Otimização do Problema da Mochila

Érica da Costa Reis Carvalho[1], **Michelli Marlane da Silva**[1]

[1]Departamento de Ciência da Computação – Universidade Presidente Antônio Carlos (UNIPAC)
Rua Palma Bageto Viol S/N – Barbacena – MG – Brasil

ericacrcarvalho@yahoo.com.br, michelli1983@yahoo.com.br

Abstract. *This work, we studied a famous combinatorial optimization problem known as the knapsack problem. The problem is integral to the class NP-hard and can be stated as a situation where you need to carry a backpack containing a set of objects of different weights and values. Therefore, this backpack has limited capacity and should be maximized the total value, so that objects do not exceed the knapsack. The proposed methodology involves the solution through a heuristic and a metaheuristic, and the Greedy Algorithm and VNS, respectively, and an exact solution using dynamic programming. The knapsack problem has well known applications, such as problems of packing, loading equipment, cutting materials, budgetary control and selection of investment projects.*

Key words: *Knaspsack Problem, Heuristics, Metaheristic, Optimization.*

Resumo. *Neste trabalho, será estudado um famoso problema de otimização combinatória conhecido como Problema da Mochila. O problema é integrante da classe NP-difícil e pode ser enunciado considerando uma situação onde é necessário carregar uma mochila contendo um conjunto de objetos de pesos e valores diferentes. Logo, essa mochila possui capacidade limitada e deve ser maximizado o valor total, fazendo com o que os objetos não ultrapassem o limite da mochila. A metodologia proposta envolve a solução através de uma heurística e uma metaheurística, sendo os Algoritmos Gulosos e o VNS, respectivamente e uma solução exata utilizando a Programação Dinâmica. O problema da Mochila possui aplicações muito conhecidas, como em problemas de embalagem, de carregamento de equipamentos, de corte de materiais, de controle orçamental e de seleção de projetos de investimento.*

Palavras chave: *Problema da Mochila, Heurísticas, Metaheurística, Otimização.*

1. Introdução

O problema da mochila ou *Knapsack problem*, constitui uma classe de problemas dos mais estudados em otimização combinatória, e em subproblemas de outros problemas práticos possuindo diversas variantes agrupadas no que podem ser chamadas de famílias, ou classes de mochilas [Fincatti 2010] e [Spolador 2005]. É, também, um dos principais problemas da área de Pesquisa Operacional, visto que muitas situações onde se busca maximizar o lucro podem ser reduzidas a resolver este problema. Sua resolução demanda um grande esforço computacional, tornando-se especialmente interessante para a área de Programação Paralela [Mor 2007].

O nome surgiu devido o modelo de uma situação em que é necessário carregar uma mochila com capacidade limitada, com um conjunto de objetos com pesos e valores diferentes. O objetivo é ocupar a mochila com o maior valor possível, não ultrapassando o seu peso máximo. Assim, toda mochila possui uma restrição quanto a sua capacidade, onde a soma do peso dos itens selecionados deve respeitar a capacidade da mochila e ao mesmo tempo maximizando o seu valor total (quantidade de objetos carregados), corresponde a resolver o problema da mochila [Fincatti 2010] e [Spolador 2005].
O problema da mochila é integrante da classe NP-difícil e é um dos 21 problemas NP-completos de Richard Karp, informático estadunidense que publicou diversos trabalhos fundamentais sobre a complexidade computacional. O fato do Problema da Mochila ser NP-Completo permite que os resultados obtidos possam ser generalizados para toda a classe de problemas NP-Completos [Mor 2007].
Um problema NP-completo pertence à classe NP (não-polinomiais) e são os mais interessantes, pois não se conhece uma solução determinística capaz de ser executada em tempo polinomial. Já um problema NP-difícil é uma classe de problemas que são, informalmente, pelo menos tão difíceis quanto os problemas mais difíceis em NP [UFMG/ICEx/DCC 2008].
Devido à sua complexidade exponencial, uma solução eficiente para o problema deve ser aquela que consegue reduzir, ao máximo, os termos da expressão exponencial. Uma das aplicações mais conhecidas é o problema do corte e empacotamento, que consistem em atender uma demanda de itens a partir do corte de material, em geral, com o objetivo de minimizar a perda, ou seja, deve-se determinar a 'melhor' forma de cortar unidades maiores (objetos) de maneira a produzir um conjunto de unidades menores (itens) [Boleta and Poldi 2005].
A Programação Dinâmica de Bellman é uma técnica computacional apoiada no Princípio de Otimalidade, ou seja, suas características têm de serem tais que o conhecimento do estado corrente do sistema contenha toda a informação à cerca do seu prévio comportamento, necessária à determinação do plano ótimo a partir dele. Frequentemente, as soluções para a Programação Dinâmica são obtidas por um processo regressivo, trabalhando do fim para o princípio [Ferreira] e [Santos]. Ela foi a primeira técnica mais eficiente para resolução do problema da Mochila, logo, várias técnicas de resolução têm sido desenvolvidas, em geral especializando procedimentos consagrados da Pesquisa Operacional, tais como: relaxação lagrangeana, busca em grafos e heurísticas. Por ser um problema combinatório, pode ser resolvido por enumeração exaustiva, ou seja, tentando todas as soluções possíveis, e posteriormente comparando-as para obter a melhor solução. Porém, a terminação exaustiva destas soluções é completamente inviável, mesmo em pequenas dimensões. Por exemplo, um problema com apenas 20 itens pode possuir aproximadamente 1.048.576 soluções. [Fincatti 2010] ainda ressalva a existência de diversos métodos para análise desse tipo de problema que podem ser classificados como exatos ou aproximados. Uma das características destes problemas é o aumento do número de soluções existentes o que dificulta sua resolução. São eles:

