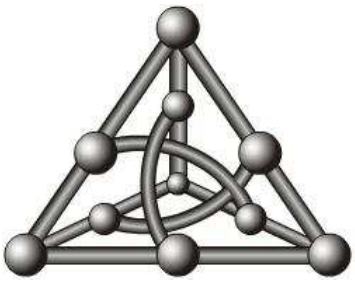
Problema da Mochila (Knapsack) e Aplicações

Pedro Luiz da Costa Silva

Trabalho de Conclusão de Curso

Orientação: Fábio Henrique Viduani Martinez

Área de Concentração: Otimização Algorítmica



Faculdade de Computação
Universidade Federal de Mato Grosso do Sul
4 de dezembro de 2023

Resumo

O presente trabalho tem como objetivo analisar diferentes variações do Problema Knapsack (problema da mochila), que é um clássico problema de otimização combinatória. São analisados diferentes trabalhos feitos com esse algoritmo, soluções exatas e heuristicas e aplicações do mundo real.

Conteúdo

1 Introdução **1**

1.1 Organização do Texto 1

2 Metodologia **2**

2.1 Identificação do Tema e Objetivos 2

2.2 Justificativa 2

2.3 Coleta de Dados 3

2.4 Análise das Aplicações 3

2.5 Discussão e Conclusão 3

3 Revisão da Literatura **4**

3.1 Considerações Gerais 4

3.2 Problema Knapsack Tradicional 5

3.3 Problema Knapsack Com Repetições Limitado 5

3.4 Problema Knapsack Com Repetições Ilimitado 6

3.5 Problema Knapsack Fracional 6

3.6 Problema Knapsack Multidimensional 8

3.7 Problema Knapsack 2D com Relação entre Itens 8

3.8 Knapsack Múltiplo 9

3.9 Outras variações 10

4 Resultados **11**

4.1 Gerenciamento de Recursos da Nuvem 11

4.2 Otimização de Produção 14

4.3 Criptossistemas Knapsack 15
4.4 Outras aplicações 15

5 Discussão e Conclusão **17**
5.1 Discussão dos Resultados 17
5.2 Conclusão 17

Capítulo 1

Introdução

A otimização combinatória, uma disciplina fundamental na área da computação, se concentra na busca das melhores soluções dentro de um espaço de possibilidades geralmente caracterizado por restrições complexas. Um exemplo notório em otimização combinatória é o problema da mochila (Knapsack), em que o objetivo é maximizar o valor total dos itens que podem ser colocados em uma mochila, considerando sua capacidade limitada. No entanto, essa abordagem vai além da escolha de itens para uma mochila; ela possui diversas variações e extensões que têm demonstrado eficácia na solução de uma ampla gama de problemas do mundo real.

Problemas como roteamento de veículos, escalonamento de produção e alocação de recursos, para citar apenas alguns, podem ser formulados como variações do problema Knapsack. Este trabalho se propõe a explorar a aplicabilidade dessas variações do problema Knapsack em contextos do mundo real, investigando como essa abordagem pode ser utilizada para aprimorar a eficiência em diversos domínios.

Ao longo do atual trabalho, analisaremos a aplicação de algoritmos e técnicas voltados para a otimização de Knapsack em situações práticas. As questões de pesquisa que orientarão este estudo são: Como as variações do problema Knapsack podem ser adaptadas para solucionar eficazmente problemas do mundo real? Quais são os desafios e limitações associados a essa abordagem? Como os resultados obtidos podem beneficiar empresas e organizações em busca de soluções mais eficientes em seus processos e operações?

1.1 Organização do Texto

O restante desta proposta está organizado da seguinte maneira: Primeiramente, é apresentado a metodologia usada. A seguir, são apresentadas as diversas variações do Problema Knapsack que encontramos na literatura e suas expressões matemáticas. Após isso, são mostrados exemplos de aplicação dessas variações, e também a maneira que os problemas foram modelados. Por fim, são discutidos os resultados, bem como as limitações encontradas na pesquisa.

Capítulo 2

Metodologia

2.1 Identificação do Tema e Objetivos

O tema desta pesquisa é a investigação das aplicações do problema Knapsack e suas variações no campo da análise combinatória para resolver problemas do mundo real.

Os objetivos são:

- Investigar as aplicações práticas do problema Knapsack em contextos do mundo real;
- Analisar as variações do problema Knapsack e como elas podem ser adaptadas para abordar problemas específicos em diversos domínios;
- Avaliar o impacto e a eficácia do uso do problema Knapsack na otimização de processos e tomada de decisões em situações reais.

2.2 Justificativa

O estudo das aplicações do problema Knapsack no mundo real é relevante e impactante devido à sua versatilidade na resolução de problemas práticos em uma variedade de domínios, como logística, manufatura, sistemas de segurança e muito mais [14].

Um estudo de 1999 do Repositório de Algoritmos da Universidade Stony Brook mostrou que, de 75 problemas algorítmicos relacionados ao campo de algoritmos combinatórios, o problema da mochila era o 19º mais popular e o terceiro mais necessário, depois das Árvores de Sufixos e do Problema do Empacotamento [32].

Com esta pesquisa, buscamos fornecer insights valiosos sobre como essa técnica pode ser aplicada de forma eficaz em cenários do mundo real, contribuindo para a melhoria de processos e tomadas de decisões em diversas áreas.

