CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

Diretoria de Pesquisa e Pós-Graduação

Programa de Pós-Graduação em Modelagem Matemática e Computacional

Algoritmos e Modelos Matemáticos para o Problema da k-Partição de Números

Projeto de Dissertação de Mestrado, submetido ao Programa de Pós-Graduação em Modelagem Matemática e Computacional, como parte dos requisitos exigidos para a obtenção do título de Mestre em Modelagem Matemática e Computacional.

Aluno : Alexandre Frias Faria

Orientador : Prof. Dr. Sérgio Ricardo de Souza

Co-Orientador : Prof. Dr. Carlos Alexandre Silva

Belo Horizonte
Fevereiro de 2016

Resumo

Este projeto de dissertação de mestrado apresenta formulação matemática e algoritmos para a versão de otimização do Problema da k-Partição de Números. Este problema consiste em distribuir os elementos de um conjunto dado em k subconjuntos disjuntos, de modo que as somas dos elementos de cada subconjunto fiquem no menor intervalo possível. O primeiro método consiste em aplicar a meta-heurística ILS (*Iterated Local Search*) na solução gerada por um algoritmo aproximado. A estratégia é aplicar método guloso em todas as fases do algoritmo, tanto na inicialização quanto para a escolha de movimentos do ILS. Outras propostas visam substituir o Oráculo do Algoritmo 1 pelo melhor resultado de um conjunto de heurísticas. Por fim, o uso de algoritmos exatos para problemas relacionados, como o Problema da Soma de Subconjunto, são adaptados ao tema desse trabalho. Alguns resultados parciais já publicados encontram-se no final do texto.

PALAVRAS-CHAVE: Problema da k-Partição de Números, Problema da Partição de Números, *Iterated Local Search*, *Complete Karmarkar-Karp Algorithm*.

Sumário

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Entende-se por partição de um conjunto S a uma coleção de subconjuntos, dois a dois disjuntos, cuja união forma S. Uma k-partição ocorre quando a partição tem exatamente k subconjuntos não vazios. Durante o decorrer do texto, os subconjuntos pertencentes à partição são denominados de partes.

O Problema da Partição de Números (*Two-Way Number Partitioning Problem* – TWNPP) já foi chamado de "*O mais fácil problema NP-completo*" por Hayes (2002), mas os problemas generalizados que derivam dele são muito mais complicados. Variações menos gerais desse problema se limitam a encontrar uma partição com partes de tamanho fixo, como o *3-Partition Problem*, encontrado em Joosten e Zantema (2013), ou mudam o objetivo original para um problema de escalonamento, como em Moffitt (2013).

Embora seja um dos 21 problemas de classe NP-completos discutidos por Karp (1972), existem algoritmos exatos para o TWNPP, como em Schroeppel e Shamir (1981), Horowitz e Sahni (1974) e Korf (1998)), como também muitos algoritmos aproximados aplicáveis ao problema analisados em Gent e Walsh (1998) e Ausiello et al. (2003).

1.1 Apresentação Inicial

Os conceitos de norma, diâmetro e bola são importantes para melhor compreensão da Seção 1.1 pois com eles é possível enunciar todos os problemas das Subseções 1.1.1, 1.1.2, 1.1.3 e 1.1.4 mantendo a mesma natureza da função objetivo. Esses conceitos podem se encontram formalmente descritos em (Lima, 2004).

Definição 1.1.1 *Uma função* $\|.\|_p : \mathbb{R}^n \to \mathbb{R}^+$ *é uma norma de* $\mathbb{R}^n$ *se para quaisquer* $x, y \in \mathbb{R}^n$ *e* $a \in \mathbb{R}$*:*

- $\|x\|_p = 0 \Rightarrow x = \vec{0}$
- $\|a.x\|_p = |a|.\|x\|_p$
- $\|x + y\|_p \leq \|x\|_p + \|y\|_p$

□

Nesse texto usa-se $p \to \infty$, ou seja, a norma do máximo que retorna a maior coordenada do vetor.

Definição 1.1.2 *Dados dois vetores $x, y \in \mathbb{R}^n$. A distância ente x e y é dada por $d(x,y) = \|x - y\|_p$.* □

Definição 1.1.3 *Dado um subconjunto $X \subset \mathbb{R}^n$. O diâmetro de X é $diam(X) = \max_{x_1,x_2 \in X}\{d(x_1,x_2)\}$.* □

Definição 1.1.4 *Dado um ponto $v_0 \in \mathbb{R}^n$ e um número real positivo r. Uma bola de cetro v_0 e raio r é o conjunto de pontos $v \in \mathbb{R}^n$ tais que $\|x - x_0\|_p \leq r$.* □

O diâmetro de um conjunto é duas vezes o raio da menor bola que contém todos os elementos deste conjunto.

Com o objetivo de apresentar o Problema da k-Partição de Números, nesta seção mostra-se uma sequência de outros problemas diretamente relacionados com o mesmo. Nota-se que é possível enunciar, genericamente, todos os problemas abaixo da seguinte forma:

Definição 1.1.5 *Dado um conjunto S de n vetores $v \in \mathbb{Z}_+^m$ e um inteiro k, o objetivo é encontrar uma partição de S com dimensão k, de modo que as somas dos elementos de cada parte fiquem contidas dentro da menor bola possível. Assim, espera-se minimizar o o raio dessa bola (o diâmetro do conjunto)* □

Em seguida, mostra-se um enunciado com exemplo para cada versão abaixo.

1.1.1 Problema da Partição de Números

O Problema da Partição de Números (*Two-Way Number Partitioning Problem* – TWNPP) é a tarefa de minimizar a diferença das somas de dois subconjuntos disjuntos de um conjunto S. Essa é a versão mais simples e popular do problema apresentada por Karp (1972) e tratada em diversos artigos, como Pedroso e Kubo (2010) e Gent e Walsh (1998). O tamanho da bola, nesse caso, é o comprimento do intervalo formado pelo resultado das duas somas. O exemplo 1.1.1 mostra essa situação.

Exemplo 1.1.1 *O conjunto $S = \{87, 6, 5, 45, 34, 2, 24, 12, 7, 6, 54, 34\}$ pode ser particionado em:*

$$\underbrace{\{87, 34, 24, 6, 5\}}_{soma=156}, \underbrace{\{54, 45, 34, 12, 7, 6, 2\}}_{soma=160}$$

O valor da função objetivo dessa partição é $160 - 156 = 4$, mas a partição ótima é

$$\underbrace{\{87, 34, 24, 6, 5, 2\}}_{soma=158}, \underbrace{\{54, 45, 34, 12, 7, 6\}}_{soma=158}$$

com valor de função objetivo $158 - 158 = 0$. □

O enunciado matemático do Problema da Partição de Números consiste em encontrar um conjunto $A \neq \emptyset$ que minimize a função:

$$f(A) = \left| \sum_{x \in A} x - \sum_{x \in S \setminus A} x \right| \tag{1.1}$$

A partir desse problema, encontram-se duas generalizações distintas, abordadas nos tópicos a seguir.

1.1.2 Problema da k-Partição de Números

Esta primeira generalização do TWNPP é o Problema da k-Partição de Números (*Multi-way Number Partitioning Problem* – MWNPP), que expande o número de subconjuntos nos quais se pretende distribuir o conjunto S.

Dado um conjunto numérico S, o objetivo é encontrar uma partição de dimensão k, de modo que as somas dos elementos de cada parte sejam as mais próximas possíveis umas das outras. Isso se resume a ter a parte de maior soma se aproximando da parte de menor soma tanto quanto for possível. Os textos que abordam essa versão são, dentre outros, Karmarkar e Karp (1982), Korf (2009), Korf et al. (2013). Um trabalho recente de forte interesse, pela sua completude, é Schreiber (2014).

O exemplo a seguir mostra uma situação desse problema.

Exemplo 1.1.2 *O conjunto* $S = \{87, 6, 5, 45, 34, 2, 24, 12, 7, 6, 54, 34\}$ *pode ser particionado para* $k = 3$ *em:*

$$\underbrace{\{87, 12, 6\}}_{soma=105}, \ \underbrace{\{45, 34, 6, 24\}}_{soma=109}, \ \underbrace{\{5, 7, 2, 54, 34\}}_{soma=102}$$

O valor da função objetivo dessa partição é $109 - 102 = 7$*, mas a partição ótima é*

$$\underbrace{\{87, 12, 6\}}_{soma=105}, \ \underbrace{\{45, 34, 2, 24\}}_{soma=105}, \ \underbrace{\{5, 7, 6, 54, 34\}}_{soma=106}$$

com valor de função objetivo $106 - 105 = 1$*. Observe que as partes possuem dimensões (número de elementos) diferentes.* □

A entrada é um conjunto S e um inteiro positivo k e a saída é uma partição de S, na forma $\{A_1, A_2, ..., A_k\}$. O objetivo é minimizar a função:

$$f(\{A_1, A_2, ..., A_k\}) = \max_i \left\{ \sum_{x \in A_i} x \right\} - \min_j \left\{ \sum_{x \in A_j} x \right\} \tag{1.2}$$

A função 1.2 é uma versão simplificada do diâmetro de um conjunto com k números reais $\sum_{x \in A_j} x$ mostrada abaixo.

$$f(\{A_1, A_2, ..., A_k\}) = \max_{(i,j)} \left\{ \sum_{x \in A_i} x - \sum_{x \in A_j} x \right\} \tag{1.3}$$

1.1.3 Problema Multidimensional da Partição de Números

A segunda generalização do (TWNPP) é o Problema Multidimensional da Partição de Números (*Multidimensional Two-Way Number Partitioning Problem* – MDTWNPP). Este problema transforma um conjunto S de inteiros em um conjunto de vetores inteiros $S = \{v_i \in \mathbb{Z}_+^m : \quad \forall i \in \{1, 2, 3, ..., n\}\}$. Para definir este problema, faz-se necessário o uso de uma norma em $\mathbb{R}^m$.

Seu enunciado matemático, proposto por Kojić (2010), é muito parecido com o (TWNPP), porém, após somar os elementos de cada um dos subconjuntos, tem-se um vetor, cuja norma será o valor do objetivo. A busca é um conjunto $A \neq \emptyset$ que minimize a função:

$$f(A) = \left\| \sum_{x \in A} x - \sum_{x \in S \setminus A} x \right\|_p \tag{1.4}$$

Quando $p \to \infty$, a função $f(\cdot)$ retorna a maior coordenada do vetor $\sum_{x \in A} x - \sum_{x \in S \setminus A} x$ sendo equivalente a função objetivo do artigo citado.

