Introdução à Teoria dos Jogos Algorítmica

Flávio Keidi Miyazawa

2010

Resumo. O advento da Internet trouxe não só novas formas de se fazer computação, mas também maneiras diferentes de se realizar várias atividades, como comunicação, busca por informações, educação, lazer, propaganda, etc. Por se tratar de um ambiente descentralizado, muitas destas atividades envolvem colaboração ou disputa pelos recursos, e com isso pode motivar que usuários atuem guiados por interesses próprios, buscando maximizar seus benefícios seja de maneira direta ou através de seus programas. Uma das teorias usadas para modelar estas situações de maneira formal é a Teoria dos Jogos. Trata-se de um ramo da matemática aplicada que tem sido aplicada a várias áreas (como economia, biologia, sociologia, etc) onde há conflito e/ou cooperação entre indivíduos. Por outro lado, a aplicação desta teoria pode esbarrar na grandeza das instâncias envolvidas, principalmente nas aplicações da Internet. Com isso, muitas soluções obtidas na área de Teoria dos Jogos não são aplicáveis diretamente por serem inviáveis computacionalmente. Tais soluções podem esbarrar na representação do jogo, na forma como os usuários fazem suas escolhas ou na intratabilidade computacional dos vários problemas necessários para sua resolução. Com isso, surge a área de Teoria dos Jogos Algorítmica, que investiga soluções algorítmicas e a complexidade computacional destes problemas. O texto é uma breve introdução à Teoria dos Jogos Algorítmica, que como a Internet, tem crescido muito nos últimos anos. Com isso, não será possível apresentar os vários aspectos e abordagens que vem sendo tratados nesta área e esperamos que este texto motive o leitor a um estudo mais detalhado.

Sumário

1 Introdução

A Teoria dos Jogos é uma área da matemática aplicada usada para modelar fenômenos onde dois ou mais agentes interagem entre si. É usada em situações onde as decisões de um agente (ou jogador) dependem ou influenciam as escolhas de outros. Se popularizou após a publicação do livro *Theory of Games and Economic Behavior* por von Neumann e Morgenstern em 1944, e teve como base trabalhos anteriores de von Neumann. De fato, von Neumann criou teorias que tiveram grande impacto em várias áreas. Uma delas é na Computação, para a qual algumas das contribuições incluem a arquitetura de computadores, princípios de programação, análise de algoritmos, entre outras [25].

Com o advento da Internet, houve uma transformação rápida na forma como as informações são divulgadas e trabalhadas. Seu crescimento rápido e descentralizado, permitiu disponibilizar e compartilhar informações de maneira nunca antes vista. Gerou várias formas alternativas de comércio, relacionamentos e maneiras como a computação é realizada. Com isso, surgiram também vários problemas computacionais onde não há um controle centralizado dos agentes envolvidos. Além disso, muitas das ações destes agentes são guiados por interesses individuais, onde a decisão de um agente pode mudar a decisão de outros. Fomentados principalmente pelas aplicações e problemas da Internet, surgiu a área de Teoria dos Jogos Algorítmica, que é a combinação de outras duas, que tiveram grande contribuição de von Neumann: a Teoria dos Jogos e a Computação. Nesta área, estamos interessados em investigar, analisar e projetar algoritmos e regras associadas a ambientes onde vários agentes tem interesses próprios, com impactos em outros.

Por já ter sido bastante investigada, muitos dos conceitos e soluções apresentados dentro da área de Teoria dos Jogos se encaixam aos problemas computacionais que veremos. Porém, muitas não consideram nem tratam de maneira adequada os modelos computacionais e suas restrições. Estas restrições são muitas vezes inviáveis de serem tratadas, dado que as aplicações, principalmente advindas da Internet, envolvem um volume de dados e quantidade de usuários extremamente grandes.

Neste texto, estamos preocupados em analisar o uso dos recursos em jogos envolvendo algoritmos computacionais bem como projetar jogos com regras que incentivem os jogadores a declararem informações vitais para o bom funcionamento do jogo. Por ser uma área bem consolidada, demandaríamos uma boa parte do texto apenas para descrever os vários tipos de jogos e modelos nesta área. Assim, optamos por focar nossa atenção em poucos modelos que permitam que o leitor aprecie algumas análises e preocupações algorítmicas que aparecem na área. Uma excelente introdução à Teoria dos Jogos é apresentada em [10], que apresenta vários modelos e exemplos diferentes com aplicações na área da Computação.

A seguir, apresentaremos um breve resumo da organização do texto. Na Seção 2 veremos alguns jogos clássicos e introdutórios na área de teoria dos jogos, com alguns exemplos para a Computação. Em seguida, veremos definições mais formais sobre jogos puros e mistos e suas estratégias.

Na Seção 3, comentamos aspectos sobre a complexidade computacional de se obter uma configuração em equilíbrio de Nash. Tais configurações são resultados do jogo onde nenhum jogador tem interesse em mudar de escolha, atingindo assim uma configuração estável.

Na Seção 4, veremos algumas medidas de importância que aparecem no estudo algorítmico de jogos puros. Para isso, consideramos jogos que convergem para equilíbrios de Nash e comparamos resultados do jogo que estão em equilíbrio com um resultado ótimo, não necessariamente em equilíbrio, mas com o melhor benefício global (social). Veremos sobre as duas principais medidas do equilíbrio nesta linha: o *preço da anarquia* e o *preço da estabilidade*. Nesta seção são considerados dois jogos, o jogo de balanceamento de carga, onde o preço da anarquia é mais interessante, e o jogo de roteamento global, onde o preço da estabilidade nos leva a análises e técnicas de maior interesse. Este segundo problema é abordado como um jogo potencial e também é um exemplo de jogo colaborativo, onde os jogadores que usam um mesmo recurso compartilham seu custo (*cost sharing*).

Na Seção 5, veremos sobre o projeto de mecanismos de jogos. Aqui estamos interessados em fazer escolhas sociais baseadas nas preferências dos jogadores. Para isso, o mecanismo deve ter regras que incentivem os jogadores a declarar informações verdadeiras para que a escolha social seja feita adequadamente. Nos restringiremos a casos onde o projeto de mecanismos considera uma valoração dos benefícios e custos por parte dos jogadores. Veremos um exemplo de leilão de item único e da construção de um caminho de custo mínimo.

Na Seção 6, apresentaremos mecanismos para problemas de leilões combinatoriais. Veremos alguns casos tratáveis computacionalmente e outro caso, cuja busca do resultado ótimo social é intratável computacionalmente. Seu tratamento será feito por algoritmos não necessariamente ótimos, mas que obtém resultados de maneira eficiente e dentro de uma aproximação do resultado ótimo social.

Neste texto consideramos questões importantes do lado computacional, como a eficiência e a complexidade computacional de se resolver os problemas. Boa parte deste texto foi baseada em capítulos do livro *Algorithmic Game Theory* [32]. Este foi o primeiro livro a tratar da Teoria dos Jogos Algorítmica e cada capítulo trata de um tópico e foi escrito por um ou mais pesquisadores especialistas da área.

2 Jogos Clássicos e Conceitos Básicos

A Teoria dos Jogos modela situações onde há dois ou mais agentes (também chamados de jogadores), cada um fazendo escolhas que podem afetar os resultados dos outros agentes. Nesta seção, apresentaremos alguns jogos clássicos da área de Teoria dos Jogos, começando com um dos mais importantes, conhecido como o Dilema dos Prisioneiros. Posteriormente, veremos outros jogos clás-

sicos e algumas aplicações na computação. Esta seção foi baseada no artigo [37]. Por limitação de espaço, não foi possível apresentar os diversos tipos de jogos e suas aplicações na computação. Para mais detalhes, recomendamos que o leitor interessado veja [10, 32, 3].

Dilema dos Prisioneiros. Dois prisioneiros, que chamaremos de A e B, estão sendo julgados por um crime. A cada um são apresentadas duas escolhas: confessar o crime ou permanecer em silêncio. Se ambos confessarem, eles terão suas penas reduzidas por colaborarem e cumprirão quatro anos cada um. Se A confessar e B ficar em silêncio, então A será usado como testemunha contra B, que terá uma pena de cinco anos, enquanto A terá sua pena reduzida para um ano. Da mesma forma, se B confessar e A ficar em silêncio, A terá pena de cinco anos e B de um ano. Se ambos ficarem em silêncio não será possível julgar por todos os crimes e cada um ficará dois anos preso, por crimes menores. As penas de cada prisioneiro são esquematizadas pela matriz da Figura 1-*(a)*. Cada

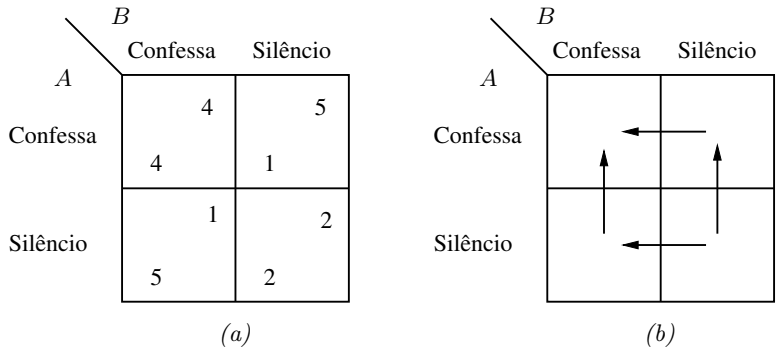


Figura 1: Dilema dos Prisioneiros.

célula representa um possível resultado do jogo, conforme as escolhas dos prisioneiros, e contém dois valores: o do canto inferior esquerdo é a pena do prisioneiro A e do canto superior direito é do prisioneiro B. Uma matriz deste tipo é chamada de matriz de custos (resp. ganhos), quando os valores representam custos (resp. ganhos). Na matriz da Figura 1-*(b)* as setas representam as preferências dos prisioneiros de um resultado para outro. A análise deste e dos outros jogos depende muito das regras envolvidas nele. Para ilustrar, vamos considerar primeiro que cada jogador deve fazer uma escolha sem o conhecimento das escolhas dos outros jogadores. Neste caso, temos um jogo com escolhas simultâneas, que exemplificaremos com o Dilema dos Prisioneiros a seguir.

Jogos com Escolhas Simultâneas. Considere o jogo do Dilema dos Prisioneiros. Vamos analisar a escolha do prisioneiro A de acordo com as possíveis escolhas do prisioneiro B. Caso o prisioneiro B escolha *Confessa*, é preferível ao prisioneiro A também escolher *Confessa* pois com isso ficará quatro anos preso, em vez de cinco. Caso o prisioneiro B escolha *Silêncio*, também é preferível para o prisioneiro A escolher *Confessa*, pois ficará um ano preso, em vez de dois. A mesma análise pode ser feita para o prisioneiro B. Assim, é sempre preferível para um prisioneiro escolher *Confessa*, independente da escolha do outro prisioneiro. Com estas escolhas, cada prisioneiro fica quatro anos

preso. Por outro lado, se ambos ficassem em silêncio, cada um ficaria apenas dois anos preso. Porém, este não é um resultado estável, mesmo que os prisioneiros combinem previamente, já que no momento da escolha, um prisioneiro tem motivação para trair e ficar menos tempo preso. A análise de resultados estáveis estão entre os principais interesses na área de Teoria dos Jogos.

Roteamento por Provedores de Serviço. Este é um exemplo de roteamento em provedores de serviço de Internet (ISP - *Internet Service Providers*) onde a mesma situação do Dilema dos Prisioneiros pode ocorrer. Suponha que temos dois provedores de serviços ISP-A e ISP-B, cujas redes estão ilustradas na Figura 2 (rede ISP-A do lado esquerdo e ISP-B do lado direito). As duas redes possuem dois pontos por onde podem transmitir dados de uma para outra rede, chamados pontos de troca, que são os pontos C e S. Quando há uma requisição de transmissão com origem em um ponto a_1 (na rede ISP-A) para um ponto b_3 (que está na rede ISP-B) o provedor ISP-A deve fazer esta transmissão através de uma rota (caminho) necessariamente passando por um dos pontos C ou S. A rota percorrida dentro de uma mesma rede é determinada pelo seu respectivo provedor e uma vez que o pacote atinge a rede de destino, todo o trajeto restante será feito na mesma rede. Posto isso, uma transmissão de a_1 para b_3 pode seguir pela rota $(a_1, C, b_1, b_2, S, b_3)$ ou pela rota

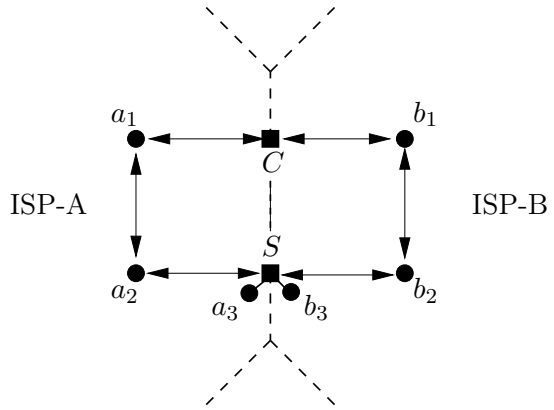


Figura 2: Roteamento por provedores de serviço de Internet.

(a_1, a_2, S, b_3) e como é o provedor ISP-A que define em qual ponto de troca ele chega (se primeiro em C ou em S), é a decisão do provedor ISP-A que define qual a rota o pacote seguirá. Como os provedores tem comportamento racional, eles tentam minimizar o tráfego em sua rede e enviam o tráfego pela rota que chegar primeiro a um ponto de troca. Vamos supor que cada aresta da rede tem custo unitário, exceto pelas ligações S—a_3 e S—b_3, que terão custo 0 por ligarem pontos muito próximos. Assim, se o provedor ISP-A escolher a rota passando por C, ele terá custo 1, porém fará com que ISP-B tenha custo 3. Caso ele escolha a outra rota, terá custo 2 e não acarretará custos para ISP-B. Para deixar esta situação igual ao do Dilema dos Prisioneiros, suponha que haja uma requisição para transmitir do ponto b_1 para o ponto a_3. Com isso, se ambos provedores se comportarem de maneira racional (fazem suas respectivas transmissões passarem por C), cada

provedor terá um custo total de 4. Se ambos não enviarem o pacote por C, cada provedor terá custo 2. Quando apenas um servidor for racional, ele terá custo 1 e o outro 5. Assim, a matriz de custo é exatamente a mesma matriz apresentada para o exemplo do Dilema do Prisioneiro (com C=Confessa e S=Silêncio).

A partir do exemplo do Dilema do Prisioneiro, é fácil fazer outras variações de jogos, simplesmente alterando os valores na matriz de custos do jogo, como veremos a seguir.

Batalha dos Sexos. Um rapaz e uma garota estão decidindo uma atividade de lazer e consideram duas possibilidades: Assistir um jogo de futebol ou um de vôlei. Ambos gostam dos jogos, mas o rapaz prefere futebol e a garota vôlei. Por outro lado, ambos preferem ficar juntos que separados. A Figura 3, mostra um exemplo de uma possível matriz de ganho (preferência) para esta situação. Este é um *jogo de coordenação* com dois resultados estáveis e seus participantes terão mais vantagens

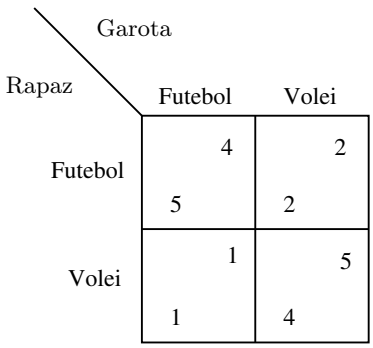


Figura 3: Batalha dos Sexos.

se houver um acordo entre eles para um melhor uso dos recursos. Na Seção 2.3 veremos uma forma de escolha mais flexível que permitirá que este jogo tenha mais um resultado estável.

Jogo de Congestionamento 1. Para ilustrar este comportamento, considere um jogo de controle de congestionamento em uma rede. Suponha que dois usuários A e B podem transferir seus dados por dois pontos de conexão, P e Q. O ponto P permite transferir os dados com uma velocidade um pouco maior que o ponto Q. Assim, é natural que ambos prefiram fazer suas conexões pelo ponto P, porém se ambos transferirem os dados por um mesmo ponto de conexão, haverá um congestionamento no ponto e o tempo de transmissão para cada um será muito maior. Para enriquecer um pouco mais este exemplo, vamos supor que o jogador A tem um pouco mais de urgência que o jogador B. Uma possível matriz de ganho (satisfação do jogador) neste jogo poderia ser a representada pela Figura 4.