2.3 Coleta de Dados

A coleta de dados será realizada por meio de várias fontes, incluindo:

- Artigos acadêmicos de bases de dados científicas, como IEEE Xplore, ACM Digital Library e Google Scholar.
- Documentos técnicos, relatórios e estudos de caso relevantes publicados por organizações e empresas.

2.4 Análise das Aplicações

Os dados coletados serão analisados com base no domínio de aplicação, a variação do problema Knapsack usada, os métodos computacionais empregados e os resultados obtidos. A análise permitirá identificar tendências e padrões nas aplicações do Knapsack no mundo real.

2.5 Discussão e Conclusão

A discussão abordará a eficácia do uso do Knapsack nas aplicações examinadas, destacando suas vantagens e limitações. A conclusão resumirá as principais descobertas e insights desta pesquisa.

Capítulo 3

Revisão da Literatura

3.1 Considerações Gerais

O problema Knapsack é um problema de otimização combinatória com inúmeros estudos publicados, devido à sua enorme aplicabilidade em áreas como alocação de recursos. Este problema vem sido estudo há mais de um século, com o primeiro estudo publicado datando de 1897 [20].

Existem muitas variações do problema da mochila que surgiram do vasto número de aplicações do problema básico. Essas variações ocorrem adicionando novas restrições ou alterando parâmetros do problema como número de itens, número de objetivos ou mesmo número de mochilas [15].

Para análise dos algoritmos, foram consideradas as definições:

- **Problemas NP**: Na teoria da complexidade computacional, a classe NP ("*Non-deterministic Polynomial*") de problemas é o conjunto dos problemas de decisão que são polinomialmente verificáveis, ou seja, que são decidíveis em tempo polinomial por uma máquina de Turing não-determinística.
- **Problemas NP-Difíceis**: Um problema X é NP-difícil se todos os problemas da classe NP são polinomialmente redutíveis a X. Ou seja, um problema é NP-difícil se for pelo menos tão difícil quanto qualquer problema em NP.
- **Problemas NP-Completos**: Um problema é NP-completo se ele, ao mesmo tempo, estiver no conjunto dos problemas NP e for NP-difícil.
- **FPTA** - *fully polynomial-time approximation scheme*: São esquemas de aproximações de algoritmos em tempo polinomial. Muitas vezes são algoritmos pseudo-polinomiais.

3.2 Problema Knapsack Tradicional

O enunciado do problema Knapsack tradicional consiste em, dado um conjunto de itens, cada um com um peso e um valor, determinar quais itens incluir na coleção (mochila) para que o peso total seja menor ou igual a um determinado limite e o valor total seja o maior possível [29]. Matematicamente, pode ser expressado como:

Dados:

- O tamanho da mochila: $m \in \mathbb{Z}_+^*$;
- A quantidade de itens: $r \in \mathbb{Z}_+^*$;
- Os valores dos itens: $P = \{p_1, p_2, ..., p_r\}$, com $p_i \in \mathbb{Z}_+^*$ e $1 \leq i \leq r$;
- Os pesos dos itens: $W = \{w_1, w_2, ..., w_r\}$, com $w_i \in \mathbb{Z}_+^*$ e $1 \leq i \leq r$;

O conjunto de itens escolhidos é dado por $X = \{x_1, x_2, ..., x_r\}$, com $x_i \in \{0, 1\}$ e $1 \leq i \leq r$, ou seja, se $x_i = 1$, o item i foi escolhido. Assim, escolha X de tal forma a:

- Maximizar: $\sum p_i x_i$
- Sujeito a: $\sum w_i x_i \leq m$

Este problema também é chamado de 0/1 Knapsack. Pode ser resolvido recursivamente, com complexidade $O(2^n)$, ou com técnicas de programação dinâmica: Top-Down with memoization ou Bottom-Up, com consumo de tempo $O(nm)$ [14]. Além disso, **não** é considerado um problema NP-completo, porém admite um FPTA, através de um algoritmo pseudo-polinomial [34].

3.3 Problema Knapsack Com Repetições Limitado

O Problema Knapsack Com Repetições Limitado é muito parecido com o tradicional, porém é possível incluir mais de um do mesmo item na mochila, até um máximo que indica a disponibilidade de cada item [16]. Matematicamente, é dado por:

Dados:

- O tamanho da mochila: $m \in \mathbb{Z}_+^*$;
- A quantidade de itens: $r \in \mathbb{Z}_+^*$;
- Os valores dos itens: $P = \{p_1, p_2, ..., p_r\}$, com $p_i \in \mathbb{Z}_+^*$ e $1 \leq i \leq r$;
- A disponibilidade dos itens: $B = \{b_1, b_2, ..., b_r\}$, com $b_i \in \mathbb{Z}_+^*$ e $1 \leq i \leq r$;
- Os pesos dos itens: $W = \{w_1, w_2, ..., w_r\}$, com $w_i \in \mathbb{Z}_+^*$ e $1 \leq i \leq r$;

O conjunto de itens escolhidos é dado por $X = \{x_1, x_2, ..., x_r\}$, tal que, se $x_i = 3$, o item i foi escolhido 3 vezes. Assim, escolha X de tal forma a:

- Maximizar: $\sum p_i x_i$
- Sujeito a: $\sum w_i x_i \leq m$ e $x_i \in \{0, 1, 2, ..., b_i\}$ para $1 \leq i \leq r$

Este problema também é chamado de bounded knapsack problem (BKP), **não** é NP-completo, mas aceita um FPTA pseudo-polinomial [23].