Seja $v \in \mathbb{Z}_+^m$ e v_{ij} com i indexando o vetor v variando de 1 até n e j sendo a coordenada do vetor v e vai de 1 até m. O conjunto I_n é o conjunto dos índices i e a variável de decisão B é um subconjunto de I_n.

$$f(B) = \max_j \left\{ \sum_{i \in I_n \setminus B} v_{ij} - \sum_{i \in B} v_{ij} \right\} \tag{1.5}$$

Exemplo 1.1.3 *O conjunto*

$$S = \{(8,3,6,3), (8,3,12,3), (8,3,0,3), (17,5,10,5), (10,5,10,5), (10,5,10,3), (8,3,6,1), (5,3,6,3)\}$$

com $m = 4$ *e* $k = 2$ *tem pode ser particionado como:*

$$A_1 = \underbrace{\{(17,5,10,5), (8,3,0,3), (10,5,10,5), (10,5,10,3)\}}_{soma=(45,18,30,16)},$$

$$A_2 = \underbrace{\{(8,3,6,3), (8,3,12,3), (8,3,6,1), (5,3,6,3)\}}_{soma=(29,12,30,10)}$$

com um valor de função objetivo $\max\{|(45,18,30,16) - (29,12,30,10)|\} = \max\{|(16,6,0,6)|\} = 16$*, mas tem partição ótima*

$$A_1 = \underbrace{\{(17,5,10,5), (10,5,10,5), (10,5,10,3)\}}_{soma=(37,15,30,13)},$$

$$A_2 = \underbrace{\{(8,3,6,3), (8,3,12,3), (8,3,0,3), (8,3,6,1), (5,3,6,3)\}}_{soma=(37,15,30,13)}$$

com valor de função objetivo $\max\{|(37,15,30,13) - (37,15,30,13)|\} = \max\{|(0,0,0,0)|\} = 0$ □

1.1.4 Problema Multidimensional da k-Partição de Números

O Problema Multidimensional da k-Partição de Números (*Multidimensional Multi-Way Number Partitioning Problem* – MDMWNPP é uma generalização tanto do (MWNPP) quanto do (MDTWNPP). Neste problema, um conjunto de vetores de $\mathbb{Z}_+^m$ deve ser dividido em k conjuntos, de modo que o diâmetro desses k vetores resultantes seja o menor possível. O exemplo 1.1.4 ilustra essa situação.

Exemplo 1.1.4 *Seja o conjunto:*

$$\begin{aligned} S \;=\; \{&(37,15,30,13),(8,3,6,3),(8,3,12,3),(8,3,0,3),(17,5,10,5),\\ &(10,5,10,5),(10,5,10,3),(8,3,6,1),(5,3,6,3)\} \end{aligned}$$

Para $m = 4$ e $k = 3$, pode ser particionado em:

$$A_1 \;=\; \underbrace{\{(17,5,10,5),(10,5,10,5),(10,5,10,3)\}}_{soma=(37,15,30,13)},$$

$$A_2 \;=\; \underbrace{\{(37,15,30,13),(8,3,0,3)\}}_{soma=(45,18,30,16)},$$

$$A_3 \;=\; \underbrace{\{(8,3,6,3),(8,3,12,3),(8,3,6,1),(5,3,6,3)\}}_{soma=(29,12,30,10)}$$

com valor de função objetivo:

$$\max\{|(45,18,30,16)-(29,12,30,10)|,|(37,15,30,13)-(29,12,30,10)|,$$
$$|(45,18,30,16)-(37,15,30,13)|\} = \max\{|(16,6,0,6)|,|(8,3,0,3)|,|(8,3,0,3)|\} = 16$$

mas com partição ótima:

$$A_1 \;=\; \underbrace{\{(17,5,10,5),(10,5,10,5),(10,5,10,3)\}}_{soma=(37,15,30,13)},$$

$$A_2 \;=\; \underbrace{\{(37,15,30,13)\}}_{soma=(37,15,30,13)},$$

$$A_3 \;=\; \underbrace{\{(8,3,6,3),(8,3,12,3),(8,3,0,3),(8,3,6,1),(5,3,6,3)\}}_{soma=(37,15,30,13)}$$

com valor de função objetivo igual a 0 pois todas as partes tem igual soma. Sendo assim o diâmetro desse conjunto é a maior coordenada de 3 vetores nulos. □

Ou seja, o objetivo é equivalente a minimizar a maior distância entre dois de todos os vetores, dada pela expressão:

$$f(\{A_1,A_2,...,A_k\}) = \max_{(i,j)}\left\{\left\|\sum_{x\in A_i} x - \sum_{x\in A_j} x\right\|_p\right\} \tag{1.6}$$

A expressão (1.6) pode ser utilizada como a função objetivo de qualquer um dos problemas enunciados acima, seja do Problema apresentado na Seção 1.1.1, seja do Problema apresentado na Seção 1.1.4 atual, sem qualquer inconsistência.

O trabalho de Pop e Matei (2013) adapta um outro modelo para o MDMWNPP, a partir da formulação matemática apresentada em Kojić (2010) para o MDTWNPP mostrado na Subseção 1.1.3.

Seja $v_i \in \mathbb{Z}_+^m$ e $V_l = \sum_{v_i \in A_l} v_i$. A função objetivo é a maior diferença entre duas coordenadas de V_{l_1} e V_{l_2} com $l_1 \neq l_2$, na forma:

$$f(\{A_1, A_2, ..., A_k\}) = \max_j \left\{ \left| \max_l \sum_{i \in A_l} v_{ij} - \min_l \sum_{i \in A_l} v_{ij} \right| \right\} \tag{1.7}$$

Nesta expressão, l indexa as partes e vai de 1 até k; i vai de 1 até n e indexa o vetor v dentro da parte A_l; j é a coordenada do vetor v e vai de 1 até m.

Note que, ao contrário de todos os enunciados até aqui, a expressão (1.7) não é igual à expressão (1.6) quando $p \to \infty$. Esta formulação está fora do padrão da Definição 1.1.5.

A Figura 1.1 resume as relações entre os quatro problemas apresentados.

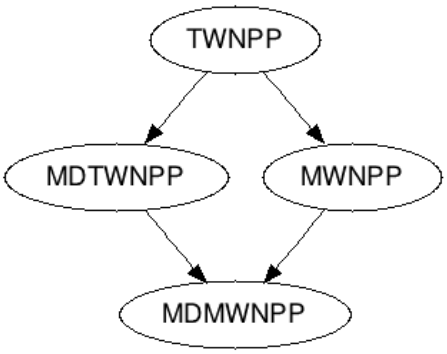

Figura 1.1: Relações entre os 4 problemas.

1.2 Organização do Projeto

O restante do texto está organizado segundo os comentários em cada capítulo a seguir. No Capítulo 2, mostra-se o problema tratado com uma visão mais detalhada, usando enunciado, modelo matemático proposto e uma visão mais teórica sobre como encontrar suas soluções. O Capítulo 3 contém algoritmos para problemas auxiliares aos métodos propostos do Capítulo 5. O Capítulo 4 contém um histórico de trabalhos sobre temas relacionados ao mesmo problema tratado nesse texto. O Capítulo 5 apresenta algoritmos propostos já implementados e testados, junto com possíveis abordagens ainda não definitivas. Apresenta-se o que é esperado para as propostas desse trabalho no Capítulo 6. Um conjunto de metas até a defesa da dissertação está no Capitulo 7. Os resultados do Capítulo 8 são preliminares e se encontram publicados.

Capítulo 2

Caracterização do Problema

Este capítulo apresenta o problema abordado neste projeto de dissertação. Está organizado da seguinte forma: a Seção 2.1 apresenta a definição do problema e discute sua dificuldade combinatória. A Seção 2.2 introduz uma formulação matemática para o problema em Otimização Linear Inteira. A Seção 2.3 mostra uma solução teórica exata.

2.1 Definição

O Problema da k-Partição de Números (*Multi-Way Number Partitioning Problem* – MWNPP apresentado na Seção 1.1.2 é a versão abordada, originalmente, em Karmarkar e Karp (1982). Sua entrada é um conjunto S e a saída é uma partição de S com tamanho k.

Definição 2.1.1 *Seja $S = \{a_1, a_2, ..., a_n\}$ um conjunto de inteiros positivos a_l, $l = 1, \ldots, n$, e k um número inteiro positivo. O Problema da k-Partição de Números consiste em encontrar uma k-partição de S, na forma $\{A_1, A_2, ..., A_k\}$, que minimize a função f definida por:*

$$f(\{A_1, A_2, ..., A_k\}) = \max_i \left\{ \sum_{x \in A_i} x \right\} - \min_j \left\{ \sum_{x \in A_j} x \right\} \tag{2.1}$$

□

A dificuldade combinatória desse problema, quando resolvido por uma enumeração ingênua das k-partições de S, é dada pelo Número de Stirling $S(n, k)$[1], que conta a quantidade de formas distintas de se repartir um conjunto de tamanho n em k partes. Encontra-se uma análise mais detalhada do Número de Stirling em Stanley (1997, pág. 80ff) e em Griffiths e Mezo (2010).

Defina $S(n-1, k)$ como o número de partições com k partes de um conjunto com $n-1$ elementos. Claro que o número de n-partições de um conjunto com n elementos é 1 (ou seja, $S(n, n) = 1$) e o número de 1-partição de qualquer conjunto é 1 (ou seja, $S(n, 1) = 1$). Se um n-ésimo elemento for adicionado ao conjunto, é possível descobrir $S(n, k)$ usando duas informações: $S(n-1, k-1)$ e $S(n-1, k)$.