- **Problema da mochila 0/1:** cada item pode ser escolhido no máximo 1 vez;
- **Problema da mochila limitado:** há uma quantidade limitada para cada tipo de item;
- **Problema da mochila com múltipla escolha:** os itens devem ser escolhidos de classes disjuntas;

- **Problema da mochila múltiplo:** várias mochilas são preenchidas simultaneamente;
- **Problema da mochila com multi-restrições:** problema de programação inteira geral com coeficientes positivos.

Este trabalho tem por objetivo principal encontrar uma heurística que melhor se adapte ao problema da Mochila. Logo, deseja-se analisar os métodos Algoritmos Gulosos, VNS e Programação Dinâmica para obtenção do melhor resultado. Por fim, objetiva-se a comparação de tais métodos propostos.
A otimização vem ganhando muito respaldo nas últimas décadas, sendo alvo de muitas pesquisas devido à sua importância operacional e, sobretudo, econômica. Logo, a principal motivação para o trabalho proposto é a busca por uma solução viável que satisfaça o problema da Mochila, envolvendo métodos e heurísticas. Para isso, foi feito um estudo aprofundado e detalhado de alguns desses métodos e heurísticas, já existentes. Uma aplicação prática conhecida e bastante estudada é o problema do corte e empacotamento. Esse problema consiste em atender uma demanda de itens a partir do corte de material, em geral, com o objetivo de minimizar a perda, ou seja, deve-se determinar a 'melhor' forma de cortar unidades maiores (objetos) de maneira a produzir um conjunto de unidades menores (itens). Para tanto, deve-se encontrar um conjunto de padrões de corte que serão repetidos um certo número de vezes e que satisfaçam algum critério a ser otimizado, por exemplo, minimizar a perda de material gerada pelo corte dos padrões. [Boleta and Poldi 2005]
Outra motivação a ser levada em consideração é a análise e comparação entre as heurísticas já existentes. A busca por uma solução mais econômica em termos de matéria-prima, mas que não perca em eficiência é cada vez mais aceitável no mercado.

2. Estado da arte

Problema da Mochila
Segundo [Luila 2008], O Problema da Mochila caracteriza uma classe de problema de programação linear inteira e são classificados na literatura, segundo a sua complexidade de resolução, como problemas NP-Difícil. O problema é um dos mais importantes em programação linear inteira e tem sido estudado intensivamente nos últimos anos por vários investigadores.
A primeira resolução do problema da mochila, por técnicas mais inteligentes, data dos anos 50, foi a aplicação da função recursiva. A partir de então, foram propostos inúmeros melhoramentos: a definição do limite superior para o valor ótimo da função objetivo, por Dantzig em 1957; a resolução do problema pela técnica de partição e avaliação sucessiva, por Kolesar em 1967; a resolução de problemas de grandes dimensões, igualmente pela técnica de partição e avaliação sucessiva, por Harowitz e Sahni, nos anos 70; o primeiro procedimento de redução da dimensão do problema de Ingargiola e Korsh em 1973; um novo limite superior, por Martello e Toth, em 1977; o algoritmo de Balas e Zemel, em 1980, baseado na ordenação de apenas um subconjunto de itens; um novo algoritmo de Martello e Toth que permite resolver instâncias difíceis.
[Arruda 2002], ainda enuncia o Problema da Mochila de forma simples: Um viajante levará consigo apenas uma mochila para sua viagem. Sua mochila possui uma dada capacidade e deve ser preenchida com alguns objetos que lhe sejam úteis durante a viagem. Cada objeto possui um peso e um dado valor para o viajante e este possui apenas uma