3.4 Problema Knapsack Com Repetições Ilimitado

O Problema Knapsack Com Repetições Ilimitado considera que as disponibilidades de todos os itens são infinitas, ou seja, podem ser escolhidos inúmeras vezes. Matematicamente, é dado por:

Dados:

- O tamanho da mochila: $m \in \mathbb{Z}_+^*$;
- A quantidade de itens: $r \in \mathbb{Z}_+^*$;
- Os valores dos itens: $P = \{p_1, p_2, ..., p_r\}$, com $p_i \in \mathbb{Z}_+^*$ e $1 \leq i \leq r$;
- Os pesos dos itens: $W = \{w_1, w_2, ..., w_r\}$, com $w_i \in \mathbb{Z}_+^*$ e $1 \leq i \leq r$;

O conjunto de itens escolhidos é dado por $X = \{x_1, x_2, ..., x_r\}$. Assim, escolha X de tal forma a:

- Maximizar: $\sum p_i x_i$
- Sujeito a: $\sum w_i x_i \leq m$ e $x_i \in \mathbb{Z}_+^*$, com $1 \leq i \leq r$

Este problema também é chamado de unbounded knapsack problem (UKP), e foi provado em 1975 que é NP-completo [19]. A solução ótima pode ser obtida pelo "algoritmo ganancioso", onde os itens mais valiosos são escolhidos primeiro, proposto em 1957 por Dantzig [11]. Além disso, também admite solução aproximada em tempo polinomial (FPTA) [24].

3.5 Problema Knapsack Fracional

O problema Knapsack fracional é uma variante do problema clássico, porém agora é possível escolher frações de objetos. Assim, os valores P passam a significar "valor por

volume", e os pesos W passam a significar o "volume máximo" que pode-se escolher de um objeto [3]. Matematicamente, o problema pode ser expressado como:

Dados:

- O tamanho da mochila: $m \in \mathbb{R}_+^*$;
- A quantidade de itens: $r \in \mathbb{Z}_+^*$;
- Os valores por unidade dos itens: $P = \{p_1, p_2, ..., p_r\}$, com $p_i \in \mathbb{R}_+^*$ e $1 \leq i \leq r$;
- Os volumes máximos disponíveis: $W = \{w_1, w_2, ..., w_r\}$, com $w_i \in \mathbb{R}_+^*$ e $1 \leq i \leq r$;

O conjunto de itens escolhidos é dado por $X = \{x_1, x_2, ..., x_r\}$. Assim, escolha X de tal forma a:

- Maximizar: $\sum p_i x_i$
- Sujeito a: $\sum w_i x_i \leq m$ e $x_i \in [0, 1]$, com $1 \leq i \leq r$

Essa variação é NP-completa e admite um FPTA. Ela pode ser resolvida com um algoritmo ganancioso: obter o máximo do objeto com maior valor por volume. Se ainda restar espaço, obter o máximo do próximo objeto mais valioso, e continuar até a mochila estiver cheia. Essa solução foi primeiro proposta em 1957 por George B. Dantzig [11].

3.6 Problema Knapsack Multidimensional

Nesta variação, o peso da mochila é dado por um vetor multidimensional, adicionando novas restrições (uma para cada dimensão). O objetivo é maximizar o valor total dos itens sem ultrapassar qualquer uma das restrições. Matematicamente, pode ser expressada por:

Dados:

- A quantidade de dimensões: $D \in \mathbb{Z}_+^*$;
- A capacidade de cada dimensão, dado pelo vetor: $M = (M_1, M_2, ..., M_D)$;
- A quantidade de itens: $r \in \mathbb{Z}_+^*$;
- Os valores dos itens: $P = \{p_1, p_2, ..., p_r\}$, com $p_i \in \mathbb{Z}_+^*$ e $i \leq r$;
- Os pesos dos itens: $W = \{\overline{w_1}, \overline{w_2}, ..., \overline{w_r}\}$, com $\overline{w_i} = (w_{i1}, w_{i2}, ..., w_{iD})$ e $i \leq r$;

O conjunto de itens escolhidos é dado por $X = \{x_1, x_2, ..., x_r\}$, com $x_i \in \{0, 1\}$ e $i \leq r$, ou seja, se $x_i = 1$, o item i foi escolhido. Assim, escolha X de tal forma a:

- Maximizar: $\sum p_i x_i$
- Sujeito a: $\sum_{i=1}^{r} w_{ij} x_i \leq M_j$, para todo $0 \leq j \leq D$

Computacionalmente, o problema **não** é NP-completo, não possuindo soluções polinomiais de aproximação eficientes, ao menos que P=NP [18] [13]. Porém, foi provado que alguns algoritmos podem gerar soluções eficientes para o Knapsack Multidimensional em instâncias esparsas [10].

3.7 Problema Knapsack 2D com Relação entre Itens

Esta variação é derivada do Knapsack Multidimensional com duas dimensões, adicionando relações 2 a 2 entre itens, e pode ser enunciada geometricamente:

Seja B uma caixa retangular com comprimento $W \in \mathbb{Z}_+^*$ e altura $H \in \mathbb{Z}_+^*$. Seja I um conjunto de itens retangulares tal que cada item $i \in I$ possui comprimento $w_i \in \mathbb{Z}_+^*$, altura $h_i \in \mathbb{Z}_+^*$ e valor $v_i \in \mathbb{Z}_+^*$. Além disso, cada par de itens i, j, com $i \neq j$, possui um valor de relação $r_{ij} \in \mathbb{Z}$. Note que esse valor pode ser negativo.

