Estudo de Variantes do Problema do Empacotamento

Candidata: Rachel Vanucchi Saraiva
Orientador: Rafael Crivellari Saliba Schouery

Resumo

O Problema do Empacotamento, em que busca-se a menor quantidade de recipientes necessários para armazenar um conjunto de itens, é muito relevante para o setor industrial por modelar problemas de logística, como armazenamento de produtos e corte de materiais.

Este projeto tem como objetivo o estudo de algoritmos de aproximação para variantes d-dimensionais do problema, e também para a variante Min-Sum, além de também abordar a variante bidimensional do Problema da Mochila.

1 Introdução e Justificativa

O *Problema do Empacotamento* é um problema clássico de otimização discreta NP-difícil onde tem-se uma lista de n itens de tamanhos $a_1, \ldots, a_n$, e uma capacidade C, tal que $0 < a_i < C$ para $i = 1, \ldots, n$. Uma solução para o Problema do Empacotamento é uma partição dos itens, tal que a soma dos tamanhos dos itens em cada parte não ultrapasse C. Chamamos cada parte de um *recipiente*. É possível, sem perda de generalidade, considerar $C = 1$, pois caso contrário basta dividir os tamanhos dos itens pela capacidade original. O Problema do Empacotamento é de grande importância para o setor industrial por modelar problemas de logística, como o armazenamento de produtos e arquivos digitais, a distribuição de tarefas para diversos computadores, e o corte de peças em placas, folhas ou rolos de um material.

Este projeto visa a continuação do estudo do Problema do Empacotamento realizado na primeira iniciação científica da candidata, com foco em variantes do problema, como as variantes d-dimensionais, que consideram itens e recipientes com d dimensões em vez de apenas uma e diversas formas geométricas, e a variante chamada *Min-Sum*, em que, em vez da quantidade p de recipientes usados, procura-se minimizar o valor $\sum_{j=1}^{p} j \cdot |B_j|$, onde B_j são os recipientes da solução, isto é, o custo de empacotar itens nos primeiros recipientes abertos é menor do que nos últimos. Uma aplicação prática do Min-Sum é o escalonamento não-preemptivo de tarefas em um processador quando busca-se minimizar o tempo de execução de todas as tarefas, ou seja, o tempo desde sua chegada no processador (assumindo que todas chegam no instante 0) até o término de sua execução. [2]

Outro problema a ser estudado, relacionado ao Problema do Empacotamento, é o *Problema da Mochila*. No Problema da Mochila são dados n itens que possuem um tamanho s_i e um valor v_i, para $i = 1, \ldots, n$, e uma mochila de capacidade B. O objetivo é achar um subconjunto desses itens cuja soma de seus valores seja a maior possível e a soma de seus tamanhos não ultrapasse B. É possível, sem perda de generalidade, considerar $B = 1$, caso contrário basta dividir os tamanhos dos itens pela capacidade original. Assim como o Problema do Empacotamento, o Problema da Mochila também é aplicado em questões de logística como carregamento de carga, buscando maximizar o aproveitamento de um espaço limitado. Neste projeto será dado foco às variantes d-dimensionais do problema.

Entre os algoritmos a serem estudados nesta iniciação científica estão o esquema de aproximação para Problema do Empacotamento Min-Sum apresentado por L. Epstein, D. S. Johnson e A. Levin em

2018 [2], o esquema de aproximação robusto para empacotamento de cubos apresentado por L. Epstein e A. Levin em 2013 [1], as aproximações para Problema da Mochila com retângulos apresentadas por K. Jansen e G. Zhang [6] e o esquema de aproximação para quadrados apresentado por K. Jansen e R. Solis-Oba [5].Também será estudada a inaproximabilidade de empacotamento de quadrados e retângulos apresentados por L. Epstein e R. v. Stee em 2005 [3]. Este projeto também busca estudar uma variante bidimensional do Problema do Empacotamento Min-Sum.

2 Abordagens de Pesquisa

A seguir, são apresentados conceitos e técnicas de projeto de algoritmos relevantes para esta iniciação científica.