unidade de cada objeto a ser escolhido. Quais objetos devem ser levados pelo viajante em sua mochila de forma a maximizar o valor da mochila?
Entende-se como peso do objeto sendo o peso real que um objeto possui. Logo, entende-se como valor, o benefício que cada objeto possui, ou seja, qual a importância que o objeto tem em um determinado cenário ou em uma determinada situação.
Seja *wj* o peso do *j* -ésimo objeto e pj seu valor. Se o objeto *xj* aparece na mochila, então *xj* = 1 caso contrário *xj* = 0. [Arruda 2002] propõem que se denotarmos por c a capacidade da mochila, e por n a quantidade de objetos disponíveis para escolha e como o maior valor obtido para a mochila de capacidade c usando os n objetos, então o problema pode ser formulado algebricamente como a seguir:

sujeito à

$$\text{F(c)} = \max \sum p_j x_j (p_j > 0)$$

$$\sum w_j x_j <= c(w_j, c > 0)$$

Segundo [Marques and Arenales 2002], "O Problema da Mochila consiste em determinar as capacidades adequadas de cada compartimento e como estes devem ser carregados de modo que o valor de utilidade total (soma dos valores de utilidade de todos os itens selecionados) seja máximo, descontando-se os custos dos compartimentos, os quais dependem dos agrupamentos com que foram preenchidos. Ou seja, esse é um clássico problema e pode ser exemplificado considerando-se a seguinte situação hipotética: um alpinista deve carregar sua mochila com possíveis itens de seu interesse. A cada item, atribuem-se o seu peso e um valor de utilidade."[Fincatti 2010], sintetiza o problema da Mochila como sendo um dos mais estudados em otimização combinatória e em subproblemas de outros problemas práticos. Muitos problemas reais podem ser formulados com o problema da Mochila ou uma variação deste. Por ser um problema combinatório, pode ser resolvido por enumeração exaustiva, ou seja, tentando todas as soluções possíveis, e posteriormente comparando-as para obter a melhor solução. "O problema da mochila pode ser aplicado conforme os exemplos:

- Um viajante deve levar consigo apenas uma mochila. Essa mochila possui uma capacidade limitada e deve ser carregada apenas com objetos que serão úteis durante a viagem. Cada objeto é único e possui um peso e um determinado valor. Que objetos dever ser levados pelo viajante de forma a maximizar o valor da mochila?
- Um contêiner com capacidade limitada deve ser carregado com diversos produtos de pesos e tamanhos diferentes. Como se deve proceder para carregar o máximo possível de produtos, desperdiçando o mínimo possível de espaço?
- Um computador está sobrecarregado de arquivos e os mesmos devem ser transferidos para mídias em CD, e sabe-se que será necessário mais de um CD. Como se deve proceder para carregar o máximo possível de arquivos em cada CD, desperdiçando o mínimo possível de espaço em cada mídia?"

Variações do Problema da Mochila

Segundo [Spolador 2005], o Problema da Mochila é amplamente estudado, possuindo diversas variantes agrupadas no que podem ser chamadas de famílias, ou classes de mochilas. "Ele se caracteriza basicamente pela escolha de um subconjunto de itens que irá otimizar um objetivo. Para tanto, cada item deve possuir um "peso"e uma "utilidade".

Por peso entende-se um custo, uma penalidade a ser paga, pelo uso do item, por exemplo, quando se escolhe arquivos para compor uma mídia o peso representa o tamanho do arquivo. A utilidade é o lucro, um benefício, que a escolha do item contribuirá no objetivo do problema. Toda mochila possui uma restrição quanto a sua capacidade, de fato, a soma do peso dos itens selecionados deve respeitar a capacidade da mochila."
[Mor 2007] enuncia que existem outros tipos de Problema da Mochila, com muitas variações nas restrições e condições de execução (Problema da Mochila Compartimentada, Problema da Mochila Multidimensional, que irão ser comentadas mais adiante). É interessante analisar, entretanto, três variações do Problema da Mochila que podem ser obtidas adicionando, cada uma, uma pequena restrição no Problema da Mochila original. São elas: O Problema da Mochila Limitado, o Problema da Mochila Ilimitado e o Problema da Mochila 0-1.

- **Problema da Mochila Limitado**: Um assaltante está furtando uma residência. Na residência, o número de itens que pode ser levado do mesmo tipo tende a ser limitado; é improvável, por exemplo, que existam mais do que três ou quatro televisores. Assim cada item pode ser incluído no somatório um número limitado de vezes.
- **Problema da Mochila Ilimitado**: Já o Problema da Mochila Ilimitado, não há número de itens de cada tipo (o valor de xi) máximo.
- **Problema da Mochila 0-1**: No mesmo cenário dos anteriores, o ladrão, desta vez, atenta em roubar uma loja de raros vasos chineses. Tal loja tem apenas um tipo de cada vaso e, deste modo, a decisão do ladrão é entre levar o item ou não.

