Universidade Federal do Rio de Janeiro
Otimização Combinatória - COS890
Professor: Abilio Lucena

Leilões Combinatórios - uma aplicação do problema de empacotamento de conjuntos através de métodos exatos

Amanda Camacho Novaes de Oliveira

Diego Amaro Ferraz da Costa

Diego Athayde Monteiro

Resumo

Neste trabalho, descrevemos o problema dos leilões combinatórios como uma aplicação do Problema de Empacotamento de Conjuntos, primeiramente proposto por Karp [1]. Aplicamos três algoritmos exatos para a resolução do mesmo: **Relaxação Lagrangeana**, **Branch-and-Bound** e **Branch-and-Cut**. Os mesmos foram implementados através do solver *Gurobi*, utilizando a linguagem de programação *Julia* e biblioteca *JuMP*. Utilizamos deferentes instâncias para comparar o desempenho dos algoritmos, obtendo ...

Colocar algum resultado de interesse

1 Introdução

A otimização combinatória é a área que estuda problemas de otimização aplicados a conjuntos discretos. Os problemas de otimização são aqueles onde se procura determinar valores extremos de um conjunto de restrições, com respeito a uma função, esta chamada de função objetivo. As variáveis envolvidas nos problemas de otimização são denominadas variáveis de decisão. Pode-se classificar os problemas de otimização combinatória de acordo com os tipos de variáveis que os compõem: variáveis lineares ou não lineares; variáveis binárias, inteiras ou reais.

Neste trabalho, definimos e aplicamos técnicas de soluções exatas para um conhecido problema, o Problema de Empacotamento de Conjuntos (PEC). Ele consiste em encontrar uma combinação de subconjuntos de elementos de forma que se obtenha peso máximo e que nenhum elemento seja selecionado por mais de um conjunto. Pode ser descrito como um problema de programação inteira onde as variáveis são lineares e podem assumir valores binários 0-1.

Este problema possui muita aplicação prática e uma das mais conhecidas é a de Leilões Combinatórias que é abordagem que fazemos neste trabalho. Um leilão combinatório consiste em distribuir um número m de produtos através de n lances. Se um lance é escolhido, um determinado lucro é obtido. O objetivo em um leilão combinatório é maximizar o lucro total respeitando a condição de que cada produto só pode estar contido em no máximo um lance. Utilizando esta aplicação do PEC, utilizamos três métodos exatos para resolver o problema: uma **Relaxação Lagrangeana**, um algoritmo **Branch-and-Bound** e um segundo algoritmo **Branch-and-Bound** com algoritmos de planos de corte associados ao mesmo, denominado **Branch-and-Cut**.

Esse texto está organizado da seguinte forma: na seção 2, detalhamos o Problema de Empacotamento de Conjuntos, e nas seções 3, 4 e 5 descrevemos um pouco cada método aplicado. Nas seções 6 e 7 exibimos os resultados obtidos e a comparação entre os métodos e, por fim, na seção 8 nossas conclusões e ideias para trabalhos futuros.

2 O Problema de Empacotamento de Conjuntos

O problema de empacotamento de conjuntos (PEC) (do inglês, *Set-Packing Problem*) foi um dos 21 problemas enunciados por Karp em [1]. Neste mesmo trabalho, Karp mostrou que se trata de um problema NP-completo através de uma redução feita do problema da clique máxima de um grafo. O PEC também possui forte relação com o problema da cobertura de conjuntos, sendo eles duais entre si.

Dual-forte ou dual-fraco?

O PEC pode ser descrito pelo ponto de vista da decisão, que consiste em determinar, dado um inteiro k, se existe um conjunto com k subconjuntos do conjunto de subconjuntos viáveis I_j. Porém neste trabalho abordamos a versão de otimização, cuja descrição formal é dada da seguinte forma:

Colocar referência???