2.1 Algoritmos de Aproximação

Formalmente, dizemos que um algoritmo A é uma α-*aproximação* se A executa em tempo polinomial e, para qualquer instância I de um problema, encontra uma solução de valor $A(I)$ tal que, no caso de um problema de minimização, $A(I) \leq \alpha OPT(I)$, onde $OPT(I)$ é o valor de uma solução ótima para a instância e $\alpha \geq 1$. Por exemplo, uma 2-aproximação devolve uma solução com valor no máximo o dobro do valor de uma solução ótima. O fator α é chamado *razão de aproximação do algoritmo*. No caso de um problema de maximização, $A(I) \geq \alpha OPT(I)$ e $\alpha \leq 1$.

Alguns algoritmos de aproximação também possuem uma garantia *a fortiori*, isto é, uma garantia calculada a partir da solução gerada e dos dados de entrada do problema. Por depender dos dados de uma instância em particular, essa razão *a fortiori* não pode ser usada como garantia da qualidade geral do algoritmo, porém é geralmente mais precisa quanto à qualidade da solução obtida, e pode indicar que esta é muito melhor (razão mais próxima de 1) do que previsto.

Ainda, para alguns problemas é possível projetar um *Esquema de Aproximação de Tempo Polinomial* (PTAS), uma família de algoritmos $\{A_\varepsilon\}$, onde para cada $\varepsilon > 0$, o algoritmo A_ε é uma $(1+\varepsilon)$-aproximação (para problemas de minimização) ou uma $(1-\varepsilon)$-aproximação (para problemas de maximização).

O estudo de algoritmos de aproximação também pode ser útil como medida do quão difícil é um problema, pois por vezes pode-se provar que, além do problema ser NP-difícil, também não pode existir α-aproximação para esse problema para um α pequeno demais, a não ser que $P = NP$. Essa dificuldade em aproximar é chamada de *inaproximabilidade* do problema.

Mesmo quando provada a inaproximabilidade de um problema, é possível encontrar aproximações melhores se a definição de razão de aproximação for relaxada, passando a permitir pequenos termos aditivos, o que é chamado de *razão de aproximação assintótica*. Um algoritmo A com razão de aproximação assintótica ρ produz solução de valor $A(I) \leq \rho \cdot OPT + c$, onde c é uma constante. Isso motiva a definição de um *Esquema de Aproximação de Tempo Polinomial Assintótico* (APTAS), uma família de algoritmos $\{A_\varepsilon\}$ com uma constante c, onde para cada $\varepsilon > 0$, existe A_ε que retorna uma solução de valor no máximo $(1 + \varepsilon) \cdot OPT + c$ para problemas de minimização. Para instâncias do problema onde é esperado que o valor da solução ótima seja muito maior que a constante c, ou seja,

quando são necessários muitos recipientes, essa constante é desprezível e o algoritmo é praticamente uma ρ-aproximação.

Outra definição usada no estudo e projeto de algoritmos de aproximação é a de um *Esquema de Aproximação de Tempo Plenamente Polinomial* (FPTAS), uma família de algoritmos com razão de aproximação $1+\varepsilon$ e tempo polinomial tanto no tamanho de sua entrada quanto em $1/\varepsilon$.

2.2 Algoritmos Online

Um problema é chamado de *problema offline* se todos os dados de entrada da instância são recebidos ao mesmo tempo. No caso do Problema do Empacotamento, ele é um problema *offline* quando já se sabe todos os itens a serem empacotados e seus tamanhos. Porém, em algumas situações os dados são recebidos com o tempo, não todos de uma vez, e é necessário tomar decisões assim que chegam sem ter conhecimento do resto da instância que chegará mais tarde. Esse tipo de problema é chamado *problema online*, e assim um algoritmo que devolve uma solução para esse tipo de problema é um *algoritmo online*. O Problema do Empacotamento é *online* se os itens são recebidos um a um e precisam ser colocados nos recipientes assim que chegam, sem ter conhecimento dos próximos itens e sem poder mais tarde retirar um item do recipiente escolhido, ou seja, a decisão feita no recebimento do item é final.

