Teoria dos Jogos Algorítmica e Otimização Combinatória

Atol Fortin de Oliveira

5 de fevereiro de 2010

Sumário

1 Introdução

Nos últimos anos, notou-se que vários problemas de otimização combinatória podem ser olhados sob o ponto de vista de teoria dos jogos, e isso tem trazido resultados interessantes para tais problemas, pois permite que técnicas e observações da área de teoria dos jogos sejam usadas ou sirvam de inspiração. Ao mesmo tempo, o crescimento da internet trouxe consigo uma variedade de problemas novos, muitos deles envolvendo aspectos de teoria dos jogos. Neste projeto mostraremos problemas de otimização combinatória sob o enfoque de teoria dos jogos.

Mais precisamente, neste projeto trataremos de quatro problemas. Os dois primeiros deles são bons exemplos para situar o leitor sobre algumas características que encontramos em problemas da área de teoria dos jogos, pois possuem soluções simples e até intuitivas. Já os outros dois problemas são mais sofisticados, além de apresentarem interesses práticos.

2 Casamento Estável

Considere um grupo de homens e um grupo de mulheres que se conhecem, todos solteiros. Podemos imaginar que cada homem tem uma ordem de preferência para as mulheres do grupo, desde a que mais gosta até a mulher que menos gosta, sem empates. O mesmo vale para as mulheres, de maneira análoga. O problema em que estamos interessados é chamado de *problema dos casamentos estáveis*.

Supondo que os dois grupos têm o mesmo tamanho, queremos escolher uma esposa para cada homem, levando em conta as ordens de preferência, de modo que os casamentos sejam *estáveis*. Caso os grupos não sejam do mesmo tamanho, podemos adicionar "manequins" no grupo menor, de forma que não atrapalhem nas decisões finais de casamentos, e ao final consideramos que os que fiquem casados com tais manequins ficam solteiros.

2.1 Notação

Denotamos por H o conjunto dos homens e por M o das mulheres, e consideramos então que $|H| = |M| = n$. Todo $m \in M$ tem uma ordem de preferência nos elementos de H: para a e $b \in H$, e $m \in M$, $a \succ_m b$ denota que a mulher m prefere a mais do que prefere b. Assim temos uma ordem de preferência estrita:

$$h_1 \succ_m h_2 \succ_m h_3 \succ_m \cdots \succ_m h_i \succ_m \cdots \succ_m h_n,$$

onde o elemento m tem maior preferência por h_1, e menor preferência por h_n.

O mesmo vale para cada elemento $h \in H$, que possui uma ordem de preferência $\succ_h$ nos elementos de M. Logo, cada elemento de $M \cup H$ possui sua ordem de preferência, indicando com qual elemento tem preferência a se casar.

2.2 Estabilidade

Agora que temos em mãos os conjuntos e suas preferências, precisamos descrever quando um grupo de casamentos entre H e M é estável.

Um grupo de casamentos entre H e M é dito *instável* se temos dois homens h_1 e $h_2 \in H$ e duas mulheres m_1 e $m_2 \in M$ onde h_1 está casado com m_1 e h_2 está casado com m_2, mas $m_2 \succ_{h_1} m_1$ e $h_1 \succ_{m_2} h_2$, ou seja, h_1 prefere m_2 a m_1, e m_2 prefere h_1 a h_2. Nesse caso, dizemos o par (h_1, m_2) é um *par bloqueador*, e temos então um grupo de casamentos instável. Um grupo de casamentos que não possui nenhum par bloqueador é *estável*.

2.3 Exemplo

Vamos considerar três homens h_1, h_2 e h_3 e três mulheres m_1, m_2 e m_3, com as seguintes ordens de preferência:

$$\begin{array}{lccccc}
h_1\text{:} & m_2 & \succ_{h_1} & m_1 & \succ_{h_1} & m_3 \\
h_2\text{:} & m_1 & \succ_{h_2} & m_3 & \succ_{h_2} & m_2 \\
h_3\text{:} & m_1 & \succ_{h_3} & m_2 & \succ_{h_3} & m_3 \\
m_1\text{:} & h_1 & \succ_{m_1} & h_3 & \succ_{m_1} & h_2 \\
m_2\text{:} & h_3 & \succ_{m_2} & h_1 & \succ_{m_2} & h_2 \\
m_3\text{:} & h_1 & \succ_{m_3} & h_3 & \succ_{m_3} & h_2
\end{array}$$

Suponha que temos os casamentos (h_1, m_1), (h_2, m_2) e (h_3, m_3). Podemos então ver que esse grupo de casamentos é instável, pois (h_1, m_2) é um par bloqueador. Já o grupo de casamentos (h_1, m_1), (h_2, m_3) e (h_3, m_2) é estável.

2.4 Algoritmo da proposta masculina

Vamos ver um algoritmo que encontra um grupo estável de casamentos entre elementos de H e M. Este algoritmo deve-se a Gale e Shapley (1962) [3]. Ele recebe os conjuntos H e M e a ordem de preferência de cada pessoa em $H \cup M$, e é iterativo.

Começamos com um conjunto S de homens solteiros, um conjunto de mulheres solteiras G, e uma lista L de casais, que descreve com qual homem uma mulher está casada. Inicialmente $S = H$, $G = M$ e $L = \emptyset$. (Estamos usando o conjunto G apenas para facilitar.) Temos também uma marcação para identificar qual será a próxima mulher que receberá a proposta de um homem $h \in H$, e chamaremos essa marcação de $F(h)$. Inicialmente cada homem h irá fazer a proposta à mulher mais bem colocada na sua ordem de preferência, ou seja, $F(h)$ é a primeira mulher na ordem de preferência de h.

Então pegamos o conjunto $S' = S$, e para cada homem $s \in S'$ olhamos para a próxima mulher $p = F(s)$ a quem ele deve fazer uma proposta (que é a mulher mais bem colocada na sua ordem de preferência que ainda não o rejeitou).

Caso 1. p está no grupo G de mulheres solteiras. Então removemos p de G, e também removemos s de S, além de atualizar $F(h)$ com a próxima mulher logo abaixo de p na sua ordem de preferência, e atualizar a lista L marcando que s está casado com p. (Observação: este casamento pode ser desmanchado ainda nesta iteração; veja o Caso 2 para entender.)

Caso 2. p não está em G, ou seja, a mulher p já está casada com algum outro homem. Então ela deve decidir se irá se casar com s, ou manter seu casamento, e para isso basta ela escolher utilizando sua ordem de preferência.

Caso 2.1 p opta por manter seu casamento (rejeitando s). Basta atualizar a próxima mulher a quem s fará sua proposta.

Caso 2.2 p decide se casar com s (rejeitando seu antigo marido). Então precisamos remover s de S, e inserir o antigo marido da mulher p em S. (Repare que, como ele não era solteiro, não haverá duplicações no conjunto S.) Também atualizamos a lista L e $F(s)$. Começamos uma nova iteração, com os novos conjuntos S e G, a lista L, e a função F.

O algoritmo termina quando o conjunto S fica vazio, o que significa que todos os homens estão casados, chegando, como veremos, a um grupo estável de casamentos.

Algoritmo PROPOSTA MASCULINA $(H, M, \prec_h)$

1 $S \leftarrow H$, $L(m) \leftarrow$ indefinido, para cada $m \in M$

```
2   para todo h ∈ H faça
3       F(h) é a primeira mulher na ordem de preferência de h
4   enquanto S ≠ ∅ faça
5       seja h um homem em S faça
6           se L(F(h)) está indefinido        // a mulher F(h) está solteira
7               então L(F(h)) ← h, S ← S \ {h}
8           senão
9               se h ≻_{F(h)} L(F(h))        // h é melhor na ordem de preferência de F(h)
10                  então S ← S ∪ {L(F(h))}, L(F(h)) ← h, S ← S \ {h}
11              // senão a mulher F(h) rejeita o homem h, e nada precisa ser feito
12          atualizamos F(h) para a próxima mulher na ordem de preferência do homem h
13  devolva L
```

2.4.1 Correção do algoritmo

Suponha que, ao final do algoritmo, não temos um grupo estável de casamentos. Logo temos um par bloqueador. Seja ele o par (h_1, m_2), onde h_1 é casado com m_1 e m_2 é casada com h_2. Pela definição de par bloqueador, $m_2 \succ_{h_1} m_1$, e então o homem h_1 teria feito uma proposta à mulher m_2 antes de fazer à mulher m_1. Como não temos o casal (h_1, m_2), isso implica que m_2 recebeu uma proposta de alguém que ela prefere em relação ao homem h_1. Dado que m_2 está casada com h_2 ao final do algoritmo, temos então $h_2 \succ_{m_2} h_1$, uma contradição.

Portanto não existe tal par bloqueador, o que prova que o algoritmo realmente encontra um grupo estável de casamentos entre os conjuntos H e M.

Veja que o algoritmo produz um grupo estável de casamentos independente da escolha de h na linha 5. É natural no entanto pensar em S como uma fila.

2.4.2 Implementação

O algoritmo acima foi implementado e pode ser acessado em
`http://www.linux.ime.usp.br/~atol/ic/teoria_dos_jogos/algoritmos`.

Temos a iteração principal, onde algum homem solteiro faz proposta à próxima mulher da sua ordem de preferência. Para deixar o código mais eficiente, temos que iterar apenas nos homens solteiros, e para isso utilizei uma fila. Caso o homem escolhido na linha 5 seja rejeitado, inserimos ele de volta na fila. Essa fila é usada no algoritmo acima fazendo o papel de S.

Dentro desta iteração principal, temos que fazer a proposta de algum homem solteiro. Para isso precisamos ter informações sobre a mulher m a quem um homem h está propondo. A mulher m só aceitará a proposta do homem h se este aparecer antes do que seu atual marido na sua ordem de preferência. Para isso, da forma como está descrito, parece que estamos varrendo a ordem de preferência da mulher m e verificando quem aparece antes, o que seria ineficiente. Por isso temos ainda uma outra matriz que guarda um valor atribuído a cada homem por cada mulher. Esses valores são inteiros de 1 a n, sendo que n é o valor do homem preferido de cada mulher, $n-1$ é o valor do segundo homem preferido, etc. Então, na verdade, basta verificar os valores atribuídos pela mulher m tanto a seu atual marido, quanto ao homem h que está lhe fazendo uma proposta, acessando as posições da matriz de valores sem precisar de varredura, para decidir quem será o marido da mulher m ao final desta proposta.

2.4.3 Consumo de tempo

Sabemos que cada homem só faz proposta a cada mulher no máximo uma única vez, ou seja, cada homem vai fazer no máximo n propostas, ao longo de todo algoritmo. Como temos n homens, e cada proposta é feita em tempo constante, chegamos a $O(n^2)$ operações durante o algoritmo.

Existem casos onde o algoritmo termina em tempo $O(n)$. Por exemplo, quando cada homem prefere uma mulher diferente do que todos os outros homens, o algoritmo termina após n propostas. Um dos piores casos para o algoritmo é quando todo homem possui a mesma ordem de preferência nas mulheres, chegando a algo próximo de $n^2/2$ propostas.

2.5 Ótimo masculino

Claro que poderíamos fazer o algoritmo acima trocando os papéis dos homens e das mulheres. Nesse caso, teríamos uma *proposta feminina*, que também resultaria em um grupo estável de casamentos, mas não necessariamente igual.

Pode existir mais do que um grupo estável de casamentos entre os conjuntos. O mais estudado dentre esses grupos é justamente o grupo encontrado pelo algoritmo de Gale-Shapley (1962), descrito na subseção 2.4, também chamado de proposta masculina devido ao modo como ele funciona. Vamos ver uma propriedade sobre esse grupo estável de casamentos que encontramos através deste algoritmo.

Seja X um grupo de casamentos. A mulher casada ao homem h no grupo X será denotada por $X(h)$, assim como o homem casado à mulher m será denotado por $X(m)$. Um grupo estável de casamentos X é *ótimo masculino* se não existe um grupo estável de casamentos Y tal que $Y(h) \succ_h X(h)$ ou $Y(h) = X(h)$ para todo h, com pelo menos uma ocorrência da primeira.

A situação é muito semelhante para um grupo estável de casamentos ser *ótimo feminino*.

O algoritmo da proposta masculina encontra um grupo estável de casamentos ótimo masculino. Seja X o grupo estável de casamentos encontrado pelo algoritmo e suponha que ele não é ótimo masculino. Então deve existir um grupo estável de casamentos Y tal que $Y(h) \succ_h X(h)$ ou $Y(h) = X(h)$ para todo $h \in H$, com pelo menos uma ocorrência da primeira condição. Porém, se $Y(h) \succ_h X(h)$ para algum $h \in H$, então o homem h propos à mulher $Y(h)$ antes de propor à mulher $X(h)$, segundo o algoritmo. Mas já que o homem h não terminou casado com $Y(h)$ no grupo de casamentos X, isso implica que algum homem $p \in H$ propos à mulher $Y(h)$ e este homem p tem maior preferência do que h na ordem de preferência da mulher $Y(h)$, ou seja, $p \succ_{Y(h)} h$. Por outro lado, o homem p não terminou casado com a mulher $Y(h)$ no grupo de casamentos Y, o que implica que ele terminou casado com uma mulher de menor preferência (se fosse de maior preferência, ele nunca teria proposto à mulher $Y(h)$, durante a execução do algoritmo para gerar o grupo de casamentos X), logo $Y(h) \succ_p Y(p)$. Juntando as duas desigualdades, temos um par bloqueador no grupo de casamentos Y, uma contradição. Portanto não existe tal grupo de casamentos Y, concluindo que o grupo de casamentos encontrado pelo algoritmo é ótimo masculino.

2.6 Prova de estratégia

Vamos mostrar que uma pessoa pode se beneficiar ao declarar sua ordem de preferência diferente de suas reais preferências. Por exemplo, os casamentos resultantes da aplicação do algoritmo às ordens do exemplo da subseção 2.3 são (h_1, m_2), (h_2, m_3) e (h_3, m_1). Agora vamos alterar a ordem de preferência da mulher m_1 para: $h_1 \succ_{m_1} h_2 \succ_{m_1} h_3$.

Repare que no resultado do algoritmo para o exemplo inicial a mulher m_1 havia terminado casada com o homem h_3, que é o segundo na sua ordem de preferência original. Mas ao alterar sua ordem de preferência, ela consegue acabar casada com o homem de maior preferência h_1. Veja a simulação do algoritmo na subseção (2.6.1).

Com isso, chegamos à conclusão de que esse algoritmo não é à prova de estratégia. Ou seja, uma pessoa pode relatar uma ordem de preferência que não é verdadeira para se beneficiar. Essa pessoa, no caso do algoritmo da prosposta masculina, é na verdade uma mulher. Segue a prova de que os homens não conseguem nenhum benefício alterando a declaração de suas ordens de preferência.

Suponha que para o conjunto de ordens de preferências reais $P = (\succ_{h_1}, \succ_{h_2}, \ldots, \succ_{h_n})$ exista um homem que falsifique sua ordem de preferência e consiga se beneficiar com isso. Sem perda de generalidade, suponha que esse homem seja o h_1. Seja μ o grupo estável de casamentos devolvido pelo algoritmo aplicado a P, onde $\mu(h_i)$ representa a mulher casada ao homem h_i em μ, e $\mu(m_i)$ representa o homem casado à mulher m_i em μ, para $1 \leq i \leq n$. Seja μ' o grupo estável de casamentos devolvido pelo algoritmo aplicado a P', onde P' representa o conjunto de ordens de preferência quando o homem h_1 falsifica suas reais preferências, declarando $\succ_*$. Ou seja, $P' = (\succ_*, \succ_{h_2}, \ldots, \succ_{h_n})$. Como o homem h_1 se beneficiou ao declarar $\succ_*$ no lugar de $\succ_{h_1}$, temos que $\mu'(h_1) \succ_{h_1} \mu(h_1)$. Mostraremos que se $\mu'(h_1) \succ_{h_1} \mu(h_1)$, então o grupo de casamentos μ' não é estável com relação a P', uma contradição.

Seja $R = \{h : \mu'(h) \succ_h \mu(h)\}$, ou seja, R é o conjunto de homens que se beneficiaram no grupo de casamentos μ'. Mostraremos que, para todo $h \in R$, se $m = \mu'(h)$ e $h' = \mu(m)$, vale que $h' \in R$. Ou seja, se um homem h se beneficiou em μ', então o homem h' que estava casado com $\mu'(h)$ em μ também se beneficou em μ'. Como $m \succ_h \mu(h)$, a estabilidade de μ implica que $h' = \mu(m) \succ_m h$. A estabilidade de μ' (com relação a P') implica que $\mu'(h') \succ_{h'} m$, já que $h' \succ_m h$ mas m ficou casada com h em μ'. Como h' estava casado com m em μ, temos que $h' \in R$. Definimos $S = \{m : \mu'(m) \in R\}$. Observe que, pela discussão anterior, $S = \{m : \mu(m) \in R\}$.