- $I = \{1, ..., m\}$: conjunto finito de elementos
- $J = \{1, ..., n\}$: conjunto finito de índices
- $\{I_j \subseteq : j \in J\}$: conjuntos dos subconjuntos viáveis de I
- Existe um peso c_j à cada I_j, $j \in J$ $\{I_j \subseteq : j \in J\}$: conjuntos dos subconjuntos viáveis de I
- $\{I_j : j \in J_i\}$: subconjuntos dos subconjuntos viáveis de I que contém $i \in I$
- Variáveis binárias $x_j \in \{0,1\}$ que expressam a decisão de escolher ou não I_j, $j \in J$

Dadas estas definições, sua formulação como um problema de programação inteira está dada pela equação 1.

$$\begin{aligned} \max \quad & \sum_{j \in J} c_j x_j \\ \text{sujeito a} \quad & \sum_{j \in J_i} x_j \leq 1, \qquad i \in I \\ & x_j \in \{0,1\}, \quad j \in J \end{aligned} \tag{1}$$

Este problema possui uma ampla aplicação prática. Como um exemplo destas, temos o escalonamento de tripulações de companhias aéreas para aviões. Cada avião da frota precisa ter uma tripulação designada a ele, composta por um piloto, copiloto e navegador [2]. Como mencionado anteriormente, a abordagem que fazemos aqui ao PROBLEMA DE EMPACOTAMENTO DE CONJUNTOS é feita no contexto dos leilões combinatórios, que é objeto de estudo de muitos estudiosos em diversos trabalhos ([3], [4], [5]). Nesta aplicação os conjuntos I e J podem ser enxergados como um conjunto de m produtos e um conjunto de n lances, respectivamente. Nosso objetivo então é maximizar o lucro total obtido sem violar a restrição de que um produto só pode ser escolhido por, no máximo, um lance. E este lucro é um valor associado a seleção de cada lance.

Para a solução desse problema implementamos uma formulação alternativa, dada pela equação 2, onde a_{ij} vale 1 se o lance j seleciona o produto i, ou 0 caso contrário.

$$\begin{aligned} \max \quad & \sum_{j=0}^{n} c_j x_j \\ \text{sujeito a} \quad & \sum_{j=0}^{n} a_{ij} x_j \leq 1, \quad i = 1, \cdots, m \\ & x_j \in \{0,1\}, \quad j = 1, \cdots, n \end{aligned} \tag{2}$$

Esta formulação foi utilizada por entendermos que a mesma é mais intuitiva. Entretanto, foi observado ao longo da implementação que a formulação dada pela equação 1 é mais eficiente para instâncias muito grandes, uma vez que na maioria dos casos as matrizes A induzidas pela seleção de produtos por lances são esparsas.

Está explicado bem?

3 Relaxação Lagrangeana

A relaxação lagrangeana consiste em um método de decomposição das restrições de um determinado problema em dois grupos: as restrições "convenientes" e as restrições "inconvenientes". Chamamos de restrições "inconvenientes" aquelas que fazem com que o problema se torne difícil de resolver. Dessa forma tais restrições são retiradas do problema de programação inteira e adicionadas à função objetivo em um termo $u(d - Dx)$. Estas variáveis u_i são denominadas variáveis duais ou multiplicadores de Lagrange.

Nem sempre a relaxação lagrangeana nos oferece uma solução ótima para o problema. Ela é limitada pelos resultados da relaxação linear. Porém, mesmo nos casos em que a solução obtida não é ótima, podemos extrair informações interessantes ao interpretar o resultado obtido. O segundo grupo de restrições, as "convenientes" tornam o problema mais fácil de ser resolvido e eventualmente podemos obter a solução ótima para o problema original através da relaxação.

Isso está correto?

O ponto de partida para a aplicação deste método é a elaboração do subproblema lagrangeano. Baseado no modelo de programação inteira que apresentamos na seção 2, o subproblema lagrangeano para o PEC é:

$$\begin{aligned} \max \quad & \sum_{j \in J} (c_j - \sum_{i \in I_j} u_j) x_j + \sum_{i \in I} u_i \\ \text{sujeito a} \quad & x_j \in \{0,1\}, j \in J \end{aligned}$$

Descrição do algoritmo Lagrangeano (pseudo-código do que implementamos). Critérios de parada

Algorithm 1 Subgradiente para o problema de empacotamento de conjuntos

• **Parameters:** $n, t \in \mathbb{N}$, where $t < n$

Variáveis:
MaxIter := Número máximo de iterações definido
s = zeros(m), sqrSum = 0, $\pi_{min} = 0.0001$
$k = 0$ ▷ Índice da iteração
Definir $\pi = 2$ e multiplicadores de Lagrange $u = 0$
Resolver o subproblema Lagrangeano, obtendo: $\overline{x}$, $\overline{z}$
$k := k + 1$

if $k =$ MaxIter **then**
    return PARE

end if

if $\overline{z} - \underline{z} < 1$ **then**
    return $\underline{\text{x}}$ ▷ Valor ótimo encontrado