Note que neste jogo também há dois resultados estáveis. Em um a soma dos ganhos é igual a 12 e no outro a soma dos ganhos é igual a 10. Com isso, se medirmos a satisfação média dos jogadores (que nos dá um valor social), vemos que é preferível ter o resultado onde o jogador A está na máquina P e o jogador B está na máquina Q.

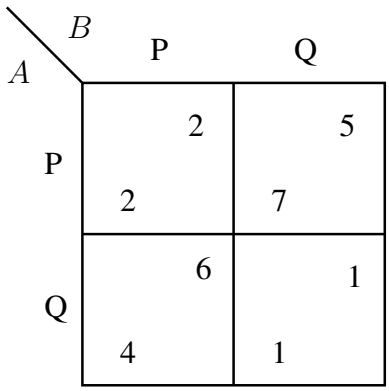


Figura 4: Congestionamento em Redes.

Cara ou Coroa. Neste jogo, conhecido em inglês por *Matching Pennies*, temos dois jogadores, cada um controla uma moeda e pode mostrar *Cara* ou *Coroa*. O jogador A ganha se os dois mostram as moedas com as mesmas faces e o jogador B ganha se as faces das duas moedas forem diferentes. A matriz deste jogo é dada na Figura 5 e o valor 1 representa ganho e -1 representa derrota. Quando

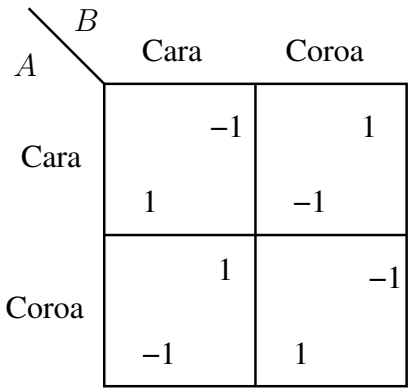


Figura 5: Cara ou Coroa.

todos os resultados de um jogo tiverem soma das utilidades iguais a uma constante, temos um *jogo de soma constante*. Note que a soma das utilidades de cada resultado no jogo Cara ou Coroa é igual a zero, e neste caso temos um *jogo de soma zero*. Da forma como estamos considerando até agora, este jogo não tem um um resultado estável. Para ver isso, note que em qualquer resultado há um vencedor e um perdedor, porém o perdedor pode mudar sua alternativa para um resultado onde vence.

Jogo de Congestionamento 2. O exemplo do jogo de controle de congestionamento pode ser adaptado para uma situação parecida com o jogo Cara ou Coroa, onde não há resultado estável. Considere a situação onde os jogadores A e B podem fazer suas transmissões de dados pelos pontos P e Q. Suponha agora que ambos pontos tem a mesma taxa de transmissão e o preço cobrado

pela empresa que controla estes pontos depende do tempo que levou para transmitir os dados. Se a empresa transmitir os dados em pouco tempo, ela cobra uma valor fixo e se ela demora para transmitir, ela não cobra pelo serviço. Agora, suponha que o jogador A precisa fazer a transmissão rapidamente e não pode transmitir a partir de um ponto congestionado, e consequentemente quer pagar por este serviço. Já o jogador B não tem dinheiro nem pressa e portanto só pode se conectar por um ponto congestionado. Se considerarmos que um ponto fica congestionado quando há dois jogadores transmitindo por ele, então este jogo não tem um resultado estável.

Tragédia dos Comuns. Neste jogo, temos vários jogadores que obtêm mais vantagens atuando de maneira racional, mas tal comportamento levará o jogo a um resultado com benefício muito aquém daquele obtido se todos fizessem um acordo para uma melhor utilização dos recursos.

Exemplificaremos esta situação com um problema de compartilhamento de largura de banda em um canal de transmissão.

Compartilhamento de Largura de Banda. Suponha que n jogadores querem transmitir dados por um cabo de transmissão que tem capacidade máxima 1 (largura de banda). O fluxo de dados usados por um jogador é a quantidade de largura de banda usada por ele durante a transmissão. Denotaremos por N o conjunto de jogadores e por $x_i \in [0,1]$ o fluxo do jogador $i \in N$. A medida que o fluxo total se aproxima da capacidade do cabo, a qualidade da transmissão se deteriora e com isso diminui o benefício dos jogadores. Neste modelo, se a banda total requisitada é $\sum_{j\in N} x_j > 1$ então o benefício para cada jogador é nulo. Caso $\sum_{j\in N} x_j \leq 1$ o benefício do jogador i é dado por $x_i(1 - \sum_{j\in N} x_j)$. Esta função de benefício mostra que cada jogador deve fazer um balanço entre seu fluxo e o fluxo total de transmissão.

Agora, considere um jogador i e sua escolha por um valor para o seu fluxo considerando que os outros jogadores já definiram seus fluxos. Vamos supor que $t = \sum_{j\in N\setminus\{i\}} x_j < 1$ é a quantidade de fluxo total usada pelos outros jogadores. Como o interesse de i é maximizar seu benefício, ele escolhe seu fluxo como o valor x que maximiza $x(1-t-x)$. Resolvendo isso, ele obtém o valor $x_i = (1-t)/2$. Por outro lado, como todos os jogadores são racionais e desejam maximizar seu benefício, todos farão estas mesmas contas e portanto teremos um resultado estável se $x_i = (1 - \sum_{j\in N\setminus\{i\}} x_j)/2$ para todo i. Estas equações nos levam a um sistema com solução única dada por $x_i = 1/(n+1)$, para todo jogador i.

Portanto, o benefício de cada jogador i é $x_i(1 - \sum_{j\in N} x_j) = 1/(n+1)^2$ o que dá um benefício total de $n/(n+1)^2 \approx 1/n$. Por outro lado, se cada jogador definir seu fluxo como $x_i = 1/(2n)$ temos um fluxo viável onde cada jogador tem benefício de $1/(4n)$ resultando em um benefício total de $1/4$, que são valores muito melhores que os obtidos no resultado estável (aproximadamente $n/4$ vezes maior).

Jogos Sequenciais. Até agora, consideramos que cada jogador faz sua escolha sem o conhecimento das escolhas dos outros jogadores. Uma outra forma de jogo é quando os jogadores se alternam na

escolha de suas decisões. Para exemplificar esta situação, considere o jogo de Compartilhamento de Largura de Banda, visto anteriormente. Suponha que, por restrições no sistema, não é possível ou fica muito custoso para o jogador saber a quantidade de fluxo que cada jogador está enviando, mas ele sabe apenas o fluxo total que está sendo enviado pelo canal em cada momento. Além disso, suponha que um jogador possa modificar a quantidade de fluxo a cada momento para obter um melhor benefício individual. Por simplicidade, vamos supor que as atualizações dos jogadores são realizadas em alguma ordem (i.e., dois jogadores não atualizam seus fluxos ao mesmo tempo).

Sabendo o valor do fluxo atual, um jogador i pode facilmente calcular o fluxo total dos outros jogadores, dado por $t = \sum_{j \in N \setminus \{i\}} x_j$. Neste momento o jogador calcula o valor do fluxo x que maximiza seu benefício, de $x(1-t-x)$, obtendo o valor $x_i = (1-t)/2$. Naturalmente, a mudança feita pelo jogador i pode ser considerada pelos próximos jogadores, que farão também mudanças nos seus fluxos para obter um maior benefício. Se os jogadores forem ajustando seus fluxos um após o outro, de maneira sequencial, o fluxo final convergirá na mesma solução vista anteriormente, com benefício global próximo de $1/n$ e portanto muito aquém daquela solução de benefício total $1/4$.

Jogos Repetidos. Podemos também considerar jogos que são repetidos entre os jogadores várias vezes. Isto é, há várias partidas de um mesmo jogo entre os mesmos jogadores. Além disso, os jogadores podem manter um histórico das partidas anteriores que poderiam ser usadas para a tomada de suas decisões. O custo obtido neste jogo é o custo total obtido em todas as partidas. Considere por exemplo o jogo do Dilema dos Prisioneiros. Note que se o número de partidas é fixo, é sempre preferível para um prisioneiro confessar na última partida, pois diminuirá o número de anos na cadeia nesta partida. Isto também pode ser aplicado à penúltima partida, assim por diante. Com isso, se o número de partidas for repetido um número finito de vezes, temos um resultado estável também onde os prisioneiros apenas confessam. Por outro lado, se o número de partidas for infinito ou não for conhecido, pode valer a pena aos jogadores ficar em silêncio. Isto ocorre pois o histórico de escolhas forma uma reputação do jogador que poderá ser usada pelo outro prisioneiro para decisões futuras. Com isso, pode valer a pena um prisioneiro ficar em silêncio, sabendo que se trair confessando o crime ele poderá ser retaliado nas próximas partidas. Uma regra de escolha interessante que um jogador pode tomar é escolher cooperar na primeira partida e para as próximas partidas escolher exatamente a escolha do outro prisioneiro na partida imediatamente anterior. Neste caso, se ambos prisioneiros escolherem inicialmente ficar em silêncio, ambos continuarão em silêncio para sempre. Se um prisioneiro escolher confessar e o outro não, o prisioneiro que não confessou irá retaliar na próxima partida. Esta estratégia é conhecida como *Olho por Olho*, ou em inglês *Tit for Tat*, e foi adaptada em protocolos de redes distribuídas par-a-par (*peer-to-peer*), exemplificado a seguir.

Transferências em sistemas Peer-to-Peer. Neste tipo de sistema, um usuário pode transferir arquivos da Internet para seu computador mas também tem incentivo a disponibilizá-los a partir

de seu computador, minimizando o tráfego no geral, já que outros usuários terão mais alternativas de onde obter este arquivo. Naturalmente um usuário pode não querer disponibilizar o arquivo de seu computador, tornando-se assim, um aproveitador do sistema. Para incentivar que os usuários também disponibilizem os arquivos, muitos protocolos mantém uma reputação dos usuários que melhora quando o usuário disponibiliza os arquivos do seu computador e piora quando ele não disponibiliza o arquivo. Além disso, sempre que o número de usuários chegar ao máximo o sistema dará preferência a um novo usuário com melhor reputação e interromperá a transmissão de um usuário com reputação pior. Com isso, usuários tentarão cooperar, para não sofrerem retaliações no futuro.

2.1 Conceitos básicos em Teoria dos Jogos

Os exemplos vistos são problemas modelados na área de Teoria dos Jogos. Apesar de existirem jogos com quantidade infinita de jogadores, nos restringiremos neste texto a jogos onde há quantidade finita de jogadores.

Durante um jogo, cada jogador i possui um conjunto de escolhas possíveis que pode fazer, digamos dado pelo conjunto S_i. É uma convenção da área de Teoria dos Jogos chamar cada escolha em S_i de *estratégia* do jogador i. Quando o conjunto de jogadores, digamos N, estiver bem definido e claro pelo contexto, muitas vezes omitiremos o conjunto N em somatórias e produtórios, entre outros, para não sobrecarregar a notação. Com isso, os termos $\sum_i x_i$ e $\Pi_i x_i$ são iguais aos termos $\sum_{i \in N} x_i$ e $\Pi_{i \in N} x_i$, respectivamente.

Denotaremos por $S = \times_i S_i$ o conjunto de vetores de escolhas dos jogadores. Assim, se o conjunto de jogadores é dado pelo conjunto $N = [n]$, onde $[n] = \{1, \ldots, n\}$, o conjunto S é dado pelo conjunto $S_1 \times \ldots \times S_n$. Cada elemento $s \in S$ representa um resultado do jogo, também chamado de *vetor de estratégias* (ou perfil de estratégias) do jogo. Além disso, cada jogador deve ter uma ordem de preferência, que deve ser uma relação completa, transitiva, reflexiva e binária sobre os possíveis resultados do jogo. É esta relação que dirá que o jogador prefere mudar sua estratégia, e com isso o resultado do jogo muda de um vetor de estratégias para outro. Uma maneira simples de representar isso será através de uma função de utilidade (ou benefício) $u_i : S \to \mathbb{R}$ para cada jogador i, que devolve um valor numérico para cada vetor de estratégias. Assim, se i mudou de estratégia, migrando de um vetor de estratégias s para um vetor s', então isso só ocorreu porque $u_i(s') > u_i(s)$. Observe que a função de utilidade de um jogador está definida para cada $s \in S$ e portanto a utilidade ou benefício de um resultado para um jogador depende também das escolhas dos outros jogadores. Naturalmente, podemos também reescrever isto usando uma função de custo (ou penalidade), em vez da função de utilidade. Dado uma função de utilidade u, podemos definir uma função de custo $c : S \to \mathbb{R}$ fazendo $c_i(s) = -u_i(s)$. Neste caso, o jogador deseja minimizar seu custo.

Definição 1 *Um jogo J consiste de um conjunto N de jogadores e conjunto de estratégias S_i e função de utilidade u_i sobre o conjunto de vetores de estratégias do jogo, para cada jogador i.*

Na definição acima, os conjuntos de estratégias nem sempre serão dados listando todos seus elementos, como fizemos por matrizes de utilidade e custo. No jogo Compartilhamento de Largura de Banda, por exemplo, cada jogador tem infinitas estratégias. Mesmo quando o conjunto de estratégias de cada jogador for finito, pode ser inviável representar todos os vetores de estratégia explicitamente, como veremos na Seção 3.

Quando um vetor v estiver indexado no conjunto de jogadores N, a notação padrão na área é definir v_i como o elemento do vetor v para o jogador i e por v_{-i} o vetor v removendo se a posição de i. Dado estratégia v'_i do jogador i, denotamos por (v'_i, v_{-i}) o vetor obtido de v trocando o elemento v_i por v'_i. Assim, se o conjunto de jogadores é dado por $N = \{1, \ldots, n\}$ e temos dois vetores de estratégia $s = (s_1, \ldots, s_n)$ e $r = (r_1, \ldots, r_n)$, então as seguintes igualdades são válidas: $s = (s_i, s_{-i})$, $s_{-i} = (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n)$ e $(r_i, s_{-i}) = (s_1, \ldots, s_{i-1}, r_i, s_{i+1}, \ldots, s_n)$.

Exemplo 1 *No jogo Dilema dos Prisioneiros, temos:*

(i) Um conjunto com dois jogadores $N = \{A, B\}$.

(ii) Conjuntos de estratégias para cada jogador:

$$S_A = \{\textit{Confessa},\ \textit{Silêncio}\},$$
$$S_B = \{\textit{Confessa},\ \textit{Silêncio}\}.$$

(iii) As preferências de cada preso são dadas pelas funções de custo c_A e c_B, que dão a quantidade de anos que ficarão presos em cada resultado, apresentadas na matriz de custo da Figura 1-(a).

Assim, se o par (s_A, s_B) representa a célula onde o jogador A escolhe a estratégia s_A e o jogador B escolhe a estratégia s_B, então, dados os vetores de estratégias $s = (\textit{Silêncio}, \textit{Confessa})$ e $s' = (\textit{Confessa}, \textit{Confessa})$ temos, que $c_A(s) = 5$, $c_B(s) = 1$, $c_A(s') = 4$, $c_B(s') = 4$. Com isso, o jogador A prefere o vetor de estratégias s' ao vetor de estratégias s, pois $c_A(s) > c_A(s')$.

Vimos que no jogo Dilema dos Prisioneiros, é sempre preferível um jogador escolher a estratégia *Confessa*. Quando um jogador possuir uma estratégia que é sempre preferível, independente das escolhas dos outros jogadores, a chamaremos de *estratégia dominante*. Quando cada jogador possuir uma estratégia dominante, o jogo é dito ser com estratégias dominantes. Apesar de ser uma situação desejável, muitos jogos não são com estratégias dominantes.

2.2 Equilíbrio de Nash em jogos com estratégias puras

Nos jogos que vimos até agora, cada jogador faz a cada momento a escolha de apenas uma estratégia. Cada uma destas estratégias é chamada de estratégia pura e estes jogos são chamados de *jogos com estratégias puras*. Quando os jogadores atingem um resultado onde cada jogador não tem interesse em mudar sua estratégia, atingimos um resultado estável. Diremos que um vetor de estratégias que representa tal resultado é um *equilíbrio de Nash em estratégias puras*. Por exemplo, no Dilema dos Prisioneiros, o resultado onde ambos confessam é um equilíbrio de Nash, pois estando neste resultado, nenhum deles diminui seus anos de cadeia ao mudarem (individualmente) suas escolhas. Formalmente temos:

Definição 2 *Um vetor de estratégias $s \in S$ é dito estar em* equilíbrio de Nash em estratégias puras*, se para todo jogador i e estratégia $s'_i \in S_i$, temos $u_i(s_i, s_{-i}) \geq u_i(s'_i, s_{-i})$.*

Isto é, um vetor de estratégias s está em equilíbrio de Nash em estratégias puras, se para todo jogador i a mudança da sua estratégia s_i para uma estratégia s'_i não melhora sua utilidade (mantendo fixas as estratégias s_{-i} dos outros jogadores). Durante um jogo, vamos dizer que um jogador i está *satisfeito* em relação a um vetor de estratégias atual s se $u(s'_i, s_{-i}) \leq u(s)$, para todo $s'_i \in S_i$. Caso contrário, diremos que i está *insatisfeito*. Claramente, em um equilíbrio de Nash em estratégias puras, todos os jogadores estão satisfeitos.