Como $\mu(m) \succ_m \mu'(m)$ para toda mulher $m \in S$, durante a execução do algoritmo da proposta masculina aplicada a P, cada $m \in S$ rejeita $\mu'(m) \in R$ em alguma iteração. Seja $h \in R$ o último homem a fazer uma proposta durante a execução do algoritmo para P. Essa proposta é feita à $m = \mu(h) \in S$, que, pela escolha de h, rejeitou $\mu'(m)$ em alguma iteração anterior do algoritmo. Isso significa que quando h propõe a m, ela rejeita a proposta de algum $h' \notin R$, tal que $h' \succ_m \mu'(m)$. Como $h' \notin R$, temos que $m \succ_{h'} \mu(h') \succeq_{h'} \mu'(h')$. Portanto (h', m) forma um par bloqueador em μ' para P', concluindo que o algoritmo é à prova de estratégia para os homens.

Por outro lado, aplicando o algoritmo da prosposta feminina, temos um resultado oposto, ou seja, de nada adianta às mulheres falsificarem suas ordens de preferência, pois apenas os homens podem obter alguma vantagem com essa estratégia. O algoritmo da proposta feminina é muito semelhante ao algoritmo da proposta masculina. Basta inverter a posição de homens e mulheres durante a execução do algoritmo, ou seja, as mulheres fazem propostas e os homens aceitam ou rejeitam tais propostas.

2.6.1 Simulação

Para entender melhor como acontece a modificação do resultado final, podemos acompanhar a simulação do algoritmo para a entrada original, e depois observar as modificações que ocorrem ao alterarmos a ordem de preferência da mulher m_1.

Primeiras três iterações: h_1 propõe à mulher m_2, h_2 propõe à mulher m_1, e h_3 propõe à mulher m_1 também. Nesse ponto, m_1 rejeita a proposta de h_2, e ao final destas iterações temos

os casais (h_1, m_2), (h_2,__), (h_3, m_1).

Quarta iteração: agora temos apenas h_2 solteiro, que propõe à mulher m_3, terminando assim com o grupo de casamentos (h_1, m_2), (h_2, m_3), (h_3, m_1).

Agora suponha que a mulher m_1 em vez de rejeitar o homem h_2 tivesse rejeitado o outro homem que lhe tinha feito proposta nas primeiras iterações, h_3. Então, na quarta iteração, o homem h_3 irá propor à mulher m_2 que estava casada com h_1. Seguindo sua ordem de preferência, m_2 irá rejeitar o seu antigo marido h_1, criando um novo casal (h_3, m_2). Após isso, na quinta iteração, temos apenas o homem h_1 solteiro, que irá propor à mulher m_1, que estava casada com o homem h_2, mas irá aceitar essa nova proposta, deixando o homem h_2 solteiro e criando o casal (h_1, m_1). Na sexta iteração, o homem h_2, que está solteiro, irá propor à mulher m_3 que estava solteira até então, teminando o algoritmo com o grupo de casamentos (h_1, m_1), (h_2, m_3), (h_3, m_2).

2.7 Olhando por outro ângulo

Uma outra maneira de resolver o problema dos casamantos estáveis é usando programação linear. Para cada h em H e m em M, temos uma variável x_{hm}. Definindo que $x_{hm} = 1$ se um homem $h \in H$ e uma mulher $m \in M$ estiverem casados, e $x_{hm} = 0$ caso contrário , as seguintes desigualdades devem ser satisfeitas para que essas variáveis representem um grupo estável de casamentos entre H e M:

$$\begin{aligned}
\sum_{m \in M} x_{hm} &= 1 \quad \forall h \in H \\
\sum_{h \in H} x_{hm} &= 1 \quad \forall m \in M \\
\sum_{j \prec_h m} x_{hj} + \sum_{i \prec_m h} x_{im} + x_{hm} &\leq 1 \quad \forall h \in H,\ \forall m \in M \\
x_{hm} &\geq 0 \quad \forall h \in H,\ \forall m \in M.
\end{aligned}$$

As duas primeiras restrições garantem que cada pessoa estará casada com exatamente uma pessoa do sexo oposto se $x_{hm} \in \{0, 1\}$ para todo m e h. A terceira restrição garante estabilidade. Vamos verificar isto.

Primeiro observe que, dado um grupo estável de casamentos, fixando $x_{hm} = 1$ se h está casado com m e $x_{hm} = 0$ caso contrário, então os valores x_{hm} satisfazem o programa linear acima. Se $x_{hm} = 1$ então as duas primeiras restrições garantem que as somatórias da terceira restrição valem zero, satisfazendo a terceira restrição. Se $x_{hm} = 0$, então suponha que a terceira restrição não seja satisfeita. Para isso, seria necessário que ambas as somátorias valessem 1, ou seja, $\sum_{j \prec_h m} x_{hj} = 1$ e $\sum_{i \prec_m h} x_{im} = 1$. Então, o homem h está casado com uma mulher j que está pior colocada na sua ordem de preferência do que a mulher m. Da mesma forma, a mulher m está casada com um homem i que está pior colocado na sua ordem de preferência do que o homem h. Assim teríamos o par bloqueador (h, m). Logo, a restrição é satisfeita, pois o grupo de casamentos inicial é estável, e portanto não há pares bloqueadores. Isso nos leva ao fato de que, dado um grupo estável de casamentos, fixando os valores x_{hm} conforme a definição, satisfazemos todas as restrições do programa linear.

Isso significa que o programa linear acima sempre tem uma solução viável onde $x_{hm} \in \{0, 1\}$ para todo h e m. Além disso, pelo teorema abaixo, podemos usar por exemplo o algoritmo simplex para encontrar valores de x_{hm} em $\{0, 1\}$ que satisfaçam o programa linear acima.

Teorema 2.1 *O poliedro definido pelas inequações acima é o casco convexo de todos os grupos de casamentos estáveis.*

A partir destes valores, podemos construir um grupo de casamentos onde o homem $h \in H$ está casado com a mulher $m \in M$ se e somente se $x_{hm} = 1$, e podemos mostrar que ele é estável. Das duas primeiras restrições, conseguimos perceber que todo homem está casado com exatamente uma mulher, e toda mulher está casada com exatamente um homem. Pela terceira restrição, temos que ou o homem h está casado com uma mulher j, com $j \prec_h m$; ou a mulher m está casada com um homem i, com $i \prec_m h$; ou temos o casal (h, m); ou nenhuma destas situações ocorre. Como a primeira e a segunda opções não ocorrem simultaneamente para nenhum h e m, não temos nenhum par bloqueador neste grupo de casamentos, e portanto temos um grupo estável de casamentos.

Então chegamos à conclusão de que os problemas são equivalentes.

2.8 Extensões do problema

Vamos ver superficialmente algumas variações do problema dos casamentos estáveis. Para mais exemplos de aplicações ou variantes, veja o capítulo 12 do livro de Ahuja [2] e outros.

2.8.1 Mercado de admissão em faculdades

Neste problema os casamentos são um pouco diferentes. Os elementos de um dos grupos estão interessados em um ou mais elementos do outro grupo, como é o caso de estudantes e faculdades. Considerando um grupo de faculdades, e um grupo de estudantes, então os estudantes procuram apenas uma faculdade, com alguma ordem de preferência, e as faculdades aceitam mais do que apenas um estudante, e também tem alguma ordem de preferência. Então estamos procurando alguma forma de "casar" os estudantes às faculdades de maneira estável.

2.8.2 Problema da república

A modificação deste problema em relação ao original é que aqui temos apenas um conjunto de pessoas. Os elementos deste único conjunto estão interessados em elementos do próprio conjunto, que é o problema de procurar parceiros de quarto. As pessoas do conjunto têm uma ordem de preferência sobre as outras, e querem escolher alguém para dividir o quarto. Nesse problema, nem sempre existe uma solução, mas conseguimos verificar se existe ou não em tempo quadrático no tamanho do conjunto. Podemos também construir um problema de programação linear para resolver este.

Veja um exemplo de uma instância desse problema que não possui solução. Para o conjunto de pessoas $P = (p_1, p_2, p_3, p_4)$ com as seguintes ordens de preferência

$$\begin{array}{lllllll} p_1\colon & p_3 & \succ_{p_1} & p_2 & \succ_{p_1} & p_4 \\ p_2\colon & p_1 & \succ_{p_2} & p_3 & \succ_{p_2} & p_4 \\ p_3\colon & p_2 & \succ_{p_3} & p_1 & \succ_{p_3} & p_4 \\ p_4\colon & p_1 & \succ_{p_4} & p_2 & \succ_{p_4} & p_3 \end{array}$$

não existe casamento estável. É fácil verificar que em todos os possíveis grupos de casamento existe um par bloqueador.

3 Alocação de bens indivisíveis

Vamos estudar o problema da alocação de casas para entender o modelo de alocação de bens indivisíveis. Nesse problema temos um conjunto $N = \{1,2,\ldots,n\}$ de negociantes e cada negociante possui inicialmente uma casa. As casas são transferíveis e os negociantes podem trocar de casa usando como moeda a sua própria casa. Cada negociante possui também uma ordem de preferência estrita sobre as n casas em questão, ou seja, uma lista ordenada das casas, desde a casa que mais gosta até a casa que menos gosta, sem empates. Nosso objetivo é realocar as casas entre os negociantes, de forma a satisfazer algumas condições, chegando a uma alocação dita estável.

3.1 Notação

A casa j denota a casa inicial do negociante j. Para descrever uma alocação de casas usaremos um vetor x de tamanho n, onde x_i é o número da casa atribuída ao negociante i. Assim temos a alocação inicial $x_i = i$, para i de 1 a n.

Para denotar a ordem de preferência de um negociante i usaremos $\succ_i$, onde $a \succ_i b$ indica que o negociante i prefere a casa a à casa b. Podemos também imaginar uma matriz quadrada M para descrever as ordens de preferência, onde $M = (a_{ij})$, e $a_{ij} > a_{ik}$ significa que o negociante i prefere a casa j à casa k.

Uma alocação de casas x deve satisfazer $x_i \neq x_j$ para todo $i \neq j$, ou seja, uma casa só pode pertencer a um negociante, e será então uma *alocação viável*. O contrário também deve ser válido: todo negociante só pode possuir uma única casa.

3.2 Alocação estável

Intuitivamente, uma alocação estável das casas ocorre quando, levando em conta a alocação inicial das casas, nenhum conjunto de negociantes está disposto a trocar de casas entre si. Para definir isso formalmente, chame de T o conjunto de todas as alocações viáveis. Agora, para todo conjunto $S \subseteq N$, chame de $T(S)$ o conjunto de todas as possíveis alocações viáveis de casas onde os negociantes em S ficam com casas de S e os negociantes fora de S ficam com casas fora de S. Assim $T(S) = \{t \in T\colon\ t_i \in S\ \forall i \in S\}$. Após essas definições, dada uma alocação viável t em T, um conjunto S de negociantes forma um *conjunto bloqueador* em t se existe uma alocação p em $T(S)$ tal que, para todo negociante $i \in S$, temos $p_i \succ_i t_i$ ou $p_i = t_i$, com pelo menos uma ocorrência da primeira. Assim um conjunto bloqueador S pode chegar a uma alocação mais favorável a cada pessoa de S, ou seja, onde cada pessoa de S recebe uma casa que não seja pior na sua ordem de preferência, e pelo menos uma delas recebe uma casa melhor na sua ordem de preferência. (Eventualmente, como veremos no exemplo a seguir, alguma pessoa fora de S ficaria com uma casa pior nesta alocação.) Uma alocação que não possui nenhum conjunto bloqueador é uma alocação *estável*. O conjunto de todas as alocações estáveis é chamado *núcleo*.

3.3 Exemplo

Vamos mostrar um exemplo para ilustrar o problema. Considere as seguintes ordens de preferência:

$$\begin{array}{lccccc} 1\colon & 1 & \succ_1 & 2 & \succ_1 & 3. \\ 2\colon & 1 & \succ_2 & 2 & \succ_2 & 3. \\ 3\colon & 2 & \succ_3 & 3 & \succ_3 & 1. \end{array}$$

A alocação (1,2,3) é estável. Já a alocação $t = (1,3,2)$ não é estável pois $S = \{2\}$ é um conjunto bloqueador. De fato, $T(\{2\}) = \{(1,2,3), (3,2,1)\}$ e tomando a alocação $p = (1,2,3)$ temos que $p_2 \succ_2 t_2$. Observe que $S = \{2,3\}$ não é um conjunto bloqueador em t, pois $T(\{2,3\}) = \{(1,2,3), (1,3,2)\}$ e, para $p = (1,2,3)$, temos que $p_3 \prec_3 t_3$.

3.4 Algoritmo TTCA

Veremos um algoritmo que encontra uma alocação estável para os negociantes e casas. Esse algoritmo recebe um conjunto N de negociantes, e suas ordens de preferência sobre as casas. O algoritmo chama-se *Top Trading Cycle Algorithm*, abreviado para TTCA.

Construa um grafo dirigido G, onde cada negociante em N é um vértice. Para cada negociante i em N, insira um arco do vértice i para o vértice j se a casa j for a casa preferida do negociante i dentre as casas de N. Como temos tanto $|N|$ vértices como arcos e o grau de saída de cada vértice é exatamente um, temos pelo menos um circuito em G, que pode ser de tamanho 1. Mais do que isso, cada circuito é disjunto dos outros, ou seja, os vértices que pertencem a um circuito não pertencem a nenhum outro circuito de G. Ademais, cada componente conexa possui exatamente um circuito.

Então, podemos definir qual casa deve ser atribuída a cada negociante (vértice) pertencente a cada circuito, fazendo o seguinte. Para cada circuito C em G, digamos de tamanho r, que é da forma $i_1 \to i_2 \to \cdots \to i_r$, ou seja, o negociante i_1 tem como casa preferida a casa do negociante i_2, e etc., basta atribuir a casa i_2 ao negociante i_1, e genericamente, para $x \geq 2$, atribuir a casa i_x ao negociante i_{x-1}. A casa i_1 deve ser atribuída ao negociante i_r. Após a atribuição de casas, basta removermos os negociantes que estão em cada circuito C do conjunto inicial N, e recomerçamos o algoritmo se $N \neq \emptyset$.

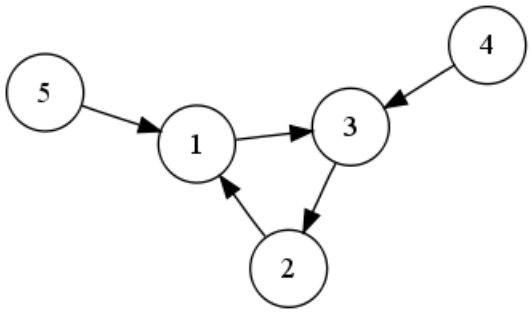
No pseudo-código abaixo, $A(v)$ denota a casa atribuída ao negociante v pelo algoritmo.

Algoritmo TTCA (N)
1 enquanto $N \neq \emptyset$ faça
2 $G \leftarrow (N, \emptyset)$
3 para todo $u \in N$ faça
4 seja v o negociante cuja casa é de maior preferência para u dentre os negociantes em N
5 $E(G) \leftarrow E(G) \cup \{uv\}$ // adicione o arco
6 para todo circuito C em G faça
7 para todo vértice $v \in C$ faça
8 seja u em N tal que a aresta $vu \in E(C)$
9 $A(v) \leftarrow u$
10 $N \leftarrow N \setminus V(C)$
11 devolva A

Veja um exemplo de simulação do algoritmo para uma instância com 5 negocintes que possuem as seguintes ordens de preferência:

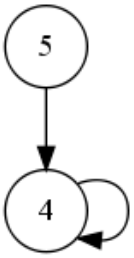
$$
\begin{array}{lccccccccc}
1: & 3 & \succ_1 & 4 & \succ_1 & 5 & \succ_1 & 2 & \succ_1 & 1. \\
2: & 1 & \succ_2 & 5 & \succ_2 & 4 & \succ_2 & 3 & \succ_2 & 2. \\
3: & 2 & \succ_3 & 1 & \succ_3 & 5 & \succ_3 & 4 & \succ_3 & 3. \\
4: & 3 & \succ_4 & 2 & \succ_4 & 1 & \succ_4 & 5 & \succ_4 & 4. \\
5: & 1 & \succ_5 & 3 & \succ_5 & 2 & \succ_5 & 4 & \succ_5 & 5.
\end{array}
$$

Figura 1: Primeira iteração.



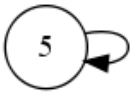
Nessa primeira iteração, o algoritmo aloca a casa 3 ao negociante 1, a casa 1 ao negociante 2, e a casa 2 ao negociante 3.

Figura 2: Segunda iteração.



Após alocar a casa 4 ao negociante 4, temos o grafo da última iteração do algoritmo.

Figura 3: Terceira iteração.



O algoritmo aloca a casa 5 ao negociante 5, e encontra a alocação $A = (3, 1, 2, 4, 5)$.

