics*, Elsevier, v. 121, n. 1, p. 121–129, 2009.

LIU, A.; WANG, J.; HAN, G.; WANG, S.; WEN, J. Improved simulated annealing algorithm solving for 0/1 knapsack problem. In: IEEE. *Intelligent Systems Design and Applications, 2006. ISDA'06. Sixth International Conference on*. [S.l.], 2006. v. 2, p. 1159–1164.

LUENBERGER, D. G.; YE, Y. *Linear and nonlinear programming*. [S.l.]: Springer, 2008. v. 116.

LUGER, G. F. *Artificial intelligence: structures and strategies for complex problem solving*. [S.l.]: Pearson education, 2005.

MARQUES, F. d. P. *O problema da mochila compartimentada*. Tese (Doutorado) — Universidade de São Paulo, 2000.

MARTELLO, S.; PISINGER, D.; TOTH, P. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, INFORMS, v. 45, n. 3, p. 414–424, 1999.

____. New trends in exact algorithms for the 0–1 knapsack problem. *European Journal of Operational Research*, Elsevier, v. 123, n. 2, p. 325–332, 2000.

MATHUR, K.; VENKATESHAN, P. A new lower bound for the linear knapsack problem with general integer variables. *European journal of operational research*, Elsevier, v. 178, n. 3, p. 738–754, 2007.

MENDES, R.; KENNEDY, J.; NEVES, J. The fully informed particle swarm: simpler, maybe better. *Evolutionary Computation, IEEE Transactions on*, IEEE, v. 8, n. 3, p. 204–210, 2004.

MITCHELL, J. E. Branch-and-cut algorithms for combinatorial optimization problems. *Handbook of applied optimization*, p. 65–77, 2002.

MUNARI, P.; GONZÁLEZ-BREVIS, P.; GONDZIO, J. A note on the primal-dual column generation method for combinatorial optimization. *Electronic Notes in Discrete Mathematics*, Elsevier, v. 37, p. 309–314, 2011.

MURTY, K. G. Solving the fixed charge problem by ranking the extreme points. *Operations Research*, INFORMS, v. 16, n. 2, p. 268–279, 1968.

NAVARRO, A.; RUDNICK, H. Large-scale distribution planning-part ii: Macro-optimization with voronoi's diagram and tabu search. *Power Systems, IEEE Transactions on*, IEEE, v. 24, n. 2, p. 752–758, 2009.

NEGNEVITSKY, M. *Artificial intelligence: a guide to intelligent systems*. [S.l.]: Pearson Education, 2005.

NEUHAUSER, L.; KREPS, G. L.; MORRISON, K.; ATHANASOULIS, M.; KIRIENKO, N.; BRUNT, D. V. Using design science and artificial intelligence to improve health communication: Chronologymd case example. *Patient education and counseling*, Elsevier, v. 92, n. 2, p. 211–217, 2013.

NOCEDAL, J.; WRIGHT, S. *Numerical Optimization*. [S.l.]: Springer, 2006.

O'LEARY, D. E. Artificial intelligence and big data. *IEEE Intelligent Systems*, IEEE Computer Society, v. 28, n. 2, p. 0096–99, 2013.

OLIVEIRA, A. R. L. de. Implementação de um método de pontos interiores para programação linear. 1991.

PARSOPOULOS, K. E.; VRAHATIS, M. N. Parameter selection and adaptation in unified particle swarm optimization. *Mathematical and Computer Modelling*, Elsevier, v. 46, n. 1, p. 198–213, 2007.

PEDERSEN, M. B.; CRAINIC, T. G.; MADSEN, O. B. Models and tabu search metaheuristics for service network design with asset-balance requirements. *Transportation Science*, INFORMS, v. 43, n. 2, p. 158–177, 2009.

PISINGER, D. Algorithms for knapsack problems. Citeseer, 1995.

____. Linear time algorithms for knapsack problems with bounded weights. *Journal of Algorithms*, Elsevier, v. 33, n. 1, p. 1–14, 1999.

____. Where are the hard knapsack problems? *Computers & Operations Research*, Elsevier, v. 32, n. 9, p. 2271–2284, 2005.

PISINGER, W. D.; RASMUSSEN, A. B.; SANDVIK, R. Solution of large quadratic knapsack problems through aggressive reduction. *INFORMS Journal on Computing*, INFORMS, v. 19, n. 2, p. 280–290, 2007.

POLI, R.; KENNEDY, J.; BLACKWELL, T. Particle swarm optimization. *Swarm intelligence*, Springer, v. 1, n. 1, p. 33–57, 2007.

QIAN, F.; DING, R. Simulated annealing for the 0/1 multidimensional knapsack problem. *NUMERICAL MATHEMATICS-ENGLISH SERIES*-, Citeseer, v. 16, n. 4, p. 320, 2007.

RAIDL, G. R. An improved genetic algorithm for the multiconstrained 0-1 knapsack problem. In: IEEE. *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. [S.l.], 1998. p. 207–211.

RAO, S. S. *Engineering optimization: theory and practice*. [S.l.]: John Wiley & Sons, 2009.

RONG, A.; FIGUEIRA, J. R. Dynamic programming algorithms for the bi-objective integer knapsack problem. *European Journal of Operational Research*, Elsevier, v. 236, n. 1, p. 85–99, 2014.

SASTRY, K.; GOLDBERG, D.; KENDALL, G. Genetic algorithms. In: *Search methodologies*. [S.l.]: Springer, 2005. p. 97–125.

SCHUETZ, H.-J.; KOLISCH, R. Approximate dynamic programming for capacity allocation in the service industry. *European Journal of Operational Research*, Elsevier, v. 218, n. 1, p. 239–250, 2012.

SIMON, H. A. Artificial intelligence: an empirical science. *Artificial Intelligence*, Elsevier, v. 77, n. 1, p. 95–127, 1995.

TAHA, H. A. *On the solution of zero-one linear programs by ranking the extreme points*. [S.l.]: Associate Professor of Industrial Engineering, 1972.

____. *Integer programming: theory, applications, and computations*. [S.l.]: Academic Press, 1974.

VINCENT, F. Y.; LIN, S.-W.; LEE, W.; TING, C.-J. A simulated annealing heuristic for the capacitated location routing problem. *Computers & Industrial Engineering*, Elsevier, v. 58, n. 2, p. 288–299, 2010.

WHITLEY, D. A genetic algorithm tutorial. *Statistics and computing*, Springer, v. 4, n. 2, p. 65–85, 1994.

WRIGHT, S. J.; NOCEDAL, J. *Numerical optimization*. [S.l.]: Springer New York, 1999. v. 2.

ZHANG, C.; WANG, J.; XIE, W.; ZHOU, G.; LONG, M.; ZHANG, Q. Dynamic programming procedure for searching optimal models to estimate substitution rates based on the maximum-likelihood method. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 108, n. 19, p. 7860–7865, 2011.

ZHAO, C.; LI, X. Approximation algorithms on 0–1 linear knapsack problem with a single continuous variable. *Journal of Combinatorial Optimization*, Springer, p. 1–7, 2013.

ZHONG, T.; YOUNG, R. Multiple choice knapsack problem: Example of planning choice in transportation. *Evaluation and program planning*, Elsevier, v. 33, n. 2, p. 128–137, 2010.