[1] `http://mathworld.wolfram.com/StirlingNumberoftheSecondKind.html`

Inserindo o n-ésimo elemento como a k-ésima parte, existem $S(n-1,k-1)$ possibilidades. Inserindo o n-ésimo elemento em uma das k partes já existentes, contadas por $S(n-1,k)$, tem-se $k\ S(n-1,k)$ possibilidades:

$$S(n,k) = S(n-1,k-1)\ +\ k\ S(n-1,k) \tag{2.2}$$

Exemplo 2.1.1 *Seja o conjunto de inteiros* $A = \{1,2,3\}$*, para* $n = 3$ *e considere* $k = 2$*. Assim, o Número de Stirling produz* $S(3,2) = 3$*. Desse modo as partições são dadas por:*

$$\{1,2\},\{3\};$$
$$\{1,3\},\{2\};$$
$$\{2,3\},\{1\};$$

Seja agora o mesmo conjunto de inteiros $A = \{1,2,3\}$*, para* $n = 3$ *e considere* $k = 1$*. Assim, o número de Stirling leva a* $S(3,1) = 1$ *e à partição:*

$$\{1,2,3\}$$

Considere agora o conjunto de inteiros $A = \{1,2,3,4\}$ *para* $n = 4$ *e seja* $k = 2$*. Desse modo, as partições são:*

$$\{1,2,3\},\{4\};\{1,2\},\{3,4\};$$
$$\{1,2,4\},\{3\};\{1,3\},\{2,4\};$$
$$\{1,3,4\},\{2\};\{1,4\},\{2,3\};$$
$$\{2,3,4\},\{1\}$$

Ou seja, a partir da expressão (2.2), tem-se que:

$$S(4,2) = S(3,1) + 2.S(3,2) = 7$$

□

Além de enumerar todas as partições de tamanho k, ainda é justo inserir um fator multiplicador $O(n)$, responsável pelo custo de se avaliar a função objetivo. No fim dos cálculos, a ordem de complexidade do algoritmo ingênuo seria $O(n.S(n,k))$.

O Número de Stirling não tem fórmula fechada. Uma outra alternativa para expressar essa quantidade, vista em Stanley (1997) é a expressão:

$$S(n,k) = \frac{1}{k!}\sum_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^n \tag{2.3}$$

Esse número supera as complexidades de ordem exponenciais quando o valor de k segue a sequência A024417[2]. Se $a(n) \in A024417$ então $S(n,a(n)) \geq S(n,k), \quad \forall k$. Esse fato pode ser conferido em A002870[3], a sequência do Maior Número de Stirling indexada em n. Se $a(n) \in A024417$, então $S(n,a(n)) \in A002870$. O gráfico abaixo mostra casos exponenciais em escala logarítmica comparados ao n-ésimo Maior Número de Stirling. Este gráfico é composto por retas sendo cortadas por uma função crescente.

[2] `http://oeis.org/A024417`
[3] `http://oeis.org/A002870`

A *On-Line Encyclopedia of Integer Sequences* - OEIS é uma base de dados pública com variadas sequências de números inteiros criada por (Sloane, 1991). Nela é possível buscar uma sequência numérica apenas digitando seus primeiros termos e, também, encontrar os principais artigos ligados a mesma. É a fonte das sequências A002870 e A024417.

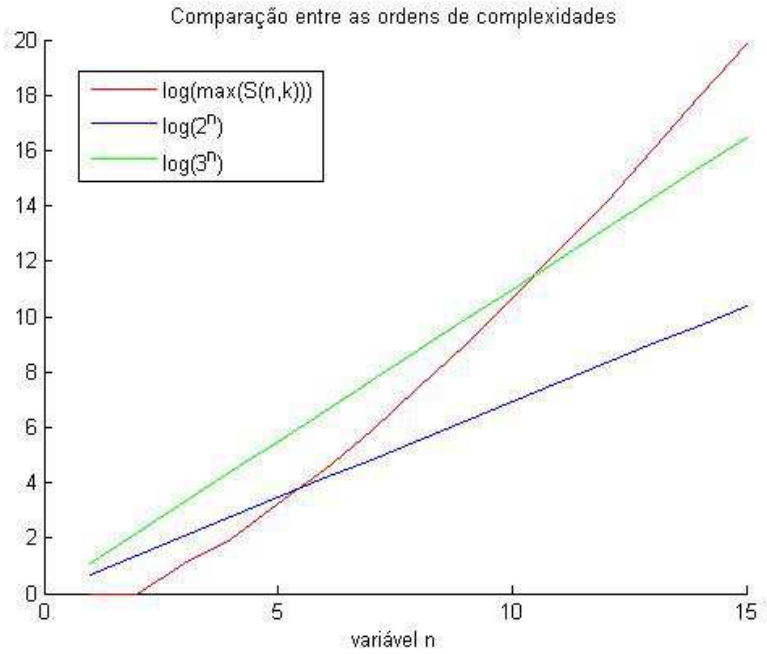

Figura 2.1: O maior dos Números de Stirling vs funções exponenciais. Dados retirados de Sloane (1991)

Esta figura mostra que o Maior Número de Stirling domina assintoticamente funções exponenciais, ou seja, para todo $c > 1$ existe um n suficientemente grande tal que$S(n, a(n)) > c^n$.

2.2 Formulação Matemática

O modelo matemático de Otimização Linear Inteira proposto nesse trabalho, como contribuição efetiva, usa duas variáveis inteiras para representar o limite inferior t_1 e superior t_2, para a soma dos elementos de todas as k partes. As demais variáveis x_{ij} são binárias e informam se um elemento está ou não numa partição. A notação $I_m = \{y \in \mathbb{Z} : \quad 1 \leq y \leq m\}$ significa todos os inteiros entre 1 e m, inclusive.

$$\min \quad t_2 - t_1 \tag{2.4}$$

$$\text{suj. a} \quad t_1 \leq \sum_{i=1}^{n} a_i x_{ij} \leq t_2, \quad \forall j \in I_k \tag{2.5}$$

$$\sum_{j=1}^{k} x_{ij} = 1, \quad \forall i \in I_n \tag{2.6}$$

$$t_1, t_2 \in \mathbb{Z}_+ \tag{2.7}$$

$$x_{ij} \in \{0, 1\} \tag{2.8}$$

A relação de pertinência (2.8) indica que $x_{ij} = 1$ se o elemento de índice i do conjunto S pertence à parte de índice j. A relação (2.7) informa que os limitantes da maior e menor soma dos elementos das partições é sempre positiva. As n equações indicadas pela expressão (2.6) garantem que as partições são disjuntas e que todos os elementos de S estão alocados em alguma parte, já que o problema só está bem definido quando $n > k$. As k inequações (2.5) mostram que o modelo garante que cada partição é não vazia, pois $t_1 > 0$, por efeito de (2.7) e, também, que estão todas contidas no intervalo $[t_1, t_2]$. Por fim, a expressão (2.4) é o tamanho do intervalo que contém todas as k somas dos elementos das partes e o objetivo de minimização.

2.3 Visão Teórica

Uma abordagem mais algorítmica do problema sugere a construção de um algoritmo polinomial para o (MWNPP), usando um algoritmo hipotético que retorna sim (1) ou não (0) para uma pergunta específica sobre a entrada. Esse algoritmo chama-se Oráculo e é denotado por $Or(S, S')$, sendo S e S' duas instâncias de tamanho n e $n-1$, respectivamente, do (MWNPP).

A pergunta em questão é:

Pergunta 2.3.1 *As entradas S e S' têm o mesmo valor ótimo de função objetivo ?*

O exemplo 2.3.1 mostra uma situação associada ao uso desse Oráculo.

Exemplo 2.3.1 *Suponha que $k = 3$, $S' = \{1, 2, 3, 4\}$ e $S = \{1, 2, 3, 5, 6\}$. O resultado de $Or(S, S') = Or(\{1, 2, 3, 5, 6\}, \{1, 2, 3, 4\}) = 1$ (SIM) pois a 3-partição ótima de S pode ser: $\{1, 2, 3\}, \{5\}, \{6\}$ ou $\{2, 3\}, \{1, 5\}, \{6\}$ ambas com valor de função objetivo igual a 1. Assim como S', que tem 3-partição ótima $\{1, 2\}, \{3\}, \{4\}$, com valor de função objetivo igual a 1.* □

A ideia central dessa técnica é decidir se dois elementos distintos do conjunto S pertencem ou não à mesma partição, com a troca de dois elementos do conjunto pela soma dos mesmos. Sempre que essa troca não mudar o valor ótimo, significa que os dois valores substituídos pertencem à mesma partição.

Exemplo 2.3.2 *Inicialmente, $S = \{1, 2, 3, 5, 6\}$. Cria-se um novo conjunto $S' = \{1 + 2, 3, 5, 6\}$. Fazendo a chamada do oráculo sobre S e S', descobre-se que 1 e 2*

estão contidos na mesma parte quando $Or(S, S') = 1$ *e que estão em partes distintas quando caso contrário. Nesse exemplo, é claro que* $Or(\{1, 2, 3, 5, 6\}, \{3, 3, 5, 6\}) = 1$ *(veja no exemplo 2.3.1).* □

Algoritmo 1: Algoritmo exato hipotético

Entrada: $Or()$ e um conjunto S
Saída: $\{A_1, A_2, ..., A_k\}$
início
  $j = 1$;
  enquanto $S \neq \varnothing$ **faça**
    $k = 1$;
    $insere(A_j, a_k)$;
    para $a_i \in S - \{a_k\}$ **faça**
      $S' = S$;
      $remove(S', a_i)$;
      $insere(S', a_i + a_k)$;
      se $Or(S, S') = 1$ **então**
        $insere(A_j, a_i)$;
        $S = S'$;
      fim
    fim
    $j = j + 1$;
  fim
  retorna $\{A_1, A_2, ..., A_k\}$;
fim

O funcionamento do algoritmo é simples e direto. Primeiro, o elemento a_1 vai para a parte A_1, sendo o elemento líder da primeira parte. Em seguida, para cada $a_i \in S - \{a_1\}$, o elemento a_i é trocado por $a_1 + a_i$, formando um novo conjunto S^i. Executa-se $Or(S, S^i)$ e, sempre que a resposta do oráculo for positiva, retira-se a_i de S, armazenando-o em A_1 e S é trocado por S^i. Repete-se o procedimento até $S = \varnothing$, $k - 1$ vezes.

No pior caso, este algoritmo faz $O(n^2)$ chamadas ao oráculo, pois compara o elemento líder de A_j com cada um dos restantes em S.

Capítulo 3

Fundamentação Teórica

Este Capítulo apresenta os fundamentos básicos relacionados ao tema do projeto. A Seção 3.1 apresenta o Algoritmo (LPT). A Seção 3.2 mostra o Algoritmo 0/1-*Interchange*, ambos desenvolvidos para problemas de escalonamento de máquinas. A Seção mochila discute a relação do Problema da k-Partição de Números com o Problema da Mochila. A Seção apresenta a versão geral da metaheurística *Iterated Local Search* (ILS). Por fim, a Seção 3.5 introduz a Heurística de Karmarkar-Karp e sua adaptação ao Problema da k-Partição de Números.