3.4.1 Correção do algoritmo

Vamos mostrar que o núcleo sempre consiste de uma única alocação: a devolvida pelo algoritmo acima. No algoritmo, cada negociante que recebe uma casa na primeira iteração recebe a sua casa favorita, ou seja, a casa que é a primeira em sua ordem de preferência. Chamemos de N_1 o conjunto de tais negociantes. Note que os negociantes de N_1 formariam um conjunto bloqueador caso chegássemos a uma alocação que não atribuísse tais casas a esses negociantes. Isso se deve ao fato de que em N_1 todos os negociantes recebem sua casa favorita em N, e também devido ao fato de que tais negociantes formam um ciclo no grafo do algoritmo e portanto tal alocação pertence a $T(N_1)$. Por isso, qualquer alocação que está no núcleo precisa atribuir as casas aos negociantes de N_1 da forma como o algoritmo faz.

Com isso, podemos usar o mesmo argumento para o conjunto de negociantes N_2 que recebem casas na segunda iteração. Cada negociante em N_2 recebe sua casa favorita excluindo as casas que já foram atribuídas aos negociantes em N_1. Portanto, se uma alocação está no núcleo, os negociantes de N_1 recebem as casas que o algoritmo atribue a eles e então os negociantes de N_2 precisam receber as mesmas casas que são atribuídas a eles pelo algoritmo, caso contrário N_2 formaria um conjunto bloqueador em qualquer alocação que atribui as casas de N_1 como na alocação do algoritmo.

Seja N_k o conjunto de negociantes para os quais o algoritmo atribui uma casa na sua k-ésima iteração. Uma prova por indução em k mostra que, em toda alocação do núcleo, o conjunto de negociantes N_k recebe exatamente as casas que lhe foram atribuídas pelo algoritmo na iteração k. Assim, ou o núcleo é vazio ou contém unicamente a alocação devolvida pelo algoritmo.

Resta provar que não existe nenhum conjunto bloqueador na alocação A devolvida pelo algoritmo. Suponha que existe um conjunto bloqueador S em A tal que S é minimal. Seja i o menor índice tal que $N_i \cap S \neq \emptyset$ e seja $p \in T(S)$ tal que $p_j \succ_j A_j$ ou $p_j = A_j$, com pelo menos uma ocorrência da primeira. As casas que cada negociante j de $N_i \cap S$ gosta mais que a casa A_j estão com negociantes de $\cup_{r=1}^{i-1} N_r$.

Como $S \cap N_r = \emptyset$ para $r = 1, \ldots, i-1$, e $p_j \in S \cap N_i$ tal que $p_j \succeq A(j)$, temos $p_j = A_j$. Então o conjunto $S \setminus N_i$ teria que ser um conjunto bloqueador, já que S era bloqueador, o que contraria a minimalidade de S.

Portanto a alocação devolvida pelo algoritmo não possui um conjunto bloqueador, e então o núcleo consiste apenas da alocação devolvida pelo algoritmo, como queríamos mostrar.

3.4.2 Prova de estratégia

Para aplicar esse algoritmo a um conjunto de negociantes, precisamos saber as preferências dos negociantes sobre as casas. Mas existe algum motivo para que os negociantes declararem suas reais preferências? Para isso, vamos olhar melhor para o algoritmo TTCA e analisar se existe motivo ou não.

Considere uma matriz M de ordens de preferência e A a alocação devolvida pelo algoritmo aplicado às ordens de preferência de M. Suponha que o negociante $j \in N_k$ para algum k não declare suas reais preferências, para tentar conseguir uma casa melhor do que A_j ao final do algoritmo. Teremos então uma matriz M' de ordens de preferência, e uma alocação A' devolvida pelo algoritmo quando aplicado à matriz M'. Se o algoritmo não for à prova de estratégia então teremos que $A'_j \succ_j A_j$. Temos que $A_i = A'_i$ para todo $i \in \cup_{r=1}^{k-1} N_r$. Mas então A'_j está em $N \setminus \{\cup_{r=1}^{k-1} N_r\}$, e como o algoritmo escolhe A_i como sendo a melhor casa para o negociante i em $N \setminus \{\cup_{r=1}^{k-1} N_r\}$ não vale que $A'_j \succ_j A_j$. Ou seja, o negociante j não se beneficia ao deixar de declarar a sua real ordem de preferência.

Portanto o TTCA é à prova de estratégia.

3.5 Extensões do problema

Uma modificação que pode ser feita no problema é permitir que as ordens de preferência não sejam estritas, ou seja, um negociante pode gostar tanto de uma casa quanto gosta de outra. Com essa modificação temos que alterar algumas propriedades do problema. Agora o que era um conjunto bloqueador será um conjunto bloqueador *fraco* e o que era núcleo será um núcleo *estrito*. Uma alteração importante é que agora um conjunto bloqueador fraco ocorre quando todas as pessoas do conjunto podem trocar de casas entre si de forma que todos consigam casas

estritamente melhores em suas ordens de preferência. Então um núcleo estrito é o conjunto de todas as alocações de casas que não possuem um conjunto bloqueador fraco.

Com preferências estritas, qualquer conjunto bloqueador fraco é também um conjunto bloqueador. Por isso quando tratamos do problema com preferências estritas o núcleo e o núcleo estrito coincidem. Já com ordens de preferência não estritas, o núcleo e o núcleo estrito podem ser diferentes. Nesse caso, o núcleo estrito pode ser vazio ou até ter mais do que uma possível alocação. A verificação de existência de uma solução para essa variante pode ser feita em tempo polinomial no número de negociantes.

Uma outra extensão do problema é atribuir aos negociantes mais de um tipo de bem indivisível. Por exemplo, os bens podem ser casas e carros. O problema fica mais complexo e pode ter o núcleo vazio até mesmo para restrições bem específicas, como independência de preferências entre os carros e as casas.

4 Leilões Combinatórios

No problema dos *Leilões Combinatórios* temos os objetos que estão sendo leiloados e os compradores. Os objetos são vendidos todos em um mesmo leilão, ao mesmo tempo. Cada comprador possui preferências sobre conjuntos de objetos, estabelecendo um certo valor que acha ideal para cada conjunto de objetos, e não apenas para cada objeto pois os objetos têm dependências entre si. O objetivo neste problema é encontrar uma forma de definir qual comprador ficará com cada objeto ao fim do leilão, de forma a maximizar a soma do valor ideal dado por cada comprador ao conjunto de objetos que receberá, que será mais adiante definido como bem estar social.

4.1 Notação e propriedades

A quantidade de compradores será denotada por n, e a quantidade de objetos por m. Cada comprador i expressa, para todo conjunto S de objetos, o valor por receber S com um número $v_i(S)$. Vamos chamar tal número de *valor ideal* do comprador i sobre o conjunto de objetos S. Esta valoração dos objetos deve possuir duas propriedades. A primeira é que o valor ideal $v_i(\emptyset) = 0$, para todo comprador i. A segunda é que, se temos dois conjuntos de objetos S e T tais que $S \subseteq T$, vale que $v_i(S) \leq v_i(T)$, para todo comprador i.

Devido à dependência entre os objetos, para um conjunto S de objetos, não necessariamente temos que a soma dos valores ideais de cada objeto em S é igual ao valor ideal de S. Sendo assim, para dois conjuntos de objetos S e T tais que $S \cap T = \emptyset$, dizemos que S e T são *complementares* para um comprador i se $v_i(S \cup T) > v_i(S) + v_i(T)$, e dizemos que são *substitutos* se $v_i(S \cup T) < v_i(S) + v_i(T)$.

Uma *alocação* X dos objetos dentre os compradores é uma sequência $X_1, X_2, \ldots, X_n$, onde cada comprador i recebe o conjunto X_i de objetos, e $X_i \cap X_j = \emptyset$, para todo $i \neq j$. Chamaremos de *bem estar social* o valor $\sum_i v_i(X_i)$ obtido em uma alocação X. Como dito anteriormente, queremos encontrar uma alocação que maximize o bem estar social, considerada então uma alocação *ótima*.

Assumimos neste problema que a valoração de valores ideais v_i de cada comprador i é uma informação que não é acessível a um comprador $j \neq i$.

Vamos ver algumas dificuldades encontradas ao tentar resolver o problema. Observe que a valoração de cada comprador é potencialmente de tamanho exponencial no número de objetos, dado que ele deve valorar cada conjunto de objetos. Por isso, se quisermos trabalhar com grandes quantidades de objetos, é usual considerar valorações que tenham uma maneira compacta de serem representadas. Infelizmente, mesmo com essa restrição, existem muitos casos em que encontrar uma alocação que maximiza o bem estar social é um problema computacionalmente díficil. Além disso, idealmente gostaríamos que o leilão fosse à prova de estratégia. Veremos mais adiante como lidar com tais dificuldades.

4.2 Caso do comprador focado (*single-minded*)

Vamos tratar de um caso bem específico do problema, para o qual há uma representação compacta das valorações dos compradores, ou seja, que requer espaço polinomial em m e n. Neste caso especial, vamos assumir que cada comprador está interessado apenas em um conjunto específico de objetos, estabelecendo valor ideal zero para qualquer conjunto de objetos que não contém o conjunto de objetos em que o comprador está interessado.

Uma valoração v é *focada* (*single-minded*) se existe um conjunto de objetos S^* e um valor v^* tal que $v(S) = v^*$ para todo $S \supseteq S^*$, e $v(S) = 0$ caso contrário. Uma *n-proposta focada* é um

par (S^*, v^*), onde $S^* = (S_1^*, \ldots, S_n^*)$, $v^* = (v_1^*, \ldots, v_n^*)$ e cada comprador i está interessado no conjunto S_i^* de objetos com valor ideal v_i^*. Ou seja, a valoração do comprador i é focada (em S_i^*).

Considere então o problema do leilão combinatório com compradores focados onde, dados n e m e uma n-proposta focada (S^*, v^*), queremos encontrar uma alocação dos objetos que maximize o bem estar social. Claramente, uma alocação ótima deve atribuir a um comprador i exatamente o conjunto S_i^* no qual o comprador i está interessado, ou não atribuir objeto nenhum ao comprador i.

Assim, a instância para um algoritmo que resolve o problema do leilão combinatório com compradores focados consiste em m, n e a n-proposta focada (S^*, v^*). Dada essa instância, o algoritmo pode devolver um conjunto W de compradores ganhadores, que representa uma alocação para o problema, onde para cada comprador i em W é atribuído o conjunto S_i^* de objetos, e para os compradores fora de W nenhum objeto é atribuído. Além disso, para todo par de compradores i e j onde $i \neq j$, deve valer que os conjuntos de objetos S_i^* e S_j^* são disjuntos, e tal alocação deve maximar o bem estar social $\sum_{i \in W} v_i^*$.

Teorema 4.1 *O problema do leilão combinatório com compradores focados é NP-difícil.*

Prova. Vamos mostrar que é possível reduzir o problema do *conjunto independente* a uma versão de decisão do problema do leilão combinatório com compradores focados.

Considere o problema de decisão do leilão combinatório com compradores focados, onde adicionamos um valor k à instância, e o algoritmo deve decidir se existe uma alocação que tenha bem estar social com valor pelo menos k ou não.

Um *conjunto independente* em um grafo G é um conjunto de vértices de G que não contém dois vértices adjacentes. O problema do conjunto independente, que é NP-completo [4], é o seguinte: dado um grafo G e um número k, decidir se G possui um conjunto independente de tamanho k.

Então, dada uma instância (G, k) do problema do conjunto independente, podemos construir uma instância do problema do leilão combinatório com compradores focados da seguinte maneira:

- O conjunto de objetos é o conjunto de arestas $E(G)$.
- O conjunto de compradores é o conjunto de vértices $V(G)$.
- Para cada i em $V(G)$, o conjunto de objetos S_i^* do comprador i é o conjunto das arestas de G incidentes em i e $v_i^* = 1$.

Vamos mostrar que existe um conjunto independente de tamanho k em G se e somente se existe uma alocação dos objetos com bem estar social pelo menos k na instância construída.

Seja I um conjunto independente em G de tamanho k. Segue que o conjunto I de compradores representa uma alocação com bem estar social k. De fato, como I é independente, I não contém vértices adjacentes, e portanto cada aresta que aparece em um conjunto S_i^* para i em I não aparece em nenhum outro conjunto S_j^* com j em I e $j \neq i$. Ou seja, os compradores representados pelos vértices em I estão interessados em conjuntos de objetos disjuntos dois a dois. Como cada comprador i tem valor ideal $v_i^* = 1$, o bem estar social do conjunto de compradores representado por I é exatamente $|I| = k$.

Seja W um conjunto de compradores ganhadores representando uma alocação cujo bem estar social é $\sum_{i \in W} v_i^* \geq k$. Segue que o conjunto W de vértices no grafo é um conjunto independente

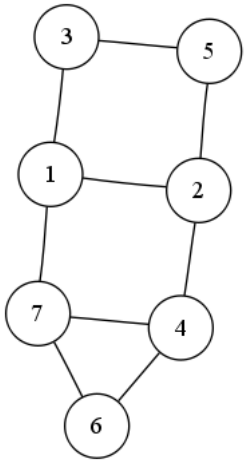
de tamanho pelo menos k. De fato, como os compradores em W estão interessados em conjuntos disjuntos de objetos, senão não teríamos uma alocação, não existe aresta entre tais compradores no grafo e, portanto, o conjunto W de vértices é independente em G. Além disso, como o valor ideal v_i^* de cada comprador i é exatamente 1, e o bem estar social da alocação dos objetos para o conjunto de compradores W é pelo menos k, temos que $|W| \geq k$.

Assim, chegamos à conclusão de que a versão de decisão do problema do leilão combinatório com compradores focados e valores ideais unitários é tão díficil quanto o problema do conjunto independente. Portanto, o problema original é NP-díficil, mesmo quando cada objeto está em apenas dois conjuntos S_i^*, pela construção da instância.

□

4.2.1 Exemplo

Considere o grafo a seguir como uma instância para o problema do conjunto independente:



A partir desta instância vamos construir uma instância para o problema em que estamos interessados, que é o problema do leilão combinatório com compradores focados e valores ideais unitários. Teremos 7 compradores e 9 objetos, onde cada comprador é um vértice, e cada objeto é uma aresta. Assim, o conjunto de objetos em que cada comprador está interessado é: $S_1^* = \{(1,2),(1,3),(1,7)\}$, $S_2^* = \{(1,2),(2,5),(2,4)\}$, $S_3^* = \{(1,3),(3,5)\}$, $S_4^* = \{(2,4),(4,6),(4,7)\}$, $S_5^* = \{(2,5),(3,5)\}$, $S_6^* = \{(4,6),(6,7)\}$, $S_7^* = \{(1,7),(4,7),(6,7)\}$.

Podemos ver que o conjunto {1,4,5} de vértices é um conjunto independente máximo no grafo, e os compradores que representam tais vértices na instância construída constituem uma alocação ótima.

Dado que o problema é NP-difícil, temos pelo menos três formas de encará-lo: estudar casos especiais, trabalhar em um algoritmo de aproximação para o problema, ou aplicar heurísticas.

4.2.2 Aproximação para o problema

Uma alocação A é dita uma α-aproximação para o problema se para toda alocação B, e especificamente para a alocação que maximiza o bem estar social, temos $\dfrac{\sum_{b\in B} v_b^*}{\sum_{a\in A} v_a^*} \leq \alpha$.

Existem resultados que mostram que encontrar uma $n^{1-\epsilon}$-aproximação para o problema do conjunto independente máximo em um grafo dado é também um problema computacionalmente difícil, para todo $\epsilon > 0$, onde n é o número de vértices do grafo. Assim, pela redução mostrada

acima, onde o bem estar social é exatamente igual ao tamanho de um conjunto independente, segue o resultado de que encontrar tal aproximação para o problema do leilão combinatório é também computacionalmente difícil. (Diz-se que essa redução preserva a razão de aproximação.) Mas sabemos que $m \leq n^2$, onde m é o número de arestas do grafo. Como podemos reduzir tal instância a uma instância do problema do leilão combinatório com compradores focados com m objetos, encontrar uma $m^{1/2-\epsilon}$-aproximação para o problema do leilão combinatório com compradores focados é também computacionalmente difícil.

4.2.3 Casos especiais

Ainda considerando os compradores focados, vamos ver como resolver alguns casos ainda mais específicos. Um desses casos é quando o conjunto desejado por cada comprador possui no máximo dois objetos. Com essa restrição, recaímos no problema do emparelhamento de peso máximo em um grafo arbitrário (não necessariamente bipartido). Para tal problema se conhece algoritmo eficiente [6].

Outro caso especial é quando, ao ordenar os objetos de alguma forma, temos que os objetos do conjunto desejado por cada comprador está arranjado de forma contígua nesta ordenação dos objetos. Ou seja, seja $\{1, 2, \ldots, m\}$ o conjunto de objetos, e suponha que o conjunto de objetos desejado pelo comprador i seja da forma $S_i^* = \{a^i, a^i + 1, a^i + 2, \ldots, k^i\}$, com $1 \leq a^i \leq k^i \leq m$. Neste caso podemos resolver o problema usando programação dinâmica. De fato, seja $b(y)$ o bem estar social de uma alocação ótima do problema quando consideramos apenas os objetos $\{1,2,\ldots,y\}$ e os jogadores com propostas (v_i^*, S_i^*), em que $S_i^* \subseteq \{1, \ldots, y\}$. Vale a seguinte recorrência para $b(y)$:

$$b(y) = \begin{cases} 0, \text{ se } y = 0 \\ b(y-1), \text{ se } k^i \neq y \text{ para todo } i \\ \max\{b(y-1), \max\{b(a^h - 1) + v_h^* : 1 \leq h \leq n \text{ e } k^h = y\}\}. \end{cases}$$

Ou seja, sempre que temos um comprador i tal que $k^i = y$, pegamos o melhor resultado entre alocar os objetos $\{a^i, a^i + 1, \ldots, k^i\}$ a um tal comprador i ou não alocar.