O problema consiste em encontrar um empacotamento P_I para maximizar o valor total V da caixa, que é dado pela soma dos valores individuais de cada item e os valores de relação, ou seja, maximizar $V = \sum_{i \in P_I} v_i + \sum_{i,j \in P_I} r_{ij}$.

Um empacotamento P_I deve satisfazer as seguintes restrições:

1. Os itens devem ser colocados de maneira ortogonal, ou seja, as arestas dos itens devem ser paralelas às arestas da caixa;

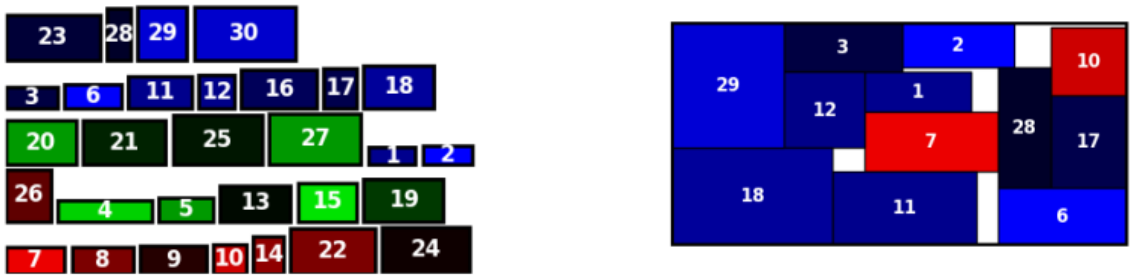


Figura 3.1: (a) Um exemplo de arranjo de itens de entrada. Itens de mesma cor possuem um lucro extra associado, e itens de cor diferente possuem um prejuízo. (b) O empacotamento ideal dos itens de (a).

2. Os itens devem seguir sua orientação original;

3. Os itens devem ser empacotados dentro dos limites da caixa, ou seja, se o item i for empacotado com seu canto inferior-esquerdo no ponto (x_i, y_i), então:

   - $0 \leq x_i \leq x_i + w_i \leq W$
   - $0 \leq y_i \leq y_i + h_i \leq H$

4. Os itens não podem se sobrepor, ou seja, se a região ocupada por um item i é dada por $R(i) = [x_i, x_i + w_i] \times [y_i, y_i + h_i]$, então $R(i) \cap R(j) = \varnothing$ para todo $i \neq j \in P_I$.

Essa variação também é conhecida por 2D-KPRI [28].

Como exemplo de empacotamento, considere os 30 itens da figura 3.1(a). Itens de mesma cor possuem valores de relação r_{ij} positivo, e itens de cor diferente possuem valores negativos. O empacotamento ideal é dado pela figura 3.1(b). Note que às vezes pode ser lucrativo empacotar itens de diferentes cores, mesmo com a penalidade associada [28].

Por ser uma variação do Knapsack Multidimensional, também não é NP-completo nem admite FPTA. Porém, através de algoritmos de chaves aleatórias (BRKGA) é possível achar soluções ótimas em tempo prático de execução [28].

3.8 Knapsack Múltiplo

Nessa variação, são consideradas várias mochilas, cada uma com uma dada capacidade. Cada item pode ser escolhido uma vez, e somente para uma mochila. O objetivo é maximizar o lucro total. Matematicamente, pode ser enunciado por:

Dados:

- A quantidade de mochilas: $n \in \mathbb{Z}_+^*$;
- Os tamanho da mochilas: $M = \{m_1, m_2, ..., m_n\}$, com $m_i \in \mathbb{Z}_+^*$ e $1 \leq i \leq n$;

- A quantidade de itens: $r \in \mathbb{Z}_+^*$;
- Os valores dos itens: $P = \{p_1, p_2, ..., p_r\}$, com $p_i \in \mathbb{Z}_+^*$ e $1 \leq i \leq r$;
- Os pesos dos itens: $W = \{w_1, w_2, ..., w_r\}$, com $w_i \in \mathbb{Z}_+^*$ e $1 \leq i \leq r$;

O conjunto de itens escolhidos é dado por $X = \{\overline{x_1}, \overline{x_2}..., \overline{x_r}\}$, com $\overline{x_i} = (x_{i1}, x_{i2}, ..., x_{in})$ e $x_{ij} \in \{0, 1\}$, com $1 \leq i \leq r$ e $1 \leq j \leq n$, ou seja, se $x_{ij} = 1$, o item i foi escolhido para a mochila j. Assim, escolha X de tal forma a:

- Maximizar: $\sum_{i=1}^{m} \sum_{j=1}^{n} p_i x_{ij}$
- Sujeito a: $\sum_{i=1}^{m} w_i x_{ij} \leq m_j$ para todo $1 \leq j \leq n$ e $\sum_{j=1}^{n} x_{ij} \leq 1$ para todo $1 \leq i \leq m$

Esta variação não é NP-completa nem possui um FPTA [9].

3.9 Outras variações

- Problema Knapsack Multiobjetivo: Essa variação adiciona mais objetivos a serem alcançados. Em vez de somente uma dimensão, como a maximização do lucro, o objetivo pode haver várias dimensões [8]. Por exemplo, pode haver preocupações ambientais ou sociais, bem como objetivos econômicos. A formulação de cada problema Knapsack Multiobjetivo é única, e as restrições podem ser variadas.
- Problema Knapsack Quadrático: Nessa variação, o objetivo é maximizar uma função quadrática sujeita a restrições de capacidade binária e linear [35]. As primeiras tratativas desse problema datam de 1975 [4].
- Knapsack com Conceitos Combinados: Também é possível combinar os conceitos dos problemas Knapsack para criar novos problemas. Por exemplo, é possível que um problema Knapsack contenha múltiplas dimensões e também possua múltiplos objetivos. É possível, portanto, criar variações do Knapsack alterando cada um dos parâmetros do problema [14].