3.1 Algoritmo (LPT)

Proposto por Graham (1966, 1969), o algoritmo guloso *Longest Processing Time first* (LPT) foi desenvolvido para um problema de Partição Mínima e para o Problema de Escalonamento de Tarefaz diferente do Problema da k-Partição de Números (MWNPP), como comenta Korf (2010).

Este algoritmo garante um resultado com razão de aproximação menor que $\frac{4}{3}-\frac{1}{3k}$ do ótimo global em relação à função objetivo $\max\limits_{1\leq i\leq k}\left\{\sum\limits_{x\in A_i} x\right\}$. Isso significa que, se $OPT(S)$ é o valor ótimo da função objetivo e $LPT(S)$ o valor de função objetivo do LPT para uma instância S, a razão entre estes valores é tal que:

$$\frac{LPT(S)}{OPT(S)} \leq \frac{4}{3} - \frac{1}{3k} \tag{3.1}$$

A demonstração de aproximabilidade se encontra em Ausiello et al. (2003).

Exemplo 3.1.1 *O resultado ótimo do Problema da k-Partição de Números para a instância:*

$$S = \{8, 7, 6, 5, 4\}$$

e $k = 2$ *é*

$$\{8, 7\} \quad e \quad \{6, 5, 4\}$$

A parte com a maior soma é $max\{8+7,\ 6+5+4\} = 15$. *Substituindo* $k = 2$ *na desigualdade 3.1 tem-se* $\left(\frac{4}{3}-\frac{1}{6}\right)\ 15 = \frac{7}{6}\ 15 = 17.5$.*Portanto, a resposta do algoritmo (LPT) sempre é uma partição em que a maior soma é menor que* 17.5 *como:*

$$\{8, 5, 4\} \quad e \quad \{7, 6\}$$

cuja maior soma é 17*, resultado encontrado pelo (LPT), mas nunca um valor* 18 *como a partição abaixo:*

$$\{8, 4\} \quad e \quad \{7, 6, 5\}$$

□

O algoritmo consiste em ordenar o conjunto S em ordem não crescente e alocar os números, sempre inserindo o novo elemento na parte de menor soma e, em caso de empate, desempata-se arbitrariamente. Este algoritmo tem um custo computacional $O(nlog(n))$ para ordenar a entrada e um custo $O(n)$ para alocar os elementos nas k partes. Finalmente, a complexidade do método é $O(nlog(n))$.

O Algoritmo *Longest Processing Time first* (LPT) está apresentado no Algoritmo 2.

Algoritmo 2: *Longest Processing Time* para k máquinas

Entrada: Conjunto S de números inteiros e um inteiro k
Saída: Partição do conjunto S, $\{A_1, A_2, ..., A_k\}$, com tamanho k
início
  para $j \in \{1, ..., k\}$ **faça**
    $A_j = \emptyset$;
    $L_j = 0$;
  fim
  Ordene S em ordem decrescente;
  para $a \in S$ **faça**
    $l = \arg\min_j L_j$;
    $A_l = A_l \cup a$;
    $L_l = L_l + a$;
  fim
  retorna $\{A_1, A_2, ..., A_k\}$
fim

3.2 Algoritmo 0/1-*Interchange*

O Algoritmo 0/1-*Interchange*, proposto por Finn e Horowitz (1979), também é um método para o Problema de Escalonamento em k máquinas idênticas. Partindo de uma distribuição inicial das tarefas, o algoritmo ordena os processadores em ordem não decrescente.

Seja $d_i = L(A_1) - L(A_k)$ a diferença entre o instante de finalização do processador mais carregado e do menos carregado na iteração i. Enquanto existir uma tarefa $p \in A_1 \ : \ p < d_i$, está será realocada para A_k. Este algoritmo tem uma complexidade de tempo computacional da ordem $O(n.log(k))$, como demonstrado por Langston (1982)

O Algoritmo 0/1-*Interchange* está apresentado no Algoritmo 3.

Algoritmo 3: Algoritmo 0/1-*Interchange*

Entrada: K-Partição $\{A_1, A_2, ..., A_k\}$ do conjunto S
Saída: K-Partição do conjunto S, $\{A_1, A_2, ..., A_k\}$
início
  Ordene os tempos de cada máquina $L(A_1) \geq ... \geq L(A_k)$;
  $d = L(A_1) - L(A_k)$;
  enquanto $\exists q \in A_1 \ : \ L(q) < d$ **faça**
    $remove(A_1, q)$;
    $insere(A_k, q)$;
    $L(A_1) = L(A_1) - L(q)$;
    $L(A_k) = L(A_k) + L(q)$;
    $d = L(A_1) - L(A_k)$;
  fim
  retorna $\{A_1, A_2, ..., A_k\}$;
fim

3.3 Problema da Mochila/Soma de Subconjuntos

O Problema da Mochila é um dos mais estudados em Otimização Combinatória e costuma ser um subproblema muito comum em vários trabalhos. Todas as variações do Problema da Mochila são NP-completos, mas admitem uma solução pseudo-polinomial, como pode ser encontrado em Garey e Johnson (1979) e Ausiello et al. (2003).

Definição 3.3.1 *Seja $W \in \mathbb{N}$ e um conjunto de itens $I = \{1, 2, ..., n\}$, tal que cada elemento de I indexe um peso w_i e um valor p_i. Encontre um subconjunto de itens que maximize o valor da mochila, respeitando sua capacidade de peso, dada por W.* □

A formulação matemática para esse problema é:

$$\max \quad \sum_{j=1}^{n} p_j x_j \tag{3.2}$$

$$\text{s.a} \quad \sum_{j=1}^{n} w_i x_j \leq W \tag{3.3}$$

$$x_j \in \{0, 1\} \tag{3.4}$$

O Problema da Soma de Subconjunto é um caso particular do Problema da Mochila, caracterizado por $\forall i \in I \ : \ w_i = p_i$. O algoritmo pseudo-polinomial para resolve-lo tem custo $O(nW)$ e, por isso, costuma ser o mais indicado quando o conjunto de itens I possui grande dimensão, mas o número W tem poucos algarismos. Caso contrário, os algoritmos exatos mais eficientes são os propostos por Horowitz e Sahni (1974) e Schroeppel e Shamir (1981). A ideia da solução é a seguinte:

- Caso o item i faça parte da solução ótima considerando os itens $\{i, ..., n\}$, tem-se uma solução com valor p_i a mais do que a solução ótima para os itens $\{i+1, ..., n\}$, com capacidade restante $W - w_i$. Esta ideia corresponde a fazer $x_i = 1$, gerando o subproblema:

$$p_i + \max\left\{\sum_{j=i+1}^{n} p_j x_j \mid \sum_{j=i+1}^{n} w_j x_j \leq w - w_i, \quad x_j \in \{0,1\}\right\} \tag{3.5}$$

- Caso contrário, tem-se um valor correspondente à solução ótima para itens $\{i+1, ..., n\}$ com capacidade W. Esta ideia corresponde a fazer $x_i = 0$, gerando o subproblema

$$\max\left\{\sum_{j=i+1}^{n} p_j x_j \mid \sum_{j=i+1}^{n} w_j x_j \leq w, \quad x_j \in \{0,1\}\right\} \tag{3.6}$$

Assim, seja $M(i, w)$ o valor da solução ótima para os itens $\{i, ..., n\}$ e capacidade W, na forma:

$$M(i, w) = \max\{M(i+1, w), M(i+1, w - w_i) + p_i\}$$

e o valor ótimo do problema será $M(1, W)$.

O Algoritmo 4 formaliza o procedimento.

Algoritmo 4: Algoritmo exato para Soma de Subconjuntos

Entrada: Vetor v de tamanho n e uma capacidade W
Saída: Subconjunto S de índices de v
início
  Inicialize uma matriz $M \in \Re^{(n+1)\times(W+1)} = 0$;
  para *i=n até 1* **faça**
    para *w=0 até W* **faça**
      se $i > n$ *ou* $w = 0$ **então**
        $M(i, w) = 0$;
      fim
      se $w_i > w$ **então**
        $M(i, w) = M(i+1, w)$;
      fim
      se $max\{M(i+1, w), M(i+1, w - w_i) + w_i\}$ **então**
        $M(i, w) = 0$;
      fim
    fim
  fim
  retorna $M(1, W)$
fim

O Algoritmo 4 cria uma matriz $n \times W$ com elementos $M_{i,j}$ é o valor ótimo do Problema da Mochila com capacidade j usando os i primeiros itens de S.

3.4 *Iterated Local Search (ILS)*

A meta-heurística *Iterated Local Search* (ILS) parte do pressuposto de que os ótimos locais podem ser totalmente percorridos por um tipo de movimento. Assim como todas as permutações de n números podem ser expressas como um produto de transposições, o movimento definido para o (ILS) deve conseguir gerar todo o espaço de busca do problema abordado.

O (ILS) proposto em Lourenço et al. (2003) funciona, genericamente, com alguns passos fundamentais.

Gera-se uma solução inicial para o problema. Faz-se uma busca local da melhor solução na vizinhança. A vizinhança de x é tudo que pode ser alcançado partido de x com um único movimento. Perturba-se o espaço de soluções atuais quando este já estiver estagnado. A perturbação deve levar a próxima solução para fora da vizinhança da atual. Aceita-se ou não a solução atual de acordo com um critério, geralmente baseado no valor da função objetivo. O procedimento termina com o número máximo de iterações.

O Algoritmo 5 apresenta a estrutura típica do (ILS).

Algoritmo 5: Algoritmo ILS

Entrada: Gerador de solução inicial
Saída: Solução do espaço de busca
início
  $s_0 = SolucaoInicial$;
  $s = BuscaLocal(S_0)$;
  $i = 0$;
  enquanto $i < Itermax$ **faça**
    $i = i + 1$;
    $s' = Perturba(s)$;
    $s'' = BuscaLocal(s')$;
    $s = Criterio(s, s', s'')$;
  fim
  retorna s
fim

No Capítulo 5 na Seção 5.2, demonstram-se as principais funções do (ILS) de forma específica para o (MWNPP)

3.5 Heurística de Karmarkar-Karp para $k > 2$

A Heurística de Karmarkar-Karp, introduzida por Karmarkar e Karp (1982), é um método construtivo para o (TWNPP) que pode ser adaptado para o (MWNPP). Esse algoritmo escolhe alocar os dois maiores elementos de um conjunto, S, em partes distintas e, em seguida, adiciona a diferença entre os dois maiores elementos no conjunto como um novo elemento descontando o erro produzido pela alocação.

