UM ALGORITMO ITERATIVO PARA O PROBLEMA DA K-PARTIÇÃO DE NÚMEROS

Alexandre Frias Faria

Sérgio Ricardo de Souza

Carlos Alexandre Silva

Moacir Felizardo de França Filho

alexandrefrias1@hotmail.com

sergio@dppg.cefetmg.br

carlos.silva@ifmg.edu.br

franca@des.cefetmg.br

Centro Federal de Educação Tecnológica de Minas Gerais

Avenida Amazonas-7675-Nova Gameleira, 30510-000, Minas Gerais, Belo Horizonte, Brasil

Resumo. *Este artigo apresenta um algoritmo para a versão de otimização do Problema da k-Partição de Números (NPPk). 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 método consiste em aplicar a meta-heurística* Iterated Local Search *(ILS) na solução gerada por um algoritmo aproximado. A estratégia é aplicar método guloso em todas as fases do algoritmo, tanto na inicialização quanto na escolha de movimentos do ILS. Usando um algoritmo heurístico para o Problema da Partição de Números (NPP) em 2 subconjuntos, pode-se construir um algoritmo polinomial iterativo para o Problema da Partição de Números em k subconjuntos. Realiza-se um estudo comparativo com outro algoritmo da literatura usando a análise de complexidade e instâncias geradas aleatoriamente com ótimo conhecido.*

Palavras-chave: *Problema de Partição de Números, Problema da Partição de Números em Múltiplas Formas, Otimização Combinatória,* Iterated Local Search

1 INTRODUÇÃO

O NPP (Problema da Partição de Números) consiste na tarefa de decidir se um determinado conjunto de inteiros positivos pode ser particionado em dois subconjuntos, de tal forma que a soma dos números do primeiro subconjunto seja igual à soma do segundo. Durante o decorrer do texto, os subconjuntos pertencentes à partição são denominados de partes, pois os conjuntos têm, necessariamente, todos os seus elementos distintos. O problema é considerado na sua versão de otimização, que consiste em minimizar a diferença das somas resultantes dessas duas partes. Embora o NPP seja um dos 21 problemas de classe NP-completos discutidos em Karp (1972), existem algoritmos exatos, como em Schroeppel and Shamir (1981), Horowitz and Sahni (1974), Korf (1998) e, também, muitos algoritmos aproximados aplicáveis ao problema analisados em Gent and Walsh (1998).

Esse problema já foi chamada de "O mais fácil problema NP-completo", conforme aponta Mertens (2003), mas o problema generalizado é muito mais complicado e ainda não apresenta nenhum algoritmo pseudo-polinomial descrito na literatura levantada durante esse trabalho. Algumas variações mais gerais desse problema se limitam a encontrar uma partição com partes de tamanho fixo como o *3-partition problem* em Joosten and Zantema (2013) ou mudam o objetivo original para minimização da parte de maior soma, como em Moffitt (2013).

2 DESCRIÇÃO DO PROBLEMA

O NPP é generalizado da seguinte maneira: dado um conjunto numérico, o objetivo é encontrar uma partição de tamanho k, de modo que as somas dos elementos de cada parte sejam as mais próximas possíveis umas das outras (Korf; 2009). Isso se resume a obter a parte de maior soma, se aproximando da parte de menor soma tanto quanto for possível (Karmarkar and Karp; 1982). Assim, a partição ficará tão equilibrada quanto a instância do problema permitir.
Exemplo 2.1. *O conjunto* $\{87, 6, 5, 45, 34, 2, 24, 12, 7, 6, 54, 34\}$ *é particionado em duas partes:*

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

ou em três partes:

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

As partes não precisam ter necessariamente o mesmo tamanho.

A formulação matemática do problema generalizado pode ser encontrada em Karmarkar and Karp (1982). A entrada é um conjunto S e um inteiro positivo k e a saída é uma partição de S, dada por $\{A_1, A_2, ..., A_k\}$, que minimize a função:

$$\max_i \left\{ \sum_{x \in A_i} x \right\} - \min_j \left\{ \sum_{x \in A_j} x \right\} \tag{1}$$

Sua versão em modelo de Otimização Linear Inteira é

$$\min \quad t_2 - t_1 \tag{2}$$

$$\text{suj. a} \quad t_1 \leq \sum_{i=1}^{n} a_i x_{ij} \leq t_2, \quad \forall j \in I_k \tag{3}$$

$$\sum_{j=1}^{k} x_{ij} = 1, \quad \forall i \in I_n \tag{4}$$

$$t_1 \leq t_2, \quad t_1, t_2 \in \mathbb{Z}, \quad x_{ij} \in \{0,1\} \tag{5}$$

em que $\{a_i\}_{i=1}^{n}$ são os elementos do conjunto a ser particionado.