Capítulo 4

Resultados

4.1 Gerenciamento de Recursos da Nuvem

A variação Knapsack Fracional pode ser utilizada para modelar o gerenciamento de recursos de nuvem em aplicações SaaS (Software as a Service) [25].

Para essa modelagem, considera-se um servidor de Computação em Nuvem realizando processamento de tarefas (jobs) de vários clientes. Cada trabalho possui uma quantidade de memória exigida e um lucro atribuído. O servidor possui um total máximo de memória e deve alocá-la aos clientes de forma a maximizar o lucro.

Diferente do problema do Knapsack clássico, porém, a alocação de recursos é dinâmica: novas tarefas podem chegar ao servidor enquanto ele já possui tarefas em execução, e, caso não haja memória para processar todas, deve-se decidir se é mais lucrativo continuar o processamento das que já estão alocadas ou pausar alguma tarefa para processar aquela que chegou.

A memória alocada e o lucro atribuído para cada tarefa são definidos contratualmente pelo *Service Level Agreement* (SLA), ou Acordo de Nível de Serviço. Ao não fornecer (ou fornecer parcialmente) memória para alguma tarefa, podem decorrer multas pelo não cumprimento do SLA e isso deve ser levado em consideração no cálculo do lucro.

Assim, a modelagem define:

- A memória total disponível: $M \in \mathbb{R}_+^*$;
- O lucro de uma tarefa i: $p_i \in \mathbb{R}_+^*$;
- A memória requerida por uma tarefa i: $w_i \in \mathbb{R}_+^*$;

Assume-se que existem tarefas j_1 até j_{i-1} em execução quando uma nova tarefa j_i solicita recursos. Caso a memória necessária para j_i seja maior que a disponível, então somente uma parte da memória pode ser alocada, o que pode ocasionar em violação do contrato. A figura 4.1 ilustra essa situação. As tarefas que não foram alocadas são dispostas em uma fila, conforme a figura 4.2.

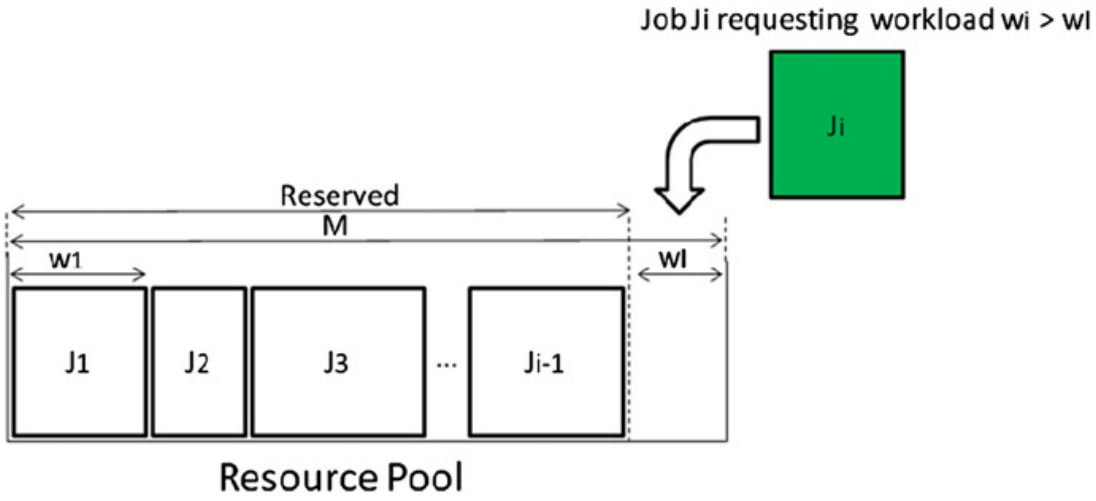


Figura 4.1: Tarefa requisitando recursos maiores que os disponíveis.

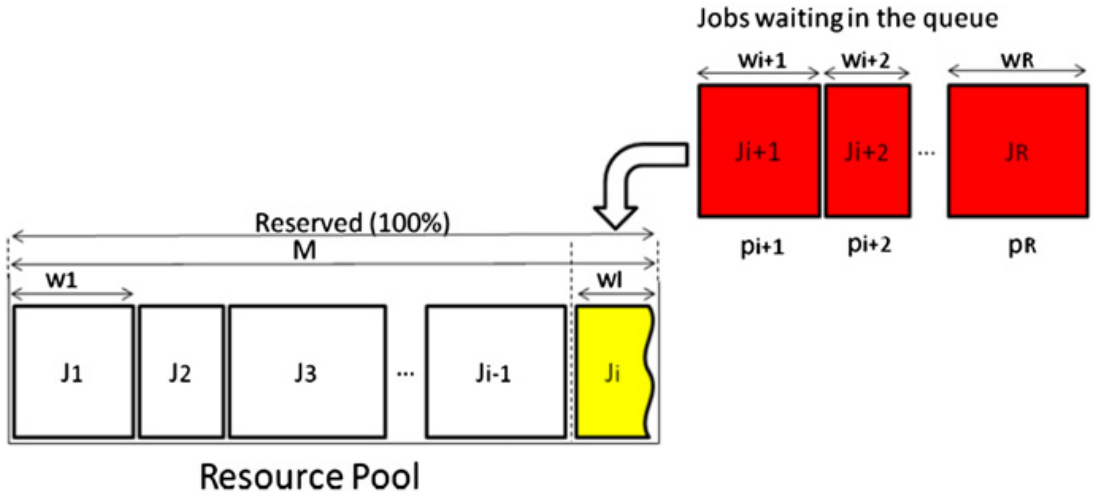


Figura 4.2: Tarefas esperando para serem alocadas.