Por exigir que as decisões para compor a solução sejam tomadas sem que se tenha todos os dados do problema, os problemas *online* são geralmente mais difíceis de se achar solução ótima ou até mesmo de se aproximar. A eficiência de um algoritmo *online* é analisada de forma semelhante à análise de algoritmos de aproximação, ou seja, compara-se o valor da solução gerada com o valor da solução ótima do problema *offline* (solução que leva em consideração todos os dados da instância). Formalmente, um algoritmo *online* A é chamado α-*competitivo* se, para qualquer instância I de um problema online de minimização, retorna uma solução de valor $A(I)$ tal que $A(I) \leq \alpha \cdot OPT$, onde OPT é o valor da solução ótima do problema *offline*. Assim como no caso da aproximação, essa é uma análise de pior caso.

2.3 Funções de Ponderação

Funções de ponderação são usadas para a análise de certos algoritmos de aproximação ou online.

Definição 1. *Para um algoritmo $A(I)$, uma função de ponderação $W_A(a)$ é uma função que associa cada item a a um valor real seguindo duas propriedades:*

$$\sum_{a \in I} W_A(a) \geq A(I) - c \tag{1}$$

$$\sum_{a \in B} W_A(a) \leq \alpha \tag{2}$$

onde I é uma instância do Problema do Empacotamento, B é qualquer subconjunto de itens de I cuja soma dos tamanhos não ultrapassa 1 (ou seja, qualquer subconjunto que possa ser empacotado em um único recipiente), e c e α são constantes não negativas.

Para facilitar a notação, podemos definir que, para qualquer conjunto S de itens, $W_A(S) = \sum_{a \in S} W_A(a)$. Portanto $\sum_{a \in I} W_A(a) = W_A(I)$ e $\sum_{a \in B} W_A(a) = W_A(B)$.

Lema 1. *Se $W_A(a)$ é uma função que respeita as propriedades (1) e (2), então o valor de uma solução dada pelo algoritmo é $A(I) \leq \alpha OPT(I) + c$.*

Demonstração. Com a propriedade (2) temos que, se $\{B_1^*, B_2^*, \ldots, B_{OPT(I)}^*\}$ são os recipientes de uma solução ótima, então $W_A(I) = \sum_{j=1}^{OPT(I)} W_A(B_j^*) \leq \alpha OPT(I)$. Assim, com a propriedade (1), temos que $A(I) \leq \alpha OPT(I) + c$, e assim encontramos um limitante superior para o algoritmo. □

2.4 Programação Linear

Programação linear é um conceito muito utilizado na análise e desenvolvimento de algoritmos de aproximação. A forma padrão de um programa linear é:

$$\begin{aligned} \text{minimizar} \quad & \sum_{j=1}^{n} c_j x_j \\ \text{sujeito a} \quad & \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \qquad i = 1, \ldots, m \\ & x_j \geq 0, \qquad j = 1, \ldots, n \end{aligned}$$

onde a_{ij}, b_i e c_j são constantes, x_j são variáveis que representam alguma decisão a ser tomada para compor uma solução, $\sum_{j=1}^{n} a_{ij} x_j \geq b_i$ é uma desigualdade linear chamada de *restrição*, que restringe os valores que as variáveis podem receber, e $\sum_{j=1}^{n} c_j x_j$ é a *função objetivo*, cujo valor procura-se minimizar.

Um conjunto de valores para as variáveis que respeita todas as restrições impostas é uma *solução viável* para o programa linear, e uma solução viável que minimiza a função objetivo é uma *solução ótima* para o programa, no caso de um problema de minimização. Programas lineares que não estão em forma padrão também podem ter restrições na forma $\sum_{j=1}^{n} a_{ij} x_j \leq b_i$ ou igualdades, variáveis apenas negativas ou sem qualquer restrição de seus valores, ou serem problemas de maximização.

Um programa inteiro, além das mesmas características descritas acima, possui a restrição extra em que os valores que as variáveis podem assumir são sempre inteiros. Removendo-se apenas essa restrição, o programa inteiro é relaxado para um programa linear, para o qual conhece-se algoritmo polinomial, ao contrário do programa inteiro, para o qual não se conhece algoritmo polinomial e a existência de tal algoritmo implica que $P = NP$.