O pseudo-código da Heurística de Karmarkar-Karp está no Algoritmo 6.

Algoritmo 6: Heurística de Karmarkar-Karp com $k = 2$

Entrada: Um conjunto S
Saída: Um inteiro
início
  enquanto $|S| \neq 1$ **faça**
    $s1 = removemax(S)$;
    $s2 = removemax(S)$;
    $v = s1 - s2$;
    $insereordenado(S, v)$;
  fim
  retorna $v[0]$
fim

O Exemplo 3.5.1 mostra uma aplicação dessa heurística.

Exemplo 3.5.1 *Seja* $S = \{8, 7, 6, 5, 4\}$ *e* $k = 2$.

- *iteração 1:*$\{\mathbf{8}, \mathbf{7}, 6, 5, 4\} \Rightarrow \{6, 5, 4, 1\}$
- *iteração 2:*$\{\mathbf{6}, \mathbf{5}, 4, 1\} \Rightarrow \{4, 1, 1\}$
- *iteração 3:*$\{\mathbf{4}, \mathbf{1}, 1\} \Rightarrow \{3, 1\}$
- *iteração 4:*$\{\mathbf{3}, \mathbf{1}\} \Rightarrow \{2\}$

O procedimento retorna o valor de função objetivo igual a 2 *e a partição* $\{A_1, A_2\} = \{(8, 6), (7, 5, 4)\}$.

É importante observar que este resultado não se constitui na solução ótima do problema para esta instância.

Para uma implementação de baixo custo computacional, usando uma estrutura de dados `max-heap`, a complexidade desse algoritmo é $O(n.log(n))$.

A adaptação desse algoritmo para o (MWNPP) consiste em substituir a operação de diferença entre os dois maiores elementos do conjunto pelo Algoritmo 7 e seguir a mesma lógica de funcionamento do Exemplo 3.5.1. A diferença é que os elementos são vetores de dimensão k e não escalares. Logo, a estrutura de dados `max-heap` deve operá-los em ordem lexicográfica.

Algoritmo 7: Modificação na Heurística de Karmarkar-Karp para o (MWNPP)

Entrada: Dois vetores ordenados, x e y, e um inteiro k
Saída: Um vetor ordenado de tamanho k
$opera(x, y, k)$;
início
  $s[k]$;
  para $i = 0$ *até* $n - 1$ **faça**
    $s[i] = x[i] + y[n - i]$;
  fim
  ordene s em ordem não crescente;
  para $i = 0$ *até* $n - 1$ **faça**
    $s[i] = s[i] + s[n - 1]$;
  fim
  retorna s
fim

O pseudo-código usado para resolver o exemplo 3.5.2 está no Algoritmo 8. A única mudança em relação à Heurística de Karmarkar-Karp original é a troca de $v = s1 - s2$ por $v = opera(s1, s2, k)$.

Algoritmo 8: Heurística de Karmarkar-Karp com $k > 2$

Entrada: Um conjunto S, e um inteiro k
Saída: Um inteiro
início
  enquanto $|S| \neq 1$ **faça**
    $s1 = removemax(S)$;
    $s2 = removemax(S)$;
    $v = opera(s1, s2, k)$;
    $insereordenado(S, v)$;
  fim
  retorna $v[0]$
fim

O exemplo 3.5.2 mostra o funcionamento da Heurística de Karmarkar-Karp para $k > 2$ fazendo cada iteração em duas ou 3 fases para demonstrar o nova .

Exemplo 3.5.2 *Seja* $S = \{8, 7, 6, 5, 4\}$ *e* $k = 3$.

- *iteração 1:*$\{(\mathbf{8}, \mathbf{0}, \mathbf{0}), (\mathbf{7}, \mathbf{0}, \mathbf{0}), (6, 0, 0), (5, 0, 0), (4, 0, 0)\} \Rightarrow$
  $\{(8, 7, 0), (6, 0, 0), (5, 0, 0), (4, 0, 0)\}$

- *iteração 2:*$\{(\mathbf{8}, \mathbf{7}, \mathbf{0}), (\mathbf{6}, \mathbf{0}, \mathbf{0}), (5, 0, 0), (4, 0, 0)\} \Rightarrow \{(8, 7, 6), (5, 0, 0), (4, 0, 0)\} \Rightarrow$
  $\{(5, 0, 0), (4, 0, 0), (2, 1, 0)\}$

- *iteração 3:*$\{(\mathbf{5}, \mathbf{0}, \mathbf{0}), (\mathbf{4}, \mathbf{0}, \mathbf{0}), (2, 1, 0)\} \Rightarrow \{(5, 4, 0), (2, 1, 0)\}$

- *iteração 4:*$\{(\mathbf{5},\mathbf{4},\mathbf{0}),(\mathbf{2},\mathbf{1},\mathbf{0})\}\} \Rightarrow \{(5,5,3)\} \Rightarrow \{(3,3,0)\}$

O procedimento retorna um valor de função objetivo igual a 3 *e uma partição* $\{8\},\{7,4\},\{6,5\}$

Capítulo 4

Revisão Bibliográfica

Neste capítulo, apresentam-se os trabalhos correlatos disponíveis na literatura. Prioriza-se um conjunto de artigos e livros mais ligados as ideias usadas no capítulo 5 e os trabalhos mais importantes. Na seção 4.1, descrevem-se as publicações relacionadas ao (MWNPP) desde a origem até as mais recentes. Na seção 4.2, apresenta-se um panorama temporal, classificando as técnicas adotadas nos trabalhos correlatos por autor e ano.

4.1 Histórico

Existe uma vasta literatura sobre a solução do Problema da Partição de Números e suas variações. Ele foi listado de modo formal no trabalho de (Karp, 1972) como um dos 6 problemas NP-completos básicos e, mais tarde, por (Garey e Johnson, 1979). Ambos demonstram uma série de equivalências entre o (NPP) e outros problemas NP-completos.

O (MWNPP) aparece explicitamente num artigo sobre a análise de uma heurística construtiva, *Differencing Method*, mais conhecida como Heurística de Kamarkar-Karp proposta por (Karmarkar e Karp, 1982) focada na ideia de dividir os maiores números em partes distintas, inserindo as diferenças entre os elementos retirados do conjunto dos não alocados. Este é o artigo mais citado sobre o (MWNPP) até hoje. Os autores demonstram vários resultados sobre a eficiência da heurística proposta e sobre a dificuldade das instâncias do problema.

Mais tarde, (Michiels et al., 2003) demonstra uma razão de aproximação menor ou igual a $\frac{4}{3} - \frac{1}{3(k-1)}$, similar a do (LPT) descrito na seção 3.1, para a Heurística de Karmakar-Karp. Isso põe os resultados obtidos com a Heurística de Karmarkar-Karp dentro de um intervalo de erro limitado.

Existe, também, um ramo de estudo do interesse da física estatística. Muitos são motivados pela análise do problema feita por (Karmarkar e Karp, 1982). Esses trabalhos pesquisam uma forma de classificar as instâncias do (TWNPP) e seus relativos, buscando uma metodologia que garanta um conjunto livre de partições perfeitas. Partições perfeitas são quelas que tem um valor ótimo de função objetivo igual a 0 ou 1. Percebe-se que uma instância com muita variedade de partições perfeitas poderia ser resolvida até mesmo por uma enumeração ingênua porque não existem valores de função objetivo menores para se encontrar. Os trabalhos que apresentam essa pesquisa são: (Gent e Walsh, 1995), mostrando a transição de fases

para a identificar as características de uma instância livre de partições perfeitas; (Mertens, 2006), promove um avanço no trabalho de Gent em 1995 mostrando uma nova fórmula que classifica instâncias quase certamente livres de partições perfeitas e (Ferreira, 2001), que apresenta uma nova equação para o valor esperado da função objetivo do (TWNPP).

Alguns trabalhos como (Gent e Walsh, 1998) e (Berretta e Moscato, 1999) mostram que o (TWNPP) é um problema muito difícil para meta-heurísticas de uso geral como Algoritmo Genético, *Simulated Annealing* e outros. Em muitos casos esses métodos perdem em tempo e desempenho para a Heurística de Kamarkar-Karp e até mesmo para um algoritmo guloso, (LPT), como. Alguns textos mais recentes de (Kojić, 2010) e (Pop e Matei, 2013) usam várias classes de meta-heurísticas em versões mais pesadas do Problema da Partição de Números como o (MDTWNPP) e (MDMWNPP), mas também não mostram resultados muito significativos exceto pela formulação matemática.

Tanto para o (TWNPP) quanto pata o (MWNPP), sempre houve a possibilidade de usar os algoritmos de (Horowitz e Sahni, 1974) e (Schroeppel e Shamir, 1981) fazendo a conversão para o Problema da Mochila, mas o espaço de memória requerido é inviável para problemas de maior cardinalidade. Uma nova abordagem surgiu quando (Korf, 1998) propôs um algoritmo exato fazendo um procedimento *Backtrack* em heurísticas construtivas como o LPT e a Heurística de Karmarkar-Karp, *Complete Greedy Algorithm* e *Complete Karmarkar-Karp Algorithm* respectivamente. primeira melhoria desses trabalhos acontece *Recursive Number Partitioning* proposto por (Korf, 2009). A segunda melhoria é a contribuição de (Pedroso e Kubo, 2010). Uma nova estrutura de dados aplicada ao trabalho de Korf agilizando a busca na Árvore de Kamarmar-Karp. A discussão sobre a função objetivo do (MWNPP) por (Korf, 2010) mostra as diferenças dos objetivos de seus artigos e do de Karmarkar e Karp em 1972. Por meio de sucessivas conversões do (MWNPP) de uma partição de tamanho $k-1$ para k (Moffitt, 2013) propõe um algoritmo baseado em programação dinâmica. Atualmente o estado da arte são os algoritmos *Sequential Number Partitioning* do artigo de (Korf et al., 2013) e o *Cached Iterative Weakening* encontrado em (Schreiber e Korf, 2014). Um trabalho completo e de grande interesse sobre esses algoritmos com todos os detalhes encontra-se em (Schreiber, 2014).