Otimização

Segundo [Bennaton 2001], "Otimização é melhorar até onde pudermos. No ideal, melhorar até o máximo. Ou seja, melhorar até aquela situação ideal na qual, como vulgarmente se diz, "se mexer mais, piora". Melhorar só é possível se temos escolha. Escolher uma dentre várias alternativas. Se uma alternativa houver, capaz de introduzir alguma melhoria, ficamos com ela. Caso contrário, o que temos em mãos já é a escolha ótima. O autor acrescenta Otimizar é selecionar algo melhor. Mas, quase sempre, ficamos restritos a escolhê-lo dentre um conjunto limitado de alternativas. Obviamente, o desejo de otimizar não basta. Sem critério de escolha, por exemplo, nem adianta conhecer o universo de alternativas. Por outro lado, desconhecendo-se este, não adianta ter critério. Informação, portanto, é fundamental. Quanto mais, melhor; mais depressa chegamos às alternativas ótimas.
Otimização acabou se constituindo numa vasta e sólida área do conhecimento. Uma área híbrida, eclética e pragmática. Com um pé na matemática e outro pé na computação. Que se nutre das ciências exatas, das biológicas, das tecnológicas. Que se dedica a solucionar problemas independentemente do contexto onde surgem.
.

Heurística

Para [Ribeiro 2010], Heurística é usada em algoritmos que acham soluções para um dado problema. Tais abordagens não garantem a obtenção da solução ótima, ou seja, elas não necessariamente obtêm as melhores soluções dentre as possibilidades existentes. Porém, algoritmos que usam heurísticas acham frequentemente soluções muito próximas da ótima, além de fazê-lo de forma rápida e fácil.

O uso de heurísticas aplica-se em problema que se deseja obter respostas com um grau de precisão bom. Porém, em problemas nos quais a solução obtida deve ser necessariamente a melhor, o uso de heurísticas não é adequado.

.

Metaheurística

[Martins 2010] anuncia como sendo Metaheurísticas técnicas usadas em situações que podem ser modeladas como problemas de maximizar (ou minimizar) uma função cujas variáveis têm certas restrições. São estratégias comumente utilizadas para resolver problemas NP - difíceis por oferecerem melhores soluções e geralmente com tempo de processamento menor do que por outros tipos de técnicas. De forma geral, utilizam combinação de escolhas aleatórias e conhecimento histórico (dos resultados anteriores adquiridos pelo método) para se guiarem e realizar suas buscas pelo espaço de pesquisa em vizinhanças dentro do espaço de pesquisa, o que evita paradas prematuras em ótimos locais.
Para [Chaves 2003] as Metaheurísticas podem ser divididas em:

- Metaheurística de busca por entornos: percorrem o espaço de busca levando em conta, fundamentalmente, a vizinhança da solução em mãos, definida como o conjunto de soluções que podem ser obtidas a partir da aplicação de algum operador à solução atual;
- Metaheurística de relação: simplificam o problema (criando um problema relaxado) e utilizam a solução encontrada como guia para o problema original;
- Metaheurísticas construtivas: definem de forma meticulosa o valor de cada componente da solução;
- Metaheurísticas evolutivas: lidam com uma população de soluções, que evolui, principalmente, através da interação entre seus elementos.

3. Metodologia

[Mor 2007] enuncia sendo Problema da Mochila um problema de otimização combinatória de especial importância em Ciência da Computação e pertence à classe de problemas NP-Completos, ou seja, é integrante da classe NP-difícil, conforme [UFMG/ICEx/DCC 2008] é uma classe de problema que não se conhece uma solução determinística capaz de ser executada em tempo polinomial. Para isso, a metodologia de estudo é a definição de uma heurística, uma metaheurística e um resultado exato, são eles: Algoritmos Gulosos, VNS e Programação Dinâmica, respectivamente.