A alocação, assim, pode ser resolvida com um algoritmo ganancioso:

1. Ordenar os trabalhos em uma fila de acordo com a razão p_i/w_i;
2. Alocar os recursos do primeiro trabalho e removê-lo da fila;
3. Repetir 2 até que não haja recursos suficientes para atender a próxima tarefa;
4. Recalcule os lucros (considerando as penalidades da quebra de contrato), e aloque os recursos restantes para aquele com maior razão p_i/w_i.

O cálculo dos lucros p_i em si, porém, é mais complexo, pois deve levar em consideração, além do valor obtido pela execução de uma tarefa, a probabilidade da quebra de contrato das tarefas que não foram alocadas ou foram alocadas parcialmente.

	Fractional Knapsack meta-scheduler	GRIA without any meta-scheduler	LAS policy meta-scheduler
1st Iteration			
Job success rate	83.3%	66.7%	70%
Mean disk utilization	7.4%	6.8%	6.7%
Mean memory utilization	49.8%	39.1%	45%
Mean CPU utilization	37.5%	35.5%	36.9%
Total payoff	16.557	11.670	12.540
2nd Iteration			
Job success rate	86.7%	73.3%	83.3%
Mean disk utilization	8%	8,4%	8.4%
Mean memory utilization	42.4%	41.9%	45.5%
Mean CPU utilization	32%	29.8%	29.9%
Total payoff	16.244	11.198	14.866
3rd Iteration			
Job success rate	86.7%	70%	86.7%
Mean disk utilization	9.7%	7.8%	9.7%
Mean memory utilization	51.1%	46.7%	59.1%
Mean CPU utilization	54.7%	50.7%	53.2%
Total payoff	16.816	12.894	16.598

Figura 4.3: Resultados da comparação dos escalonadores.

Para esse cálculo, são considerados, para cada tarefa i:

- PR_i: preço pago pela execução;
- EC_i: custos da execução;
- VC_i: custos caso haja violação contratual;
- VP_i: probabilidade de violação contratual, considerando histórico de falhas para cada aplicação e a fração da memória fornecida para a tarefa.

Assim, o lucro potencial é dado por $p_{i,1} = PR_i - VP_i \times VC_i - EC_i$. Também é definida a probabilidade de sucesso, dada por $SP_i = 1 - VP_i$. Dessa forma, outra maneira de medir o lucro é através do custo de violação que será economizado caso o trabalho seja concluído, dado por $p_{i,2} = SP_i \times VC_i$.

A modelagem final considera, além da memória para cada tarefa, outras demandas de recurso, tornando o problema em um Knapsack Multidimensional. Um algoritmo ganancioso é proposto e implementado [26] usando um middleware GRIA (Grid Resources for Industrial Applications) [2]. Tal middleware oferece modelos para garantia da qualidade de serviço, permitindo o gerenciamento de múltiplos clientes e SLAs.

Essa modelagem foi comparada com outros dois escalonadores: o escalonador padrão do GRIA e um escalonador LAS (Least-Attained-Service). O escalonador usando a modelagem proposta conseguiu gerar maior lucro que ambos, enquanto utilizou menos recursos de memória e processamento [27]. A figura 4.3 mostra os resultados do experimento.

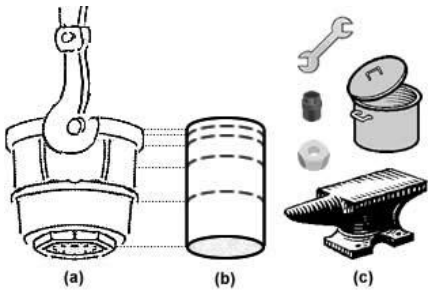


Figura 4.4: Visualizando a fornalha como uma mochila (Knapsack).

4.2 Otimização de Produção

O Knapsack com Repetições Limitado foi usado para otimizar cargas em uma fundição de pequeno porte em São Carlos, SP (Brasil) [5]. O processo de fornada consiste em derreter diferentes metais em uma fornalha e colocar o metal líquido em moldes para produzir peças diferentes. Para otimizar esse processo, é importante criar um plano de produção, o que envolve duas etapas:

1. Determinar as ligas metálicas que serão utilizadas em cada fornada;
2. Para cada fornada, determinar as peças que serão produzidas.

A etapa (1) é feita através de um algoritmo genético. Já na etapa (2), cada fornada representa um sub-problema de dimensionamento de lote, que pode ser resolvido como um problema Knapsack.

A fornalha pode ser enxergada como uma mochila Knapsack, pois ela possui uma capacidade máxima (equivalente ao tamanho da mochila), no qual devemos colocar quantidades de metal para produzir as peças demandadas (equivalente aos itens inclusos na mochila). Cada peça possui um lucro atribuído e o objetivo é determinar as peças produzidas de forma a maximizar o lucro. Essa modelagem está ilustrada na figura 4.4.