2.3 Equilíbrio de Nash em jogos com estratégias mistas

Uma outra maneira de definir o jogo é usando probabilidades para as estratégias puras de cada jogador. Com isso, a escolha do jogador é feita aleatoriamente dentre uma das estratégias puras, de acordo com uma distribuição de probabilidade. Neste caso, em vez de escolher uma das estratégias puras, cada jogador escolhe uma distribuição de probabilidade para suas estratégias puras. A partir da distribuição dos jogadores, temos também uma distribuição de probabilidade para os vetores de estratégias do jogo e a utilidade de cada jogador é obtida pelo valor esperado dada por esta distribuição.

Vamos formalizar isso para o caso de jogos com número finito de estratégias. Considere que os jogadores são dados pelo conjunto $N = \{1, \ldots, n\}$ e seja S_i o conjunto de estratégias do jogador i e p_i sua distribuição de probabilidade para os elementos de S_i (isto é, temos $p_i(s_i) \geq 0$ para todo $s_i \in S_i$ e temos que $\sum_{s_i \in S_i} p_i(s_i) = 1$). Se $p = (p_1, \ldots, p_n)$ é o vetor das distribuições de probabilidade dos jogadores, então o valor esperado da utilidade do jogador i é dada por $E[U_i(p)]$, onde

$$E[U_i(p)] = \sum_{s \in S} \Pi_j p_j(s) u_i(s).$$

Definição 3 *Um vetor $p = (p_1, \ldots, p_n)$ das distribuições de probabilidade dos jogadores é um equilíbrio de Nash em estratégias mistas, se para todo jogador i, temos que $E[U_i(p_i', p_{-i})] \leq E[U_i(p)]$, para toda distribuição de probabilidade p_i' sobre S_i.*

Assim, se um vetor p das distribuições de probabilidade dos jogadores está em equilíbrio de Nash, então não adianta o jogador i trocar sua distribuição p_i pois seu benefício esperado não irá aumentar.

Exemplo 2 *Vamos exemplificar esta definição com o jogo Cara ou Coroa (Matching Pennies). Vamos supor que os jogadores A (ganha se faces são iguais) e B (ganha se faces são diferentes) usam as distribuições de probabilidade p_A e p_B, respectivamente, onde*

$$p_A(Cara) = 1/3, \quad p_A(Coroa) = 2/3,$$

$$p_B(Cara) = 1/4, \quad p_B(Coroa) = 3/4.$$

Se $p = (p_A, p_B)$, a utilidade esperada de A, dada por $E[U_A(p)]$ é igual a

$$E[U_A(p)] = \frac{1}{3}\frac{1}{4}(+1) + \frac{1}{3}\frac{3}{4}(-1) + \frac{2}{3}\frac{1}{4}(-1) + \frac{2}{3}\frac{3}{4}(+1) = \frac{1}{6}.$$

Fazendo as contas da utilidade esperada de B, temos $E[U_B(p)] = \frac{-1}{6}$. Note que com estas distribuições, a chance do jogador A ganhar é maior.

Agora, considere o caso onde o jogador A escolhe cada uma de suas estratégias com probabilidade $\frac{1}{2}$ e o jogador B escolhe uma outra distribuição qualquer, digamos dado pelas probabilidades ρ e $(1-\rho)$, onde $0 \leq \rho \leq 1$ de escolher cara e coroa, respectivamente. Neste caso, $p_A = (1/2, 1/2)$, $p_B = (\rho, 1-\rho)$ e $p = (p_A, p_B)$. A utilidade de A dada por esta distribuição é igual a

$$E[U_A(p)] = \frac{1}{2}\rho(+1) + \frac{1}{2}(1-\rho)(-1) + \frac{1}{2}\rho(-1) + \frac{1}{2}(1-\rho)(+1) = 0.$$

Assim, ao usar $p_A = (1/2, 1/2)$, não importa qual a distribuição escolhida por B que a utilidade esperada do jogador A é igual a 0. O mesmo ocorre se B também escolher $p_B = (1/2, 1/2)$. Com isso, se ambos jogadores escolherem estas distribuições p_A e p_B, nenhum dos jogadores irá conseguir melhorar a utilidade esperada com outra distribuição, e portanto o vetor $p = (p_A, p_B)$ é um equilíbrio de Nash em estratégias mistas.

Note que um jogo com estratégias puras é um caso particular de jogo com estratégias mistas, onde a distribuição de cada jogador é definida dando probabilidade 1 para a estratégia escolhida pelo jogador e 0 para as demais. Assim, notamos que o jogo Cara ou Coroa apresenta um equilíbrio de Nash em estratégias mistas, mas o mesmo não é verdade em estratégias puras. O próximo teorema, provado por Nash, é um dos resultados de maior impacto na área de Teoria dos Jogos [29].

Teorema 1 *Todo jogo com número finito de jogadores e estratégias possui um equilíbrio de Nash em estratégias mistas.*

3 Complexidade Computacional de se Encontrar um Equilíbrio

3.1 Preliminares

Existem diversas classes de complexidade computacional investigadas na literatura. Tais classes foram definidas considerando-se o que é computável a partir de um certo modelo e em quantos passos computacionais. Nesta seção, iremos considerar modelos de computação e notação usual em análise e projeto de algoritmos.

Quando medimos a complexidade computacional de algoritmos, a maneira padrão é defini-la em termos do tamanho (*e.g.* número de bits) da instância. No caso da complexidade de tempo computacional, deve-se apresentar uma função que dado o tamanho da instância nos dá o número de passos do algoritmo dentro do modelo computacional considerado. No desenvolvimento da área, os algoritmos de tempo polinomial apresentam na sua grande maioria um bom desempenho computacional e com isso são chamados de *algoritmos eficientes*; apesar de algoritmos com complexidades de tempo dadas por um polinômio de grau alto, como $O(n^{100})$, não serem práticos (felizmente eles raramente ocorrem na prática). Por outro lado, há problemas para os quais ninguém conseguiu produzir algoritmos de tempo polinomial para resolvê-los.

Nesta busca, duas importantes classes de complexidade computacional se destacam: as classes P e NP. A classe P contempla os problemas de decisão (problemas com resposta SIM ou NÃO) para os quais há algoritmo de tempo polinomial para resolvê-lo. A classe NP são os problemas de decisão para os quais a resposta SIM pode ser verificada por um algoritmo de tempo polinomial usando um certificado de tamanho polinomial. Sabe-se que P $\subseteq$ NP, mas há grandes indicativos de que estes dois conjuntos sejam diferentes. Um dos motivos disto é a existência da classe NP-completo dentro da classe NP, para os quais os únicos algoritmos conhecidos para resolvê-los são algoritmos de tempo exponencial. Além disso, a classe NP-completo contém muitos problemas e caso *um* deles seja resolvido por um algoritmo de tempo polinomial, então *todos* os problemas de NP também serão resolvidos em tempo polinomial. A questão "P = NP ?" é um dos principais problemas em aberto da Ciência da Computação.

Para dar uma idéia de como funções exponenciais crescem rápido, comparados com funções polinomiais, vamos supor que temos um computador com velocidade de 1 Terahertz (mil vezes mais rápido que um computador de 1 Gigahertz) e funções que para cada tamanho n nos dão o número de instruções executadas neste computador. A Tabela 1 mostra os tempos obtidos para

algumas funções polinomiais e exponenciais, onde os tempos são dados em segundos (seg), dias e séculos (séc). Nota-se, para este exemplo, que o tempo computacional dado pelas funções de tempo exponencial crescem muito mais rapidamente que as funções de tempo polinomial.

$f(n)$	$n = 20$	$n = 40$	$n = 60$	$n = 80$	$n = 100$
n	$2{,}0 \times 10^{-11}$seg	$4{,}0 \times 10^{-11}$seg	$6{,}0 \times 10^{-11}$seg	$8{,}0 \times 10^{-11}$seg	$1{,}0 \times 10^{-10}$seg
n^2	$4{,}0 \times 10^{-10}$seg	$1{,}6 \times 10^{-9}$seg	$3{,}6 \times 10^{-9}$seg	$6{,}4 \times 10^{-9}$seg	$1{,}0 \times 10^{-8}$seg
n^3	$8{,}0 \times 10^{-9}$seg	$6{,}4 \times 10^{-8}$seg	$2{,}2 \times 10^{-7}$seg	$5{,}1 \times 10^{-7}$seg	$1{,}0 \times 10^{-6}$seg
2^n	$1{,}05 \times 10^{-6}$seg	1,1seg	13,3 dias	383,1séc	$4{,}02 \times 10^{8}$séc
3^n	$3{,}5 \times 10^{-3}$seg	140,7 dias	$1{,}3 \times 10^{7}$séc	$4{,}7 \times 10^{16}$séc	$1{,}6 \times 10^{26}$séc

Tabela 1: Comparação de algumas funções de tempo computacional.

Uma possibilidade para tratar algoritmos com alta complexidade de tempo é usar computadores muito mais potentes para obter a resolução de instâncias maiores. Mesmo assim, esta estratégia não nos leva a resolver instâncias muito maiores quando executamos algoritmos de tempo exponencial. Suponha por exemplo que um problema está sendo resolvido em um computador atual, mas para tentar resolver instâncias maiores, compramos um supercomputador mil vezes mais rápido. Se um algoritmo com tempo de execução n^2 conseguir resolver instâncias de tamanho N no computador atual, ele conseguirá resolver instâncias de tamanho $31{,}6N$ no supercomputador, no mesmo tempo. Por outro lado, se o algoritmo tiver tempo de execução 2^n, então ele só conseguirá executar instâncias de tamanho $N + 9{,}97$ (apenas uma constante aditiva a mais) no supercomputador. Para mais detalhes, veja [17, 28].

Diante disso, mostrar que um problema é NP-completo é um bom indicativo de que ele é intratável computacionalmente (para instâncias grandes), dado que o tempo computacional cresce muito rápido em relação ao tamanho da instância. Apesar desta comparação ter sido feita com tempo computacional, ela também vale para outros tipos de recursos, como quantidade de memória, processadores, etc. Depois que Cook provou que o problema Sat é NP-completo [7], vários outros foram provados serem NP-completos, veja [8, 17, 2]. Além disso, muitos destes problemas são simplificações de problemas práticos da área. A seguir, listamos alguns destes problemas, que usaremos posteriormente e cujas provas podem ser encontradas em [17].

Uma fórmula booleana $\phi(x_1, \ldots, x_n)$ sobre variáveis booleanas $x_1, \ldots, x_n$ está em forma normal conjuntiva se ϕ é uma conjunção de cláusulas, onde cada cláusula é uma disjunção de literais. Uma literal é uma variável ou sua negação. Por exemplo, a fórmula $\phi(x, y, z) = (x \vee y) \wedge (\overline{x} \vee \overline{y} \vee z) \wedge (y \vee \overline{z})$ está em *forma normal conjuntiva*.

Problema 1 (Sat) *Dada uma fórmula booleana $\phi(x_1, \ldots, x_n)$ em forma normal conjuntiva, decidir se há atribuição para as variáveis $x_1, \ldots, x_n$ de maneira a tornar $\phi(x_1, \ldots, x_n)$ verdadeira.*

Problema 2 (MaxSat-D) *Dado uma fórmula booleana $\phi(x_1, \ldots, x_n)$ em forma normal conjuntiva e um inteiro k, decidir se a fórmula contém uma atribuição que satisfaz pelo menos k cláusulas.*

Dado grafo não-orientado $G = (V, E)$, um conjunto de vértices $S \subseteq V$ é dito ser um *conjunto independente* de G se nenhum par de vértices $u, v \in S$ é ligado por uma aresta em G.

Problema 3 (Conjunto Independente-D) *Dado grafo $G = (V, E)$ não-orientado e inteiro k, decidir se existe conjunto independente $S \subseteq V$ de cardinalidade k.*

Problema 4 (Partição) *Dado conjunto de inteiros S, decidir se há conjunto $T \subseteq S$ tal que $\sum_{x \in T} x = \sum_{y \in S \setminus T} y$.*

Teorema 2 *Os problemas Sat, MaxSat-D, Conjunto Independente-D e Partição são* NP*-completos.*

Apesar da classe NP ser de problemas de decisão, ela contempla vários problemas que são simplificações de problemas de busca e de otimização combinatória. Convencionou se chamar de problemas NP-difíceis os problemas que são tão difíceis quanto os problemas NP-completos usando reduções de tempo polinomial. Assim, o problema de se encontrar uma atribuição para variáveis booleanas de maneira a satisfazer uma fórmula em forma normal conjuntiva é um problema NP-difícil. Note que no problema de decisão só queremos decidir se existe alguma atribuição que satisfaz a fórmula, sem se preocupar em qual atribuição de fato a satisfaz. O problema de otimização do conjunto independente é definido como:

Problema 5 (Conjunto Independente) *Dado grafo $G = (V, E)$ não-orientado, encontrar um conjunto independente $S \subseteq V$ de cardinalidade máxima.*

Este é um típico problema de otimização combinatória, onde queremos encontrar uma certa estrutura (no caso um conjunto independente) que minimiza ou maximiza certa função (no caso a cardinalidade do conjunto independente). Certamente se soubermos qual o tamanho de um conjunto independente de cardinalidade máxima, então saberemos também responder a pergunta se existe algum conjunto independente de cardinalidade k. Para mais detalhes sobre modelos de computação, classes de complexidade e otimização combinatória, veja [8, 17, 33].

Na prática existem diversos problemas NP-difíceis, como problemas de escalonamento de tarefas, balanceamento de carga, projeto de redes de telecomunicações e circuitos VLSI, roteamento de veículos, empacotamento de objetos em *containers*, localização de centros distribuidores, alinhamento de DNA e proteínas, classificação de dados, etc [39, 5].

Uma vez que um problema é identificado ser NP-difícil sabemos que as chances de existir um algoritmo eficiente (de tempo polinomial) para resolvê-lo é remota. Com isso, podemos de maneira mais conciente, fazer algoritmos exatos mas sabendo que dificilmente eles resolverão instâncias

grandes. Para instâncias maiores um algoritmo de tempo exponencial pode ser inviável e nos sugere que devemos procurar por algoritmos eficientes, mas que não necessariamente cobrem todas as restrições do problema. No caso dos problemas de otimização é comum fazer algoritmos eficientes que encontram "soluções" que satisfazem todas as restrições do problema, exceto possivelmente de ser uma com valor da função objetivo mínimo, para problemas de minimização, ou de valor máximo, para problemas de maximização. Chamaremos estas soluções de *soluções viáveis* do problema. Por outro lado, é desejado que tal solução não se distancie muito de uma solução exata. Uma das principais abordagens neste caso é desenvolver um algoritmo de aproximação.

Definição 4 *Dizemos que um algoritmo $\mathcal{A}$ para um problema de otimização tem um fator de aproximação α se a seguinte desigualdade vale*

$$\alpha \geq \max\left\{\frac{\mathcal{A}(I)}{\mathrm{OPT}(I)}, \frac{\mathrm{OPT}(I)}{\mathcal{A}(I)}\right\}, \quad \textit{para toda instância } I,$$

onde $\mathcal{A}(I)$ (resp. $\mathrm{OPT}(I)$) é o valor da solução obtida pelo algoritmo $\mathcal{A}$ (resp. de uma solução ótima) para a instância I.

Assim, se um algoritmo é uma 2-aproximação para um problema de minimização, ele devolve soluções que são no pior caso duas vezes a ótima, para *toda* instância do problema.

É importante observar que provar que um algoritmo de aproximação é α-aproximado, não quer dizer que ele sempre irá devolver soluções com valor próximos deste fator, mas sim que nunca atingirá um desempenho pior que este fator. Ademais, a busca por melhores fatores de aproximação força que a estrutura combinatória dos problemas seja melhor investigada, e com isso incentiva o surgimento de técnicas e ferramentas capazes de explorá-las. Um exemplo deste comportamento pode ser visto no problema de alocação de recursos (*facility location problem*) para o qual foram apresentados algoritmos com fatores de aproximação 1,861 e 1,61 e que produziram resultados, em poucos segundos, com distância média do valor da solução ótima de 3% e 1%, respectivamente [22]. Para ver mais sobre algoritmos de aproximação, sugerimos a leitura dos livros [8, 39, 5].