3.1. Algoritmos

3.1.1. Algoritmos Gulosos

Segundo [Rocha and Dorini 2004] os algoritmos relacionados com otimização lidam com uma sequência de passos, sendo que em cada passo há um conjunto de escolhas/opções. Uma estratégia para resolver problemas de otimização são os algoritmos gulosos, os quais escolhem a opção que parece ser a melhor no momento (escolha ótima), e esperam que desta forma consiga-se chegar a uma solução ótima global. Embora nem sempre seja possível chegar a uma solução ótima a partir da utilização de algoritmos gulosos, eles são eficientes em uma ampla variedade de problemas. Os algoritmos gulosos tomam decisões com base apenas na informação disponível, sem se preocupar com os efeitos futuros de tais decisões, isto é, eles nunca reconsideram as decisões tomadas, independentemente das

consequências. Não há, portanto, necessidade de avaliar as alternativas nem de empregar procedimentos elaborados permitindo que decisões anteriores sejam desfeitas. Devido a tais características, de forma geral eles são fáceis de se ”inventar”e implementar, e são eficientes quando funcionam. A estratégia consiste então em selecionar convenientemente objetos de tal forma a colocar a maior fração possível do objeto selecionado na mochila, e parar quando a mochila estiver totalmente cheia.
O método Guloso, segundo [CALDAS 2004] é a técnica de projeto mais simples que pode ser aplicada a uma grande variedade de problemas. O método Guloso sugere que podemos construir uma algoritmo que trabalhe em estágios, considerando uma entrada por vez. Em cada estágio, uma decisão é tomada considerando se uma entrada particular é uma solução ótima. Isto é conseguido considerando as entradas numa ordem determinada por algum processo de seleção. Se a inclusão da próxima entrada na solução ótima construída parcialmente resultará numa solução inviável, então esta entrada não será adicionada a solução parcial. O processo de seleção em si é baseada em alguma medida de otimização. Esta medida pode ou não ser a função objetivo. Na verdade várias medidas de otimização diferentes podem aplicáveis para um determinado problema. Muitas destas, entretanto, resultarão em algoritmos que gerarão solução sub-ótimas. O algoritmo usa estratégias gulosas mais interessante, estas estratégias fazem uma negociação entre a taxa em que a utilidade decresce e o peso é usado. Em cada passo será incluído um objeto que tem a maior utilidade por unidade de peso usado. Isto significa que o objeto será considerado na ordem decrescente da Maior Utilidade sobre o Peso (*U(i)=P(i)*). Se os objetos estiverem ordenado numa ordem decrescente de Utilidade sobre o Peso (U(i) = P(i) = U(i + 1)=P(i + 1)), o algoritmo resolve o problema da Mochila utilizando a estratégia gulosa. Sem considerar o tempo de ordenação da entrada, o algoritmo executa a estratégia gulosa em tempo O(N). Esse algoritmo realiza os seguintes passos:

- O vetor solução X é inicializado;
- A capacidade utilizada da mochila é armazenado em cum;
- O primeiro laço FOR, coloca cada ítem na mochila, subtraindo o peso do ítem do valor de cum. Isso é feito até que a cum não comporte o próximo peso. Neste ponto todos os ítem de maior relação U(i) = P(i) foram colocados na Mochila.
- O segundo laço FOR muda de estratégia e procura nos ítens restantes aquele(s) que possua um peso que caiba no valor restante de cum para preencher o espaço da Mochila.

Metodo Guloso(U[1..n], P[1..n],M, n)
1: X $\leftarrow 0$
2 : $cum \leftarrow M$
3 : **for**$i \leftarrow 1ton$
4 : **do**
5 : **if**$P(i) > cum$
6 : **then**$Exit$;
7 : $X(i) \leftarrow 1$;
8 : $cum \leftarrow cum - P(i)$
9 : **for**$j \leftarrow iton$
10 : **do**
11 : **if**$P(j) \leq cum$

12 : **then**$X(i) \leftarrow 1$;
13 : $cum \leftarrow cum - P(j)$

3.1.2. VNS

Variable Neighborhood Search, segundo [Erito M. S. Filho 2007] é uma metaheurística proposta por Hansen e Mladenovic (1997) e se baseia em um princípio simples: mudança sistemática da estrutura de vizinhança dentro da busca. [André 2007] ainda anuncia que a VNS tem o objetivo de descer até mínimos locais e tentar escapar dos vales que os contêm. Esta técnica faz uso da pesquisa local, ou seja, existe uma movimentação de uma solução para outra de acordo com regras pré estabelecidas.
O VNS pode ser considerado uma metaheurística moderna com o objetivo de resolver problema de otimização e foi aplicada com sucesso para resolver diversos problemas como o p-mediano, o problema de roteamento com várias estações e muitos outros problemas clássicos.
Para [Sucupira], o Método de Pesquisa em Vizinhança Variável (Variable Neighborhood Search, VNS), Mladenovic Hansen (1997), é um método de busca local que consiste em explorar o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança. Contrariamente a outras metaheurísticas baseadas em métodos de busca local, o método VNS não segue uma trajetória, mas sim explora vizinhanças gradativamente mais ”distantes”da solução corrente e focaliza a busca em torno de uma nova solução, se e somente se, um movimento de melhora é realizado. O método inclui, também, um procedimento de busca local a ser aplicado sobre a solução corrente. Esta rotina de busca local também pode usar diferentes estruturas de vizinhança.
Um algoritmo VNS pode ser descrito a seguir:

1: **Procedimento** VNS
2: Seja so uma solução inicial;
3: Seja r o número de estruturas diferentes de vizinhança
4: s $\leftarrow so$;
5 : **Enquanto**$critériodeparadanãosatisfaça$**faça**
6 : $k \leftarrow 1$
7 : **Enquanto**$k <= r$**faça**
8 : $Gereumvizinhoqualquers'pertenceteaN(k)(s)$
9 : $s'' \leftarrow BuscaLocal(s')$
10 : **Se**$f(s'') < f(s)$**entao**
11 : $s \leftarrow s''$
12 : $k \leftarrow 1$
13 : **Senao**
14 : $k \leftarrow k + 1$
15 : **Fim Enquanto**
16 : **Fim Enquanto**
17 : **Retorne**s
18 : **Fim Procedimento**

Nesse algoritmo, parte-se de uma solução inicial qualquer e a cada iteração selecionase aleatoriamente um vizinho s' dentro da vizinhança N(k)(s) da solução s corrente. Esse vizinho é então submetido a um procedimento de busca local. Se a solução ótima local, s", for melhor que a solução s corrente, a busca continua de s" recomeçando da primeira estrutura de vizinhança N(1)(s). Caso contrário, continua-se a busca a partir da próxima estrutura de vizinhança N(k+1)(s). Este procedimento é encerrado quando uma condição de parada for atingida, tal como o tempo máximo de iterações consecutivas entre dois melhoramentos. A solução s' é gerada aleatoriamente, de forma a evitar ciclagem, situação que pode ocorrer se alguma regra determinística for usada.

3.1.3. Programação Dinâmica

Segundo [CAMPONOGARA] a Programação Dinâmica é uma técnica poderosa para resolver problemas de decisão sequencial e de controle, que consistem em quebrá-los em subproblemas menores, mais fáceis de serem resolvidos. Quando quebramos um problema em subproblemas do mesmo tipo, tipicamente podemos produzir algoritmos recursivos,frequentemente empregados para resolver problemas computacionais. Esse paradigma segue alguns passos:

- Remove um elemento do problema;
- Resolve o problema menor;
- Usa a solução do problema menor para adicionar o elemento removido de maneira adequada, produzindo uma solução para o problema maior.

Para [CALDAS 2004], Programação dinâmica pode ainda ser enunciada como uma técnica que tem como objetivo diminuir o número de seqüências de decisões. Para que o paradigma da Programação Dinâmica possa ser aplicado, [FEOFILOFF 2001] anuncia que é preciso que o problema tenha estrutura recursiva: a solução de toda instância do problema deve "conter"soluções de subinstâncias da instância. Essa estrutura recursiva pode, em geral, ser representada por uma recorrência e a recorrência pode ser traduzida em um algoritmo recursivo. O algoritmo recursivo é tipicamente ineficiente porque refaz, muitas vezes, a solução de cada subinstância. Uma vez escrito o algoritmo recursivo, entretanto, é fácil construir uma tabela para armazenar as solução das subinstância e assim evitar que elas sejam recalculadas. Nota: A palavra programação na expressão programação dinâmica não tem relação direta com programação de computadores. Ela significa planejamento e refere-se à construção da tabela que armazena as soluções das subinstâncias.
Programação dinâmica, segundo [CALDAS 2004] é uma técnica que tem como objetivo diminuir o número de seqüências geradas. A programação dinâmica trata o número de combinações da seguinte forma: Vamos considerar n ítens, dentre os quais devemos escolher r. Para escolher os r ítens, podemos proceder de duas formas:

1. Escolher o primeiro item. Escolher depois r - 1 itens dos n - 1 itens restantes.
2. Não escolher o primeiro item. Então devemos escolher r itens dos n - 1 itens restantes

Está solução pode ser traduzida da seguinte forma: Se usarmos uma estratégia de divisão e conquista para implementar uma solução obteremos um algoritmo com

complexidade O(2n), sendo o maior problema desta abordagem o número de vezes que o mesmo problema é resolvido. Uma outra abordagem seria usar uma tabela para armazenar as soluções que vão se repetir, e é essa a idéia principal da programação dinâmica. Usando esta abordagem para o problema podemos melhor a complexidade de problemas para O(n2).