Os lucros dos itens são determinados empiricamente considerando o peso da peça, sub-utilização da fornalha, desperdício de material, entre outros fatores. Mais de um tipo do mesmo item pode ser produzido na fornada, porém foram estabelecidos limites para cada tipo. Portanto, essa modelagem é a de um Knapsack com Repetições Limitado.

O problema foi resolvido utilizando programação dinâmica, com a execução feita recursivamente usando matrizes. Assim, foi criado um algoritmo, que foi implementado em um programa escrito na linguagem C [6].

O algoritmo criado foi comparado com um algoritmo heurístico proposto por Tonaki e Toledo [33] chamado *Balance Furnace Loading Heuristic* (BLFH), que também visava fundições pequenas. Provou-se que o algoritmo com base no Knapsack produzia resultados com maior lucro, menos desperdício de material e melhor tempo de execução. Além disso, o programa criado não necessita grande capacidade computacional e nem software comercial, sendo ideal para fundições de pequeno porte.

4.3 Criptossistemas Knapsack

Criptossistemas Knapsack são sistemas cuja segurança é baseada na dificuldade de resolver problemas Knapsack [30]. Esse tipo de criptossistema é um bom candidato para criptografia pós-quântica, pois são problemas difíceis de serem resolvidos até por computadores quânticos. Esse não é o caso de sistemas como o RSA (baseados na fatoração de inteiros grandes) ou o ECDSA (computação de logaritmos discretos), pois podem ser resolvidos em tempo polinomial usando algoritmos quânticos, como o Algoritmo de Shor [31].

O criptossistema à base do Knapsack mais famoso é Merkle-Hellman. É um sistema proposto em 1978 por Ralph Merkle e Martin Hellman [21]. Foi um dos primeiros sistemas a usar o conceito de "função unidirecional de alçapão" criado pelo próprio Martin Hellman e Whitfield Diffie em 1976 [12].

O sistema Merkle-Hellman hoje é considerado inseguro pois já foram encontradas soluções polinomiais que o quebram [1]. Porém, existem criptossistemas Knapsack considerados seguros, como o de Nasako-Murakami, desenvolvido em 2006 [22].

4.4 Outras aplicações

- O Knapsack Múltiplo é utilizado em diversos problemas de carregamento e escalonamento em Pesquisa Operacional [9];
- O Knapsack Quadrático foi usado nas primeiras formulações de sistemas eletrônicos de mensagens [4];
- Empresas de logística frequentemente usam o Knapsack Multdimensional para otimizar o transporte de produtos, considerando limitações de capacidade e volume, a fim de minimizar custos operacionais, como combustível e tempo de entrega [14];
- Empresas de telecomunicações podem aplicar o Knapsack Multidimensional para alocar recursos, como largura de banda e capacidade de transmissão, de forma eficiente, considerando restrições de custo, qualidade de serviço e capacidade, e algoritmos são frequentemente aplicados para escalamento de pacotes em redes sem fio com nós de retransmissão [10];
- O Knapsack Multiobjetivo já foi utilizado na geração de um algoritmo genético para otimizar subestações do Sistema Ferroviário de Washington, D.C.[7];
- O Knapsack Multiobjetivo é frequentemente empregado para tomar decisões de investimento em uma coleção de instrumentos financeiros ou ativos (otimização de portfólio) [8].
- O problema Knapsack Multidimensional também é frequentemente usado para otimizar o corte de materiais, como rolos de tecido, para atender a múltiplas demandas de produção [17].

- A variação Knapsack 2D com relação entre itens pode ser utilizada para armazenamento de itens em um depósito, garantindo o melhor uso possível do espaço e, ao mesmo tempo, impedindo que itens como comida e itens de limpeza sejam armazenados juntos [28].

Capítulo 5

Discussão e Conclusão

5.1 Discussão dos Resultados

Foi constatado na literatura que o problema Knapsack possui uma grande gama de aplicações. Julga-se que isso é devido dois fatores:

1. A essência do problema Knapsack é a alocação de recursos limitados, o que é algo muito comum, sendo útil a procura de soluções que otimizem essa alocação;
2. A grande quantidade de variações do problema, permitindo formar modelagens que se adaptem à diversos problemas reais.

Uma importante limitação a ser levantada é que muitas das aplicações encontradas são teóricas. Os criptossistemas Knapsack, por exemplo, podem ter aplicação prática somente quando computadores quânticos se tornarem difundidos, o que quebraria os sistemas criptográficos atuais, obrigando a adoção de novos sistemas.

5.2 Conclusão

Em síntese, a pesquisa empreendida proporcionou uma visão abrangente sobre as aplicações das variações do problema Knapsack na resolução de problemas do mundo real, evidenciando a versatilidade e relevância dessas abordagens na otimização de processos em diversos domínios, como alocação de recursos, otimização de produção, logística e roteamento de transporte público.

A análise ressalta a importância contínua dessas abordagens na busca por soluções mais eficientes e adaptáveis. Dessa forma, o atual trabalho destaca a aplicabilidade concreta do problema Knapsack, proporcionando uma base sólida para futuras pesquisas e práticas na área de otimização combinatória e ciência da computação como um todo.

Referências Bibliográficas

[1] Leonard Adleman. *Advances in Cryptology*, capítulo On Breaking the Iterated Merkle-Hellman Public-Key Cryptosystem, páginas 303—-308. Springer, 1983.

[2] Vários autores. The gria project. In *Grid Resources for Industrial Applications*. 2009.