end if

if $\overline{z} = \underline{z}$ **then**
    return o valor ótimo encontrado

end if

Cálculo do tamanho do passo:

for cada produto i **do**
    $s[i] = 1 - \sum_{j \in L} a[i, j] * \overline{x}[j]$
    $u_i = \max\{0, (1 + \epsilon)T \cdot s_i\}, \quad \forall i = 1, \ldots, m$
    sqrSum = sqrSum + $s[i]^2$
end for
$T = \dfrac{\pi * (\overline{z} - \underline{z})}{sqrSum}$

if $T < \pi_{min}$ **then**
    return valor do t muito pequeno - PARE

end if
retornar para a linha ....

Critérios de Parada do Código

4 Branch and Bound

Uma técnica muito aplicada para resolução de problemas combinatórios é a técnica de dividir e conquistar que consiste em resolver subproblemas mais fáceis que o original e após este processo combinar as soluções obtidas até obtermos a solução do problema original. Porém, esta enumeração de subproblemas de forma explícita pode ser muito custosa, e justamente neste contexto, é que algoritmo *Branch-and-Bound* entra em ação.

O algoritmo *Branch-and-Bound* consiste em uma técnica de enumeração implícita de soluções, onde através de limites locais, obtemos limites globais. Toda a eficiência e ganho que o algoritmo *Branch-and-Bound* nos oferece é por conta das podas que acontecem na árvore de enumeração implícita, que podem ocorrer por: otimalidade, inviabilidade e limites duais. A técnica foi definida primeiramente em [6] por Ailsa Land e Alison Doig em 1960, e figura como um dos mais importantes métodos exatos no contexto de resolução de problemas de otimização até os dias atuais.

5 Branch and Cut

Para fortalecermos ainda mais o algoritmo *Branch-and-Bound*, é possível utilizarmos de algoritmos de planos de corte em cada nó de sua árvore de enumeração. Dessa forma pode-se fortalecer a formulação do problema através da adição de informações especialista antes de efetuar o *Branching* de um determinado nó. Isso faz com que se gaste mais tempo em cada nó, na esperança de que isso diminua o número de nós explorados e o tempo total de utilização da CPU.

Como desigualdades válidas para o PEC, utilizamos desigualdades de cliques geradas a partir de um grafo conflito. Este grafo é definido da seguinte forma: cada lance no problema de leilões combinatórios define um vértice do grafo, e dois vértices possuem aresta entre si se ambos os lances selecionam um mesmo produto. Como quaisquer lances conectados selecionam pelo menos um produto em comum, no máximo um lance de cada clique pode ser selecionado para uma solução viável.

Descrição do algoritmo (pseudo-código do que implementamos).

6 Resultados

6.1 Caso Base

Para fins comparativos com os métodos que foram aplicados para a resolução do PROBLEMA DE EMPACOTAMENTO DE CONJUNTOS, utilizamos como instância de teste a chamada *toy2*, com 3 produtos e 4 lances, que pode ser observada na tabela 1.

Produto \ **Lance**	l_1	l_2	l_3	l_4
p_1	1		1	
p_2		1		1
p_3			1	1
Custo	10	12	18	22

Tabela 1: Instância de teste *toy2*.

Esta é uma instância muito simples, criada especificamente para testar o funcionamento adequado dos métodos. É esperado que, para este caso, todos os métodos sempre encontrem o valor ótimo, e as observações práticas corresponderam às espectativas.

Descrever como é a tabela

Colocar Referência das instâncias!

7 Discussão

8 Conclusão

Referências

[1] Richard Karp. Reducibility among combinatorial problems. volume 40, pages 85–103, 01 1972.

[2] A. Tajima and S. Misono. Using a set packing formulation to solve airline seat allocation/reallocation problems. *Journal of the Operations Research Society of Japan*, 42, 03 1999.

[3] Arne Andersson, Mattias Tenhunen, and F. Ygge. Integer programming for combinatorial auction winner determination. *ICMAS*, 05 2000.

[4] Yuzo Fujishima, Kevin Leyton-Brown, and Yoav Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. volume 1, pages 548–553, 01 1999.

[5] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. Cabob: A fast optimal algorithm for combinatorial auctions. pages 1102–1108, 01 2001.

[6] A. Land and A. Doig. An automatic method of solving discrete linear programming problems. *Econometrica*, 28:497–520, 07 1960.