4.2 Panorama

A tabela 4.1 segue a ordem cronológica das publicações mais relevantes sobre o (MWNPP)

- Objetivos: MmaxP - minimizar a parte com maior soma, MminP - maximizar a parte com menor soma, MdianP - minimizar o diâmetro das somas das partes, Mpitem - maximizar o valor dos itens na mochila;
- Método: A - aproximado, E - exato, H - heurístico, MH - metaheurístico, Est - estatístico;
- Complexidade: $O(2^{\frac{n}{2}})$, $O(n.log(n))$, $O(2^n)$, $O(k^n)$, $O(\frac{k^n}{k!})$ sendo ou não amortizadas;

- Gerais: # - nulo, ? - incerto, ! - variado.

Autor	Problema	Objetivo	Método	Complexidade
(Karp, 1972)	Coletânea	#	#	#
(Horowitz e Sahni, 1974)	Mochila	Mpitem	E	$O(2^{\frac{n}{2}})$
(Garey e Johnson, 1979)	Coletânea	#	#	#
(Schroeppel e Shamir, 1981)	Mochila	Mpitem	E	$O(2^{\frac{n}{2}})$
(Karmarkar e Karp, 1982)	MWNPP	MdiaP	H	$O(n.log(n))$
(Gent e Walsh, 1995)	TWNPP	MmaxP, MminP, MdiaP	Est	#
(Gent e Walsh, 1998)	TWNPP	MmaxP, MminP, MdiaP	Est, MH, H	(!)
(Korf, 1998)	TWNPP	MmaxP, MminP, MdiaP	E	$O(2^n)$, amortizado
(Berretta e Moscato, 1999)	TWNPP	MmaxP, MminP, MdiaP	MH, H	(!)
(Ferreira, 2001)	TWNPP	MmaxP, MminP, MdiaP	Est	#
(Hayes, 2002)	TWNPP	MmaxP, MminP, MdiaP	Est	#
(Mertens, 2006)	TWNPP	MmaxP, MminP, MdiaP	Est	#
(Michiels et al., 2003)	MWNPP	MmaxP	A	$O(n.log(n))$
(Korf, 2009)	MWNPP	MmaxP	E	$O(k^n)$, amortizado
(Pedroso e Kubo, 2010)	TWNPP	MmaxP, MminP, MdiaP	E	$O(2^n)$, amortizado
(Korf, 2010)	MWNPP	MmaxP, MminP, MdiaP	#	#
(Kojić, 2010)	MDTWNPP	MdianP	MH	#
(Pop e Matei, 2013)	MDMWNPP	MdianP (?)	MH	#
(Moffitt, 2013)	MWNPP	MmaxP	E	$O(k^n)$, amortizado
(Korf et al., 2013)	MWNPP	MmaxP	E, H	$O(\frac{k^n}{k!})$, amortizado
(Schreiber e Korf, 2014)	MWNPP	MmaxP	E, H	$O(\frac{k^n}{k!})$, amortizado
(Schreiber, 2014)	MWNPP	MminP	E, H	(!)

Tabela 4.1: Quadro das contribuições mais relevantes

Os artigos onde o item "Problema"está classificado como "Coletânea"são abordagens teóricas sobre problemas NP-completos, por isso, "Objetivo", "Método"e "Complexidade"estão marcados com "#".

Nos artigos onde o "Método"está marcado com apenas "Est", não existe nenhum algoritmo proposto, mas sim, uma análise da dificuldade das instâncias.

O termo "amortizado"significa que o algoritmo faz cortes em suas ramificações e quase nunca recai no pior caso.

Capítulo 5

Algoritmos Propostos para a solução do Problema da k-Partição de Números

Este capítulo apresenta Algoritmos heurísticos e exatos propostos para a solução do Problema da k-Partição de Números (MWNPP). O Capítulo é iniciado com Seção 5.1, em que uma proposta de algoritmo geral para a solução do Problema da k-Partição de Números é introduzido. Em seguida, a Seção 5.2 mostra uma proposta de adaptação da meta-heurística *Iterated Local Search* (ILS) para a solução do problema em análise. Por fim, a Seção refkpart-exato mostra uma proposta de algoritmo exato para a solução do Problema da k-Partição de Números.

5.1 Algoritmos

Seja S um conjunto de números e considere a existência de um algoritmo, denominado $part_2(S)$, que resolve o Problema de Partição de Números (TWNPP) e um algoritmo, $part_1(S, k)$, que encontra uma solução inicial para o (MWNPP). É possível, então, propor um algoritmo, inspirado no Algoritmo 0/1-*Interchange*, mostrado no Algoritmo 3 da Seção 3.2, que solucione o Problema da k-Partição de Números (MWNPP).

O Algoritmo 9 contempla esta proposição, sendo uma proposta de solução geral para o Problema da k-Partição de Números.

Algoritmo 9: Algoritmo de solução Geral para o (MWNPP)

Entrada: Um conjunto S com n elementos, $part_1$, $part_2$ e um inteiro k
Saída: K-Partição do conjunto S
início
  $\{A_1, ..., A_k\} = part_1(S, k)$;
  repita
    $obj = \max_i \left\{ \sum_{x \in A_i} x \right\} - \min_j \left\{ \sum_{x \in A_j} x \right\}$;
    Encontre $A_i = \arg\max_i \left\{ \sum_{x \in A_i} x \right\}$ e $A_j = \arg\min_j \left\{ \sum_{x \in A_j} x \right\}$;
    $X = A_i \cup A_j$;
    $\{A_i, A_j\} = part_2(X)$;
  até $obj == \max_i \left\{ \sum_{x \in A_i} x \right\} - \min_j \left\{ \sum_{x \in A_j} x \right\}$;
fim

Após a linha 9 do Algoritmo 9 descrito acima, pode-se listar dois casos: (i) a parte da maior soma fica menor e a parte de menor soma fica maior ou (ii) ambas as partes permanecem inalteradas. Enquanto o primeiro caso ocorre, a função objetivo do problema decresce. Portanto, o ótimo ainda não foi encontrado.

Observe que o Algoritmo 9 é um algoritmo geral para a solução do Problema da k-Partição de Números. O algoritmo $part_2$ é parte desse algoritmo geral assim como o $part_1$ que retorna uma solução inicial. O Algoritmo 9 cumpre a função de gerar a partição do conjunto que leve à uma solução viável do (MWNPP). Qualquer método que execute as funções de $part_1$ e $part_2$ pode vir a ser implementado, seja um método exato, uma heurística ou uma meta-heurística.

A Seção 5.2 apresenta uma adaptação da meta-heurística (ILS), mostrada em sua versão geral no Algoritmo 5 da Seção 3.4, para atuar com a função do algoritmo $part_2$.

5.2 Adaptação do ILS para a Solução do Problema da k-Partição de Números

Esta seção apresenta a adaptação da meta-heurística *Iterated Local Search* (ILS) para a etapa de execução do método $part_2$ do algoritmo geral de solução constituído pelo Algoritmo 9.

Com uma solução inicial já bem aproximada para o problema, basta encontrar uma boa estratégia para rearranjar alguns elementos e buscar uma solução ainda melhor. O refinamento acontece pela busca de um elemento a ser realocado, na parte de maior soma busca-se o elemento mais próximo da metade do valor da função objetivo para partição atual. O mesmo se faz na parte de menor soma, porém, buscando um elemento cuja diferença com outro da parte de maior soma se

aproxime da metade do valor da função objetivo atual.

5.2.1 Movimento e Vizinhança

O movimento de realocação sempre aproxima a parte de maior soma, encolhendo-a, da parte de menor soma, aumentando-a. O movimento $m_{i,j}$ significa que um elemento sai da parte de índice i e vai para parte de índice j. Assim, gera-se a vizinhança de s, na forma:

$$N(s) = \{s' \ : \ s' \leftarrow s \oplus m_{i,j}, \quad \forall i \in A\} \tag{5.1}$$

A dimensão da vizinhança é $|A|$. Essa operação é a perturbação do (ILS) descrito no Algoritmo 5 na Seção 3.4 do Capítulo 3.

5.2.2 Estratégia de Realocação

A ideia principal do método é testar, dentre todos os elementos da parte de maior soma, aqueles que mais reduzem o valor da função objetivo. Essa busca gera um subproblema, que consiste em encontrar:

$$\max_{a_i \in A_{\max}} \quad \{a_i\} \tag{5.2}$$

$$a_i \leq \frac{1}{2}\left(\sum_{x \in A_{\max}} x - \sum_{x \in A_{\min}} x\right) \tag{5.3}$$

A ideia secundária é testar, dentre todos os elementos a_i, da parte de maior soma, e b_j, da parte de menor soma, aqueles que mais aumentam a função objetivo. Para isso, troca-se os dois elementos cuja diferença seja a mais próxima possível da metade do valor da função objetivo. Essa busca gera um subproblema, que consiste em encontrar:

$$\min_{a_i \in A_{\max}, b_j \in A_{min}} \quad \{a_i - b_j\} \tag{5.4}$$

$$a_i - b_j \leq \frac{1}{2}\left(\sum_{x \in A_{\max}} x - \sum_{x \in A_{\min}} x\right) \tag{5.5}$$

A solução desses problemas aponta os elementos que devem ser trocados ou realocados. Essa operação é a busca local do (ILS) descrito no Algoritmo 5 na Seção 3.4 do Capítulo 3.

5.2.3 Critérios de Parada

Os movimentos continuam enquanto houver possibilidade de melhora da função objetivo pelo movimento. Caso contrário o algoritmo para. Esse critério evita ciclos, reconhecendo trocas já realizadas.

5.2.4 Implementação

O (ILS) deve ser usado no lugar da função $part_2$ dentro do algoritmo 9, já que este opera trocas e realocações somente entre a parte de maior e menor soma. O Algoritmo 10 atualiza suas entradas realocando um ou dois elementos de X_1 para X_2 ou trocando dois elementos entre essas partes. Após reduzir o custo da função objetivo, ele retorna ao Algoritmo 9, que, por sua vez, recalcula as novas partes de maior e menor soma.