A partir desta introdução, veremos nas próximas seções sobre a complexidade de se encontrar equilíbrios de Nash em estratégias puras e mistas. No decorrer do texto, veremos que a abordagem de aproximação é usada tanto nas medidas do equilíbrio como no desenvolvimento de mecanismos eficientes.

3.2 Jogos com estratégias puras

Considere um jogo com estratégias puras e n jogadores, onde cada jogador i possui conjunto S_i de exatamente m estratégias. Se representarmos de maneira direta cada possível vetor de estratégias, então teremos m^n possíveis resultados para o jogo. Nesta representação, conhecida como *forma*

padrão, fica fácil verificar a existência de um equilíbrio. Para isto, basta verificar se algum vetor de estratégias s está em equilíbrio para todos os jogadores. Para verificar se s está em equilíbrio para um jogador i, basta percorrer os vetores de estratégias (s'_i, s_{-i}), para todo $s'_i \in S_i \setminus \{s_i\}$, e verificar se uma destas $m-1$ estratégias dá a ele uma utilidade maior. Como temos n jogadores, a verificação de s pode ser feita facilmente em tempo $O(nm)$. Com isso, a busca por um equilíbrio de Nash pode ser determinada com complexidade de tempo $O(nm^{n+1})$. Uma vez que a representação do problema já consumiu espaço $O(m^n)$, a complexidade de tempo obtida para verificar se há equilíbrio é polinomial em relação ao tamanho da instância. Por outro lado, tal representação só é possível para jogos que não são grandes. Note que mesmo quando cada jogador tem apenas duas estratégias disponíveis, isto já nos leva a uma complexidade de tempo de $O(n2^n)$.

Assim, é natural que muitos jogos computacionais não sejam dados na forma padrão, mas usem alguma estrutura adicional compacta, de preferência de tamanho polinomial no número de jogadores e estratégias. Nestes casos, a idéia é usar um algoritmo que para cada vetor de estratégias, use esta estrutura para dizer se ele está em equilíbrio, ou apresente uma estratégia melhor se não estiver. Veremos alguns exemplos deste tipo posteriormente, como no jogo de balanceamento de carga e no jogo de conexão global.

Para entender a complexidade computacional de se decidir a existência de um equilíbrio, considere um jogo gráfico (*graphical games*) onde os jogadores são representados pelos vértices de um grafo $G = (V, E)$ e as utilidades de um jogador i dependem apenas de suas estratégias e das estratégias dos vizinhos de i em G. Esta é uma representação interessante quando as escolhas de cada jogador dependem de um grupo restrito de jogadores. Em alguns casos o número de jogadores que influenciam um jogador é limitado por uma constante (i.e., o grau de cada vértice no grafo é limitado por constante). Assim, se o grau de cada vértice é limitado por uma constante k e cada jogador tem no máximo m estratégias, então o número de valores de utilidades diferentes para um jogador é limitado por m^{k+1}, o que é polinomial no número de estratégias. Portanto, o tamanho da instância neste jogo é limitado a nm^k. Contudo, apesar de ganharmos uma representação compacta para o jogo, mesmo quando m é constante o problema de se encontrar um equilíbrio de Nash é difícil [18].

Teorema 3 *Decidir a existência de equilíbrio puro de Nash em jogos gráficos é NP-completo, mesmo quando o grau de cada vértice é limitado a 3.*

Na literatura há várias outras situações onde decidir a existência de equilíbrio puro nos dá um problema NP-completo. Para uma discussão mais profunda sobre isso, veja [13].

3.3 Tempo de convergência em jogos com estratégias puras

Nesta seção iremos considerar o número de passos (escolhas realizadas durante o jogo) para se chegar a um equilíbrio de Nash em jogos sequenciais, onde cada jogador sabe apenas o resultado atual do jogo. Vimos na subseção anterior, que decidir a existência de um equilíbrio é um problema NP-completo e há jogos que não terminam em um equilíbrio de Nash. Um exemplo é o jogo Cara ou Coroa, que nunca atingia um equilíbrio de Nash em estratégias puras. Outro exemplo foi visto no jogo de Compartilhamento de Largura de Banda, onde havia um equilíbrio, mas os jogadores poderiam fazer melhorias de valores cada vez menores que de fato convergiam para um equilíbrio, sem no entanto chegar a ele em número finito de passos. Assim, vamos considerar jogos que sempre têm equilíbrios de Nash e os jogadores chegam a um equilíbrio em número finito de passos. O problema de atingir um equilíbrio de Nash em jogos com estratégias puras tem relação com a classe PLS (*Polynomial-time Local Search*). Esta classe foi definida para problemas de otimização combinatória que usam uma estrutura de vizinhança entre as soluções viáveis do problema [23]. A vizinhança é definida em geral através de pequenas alterações de uma solução viável para outra. Um algoritmo de busca local começa com uma solução viável e percorre suas soluções vizinhas. Caso uma das soluções vizinhas tenha valor estritamente melhor, então o algoritmo abandona a solução corrente e passa a usar uma daquelas soluções vizinhas com valor melhor. Este processo se repete até que se encontre uma solução sem vizinhos melhores; tal solução é chamada de *solução ótima local*. Por ser menos conhecida, mas de importância para o estudo da complexidade de jogos com estratégias puras, formalizaremos a classe PLS a seguir.

Em um *problema de minimização* Π temos: *(i)* um conjunto de instâncias I_Π e algoritmo de tempo polinomial que verifica se $x \in I_\Pi$; *(ii)* para cada instância $x \in I_\Pi$, temos um conjunto de soluções viáveis $F_\Pi(x)$; *(iii)* para toda solução $s \in F_\Pi(x)$ temos um custo $c_x(s)$; *(iv)* um algoritmo de tempo polinomial que diz se s pertence ou não à $F_\Pi(x)$ e em caso positivo, computa $c_x(s)$. O problema consiste em dado $x \in I_\Pi$, encontrar $s \in F_\Pi(x)$ tal que $c_x(s)$ é mínimo. Em um *problema de minimização local* Π temos as propriedades *(i)–(iv)* e mais as seguintes: *(v)* há uma vizinhança $N_x(s) \subseteq F_\Pi(x)$ para cada $x \in I_\Pi$ e $s \in F_\Pi(x)$; *(vi)* uma solução s de $F_\Pi(x)$ é dita ser um *mínimo local* se $c_x(s) \leq c_x(s')$ para todo $s' \in N_x(s)$. O objetivo é encontrar uma solução que é mínimo local.

A definição de *problema de maximização* e problemas de maximização local são análogos e assim deixaremos para o leitor sua descrição. Chamaremos de *problemas de otimização*, os problemas de minimização ou maximização e por *solução ótima local* as soluções que são mínimos locais ou máximos locais para os respectivos problemas.

Um problema de otimização local Π pertence à *classe* PLS se temos um algoritmo com complexidade de tempo polinomial que, para qualquer instância $x \in I_\Pi$ e solução $s \in F_\Pi(x)$, decide se s é ótima local, e se não for, devolve $s' \in N_x(s)$ com $c_x(s') < c_x(s)$, se for um problema de

minimização, ou $c_x(s') > c_x(s)$ se for um problema de maximização. Uma PLS-redução de um problema de otimização local Q para outro P consiste de: *(a)* funções h e g computáveis em tempo polinomial tal que h mapeia instâncias de $x \in I_Q$ para instâncias $h(x) \in I_P$ e g mapeia soluções de $h(x)$ para soluções de x; *(b)* para toda instância x de Q, se s é uma solução ótima local de $h(x)$, então $g(s, x)$ é uma solução ótima local de x. Um problema P é PLS-*completo* se está em PLS e se para todo problema Q de PLS, há uma PLS-redução de Q para P [23].

O Max2Sat com vizinhança Flip é um problema PLS-completo e é definido através de uma vizinhança de soluções para o problema de otimização Max2Sat [35]. Definimos estes problemas a seguir.

Problema 6 (Max2Sat) *Dada fórmula booleana ϕ em forma normal conjuntiva, cada cláusula com um peso associado, encontrar uma atribuição lógica das variáveis de maneira a maximizar o peso total das cláusulas satisfeitas.*

Problema 7 (Max2Sat com vizinhança Flip) *Encontrar uma solução ótima local para o problema Max2Sat, onde a vizinhança é dada trocando o valor de uma das variáveis.*

Além deste, vários outros problemas de busca local, definidos sobre problemas de otimização mais gerais, como MaxSat, MaxCut, TSP, entre outros, foram provados serem PLS-completos. Se existir um algoritmo que sempre obtém um ótimo local em tempo polinomial para um destes problemas, então será possível obter algoritmos eficientes para encontrar uma solução ótima local para qualquer problema em PLS. Apesar de ser também uma classe grande, ninguém conseguiu mostrar a existência de um algoritmo de tempo polinomial que garantidamente obtém soluções ótimas locais para qualquer problema em PLS-completo. Para mais detalhes, veja [23, 35]. Isto mostra de certa maneira a dificuldade de se obter soluções ótimas locais eficientemente.

Na Subseção 4.3, veremos uma classe de jogos que sempre possuem equilíbrios de Nash, mas encontrá-los foi provado ser um problema PLS-completo. Assim, parece igualmente difícil obter maneiras de se explorar os movimentos deste jogo e garantir uma convergência para um equilíbrio de Nash em tempo polinomial.

Uma tentativa interessante de se contornar esta situação, é considerar que os jogadores escolhem, dentre todas as estratégias que melhoram sua utilidade, uma que devolve o maior benefício. Mais formalmente, vamos definir os conceitos de *resposta de melhoria* e *melhor-resposta*. Dado jogador i, sua função de utilidade u_i e vetor de estratégias s, dizemos que s'_i é uma resposta de melhoria para s se $u_i(s'_i, s_{-i}) > u_i(s)$. Isto é, se o jogador i mudar sua escolha de s_i para s'_i, sua utilidade irá aumentar. Naturalmente, dentre todas as respostas de melhoria, deve haver uma que dá a ele o maior incremento na utilidade. Assim, uma estratégia s'_i é uma *melhor-resposta* do jogador i se $u_i(s'_i, s_{-i})$ é máximo dentre todas as estratégias de melhoria s'_i de i.

Apesar de ser uma escolha interessante, pois dá ao jogador a chance de fazer o melhor possível dentro da sua visão local, pode haver jogos onde a melhor-resposta leva a um número exponencial de passos. Em particular, o jogo da Subseção 4.3 pode gastar uma quantidade exponencial de passos, mesmo usando estratégias de melhor-resposta.

3.4 Jogos com estratégias mistas

Nesta seção, iremos considerar jogos com número finito de jogadores e estratégias. Quando consideramos estratégias mistas, o Teorema de Nash nos garante que todo jogo nesta situação tem um equilíbrio de Nash. Porém, as demonstrações existentes deste resultado não são construtivas e não indicam como encontrar um equilíbrio, muito menos por algoritmos eficientes. Vamos denotar por NASH o problema de se encontrar um equilíbrio de Nash em estratégias mistas em jogos com número finito de jogadores e estratégias.

A teoria de NP-completude não parece ser adequada para investigar a complexidade computacional do problema NASH. Problemas da classe NP-completo pedem a existência de soluções com certas restrições e no caso do problema NASH, sempre há um equilíbrio de Nash.

O problema NASH foi provado ser PPAD-completo [9]. Não iremos definir esta classe, pois ela é pouco intuitiva e necessita de maior fundamentação teórica. É importante observar que a existência de um algoritmo de tempo polinomial para resolver NASH (neste caso teríamos P = PPAD) nos daria também algoritmos eficientes para vários outros problemas que são PPAD-completo e tem resistido a existência de tais algoritmos há décadas. Alguns destes problemas são os problemas de se encontrar pontos fixos de Brouwer e Borsuk-Ulam, problema *ham sandwich*, busca de equilíbrios de Arrow-Debreu em mercados, etc. Para mais informações sobre argumentos nesta linha, veja [9, 32].

4 Medidas do Equilíbrio

Nesta seção, iremos considerar jogos que sempre têm equilíbrios de Nash e os jogadores sempre chegam a um equilíbrio. Com isso, cabe perguntar *Como medir a qualidade de um equilíbrio de Nash ?*

No jogo Dilema dos Prisioneiros, o único equilíbrio existente é quando ambos confessam, o que dava a cada um uma pena de 4 anos. Por outro lado, se houvesse coordenação entre eles e ambos ficassem em silêncio, cada um ficaria preso por 2 anos. Nota-se facilmente a ineficiência qualitativa do resultado em equilíbrio, uma vez que há outro resultado onde ambos jogadores obtêm maior benefício. Este tipo de comparação pode ser interessante quando comparamos quantidades abstratas e preferências dos jogadores. Para muitos jogos, os custos e benefícios dos jogadores representam gastos, lucro ou outros valores mensuráveis. Com isso, podemos usar funções para medir a qualidade dos resultados, mesmo entre resultados que são pareto ótimos. Vamos considerar

que temos uma função de valoração, chamada de *função social*, que dado um resultado do jogo (vetor de estratégias), nos devolve um valor numérico. Esta função social representa a utilidade social (ou custo social) do resultado. Para diferir os casos onde queremos maximizar a utilidade social ou minimizar o custo social, usaremos também os termos *função de benefício social* ou *função de custo social*.

No jogo de Congestionamento 1, da Seção 2, havia dois equilíbrios, um com utilidade total 10 e outro com utilidade total 12. Neste caso, poderia ser mais interessante para a sociedade que fosse escolhido um equilíbrio onde a satisfação média entre os jogadores é maior (é equivalente a ter soma das utilidades maior). De fato, uma função social comum nestes jogos é a soma das utilidades (ou custos) dos jogadores. Este tipo de função é chamada de *função utilitária* do jogo. Um exemplo de função social que não é utilitária é uma que devolve a maior insatisfação/custo entre todos os jogadores (e não a soma dos custos). Este tipo de função é chamada de *função igualitária*. Naturalmente, podemos definir funções diversas para cada jogo, e sua definição terá impactos diretos nas medidas usadas em cada situação.

4.1 Preço da Anarquia e Preço da Estabilidade

Uma vez definida a função social do jogo, precisamos saber como os resultados em equilíbrio se comparam com os melhores resultados do jogo, definido por esta função social. Dada uma função social, um resultado do jogo é chamado de *resultado ótimo social* se a função aplicada ao resultado nos dá a maior utilidade/benefício (ou menor custo) dentre todos os possíveis resultados do jogo. Um resultado ótimo social pode inclusive não estar em equilíbrio.

Por exemplo, se considerarmos a função social do Dilema dos Prisioneiros, como a média dos anos que os prisioneiros ficam presos, o resultado ótimo social é aquele onde a média é de 2 anos. Já o resultado em equilíbrio tem a média de 4 anos, que é duas vezes maior que a solução ótima social.

Baseado neste tipo de comparação, definimos a seguir o *preço da anarquia* e o *preço da estabilidade*.

Se J é um jogo com conjunto de vetores de estratégias S e $f : S \to \mathbb{R}$ é uma função de custo social (resp. função de benefício social), denotamos por $\mathcal{E}(J)$ o conjunto de vetores de estratégias de J que estão em equilíbrio e $\mathrm{OPT}(J) = \min\{f(s) : s \in S\}$ (resp. $\mathrm{OPT}(J) = \max\{f(s) : s \in S\}$). Com isso, s é um resultado ótimo social se $f(s) = \mathrm{OPT}(J)$. O desejo, para o sistema como um todo, é que o resultado em equilíbrio atingido pelos jogadores esteja próximo do ótimo social, de preferência que tenha o mesmo valor.

O *preço da anarquia*, PA, de um jogo de minimização J é a maior razão entre o valor de um

resultado em equilíbrio e o valor de um resultado ótimo social [24]. Mais formalmente, temos que

$$\text{PA} = \max\left\{\frac{f(s)}{\text{OPT}(J)} : \text{ onde } s \in \mathcal{E}(J)\right\}.$$

Se J é um jogo de maximização a definição é análoga, e temos

$$\text{PA} = \max\left\{\frac{\text{OPT}(J)}{f(s)} : \text{ onde } s \in \mathcal{E}(J)\right\}.$$

O *preço da estabilidade*, PE, de um jogo de minimização J é a menor razão entre o valor de um resultado em equilíbrio e o valor de um resultado ótimo social [1]. Formalmente, temos

$$\text{PE} = \min\left\{\frac{f(s)}{\text{OPT}(J)} : \text{ onde } s \in \mathcal{E}(J)\right\}.$$

Se J é um jogo de maximização a definição é análoga, e temos

$$\text{PE} = \min\left\{\frac{\text{OPT}(J)}{f(s)} : \text{ onde } s \in \mathcal{E}(J)\right\}.$$

Para todos os casos, temos que $\text{PA} \geq 1$ e $\text{PE} \geq 1$. O preço da anarquia, nos diz o quão longe o resultado em equilíbrio pode estar de um resultado ótimo social e o preço da estabilidade diz o quão perto ele pode estar deste resultado. Quando a instância J do jogo estiver clara pelo contexto, usaremos apenas OPT como o valor de um resultado ótimo social.