Assim, se um problema de minimização NP-difícil pode ser representado como um programa inteiro, é possível encontrar um limite inferior da solução ótima a partir da relaxação do problema para sua forma linear. Isso porque toda solução viável para o programa inteiro também é uma solução viável do programa linear e portanto o valor de uma solução ótima do programa linear será menor ou igual ao valor de uma solução do programa inteiro. O análogo acontece para problemas de maximização.

A partir de qualquer programa linear de minimização $LP(P)$ (chamado de programa *primal*), é

possível criar outro programa linear $LP(D)$, chamado de *dual* do programa primal, cuja forma padrão é:

$$\begin{aligned} \text{maximizar} \quad & \sum_{i=1}^{m} b_i y_i \\ \text{sujeito a} \quad & \sum_{i=1}^{m} a_{ij} y_i \geq c_j, \qquad j = 1, \ldots, n \\ & y_i \geq 0, \qquad i = 1, \ldots, m \end{aligned}$$

Cada variável y_i do programa dual está relacionada a uma restrição do primal pela função objetivo do dual, e cada uma de suas restrições está relacionada a uma variável do programa primal pelas constantes c_j da função objetivo do primal.

Existem dois teoremas importantes quanto à dualidade do programa linear. O *Teorema da Dualidade Fraca* diz que, caso exista solução ótima para o programa dual, o valor dessa solução é um limitante inferior para o valor de uma solução ótima do programa primal. O *Teorema da Dualidade Forte* diz que, caso existam soluções ótimas para ambos os programas, primal e dual, os valores dessas soluções são iguais [4].

2.5 Técnicas de Projeto de Algoritmos de Aproximação

A seguir são apresentadas técnicas de projeto de algoritmo relevantes para esta iniciação científica.

Arredondamento determinístico. Ao representar o problema como um programa inteiro, algumas vezes é possível derivar um algoritmo de aproximação a partir da solução do programa linear obtido da relaxação desse programa inteiro, convertendo a solução ótima obtida para o programa linear em uma solução inteira viável, arredondando as variáveis de decisão por algum critério de forma a respeitar as restrições do programa inteiro. Esse processo é chamado de *arredondamento determinístico* da solução fracionária. Após isso, é necessário achar uma relação entre o valor da solução inteira criada e o valor da solução linear original. Como o valor da solução linear é um limite inferior do valor da solução ótima do problema, a partir dessa relação é possível calcular a razão de aproximação do algoritmo criado.

As vezes é possível fazer projeto semelhante utilizando o dual do programa linear, convertendo sua solução em uma solução viável para o problema original, e aproveitando-se do Teorema da Dualidade Fraca para saber que o valor da solução do dual é limite inferior do programa primal e portanto também limite inferior da solução ótima do problema.

Arredondamento aleatório. No *arredondamento aleatório* da solução linear, em vez de adaptar a solução linear para uma solução inteira a partir de um critério fixo como no caso determinístico, arredonda-se as variáveis de decisão de forma aleatória, interpretando os valores fracionários da solução linear como probabilidades das variáveis inteiras assumirem certo valor.

Por ser aleatório, em alguns casos não há a garantia de que a solução gerada seja uma solução viável para o programa inteiro. Nesses casos objetiva-se encontrar uma garantia de que o algoritmo tenha alta probabilidade de gerar uma solução viável. Por exemplo, podemos considerar algoritmos tais que a probabilidade de gerar uma solução inviável é no máximo n^{-c}, onde n é o tamanho da entrada e c uma constante arbitrária positiva.

Algoritmos aleatórios podem às vezes ser desaleatorizados, isto é, é possível derivar deles um algoritmo determinístico com a mesma razão de aproximação. Porém, esse algoritmo determinístico pode ser mais difícil de descrever, implementar ou analisar.

Algoritmo primal-dual. Diferente das outras técnicas apresentadas acima, um algoritmo primal-dual não resolve nem o programa linear nem seu dual, e sim constrói uma solução dual viável, e a partir desta infere uma solução para o primal. Caso essa solução inferida seja inviável, a solução dual é modificada até que seja possível inferir dela uma solução primal viável.

