

Método Exato para Um Problema de Alocação Justa

Edênis F. Azevedo^{1,4}, Carlos E. Ferreira¹, Alexandre S. Freire²,
Aritanan Gruber³, Augusto Vellozo⁴

¹ Instituto de Matemática e Estatística(IME-USP) - São Paulo,SP - Brasil

²Escola de Artes,Ciências e Humanidades(EACH-USP) - São Paulo,SP - Brasil

³Universidade Federal do ABC(UFABC) - Santo André,SP - Brasil

⁴Tecsinapse - São Paulo,SP - Brasil

{denisfa,cef,afreire}@ime.usp.br, aritanan.gruber@gmail.com,

augusto.vellozo@tecsinapse.com.br

Agradecemos a TecSinapse e a FAPESP (projetos 2012/17585-9,2013/03447-6 e 2014/23269-8) pelo apoio financeiro durante a execução deste trabalho.

Abstract. *We consider the problem of fairly allocating a collection of indivisible goods to agents. It is desired to find an allocation such that all agents are equally satisfied. This problem has an application of a company that wants to allocate its production of vehicles to its dealers. We give an exact method for the problem and some preliminary results of the execution with some instances of the application.*

Resumo. *Em um problema de alocação justa, é dada uma coleção de itens que deve ser alocada aos competidores e deseja-se encontrar uma alocação em que todos os competidores fiquem igualmente satisfeitos. Este problema possui uma aplicação na qual uma distribuidora deseja distribuir sua produção de veículos entre concessionárias. Apresentamos um método exato para o problema e alguns resultados preliminares da execução com algumas instâncias da aplicação.*

1. Introdução

Consideraremos um problema de alocação justa no qual é dada uma coleção de itens I que deve ser dividida entre um conjunto de competidores K . Cada competidor deseja maximizar sua satisfação considerando os itens alocados a ele. Desta forma, desejamos encontrar uma alocação de itens aos competidores que seja justa segundo a métrica que será apresentada adiante. Cada competidor expressa sua preferência por meio de uma função que avalia numericamente sua satisfação ao receber uma coleção de itens. O objetivo é encontrar uma alocação de itens para os competidores que maximize suas satisfações e satisfaça as propriedades de uma métrica de justiça considerada.

Para classificar os itens da coleção I , existe um conjunto de categorias H nas quais todo item da coleção I pertence a alguma categoria de H . Cada competidor está interessado na quantidade de itens de cada categoria alocados a ele. Para cada categoria h de H , há uma quantidade a_h que corresponde ao número de itens da categoria h disponíveis para alocação. Cada competidor deve receber uma quantidade fixa de itens, denominada

demanda, dada por um vetor m de $|K|$ posições onde m_k é a quantidade exata de itens que deve ser alocada para o competidor k . Desta maneira, uma alocação deve gerar, para cada competidor k , um vetor indexado pelas categorias de H , no qual cada posição representa a quantidade de itens da categoria alocadas a ele. Portanto, uma alocação é definida como uma função $\alpha : K \rightarrow \mathbb{N}^H$, tal que $\sum_{k \in K} \alpha(k)_h \leq a_h$, para todo h em H e $\sum_{h \in H} \alpha(k)_h = m_k$, para todo competidor k em K . Em seguida, apresentamos a expressão das preferências de um competidor sobre a coleção de itens. A preferência é expressa por uma matriz $P_{|K| \times |H|}$, na qual $P_{k,h}$ corresponde à quantidade de itens da categoria h que o competidor k deseja receber. A satisfação máxima por categoria h do competidor k é atingida quando a quantidade alocada de h para k é exatamente igual a $P_{k,h}$. A função de satisfação é construída considerando as preferências dos competidores, de forma a favorecer o atendimento dos valores $P_{k,h}$ de cada competidor k e categoria h . Além disso, deve favorecer o atendimento de categorias com quantidades desejadas maiores e atender proporcionalmente quantidades desejadas iguais. Desta forma, definimos a função de satisfação como $\phi : K \times \mathbb{N}^H \rightarrow \mathbb{R}$, onde $\phi(k, \alpha) = - \sum_{h \in H} \frac{|P_{k,h} - \alpha(k)_h|^2}{m_k}$ para um competidor k e uma alocação α .