Segue o algoritmo:

Metodo Dinamico(v[1..n], w[1..n], n,W)
1: **for** w $\leftarrow 0toW$
2 : **do**
3 : $c[0, w] \leftarrow 0$
4 : **for**$i \leftarrow 1ton$
5 : **do**
6 : $c[i, 0] = 0$
7 : **for**$i \leftarrow 1ton$
8 : **do**
9 : **for**$w \leftarrow 1toW$
10 : **do**
11 : **if**$w[i] \leq w$
12 : **then**
13 : **if**$v[i] + c[i-1, w-w[i]] > c[i-1, w]$
14 : **then**
15 : $c[i; w] = v[i] + c[i-1, w-w[i]]$
16 : **else**$c[i, w] = c[i-1, w]$
17 : **else**$c[i; w] = c[i-1, w]$

Um exemplo de aplicação usando a Programação Dinâmica é o programa para cálculos numéricos chamado VCN (Visual Cálculo Numérico). O VCN é um programa que oferece mais de 100 opções de cálculo para estudantes e profissionais de engenharia, computação, matemática ou qualquer curso da área de Exatas. VCN implementa opções de Tabelamento de Funções (Algoritmos de Parser); Erros e Representação Numérica; Operadores Numéricos (Diferenças Finitas Ascendente, Descendente e Central); Interpolação e Extrapolação Numérica; Derivação Numérica; Integração Numérica; Equações Diferenciais; Matrizes e Sistemas Lineares; Cálculo de Raízes e Zero de Funções; Sistemas não Lineares; Ajuste de Curvas; Aproximações de Funções; Otimização (Programação Linear, Inteira e etc.). Utilizaremos o programa VCN na opção Otimização, Programação Dinâmica - problema de Knapsack, com a finalidade de buscar uma possível solução para o problema da Mochila proposto.

3.2. Entrada de dados

Foram feitos 3 testes para cada método proposto, Algoritmo Guloso, VNS e Programação Dinâmica (VCN), utilizando a mesma entrada de dados em cada teste. Segue:

- **Entrada 1**
  A capacidade da Mochila
  C= 364

  Um vetor de 15 posições armazenou os seguintes pesos associados:
  p = [82 26 42 36 70 77 36 77 55 10 96 23 2 82 14]

  Um vetor de também 15 posições armazenou os seguintes benefícios associados:
  b = [92 29 37 37 77 83 28 83 52 5 106 31 -4 89 14]

- **Entrada 2**
  A capacidade da Mochila
  C= 698

  Um vetor de 30 posições armazenou os seguintes pesos associados:
  p = [82 26 42 36 70 77 36 77 55 10 96 23 2 82 14 22 50 69 39 71 44 96 -2 51 65 9 50 91 65 25]

  Um vetor de também 30 posições armazenou os seguintes benefícios associados:
  b = [92 29 37 37 77 83 28 83 52 5 106 31 -4 89 14 21 44 90 45 78 31 101 5 40 66 11 59 88 73 29]

- **Entrada 3**
  A capacidade da Mochila
  C= 1590

  Um vetor de 60 posições armazenou os seguintes pesos associados:
  p = [82 26 42 36 70 77 36 77 55 10 96 23 2 82 14 22 50 69 39 71 44 96 -2 51 65 9 50 91 65 25 15 22 69 101 18 58 96 79 63 11 1 23 37 61 88 12 20 85 93 82 86 16 77 81 90 100 40 92 64 -1]

  Um vetor de também 60 posições armazenou os seguintes benefícios associados:
  b = [92 29 37 37 77 83 28 83 52 5 106 31 -4 89 14 21 44 90 45 78 31 101 5 40 66 11 59 88 73 29 11 25 75 108 22 53 92 84 57 10 2 27 32 66 94 9 17 81 99 73 83 15 68 86 96 105 47 90 60 1]

3.2.1. Algoritmo Guloso

O algoritmo usado foi adaptado do artigo [CALDAS 2004]. O artigo trás um código fonte contendo três heurísticas (Backtracking, Programação Dinâmica e Algoritmos Gulosos) na linguagem de programação C. A partir daí, o código foi modificado e adaptado para ser utilizado apenas para a obtenção de resultados dos Algoritmos Gulosos. O código utilizando continua na linguagem C.

3.2.2. VNS

Para a metaheurística VNS foi utilizado um algoritmo já implementado em MATLAB que pode ser encontrado em [Brito 2010].

3.2.3. Programação Dinâmica (VCN)

Para se obter um cálculo mais exato do problema da Mochila, utilizou-se o programa para Cálculos Numéricos denominado VCN - Cálculo Numérico 5.1. Ele está disponível para download em [PUC Minas] e é bastante fácil de ser utilizado.