4.2.4 Uso de heurísticas

O resultado mostrado sobre a dificuldade do problema apenas nos revela que um algoritmo não irá devolver uma resposta ótima para *todas* as instâncias em tempo polinomial, a menos que P = NP. Porém, o problema pode ser formulado como um problema de programação inteira, e portanto podemos usar programação linear e aplicar algumas heurísticas e métodos de programação inteira chegando a resultados muito próximos do valor ótimo para muitas instâncias com até mesmo mais de mil objetos [7].

4.3 Um algoritmo de aproximação

Estamos ainda considerando compradores focados. Como já vimos, a existência de uma $m^{1/2-\epsilon}$-aproximação para o problema é possível apenas se P = NP. Vamos descrever uma $m^{1/2}$-aproximação para o problema, que na maioria dos casos é melhor do que isso. O algoritmo guloso a seguir deve-se a Lehmann et al.[5].

Algoritmo LCS-SM (S, v, n)
1 Ordene$(S, v, n) : v_1/\sqrt{|S_1|} \geq v_2/\sqrt{|S_2|} \geq \ldots \geq v_n/\sqrt{|S_n|}$

```
2   W ← ∅          U ← ∅
3   para i ← 1 até n faça
4       se S_i ∩ U = ∅
5           então W ← W ∪ {i}
6                 U ← U ∪ S_i
7   devolva W
```

Seja $m = |\cup_{i=1}^{n} S_i|$. Vamos mostrar que a alocação representada por W é uma $m^{1/2}$-aproximação para o problema, ou seja, considerando um conjunto OPT de compradores ganhadores que maximiza o bem estar social, temos que $\sum_{i \in OPT} v_i \leq \sqrt{m} \sum_{j \in W} v_j$. Um detalhe importante é que tal aproximação só é válida quando as propostas feitas pelos compradores representam seus reais interesses, o que será discutido logo adiante.

Teorema 4.2 *LCS-SM(S, v, n) é uma $m^{1/2}$-aproximação para o problema, onde $m = |\cup_{i=1}^{n} S_i|$.*

Prova. Considere a numeração dos compradores após a ordenação da linha 1 do algoritmo, onde podemos dizer que os compradores estão ordenados por ordem de prioridade. Para todo comprador i em W, seja $OPT_i = \{j \in OPT : j \geq i \text{ e } S_i \cap S_j \neq \emptyset\}$, ou seja, o conjunto de compradores que estão em OPT e que não estão em W, exceto pelo próprio i, justamente porque o comprador i ganhou o leilão, impedindo tais compradores de OPT de pertencerem ao conjunto W devolvido pelo algoritmo.

Observe que $OPT = \bigcup_{i \in W} OPT_i$. (Mas os OPT_i não são necessariamente disjuntos.) Como todo j em OPT_i é tal que $j \geq i$, vale que:

$$\frac{v_j}{\sqrt{|S_j|}} \leq \frac{v_i}{\sqrt{|S_i|}}.$$

Disso, somando para todo j em OPT_i, concluímos que:

$$\sum_{j \in OPT_i} v_j \leq \sum_{j \in OPT_i} \frac{v_i}{\sqrt{|S_i|}} \sqrt{|S_j|} = \frac{v_i}{\sqrt{|S_i|}} \sum_{j \in OPT_i} \sqrt{|S_j|}. \tag{1}$$

A desigualdade de Cauchy-Schwarz diz que:

$$\Big(\sum_{j=1}^{n} a_j b_j\Big)^2 \leq \Big(\sum_{j=1}^{n} a_j^2\Big)\Big(\sum_{j=1}^{n} b_j^2\Big)$$

para quaisquer números $a_1, \ldots, a_n, b_1, \ldots, b_n$.

Usando a desigualdade de Cauchy-Schwarz com $a_j = \sqrt{|S_j|}$ e $b_j = 1$ para todo j em OPT_i, e aplicando raiz quadrada dos dois lados, temos que:

$$\sum_{j \in OPT_i} \sqrt{|S_j|} \;\leq\; \sqrt{|OPT_i|} \sqrt{\sum_{j \in OPT_i} |S_j|}. \tag{2}$$

Todo S_j para j em OPT_i intersecta S_i, pela definição de OPT_i. Como OPT é uma alocação, essas intersecções são todas disjuntas, e então $|OPT_i| \leq |S_i|$. Trivialmente, pelo fato de OPT ser uma alocação, também vale que $\sum_{j \in OPT_i} |S_j| \leq m$. Então, substituindo essas duas

desigualdades em (2), temos $\sum_{j \in OPT_i} \sqrt{|S_j|} \leq \sqrt{|S_i|}\sqrt{m}$. Substituindo essa última desigualdade em (1), chegamos em $\sum_{j \in OPT_i} v_j \leq v_i\sqrt{m}$. Assim, como isso vale para todo comprador i em W e como $OPT = \cup_{i \in W} OPT_i$, chegamos ao que queríamos provar:

$$\sum_{k \in OPT} v_k \leq \sum_{i \in W} \sum_{j \in OPT_i} v_j \leq \sqrt{m} \sum_{i \in W} v_i.$$

□

4.3.1 Prova de estratégia

Como dito anteriormente, a aproximação só é válida se considerarmos que a proposta de cada comprador representa seus reais interesses, com seu valor ideal e o conjunto de objetos em que está realmente interessado. Mas como estamos considerando o fato de que o valor ideal que cada comprador estabelece para os objetos não é acessível, temos que tentar garantir que os compradores não ganhem nada se fizerem propostas com um valor v^* diferente de seu valor ideal ou propostas com um conjunto S^* diferente do conjunto de objetos em que estão interessados, para satisfazermos um leilão à prova de estratégia e garantir um bom resultado devolvido pelo algoritmo acima.

Para isso, vamos estabelecer que cada comprador ganhador terá que pagar pelo conjunto que recebe. Denotemos por S_i o conjunto de objetos em que o comprador i está interessado e v_i seu valor ideal, e denotemos por (v_i^*, S_i^*) a proposta declarada pelo comprador i. Denotemos por $p(i)$ o valor pago por cada comprador i ao fim do leilão, e considere o ganho do comprador i como $g(i) = v_i(S_i^*) - p(i)$, se i recebe o conjunto de objetos S_i^*, e $g(i) = 0$ caso contrário. Cada comprador quer maximar seu ganho, então uma maneira de garantir que o leilão seja à prova de estratégia é definir o valor $p(i)$, pago por cada comprador i do conjunto W devolvido pelo algoritmo, da seguinte maneira:

- Para cada i em W, o valor que i paga pelo conjunto de objetos S_i^* é $p(i) = v_j^*/\sqrt{|S_j^*|/|S_i^*|}$, onde j é o menor índice distinto de i tal que $S_i^* \cap S_j^* \neq \emptyset$. Se não existir tal j, então o comprador i não paga nada pelo conjunto S_i^*.

Assim, podemos mostrar que um comprador i não se beneficia ao declarar um valor v_i^* diferente de seu valor ideal v_i, e também podemos mostrar que o comprador i não se beneficia ao declarar um conjunto S_i^* diferente do conjunto S_i de objetos em que ele está interessado. Então, vamos mostrar a seguir que a proposta que um comprador i deve fazer ao leilão é (v_i, S_i), qualquer que seja a proposta que seja declarada pelos outros compradores.

Primeiramente, vamos pensar no caso em que a proposta (v_i, S_i) é uma proposta perdedora, ou seja, o comprador i não pertence ao conjunto W devolvido pelo algoritmo quando propõe (v_i, S_i). Assim, para todo conjunto S_i' de objetos e para todo valor v_i' tal que $v_i'/\sqrt{|S_i'|} \leq v_i/\sqrt{|S_i|}$, a proposta (v_i', S_i') continua sendo perdedora, pois o comprador i tenderia a perder prioridade em ganhar o leilão, devido à ordenação feita pelo algoritmo. Então, para tentar ganhar prioridade, ou seja, receber um índice menor durante a ordenação feita pelo algoritmo, o comprador i pode fazer uma proposta (v_i', S_i'), com $v_i'/\sqrt{|S_i'|} > v_i/\sqrt{|S_i|}$. Porém, para todo conjunto de objetos $S_i' \not\supseteq S_i$, o comprador tem ganho $g(i) \leq 0$, pois o valor ideal $v_i(S_i') = 0$. Além disso, se $S_i' \supsetneq S_i$, a proposta (v_i', S_i) teria prioridade pelo menos tão boa quanto (v_i', S_i') e uma chance maior de ganhar o leilão pois S_i intersecta com menos conjuntos de objetos do que S_i'. Por isso podemos considerar que $S_i' = S_i$. Também, para todo valor $v_i' > v_i$, o comprador i irá pagar um valor $p(i)$ maior do que o valor ideal $v_i(S_i)$, caso ganhe o leilão. De

fato, se j é o comprador com menor índice tal que $S_j^* \cap S_i \neq \emptyset$, temos que $j < i$ (pois i não ganhou o leilão com a proposta (v_i, S_i)). Assim $p(i) = v_j^*/\sqrt{|S_j^*|/|S_i|}$, e como $j < i$, vale que $v_j^*/\sqrt{|S_j^*|} \geq v_i'/\sqrt{|S_i|}$. Segue que $p(i) \geq v_i'$, e se i ganhar o leilão com a proposta (S_i, v_i'), então i terá um ganho $g(i) < 0$, já que $v_i' > v_i$ e $g(i) = v_i - p(i)$. Portanto, um comprador perdedor não consegue se beneficar ao declarar uma proposta que não seja sua proposta verdadeira (v_i, S_i).

Resta mostrar que um comprador i com uma proposta verdadeira (v_i, S_i) ganhadora também não se beneficia ao declarar uma proposta falsa. Se o comprador i declarar uma proposta (v_i', S_i') com $S_i' \not\supseteq S_i$, o comprador terá ganho não positivo, pois $v_i(S_i') = 0$. Por outro lado, se $S_i' \supseteq S_i$ então o comprador pode vir a perder o leilão, pois pode perder prioridade na ordenação feita pelo algoritmo, já que $|S_i'| \geq |S_i|$, ou pode deixar de ser ganhador pois S_i' pode intersectar o conjunto S_j^* de um comprador j previamente incluído em W, logo isso não é lucrativo para o comprador. Do contrário, se ele continua ganhador com a proposta (v_i', S_i'), então o comprador i irá pagar um valor $p'(i)$ pelo conjunto S_i' maior ou igual que o valor $p(i)$ pago pelo conjunto S_i. De fato, seja j' o comprador com menor índice estabelecido pelo algoritmo tal que $S_{j'}^* \cap S_i' \neq \emptyset$, e j o comprador com menor índice estabelecido pelo algoritmo tal que $S_j^* \cap S_i \neq \emptyset$. Temos que o comprador $j' \leq j$, pois $S_i' \supseteq S_i$. Então $p'(i) = v_{j'}^*/\sqrt{|S_{j'}^*|/|S_{i'}|} \geq v_j^*/\sqrt{|S_j^*|/|S_i|} = p(i)$. Portanto, um comprador não pode se beneficiar ao declarar um conjunto de objetos S_i' diferente do conjunto verdadeiro S_i.

Suponha então que o comprador i irá fazer uma proposta falsa (v_i', S_i). Claramente, se $v_i' \geq v_i$, então o comprador i continua sendo um comprador ganhador mas o valor pago pelo comprador não se altera, não trazendo nenhum benefício ao comprador. (Porém, ao fazer a proposta, o comprador não sabia se sua proposta seria ganhadora ou perdedora, e como visto acima, não seria bom arriscar ter um ganho negativo.) Assim, a única opção que resta ao comprador i é declarar um valor $v_i' \leq v_i$. Porém, como o valor pago pelo comprador i não depende do valor v_i' declarado, caso o comprador continue sendo um comprador ganhador, ele irá continuar pagando o mesmo valor que pagaria ao declarar o valor v_i. Mas, por outro lado, o comprador i pode vir a perder o leilão ao diminuir o valor de sua proposta, já que pode diminuir a sua prioridade, tendo então $g(i) = 0$. Assim, um comprador ganhador não se benefica ao declarar um valor diferente de seu valor ideal.

Portanto, definindo o valor pago por cada comprador da maneira como foi feito, conseguimos garantir um leilão à prova de estratégia, e satisfazer a aproximação do algoritmo descrito.

5 Leilões para publicidade associada à busca

Em sites com mecanismos de busca, costumamos encontrar publicidade em alguns locais da página, com assuntos relacionados à busca feita por um usuário. Esses locais da página são pré-determinados, e portanto, limitados.

Aggarwal, Muthukrishnan, Pál e Pál [1] consideraram o seguinte problema: encontrar uma maneira de alocar os locais da página (*itens*) aos publicitários (*compradores*) de forma a maximizar a satisfação dos publicitários (que será definida como *utilidade*), usando os conceitos de *alocação estável* e *alocação ótima*. Para isso, cada publicitário deve fazer uma proposta ao dono da página (*vendedor*) com o valor que pretende pagar por cada local da página. O mecanismo proposto por estes autores para encontrar tal alocação nos permite especificar valores mínimos e máximos diferentes que cada publicitário poderá pagar por cada local da página, onde o valor mínimo é definido pelo próprio dono da página, e o valor máximo pelo publicitário. Passamos a descrever tal mecanismo.

5.1 Notação

Denotamos por I o conjunto $\{1, 2, \ldots, n\}$ de compradores, e por J o conjunto $\{1, 2, \ldots, k\}$ de itens disponíveis. Cada comprador i tem um valor $v_{i,j}$ para cada item j de J que expressa o valor que o item j tem para o comprador i, e um valor máximo $m_{i,j}$ que é o valor máximo que o comprador i pretende pagar pelo item j. Além disso, o vendedor pode especificar, para cada item j de J, um valor mínimo de venda para cada comprador i, denotado por $r_{i,j}$.

Assim, podemos assumir que $r_{i,j} \geq 0$, $v_{i,j} \geq 0$, e $m_{i,j} \leq v_{i,j}$ (o valor máximo que um comprador i pretende pagar por um item j não pode exceder o valor que o item j tem para o comprador i). Caso um comprador i esteja interessado em um item j, então $m_{i,j} \geq r_{i,j}$, caso contrário o comprador i declara um valor $m_{i,j} < 0$.

As matrizes v, m e r de dimensões n por k são as informações necessárias para realizar a alocação de itens, sendo então (v, m, r, n, k) uma instância para o leilão, ou simplesmente um leilão (v, m, r, n, k).

O *grafo do leilão* de (v, m, r, n, k) é um grafo dirigido com pesos e bipartido em $(I, J \cup \{j_0\})$, onde j_0 é um item manequim. Existem cinco tipos de arcos nesse grafo. Para cada comprador i de I e para cada item j de J existe

- um arco *à frente* de i para j com peso $-v_{i,j}$,
- um arco *inverso* de j para i com peso $v_{i,j}$,
- um arco *de preço mínimo* de i para j com peso $r_{i,j} - v_{i,j}$,
- um arco *de preço máximo* de i para j com peso $m_{i,j} - v_{i,j}$,
- um arco *terminal* de i para j_0 com peso 0.

Um *passeio alternante* no grafo do leilão começa em um vértice i, segue alternando arcos à frente e arcos inversos, e termina em um arco de preço mínimo, preço máximo, ou arco terminal. Além disso, todos os vértices do passeio devem ser distintos, exceto pelo útimo vértice, que pode se repetir uma única vez.

5.2 Alocação estável e alocação ótima

Uma *alocação* consiste de cinco variáveis (u, p, μ, n, k), onde u é o vetor $(u_1, u_2, \ldots, u_n)$ de utilidades dos compradores, p é o vetor $(p_1, p_2, \ldots, p_k)$ de preços dos itens, $\mu \subseteq I \times J$ é o conjunto de pares (i, j), onde i é um comprador e j é um item, que indica que o item j foi alocado ao comprador i, n é o número de compradores, e k é o número de itens. Nenhum comprador e nenhum item ocorre mais de uma vez no conjunto μ, e compradores e itens que não estão em μ são ditos não alocados. Denotamos por $\mu(i)$ o item alocado ao comprador i, e por $\mu(j)$ o comprador alocado ao item j. Quando n e k estão claros no contexto, escrevemos apenas (u, p, μ) para a alocação (u, p, μ, n, k).

Uma alocação (u, p, μ) é *viável* para um leilão (v, m, r, n, k) se, para todo (i, j) de μ, temos:

$$r_{i,j} \leq p_j \leq m_{i,j},$$

$$u_i + p_j = v_{i,j},$$

e para cada comprador i não alocado, temos $u_i = 0$, e para cada item j não alocado, temos $p_j = 0$.