Esse problema possui uma aplicação na qual uma distribuidora deseja distribuir sua produção de veículos entre concessionárias. Os veículos são divididos em modelos e cores e as demandas das concessionárias são determinadas pela distribuidora. As preferências das concessionárias são expressas por pedidos de quantidades de cada modelo e cor. A satisfação de cada concessionária é dada pelo atendimento de seus pedidos. Por regras de negócio internas, toda concessionária pode questionar o resultado de uma alocação. Por isso, o problema foi desenvolvido para oferecer um certificado de justiça que possa ser utilizado pela distribuidora para justificar uma alocação para suas concessionárias.

Os resultados conhecidos para problemas de alocação justa com itens indivisíveis são recentes e, em geral, variam de acordo com as restrições impostas sobre a função de satisfação e com a métrica de justiça escolhida. Para funções de satisfação monótonas e polinomiais, [Golovin 2005] demonstrou que o problema é NP-difícil. Para o caso da função de satisfação genérica, não necessariamente monótona, [Golovin 2005] demonstrou que computar uma aproximação também é NP-difícil. Para a variante considerada, não se sabe se o problema permanece NP-difícil.

O trabalho está organizado da seguinte forma. Na seção 2 definimos a métrica de justiça, enquanto na seção 3 modelamos um problema auxiliar usado pelo método. Já na seção 4, apresentamos o método exato. Por fim, na seção 5 apresentamos a conclusão.

2. Métricas de justiça para uma alocação

Uma alocação α é dita *justa* se aumentar a satisfação de um competidor k necessariamente implica em diminuir a satisfação de um competidor l para um valor menor ou igual ao de k antes de sua satisfação ser aumentada. Mais precisamente, para toda alocação $\beta \neq \alpha$, vale que se $\phi(k, \beta) > \phi(k, \alpha)$ para algum k em K , então existe um competidor l em $K \setminus \{k\}$ tal que $\phi(l, \beta) < \phi(l, \alpha)$ e $\phi(l, \beta) \leq \phi(k, \alpha)$. O conceito-chave para a montagem desta métrica é a justiça *max-min* como definido por [Le Boudec 2008], que surgiu na área de redes de telecomunicações. O PROBLEMA DE ALOCAÇÃO JUSTA consiste em, dada uma instância (K, H, I, P, m) , encontrar uma alocação justa dos itens de I para os competido-

res de K , satisfazendo as demandas dadas. Definimos um outro problema importante para o método exato, uma variante do problema estudado por [Bansal e Sviridenko 2006], denominado PROBLEMA DE CONJUNTO MÍNIMO DE PIOR CASO (PCMPC). Este problema consiste em, dada uma instância (K, H, I, P, m) , encontrar uma alocação que maximize o valor de satisfação do competidor mais insatisfeito. Além disso, o conjunto dado pelos competidores mais insatisfeitos deve ser mínimo.

3. Modelo de programação quadrática para o PCMPC

Sejam $x_{k,h}$ variáveis inteiras para representar a quantidade de unidades da categoria h alocadas ao competidor k , z uma variável contínua que representa a menor satisfação entre os competidores de K e b_k uma variável binária que, para cada competidor k em K , é igual a 1 se sua satisfação é maior do que o valor z e 0, caso contrário. Seja $\epsilon > 0$ tal que $b_k = 1$ se $-\sum_{h \in H} \frac{(p_{k,h} - x_{k,h})^2}{m_k} - z > \epsilon$. Seja $\epsilon^* = \frac{1}{(|K|+1) \cdot \max_k(m_k)}$ de forma que a função objetivo assegura o comportamento desejado de primeiro maximizar a satisfação z do competidor de K mais insatisfeito e depois encontrar um conjunto mínimo de competidores com satisfação z .