[3] Paul Black. Fractional knapsack problem. In *National Institute of Standards and Technology (February 4)*. 2009.

[4] Witzgall C. Mathematical methods of site selection for electronic message systems (ems). In *NBS Internal Report. 76*. 1975.

[5] Victor Camargo, Leandro Mattiolli, e Franklina Toledo. A knapsack problem as a tool to solve the production planning problem in small foundries. *Computers & OR*, 39, 01 2012.

[6] Victor Camargo, Leandro Mattiolli, e Franklina Toledo. A knapsack problem as a tool to solve the production planning problem in small foundries. *Computers & OR*, 39:14–19, 01 2012.

[7] C. S. Chang. Genetic algorithm based bicriterion optimization for traction substations in dc railway system. In *Fogel [102]*, páginas 11–16. 1998.

[8] T. J. Chang. Heuristics for cardinality constrained portfolio optimization. In *Technical Report, London SW7 2AZ, England: The Management School, Imperial College*. 1998.

[9] Chandra Chekuri e Sanjeev Khanna. A ptas for the multiple knapsack problem. In *SIAM Journal on Computing. 35 (3)*, páginas 713—-728. 2005.

[10] R. Cohen e G Grebla. Multi-dimensional ofdma scheduling in a wireless network with relay nodes. In *Proc. IEEE INFOCOM'14*, páginas 2427–2435. 2014.

[11] George B. Dantzig. Discrete-variable extremum problems. In *Operations Research, 5*, páginas 266–277. 1957.

[12] Whitfield Diffie e Martin Hellman. *IEEE Transactions on Information Theory*, capítulo New directions in cryptography, página 644. IEEE, 1976.

[13] G. V. Gens e E. V. Levner. Complexity and approximation algorithms for combinatorial problems: A survey. 1979.

[14] Hans Kellerer e David Pisinger. Knapsack problems. In *ISBN 978-3-540-40286-2*, páginas 449–465. 2004.

[15] Hans Kellerer e David Pisinger. Knapsack problems. In *ISBN 978-3-540-40286-2*. 2004.

[16] Hans Kellerer e David Pisinger. Knapsack problems. In *ISBN 978-3-540-40286-2*, páginas 185–209. 2004.

[17] Hans Kellerer e David Pisinger. Knapsack problems. In *ISBN 978-3-540-40286-2*, página 449. 2004.

[18] A. Kulik e H. Shachnai. There is no eptas for two dimensional knapsack. In *Inf. Process. Lett. 110*, páginas 707–712. 2010.

[19] G.S. Lueker. Two np-complete problems in nonnegative integer programming. In *Report No. 178, Computer Science Laboratory, Princeton*. 1975.

[20] G. B Mathews. On the partition of numbers. In *Proceedings of the London Mathematical Society. 28*, páginas 486–490. 1897.

[21] Ralph Merkle e Martin Hellman. *IEEE Transactions on Information Theory*, capítulo Hiding information and signatures in trapdoor knapsacks, páginas 525—530. IEEE, 1978.

[22] T. Nasako e Y. Murakami. *Japan Society for Industrial and Applied Mathematics, 16*, capítulo A high-density knapsack cryptosystem using combined trapdoor, páginas 519–605. 2006.

[23] David Pisinger. A minimal algorithm for the bounded knapsack problem. In *International Conference on Integer Programming and Combinatorial Optimization*. 1995.

[24] Vincent Poirriez, Nicola Yanev, e Rumen Andonov. A hybrid algorithm for the unbounded knapsack problem. In *Discrete Optimization*, páginas 110–124. 2009. ISSN 1572-5286.

[25] K. Rollman, W. Cardoso, V. L. Lima, e F. K. Miyazawa. Resource management in software as a service using the knapsack problem model. volume 141. 2011.

[26] K. Rollman, W. Cardoso, V. L. Lima, e F. K. Miyazawa. Resource management in software as a service using the knapsack problem model. volume 141, páginas 470–472. 2011.

[27] K. Rollman, W. Cardoso, V. L. Lima, e F. K. Miyazawa. Resource management in software as a service using the knapsack problem model. volume 141, páginas 472–475. 2011.

[28] K. Rollman, W. Cardoso, V. L. Lima, e F. K. Miyazawa. Heuristic and exact algorithms for the 2d knapsack problem with relations between items. In *IC-PFG-17-13*. 2017.

[29] Sartaj Sahni. Approximate algorithms for the 0/1 knapsack problem. In *Journal of the ACM 22*. 1975.

[30] Bruce Schneier. *Secrets and Lies*, página 95. Wiley Publishing, Inc, 2004.

[31] Peter Shor. *SIAM Journal on Computing. 26*, capítulo Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, páginas 1484–1509. 1997.

[32] S Skiena. Who is interested in algorithms and why? lessons from the stony brook algorithm repository. In *ACM SIGACT News*, páginas 65–74. 1999. ISSN 0163-5700.

[33] V. S. Tonaki e F. M. B. Toledo. An approach to solve the lot sizing and scheduling problem in market-driven foundries. In *Journal of Operational Research Society 200*, páginas 747–754. 2010.

[34] Vijay Vazirani. Approximation algorithms. In *Springer-Verlag Berlin Heidelberg*. 2003.

[35] Z. Y. Wu, Y. J. Yang, F. S. Bai, e M. Mammadov. Global optimality conditions and optimization methods for quadratic knapsack problems. In *J Optim Theory Appl. 151*, páginas 241–259. 2011.