Algoritmos gulosos. Um algoritmo guloso é um algoritmo que toma uma sequência de decisões, buscando sempre otimizar a decisão em particular no momento, sem garantia de que isso levará ao melhor resultado ao fim. Algoritmos gulosos são muito usados porque costumam ser fáceis de implementar, e em alguns casos é possível provar que o algoritmo é de fato uma α-aproximação.

Programação dinâmica. Programação dinâmica é uma técnica muito útil no desenvolvimento de algoritmos, onde a solução ótima para um problema é construída a partir das soluções ótimas de subproblemas com a mesma estrutura do problema original. Alguns problemas NP-difíceis até podem ser resolvidos por programação dinâmica em tempo polinomial no tamanho da entrada, se os dados de entrada forem representados em sistema unário em vez de binário, já que esse mudança de representação muda o tamanho da entrada em si. Algoritmos assim são chamados *pseudopolinomiais*. Algoritmos de aproximação podem ser desenvolvidos a partir de programação dinâmica, em geral baseando-se nos algoritmos pseudopolinomiais e arredondando os dados de entrada de alguma forma.

3 Objetivos

O objetivo do projeto é o estudo de variantes do Problema do Empacotamento. As provas estudadas serão reunidas e organizadas de forma clara e didática em um relatório técnico ao fim do projeto.

Além disso, como iniciação científica o projeto irá complementar a formação da candidata na área de Ciência da Computação, aprofundando seu conhecimento de problemas de otimização, e técnicas de projeto e análise de algoritmos.

4 Plano de Trabalho e Cronograma

Os dois primeiros meses do projeto serão dedicados ao estudo da inaproximabilidade do empacotamento de retângulos e quadrados. Nos próximos dois meses serão estudados algoritmos para o empacotamento de cubos.

Posteriormente, quatro meses serão dedicados ao estudo da variantes Min-Sum unidimensional e bidimensional, que necessita de mais tempo por ser um tópico com menos informação disponível. Os últimos meses do projeto serão dedicados ao Problema da Mochila Bidimensional.

Durante todos os meses também estará sendo realizada a escrita e revisão do relatório técnico em paralelo aos estudos.

Atividades	Meses											
	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º
Empacotamento 2D	•	•										
Empacotamento de cubos			•	•								
Min-Sum					•	•	•	•				
Problema da Mochila 2D									•	•	•	•
Escrita de Relatório	•	•	•	•	•	•	•	•	•	•	•	•

5 Materiais e Métodos

Durante o projeto, a candidata estudará artigos importantes quanto ao Problema do Empacotamento, sendo o acesso a esses artigos fornecido gratuitamente pela Unicamp, além de livros sobre otimização discreta e outras áreas relevantes, como programação linear, disponibilizados pela biblioteca do Instituto de Matemática, Estatística e Computação Científica.

Além disso, serão realizadas reuniões quinzenais entre a candidata e o orientador para averiguar o andamento do projeto e discutir os conteúdos estudados.

Referências

[1] L. Epstein and A. Levin. Robust approximation schemes for cube packing. *SIAM Journal on Optimization*, 23(2):1310–1343, 2013.

[2] Leah Epstein, David S. Johnson, and Asaf Levin. Min-sum bin packing. *Journal of Combinatorial Optimization*, 36(2):508–531, Aug 2018.

[3] Leah Epstein and Rob van Stee. Online square and cube packing. *Acta Informatica*, 41(9):595–606, Oct 2005.

[4] D. Gale. *The theory of linear economic models*. McGraw-Hill New York, 1960.

[5] Klaus Jansen and Roberto Solis-Oba. A polynomial time approximation scheme for the square packing problem. In Andrea Lodi, Alessandro Panconesi, and Giovanni Rinaldi, editors, *Integer Programming and Combinatorial Optimization*, pages 184–198, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[6] Klaus Jansen and Guochuan Zhang. On rectangle packing: Maximizing benefits. volume 15, pages 204–213, 01 2004.