$$\text{maximize } z + \epsilon^* \sum_{k \in K} b_k \quad (1)$$

$$\text{sujeito a } \sum_{h \in H} x_{k,h} = m_k \quad \forall k \in K \quad (2)$$

$$(PQ_{PCMPC}) \quad \sum_{k \in K} x_{k,h} \leq a_h \quad \forall h \in H \quad (3)$$

$$-\sum_{h \in H} \frac{(p_{k,h} - x_{k,h})^2}{m_k} \geq z \quad \forall k \in K \quad (4)$$

$$-\sum_{h \in H} \frac{(p_{k,h} - x_{k,h})^2}{m_k} - (\epsilon \cdot b_k) \geq z \quad \forall k \in K \quad (5)$$

$$x_{k,h} \in \mathbb{N}, b_k \in \{0, 1\}, z \in \mathbb{R}$$

O método exato que será apresentado resolve um problema parecido com o PCMPC, no qual considera dois conjuntos de competidores R e $\bar{R}$, onde $R \cup \bar{R} = K$ e $R \cap \bar{R} = \emptyset$. O problema do método apresenta uma restrição na qual todo competidor de R deve ter sua satisfação fixada em um valor por uma alocação. Esses valores são dados por um vetor s de $|R|$ posições onde s_k é a satisfação exata que deve ser obtida por uma alocação para o competidor k . As alterações de (PQ_{PCMPC}) para obter o modelo do método (PQ_{PCMPCR}) consistem em substituir o conjunto K por $\bar{R}$ em (4) e (5), além de adicionar a seguinte restrição.

$$-\sum_{h \in H} \frac{(p_{k,h} - x_{k,h})^2}{m_k} = s_k \quad \forall k \in R \quad (6)$$

4. Método exato para o problema de alocação justa

Nesta seção, apresentaremos um método exato para resolver o problema de alocação justa. A ideia do algoritmo é construir iterativamente um conjunto R de competidores e, em cada iteração, resolver uma instância do (PQ_{PCMPCR}) para $\bar{R}$. Inicialmente, $R = \emptyset$ e $\bar{R} = K$. Em cada iteração, todo competidor que estiver no conjunto mínimo de uma solução do

(PQ_{PCMPCR}) será adicionado ao conjunto R e removido de $\bar{R}$. Além disso, a satisfação dos competidores no conjunto mínimo será fixada no valor encontrado por uma solução do (PQ_{PCMPCR}) para as iterações seguintes. Desta maneira, a última alocação encontrada no laço do método deve verificar as propriedades da justiça definida anteriormente.

Algoritmo 1: Método exato para o problema de alocação justa.

Entrada: Instância (K, H, I, P, m) do problema de alocação justa.

Saída: Vetor x representando a alocação encontrada.

```

1  início
2  |  $(R, \bar{R}) \leftarrow (\emptyset, K)$ 
3  | faça
4  |    $(z, x, b) \leftarrow \text{solução do } (PQ_{PCMPCR}) \text{ para } (\bar{R}, R, H, I, P, m, s)$ 
5  |    $D \leftarrow \{k \in \bar{R} | b_k = 0\}$ 
6  |   para  $k \in D$  faça
7  |   |  $s_k \leftarrow z$ 
8  |    $(R, \bar{R}) \leftarrow (R \cup D, \bar{R} \setminus D)$ 
9  | enquanto  $\bar{R} \neq \emptyset$ 
10 | devolve  $x$ 

```

Considere uma alocação gerada pelo método. A ideia da prova de corretude é supor que existe uma outra alocação na qual é possível aumentar a satisfação de um competidor sem violar a propriedade da justiça definida anteriormente. Ao considerar a iteração em que o competidor em questão foi adicionado a R pelo método, deve se verificar um absurdo com relação a otimalidade de uma solução do (PQ_{PCMPCR}) da iteração.

4.1. Resultados preliminares

O método descrito acima foi implementado em python e o resolvedor *GUROBI* foi utilizado na implementação. As instâncias da aplicação apresentam, em média, 130 competidores e 35 categorias. Foram executadas 11 instâncias da aplicação. O tamanho da coleção de itens varia entre 3000 e 4500 unidades. O valor $\sum_{h \in H} P_{k,h}$ é próximo de m_k do competidor k em todas as instâncias. O tempo de execução médio foi de 4 horas.

5. Conclusão e trabalhos futuros

Definimos um problema de alocação justa a partir de uma aplicação e propusemos um método exato para resolver o problema. O método foi implementado e resolveu instâncias da aplicação em tempo computacional aceitável. Como trabalho futuro, queremos apresentar experimentos mais detalhados, formalizar a demonstração da corretude do método e provar que o problema é NP-difícil.