Algoritmo 10: Adaptação do (ILS) para a solução do Problema da k-Partição de Números

Entrada: $\{X_1, X_2\}, obj$
Saída: $\{X_1, X_2\}, obj$
início
  $obj1 = obj$;
  para $a \in X_1$ **faça**
    $troco = a - obj1/2$;
    se $troco > 0$ **então**
      $s1 = busca\text{-}remove(a, X_1)$;
      $s2 = busca\text{-}remove(troco, X_2)$;
      $insere(s1, X_2)$;
      $insere(s2, X_1)$;
      $obj = obj - (s1 - s2)$;
    fim
    senão se $troco < 0$ **então**
      $s1 = busca\text{-}remove(a, X_1)$;
      $s2 = busca\text{-}remove(-troco, X_1)$;
      $insere(s1, X_2)$;
      $insere(s2, X_2)$;
      $obj = obj - (s1 + s2)$;
    fim
    senão
      $s1 = busca\text{-}remove(a, X_1)$;
      $insere(s1, X_2)$;
      $obj = obj - s1$;
    fim
  fim
  retorna $\{X_1, X_2\}, obj$
fim

A função $busca-remove(a, S)$ procura o maior elemento menor ou igual a numa lista encadeada. Por isso, os dados de X_1 e X_2 são armazenados numa tabela *hash* afim de amortizar o custo dessa busca. A função *hash* usada na tabela tem as propriedades de um histograma, mantendo elementos de um determinado intervalo dentro da mesma posição.

Seja o conjunto S com todos os seu elementos contidos no intervalo (a, b). Para alocar os elementos de S numa tabela com tamanho m usa-se a função:

$$hash(x) = floor(m.\frac{(x-a)}{(b-a)}) \tag{5.6}$$

Como as instâncias tem distribuição uniforme, espera-se que as listas em cada posição da tabela contenham $\frac{n}{m}$ elementos. Essa propriedade acelera a busca pela variável `troca`, permitindo que o algoritmo tenha uma complexidade amortizada de $O(n)$.

5.3 Algoritmos Exatos

É possível resolver o (MWNPP) com um algoritmo exato para o Problema da Soma de Subconjuntos. A ideia é obter uma parte com a suma mais próxima de $\frac{1}{k}\sum_{x\in S} x$, retira-la de S, decrementar k e fazer isso até $S = \emptyset$.

Algoritmo 11: O algoritmo, denominado $kpart(S, k)$ retorna parte a parte usando, k vezes, um algoritmo exato para o Problema da Soma de Subconjuntos

Entrada: Conjunto S e inteiro k, $subsetsum(S, c)$
Saída: K-Partição do conjunto S, $\{A_1, A_2, ..., A_k\}$
início
  se $S = \emptyset$ **então**
    break;
  fim
  $A_k = subsetsum(S, \frac{1}{k}\sum_{x\in S} x)$;
  $S = S \setminus A_k$;
  $\{A_1, A_2, ..., A_{k-1}\} = kpart(S, k-1)$;
  retorna $\{A_1, A_2, ..., A_k\}$;
fim

Em muitos casos isso é inviável porque usa um algoritmo exponencial k vezes. Caso a instância seja livre de partições perfeitas, com valor de função objetivo igual a 0, o Algoritmo 11 pode falhar.

Um método correto de programação dinâmica é considerar a função $f(i, sum_j)$ a solução ótima do (MWNPP) usando os i primeiros elementos de S com a soma da parte j igual a sum_j para cada $j \in \{1, ..., k-1\}$. Pode-se escrever esta função da seguinte maneira:

$$f(i, sum_j) = \min_{1\leq l\leq k-1}\{f(i-1, sum_{j\neq l} \cup sum_l - a_i), f(i-1, sum_j)\} \tag{5.7}$$

Esta equação de recorrência 5.7 informa que $f(i, sum_j)$ é o menor valor de todos os subproblemas $f(i-1, sum_{j\neq l} \cup sum_l - a_i)$, supondo que o elemento a_i está na parte l para cada $l \in \{1, ..., k-1\}$, ou o subproblema $f(i-1, sum_j)$ que indica que elemento a_i está na parte k, segundo . A solução procurada será $f(n, sum_j)$. Esse método é inviável caso os elementos de S tenham muitos algarismos.

Porém, existem outros métodos exatos que conseguem escapar, em parte, da busca exaustiva reconhecendo buscas irrelevantes. Veja o exemplo de uma árvore de busca que enumera as k-partições de um conjunto na figura 5.1.

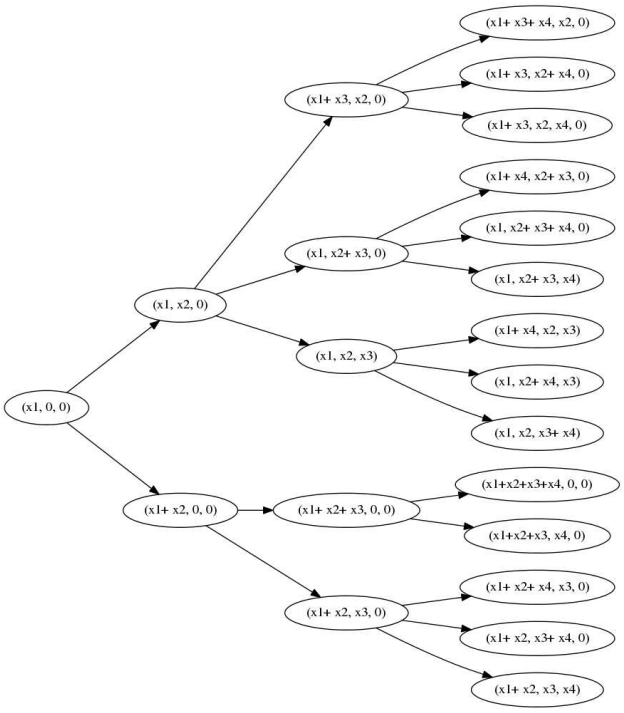

Figura 5.1: Exemplo de árvore de busca ara o (MWNPP): $|S| = 4$ e $k = 3$

Também parece um método impraticável, mas algumas boas ideias ainda podem melhorá-lo.

A maior vantagem das heurísticas simples, com baixa complexidade, é que elas são uma parte importante dos métodos exatos de enumeração implícita. Uma heurística construtiva pode ser aplicada dentro de um algoritmo exato para gerar limites superiores enquanto uma relaxação serve para encontrar limites inferiores. Seja $H(I)$ uma heurística, $R(I)$ uma relaxação e $Opt(I)$ o ótimo global para o problema de minimização I. Conclui-se que

$$R(I) \leq Opt(I) \leq H(I) \tag{5.8}$$

Logo, a combinação inteligente dessas duas coisas pode gerar cortes numa árvore de busca e acelerar muito um método de enumeração implícita.

Se um valor $R(I)$ construído a partir de um nó, x, num determinado nível da árvore de busca for maior que o menor dos $H(I)$, corta-se a ramificação desse nó por contradição com as Inequações 5.8 e o mesmo vale para . Esta ideia evita que a árvore de busca cresça como na Figura 5.1.

Capítulo 6

Objetivos e Metodologia

Este capítulo apresenta objetivos gerais, na Seção 6.1, e específicos, na Seção 6.2 desse projeto de dissertação seguido da metodologia empregada, na Seção 6.3, finalizando com uma breve justificativa, na Seção 6.4.

6.1 Objetivos Gerais

O objetivo geral desse trabalho é o estudo do Problema da k-Partição de Números e suas soluções heurísticas e exatas.

6.2 Objetivos Específicos

Os objetivos específicos do presente trabalho é apresentar uma avaliação da aplicação de algoritmos heurísticos com movimentos "gulosos"de troca e realocação, testar suas aplicabilidades como limitantes de algoritmos exatos de enumeração implícita e verificar se as heurísticas propostas geram limites superiores melhores que as da literatura.

6.3 Metodologia

Para alcançar os objetivos, 6.1 e 6.2, usam-se instâncias geradas aleatoriamente, como nos poucos trabalhos da literatura, os resultados experimentais finalizarão a parte de testes da pesquisa. A avaliação será composta por análises estatísticas, comparações de resultados obtidos na literatura e estratégias abordadas nos trabalhos correlatos, citados no Capítulo 4.

Para alcançar esse objetivo, destaca-se a seguinte sequência metodológica:

(i) Finalizar a pesquisa bibliográfica sobre o (MWNPP) e sobre as técnicas utilizadas para solucioná-lo mesmo em artigos que modificaram o objetivo original do problema.

(ii) Verificar outras formas de resolver o (MWNPP) usando algoritmos desenvolvidos para problemas relacionados.

(iii) Analisar os resultados obtidos com cada método desenvolvido.

(iv) Destacar os resultados de forma comparativa.

6.4 Justificativa

O (MWNPP) apresenta uma série de algoritmos exatos baseados em métodos de enumeração implícita e programação dinâmica. Embora a complexidade desses algoritmos seja exponencial, na prática, eles são bem mais rápidos que o esperado. O mais importante é que um algoritmo que resolva o (MWNPP) em tempo viável pode ser aplicado a outros problemas NP-completos que, geralmente, são questões auxiliares de problemas reais.

Capítulo 7

Cronograma do Trabalho

Neste capítulo apresenta-se o cronograma do trabalho explicitando a metodologia adotada até a conclusão do mesmo. Listam-se as principais tarefas desde a qualificação do projeto até a defesa da dissertação.

7.1 Perspectivas

A tabela 7.1 mostra o cronograma do trabalho até a data prevista para a defesa da dissertação. Reuniões com a orientação estão implícitas.

Tabela 7.1: Cronograma do trabalho para o ano de 2016

Mês	**Tarefas**
Junho	Qualificação e finalização da implementação e testes
Julho	Finalização da dissertação e marcação da data da defesa da dissertação.
Agosto	Defesa da dissertação para obtenção do título de Mestre.

Após a qualificação do projeto de dissertação será discutida a possibilidade de inclusão de outras estratégia para gerar instâncias difíceis para o (MWNPP). Em junho, alguns algoritmos já prontos serão codificados para linguagem C++ e comparados. Em julho, os experimentos computacionais finais com instâncias difíceis geradas aleatoriamente e de outros trabalhos da literatura terminam juntamente com a escrita da dissertação. A finalização da dissertação será apresentada, inserindo as considerações finais da orientação, e a data da defesa será marcada com um mês de antecedência, conforme solicitado pelo programa. Por fim, em setembro, a defesa da dissertação para obtenção do título de Mestre concluirá o presente trabalho.