Uma alocação (u, p, μ) é *estável* para um leilão (v, m, r, n, k) se para todo $(i, j) \in I \times J$ pelo menos uma das seguintes desigualdades é válida:

$$u_i + p_j \geq v_{i,j},$$

$$p_j \geq m_{i,j},$$

$$u_i + r_{i,j} \geq v_{i,j}.$$

Um par $(i, j) \in I \times J$ que não satisfaz nenhuma das desigualdades é dito *par bloqueador*.

Uma alocação viável não é necessariamente estável, e uma alocação estável não é necessariamente viável. O que procuramos é uma alocação que seja viável e estável, e além disso, que seja o melhor possível em termos de utilidade. Uma alocação (u^*, p^*, μ^*) é *ótima* se para toda alocação viável e estável (u, p, μ) temos $u_i^* \geq u_i$, para todo comprador i de I.

5.3 Exemplo ilustrativo

Vamos considerar o seguinte leilão com $n = 6$ e $k = 3$.

$v_{i,j} =$

	1	2	3
1	44	67	11
2	77	73	45
3	87	16	66
4	88	84	94
5	14	9	18
6	6	88	14

$m_{i,j} =$

	1	2	3
1	15	50	4
2	21	53	23
3	42	9	8
4	83	2	18
5	12	2	5
6	6	2	3

$r_{i,j} =$

	1	2	3
1	2	10	3
2	18	6	22
3	31	3	5
4	53	1	9
5	9	1	3
6	3	1	2

A solução (u, p, μ) para esse leilão é:

- $\mu = \{(1, 2), (2, 3), (4, 1)\}$.
- $p_1 = 53$, $p_2 = 50$, $p_3 = 22$.
- $u_1 = 17$, $u_2 = 23$, $u_3 = 0$, $u_4 = 35$, $u_5 = 0$, $u_6 = 0$.

A noção de viabilidade é muito clara. Já a noção de estabilidade merece certa atenção. Vamos elaborar sobre a necessidade de cada uma das desigualdades que definem a estabilidade.

Veja o caso do par $(1, 2)$. Suponha que o comprador 1 tentasse pagar um preço menor pelo item 2. Assim, em vez de pagar $p_2 = 50$, teríamos $p_2 = 49$, aumentando sua utilidade para $u_1 = 18$. Caso a alocação (u, p, μ) não fosse ótima, isso seria possível. Porém, note que ao tentar "melhorar" a alocação, chegamos a uma alocação que não é estável, já que para $i = 2$ e $j = 2$ nenhuma das desigualdades que definem a estabilidade são válidas. Isso ocorre pois caso o comprador 2 pagasse $p_2 = 49 + \epsilon \leq m_{2,2} = 53$ pelo item 2, ele teria utilidade $u_2 = v_{2,2} - (49 + \epsilon) = 24 - \epsilon$, melhorando sua utilidade. Nesse caso, a desigualdade que passaria a valer é $u_2 + p_2 \geq v_{2,2}$. Ou seja, enquanto o preço de item j é pequeno ($p_j < m_{i,j}$) para um comprador i, este comprador pode melhorar sua utilidade caso $u_i + p_j < v_{i,j}$ (e também caso $u_i + r_{i,j} < v_{i,j}$, que será explicado no próximo parágrafo). Se um comprador i pode melhorar sua utilidade dessa maneira, comprando o item j, então a alocação não é estável, já que nenhuma das desigualdes é valida para (i, j).

Na verdade, existem alguns casos a serem tratados, mas que não ocorrem no exemplo acima. Vamos considerar aqui que $u_i + p_j < v_{i,j}$ e $p_j < m_{i,j}$, e observar qual a influência da desigualdade $u_i + r_{i,j} \geq v_{i,j}$ para a estabilidade. Essa terceira desigualdade está relacionada ao preço mínimo de um item para um comprador. Veja que se temos $u_i + r_{i,j} \geq v_{i,j}$, significa que o preço mínimo ainda não foi atingido, e portanto o comprador não pode simplesmente pagar $p_j + \epsilon$, pois $p_j + \epsilon < r_{i,j}$. Nesse caso, a terceira desigualdade significa que, mesmo que o comprador pague o preço mínimo $r_{i,j}$ do item, sua utilidade não iria subir. Ou seja, por enquanto ele está estável com relação àquele item. Por outro lado, se $u_i + r_{i,j} < v_{i,j}$ e $p_j + \epsilon < r_{i,j}$, o comprador também não poderia simplesmente aumentar de ϵ o preço do item para se beneficiar, como ocorreu no exemplo acima. Mas, neste caso, o comprador simplesmente aumenta sua utilidade pagando o preço $r_{i,j}$, passando a ficar estável com relação ao item.

Assim, conseguimos verificar que, quando as três desigualdades não são válidas para um par (i, j), é porque o comprador i pode melhorar sua utilidade pagando um preço $p_j + \epsilon$ ou $r_{i,j}$ pelo item j, dependendo do caso. Por outro lado, conseguimos verificar também que, se pelo menos uma das desigualdades é satisfeita, então o comprador i não pode melhorar sua utilidade comprando o item j.

5.4 Posição geral

Uma definição importante para este problema é a *posição geral* de um leilão (v, m, r, n, k). Um leilão está em posição geral se, para todo comprador i, não existem dois passeios alternantes no grafo do leilão que começam no vértice i que tenham pesos iguais. Relembrando, temos os seguinte arcos o grafo do leilão.

- um arco *à frente* de i para j com peso $-v_{i,j}$,
- um arco *inverso* de j para i com peso $v_{i,j}$,
- um arco *de preço mínimo* de i para j com peso $r_{i,j} - v_{i,j}$,
- um arco *de preço máximo* de i para j com peso $m_{i,j} - v_{i,j}$,
- um arco *terminal* de i para j_0 com peso 0.

Veja que se, para um par (i, j), temos $r_{i,j} = v_{i,j}$, então o passeio alternante (i, j) usando o arco de preço mínimo terá o mesmo peso que o passeio alternante (i, j_0) usando o arco

terminal. Portanto, em um leilão em posição geral, não existe um par (i, j) tal que $r_{i,j} = v_{i,j}$. Analogamente, em um leilão em posição geral não existe um par (i, j) tal que $r_{i,j} = m_{i,j}$. Além disso, se existe um par (i, j) tal que $r_{i,j} = m_{i,j}$, então os dois passeios alternantes com um único arco de i para j possuem o mesmo peso, sendo que um deles usa o arco de preço mínimo, e o outro usa o arco de preço máximo. Portanto, em um leilão em posição geral temos $r_{i,j} \neq m_{i,j}$, $r_{i,j} \neq v_{i,j}$ e $m_{i,j} \neq v_{i,j}$. Em especial, para um comprador i interessado em um item j, vale que $r_{i,j} < m_{i,j} < v_{i,j}$.

Existem muitos outros casos que não deixam um leilão em posição geral. Por exemplo, se $v_{i_1,j} = v_{i_2,j}$ para dois compradores i_1 e i_2, então o leilão não está em posição geral pois os passeios alternantes (i_1, j, i_2, j_0) e (i_1, j_0) possuem o mesmo peso.

5.5 Algoritmo

Vamos apresentar um algoritmo que encontra uma alocação ótima para um leilão (v, m, r, n, k), e depois mostraremos que tal alocação é à prova de estratégia, é encontrada em tempo $O(kn^3)$, e é única se o leilão estiver em posição geral.

O algoritmo é conhecido como *StableMatch*. Inicialmente o algoritmo começa com uma alocação vazia $(u^{(0)}, p^{(0)}, \mu^{(0)})$, onde $\mu^{(0)} = \emptyset$, $p_j^{(0)} = 0$, e $u_i^{(0)} = B$, com $B = \max(v_{i,j}) + 1$ para cada $(i, j) \in I \times J$. Cada iteração t começa com uma alocação $(u^{(t)}, p^{(t)}, \mu^{(t)})$ e tenta construir uma alocação $(u^{(t+1)}, p^{(t+1)}, \mu^{(t+1)})$. O algoritmo para quando não conseguimos mais atualizar a alocação, e devolve a alocação $(u^{(T)}, p^{(T)}, \mu^{(T)})$ da última iteração.

Para conseguirmos descrever a atualização de uma alocação $(u^{(t)}, p^{(t)}, \mu^{(t)})$, vamos definir um *grafo de atualização* e um *passeio alternante* nesse grafo.

O chamado grafo de atualização de uma alocação (u, p, μ) é o multi-grafo dirigido com pesos e bipartido em $(I, J \cup \{j_0\})$, onde j_0 é um item manequim, I é o conjunto de compradores, e J é o conjunto de itens. O grafo de atualização possui cinco tipos de arcos. Para cada comprador i de I, e para cada item j de J, temos:

- um arco *à frente* de i para j com peso $u_i + p_j - v_{i,j}$, se $r_{i,j} \leq p_j < m_{i,j}$.
- um arco *inverso* de j para i com peso $v_{i,j} - u_i - p_j$, se $(i, j) \in \mu$.
- um arco *de preço mínimo* de i para j com peso $u_i + r_{i,j} - v_{i,j}$, se $u_i + r_{i,j} > v_{i,j}$ e $m_{i,j} > r_{i,j}$.
- um arco *de preço máximo* de i para j com peso $u_i + m_{i,j} - v_{i,j}$, se $u_i + m_{i,j} > v_{i,j}$ e $m_{i,j} > r_{i,j}$.
- um arco *terminal* de i para j_0 com peso u_i, se $u_i > 0$.

Observe que se (u, p, μ) for estável, então os arcos à frente de peso não-negativo.

Um *passeio alternante* no grafo de atualização começa em um vértice que representa um comprador não alocado i_0 com $u_{i_0} > 0$, segue alternando arcos à frente e arcos inversos, e termina com um arco de preço mínimo, preço máximo ou arco terminal. Todos os vértices de um passeio alternante devem ser distintos, exceto pelo último vértice, que pode se repetir uma vez. O peso $w(P)$ de um passeio alternante P é a soma dos pesos de seus arcos.

Então, considere uma alocação $(u^{(t)}, p^{(t)}, \mu^{(t)})$ e o grafo de atualização $G^{(t)}$ correspondente. Uma iteração do algoritmo *StableMatch* consiste dos seguintes passos:

1. Se não houver passeio alternante em $G^{(t)}$, pare e devolva a alocação $(u^{(t)}, p^{(t)}, \mu^{(t)})$. Caso contrário, considere um passeio alternante P de peso $w^{(t)}(P)$ mínimo, no grafo $G^{(t)}$. Digamos que $P = (i_0, j_1, i_1, j_2, i_2, \ldots, j_l, i_l, j_{l+1})$, para algum $l \geq 0$.

2. Seja $d^{(t)}(i_0, y)$ o comprimento do menor caminho em $G^{(t)}$ de i_0 a qualquer vértice y, usando apenas arcos à frente e inversos. Se um vértice y não é atingível a partir de i_0 por um caminho assim, então $d^{(t)}(i_0, y) = \infty$.

3. Atualize a utilidade dos compradores i de I, de acordo com o vetor $u^{(t+1)}$, que representa as utilidades ao fim da iteração t:

$$u_i^{(t+1)} = u_i^{(t)} - \max\Big(w^{(t)}(P) - d^{(t)}(i_0, i), 0\Big).$$

4. Atualize o preço de cada item j de J, de acordo com o vetor $p^{(t+1)}$, definido a partir de

$$p_j^{(t+)} = p_j^{(t)} + \max\Big(w^{(t)}(P) - d^{(t)}(i_0, j), 0\Big).$$

   Os preços $p_j^{(t+1)}$ valem $p_j^{(t+)}$ com uma única exceção. No caso em que o último arco de P é um arco de preço mínimo, o preço do item j_{l+1} é dado por $p_{j_{l+1}}^{(t+1)} = \max(p_{j_{l+1}}^{(t+)}, r_{i_l, j_{l+1}})$. Ou seja, o preço do item em questão não pode ser menor do que seu preço de reserva.

5. Atualize a alocação $\mu^{(t)}$ ao longo do caminho alternante P para obter a nova alocação $\mu^{(t+1)}$. Essa operação será explicada detalhadamente a seguir.

Para obtermos a nova alocação $\mu^{(t+1)}$, removemos todos pares (i, j) de $\mu^{(t)}$ representados pelos arcos inversos do passeio alternante P, e inserimos os pares (i, j) que são representados pelos outros arcos do passeio alternante P. Chamamos esta operação de *soma disjunta* de $\mu(t)$ e P e a denotamos por $\mu^{(t+1)} = (\mu^{(t)} \cup E_P) \setminus (\mu^{(t)} \cap E_P)$. A última aresta de P e o último vértice de P merecem no entanto certa atenção, durante essa operação. Por isso, vamos considerar três casos:

Caso 1: P termina com um arco terminal, ou seja, j_{l+1} é o item manequim. Nesse caso, apenas realizamos a soma disjunta de $\mu^{(t)}$ e P. Mas como o item j_{l+1} é manequim, temos que deixar o comprador i_l não alocado, e para $x = 0, 1, \ldots, l-1$ o comprador i_x fica com item j_{x+1}.

Caso 2: P termina com um arco de preço máximo. Consideramos dois subcasos:
Caso 2.1: $j_{l+1} = j_l$. Isso significa que o comprador i_l já estava alocado ao item j_l, e agora atingiu seu preço máximo para o item j_l. Portanto, não deve ficar com item j_l. Execute a operação de soma disjunta de $\mu^{(t)}$ e P, desconsiderando o arco de preço máximo de P. Assim, para $x = 0, 1, \ldots, l-1$ o comprador i_x fica alocado ao item j_{x+1}.
Caso 2.2: $j_{l+1} \neq j_l$. O comprador i_l atinge seu preço máximo em um item que não estava alocado a ele. Não modifique a alocação, ou seja, $\mu^{(t+1)} = \mu^{(t)}$.

Caso 3: P termina com um arco de preço mínimo. Esse é o caso mais complicado, consideramos três subcasos:
Caso 3.1: Se o item j_{l+1} não está alocado em $\mu^{(t)}$, então executamos a operação de soma disjunta de P e μ. Para $x = 0, 1, \ldots, l-1$ o comprador i_x fica com o item j_{x+1}.
Caso 3.2: Se o item j_{l+1} está alocado em $\mu^{(t)}$ e o preço mínimo $r_{i_l, j_{l+1}}$ oferecido pelo comprador i_l não excede o preço atual $p_{j_{l+1}}^{(t+)}$, não modifique a alocação, ou seja, $\mu^{(t+1)} = \mu^{(t)}$.
Caso 3.3: Senão, o item j_{l+1} está alocado em $\mu^{(t)}$ à um comprador i_a, e $r_{i_l, j_{l+1}} > p_{j_{l+1}}^{(t+)}$. Se P é um caminho (não um passeio), ou seja, se P não visita o vértice j_{l+1} duas vezes, apenas desalocamos o comprador i_a que estava alocado ao item j_{l+1}, e executamos a soma disjunta

de P e μ. Isso não altera o tamanho da alocação, já que o comprador i_0 agora está alocado, enquanto o comprador i_a não está mais alocado.
Caso P visite o vértice j_{l+1} duas vezes, então $j_{l+1} = j_d$, para algum $1 \leq d < l$. Veja que não podemos ter $d = l$, pois isto significaria que o comprador i_l estava alocado ao item j_l. Isto não ocorre porque, para que o comprador i_l esteja alocado ao item j_l, seria necessário que o arco de preço mínimo de i_l para j_l já tivesse aparecido em algum passeio alternante de peso mínimo anterior, contradizendo a existência deste arco de preço mínimo na iteração atual. (Na seção sobre a complexidade do algoritmo, mostraremos que o arco (i_l, j_{l+1}) de $G^{(t)}$ não existe em nenhum grafo $G^{(t')}$, com $t' > t$.) Dessa forma, temos um circuito C no final do passeio P com pelo menos dois compradores e dois itens. Executamos a soma disjunta de C e μ. Assim, para $x = d, d+1, \ldots, l$, o comprador i_x fica com o item j_{x+1}.

5.6 Exemplo passo a passo do algoritmo

Aqui vamos indicar algumas iterações do algoritmo funcionando para o exemplo já citado anteriormente, e mostrar como ocorrem as atualizações da alocação μ.