A complexidade computacional desse problema, quando resolvido por um algoritmo de força bruta, é dada pelo Número de Stirling $S(n,k)$, que conta a quantidade de formas distintas de repartir um conjunto de tamanho n em k partes, multiplicado pela quantidade de operações necessárias para calcular a função objetivo, dada por $O(n)$. Assim, a complexidade é dada por $O(n.S(n,k))$, sendo:

$$S(n,1) = 1, \quad S(n,n) = 1, \quad S(n,k) = S(n-1,k-1) + k.S(n-1,k) \tag{6}$$

$$S(n,k) = \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^n \tag{7}$$

O exemplo 2.2 mostra esse resultado.

Exemplo 2.2. $A = \{1,2,3,4\}$ *para* $n = 4$ *e* $k = 2$.

Portanto:

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

Ou seja, $S(4,2) = 7$.

3 ALGORITMOS

Assumindo a existência de um algoritmo $O(1)$ que resolve o NPP, dado por $part_2(S)$, sendo S um conjunto, é possível propor um algoritmo inspirado no 0/1-*Interchange* proposto por Finn and Horowitz (1979) que resolve o problema das k-partições.

O algoritmo 0/1-*Interchange* é um algoritmo proposto para o Problema de Escalonamento em Máquinas idênticas. Partindo de uma distribuição inicial das tarefas, o algoritmo ordena os processadores em ordem não decrescente. Seja $d_i = C_1^i - C_m^i$ o instante de finalização do processador mais carregado e do menos carregado na iteração i. Enquanto existir uma tarefa em

$p_j < d_i$ em C_1, está será realocada para C_m. Este algoritmo tem uma complexidade de tempo computacional da ordem $O(n.log(k))$ (Langston; 1982).

Uma possível melhora para o 0/1-*Interchage* é escolher os $\max_j\{p_j\} < d_i$ no momento da troca.

Entrada: Partição do conjunto S com tamanho k
Saída: Partição do conjunto S com tamanho k refinada
begin
  repeat
    Encontre $A_i = argmax_i\{\sum_{x\in A_i} x\}$ e $A_j = argmin_j\{\sum_{x\in A_j} x\}$;
    $X = A_i \cup A_j$;
    $\{A_i, A_j\} = part_2(A_i, A_j)$;
    Substitua as duas novas partes na partição;
  until $\max_i\{\sum_{x\in A_i} x\} - \min_j\{\sum_{x\in A_j} x\}$ *seja constante*;
end

Algoritmo 1: Método de solução Geral para o PPN

Após a segunda linha do método descrito acima, pode-se listar 2 casos: a parte da maior soma fica menor e a parte de menor soma fica maior ou ambas as partes permanecem inalteradas. Enquanto o primeiro caso ocorre, a função objetivo do problema decresce. Portanto, o ótimo ainda não foi encontrado. O método $part_2$ pode ser substituído por um método heurístico como o ILS, no caso desse artigo.

3.1 Inicialização

A construção inicial é feita com um algoritmo guloso, que garante um resultado aproximado menor que $\frac{4}{3} - \frac{1}{3k}$. Proposto por Graham (1969) do ótimo global em relação à função objetivo $max_{1\leq i\leq k}\{\sum_{x\in A_i} x\}$ – um problema de partição mínima um pouco diferente do NPPk. A demonstração de aproximabilidade também se encontra em Ausiello et al. (2003)).
Exemplo 3.1. *O resultado ótimo do problema para:*

$$\{1, 2, 3, 4, 5, 6, 7, 8, 910, 1112, 13, 14, 1516, 1718\}$$

e $k = 2$ *é*

$$\{1718, 910, 14, 7, 6, 3, 2\} \quad e \quad \{1516, 1112, 13, 8, 5, 4, 1\}$$

A parte maior tem soma igual a 2660*. Portanto, a resposta do algoritmo sempre é uma partição em que a maior soma é menor que* $\frac{7}{6}2660 \cong 3546$ *como:*

$$\{1718, 1516\} \quad e \quad \{1112, 13, 8, 5, 4, 1, 910, 14, 7, 6, 3, 2\}$$

mas não:

$$\{1718, 1516, 910\} \quad e \quad \{1112, 13, 8, 5, 4, 1, 14, 7, 6, 3, 2\}$$

A geração da solução inicial consiste em ordenar o conjunto S em ordem não crescente e alocar os números um a um, sempre inserindo o novo elemento na parte de menor soma e, em caso de empate, desempatando arbitrariamente. Este algoritmo tem um custo computacional

$O(n.log(n))$ para ordenar a entrada e um custo $O(n)$ para alocar os elementos nas k. Finalmente, a complexidade do método é $O(n.log(n))$.