4.2 Jogo de Balanceamento de Carga

O problema de balanceamento é um problema básico em sistemas computacionais e foi bastante investigado na literatura como um problema de otimização em escalonamento de tarefas. A versão para jogos generaliza o problema de congestionamento, apresentado como exemplo do jogo Batalha dos Sexos. Quando temos vários jogadores e pontos de acesso, cada jogador procura transferir seus dados a partir de um ponto menos congestionado. Esta seção foi baseada no artigo [41].

Jogo de Balanceamento de Carga. Temos um conjunto de m máquinas e um conjunto de n tarefas, cada uma pertencente a um jogador. Para simplificar, usaremos um número i para indicar tanto a tarefa como seu correspondente jogador. Cada jogador deve alocar sua tarefa i, que tem um peso w_i, em uma das m máquinas. Um vetor de estratégias neste caso é simplesmente uma atribuição de tarefas para máquinas, $A : [n] \to [m]$, onde $A(j)$ nos dá a máquina escolhida pelo jogador j. Denotaremos por A_i o conjunto de tarefas atribuídas à máquina i e por ℓ_i o peso total das tarefas em i, isto é, $\ell_i = \sum_{j \in A_i} w_j$. O custo do jogador da tarefa j é o peso da máquina onde

j foi atribuído. Já o custo social de uma atribuição A é uma função igualitária dada pelo peso máximo de uma máquina, i.e., $c(A) = \max\{\ell_i : i \in [m]\}$. Chamaremos este custo de *makespan* da atribuição.

Exemplo 3 *Considere um resultado inicial do problema de balanceamento de carga dado na Figura 6-(a), onde o número dentro de cada item corresponde ao seu peso. Neste resultado o jogador do item de peso* 4 *tem um custo de* 6 *e tem incentivo para migrar para a máquina da direita, onde pagaria* 5*. O resultado após esta migração é dado na Figura 6-(b). Agora, o jogador do item de peso* 1*, que inicialmente tinha custo* 1*, fica com custo* 5 *e faz a migração para a máquina da esquerda, onde pagará* 3*. Após esta migração, temos o resultado da Figura 6-(c), que é uma configuração em equilíbrio.*

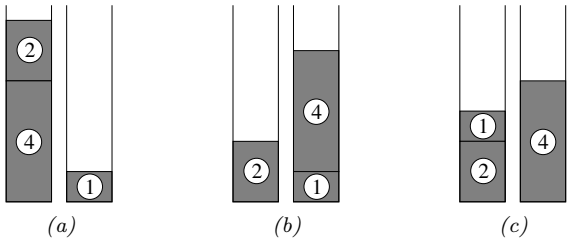


Figura 6: Sequência de movimentos até o equilíbrio.

Note que definimos este jogo de maneira sucinta onde a quantidade de dados de entrada é linear no número de jogadores (tarefas) e máquinas. Além disso, o custo de cada jogador, dada uma atribuição, é computada eficientemente em tempo linear. Se tivéssemos representado todos os vetores de estratégia possíveis de maneira tabular, teríamos m^n possibilidades. Certamente isso demandaria, para instâncias de tamanho médio ou grande, uma quantidade de memória e processamento proibitivos.

A versão de otimização do problema de balanceamento de carga é o problema de se buscar uma atribuição com *makespan* mínimo e é NP-difícil, mesmo para o caso onde há apenas duas máquinas [17]. De fato, note que se existir um algoritmo que produza um balanceamento de carga com *makespan* mínimo em tempo polinomial, então poderemos decidir o problema da Partição, dado como NP-completo no Teorema 2.

Este jogo não só tem um equilíbrio de Nash, mas sempre converge para um [16].

Teorema 4 *O jogo de balanceamento de carga sempre converge para um equilíbrio de Nash.*

Prova. No jogo de balanceamento de carga, uma atribuição A está em equilíbrio de Nash se e somente se todos os jogadores estiverem satisfeitos. Isto é,

$$\ell_{A(j)} \leq \ell_i + w_j, \quad \text{para todo jogador } j \text{ e máquina } i \neq A(j).$$

Considere uma atribuição A qualquer. Seja $\overleftarrow{w}(A) = (\lambda_1, \ldots, \lambda_m)$ o vetor das cargas das máquinas da atribuição A, em ordem não crescente de peso ($\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$). Considere um movimento de um jogador insatisfeito. Vamos mostrar que após o movimento deste jogador, a nova atribuição A' dará um vetor $\overleftarrow{w}(A')$ que é sempre lexicograficamente menor que $\overleftarrow{w}(A)$. Como o número de resultados possíveis é limitado (por m^n) não é possível ter um número infinito de passos, cada um obtendo um vetor lexicograficamente menor. Portanto, o jogo deve chegar a um equilíbrio de Nash em estratégias puras em quantidade finita de passos.

Agora, considere o vetor $\overleftarrow{w}(A) = (\lambda_1, \ldots, \lambda_m)$ e um jogador j insatisfeito que está em uma máquina i. Para simplificar, vamos renomear as máquinas de maneira que a carga λ_i é relativa a máquina i. Ao fazer um passo de melhoria, o jogador j tira sua tarefa de i e atribui para uma máquina k. Certamente $i > k$, pois as cargas (e máquinas) estão ordenadas de maneira não crescente. Apenas as máquinas i e k terão suas cargas modificadas após o movimento do jogador j. A escolha de k foi tal que mesmo somando a carga de j, a máquina k ficou com carga menor que a carga da máquina i antes do movimento. Isto é: $\lambda_k + w_j < \lambda_i$.

O movimento diminui a carga de i e aumenta a carga de k, mas a nova carga de k é ainda estritamente menor que a carga antiga de i. Como o número de máquinas com carga pelo menos λ_i continua o mesmo e a carga λ_i diminui, temos que o novo vetor de cargas ordenado de maneira não crescente, é lexicograficamente menor. □

Note que na prova, assumimos apenas que em cada passo um jogador insatisfeito movia sua tarefa para alguma máquina. Independente se o movimento era de melhoria ou melhor-resposta.

Além disso, note que se começarmos o jogo com uma solução ótima para o problema de otimização, obteremos um resultado em equilíbrio com o mesmo *makespan*. Para isto, basta começar o jogo a partir de um resultado ótimo do problema de otimização. Note pela prova do teorema anterior, que durante o jogo, o *makespan* nunca aumenta. Assim, encontrar um resultado em equilíbrio com *makespan* mínimo é um problema NP-difícil. O seguinte teorema é uma consequência direta deste fato.

Teorema 5 *Encontrar um melhor balanceamento de carga em equilíbrio é um problema* NP*-difícil.*

Prova. Para provar este resultado, vamos reduzir o problema da partição (veja Teorema 2) colocando cada inteiro da instância da Partição como uma tarefa com o mesmo peso. O problema da partição tem solução se e somente se há um balanceamento onde cada máquina tem o mesmo peso total. Além disso, um balanceamento onde cada máquina tem o mesmo peso total certamente está em equilíbrio. Portanto, a busca de um equilíbrio de custo social mínimo nos define também se o problema da partição tem solução. □

Tempo de Convergência para o Equilíbrio

Apesar do jogo de balanceamento de carga convergir para um equilíbrio, este pode ser dado em um tempo muito grande. Por enquanto sabemos apenas que o número de passos é no máximo o número de resultados, que é m^n. Em alguns casos é comum os jogadores usarem movimentos de melhor-resposta. O próximo teorema mostra que mesmo nestas condições o número de passos pode ser demasiadamente grande [12].

Teorema 6 *Existe uma instância do jogo balanceamento de carga com n tarefas e m máquinas e sequência de movimentos de melhor-resposta onde o equilíbrio de Nash é alcançado em pelo menos* $\left(\frac{n}{k^2}\right)^k$ *passos, onde* $k = m - 1$.

Note que para instâncias onde $k = \sqrt{n/4}$ temos pelo menos $2^{\sqrt{n}}$ passos. A prova do teorema envolve bastante detalhes, assim, daremos um exemplo que ilustra sua idéia para o caso de duas máquinas, onde todos os movimentos de melhoria são também de melhor-resposta. Para este caso de duas máquinas o jogo pode levar a uma sequência de $n^2/4$ movimentos para atingir um equilíbrio [12].

Exemplo 4 *Considere um jogo com duas máquinas e n tarefas dadas pelo conjunto* $\{1, \ldots, n\}$, *para n par, onde as tarefas* $2i - 1$ *e* $2i$ *tem peso* 3^i, *para* $i = 1, \ldots, n/2$. *Considere um resultado inicial onde todas as tarefas estão em uma das máquinas e em cada passo, migre a tarefa de um jogador insatisfeito que tiver o menor peso. Isto nos dará a quantidade total de* $n^2/4$ *movimentos, que deixaremos como exercício para o leitor. A seguir, apresentamos uma simulação para* $n = 6$.

O resultado inicial é dado pela Figura 7-(a). Neste resultado todos os jogadores estão insatisfeitos. Considerando que migramos sempre a tarefa do jogador insatisfeito com menor peso, migramos inicialmente a tarefa de peso 1. Isto nos leva ao resultado da Figura 7-(b). Continuando com as migrações das tarefas de peso 1 *e* 3, *chegamos no resultado da Figura 7-(e) com todos jogadores das tarefas de peso* 1 *e* 3 *satisfeitos. Agora, note que ao migrar uma das tarefas de peso* 9, *Figura 7-(f), todos os jogadores das tarefas de peso menor (pesos* 1 *a* 3*) ficam todos em uma máquina e voltam a ficar insatisfeitos. Nota-se que as tarefas de maior peso (peso* 9*) nunca mais voltarão a ficar insatisfeitas e com isso, voltamos a ter um resultado como o inicial, mas com 4 tarefas insatisfeitas* $(n - 2$ *para um caso geral).*

Por outro lado, a ordem em que são feitas as migrações é importante. Notamos que para qualquer resultado inicial podemos chegar a um equilíbrio no jogo de balanceamento de carga rapidamente [41]. A prova é relativamente simples e considera uma sequência onde a cada iteração, é feita a migração de uma tarefa mais pesada e que pertence a um jogador insatisfeito.

Teorema 7 *Para qualquer vetor de estratégias do jogo de balanceamento de carga com n tarefas e m máquinas, existe uma sequência de no máximo n passos de melhor-resposta para se atingir um equilíbrio de Nash.*

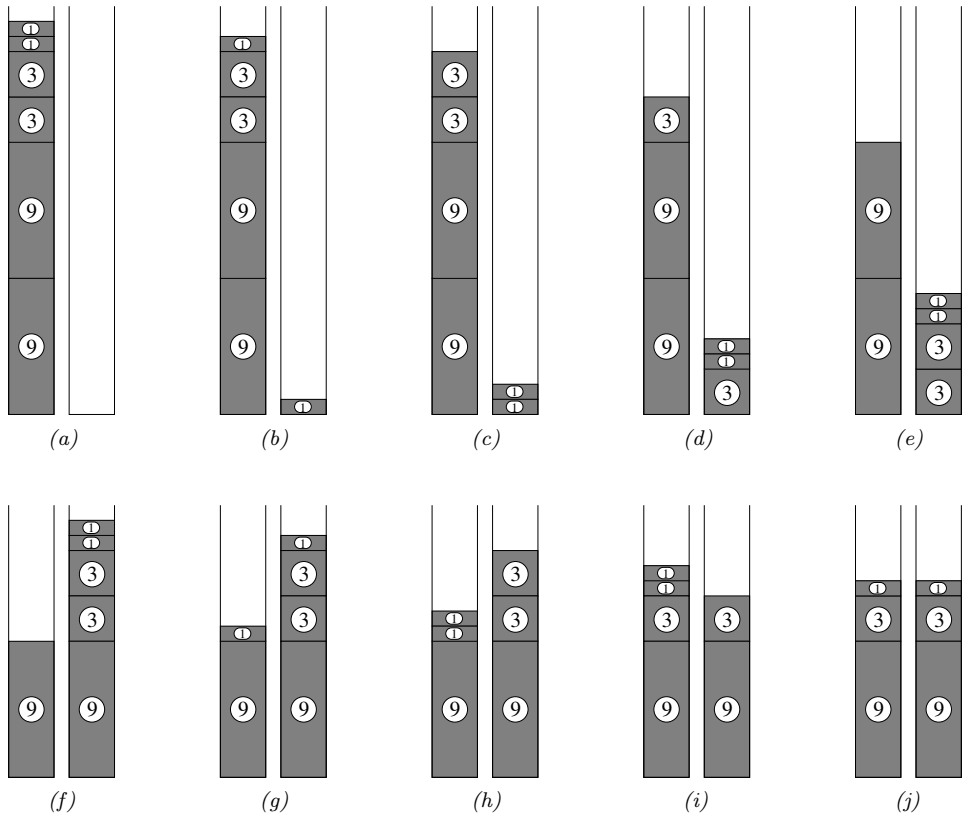


Figura 7: Sequência de pior movimentos em duas máquinas.

Como já comentado, se começarmos o jogo a partir de uma solução ótima para a versão de otimização (que obtém o resultado de menor custo social) o jogo termina em equilíbrio de Nash sem aumentar o *makespan*. O seguinte resultado é uma consequência direta deste fato.

Teorema 8 *O preço da estabilidade do jogo de balanceamento de carga é 1.*

O jogo de balanceamento foi investigado como um problema de busca local e foi mostrado ter preço da anarquia menor que 2 [15].

Teorema 9 *O preço da anarquia do jogo de balanceamento com n tarefas e m máquinas é no máximo* $2 - \frac{2}{m+1}$.

Prova. Seja A uma atribuição em equilíbrio e $c(A)$ o custo social desta atribuição. Seja i^* uma máquina mais carregada deste resultado (i.e., $c(A) = \ell_{i^*}$). Se i^* tem apenas uma tarefa, o resultado é claramente válido. Assim, vamos considerar que i^* tem pelo menos duas tarefas. Seja j uma tarefa de menor peso na máquina i^*. Como há pelo menos duas tarefas em i^* e j é de peso mínimo, temos que $w_j \leq \frac{\ell_{i^*}}{2}$.

Agora, considere uma máquina i diferente de i^*. Como a atribuição A está em equilíbrio, temos que $\ell_i \geq \ell_{i^*} - w_j$ para toda máquina i (caso contrário, a tarefa j poderia migrar para i). Com isso, temos que

$$\ell_i \geq \ell_{i^*} - w_j \geq \ell_{i^*} - \frac{\ell_{i^*}}{2} = \frac{1}{2}\ell_{i^*} = \frac{1}{2}c(A).$$

Agora, usando o fato que $\text{OPT} \geq \frac{\sum_j w_j}{m}$, temos

$$\begin{aligned} \text{OPT} &\geq \frac{\sum_j w_j}{m} \\ &= \frac{\sum_i \ell_i}{m} = \frac{\ell_{i^*} + \sum_{i \neq i^*} \ell_i}{m} \\ &\geq \frac{c(A) + \sum_{i \neq i^*} \frac{1}{2} c(A)}{m} \\ &= \frac{(m+1)c(A)}{2m}. \end{aligned}$$

Isolando $c(A)$, temos o limitante para o preço da anarquia. □

Observamos também que o preço da anarquia apresentado no Teorema 9 é justo [41]. Isto é, para cada valor de m, existe uma instância onde o preço da anarquia é exatamente $2 - \frac{2}{m+1}$.

4.3 Jogo de Conexão Global e Jogos Potenciais

Para ilustrar os jogos potenciais, iremos considerar um problema de formação de redes. Nestes jogos, os equilíbrios puros sempre existem e a dinâmica de melhor-resposta também converge para um equilíbrio de Nash. Esta seção foi baseada no artigo [38].