4. Resultados

Os resultados obtidos foram satisfatórios e podem ser observados de uma maneira mais claro na Figura 1 através de um gráfico que analisa o desempenho de cada método e cada resultado de acordo com o peso e o benefício de cada item da Mochila. Ao final da análise de cada teste proposto, observa-se que o VNS mostrou um resultado mais real e satisfatório. O algoritmo em MATLAB obteve o melhor resultado em todos os testes, ou seja, o VNS obteve um valor mais aproximado da capacidade total da mochila e um benefício dos itens melhor.
O algoritmo guloso, por sua vez, gerou um resultado bom, porém, não o melhor.
A Programação Dinâmica não obteve um resultado tão satisfatório. Com isso, pode-se confirmar a teoria que os problemas NP-completos, como é o caso do Problema da Mohcila, não podem ser resolvidos através de soluções exatas (métodos exatos) e sim, conforme mencionado, através de heurísticas que buscam achar um resultado mais aproximado e que não perca em eficiência.

Figura 1. Gráfico com os resultados da análise dos três métodos

5. Conclusões

Neste artigo foi discutido um problema de Otimização bastante estudado denominado Problema da Mochila. Os problemas de Otimização buscam soluções que exigem a determinação de valores máximos ou mínimos absolutos das funções que os representam. O problema da Mochila pode ser considerado um problema de Otimização pois as soluções encontradas são as melhores possíveis, ou seja, é possível resolver este problema com as técnicas de máximos buscando encontrar uma solução ótima para ele.
Para resolver o problema da Mochila foram propostos três métodos: uma Heurística denominada Algoritmos Gulosos, uma Metaheurística denominada VNS e um cálculo exato através da Programação Dinâmica utilizando o programa VCN - Cálculo Numérico 5.1.

Destaca-se que o objetivo principal deste artigo foi o estudo de métodos que satisfação um problema de Otimização que pertence a classe de problemas NP-completos, ou seja, não existe um algoritmo totalmente eficiente que resolva o tal problema em tempo polinomial.
No artigo, não se preocupou com a criação de um outro método matemático que resolva o problema da Mochila, e sim, a comparação de métodos já existentes e já estudados com o objetivo de obter o melhor método e/ou o que satisfaça com mais exatidão a otimização proposta.
Como desenvolvimentos futuros podem ser considerados:

- Analisar outro método bastante conhecido e de muita importância na área de Otimização, conhecido como Algoritmo Genético;
- Incorporar mais complexidade para a entradada de dados nos métodos propostos no artigo;
- Avaliar a possibilidade de um outro método baseado em Programação Dinâmica afim de obter um resultado mais satisfatório do que o encontrado.

Referências

André, N. M. D. S. (2007). Variable neighborhood search.

Arruda, F; Goldma, A. (2002). Eliminação paralela de termos dominantes no problema da mochila.

Bennaton, J. (2001). O que é otimizar?

Boleta, D.; Araujo, S. C. A. and Poldi, K. (2005). Uma heuristica para o problma de corte de estoque unidimensional. Disponível em: http://www.decom.ufop.br/prof/marcone/Disciplinas/InteligenciaComputacional/CorteEstoqueUnidimen Acesso em 17 de setembro de 2011.

Brito, P. (2010). Vns aplicado ao problema da mochila.

CALDAS, R. B. (2004). Projeto e análise de algoritmos.

CAMPONOGARA, E. Programação dinâmica.

Chaves, A. A. (2003). Modelagens exata e heurística para resolução do problema do caixeiro viajante com coleta de prêmios.

Erito M. S. Filho, Virgílio J. M. F. Filho, L. S. L. (2007). Variable neighborhood search(vns) aplicado ao problema de distribuição dutoviária com restrições de capacidade.

FEOFILOFF, P. (2001). Análise de algoritmo: Programação dinâmica.

Ferreira, J. Programação dinâmica.

Fincatti, C. (2010). Problema da mochila.

Luila, E. (2008). Problema da mochila multicritério.

Marques, F. and Arenales, M. (2002). O problema da mochila compartimentada e aplicações.

Martins, A. (2010). Definição: O que são meta-heurísticas.

Mor, S. D. K. (2007). Emprego da técnica de workstealing: Estudo de caso com o problema da mochila e mpi.

PUC Minas, M. http://ziggi.uol.com.br/downloads/visual-calculo-numerico.

Ribeiro, B. M. d. C. (2010). Seleção de visões multidimensionais a partir de perfis de usuários e grasp.

Rocha, A. and Dorini, B. (2004). Algorirmtos gulosos: definições e aplicações.

Santos, J. Programação dinâmica.

Spolador, F. (2005). O problema da mochila compartimentada.

Sucupira, I. R. Métodos heurísticos genéricos: Meta-heurísticas e hiper-heurísticas.

UFMG/ICEx/DCC (2008). Trabalho prático da disciplina.