Algoritmos de Aproximação para o 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 esse problema, tanto para sua versão unidimensional quanto para suas variantes bidimensionais e tridimensionais.

Outro problema a ser abordado é o Problema da Mochila, em que busca-se um conjunto de itens de valor total máximo cujo tamanho não ultrapasse certo limite, também relacionado à organização e logística de itens, carregamento de carga e aproveitamento de espaços.

Como iniciação científica, esse projeto visa também a introdução da candidata na área de pesquisa científica e a complementação de sua formação na área de Ciência da Computação, bem como a produção de um texto com os resultados estudados que sirva de base para outros pesquisadores da área.

1 Introdução e Justificativa

A área de otimização discreta estuda formas de tomar decisões que maximizem ou minimizem um certo valor desejado como, por exemplo, encontrar o número mínimo de recipientes necessários para guardar um conjunto de itens. Tal problema é conhecido como *Problema do Empacotamento*, e é o foco deste projeto.

Formalmente, uma instância do Problema do Empacotamento é definida por n itens de tamanhos $a_1, \ldots, a_n$, e recipientes de capacidade C, tal que $0 < a_i < C$ para $i = 1, \ldots, n$. Cada recipiente pode empacotar um subconjunto de itens cuja soma de seus tamanhos não ultrapasse C. Supõe-se que há uma quantidade infinita de recipientes disponível, e procura-se minimizar a quantidade de recipientes necessários para empacotar todos os itens.

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, e a distribuição de tarefas para diversos computadores. Soluções boas para problemas de empacotamento minimizam os gastos com essas tarefas e assim podem reduzir o custo de produtos e serviços, tornando-os mais competitivos e acessíveis. Muitas vezes esses problemas também precisam ser resolvidos com rapidez e assim justifica-se a busca por algoritmos de tempo polinomial.

O Problema do Empacotamento também pode ser aplicado em situações relacionadas ao corte de materiais, onde C representa o tamanho das placas, folhas ou rolos de material, e os itens são as peças a serem cortadas, e assim o posicionamento das peças pode ser visto como um empacotamento das peças no material. Minimizando-se a quantidade de placas de material necessárias para obter as peças desejadas, reduz-se o custo da produção das mesmas. Variantes do problema conhecidas como Problema do Empacotamento Bidimensional e Tridimensional consideram peças e materiais

com duas ou três dimensões e diferentes formas geométricas como, por exemplo, o empacotamento de caixas em containers. Nessas variantes, não basta decidir apenas quais itens serão colocados em cada recipiente, é necessário também considerar suas posições e rotações, e garantir que seus interiores não se sobreponham.

O Problema do Empacotamento é NP-difícil e portanto não pode ser resolvido eficientemente, isto é, em tempo polinomial no tamanho de sua entrada, a não ser que $P = NP$. Assim, são necessárias abordagens especiais para esse problema. A abordagem considerada neste projeto é o desenvolvimento de algoritmos de aproximação, que encontram em tempo polinomial uma solução garantidamente próxima de uma solução ótima.

Entre os algoritmos a serem estudados nesse projeto estão algoritmos clássicos como *First Fit Decreasing* (FFD) e *Best Fit Decreasing* (BFD), já analisados desde a década de 1970 por M. R. Garey, R. L. Graham e J. D. Ullman [4], onde os itens são primeiramente colocados em ordem não-crescente e então colocados se couberem no recipiente aberto a mais tempo (*First Fit*) ou no recipiente já aberto mais cheio (*Best Fit*), abrindo novos recipientes apenas quando um item não cabe em nenhum já aberto. Também serão estudadas as versões online desses algoritmos, chamadas apenas de *First Fit* e *Best Fit*, que utilizam os mesmos critérios para a abertura de recipientes e colocação dos itens, mas sem ordená-los no início; e as versões expandidas para duas dimensões, *First Fit Decreasing Height* (FFDH) e *Best Fit Decreasing Height* (BFDH), apresentadas em 1980 por E.G. Coffman Jr., M.R. Garey, D.S. Johnson e R.E. Tarjan [7].

Também serão estudados os algoritmos apresentados por W. Fernandez de la Vega e G. S. Lueker [2] em 1981, N. Karmarkar e R. M. Karp em 1982 [8], bem como outros algoritmos relacionados ao Problema do Empacotamento e suas variantes.

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, $i = 1, \ldots, n$, 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. Há também uma variante do problema com m mochilas em vez de apenas uma. 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.

Ao fim do projeto será redigido um relatório com os algoritmos estudados e as provas de suas razões de aproximação. Além disso, o projeto também visa complementar a formação da candidata, que atualmente cursa o terceiro ano do curso de Ciência da Computação na Universidade Estadual de Campinas (UNICAMP). Como sua primeira iniciação científica, o projeto irá introduzi-la ao trabalho de pesquisa.

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, devolve uma solução de valor $A(I)$ tal que, no caso de um problema de minimização, $A(I) \leq \alpha \cdot OPT$, onde OPT é o valor de uma solução ótima para a instância e $\alpha > 1$. 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, $A(I) \geq \alpha \cdot OPT$ e, portanto, $\alpha < 1$.

É importante ressaltar que, por considerarem todas as instâncias do problema, as provas das razões de aproximação são feitas a partir da análise de pior caso, e portanto o algoritmo pode ter uma performance melhor para diversos casos que ocorrem na prática. Por exemplo, o algoritmo *First Fit Decreasing* possui razão de aproximação assintótica 11/9, portanto no pior caso suas soluções podem ter valor cerca de 22% maior que o ótimo, porém em uma simulação feita com 128000 itens de tamanhos uniformemente distribuidos entre 0 e 1, o desempenho médio do algoritmo foi muito melhor, com os valores de suas soluções em média menos que 1% acima dos valores ótimos [10, 9].