No jogo de conexão global os jogadores querem construir ligações (links) entre nós distantes. Cada jogador quer conectar um par de vértices de maneira barata e a cooperação entre os jogadores se dá quando uma mesma ligação é usada por vários jogadores e o custo de uma ligação é dividido igualmente entre seus usuários. Este mecanismo de dividir igualmente o custo é chamado de *compartilhamento de custos de Shapley.*

Exemplo 5 *Considere dois jogadores que querem construir rotas de conexão de um vértice a outro, cujas possíveis ligações são dadas pelo grafo da Figura 8-(a). O jogador* 1 *quer construir uma rota de ligação de s para t_1 e o jogador 2 quer construir uma rota de ligação de s para t_2. Se uma conexão local, dada por uma aresta do grafo, é usada pelos dois jogadores, sua construção é dividida igualmente entre os dois, caso contrário, sua construção é paga pelo jogador que usar a aresta. Além disso, considere que cada jogador i começou com um caminho (estratégia) de s para t_i apresentada pelas arestas grossas da Figura 8-(b), para $i = 1, 2$. Neste resultado, o jogador* 1 *paga* 4

e não é interessante mudar para outro caminho, que passa pelo vértice central, pois este custaria a ele 6 *(*$= 5+1$*). Já o jogador* 2 *está pagando* 8 *neste resultado e nota que se trocar seu caminho por outro que passa pelo vértice central, ele diminuirá seu gasto para* 6*, e portanto faz esta mudança, como mostrado na Figura 8-(c). Uma vez que o jogador* 2 *está usando o vértice central, é melhor para o jogador* 1 *também mudar de estratégia, pois agora ele poderá compartilhar a aresta de valor 5 (pagando metade dela) e pagaria menos pelo novo caminho. Assim, este jogador faz a mudança e ambos ficam satisfeitos no resultado da Figura 8-(d), cada um pagando* 3,5 *(*$= 5/2 + 1$*).*

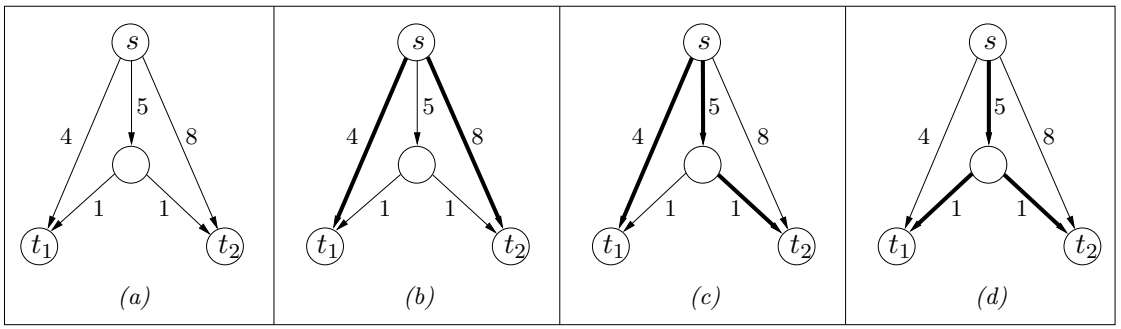


Figura 8: Evolução de um jogo de roteamento.

Jogo de Roteamento Global. Neste jogo, são dados k jogadores $\{1, \ldots, k\}$, um grafo orientado $G = (V, E)$, custo não negativo c_e para cada aresta $e \in E$ e pares de vértices s_i e t_i para cada jogador i. A estratégia do jogador i é um caminho P_i em G ligando s_i a t_i. Para simplificar a notação, dado um vetor de estratégias $P = (P_1, \ldots, P_k)$, usaremos P_i também como o conjunto de arestas no caminho considerado. Se P é um vetor de estratégias, $P = (P_1, \ldots, P_k)$, e k_e é o número de jogadores que usam a aresta e em P, então o custo para o jogador i em P é

$$c_i(P) = \sum_{e \in P_i} \frac{c_e}{k_e}.$$

Assim, o pagamento de cada aresta é dividida igualmente entre os usuários da aresta e o custo de um jogador é a soma dos pagamentos que realizou nas arestas do seu caminho. O custo social de $P = (P_1, \ldots, P_k)$ é o custo total de se construir as ligações de conexão, que é dado por

$$c(P) = \sum_{e \in E_P} c_e, \quad \text{onde } E_P = P_1 \cup \ldots \cup P_k.$$

Note que aqui também usamos uma representação compacta para representar o jogo, o que permitiu usar uma quantidade de memória polinomial no número de jogadores e no tamanho do grafo. Note que é inviável representar (para grafos medianos ou grandes) uma lista de todas as estratégias de um jogador explicitamente, pois cada estratégia de um jogador é um caminho que

liga seu vértice de origem ao vértice destino. Portanto, se tivermos um grafo com arestas de ida e volta entre cada par de vértices, qualquer sequência de vértices começando em s_i e terminando em t_i é uma estratégia para o jogador i. Armazenar todas estas estratégias certamente demandaria demasiada quantidade de memória e processamento.

Para este jogo, vamos considerar que a cada momento, um jogador realiza um movimento de melhor-resposta para o jogo. Note que para isso, o jogador i precisa resolver em cada passo do jogo um problema de caminho mínimo do vértice s_i ao vértice t_i. Este problema pode ser resolvido em tempo polinomial [11, 8] usando o mesmo grafo de ligações e pondo a distância de uma aresta como o valor a ser pago pelo jogador i ao usar esta ligação. Desta maneira um caminho mínimo de s_i a t_i nos dá o menor custo de uma rota em cada resultado.

Se temos k jogadores, é fácil ver que o preço da anarquia deste jogo está limitado a k.

Proposição 10 *O preço da anarquia do jogo de roteamento global é no máximo o número de jogadores. Isto é,* $\mathrm{PA} \leq k$.

Prova. Seja $P = (P_1, \ldots, P_k)$ um vetor de estratégias em equilíbrio, $P^* = (P_1^*, \ldots, P_k^*)$ um vetor de estratégias onde P_i^* é um caminho mínimo de s_i a t_i em G e O^* um vetor de estratégias que é um resultado ótimo social do jogo (P^* e O^* não necessariamente estão em equilíbrio). Como O^* deve conectar o vértice s_i ao vértice t_i, temos que $\sum_{e \in P_i^*} c_e \leq c(O^*)$. Além disso, temos que $c_i(P) \leq \sum_{e \in P_i^*} c_e$, pois caso contrário o jogador i mudaria sua estratégia para o caminho P_i^*. Com isso, temos

$$\begin{aligned} c(P) &= \sum_{e \in E_P} c_e = \sum_i c_i(P) \\ &\leq \sum_i \sum_{e \in P_i^*} c_e \\ &\leq \sum_i c(O^*) \\ &= k\, c(O^*) = k\, \mathrm{OPT}. \end{aligned}$$

□

Exemplo 6 *A Figura 9-(a) apresenta um exemplo que mostra que o limite da proposição 10 não pode ser melhorado. Neste exemplo temos k jogadores e todo jogador i quer obter uma conexão do vértice s ao vértice t (todos jogadores tem a mesma origem e o mesmo destino). O resultado ótimo social é usar a aresta de peso* 1*, que está claramente em equilíbrio com cada jogador pagando* $1/k$*. O resultado onde todos usam a aresta de peso k também é uma solução em equilíbrio e cada jogador paga* 1 *e portanto não tem incentivo em mudar (individualmente) para a outra aresta. É interessante observar que se houvesse uma coalizão de jogadores, estes poderiam migrar para a*

aresta de peso 1 e os demais fora da coalizão também migrariam para a aresta de peso 1, alcançando o menor equilíbrio.

Agora, considere o exemplo da Figura 9-(b) onde o jogador i quer conectar o vértice s (todos jogadores têm o mesmo vértice de origem) ao vértice t_i. O resultado ótimo social é aquele onde todos os jogadores usam a aresta de peso $1+\epsilon$, onde ϵ é um valor positivo bem pequeno, e depois seguem para as respectivas arestas de peso 0. Neste resultado cada jogador paga $(1+\epsilon)/k$. Apesar de ter um custo baixo, este resultado não está em equilíbrio, pois o jogador k está insatisfeito e pode pagar menos usando apenas a aresta de peso $1/k$. Após a migração do jogador k, ele fica satisfeito e os demais devem dividir o peso de $1+\epsilon$ entre os $k-1$ jogadores restantes, pagando $(1+\epsilon)/(k-1)$. Neste momento, o jogador $k-1$ fica insatisfeito, pois pode pagar menos usando apenas a aresta de peso $1/(k-1)$. Continuando desta maneira, vemos que o jogo termina em um resultado em equilíbrio de peso $H_k = 1 + \frac{1}{2} + \ldots + \frac{1}{k}$. Este exemplo mostra que o preço da estabilidade deve ser pelo menos H_k.

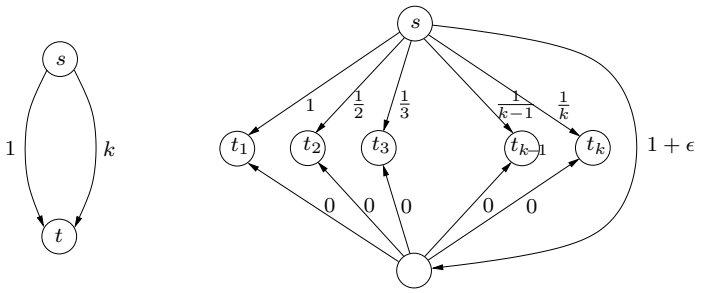


Figura 9: *(a)* Instância com PA $= k$. *(b)* Instância com PE $= H_k$.

A seguir, mostraremos que o preço da estabilidade deste jogo é no máximo H_k, usando o método da função potencial. Vamos definir o custo social de um vetor de estratégias s por $c(s)$ e o custo pago pelo jogador i ao participar do vetor de estratégias s como $c_i(s)$.

Definição 5 *Uma* função potencial exata Φ *é uma função que mapeia cada vetor de estratégias s para um valor real tal que para todo jogador i, temos*

$$\Phi(s_i, s_{-i}) - \Phi(s'_i, s_{-i}) = c_i(s_i, s_{-i}) - c_i(s'_i, s_{-i}), \quad \textit{para todo } s'_i \in S_i.$$

Neste caso dizemos que temos um jogo potencial.

Teorema 11 *Se um jogo tem número finito de estratégias e uma função potencial exata Φ, o jogo sempre converge para um equilíbrio de Nash.*

Prova. Considere um vetor de estratégias s. Se s não está em equilíbrio então há um jogador insatisfeito i. Considere a mudança de estratégia de s_i para s'_i pelo jogador i. Pela função potencial, temos $\Phi(s_i, s_{-i}) - \Phi(s'_i, s_{-i}) = c_i(s_i, s_{-i}) - c_i(s'_i, s_{-i})$. Como o jogador i diminuiu seu custo, temos que $\Phi(s_i, s_{-i}) > \Phi(s'_i, s_{-i})$ e portanto seu movimento fez o potencial do novo vetor de estratégias do jogo ficar estritamente menor que o do potencial anterior. O potencial do vetor de estratégias corrente sempre diminui a cada iteração e o jogo nunca passará pelo mesmo vetor de estratégias duas vezes. Portanto, se o jogo tem número finito de estratégias, ele deve convergir em algum momento para um equilíbrio de Nash. □

Para mostrar que o jogo de roteamento global sempre tem um equilíbrio de Nash, definimos a seguinte função potencial: Dado vetor de estratégias $P = (P_1, \ldots, P_k)$ do jogo de roteamento global, definimos a função potencial

$$\Psi(P) = \sum_{e \in E} c_e H(k_e),$$

onde k_e é o número de caminhos de P que usam a aresta e e $H(t) = 1 + \frac{1}{2} + \cdots + \frac{1}{t}$ para $t \geq 1$ e $H(0) = 0$.

Lema 12 *A função Ψ definida para o jogo de roteamento global é uma função potencial exata.*

Prova. Seja $P = (P_1, \ldots, P_k)$ um vetor de estratégias e $P' = (P'_i, P_{-i})$ o vetor de estratégias obtido após um movimento de melhoria do jogador i. Isto é, o jogador i trocou o caminho P_i pelo caminho P'_i. Seja também k_e e k'_e o número de caminhos usando a aresta e nos vetores de estratégia P e P', respectivamente. Para toda aresta e que pertence a P e P' ou que não pertença a P nem a P', o valor de k_e é o mesmo de k'_e e portanto $c_e H(k_e) = c_e H(k'_e)$. Com isso, temos

$$\begin{aligned}
\Psi(P) - \Psi(P') &= \sum_{e \in E} c_e H(k_e) - \sum_{e \in E} c_e H(k'_e) \\
&= \sum_{e \in E} (c_e H(k_e) - c_e H(k'_e)) \\
&= \sum_{e \in P_i \setminus P'_i} (c_e H(k_e) - c_e H(k'_e)) + \sum_{e \in P'_i \setminus P_i} (c_e H(k_e) - c_e H(k'_e)) \qquad (1) \\
&= \sum_{e \in P_i \setminus P'_i} \frac{c_e}{k_e} + \sum_{e \in P'_i \setminus P_i} \frac{-c_e}{k'_e} \\
&= \sum_{e \in P_i \setminus P'_i} \frac{c_e}{k_e} + \sum_{e \in P_i \cap P'_i} \frac{c_e}{k_e} - \sum_{e \in P'_i \setminus P_i} \frac{c_e}{k'_e} - \sum_{e \in P_i \cap P'_i} \frac{c_e}{k'_e} \qquad (2) \\
&= \sum_{e \in P_i} \frac{c_e}{k_e} - \sum_{e \in P'_i} \frac{c_e}{k'_e} \\
&= c_i(P) - c_i(P'),
\end{aligned}$$

onde (1) é válida pois $k_e = k'_e$ quando e não pertence a P nem a P' ou quando e pertence a P e P' e (2) é válida pois somamos $\sum_{e \in P_i \cap P'_i} \frac{c_e}{k_e}$ e subtraímos $\sum_{e \in P_i \cap P'_i} \frac{c_e}{k'_e}$ e nestas duas somatórias temos também que $k_e = k'_e$. □

Corolário 13 *O jogo de roteamento global com a dinâmica de melhor-resposta converge para um equilíbrio de Nash.*

Prova. Pelo Lema 12 temos que Ψ é uma função potencial exata e portanto, pelo Teorema 11, o jogo converge para um equilíbrio de Nash. □

O seguinte lema mostra que a função $\Psi(P)$ está próxima do valor do custo de S.

Lema 14 *Se $P = (P_1, \ldots, P_k)$ é um vetor de estratégias, então*

$$c(P) \leq \Psi(P) \leq H(k)c(P). \tag{3}$$

Prova. A primeira desigualdade de (3) segue direto da definição de Ψ e do fato que $H(k_e)$ é pelo menos 1 se e pertence a P e 0 caso contrário. Isto é,

$$c(P) = \sum_{e \in E_P} c_e \leq \sum_{e \in E_P} c_e H(k_e) = \sum_{e \in E} c_e H(k_e) = \Psi(P).$$

A segunda desigualdade de (3) segue do fato que $H(k_e) \leq H(k)$, uma vez que o número de caminhos que usam uma aresta é no máximo o número de jogadores.

$$\Psi(P) = \sum_{e \in E} c_e H(k_e) = \sum_{e \in E_P} c_e H(k_e) \leq \sum_{e \in E_P} c_e H(k) = H(k)c(P).$$

□

Teorema 15 *O preço da estabilidade do jogo de roteamento global é no máximo $H(k)$.*

Prova. Seja O^* um vetor de estratégias ótimo (ótimo social não necessariamente em equilíbrio); O um vetor de estratégias em equilíbrio obtido a partir de O^* e P um vetor de estratégia em Equilíbrio de Nash de menor custo. Com isso, temos

$$\begin{aligned} c(P) &\leq c(O) && (4)\\ &\leq \Psi(O) && (5)\\ &\leq \Psi(O^*) && (6)\\ &\leq H(k)c(O^*) = H(k)\mathrm{OPT}. && (7) \end{aligned}$$

A desigualdade (4) segue pois dentre os resultados em equilíbrio, P é a de menor custo. A desigualdade (5) segue do Lema 14. A desigualdade (6) vale pois O foi obtido de O^* após movimentos de melhoria, que só diminuem o valor do potencial. A última desigualdade segue novamente do Lema 14. □

O problema de busca do resultado ótimo social para o caso especial onde $s_i = s_j$, para $1 \leq i, j \leq k$, é o problema da árvore de Steiner em grafos orientados, que além de ser NP-difícil [17] é improvável ter fator de aproximação $(1 - \epsilon) \log |V|$ para qualquer $\epsilon > 0$ [14]. Notamos também que mesmo usando estratégias de melhor-resposta, o tempo de convergência do problema de roteamento global pode ser exponencial no tamanho da entrada, como mostrado em [1]. O seguinte teorema mostra a dificuldade de se obter um equilíbrio de Nash em jogos potenciais, mesmo usando a dinâmica de melhor-resposta [13].