$v_{i,j} =$

	1	2	3
1	44	67	11
2	77	73	45
3	87	16	66
4	88	84	94
5	14	9	18
6	6	88	14

$m_{i,j} =$

	1	2	3
1	15	50	4
2	21	53	23
3	42	9	8
4	83	2	18
5	12	2	5
6	6	2	3

$r_{i,j} =$

	1	2	3
1	2	10	3
2	18	6	22
3	31	3	5
4	53	1	9
5	9	1	3
6	3	1	2

Começamos com a alocação $(u^{(0)}, p^{(0)}, \mu^{(0)})$:

- $\mu^{(0)} = \emptyset$
- $p_1^{(0)} = 0$, $p_2^{(0)} = 0$, $p_3^{(0)} = 0$
- $u_1^{(0)} = 95$, $u_2^{(0)} = 95$, $u_3^{(0)} = 95$, $u_4^{(0)} = 95$, $u_5^{(0)} = 95$, $u_6^{(0)} = 95$

Na primeira iteração encontramos o passeio alternante de peso mínimo $(i_6,\ j_2)$ que termina com um arco de preço mínimo. Caímos no caso 3.1 da atualização de μ, e obtemos $(u^{(1)}, p^{(1)}, \mu^{(1)})$:

- $\mu^{(1)} = \{(6, 2)\}$
- $p_1^{(1)} = 0$, $p_2^{(1)} = 1$, $p_3^{(1)} = 0$
- $u_1^{(1)} = 95$, $u_2^{(1)} = 95$, $u_3^{(1)} = 95$, $u_4^{(1)} = 95$, $u_5^{(1)} = 95$, $u_6^{(1)} = 87$

Na segunda iteração encontramos o passeio alternante de peso mínimo (i_4, j_3) que termina com um arco de preço mínimo e novamente caímos na Caso 3.1, obtendo $(u^{(2)}, p^{(2)}, \mu^{(2)})$:

- $\mu^{(2)} = \{(6, 2), (4, 3)\}$
- $p_1^{(2)} = 0$, $p_2^{(2)} = 1$, $p_3^{(2)} = 9$
- $u_1^{(2)} = 95$, $u_2^{(2)} = 95$, $u_3^{(2)} = 95$, $u_4^{(2)} = 85$, $u_5^{(2)} = 95$, $u_6^{(2)} = 87$

Na terceira iteração, o passeio alternante de peso mínimo encontrado é (i_2, j_2) terminando em um arco de preço mínimo. Mas nessa iteração caímos no caso 3.3, pois o ítem j_2 já está alocado ao comprador i_6, porém com um preço baixo para o comprador i_2, ou seja, $r_{i_2,j_2} > p^{(2+)}$. Assim, obtemos a alocação $(u^{(3)}, p^{(3)}, \mu^{(3)})$.

- $\mu^{(3)} = \{(2,2), (4,3)\}$
- $p_1^{(3)} = 0$, $p_2^{(3)} = 6$, $p_3^{(3)} = 9$
- $u_1^{(3)} = 95$, $u_2^{(3)} = 67$, $u_3^{(3)} = 95$, $u_4^{(3)} = 85$, $u_5^{(3)} = 95$, $u_6^{(3)} = 87$

Utilizando as atualizações de preço, utilidade e pares alocados da maneira descrita na seção anterior, obtemos a alocação:

- $\mu^{(T)} = \{(1,2), (2,3), (4,1)\}$.
- $p_1^{(T)} = 53$, $p_2^{(T)} = 50$, $p_3^{(T)} = 22$.
- $u_1^{(T)} = 17$, $u_2^{(T)} = 23$, $u_3^{(T)} = 0$, $u_4^{(T)} = 35$, $u_5^{(T)} = 0$, $u_6^{(T)} = 0$.

5.7 Correção do algoritmo

Para verificar que a alocação obtida ao fim do algoritmo é estável e viável, vamos demonstrar alguns invariantes do algoritmo, que valem ao início de cada iteração do algoritmo.

- Invariante $(A3)$: Todo item não alocado em $\mu^{(t)}$ tem preço $p^{(t)}$ zero.
- Invariante $(A2)$: Para todo par alocado $(i,j) \in \mu^{(t)}$ temos viabilidade, ou seja:

$$r_{i,j} \leq p_j^{(t)} \leq m_{i,j} \tag{1}$$
$$u_i^{(t)} + p_j^{(t)} = v_{i,j} \tag{2}$$

- Invariante $(A1)$: A alocação $(u^{(t)}, p^{(t)}, \mu^{(t)})$ é estável para o leilão (v, m, r), ou seja, pelo menos uma das seguintes desigualdades é válida:

$$u_i^{(t)} + p_j^{(t)} \geq v_{i,j} \tag{3}$$
$$p_j^{(t)} \geq m_{i,j} \tag{4}$$
$$u_i^{(t)} + r_{i,j} \geq v_{i,j} \tag{5}$$

  para todo $(i,j) \in I \times J$.

Antes de mostrar que tais invariantes são válidos, vamos verificar que esses invariantes garantem estabilidade e viabilidade ao fim do algoritmo. Seja $(u^{(T)}, p^{(T)}, \mu^{(T)})$ a alocação devolvida pelo algoritmo. Essa alocação é estável, pelo invariante $(A1)$. Pelo invariante $(A3)$, os itens não alocados têm preço zero. Ademais, a utilidade dos compradores não alocados em $\mu^{(T)}$ é zero, pois se existir um comprador não alocado $\mu^{(T)}$ com utilidade positiva, o caminho dele para o item manequim j_0 com um único arco é um caminho alternante em $G^{(t)}$, e portanto o algoritmo não teria terminado. Isso e o invariante $(A2)$ garantem que a alocação $(u^{(T)}, p^{(T)}, \mu^{(T)})$ é também viável.

Vamos agora às demostrações dos invariantes. Em seguida, mostraremos a otimalidade do algoritmo.

5.7.1 Prova dos invariantes $(A1)$, $(A2)$ e $(A3)$

A prova dos invariantes $(A1)$, $(A2)$ e $(A3)$ depende de dois invariantes extras do algoritmo:

- Invariante $(B1)$: Se um comprador i está interessado em um item j e $u_i^{(t)} + m_{i,j} = v_{i,j}$, então $(i,j) \notin \mu^{(t)}$.
- Invariante $(B2)$: Se um comprador i está interessado em um item j e $u_i^{(t)} + r_{i,j} = v_{i,j}$, então $(i,j) \in \mu^{(t)}$ ou $p_j^{(t)} \geq r_{i,j}$.

Vamos assumir que os invariantes $(B1)$ e $(B2)$ valem no ínicio de cada iteração.

Para $t = 0$, temos $u_i^{(0)} = B$, $p_j^{(0)} = 0$, $r_{i,j} \geq 0$, e $\mu^{(t)} = \emptyset$ para todo comprador i e item j, onde $B = (\max v_{i,j}) + 1$.

Então, para todo par (i,j), vale (5), satisfazendo o invariante $(A1)$. Trivialmente, os invariantes $(A2)$ e $(A3)$ também valem ao ínicio da primeira iteração, pois $\mu^{(0)} = \emptyset$, e $p_j^{(0)} = 0$ para todo ji em J.

Seja $t \geq 0$ e suponha, por indução, que a alocação $(u^{(t)}, p^{(t)}, \mu^{(t)})$ satisfaz os invariantes $(A1)$, $(A2)$ e $(A3)$. Suponha que t não é a última iteração do algoritmo. Vamos mostrar que a alocação $(u^{(t+1)}, p^{(t+1)}, \mu^{(t+1)})$ satisfaz $(A1)$, $(A2)$ e $(A3)$.

Prova do invariante $(A1)$.

Podemos dividir em três casos para mostrar que a alocação $(u^{(t+1)}, p^{(t+1)}, \mu^{(t+1)})$ satisfaz $(A1)$, e portanto é estável. Para cada $i \in I$ e $j \in J$, temos:

- *Caso 1:* $p_j^{(t)} \geq m_{i,j}$.
  A desigualdade (4) é válida, e continua sendo válida para $p_j^{(t+1)}$, pois $p_j^{(t+1)} \geq p_j^{(t)}$.
- *Caso 2:* $r_{i,j} \leq p_j^{(t)} < m_{i,j}$.
  Como $(u^{(t)}, p^{(t)}, \mu^{(t)})$ é estável, sabemos que (3) é válida, pois (4) não é válida e $p_j^{(t)} \geq r_{i,j}$, o que implica que se (5) é válida, (3) também é. Se $d^{(t)}(i_0, i) \geq w^{(t)}(P)$, então, como $u_i^{(t+1)} = u_i^{(t)}$ e $p_j^{(t+1)} \geq p_j^{(t)}$, temos que $u_i^{(t+1)} + p_j^{(t+1)} \geq u_i^{(t)} + p_j^{(t)} \geq v_{i,j}$.

  Por outro lado, se $d^{(t)}(i_0, i) < w^{(t)}(P)$, então

$$u_i^{(t+1)} = u_i^{(t)} - \big(w^{(t)}(P) - d^{(t)}(i_0, i)\big) \text{ e} \tag{6}$$

$$p_j^{(t+1)} \geq p_j^{(t+)} \geq p_j^{(t)} + \big(w^{(t)}(P) - d^{(t)}(i_0, j)\big). \tag{7}$$

  Como $r_{i,j} \leq p_j^{(t)} < m_{i,j}$, temos um arco à frente de i para j em $G^{(t)}$, e então

$$d^{(t)}(i_0, j) \leq d^{(t)}(i_0, i) + \big(u_i^{(t)} + p_j^{(t)} - v_{i,j}\big). \tag{8}$$

  Juntando (6), (7) e (8), deduzimos que

$$\begin{aligned} u_i^{(t+1)} + p_j^{(t+1)} &\geq \big(u_i^{(t)} - w^{(t)}(P) + d^{(t)}(i_0, i)\big) + \big(p_j^{(t)} + w^{(t)}(P) - d^{(t)}(i_0, j)\big) \\ &= u_i^{(t)} + d^{(t)}(i_0, i) + p_j^{(t)} - d^{(t)}(i_0, j) \\ &\geq u_i^{(t)} + d^{(t)}(i_0, i) + p_j^{(t)} - \big(d^{(t)}(i_0, i) + u_i^{(t)} - p_j^{(t)} - v_{i,j}\big) \\ &= v_{i,j}. \end{aligned}$$

  Portanto, $u_i^{(t+1)}$ e $p_j^{(t+1)}$ satisfazem (3).

- *Caso 3:* $p_j^{(t)} < r_{i,j}$.
  Vamos considerar apenas os casos em que $m_{i,j} \geq r_{i,j}$, pois caso contrário, o Caso 2 nos garante estabilidade. Logo (4) não vale.

  Como $(u^{(t)}, p^{(t)}, \mu^{(t)})$ é estável e não vale (4), então com certeza (5) é válida, pois (3) implica (5) neste caso. Se $d^{(t)}(i_0, i) \geq w^{(t)}(P)$, então $u_i^{(t+1)} = u_i^{(t)}$ e (5) continua válida com $u_i^{(t+1)}$ no lugar de $u_i^{(t)}$.

  Por outro lado, se $d^{(t)}(i_0, i) < w^{(t)}(P)$, temos

$$u_i^{(t+1)} = u_i^{(t)} - \big(w^{(t)}(P) - d^{(t)}(i_0, i)\big). \tag{9}$$

  Se existir um arco de preço mínimo de i para j em $G^{(t)}$, temos que

$$w^{(t)}(P) \leq d^{(t)}(i_0, i) + \big(u_i^{(t)} + r_{i,j} - v_{i,j}\big). \tag{10}$$

  Para mostrar que tal arco de preço mínimo existe, basta mostrar que $u_i^{(t)} + r_{i,j} > v_{i,j}$. Primeiro, note que, pelo invariante $(A2)$, $(i,j) \notin \mu^{(t)}$ pois $p_j^{(t)} < r_{i,j}$. Isso e o invariante $(B2)$ implicam que $u_i^{(t)} + r_{i,j} \neq v_{i,j}$. Como $u_i^{(t)}$ satisfaz (5), deduzimos que $u_i^{(t)} + r_{i,j} > v_{i,j}$, e o arco de preço mínimo de fato existe.

  Mas então, por (9) e (10) temos que

$$\begin{aligned} u_i^{(t+1)} &= u_i^{(t)} - w^{(t)}(P) + d^{(t)}(i_0, i) \\ &\geq u_i^{(t)} - \big(d^{(t)}(i_0, i) + u_i^{(t)} + r_{i,j} - v_{i,j}\big) + d^{(t)}(i_0, i) \\ &= v_{i,j} - r_{i,j}. \end{aligned}$$

  Concluindo que (5) continua válida com $u_i^{(t+1)}$ no lugar de $u_i^{(t)}$.

Com isso, temos a verificação do invariante $(A1)$.

Prova do invariante $(A2)$**.**

Vamos primeiro verificar que a alocação $(u^{(t+1)}, p^{(t+)}, \mu^{(t)})$ satisfaz $(A2)$. Considerando apenas os pares $(i,j) \in \mu^{(t)}$, sabemos que em $G^{(t)}$ existe um arco inverso de j para i de peso 0, pois $u_i^{(t)} + p_j^{(t)} = v_{i,j}$. Portanto

$$d^{(t)}(i_0, i) = d^{(t)}(i_0, j). \tag{11}$$

Assim, pelas atualizações dos passos 3 e 4 do algoritmo, vale que $u_i^{(t+1)} + p_j^{(t+)} = u_i^{(t)} + p_j^{(t)}$, mantendo a igualdade (2) para $(u^{(t+1)}, p^{(t+)}, \mu^{(t)})$. Para mostrar que (1) vale para $(u^{(t+1)}, p^{(t+)}, \mu^{(t)})$, consideramos dois casos. Se $w^{(t)}(P) \leq d^{(t)}(i_0, j)$, então $p_j^{(t+)} = p_j^{(t)}$, e (1) trivialmente vale para $(u^{(t+1)}, p^{(t+)}, \mu^{(t)})$. Caso contrário, $w^{(t)}(P) > d^{(t)}(i_0, j)$, e teremos a atualização de preço

$$p_j^{(t+)} = p_j^{(t)} + \big(w^{(t)}(P) - d^{(t)}(i_0, j)\big). \tag{12}$$

Se existir o arco de preço máximo de i para j, temos que

$$w^{(t)}(P) \leq d^{(t)}(i_0, i) + (u_i^{(t)} + m_{i,j} - v_{i,j}) \tag{13}$$

e conseguimos mostrar que (1) é válida para $(u^{(t+1)}, p^{(t+)}, \mu^{(t)})$. De fato, ao somar (11), (12), (13) e cancelar termos comuns, obtemos que $p^{(t+)} \leq (u_i^{(t)} + p_j^{(t)} - v_{i,j}) + m_{i,j}$. Como $(u_i^{(t)} + p_j^{(t)} - v_{i,j}) = 0$, vale que $p^{(t+)} \leq m_{i,j}$. Por outro lado, $p^{(t+)} \geq p^{(t)} \geq r_{i,j}$, e portanto $p^{(t+)}$ satisfaz (1)

Basta agora mostrar que o arco de preço máximo de i para j existe, e para isso precisamos verificar que $u_i^{(t)} + m_{i,j} > v_{i,j}$. Como $p^{(t)} \leq m_{i,j}$, temos que $u_i^{(t)} + m_{i,j} \geq u_i^{(t)} + p_j^{(t)} = v_{i,j}$, já que a alocação atual satisfaz $(A2)$. Além disso, podemos ver também que $u_i^{(t)} + m_{i,j} \neq v_{i,j}$, por $(B1)$, confirmando que o arco de preço máximo existe.

Agora vamos provar que o invariante $(A2)$ vale para a alocação $(u^{(t+1)}, p^{(t+1)}, \mu^{(t+1)})$. Veja que os pares $(i,j) \in \mu^{(t)} \cap \mu^{(t+1)}$ foram tratados no caso acima, pois itens alocados pelo algoritmo nunca são desalocados, e portanto, para tais itens, temos $p_j^{(t+1)} = p_j^{(t+)}$. Falta mostrar que os pares $(i,j) \in \mu^{(t+1)} \setminus \mu^{(t)}$ também satisfazem (1) e (2). Seja $P = (i_0, j_1, i_1, \ldots, j_l, i_l, j_{l+1})$ o caminho alternante encontrado na iteração t. Os pares $(i,j) \in \mu^{(t+1)} \setminus \mu^{(t)}$ são arestas de P da forma $(i,j) = (i_x, j_{x+1})$. Para mostrar que tais pares satisfazem (1) e (2), vamos considerar dois casos.

Caso 1: $x < l$.
Nesse caso, o par $(i,j) = (i_x, j_{x+1})$ é um arco à frente no grafo $G^{(t)}$ de peso $u_i^{(t)} + p_j^{(t)} - v_{i,j}$, e como ele está no passeio alternante de peso mínimo,

$$d^{(t)}(i_0, j) = d^{(t)}(i_0, i) + \big(u_i^{(t)} + p_j^{(t)} - v_{i,j}\big). \tag{14}$$

Como $w^{(t)}(P) \geq d^{(t)}(i_0, i)$ e $w^{(t)}(P) \geq d^{(t)}(i_0, j)$, fazemos as seguintes atualizações nos preços e nas utilidades:

$$u_i^{(t+1)} = u_i^{(t)} - \big(w^{(t)}(P) - d^{(t)}(i_0, i)\big), \tag{15}$$

$$p_j^{(t+1)} = p_j^{(t)} + \big(w^{(t)}(P) - d^{(t)}(i_0, j)\big). \tag{16}$$

Mas então

$$\begin{aligned}
u_i^{(t+1)} + p_j^{(t+1)} &= \Big(u_i^{(t)} - w^{(t)}(P) + d^{(t)}(i_0, i)\Big) + \Big(p_j^{(t)} + w^{(t)}(P) - d^{(t)}(i_0, j)\Big) \\
&= u_i^{(t)} + d^{(t)}(i_0, i) + p_j^{(t)} - d^{(t)}(i_0, j) \\
&= u_i^{(t)} + d^{(t)}(i_0, i) + p_j^{(t)} - \big(d^{(t)}(i_0, i) + u_i^{(t)} + p_j^{(t)} - v_{i,j}\big) && \text{por (14)} \\
&= v_{i,j}\,,
\end{aligned}$$

e (2) vale para $u_i^{(t+1)}$ e $p_j^{(t)}$.