Referências

- Bansal, N. and Sviridenko, M. (2006). The santa claus problem. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing, STOC '06*, pages 31–40, New York, NY, USA. ACM.
- Golovin, D. (2005). Max-min fair allocation of indivisible goods. Technical Report CMU-CS-05-144, School of Computer Science, Carnegie Mellon University.
- Le Boudec, J.-Y. (2008). Rate adaptation, Congestion Control and Fairness: A Tutorial.

Teorema 4 (Moser e Tardos [Moser and Tardos 2010]). *Seja $\mathcal{P}$ um conjunto finito de variáveis aleatórias mutuamente independentes num mesmo espaço de probabilidade e $\mathcal{A}$ uma coleção finita de eventos determinados por essas variáveis. Se $\mathcal{A}$ satisfaz as condições do LLL, então existe uma atribuição de valores às variáveis de $\mathcal{P}$ que não viola nenhum dos eventos de $\mathcal{A}$. Além disso, o número esperado de reamostragens do evento $A \in \mathcal{A}$ que o algoritmo aleatório acima faz é no máximo $x(A)(1 - x(A))^{-1}$. Portanto, o número total esperado de amostragens é $\sum_{A \in \mathcal{A}} x(A)(1 - x(A))^{-1}$. ■*

4. O algoritmo na prática

Uma análise mais cuidadosa do algoritmo de Moser e Tardos é feita em [Haeupler et al. 2011]. Um dos parâmetros fundamentais da análise é

$$\delta := \min_{A \in \mathcal{A}} x(A) \prod_{B \in \Gamma(A)} (1 - x(B)).$$

Note que, nas hipóteses do LLL, vale que $\delta \geq \min_{A \in \mathcal{A}} \mathbb{P}(A)$. O primeiro resultado de [Haeupler et al. 2011] é o seguinte.

Teorema 5 (Haeupler, Saha e Srinivasan). *Suponha que estamos nas mesmas condições do Teorema 4. Então o número total esperado E de amostragens realizadas pelo algoritmo satisfaz $E \leq n(\log 1/\delta) \max_{A \in \mathcal{A}} (1 - x(A))^{-1}$. ■*

Aplicação em colorações frugais

Proposição 6. *Seja G um grafo nas condições do Teorema 3. Existe um algoritmo aleatório com tempo esperado $O(\beta n^2 \Delta \log \Delta)$ que encontra uma coloração β -frugal de G .*

Demonstração. Note que é possível descobrir se uma coloração não é frugal em tempo $O(n\Delta)$. Além disso, na prova do Teorema 3 os eventos satisfazem $\delta \geq \min_{A \in \mathcal{A}} \mathbb{P}(A) = 1/C^{\beta-1}$. Portanto, $\log 1/\delta \leq \log(C^{\beta-1}) \leq \beta \log \Delta$. Temos ainda que $\max_{A \in \mathcal{A}} (1 - x(A))^{-1} = 1/(1 - 2/C) \leq 3$. Segue que o tempo esperado do algoritmo de Moser e Tardos nesse problema é $E = O(\beta n^2 \Delta \log \Delta)$. ■

Referências

- Erdős, P. and Lovász, L. (1975). Problems and results on 3-chromatic hypergraphs and some related questions. In *Infinite and finite sets*, pages 609–627. North-Holland, Amsterdam.
- Haeupler, B., Saha, B., and Srinivasan, A. (2011). New constructive aspects of the Lovász local lemma. *J. ACM*, 58(6):Art. 28, 28.
- Hind, H., Molloy, M., and Reed, B. (1997). Colouring a graph frugally. *Combinatorica*, 17(4):469–482.
- Hind, H., Molloy, M., and Reed, B. (1999). Total coloring with $\Delta + \text{poly}(\log \Delta)$ colors. *SIAM J. Comput.*, 28(3):816–821.
- Leighton, T., Maggs, B., and Richa, A. W. (1999). Fast algorithms for finding O (congestion + dilation) packet routing schedules. *Combinatorica*, 19(3):375–401.
- Molloy, M. and Reed, B. (1999). Further algorithmic aspects of the local lemma. In *STOC '98 (Dallas, TX)*, pages 524–529. ACM, New York.
- Moser, R. A. and Tardos, G. (2010). A constructive proof of the general Lovász local lemma. *J. ACM*, 57(2):Art. 11, 15pp.