Teorema 16 *O problema de encontrar um equilíbrio puro de Nash em jogos potenciais, onde a melhor-resposta é computada em tempo polinomial, é* PLS-*completo.*

O seguinte teorema, provado por Syrgkanis [36], estabelece a complexidade computacional de se encontrar um equilíbrio de Nash no jogo de roteamento global.

Teorema 17 *O problema de encontrar um equilíbrio de Nash no jogo de roteamento global é* PLS-*completo.*

Por fim, observamos que para grafos não-orientados, podemos definir outras formas de compartilhamento de custos onde o preço da anarquia é limitado a 2 [38].

5 Projeto Algorítmico de Mecanismos

Nesta seção apresentamos uma introdução ao projeto algorítmico de mecanismos. Um mecanismo é um processo algorítmico que escolhe uma solução social baseado nas preferências dos jogadores. Para isso, o mecanismo deve ter regras de funcionamento que incentivem os jogadores a declarar informações verdadeiras para que a escolha social seja feita adequadamente. Um problema quando lidamos com jogadores que tem interesses próprios e independentes, é que eles podem não dizer seus verdadeiros benefícios sobre os resultados e mentir para que possivelmente o jogo termine com um resultado mais favorável a ele. Este tipo de situação ocorre em vários problemas onde há vários participantes, como eleições e escolhas de lideres, definição de prioridades, alocação e pareamento por preferências, leilões, etc.

Neste texto nos restringiremos aos mecanismos cujos resultados, tanto de maneira social como individual, podem ser medidos numericamente. O texto desta seção foi baseado no artigo [30]. Primeiramente, veremos um problema de leilão que se encaixa neste tipo de jogo e apesar de simples, contempla as principais idéias envolvidas no projeto de mecanismos.

5.1 Leilão de Vickrey

Considere um leilão de um item único, onde há potenciais compradores (jogadores) que fazem lances para possuir o item e nenhum jogador sabe até quanto os outros jogadores poderão pagar pelo item. Um possível meio de se implementar tal leilão com apenas uma rodada de lances é fazendo com que cada jogador declare um valor em um envelope fechado e leva o item o jogador que tiver o maior valor, e este será também o valor pago por ele. Note que cada jogador j deve ter um valor privado v_j (só conhecido pelo jogador j) que é o limite que ele poderá pagar pelo item e ainda ter algum benefício ao adquiri-lo. Um dos problemas deste tipo de leilão, é que os jogadores tendem a mentir sobre o valor a ser declarado no envelope. Suponha que o ganhador do item colocou um valor de 10 no envelope e o segundo maior valor foi 5. Assim, apesar do ganhador ainda ter benefício ao declarar seu valor privado de 10, ele não fica contente, pois sabe que se tivesse declarado 6 também teria ganho e seu benefício seria ainda maior. Assim, este tipo de regra (mecanismo) pode levar os jogadores a mentir sobre os valores declarados, na esperança de aumentar seus benefícios.

Uma maneira de resolver isso, é fazer várias rodadas começando com algum valor mínimo, possivelmente 0. Em cada rodada o leiloeiro pergunta se algum outro participante cobre o valor da rodada anterior por um pequeno valor adicional. O primeiro a se manifestar é o ganhador daquela rodada com o valor anterior acrescido do pequeno adicional. Este processo se repete, sempre adicionando um pequeno valor ao definido na rodada anterior. Quando chegar a uma rodada onde não há outro participante disposto a pagar pelo novo valor, o leiloeiro para e declara o último vencedor como ganhador do leilão pagando seu correspondente valor.

Se neste mecanismo o item foi vendido por um valor σ para o jogador j^*, então certamente ainda compensa para o jogador j^* pagar pelo item, e portanto $v_{j^*} \geq \sigma$. E como não sobrou nenhum outro jogador disposto a pagar pelo item, o valor σ supera o valor privado dos outros jogadores, e portanto $v_j < \sigma$, para todo $j \neq j^*$. Além disso, como o leiloeiro foi aumentando o preço do item de uma pequena quantia, então o valor pago pelo jogador j^* é pouco acima do segundo maior valor privado. Se considerarmos que a quantia acrescida ao valor do item em cada rodada é bem pequena, o jogador j^* basicamente paga o valor do segundo maior valor privado. Este é conhecido como um leilão inglês, ou ascendente e neste caso o comprador (vencedor) fica satisfeito, pois sabe que por pouco menos que σ, haveria um outro comprador que também poderia comprar o item.

Um mecanismo que simplifica o processo anterior usando apenas uma rodada pode ser definido pelas seguintes regras [40]: *(i)* todos os jogadores declaram apenas um valor no envelope; *(ii)* o ganhador é definido como o jogador do maior lance e *(iii)* o valor a ser pago pelo ganhador é o valor do segundo maior lance. Chamaremos este mecanismo de Leilão de Vickrey.

Por ser simples, o leilão de Vickrey é empregado por muitos sites de leilões eletrônicos. Para leiloar um item, começa-se com um valor baixo e os possíveis compradores fazem seus lances, durante um certo intervalo de tempo. Cada possível comprador pode ofertar um valor bem maior

que o atual valor do item, uma vez que o acordo oferecido pela empresa que controla o leilão é de que não divulgará seu valor privado e se ele for o maior lance, apenas parte suficiente dele será usado para cobrir, por alguma pequena graduação monetária, o segundo maior valor. Desta maneira, os possíveis compradores enviarão seus valores privados, sabendo que pagará apenas o suficiente para cobrir o segundo maior lance.

Uma importante propriedade do Leilão de Vickrey é que os jogadores são incentivados a declararem nos envelopes seus valores privados. Isto é, um jogador j maximiza seu benefício se declarar seu valor v_j. Para cada jogador j, denote por b_j o valor (lance) declarado.

Proposição 18 *No leilão de Vickrey, ao declarar o valor b_j igual ao seu valor privado v_j, o jogador j maximizará seu benefício.*

Prova. Por simplicidade, vamos supor que neste jogo não ocorrem empates. Deixaremos como exercício a extensão quando há regras de desempates. Considere um jogador j e seja σ o maior valor dos outros jogadores (*i.e.*, $\sigma = \max_{i \neq j} b_i$). Vamos mostrar que o jogador j maximiza seu benefício se declarar o valor $b_j = v_j$, considerando os seguintes casos:

Caso 1: $v_j < \sigma$**.** Se j declarar o valor b_j igual a v_j, então ele não leva o item e seu benefício é 0. De fato, isto ocorre para qualquer valor declarado $b_j < \sigma$. Se declarar $b_j > \sigma$ ele ganhará o leilão, porém deverá pagar um valor $p_j \geq \sigma$ e seu benefício será $v_j - p_j$. Como $p_j \geq \sigma > v_j$, seu benefício será negativo e portanto foi pior ter mentido.

Caso 2: $v_j > \sigma$**.** Se j declarar o valor v_j, então ele ganhará o leilão e seu benefício será o valor positivo $v_j - \sigma$. Ele também terá o mesmo benefício se declarar $b_j > v_j$. Caso declare um valor $b_j < \sigma$, então ele perde o leilão e seu benefício é 0, em vez de um valor estritamente positivo.

Em ambos os casos, o jogador j maximiza seu benefício se declarar seu valor privado v_j. □

Note que no leilão de Vickrey, pudemos fazer os jogadores declararem seus valores privados por termos definido o pagamento do vencedor baseado nos valores declarados pelos outros jogadores. De fato esta idéia é fundamental no mecanismo VCG, que veremos a seguir.

5.2 Definições e o Mecanismo VCG

Para deixar alguns termos mais uniformes, usaremos A como o conjunto de alternativas possíveis para a escolha do mecanismo e V_i como o conjunto de estratégias do jogador i. A preferência do jogador i é modelada por uma função de valoração $v_i : A \to \mathbb{R}$, onde $v_i \in V_i$. Nesta seção, $V_i \subseteq \mathbb{R}^A$ é o conjunto possível das valorações (estratégias) que podem ser escolhidas pelo jogador

i. Baseado nas estratégias dos jogadores, um mecanismo deve escolher uma alternativa de um conjunto A (escolha social) e deve definir quanto cada jogador deve pagar por esta escolha. Estamos interessados em mecanismos para os quais cada jogador tem como estratégia dominante declarar a sua informação privada para cada resultado (mentir não o leva a ter mais vantagens).

Definição 6 *Um mecanismo (de revelação direta) é dado por uma função de escolha social* $f : V_1 \times \cdots \times V_n \to A$ *e funções de pagamento* $p_1, \ldots, p_n$ *onde* $p_i : V_1 \times \cdots \times V_n \to \mathbb{R}$ *é o valor que o jogador* i *paga.*

Um jogador i possui um valor privado $v_i(a)$ para cada alternativa $a \in A$ e sua utilidade é dada por $u_i(a) = v_i(a) - p_i(v)$, que é o quanto a alternativa vale menos o quanto o jogador pagou por ela. Nosso interesse é fazer mecanismos que incentivem o jogador a declarar o valor verdadeiro $v_i(a)$ e a declaração de um outro valor $v'_i(a)$ não o leva a ter um benefício maior. Mais formalmente, temos:

Definição 7 *Um mecanismo* $(f, p_1, \ldots, p_n)$ *é dito ser incentivo-compatível (do inglês incentive-compatible, também usado como truthful ou strategy-proof) se para todo jogador* i*, todo* $v_1 \in V_1$*, ..., todo* $v_n \in V_n$ *e todo* $v'_i \in V_i$ *temos*

$$v_i(a) - p_i(v_i, v_{-i}) \geq v_i(a') - p_i(v'_i, v_{-i}),$$

onde $a = f(v)$ *e* $a' = f(v'_i, v_{-i})$.

Definição 8 *Um mecanismo é dito ter uma função de utilidade (utilitarian function) se sua função de escolha social é a somatória das declarações dos jogadores (isto é,* $\sum_i v_i(a)$*).*

Uma das principais abordagens para projetar mecanismos é aplicar o método VCG [40, 6, 19], que permite definir mecanismos incentivo-compatíveis para funções de utilidade.

Definição 9 *Um mecanismo* $(f, p_1, \ldots, p_n)$ *é dito ser um mecanismo VCG (Vickrey-Clarke-Groves) se*

- $f(v_1, \ldots, v_n) \in \arg\max_{a \in A} \sum_j v_j(a)$*, isto é,* f *maximiza o benefício social e*
- *para funções* $h_1, \ldots, h_n$*, onde* $h_i : V_{-i} \to \mathbb{R}$ *para todo* $v_1 \in V_1$*,..., para todo* $v_n \in V_n$*, temos* $p_i(v_1, \ldots, v_n) = h_i(v_{-i}) - \sum_{j \neq i} v_j(f(v_1, \ldots, v_n))$.

No primeiro item acima temos que o mecanismo deve escolher uma alternativa que maximiza o benefício social dadas as escolhas dos jogadores; chamaremos estas alternativas de *alternativas economicamente eficientes*. O segundo ponto diz que a função de pagamento de um usuário i depende de uma função $h_i(v_{-i})$ e da soma $\sum_{j \neq i} v_i(f(v_1, \ldots, v_n))$ que é a soma total dos valores dos

outros jogadores. Note que o valor pago ao jogador i não depende do valor v_i. Na visão do jogador i, o termo h_i é basicamente uma constante, pois só depende dos valores declarados pelos outros jogadores.

Teorema 19 *Todo mecanismo VCG é incentivo-compatível.*

Prova. Seja $v = (v_1, \ldots, v_n)$ as declarações (funções) dos jogadores. Vamos mostrar que um jogador i não tem vantagens ao declarar v_i' em vez de v_i (fixando a declaração dos outros jogadores). Vamos denotar por $v' = (v_i', v_{-i})$ (declarações dos jogadores trocando v_i por v_i') e alternativas $a = f(v)$ e $a' = f(v')$. A utilidade de i, quando declarar v_i é

$$u_i(a) = v_i(a) - p(v) = v_i(a) - h_i(v_{-i}) + \sum_{j \neq i} v_j(a).$$

Ao declarar v_i', a utilidade de i é

$$u_i(a') = v_i(a') - p(v') = v_i(a') - h_i(v_{-i}) + \sum_{j \neq i} v_j(a').$$

Como no mecanismo VCG, v e a são tais que $\sum_j v_j(a)$ é máximo, temos que $\sum_j v_j(a) \geq \sum_j v_j(a')$. Assim,

$$\begin{aligned} u_i(a) &= v_i(a) - h_i(v_{-i}) + \sum_{j \neq i} v_j(a) \\ &= -h_i(v_{-i}) + \sum_{j} v_j(a) \\ &\geq -h_i(v_{-i}) + \sum_{j} v_j(a') \\ &= v_i(a') - h_i(v'_{-i}) + \sum_{j \neq i} v_j(a') \qquad (8) \\ &= u_i(a'), \end{aligned}$$

onde (8) vale pois v e v' tem as mesmas estratégias para os jogadores que não são i. Com isso, temos que a utilidade do jogador i não é melhor ao declarar v_i'. □

A função h usada na definição do mecanismo VCG pode inclusive fazer situações onde alguns jogadores recebem e outros pagam ao mecanismo, o que em muitos casos não é natural. Em geral, também não é normal os jogadores pagarem mais que o item vale para ele. Assim, é interessante que todos os benefícios/utilidades sejam não negativos e com isso nenhum jogador tenha prejuízo ao participar do jogo. As seguintes definições formalizam o que queremos:

Definição 10 *Um mecanismo é dito ser* individualmente racional *se os jogadores sempre obtêm benefício não negativo. Isto é, se para todo* $v_1, \ldots, v_n$ *temos* $v_i(f(v_1, \ldots, v_n)) - p_i(v_1, \ldots, v_n) \geq 0$. *Dizemos que um mecanismo* não tem transferências positivas *se nenhum jogador recebe do mecanismo (em vez de pagar). Isto é, se para todas funções* $v_1, \ldots, v_n$ *e todo jogador* i, *temos* $p_i(v_1, \ldots, v_n) \geq 0$.

É possível obter mecanismos VCG que atendam a estas duas condições definindo h_i como sendo $h_i(v_{-i}) = \max_{b \in A} \sum_{j \neq i} v_j(b)$ [6].

Note que no mecanismo VCG, a alternativa $a = f(v_1, \ldots, v_n)$ é tal que $\sum_i v_i(a)$ é máxima. Na regra de pagamento de Clarke para pagamento de um jogador i fazemos $p_i(v_1, \ldots, v_n) = \sum_{j \neq i} v_i(f(v_{-i})) - \sum_{j \neq i} v_i(a)$. A primeira parte busca por um resultado com benefício social máximo, mas sem contar com o jogador i. A segunda somatória nos dá o valor da alternativa que maximiza o benefício social, mas sem contar a contribuição do jogador i. Assim, o pagamento representado pela regra de Clarke ao jogador i é igual ao prejuízo que ele causaria aos outros jogadores, caso não participe da solução.

Lema 20 *Um mecanismo VCG com regra de pagamento de Clarke não faz transferências positivas. Além disso, se* $v_i(a) \geq 0$ *para todo* $v_i \in V_i$ *e* $a \in A$ *então o mecanismo é individualmente racional.*

Prova. Seja $a = f(v_1, \ldots, v_n)$ uma alternativa que maximiza $\sum_j v_j(a)$ e $b \in A$ uma alternativa que maximiza $\sum_{j \neq i} v_j(b)$. Vamos mostrar que o mecanismo é individualmente racional. A utilidade do jogador i é dada por

$$\begin{aligned} v_i(a) - p_i(v) &= v_i(a) + \sum_{j \neq i} v_j(a) - \sum_{j \neq i} v_j(b) \\ &\geq \sum_j v_j(a) - \sum_j v_j(b) \qquad (9) \\ &\geq 0, \qquad (10) \end{aligned}$$

onde a desigualdade (9) é válida pois $v_i(b) \geq 0$ e a desigualdade (10) é válida pois a foi escolhida como uma alternativa que maximiza $\sum_j v_j(a)$. Para mostrar que não há transferências positivas, note que

$$p_i(v_1, \ldots, v_n) = \sum_{j \neq i} v_j(b) - \sum_{j \neq i} v_j(a) \geq 0,$$

já que b foi escolhido como uma alternativa que maximiza $\sum_{j \neq i} v_j(b)$. ∎

A regra de pagamento de Clarke não se encaixa em muitas situações onde os valores são negativos, quando alternativas tem custos aos jogadores. No caso de custos, é o mecanismo que paga para os jogadores. Podemos adaptar a regra de pagamento de Clarke para um jogador i, modificando a regra que escolhe b como a alternativa que minimiza o custo social quando i não participa do jogo.