Falta verificar que $p_j^{(t+1)}$ satisfaz (1). Como (i,j) é um arco à frente em $G^{(t)}$, $p_j^{(t)} \in [r_{i,j}, m_{i,j})$. Por hipótese de indução, a alocação $(u^{(t)}, p^{(t)}, \mu^{(t)})$ é estável, e então $u_j^{(t)} + p_j^{(t)} \geq v_{i,j}$. Usando o fato anterior, temos que $u_i^{(t)} + m_{i,j} > v_{i,j}$, o que implica que existe um arco de preço máximo de i para j em $G^{(t)}$ com peso $u_i^{(t)} + m_{i,j} - v_{i,j}$. Assim,

$$w^{(t)}(P) \leq d^{(t)}(i_0, i) + (u_i^{(t)} + m_{i,j} - v_{i,j}). \tag{17}$$

Para verificar que $p_j^{(t+1)}$ satisfaz (1), veja que $p_j^{(t+1)} \geq p_j^{(t)} \geq r_{i,j}$, e que

$$\begin{aligned}
p_j^{(t+1)} &= p_j^{(t)} + w^{(t)}(P) - \big(d^{(t)}(i_0, i) + u_i^{(t)} + p_j^{(t)} - v_{i,j}\big) && \text{por (16) e (14)} \\
&= w^{(t)}(P) - d^{(t)}(i_0, i) - u_i^{(t)} + v_{i,j} \\
&\leq \big(d^{(t)}(i_0, i) + u_i^{(t)} + m_{i,j} - v_{i,j}\big) - d^{(t)}(i_0, i) - u_i^{(t)} + v_{i,j} && \text{por (17)} \\
&= m_{i,j}.
\end{aligned}$$

Caso 2: $x = l$.
Neste caso, temos $(i,j) = (i_l, j_{l+1}) \in \mu^{(t+1)} \setminus \mu^{(t)}$. Assim, o arco (i,j) não pode ser nem terminal, nem arco de preço máximo, e portanto é um arco de preço mínimo em $G^{(t)}$ com peso $u_i^{(t)} + r_{i,j} - v_{i,j}$. Logo,

$$w^{(t)}(P) = d^{(t)}(i_0, i) + (u_i^{(t)} + r_{i,j} - v_{i,j}). \tag{18}$$

Assim, de (16), (14) e (18), temos que

$$\begin{aligned} p_j^{(t+1)} &= p_j^{(t)} + w^{(t)}(P) - \left(d^{(t)}(i_0, i) + u_i^{(t)} + p_j^{(t)} - v_{i,j}\right) \\ &= \left(d^{(t)}(i_0, i) + u_i^{(t)} + r_{i,j} - v_{i,j}\right) - d^{(t)}(i_0, i) - u_i^{(t)} + v_{i,j} \\ &= r_{i,j}\,, \end{aligned}$$

concluindo que $p_j^{(t+1)}$ satisfaz (1).

A verificação de (2) para $u_i^{(t+1)}$ e $p_j^{(t+1)}$ segue de que $p_j^{(t+1)} = r_{i,j}$ mostrado acima, e de (15) e (18):

$$\begin{aligned} u_i^{(t+1)} + p_j^{(t+1)} &= u_i^{(t)} - w^{(t)}(P) + d^{(t)}(i_0, i) + r_{i,j} \\ &= u_i^{(t)} - \left(d^{(t)}(i_0, i) + u_i^{(t)} + r_{i,j} - v_{i,j}\right) + d^{(t)}(i_0, i) + r_{i,j} \\ &= v_{i,j}\,. \end{aligned}$$

Isso conclui a prova de que o invariante $(A2)$ é válido a cada iteração do algoritmo.

Prova do invariante $(A3)$**.**

Vamos mostrar que a alocação $(u^{(t+1)}, p^{(t+1)}, \mu^{(t+1)})$ satisfaz o invariante $(A3)$. Como todos os itens não alocados em $\mu^{(t)}$ têm preço zero, não existe arco à frente para nenhum destes itens em $G^{(t)}$, pois vale que $r_{i,j} > 0$ para todo par $i, j \in I \times J$, pela suposição do leilão estar em posição geral. Assim, a distância $d^{(t)}(i_0, j')$ é infinita, para todo item não alocado em $\mu^{(t)}$, fazendo com que seu preço não seja alterado na atualização de preços do passo 4 do algoritmo. Note que apenas um possível item j que entra na alocação $\mu^{(t+1)}$ e não estava em $\mu^{(t)}$ pode ter seu preço alterado, quando o passeio alternante de peso mínimo termina em um arco de preço mínimo, e $p_j^{(t+)}$ difere de $p_j^{(t+1)}$. Além disso, todos os itens alocados em $\mu^{(t)}$ continuam alocados em $\mu^{(t+1)}$, e portanto, o invariante $(A3)$ vale para a alocação $(u^{(t+1)}, p^{(t+1)}, \mu^{(t+1)})$.

5.7.2 Invariantes $(B1)$ e $(B2)$.

Infelizmente Aggarwal et. al. [1] não apresentaram uma prova dos invariantes $(B1)$ e $(B2)$. Eles apenas mencionam que esta prova é técnica e que por isso será omitida. Por falta de tempo, não verificamos a validade de $(B1)$ e $(B2)$ e por isso não apresentamos a sua prova aqui também. Optamos por nos concentrar nos resultados que estão explicitamente demostrados no artigo.

5.7.3 Otimalidade da alocação produzida

A demonstração da otimalidade da alocação produzida pelo algoritmo é longa e técnica. Infelizmente alguns poucos pontos dessa demostração não foram compreendidos apesar do nosso esforço. Optamos por apresentar aqui tudo que foi estudado e compreendido, deixando indicados os pontos que são citados no artigo original mas não completamente justificados.

Para um leilão (v, m, r, n, k) em posição geral, a alocação $(u^{(T)}, p^{(T)}, \mu^{(T)})$ devolvida pelo algoritmo é ótima, ou seja, é uma alocação estável e viável e, para toda alocação (u, p, μ) estável e viável, vale $u_i^{(T)} \geq u_i$, para todo comprador i em I.

Veja que a condição de posição geral é necessária para que exista uma alocação ótima. Por exemplo, no caso de termos apenas um item, e dois compradores com $m_{1,1} = m_{2,1}$, e $v_{1,1} > m_{1,1}$ e $v_{2,1} > m_{2,1}$, e $m_{1,1} > r_{1,1}$ e $m_{2,1} > r_{2,1}$, não existe uma alocação ótima. De fato, considere as seguinte alocações, (u^1, p^1, μ^1) e (u^2, p^2, μ^2) onde $u^1 = (v_{1,1} - p_1^1, 0)$, $p^1 = (m_{1,1})$, $\mu^1 = (1, 1)$, e $u^2 = (0, v_{2,1} - p_1^2)$, $p^2 = (m_{2,1})$, $\mu^2 = (2, 1)$. Ambas alocações são estáveis e viáveis, mas $u_2^1 < u_2^2$ e $u_1^2 < u_1^1$, e claramente não existe alocação (u', p', μ') estável e viável com $u_i' \geq u_i^1$ e $u_i' \geq u_i^2$, para todo $i \in I$.

Desconsiderando casos em que o leilão não está em posição geral, sempre existe uma alocação ótima. Podemos deixar um leilão em posição geral fazendo perturbações simbólicas nos valores.

Sem perda de generalidade, vamos assumir que para toda alocação (u, p, μ) não existe um par $(i, j) \in \mu$ tal que $p_j = m_{i,j}$. Podemos assumir isso, pois estamos assumindo que o leilão está em posição geral.

Para verificar que a alocação devolvida pelo algoritmo é ótima, vamos mostrar que, para toda alocação (u', p', μ') viável e estável, vale que $u_i^{(t)} \geq u_i'$ para todo comprador i em I, e $p_j^{(t)} \leq p_j'$ para todo item j em J, ao ínicio de cada iteração t do algoritmo. Veja que isso é suficiente para que a alocação devolvida pelo algoritmo seja ótima, pois já sabemos que ela é estável e viável.

Vamos mostrar isso por indução em t. Para $t = 0$, temos que $p_j' \geq 0 = p_j^{(t)}$, para todo j de J, e $u_i' \leq B = u_i^{(t)}$, para todo i de I. Para $t \geq 0$, suponha que $u_i^{(t)} \geq u_i'$ e $p_j^{(t)} \leq p_j'$.

Primeiramente, vamos provar o seguinte.

Proposição 1: $u_i^{(t+1)} \geq u_i'$ e $p_j^{(t+)} \leq p_j'$ para todo (i, j) em $I \times J$.

Prova. Para isso, vamos analisar as atualizações de preços e utilidades feitas nos passos 3 e 4 do algoritmo como funções contínuas numa variável x. Definimos por $u_i(x)$ a função contínua não-crescente, para todo comprador $i \in I$, onde

$$u_i(x) = u_i^{(t)} - \max\left(x - d^{(t)}(i_0, i), 0\right),$$

e por $p_j(x)$ a função contínua não-descrescente, para todo $j \in J$, onde

$$p_j(x) = p_j^{(t)} + \max\left(x - d^{(t)}(i_0, j), 0\right).$$

Suponha que existe um valor $y' \in [0, w^{(t)}(P)]$ tal que $u_i(y') < u_i'$ para algum $i \in I$, ou $p_j(y') > p_j'$ para algum $j \in J$. Vamos provar que tal y' não existe, o que implica que $u_i^{(t+1)} \geq u_i'$, para todo $i \in I$, e $p_j^{(t+)} \leq p_j'$, para todo $j \in J$.

Considerando o maior $y \in [0, w^{(t)}(P)]$ tal que $u_i(y) \geq u_i'$ para todo $i \in I$ e $p_j(y) \leq p_j'$ para todo $j \in J$, definimos os conjuntos I' e J' por

$$I' = \{i \in I | u_i(y) = u_i' \text{ e } d^{(t)}(i_0, i) \leq y\},$$

$$J' = \{j \in J | p_j(y) = p_j' \text{ e } d^{(t)}(i_0, j) \leq y\}.$$

Se y' existe, então $y < y' \leq w^{(t)}(P)$ por causa da monotocidade das funções $u_i(x)$ e $p_j(x)$.

Proposição 2: Todo item $j \in J'$ está alocado em $\mu^{(t)}$ a um comprador $i \in I'$, e que todo comprador $i \in I'$ está alocado em μ' a um item $j \in J'$.

A proposição 2 é demostrada na próxima subseção (5.8). Da proposição 2 podemos deduzir que $|I'| = |J'|$. Portanto, vale também o reverso destas afirmações. Ou seja, todo item $j \in J'$ está alocado em μ' a um comprador $i \in I'$, e que todo comprador $i \in I'$ está alocado em $\mu^{(t)}$ a um item $j \in J'$. Com isso, resta chegar à contradição.

Considere $j \in J'$ com o menor valor $d^{(t)}(i_0, j)$. Considere o caminho de peso mínimo em $G^{(t)}$ de i_0 para j que usa apenas arcos à frente e arcos inversos. O vértice anterior a j nesse caminho é um comprador i que não está em I', pois ele não está alocado à um item em J', pela escolha de j.

Vamos mostrar que u'_i e p'_j não satisfazem as condições de estabilidade. Para que a alocação (u', p', μ') seja estável, pelo menos uma das seguintes equações deve ser válida

$$u'_i + p'_j \geq v_{i,j} \ , \tag{19}$$
$$u'_i + r_{i,j} \geq v_{i,j} \ , \tag{20}$$
$$p'_j \geq m_{i,j} \ . \tag{21}$$

Como $j \in J'$ vale $p_j(y) = p'_j$ e $d^{(t)}(i_0, j) \leq y$. Logo $d^{(t)}(i_0, i) \leq d^{(t)}(i_0, j)$, e temos que $d^{(t)}(i_0, i) \leq y$. Assim sendo, $u_i(y) > u'_i$ já que $i \notin I'$.

Como há um arco à frente de i para j em $G^{(t)}$, temos $p_j^{(t)} \in [r_{i,j}, m_{i,j})$. Como $u_i(y) = u_i^{(t)} - y + d^{(t)}(i_0, i)$ e $p_j(y) = p_j^{(t)} + y - d^{(t)}(i_0, j)$ e também há uma arco à frente de i para j de custo $u_i^{(t)} + p_j^{(t)} - v_{i,j}$, temos

$$u_i(y) + p_j(y) = u_i^{(t)} + p_j^{(t)} - (u_i^{(t)} + p_j^{(t)} - v_{i,j}) = v_{i,j}.$$

Logo $u'_i + p'_j < u_i(y) + p_j(y) = v_{i,j}$ e portanto (19) não vale. Por indução, $p'_j \geq p_j^{(t)} \geq r_{i,j}$, e portanto (20) também não vale.

Segundo Aggarwal et. al. [1], o fato de $G^{(t)}$ ter um arco de preço máximo de i para j implica que $p'_j = p_j(y) < m_{i,j}$. Portanto, (21) também não vale.

Com isso concluímos que não existe um valor $y' \in [0, w^{(t)}(P)]$ tal que $u_i(y') < u'_i$ para algum $i \in I$, ou $p_j(y') > p'_j$ para algum $j \in J$.

Isso prova que $u_i^{(t+1)} \geq u'_i$ e $p_j^{(t+)} \leq p'_j$, em cada iteração t do algoritmo.

□

Proposição 3: $u_i^{(t+1)} \geq u'_i$ e $p_j^{(t+1)} \leq p'_j$ para todo (i, j) em $I \times J$.

Prova. Para finalizar, resta mostrar que $p_j^{(t+1)} \leq p'_j$. Sabemos que $p_j^{(t+1)}$ difere de $p_j^{(t+)}$ apenas quando o caminho alternante em $G^{(t)}$ termina em um arco de preço mínimo $(i, j) = (i_l, j_{l+1})$, e quando $p_j^{(t+)} < r_{i,j}$. Nesse caso, $p_j^{(t+1)} = r_{i,j}$. Precisamos verificar que, neste caso,

$$r_{i,j} \leq p'_j \ .$$

Como a alocação (u', p', μ') é estável, vale pelo menos uma entre (19), (20) e (21). Segundo Aggarval et. al. [1], sempre valem (19) ou (21). Vamos verficar esses dois casos.

Como passeio alternante terminou em um arco de preço mínimo, vale que $u_i^{(t+1)} = v_{i,j} - r_{i,j}$, e vale também que $u_i^{(t+1)} \geq u'_i$, por indução. Então, se vale (19), substituindo u'_i por $u^{(t+1)}$ em (19), temos

$$\left(v_{i,j} - r_{i,j}\right) + p'_j \geq v_{i,j}\,.$$

Portanto, neste caso, vale que $r_{i,j} \leq p'_j$.

Como existe arco de preço mínimo de i para j neste caso, temos que $r_{i,j} \leq m_{i,j}$. Se vale (21), ou seja, $m_{i,j} \leq p'_j$, então, neste caso, também vale que $r_{i,j} \leq p'_j$.

Portanto, para alocação $(u^{(t+1)}, p^{(t+1)}, \mu^{(t+1)})$, vale que $u_i^{(t+1)} \geq u'_i$, e que $p_j^{(t+1)} \leq p'_j$, para toda alocação (u', p', μ') estável e viável. O que implica que a alocação devolvida pelo algoritmo é ótima.

□

5.8 Prova da Proposição 2

Vamos mostrar que todo item $j \in J'$ está alocado em $\mu^{(t)}$ a um comprador $i \in I'$, e que todo comprador $i \in I'$ está alocado em μ' a um item $j \in J'$, para depois, chegarmos à contradição, verificando que tal y' não existe.

Primeiramente, vamos verificar que todo item $j \in J'$ está alocado em $\mu^{(t)}$ a um comprador $i \in I'$. Como $j \in J'$, tem-se que $d^{(t)}(i_0, j) \geq y < w^{(t)}(P)$. Se o item j estiver desalocado, então $d^{(t)}(i_0, j) = \infty$, pois não há arco à frente incidente em j. Isso é uma contradição. Portanto, o item j está alocado a algum comprador $i \in I$.

Queremos mostrar que $i \in I'$, ou seja, que $d^{(t)}(i_0, i) \leq y$ e $u_i(y) = u'_i$. Como j está alocado a i em $\mu^{(t)}$, há arco inverso de j para i em $G^{(t)}$ e seu peso é 0 pelo invariante $(A2)$. Assim, $d^{(t)}(i_0, i) = d^{(t)}(i_0, j) \leq y$, pois $j \in J'$. Resta mostrar que $u_i(y) = u'_i$. Pela escolha de y, temos que $u_i(y) \geq u'_i$. Como $d^{(t)}(i_0, i) = d^{(t)}(i_0, j)$, temos também que $u_i(y) + p_j(y) = u_i^{(t)} + p_j^{(t)} = v_{i,j}$ pela definição de $u_i(x)$ e $p_j(x)$ e pelo invariante $(A2)$. Ou seja, $u_i(y) = v_{i,j} - p_j(y) = v_{i,j} - p'_j$ pois $j \in J'$.