Capítulo 8

Resultados Preliminares

8.1 Publicações

Até o presente momento, este trabalho tem uma publicação, intitulada, *UM ALGORITMO ITERATIVO PARA O PROBLEMA DA K-PARTIÇÃO DE NÚMEROS*, publicada no evento CILAMCE-2015 (Faria et al., 2015).

8.2 Resultados Computacionais

Implementa-se os algoritmos do trabalho em MATLAB versão 7. Realizam-se testes em um computador com processador Intel Core i3-M330, 2.13 GHz, 3GB de RAM e sistema operacional Windows 32 bits.

Obtem-se resultados a partir de instâncias geradas aleatoriamente. Os conjuntos gerados tem tamanhos entre 100 e 110 e os elementos são inteiros variando de 50 até 350. O valor ótimo do problema - (opt), o tamanho do conjunto - n e o número de partes - k são parâmetros de entrada do gerador de instâncias. Um número grande é escrito como soma de outros k números de modo que a diferença entre o maior e menor seja igual ao parâmetro opt. Em seguida, esses k números são escritos como soma de outros $\frac{n}{k}$ elementos aleatórios formando um vetor de tamanho n cuja partição ótima tem função objetivo menor ou igual a opt. O algoritmo proposto encontrou o ótimo exceto nas instâncias com valores em negrito.

Abaixo, os resultados dos testes devem ser lidos da seguinte maneira: a variável si indica a i-ésima instância. O número $k = \{3, 4, 5\}$ complementa a instância indicando o tamanho da partição. A primeira coluna indica a instância do experimento. A segunda coluna contem resultados alcançados pelo (LPT). A terceira coluna são os resultados finais melhorados com o (ILS) partindo do (LPT).

Instâncias para k=3	Algoritmo 1	Algoritmo 2	*opt*
s1	9	3	3
s2	51	1	1
s3	3	3	3
s4	51	3	3
s5	39	**10**	**4**
s6	49	2	2
s7	57	2	2
s8	44	5	5
s9	3	1	1

Tabela 8.1: Experimentos demonstrando a melhora provocada pelo (ILS) para k=3

Instâncias para k=4	Algoritmo 1	Algoritmo 2	*opt*
s1	41	3	3
s2	47	0	0
s3	48	5	5
s4	50	1	1
s5	4	4	4
s6	50	**9**	**7**
s7	58	0	0
s8	48	5	5
s9	12	3	3

Tabela 8.2: Experimentos demonstrando a melhora provocada pelo ILS para k=4

Instâncias para k=5	Algoritmo 1	Algoritmo 2	*opt*
s1	16	6	6
s2	48	2	2
s3	45	6	6
s4	49	6	6
s5	53	7	7
s6	50	0	0
s7	62	3	3
s8	49	**11**	**6**
s9	51	3	3

Tabela 8.3: Experimentos demonstrando a melhora provocada pelo ILS para k=5

8.3 Análise

Apresentou-se uma proposta de algoritmo heurístico usando (ILS) com movimentos de realocação dos elementos entre as partes para o (MWNPP) na sua versão de otimização. Os resultados obtidos foram testados com instâncias geradas aleatoriamente.

O (ILS) empregado com método guloso consegue melhorar a solução inicial sempre que esta não é ótima. O algoritmo apresentado tem baixa complexidade e atingiu o ótimo global em todas as 10 instâncias testadas mostrando melhora significativa em relação a solução inicial construída.

Como trabalhos futuros tem-se a melhoria do algoritmo apresentado com uma tabela *hash* que melhore a busca pelo movimento de realocação, inserção de um movimentos de permuta entre duas ou mais partes e um quadro comparativo entre os métodos exatos e aproximados para o problema descrito.

Referências Bibliográficas

Ausiello, G.; Crescenzi, P.; Kann, V.; Marchetti-sp,; Gambosi, Giorgio e Spaccamela, Alberto M. (2003). *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*, Capítulo Sequential Algorithms for Partitioning Problems, p. 50–60. Springer-Verlag New York, Inc., 1st edição.

Berretta, Regina e Moscato, Pablo. (1999). The number partitioning problem: An open challenge for evolutionary computation? Corne, David; Dorigo, Marco; Glover, Fred; Dasgupta, Dipankar; Moscato, Pablo; Poli, Riccardo e Price, Kenneth V., editors, *New Ideas in Optimization*, p. 261–278. McGraw-Hill Ltd., UK, Maidenhead, UK, England.

Faria, Alexandre Frias; de Souza, Sérgio Ricardo; Silva, Carlos Alexandre e Filho, Moacir Felizardo França. (2015). Um algoritmo iterativo para o problema da k-partição de números. *Proceedings of the XXXVI Ibero-Latin American Congress on Computational Methods in Engineering*, volume 1, (2015).

Ferreira, Fernando Fagundes. *Análise estatística do problema da partição numérica*. Tese de Doutorado, Universidade de São Paulo, (2001).

Finn, Greg e Horowitz, Ellis. (1979). A linear time approximation algorithm for multiprocessor scheduling. *BIT Numerical Mathematics*, v. 19, n. 3, p. 312–320.

Garey, Michael R. e Johnson, David S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

Gent, Ian P. e Walsh, Toby. (1995). The number partition phase transition. Technical Report RR-95-185, Department of Computer Science, University of Strathclyde, Glasgow, Scotland.

Gent, Ian P. e Walsh, Toby. (1998). Analysis of heuristics for number partitioning. *Computational Intelligence*, v. 14, n. 3, p. 430–451.

Graham, Ronald L. (1966). Bounds for certain multiprocessing anomalies. *THe Bell System Technical Journal*, v. XLV, n. 9, p. 1563–1581.

Graham, Ronald L. (1969). Bounds on multiprocessing timing anomalies. *SIAM journal on Applied Mathematics*, v. 17, n. 2, p. 416–429.

Griffiths, Martin e Mezo, István. (2010). A generalization of stirling numbers of the second kind via a special multiset. *Journal of Integer Sequences*, v. 13, n. 2, p. 3.

Hayes, Brian. (2002). Computing science: The easiest hard problem. *American Scientist*, v. 90, p. 113–117.

Horowitz, Ellis e Sahni, Sartaj. (1974). Computing partitions with applications to the knapsack problem. *Journal of the ACM (JACM)*, v. 21, n. 2, p. 277–292.

Joosten, Sebastiaan J. C. e Zantema, Hans. (2013). Relaxation of 3-partition instances. Cornelissen, Kamiel; Hoeksma, Ruben; Hurink, Johann e Manthey, Bodo, editors, *12th Cologne-Twente Workshop on Graphs and Combinatorial Optimization, Enschede, Netherlands, May 21-23, 2013*, volume WP 13-01 of *CTIT Workshop Proceedings*, p. 133–136, (2013).

Karmarkar, Narendra e Karp, Richard M. (1982). The differencing method of set partition. Report UCB/CSD 81/113, Computer Science Division, University of California, Berkeley, CA.

Karp, Richard M. (1972). Reducibility among combinatorial problems. Miller, Raymond E.; Thatcher, James W. e Bohlinger, Jean D., editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972*, p. 85–103, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA. Springer US.

Kojić, Jelena. (2010). Integer linear programming model for multidimensional two-way number partitioning problem. *Computers & Mathematics with Applications*, v. 60, n. 8, p. 2302–2308.

Korf, Richard E. (1998). A complete anytime algorithm for number partitioning. *Artificial Intelligence*, v. 106, n. 2, p. 181–203.

Korf, Richard E. (2009). Multi-way number partitioning. *IJCAI*, p. 538–543. Citeseer, (2009).

Korf, Richard E; Schreiber, Ethan L e Moffitt, Michael D. (2013). Optimal sequential multi-way number partitioning. *International Symposium on Artificial Intelligence and Mathematics (ISAIM-2014)*, (2013).

Korf, Richard Earl. (2010). Objective functions for multi-way number partitioning. *Third Annual Symposium on Combinatorial Search*, (2010).

Langston, Michael A. (1982). Improved 0/1-interchange scheduling. *BIT Numerical Mathematics*, v. 22, n. 3, p. 282–290.

Lima, Elon Lages. (2004). *Análise real*. IMPA, Rio de Janeiro, Brasil.

Lourenço, Helena R.; Martin, Olivier C. e Stützle, Thomas. (2003). Iterated local search. Glover, Fred e Kochenberger, Gary A., editors, *Handbook of Metaheuristics*, p. 320–353. Springer US, Boston, MA.

Mertens, Stephan. (2006). The easiest hard problem: Number partitioning. Percus, A.G.; Istrate, G. e Moore, C., editors, *Computational Complexity and Statistical Physics*, p. 125–139, New York. Oxford University Press.

Michiels, Wil; Korst, Jan; Aarts, Emile e others,. (2003). Performance ratios for the karmarkar-karp differencing method. *Electronic Notes in Discrete Mathematics*, v. 13, p. 71–75.

Moffitt, Michael D. (2013). Search strategies for optimal multi-way number partitioning. *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, p. 623–629. AAAI Press, (2013).

Pedroso, João Pedro e Kubo, Mikio. (2010). Heuristics and exact methods for number partitioning. *European Journal of Operational Research*, v. 202, n. 1, p. 73–81.

Pop, Petrică C e Matei, Oliviu. (2013). A memetic algorithm approach for solving the multidimensional multi-way number partitioning problem. *Applied Mathematical Modelling*, v. 37, n. 22, p. 9191–9202.

Schreiber, Ethan L. *Optimal Multi-Way Number Partitioning*. PhD thesis, University of California Los Angeles, (2014).

Schreiber, Ethan L e Korf, Richard E. (2014). Cached iterative weakening for optimal multi-way number partitioning. *Proceedings of the Twenty-Eighth Annual Conference on Artificial Intelligence (AAAI-14) Quebec City, Canada*, (2014).

Schroeppel, Richard e Shamir, Adi. (1981). A $T = O(2^{n/2})$, $S = O(2^{n/4})$ algorithm for certain NP-complete problems. *SIAM journal on Computing*, v. 10, n. 3, p. 456–464.

Sloane, NJA. On-Line Encyclopedia of Integer Sequences. https://oeis.org/, (1991).

Stanley, Richard P. (1997). *Enumerative Combinatorics. Vol. 1, vol. 49 of Cambridge Studies in Advanced Mathematics*, volume 1 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge, 2nd edição.