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, e se aplica, por exemplo, ao Problema do Empacotamento, como pode ser visto no seguinte teorema:

Teorema 1. *Não existe uma α-aproximação para o Problema do Empacotamento com $\alpha < 3/2$, a não ser que $P = NP$.*

Demonstração. Nossa redução será a partir do *Problema da Partição*, um problema de decisão NP-difícil [5] onde, dados n números inteiros $b_1, \ldots, b_n$ cuja soma $B = \sum_{i=1}^{n} b_i$ é par, deve-se decidir se é possível particionar tais números em dois subconjuntos S e T tal que as somas destes subconjuntos sejam iguais, ou seja, $\sum_{b \in S} b = \sum_{b \in T} b$. A partir de uma instância I do Problema da Partição pode-se definir uma instância I' do Problema de Empacotamento associada a ele onde $C = B/2$ e $a_i = b_i$. Vê-se que a resposta do Problema da Partição para I é "sim" se e somente se I' pode ser empacotada em exatamente 2 recipientes, ou seja, se $OPT = 2$, onde OPT é o valor de uma solução ótima do Problema do Empacotamento para I'.

Se esse empacotamento fosse realizado por um algoritmo de aproximação A cuja razão de aproximação fosse menor que 3/2, então para cada instância I para a qual o problema da partição deve res-

ponder "sim" $A(I') < \frac{3}{2} \cdot OPT = 3$. Como os valores da solução do empacotamento são inteiros (por serem uma quantidade de recipientes), $A(I') \leq 2 = OPT$. Por outro lado, como $OPT \leq A(I') < \frac{3}{2} \cdot OPT$, se o algoritmo A encontra esse empacotamento em dois recipientes, então $OPT = 2$. Portanto esse algoritmo decide o Problema da Partição em tempo polinomial, o que só pode ocorrer se $P = NP$, já que o Problema da Partição é NP-difícil. □

Apesar desse limitante, 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. Assim, é possível achar algoritmos para o Problema do Empacotamento que possuam c pequeno e razão assintótica ρ menor que 3/2.

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 $1/\varepsilon$. O Problema da Mochila foi um dos primeiros problemas NP-difíceis para o qual foi-se encontrado FPTAS [6, 1], o que demonstra sua importância para essa área de estudo.

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 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 Du-*

alidade 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 [3].

2.4 Técnicas de Projeto de Algoritmos de Aproximação

A seguir são apresentadas técnicas de projeto de algoritmo relevantes para esta iniciação científica, com foco em sua aplicação para o projeto de algoritmos de aproximação.

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 do Problema do Empacotamento e problemas relacionados, com foco em algoritmos de aproximação para esses problemas, desde algoritmos clássicos da década de 70 até resultados mais recentes relacionados a esse problema. As provas dos algoritmos estudados 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 introduzir a candidata a técnicas e métodos de pesquisa, além de complementar sua formação na área de Ciência da Computação, aprofundando seu conhecimento de problemas de otimização, algoritmos de aproximação, e técnicas de projeto e análise de algoritmos.

4 Plano de Trabalho e Cronograma

Os dois primeiros meses do projeto serão dedicados ao estudo de algoritmos clássicos para o Problema do Empacotamento, como *First Fit*, *Best Fit*, *Any Fit* e *Next Fit*, além de suas variantes *offline*. Depois, nos próximos três meses serão estudados APTAS conhecidos para o problema, alguns dos quais requerem conhecimento mais profundo de programação linear e portanto necessitam de mais tempo.

Posteriormente, durante três meses serão estudadas variantes do Problema do Empacotamento, como suas versões bidimensionais e tridimensionais. Os dois meses seguintes serão dedicados ao estudo de algoritmos para o Problema da Mochila.

Os últimos dois meses do projeto serão dedicados à busca e ao estudo de artigos e resultados mais recentes sobre o Problema do Empacotamento.

Atividades	Meses											
	1º	2º	3º	4º	5º	6º	7º	8º	9º	10º	11º	12º
Algoritmos clássicos	•	•										
APTAS			•	•	•							
Variantes do problema						•	•	•				
Problema da Mochila									•	•		
Pesquisa de resultados recentes											•	•
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 algoritmos de aproximação 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] Timothy M. Chan. Approximation Schemes for 0-1 Knapsack. In Raimund Seidel, editor, *1st Symposium on Simplicity in Algorithms (SOSA 2018)*, volume 61 of *OpenAccess Series in Informatics (OASIcs)*, pages 5:1–5:12, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[2] W. Fernandez de la Vega and G. S. Lueker. Bin packing can be solved within $1 + \varepsilon$ in linear time. *Combinatorica*, 1(4):349–355, Dec 1981.

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

[4] M. R. Garey, R. L. Graham, and J. D. Ullman. Worst-case analysis of memory allocation algorithms. In *Proc. 4th ACM Symposium on Theory of Computing (STOC)*, pages 143–150, 1972.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[6] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *J. ACM*, 22(4):463–468, October 1975.

[7] E. G. Coffman Jr., M. R. Garey, D. S. Johnson, and R. E. Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9(4):808–826, 1980.

[8] N. Karmarkar and R. M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 312–320, Nov 1982.

[9] C. C. MecGeoch. *Experimental Analysis of Algorithms*. PhD thesis, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1987.

[10] R. E. Neapolitan. *Foundations of algorithms*. Jones & Bartlett Learning, fifth edition, 2015.