Se mostrarmos que $u'_i + p'_j \geq v_{i,j}$, a prova termina, pois, pela equação acima, teremos $u_i(y) \leq u'_i$, implicando que $u_i(y) = u'_i$. Para isso, basta mostrar que $p'_j \in [r_{i,j}, m_{i,j})$, pois isso implica que $u'_i + p'_j \geq v_{i,j}$ já que (u', p', μ') é estável. Veja que $p'_j \in [r_{i,j}, m_{i,j}]$ pois (u', p', μ') é viável. Então, temos que mostrar apenas que $p'_j < m_{i,j}$.

Os invariantes $(A2)$ e $(B1)$ implicam que $p_j^{(t)} \in [r_{i,j}, m_{i,j})$. Consequentemente, existe um arco de preço máximo de i para j em $G^{(t)}$ já que $r_{i,j} < m_{i,j}$ e $u_i^{(t)} + m_{i,j} > u_i^{(t)} + p_j^{(t)} = v_{i,j}$. Logo $w^{(t)}(P) \leq d^{(t)}(i_0, i) + u_i^{(t)} + m_{i,j} - v_{i,j}$. Mas então

$$\begin{aligned}
p'_j = p_j(y) &= p_j^{(t)} + y - d^{(t)}(i_0, j) \\
&= p_j^{(t)} + y - d^{(t)}(i_0, i) \\
&< p_j^{(t)} + w^{(t)}(P) - d^{(t)}(i_0, i) \\
&\leq p_j^{(t)} + u_i^{(t)} + m_{i,j} - v_{i,j} \\
&= m_{i,j}\,.
\end{aligned}$$

Assim, terminamos a verificação de que todo item $j \in J'$ está alocado em $\mu^{(t)}$ a um comprador

$i \in I'$.

Agora, vamos verificar que todo compradorm $i \in I'$ está alocado em μ' a um item $j \in J'$. Observe que $u'_i > 0$ já que

$$\begin{aligned} u'_i &= u_i(y) \\ &= u_i^{(t)} - \big(y - d^{(t)}(i_0, i)\big) \\ &> u_i^{(t)} - \big(w^{(t)}(P) - d^{(t)}(i_0, i)\big) = u_i^{(t+1)} \geq 0 \ . \end{aligned}$$

Assim o comprador i está alocado em μ' a algum item $j \in J$.

Queremos mostrar que $j \in J'$, ou seja, que $p_j(y) = p'_j$ e $d^{(t)}(i_0, j) \leq y$. Pela escolha de y, temos que $p_j(y) \leq p'_j$. Se provarmos que $p_j(y) \geq v_{i,j} - u_i(y)$ então, como $u_i(y) = u'_i$ já que $i \in I'$ e $u'_i + p'_j = v_{i,j}$ pois (u', p', μ') é viável, temos que

$$p_j(y) \geq v_{i,j} - u_i(y) = v_{i,j} - u'_i = p'_j \ .$$

Ou seja, para mostrar que $p_j(y) = p'_j$, é suficiente mostrarmos que $p_j(y) \geq v_{i,j} - u_i(y)$.

Da definição de $p_j(y)$, temos que

$$p_j(y) \geq p_j^{(t)} + y - d^{(t)}(i_0, j) \ .$$

Como (u', p', μ') é viável, temos que $p'_j \in [r_{i,j}, m_{i,j}]$. Por indução, $p_j^{(t)} \leq p'_j$. Como o leilão está em posição geral, vale $p_j^{(t)} \neq m_{i,j}$. Assim sendo, $p_j^{(t)} < m_{i,j}$.

Segundo Aggarwal et. al. [1], $p_j^{(t)} \geq r_{i,j}$ o que garante a existência do arco à frente de i para j em $G^{(t)}$ e

$$d^{(t)}(i_0, j) \leq d^{(t)}(i_0, i) + \big(u_i^{(t)} + p_j^{(t)} - v_{i,j}\big) \ . \qquad (*)$$

Portanto,

$$\begin{aligned} p_j(y) &\geq p_j^{(t)} + y - \big(d^{(t)}(i_0, i) + u_i^{(t)} + p_j^{(t)} - v_{i,j}\big) \\ &= v_{i,j} - \big(u_i^{(t)} - (y - d^{(t)}(i_0, i))\big) \\ &= v_{i,j} - u_i(y) \ . \end{aligned}$$

Resta mostrar que $d^{(t)}(i_0, j) \leq y$. Como $u_i(y) = u_i^{(t)} - y + d^{(t)}(i_0, i)$, de (*) deduzimos que

$$\begin{aligned} d^{(t)}(i_0, j) &\leq u_i(y) + y + p_j^{(t)} - v_{i,j} \\ &= u'_i + y + p_j^{(t)} - v_{i,j} && \text{pois } i \in I' \\ &\leq y + u'_i + p_j(y) - v_{i,j} && \text{pela hipótese de indução} \\ &= y + p_j(y) - p'_j && \text{pela viabilidade de } (u', p', \mu') \\ &= y && \text{pelo que provamos anteriormente.} \end{aligned}$$

5.9 Consumo de tempo

Inicialmente, o grafo de atualização $G^{(0)}$ possui no máximo nk arcos de preço mínimo, nk arcos de preço máximo, e n arcos terminais. Vamos mostrar que a cada iteração do algoritmo, a soma do número de arcos destes tipos diminui de um. Com isso, conseguiremos concluir que o número de iterações do algoritmo é limitado por $2nk + n$, ou seja $O(nk)$.

Considere uma iteração t do algoritmo *StableMatch*, e suponha que t não é a última iteração. A última aresta passeio alternante de peso mínimo $P = (i_0, j_1, i_1, \ldots, l_l, i_l, j_{l+1})$ em $G^{(t)}$ não aparece no grafo de atualização $G^{(t+1)}$. Vamos mostrar que a aresta $(i, j) = (i_l, j_{l+1})$ não aparece em $G^{(t+1)}$:

- *Caso 1.*
  Se (i, j) é um arco terminal, então $w^{(t)}(P) = d^{(t)}(i_0, i) + u_i^{(t)}$. Portanto,
  $$u_i^{(t+1)} = u_i^{(t)} - \left(w^{(t)}(P) - d^{(t)}(i_0, i)\right) = 0.$$

- *Case 2.*
  Se (i, j) é um arco de preço máximo, então $w^{(t)}(P) = d^{(t)}(i_0, i) + \left(u_i^{(t)} + m_{i,j} - v_{i,j}\right)$. Portanto,
  $$u_i^{(t+1)} + m_{i,j} = u_i^{(t)} - \left(w^{(t)}(P) - d^{(t)}(i_0, i)\right) + m_{i,j} = v_{i,j}.$$

- *Caso 3.*
  Se (i, j) é um arco de preço mínimo, então $w^{(t)}(P) = d^{(t)}(i_0, i) + \left(u_i^{(t)} + r_{i,j} - v_{i,j}\right)$.. Portanto,
  $$u_i^{(t+1)} + r_{i,j} = u_i^{(t)} - \left(w^{(t)}(P) - d^{(t)}(i_0, i)\right) + r_{i,j} = v_{i,j}.$$

  Como a utilidade dos compradares nunca aumenta e o preço dos itens nunca diminui ao longo do algoritmo, então o arco (i_l, j_{l+1}) não aparece no grafo de atualização $G^{(t')}$, para $t' > t$.

  Basta agora verificarmos qual é o consumo de tempo de uma iteração. Primeiramente, vamos relembrar que o custo dos arcos de $G^{(t)}$ são sempre maiores ou iguais a zero, para toda iteração t. Portanto podemos usar o *algoritmo de Dijkstra* para encontrar o passeio alternante de peso mínimo. Assim, o consumo de tempo de uma iteração do algoritmo é $O((n + k)^2)$.

  Como em geral temos $n > k$, concluímos que o consumo de tempo do algoritmo todo é $O(nk) * O(n^2)$, ou seja, $O(kn^3)$.

5.10 Prova de estratégia

Teorema 5.1 *Não existe maneira de um comprador ou mesmo um grupo de compradores manipularem os valores que declaram ao algoritmo de forma que todo comprador nesse grupo seja beneficiado por essa manipulação.*

A prova desse teorema se baseia no seguinte resultado. Seja (u, p, μ) uma alocação que é viável para o leilão (v, m, r, n, k) que está em posição geral, e seja (u^*, p^*, μ^*) a alocação ótima para o leilão. Seja
$$I^+ = \{i \in I \mid u_i > u_i^*\}.$$
Se I^+ é não vazio, então existe um par bloqueador $(i, j) \in (I \setminus I^+) \times J$.

Infelizmente não foi possível concluir essa prova, por falta de tempo. Aggarwal et. al. [1] apresentam a prova, e mencionam que o mecanismo para realizá-la é semelhante à muitos outros.

6 Conclusões e Resultados

Podemos concluir deste projeto que a área de teoria dos jogos envolve muitos conceitos, e pode ser aplicada em diversos campos. Os resultados obtidos através do estudo dos problemas mostrados eram todos resultados já conhecidos. Porém, são resultados muito interessantes.

A intenção inicial do projeto foi alcançada. Em primeiro lugar escolhemos problemas interessantes da área de teoria dos jogos. Tentamos ao máximo detalhar cada passagem matemática, e ilustrar exemplos para facilitar a compreensão dos resultados mostrados. Dessa maneira, foi possível apresentar cada um dos problemas estudados de uma maneira didática para o leitor. Acredito que este projeto pode ser de grande ajuda para uma pessoa que esteja interessada na área.

Além disso, os algoritmos foram implementados e estão disponíveis na capa do projeto. Alguns testes foram realizados para entradas geradas aleatoriamente, tentando comprovar a validade das provas demostradas para as soluções propostas. Conseguimos verificar as condições de estabilidade e viabilidade das soluções encontradas, porém as condições de otimalidade e prova de estratégia não podem ser verificadas tão facilmente por alguma rotina. Para entradas pequenas, conseguimos fazer algo como uma força bruta para gerar todas as alocações estáveis, e verificar que nenhuma delas é melhor do que a alocação encontrada pelo algoritmo implementado.

7 Parte Subjetiva

Este trabalho de conclusão de curso se originou em uma iniciação científica orientada pela professora Cristina Gomes Fernandes. Em um primeiro momento, comecei estudando alguns conceitos e definições da área de teoria dos jogos. Aos poucos, fomos escolhendo assuntos que se mostravam mais interessantes. Nesta seção irei tratar das dificuldades, desafios e frustrações que surgiram ao longo da elaboração do projeto, além de realçar a importância de certas disciplinas cursadas.

7.1 Desafios e frustrações

O assunto estudado aborda, de certa forma, aspectos que não são vistos durante o curso. Essa novidade, apesar de tornar um pouco mais difícil a compreensão do assunto, tornou também o projeto mais interessante.

Foi realmente desafiador tentar entender todas as propriedades garantidas por cada uma das soluções propostas para os problemas apresentados. Em algumas provas, os autores omitem algumas passagens, dificultando o entendimento. Além disso, como sempre, existem algumas falhas de digitação. Em algumas demonstrações encontramos o símbolo "$<$" onde deveria ser "$>$", ou encontramos a variável i onde deveria ser j, e outras pequenas falhas semelhantes.

O período de estudo sobre o problema dos casamentos estáveis, e também sobre o problema da alocação de bens indivisíveis foi um pouco frustante. Eu não esperava que problemas tão simples teriam provas tão complicadas. Acho que foi uma fase de adaptação. Mas a partir do conhecimento adquirido durante essa primeira fase, foi possível começar a trabalhar em problemas mais complexos e interessantes.

Após estudar esses dois primeiros problemas, passamos para o problema dos leilões combinatórios. Tratava-se de um problema mais complexo, que envolvia não só teoria dos jogos, mas também conceitos de complexidade computacional. Esse problema possui características interessantes. Isso se deve ao fato de que após muitas demonstrações, concluímos que um simples algoritmo guloso é a melhor solução que existe para o problema.

Durante o segundo semestre de 2009, foquei meus esforços no último problema estudado neste projeto. Dentre todos, este problema foi o que me deixou mais intrigado e curioso. Aos poucos fui entendo o motivo de certas definições. Esse problema, apesar de ser muito interessante, exigiu certa dedicação. Em especial, passei mais de um mês estudando a prova de otimalidade do algoritmo *StableMatch*. As passagens matemáticas são muito complexas. Além disso, é difícil entender algumas hipóteses que os autores assumem valer, mas não explicam muito bem os motivos. Com muita ajuda da orientadora, que me auxiliou durante toda a elaboração deste projeto, conseguimos completar quase todas as expectativas que tínhamos. Não foi possível concluir todas as provas, nem tratar muito bem o caso da posição geral do leilão na hora de implementar. Fiz pequenas modificações no algoritmo para que este pudesse resolver instâncias genéricas, e elas me pareceram realmente funcionar. Mas infelizmente os autores não explicam quais são as modificações que podemos fazer para que o algoritmo funcione sem a hipótese de posição geral do leilão, apenas citam que isso é possível.

7.2 Disciplinas relevantes

Podemos destacar algumas matérias que tiveram grande relevância para a elaboração deste projeto.

- **MAC0110 - Introdução à computação**

MAC0122 - Princípios de desenvolvimento de algoritmos
MAC0323 - Estrutura de dados
MAC0327 - Desafios de programação
Essas disciplinas foram de grande ajuda para que eu pudesse implementar os algoritmos estudados, e conseguir fazer certas verificações sobre as provas apresentadas. É um alívio ver que uma prova matemática realmente funciona na prática. Além disso, elas me proporcionaram gosto pela programação, o que me motiva bastante a estudar problemas difíceis.

- **MAC0315 - Programação linear**
  Em alguns trechos do projeto foi necessário um conhecimento sobre programação linear, para poder mostrar maneiras alternativas de se obter uma solução. Apesar de ter passado superficialmente por esses trechos, a disciplina em questão foi de grande ajuda.

- **MAC0450 - Algoritmos de aproximação**
  Essa disciplina também teve participação na área de programção linear, citada acima. Além disso, para mostrar que um dos problemas estudados é NP-difícil, foi preciso fazer uma redução polinomial. Algoritmos de aproximação trata de muitos problemas difíceis, e também aborda maneiras de se relacionar tais problemas, através da contrução de instâncias de problema a partir de instâncias de outro problemas. Portanto, esta disciplina foi muito relevante, em especial, nessa parte específica do projeto.

- **MAC0328 - Algoritmos em grafos**
  Como pode ser visto, todos os problemas estudados envolvem algum assunto de grafos. Sem a base adquirida nesta disciplina, seria impossível conseguir acompanhar os resultados estudados.

- **MAC0338 - Análise de algoritmos**
  Essa disciplina tem forte participação na compreensão das demonstrações estudadas neste projeto. Em análise de algoritmo aprendemos não só a analisar a complexidade de tempo que um algoritmo consome, mas também a provar a correção de muitos algoritmos, como algoritmos gulosos e de programação dinâmica.

- É difícil dividir a importância das matérias de maneira isolada, pois muitas delas estão fortemente relacionadas. Mas de certa forma, todas as disciplinas do curso tiveram influência positiva para a elaboração do projeto. Entre outras matérias que tiveram grande partipação para a elaboração do projeto estão **MAC0436 - Tópicos de matemática discreta**, **MAC0330 - Introdução à teoria dos grafos** e **MAC0325 - Otimização combinatória**.

7.3 Trabalhos futuros

Todos os problemas estudados podem ser aprofundados. Os dois primeiros problemas possuem variantes, que foram até citadas no projeto. Para o problema dos leilões combinatórios pode ser estudada uma representação compacta dos valores ideais dos compradores sobre todos os conjuntos de objetos, assim como a solução apresentada para esses leilões no caso geral, aprofundando um pouco na área de programação linear. Esse último problema estudado também possui características que não foram abordadas. Dentre elas temos a modelagem deste problema quando estamos tratando de usuários clicando nas publicidades em uma página de busca. Além disso, existem muitas questões em aberto para esse problema, que podem ser

encontradas no artigo. Além disso, existem também outros problemas interessantes na área de teoria dos jogos que também podem ser estudados.

Com essas inúmeras opções para trabalhos futuros, o enfoque para a continuação deste projeto pode seguir por qualquer um dos caminhos citados.

Referências

[1] Gagan Aggarwal, S. Muthukrishnan, Dávid Pál, and Martin Pál. General auction mechanism for search advertising. *CoRR*, abs/0807.1297, 2008.

[2] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows*. Prentice-Hall, 1993.

[3] D. Gale and L.S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[4] M.R. Garey and D.S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. Freeman, 1979.

[5] Daniel Lehmann, Liadan Ita Oćallaghan, and Yoav Shoham. Truth revelation in approximately efficient combinatorial auctions. *J. ACM*, 49(5):577–602, 2002.

[6] L. Lovász and M.D. Plummer. *Matching Theory*. Elsevier, 1986.

[7] N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007.