Entrada: Conjunto S de números inteiros e um inteiro k
Saída: Partição do conjunto S com tamanho k
begin
  for $j \in \{1, ..., k\}$ **do**
    $P_j = \emptyset$;
    $L_j = 0$;
  end
  Ordene S em ordem decrescente;
  for $a \in S$ **do**
    $k = argmin_j L_j$;
    $P_k = P_k \cup a$;
    $L_k = L_k + a$;
  end
end

Algoritmo 2: Gerador de Solução Inicial

4 REFINAMENTO DA SOLUÇÃO COM ILS

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 da solução da-se 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.

4.1 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.

$$N(s) = \{s' : \quad s \oplus m_{i,j}, \quad \forall i \in A\} \tag{8}$$

O tamanho da vizinhança é $|A|$.

4.2 Estratégia de Troca: Busca Tabu

A ideia central do método é testar entre todos os elementos da vizinhança aqueles que mais reduzem o valor da função objetivo. Essa busca gera um subproblema que consiste em encontrar

$$max_{a_i \in P_{max}}\{a_i\}, \quad a_i \leq \frac{1}{2}\left(\sum_{x \in P_{max}} x - \sum_{x \in P_{min}} x\right) \tag{9}$$

4.3 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 RESULTADOS

O algoritmo foi implementado em MATLAB versão 7. Os testes foram realizados em um computador com processador Intel Core i3-M330, 2.13 GHz, 3GB de RAM e sistema operacional Windows 32 bits.

Os resultados foram obtidos com 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 o menor seja igual ao parâmetro opt. Em seguida, esses k números são escritos como soma de outros $\frac{n}{k}$ números aleatórios formando um vetor de tamanho n cuja partição ótima tem função objetivo igual a opt. O algoritmo 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 algoritmo 1. A terceira coluna são os resultados finais melhorados com o ILS no algoritmo 2.

Tabela 1: Experimentos demonstrando a melhora provocada pelo ILS

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 2: Experimentos demonstrando a melhora provocada pelo ILS

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 3: Experimentos demonstrando a melhora provocada pelo ILS

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

6 CONCLUSÃO

O trabalho apresentou uma proposta de algoritmo heurístico usando ILS com movimentos de realocação de elementos entre as partes para o NPPk 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.

AGRADECIMENTOS

Os autores agradecem o apoio da CAPES, CNPq e FAPEMIG, bem como ao CEFET-MG, para a realização desse trabalho.

REFERÊNCIAS

Ausiello, G., Crescenzi, P., Kann, V., Marchetti-sp, Gambosi, G. and Spaccamela, A. M. (2003). *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*, har/cdr edn, Springer, Berlin.

Finn, G. and Horowitz, E. (1979). A linear time approximation algorithm for multiprocessor scheduling, *BIT Numerical Mathematics* **19**(3): 312–320.
URL: *http://dx.doi.org/10.1007/BF01930985*

Gent, I. P. and Walsh, T. (1998). Analysis of heuristics for number partitioning, *Computational Intelligence* **14**(3): 430–451.

Graham, R. L. (1969). Bounds on multiprocessing timing anomalies, *SIAM journal on Applied Mathematics* **17**(2): 416–429.

Horowitz, E. and Sahni, S. (1974). Computing partitions with applications to the knapsack problem, *Journal of the ACM (JACM)* **21**(2): 277–292.

Joosten, S. J. and Zantema, H. (2013). Relaxation of 3-partition instances., *CTW*, pp. 133–136.

Karmarkar, N. and Karp, R. M. (1982). *The Differencing Method of Set Partition*, 2 edn, University of California, Berbkeley, CA.

Karp, R. M. (1972). *Reducibility among combinatorial problems*, Springer.

Korf, R. E. (1998). A complete anytime algorithm for number partitioning, *Artificial Intelligence* **106**(2): 181–203.

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

Langston, M. (1982). Improved 0/1-interchange scheduling, *BIT Numerical Mathematics* **22**(3): 282–290.
URL: *http://dx.doi.org/10.1007/BF01934441*

Mertens, S. (2003). Number partitioning.

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

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