Exemplo 7 *Considere novamente o Leilão de Vickrey. Vamos de fato mostrar que o mecanismo definido no leilão de Vickrey é VCG. Cada resultado do jogo indica que um dos jogadores é o vencedor e os demais são os perdedores. O vencedor paga pelo item e os demais pagam* 0*. Denote o conjunto de jogadores por* $N = \{1, \ldots, n\}$ *e por* $A = \{1, \ldots, n\}$ *o conjunto de possíveis resultados do jogo. Assim, se o resultado do jogo é* $a \in A$*, então o jogador* a *é o vencedor do jogo.*

O valor da alternativa a' *para o jogador* i *é dado por* $v_i(a')$ *que é* 0 *se* $i \neq a'$ *ou o valor privado* $v_i(a)$ *se* $i = a'$*.*

Aplicando o mecanismo de Vickrey, o resultado do jogo é uma alternativa a*, que maximiza* $\sum_i v_i(a)$*. Para cada alternativa há apenas um valor não nulo, que é igual ao lance do jogador correspondente à alternativa. Assim, a alternativa que maximiza esta soma é a alternativa do jogador de maior lance.*

Definido o vencedor, digamos i^* *com alternativa* a*, seu pagamento é definido como*

$$p_{i^*} = \max_{b \in A} \sum_{j \neq i^*} v_j(b) - \sum_{j \neq i^*} v_j(a).$$

O primeiro termo do lado direito da igualdade é o maior valor de uma alternativa, quando o jogador i^* *não participa do jogo; e portanto é igual ao segundo maior lance. O segundo termo do lado direito, é a soma dos valores obtidos pelos outros jogadores quando* i^* *venceu, que dá* 0*, pois* $v_j(a) = 0$ *para todo* $j \neq i^*$*. Assim, o mecanismo VCG aplicado a este leilão de item único nos dá exatamente o Leilão de Vickrey e portanto é um mecanismo incentivo-compatível.*

5.3 Um Jogo de Caminho Mínimo

Muitos projetos de mecanismos podem ser resolvidos usando o mecanismo VCG. Um exemplo é o jogo do camínho mínimo [31], que definimos a seguir.

Suponha que um grafo $G = (V, E)$ representa os caminhos possíveis por onde podemos construir um caminho de s a t, onde $s, t \in V$. Cada aresta $e \in E$ representa um trecho que pode ser construído por um agente (jogador) distinto e sua construção demanda um valor $c_e \geq 0$ por seu agente. Por simplicidade, usaremos e para denotar tanto a aresta como seu agente. O objetivo é encontrar um caminho de s a t em G e determinar pagamentos p_e para cada agente e. O valor gasto pelo agente e é 0 se sua aresta não faz parte do caminho escolhido (pois não precisará construir nada) e c_e se for escolhido (neste jogo é o sistema que paga para o agente e). Note que este exemplo é de custo (em vez de utilidade). Assumiremos por simplicidade que o grafo é 2-aresta conexo (isto é, a remoção de uma aresta ainda deixa o grafo conexo).

Este problema tem aplicações na Internet, onde várias sub-redes pertencem a diferentes agentes e busca-se uma rota para fazer uma transmissão de dados pela rede. A idéia é construir um mecanismo que permita construir um caminho barato de s a t e defina pagamentos para os jogadores de maneira

que eles não tenham vantagens ao declarar valores diferentes dos custos verdadeiros das construções de suas arestas. Para isso, vamos usar o mecanismo VCG para definir o caminho e os pagamentos. A eficiência econômica é direta, pois o mecanismo VCG já nos garante que o resultado escolhido deve ser um que maximiza a utilidade social (minimiza custos). Com isso, seja P^* um caminho mínimo de s a t, em relação aos custos declarados. Assim, o custo social do resultado é dado por

$$c(P^*) = \sum_{e \in P^*} c_e. \tag{11}$$

Agora, continuando na definição do mecanismo VCG para este jogo, vamos definir os pagamentos p_e para cada agente e. Para isto, defina o conjunto $\mathcal{P}_{G-e}$ como o conjunto dos caminhos de s a t que não usam a aresta e. Assim, o pagamento do agente e é dado por

$$\begin{aligned} p_e &= \min_{P' \in \mathcal{P}_{G-e}} \sum_{f \in P'} c_f - \sum_{f \in P^* - e} c_f \\ &= \min_{P' \in \mathcal{P}_{G-e}} c(P') - c(P^*) + c_e. \end{aligned} \tag{12}$$

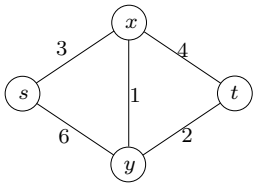
A definição do resultado e dos pagamentos pode ser feita através de no máximo n execuções de um algoritmo de caminho mínimo, sendo uma execução para a obtenção do caminho P^*, que será o resultado do jogo com valor dado em (11), e no máximo $n-1$ execuções para os pagamentos, uma para cada aresta $e \in P^*$ em (12). Para obter um caminho mínimo P' de s a t que não passa por e, basta executar um algoritmo de caminho mínimo no grafo $G - e$. Como o grafo é 2-aresta conexo, tal caminho sempre existe.

Assim, o mecanismo VCG para este jogo pode ser realizado com complexidade de tempo polinomial, uma vez que a obtenção de um caminho mínimo pode ser feita em tempo polinomial [11]. É interessante observar que em vez de fazer $n-1$ execuções do algoritmo de caminho mínimo para computar os pagamentos dos agentes no caminho ótimo, é possível obter todos os pagamentos computando apenas duas árvores de caminhos mínimos [21].

Exemplo 8 *Considere o grafo da Figura 10. O caminho mínimo de s a t é dado pela sequência de vértices $P^* = (s, x, y, t)$ e tem custo 6. Para calcular o pagamento da aresta sx, obtemos um caminho de s a t sem a aresta sx, o que nos dá o caminho $P_{-sx} = (s, y, t)$ de custo 8. Assim, o pagamento p_{sx} da aresta sx é dado por*

$$p_{sx} = c(P_{-sx}) - (c(P^*) - c_{sx}) = 8 - (6 - 3) = 5.$$

Note que se o custo da aresta sx fosse maior que 5, o custo do caminho P^* seria maior que 8 e com isso seria melhor tomar o caminho (s, y, t) que tem custo 8. Analogamente, o pagamento das



(a)

Figura 10: Construção de caminho mínimo.

arestas xy e yt são dados por

$$\begin{aligned} p_{xy} &= c(P_{-xy}) - (c(P^*) - c_{xy}) = 7 - (6 - 1) = 2, \\ p_{yt} &= c(P_{-yt}) - (c(P^*) - c_{yt}) = 7 - (6 - 2) = 3. \end{aligned}$$

Quando o problema de otimização subjacente para encontrar o resultado ótimo e os pagamentos é resolvido de maneira eficiente, podemos ter uma implementação do mecanismo VCG de maneira eficiente. Em particular, podemos ter mecanismos eficientes para o correspondente jogo que busca (constrói) uma árvore geradora de peso mínimo [8] e o jogo que busca um emparelhamento de peso máximo [27].

6 Leilões Combinatoriais

As aplicações em leilões tem motivado o estudo de vários problemas pelos pesquisadores da área de Teoria dos Jogos Algorítmica. Leilões pela Internet apresentam várias vantagens em relação aos leilões realizados em salas fechadas. Algumas destas vantagens podem incluir flexibilidade de tempo, sem limitações de presença física, permite grande número de possíveis compradores e vendedores, maior número de vendas simultaneamente, etc. As aplicações de leilões ocorrem em diversas áreas e produtos, desde venda de itens de pequeno valor como livros e discos de música, como os grandes leilões combinatoriais de espectros em telefonia celular na ordem de vários bilhões de dólares [4]. Uma das aplicações de grande atenção na área ocorre na apresentação de propagandas na Internet. Motores de busca, como `Google` e `Yahoo`, reservam alguns trechos (slots) da página de busca onde costumam apresentar links de propaganda [26]. Nesta aplicação, anunciantes oferecem valores para que seus links apareçam nos slots quando um certo conjunto de palavras for procurado. Com isso, as empresas dos motores de busca devem resolver um problema de leilão, muito rapidamente, para definir quais os vencedores dos slots para cada busca. A escolha dos vencedores deve considerar também a chance do link ser clicado, uma vez que os anunciantes só pagam pelos links clicados

(*pay-per-click*).

Nesta seção, veremos um problema de leilão onde vários itens estão sendo leiloados e cada possível comprador pode dar valores diferentes para várias combinações de itens. No exemplo visto do Leilão de Vickrey, temos apenas um item sendo leiloado. Uma maneira para tratar o leilão de vários itens é simplesmente fazer vários leilões de item único, como no caso do Leilão de Vickrey. Porém, isso não retrata situações importantes, como o interesse do jogador comprar um determinado conjunto de itens e não há interesse em comprar apenas um ou parte deles. Neste caso, dizemos que os itens deste conjunto se *complementam*. Outra situação é quando um certo conjunto de itens pode ser *substituído* por outro. Isto é, um comprador poderia estar interessado em pelo menos um dos conjuntos, mas se tiver os dois sua utilidade não será maior que ter apenas um deles.

Exemplo 9 *Considere um leilão onde dois itens $\{a, b\}$ estão sendo leiloados e há três potenciais compradores (jogadores 1, 2 e 3). A Tabela 2 dá os pesos atribuídos por cada jogador a cada possível conjunto.*

	$\{a\}$	$\{b\}$	$\{a, b\}$
Jogador 1	5	4	15
Jogador 2	6	6	6
Jogador 3	2	10	12

Tabela 2: Exemplo de instância para leilão combinatorial.

Nota-se que o Jogador 1, tem mais preferência pelo item a que pelo item b. Além disso, como deu um lance muito maior (que a soma dos lances individuais) para o conjunto $\{a, b\}$, nota se que a e b se complementam e resultam em um valor maior quando obtidos juntos para o jogador 1. Já para o jogador 2, os itens a e b possuem o mesmo valor, e além disso obter os dois itens não é mais interessante que obter apenas um. Assim, os itens a e b se substituem para o jogador 2. Já para o jogador 3, os itens são independentes e o valor dos dois itens juntos é exatamente a soma dos dois itens em separado.

6.1 Mecanismo VCG

Denotamos por I o conjunto dos itens sendo leiloados. Uma valoração v_i de um jogador i em um leilão combinatorial dos itens de I é uma função real para cada subconjunto $S \subseteq I$. Deve ser *monótona* (*i.e.*, $v_i(S) \leq v_i(T)$ para todo $S \subseteq T$) e *normalizada* (*i.e.*, $v_i(\emptyset) = 0$). As funções de valoração dos jogadores são dadas por um vetor de valoração $v = (v_1, \ldots, v_n)$ onde v_i é a função de valoração do jogador i. Uma *alocação* para um leilão combinatorial dos itens de I para n jogadores, é dado por uma sequência $\mathcal{S} = (S_1, \ldots, S_n)$, onde S_i é o conjunto de itens atribuído ao jogador i e

$S_i \cap S_j = \emptyset$ para $i \neq j$. Neste caso, o valor privado $v_i(\mathcal{S})$ é o valor $v_i(S_i)$, que é o valor do conjunto S_i para o jogador i. O benefício social desta alocação é dado pelo valor $v(\mathcal{S}) = \sum_i v_i(S_i)$, onde v é o vetor de valoração dos jogadores. Denotamos por A o conjunto de todas as alocações possíveis dos itens de I aos jogadores. Dizemos que uma alocação é *socialmente eficiente* se seu benefício social é máximo (aqui eficiência não tem a ver com complexidade de tempo polinomial).

Neste jogo, os jogadores definem os valores para os conjuntos de itens e o mecanismo deve definir quem são os vencedores e para cada vencedor dizer o seu conjunto de itens e quanto ele irá pagar pelo conjunto. Inicialmente, usaremos o mecanismo VCG de maneira a fazer os jogadores reportarem seus valores privados, obtendo uma alocação socialmente eficiente e definindo os pagamentos de cada jogador.

Mecanismo CA
Entrada: Conjunto de itens I a serem leiloados.
Subrotina: Algoritmo R para obter uma alocação socialmente eficiente.

1. Cada jogador i submete um lance $v_i(S)$ para cada $S \subseteq I$.
2. Use R para obter uma alocação $\mathcal{S} = (S_1, \ldots, S_n)$ para o vetor de valoração $v = (v_1, \ldots, v_n)$ que maximiza $v(\mathcal{S})$.
3. O pagamento do jogador i é definido como o valor p_i, dado por

$$p_i = \max\{v(\mathcal{S}') : \mathcal{S}' \in \mathbb{S}_{-i}\} - \sum_{j \neq i} v_j(S_j),$$

onde $\mathbb{S}_{-i}$ é o conjunto de todas as alocações que não atribuem conjuntos a i. Utilize a rotina R para resolver o problema de maximização.

Note que o mecanismo acima é uma aplicação do mecanismo VCG para o problema de leilões combinatoriais, uma vez que nos passos 2 e 3 buscamos por resultados ótimos sociais (que maximizam $v(\mathcal{S})$, com e sem o jogador i) e o pagamento é o quanto a ausência do jogador i causa de prejuízo para os outros jogadores, ao se ausentar da solução.

Como o pagamento definido acima usa a regra de Clarke, veja Lema 20, segue pelo Teorema 19 que o mecanismo acima é incentivo-compatível.

Teorema 21 *O mecanismo CA é incentivo-compatível.*

Para vários problemas, a resolução de partes do jogo podem envolver problemas computacionalmente difíceis de serem tratados e mesmo para problemas simples maximizar o benefício social pode ser um problema NP-difícil. No caso dos leilões combinatoriais, um primeiro cuidado que devemos ter é na definição da função de valoração, que como definido acima é dada para todo subconjunto

de itens. Assim, se o leilão tiver n itens, teremos 2^n subconjuntos possíveis e portanto precisaremos desta quantidade de valores para representar a valoração de cada jogador. Porém, na prática os jogadores se importam com poucas configurações de conjuntos e definem regras (algoritmos) para valorar outras situações baseadas nestas configurações básicas. Por exemplo, um jogador pode estar interessado em obter apenas o conjunto $\{a, b\}$ e tem um certo valor de utilidade para ele. Para todos os outros conjuntos que contêm estes dois itens, a utilidade é a mesma; já para os outros conjuntos que não contêm estes dois itens sua utilidade é nula. Neste caso, basta armazenar o conjunto de interesse deste jogador e a regra de valoração para os demais. Naturalmente, a eficiência do jogo também é dependente da forma como estes algoritmos avaliam os conjuntos.

Um segundo ponto que devemos tomar cuidado é na busca de uma alocação socialmente eficiente. Note que no mecanismo CA, o passo 2 é um problema de busca de uma alocação socialmente eficiente, e no passo 3 o pagamento do jogador i é calculado resolvendo se a busca de uma alocação socialmente eficiente sem o jogador i. Com isso, o mecanismo CA deve resolver $n+1$ problemas de se encontrar uma alocação socialmente eficiente. A dificuldade aqui é que o problema de se obter uma alocação socialmente eficiente é um problema NP-difícil. Veremos alguns casos particulares para os quais há algoritmos eficientes e um outro que apesar de ser NP-difícil, apresenta um mecanismo guloso que é incentivo-compatível e obtém uma alocação aproximada.

A vantagem de se usar o mecanismo VCG é que ele já nos garante que obteremos uma alocação que é incentivo-compatível. Assim, para instâncias pequenas do jogo, podemos utilizar um algoritmo exato para resolver o problema de alocação socialmente eficiente. Neste caso, há um algoritmo por programação dinâmica com complexidade de tempo $O(3^n)$ que poderia ser utilizado no mecanismo CA para obter alocação e pagamentos de maneira incentivo-compatível [34]. O seguinte teorema mostra que quando há poucos itens em relação ao número de compradores ou há poucos compradores em relação ao número de itens, o problema de alocação pode ser resolvido de maneira eficiente [34].

Teorema 22 *No jogo de leilão combinatorial com n itens e m jogadores, o mecanismo VCG pode ser implementado em tempo polinomial se $n \leq \log m$ ou se $m \leq \log n$.*

Um caso particular que também pode ser resolvido obtendo um mecanismo eficiente, é quando cada jogador i está interessado em apenas um conjunto S_i de cardinalidade máxima 2. Neste caso, o problema de alocação pode ser resolvido através de um problema de emparelhamento de peso máximo da seguinte forma: Seja I o conjunto de itens. Como um jogador i está interessado em apenas um conjunto S_i, denotaremos por (S_i, v_i) seu lance e denotaremos simplesmente por v_i o valor $v_i(S_i)$. Vamos construir um grafo inicialmente com cada vértice igual a um item. Para cada lance (S_i, v_i) com $|S_i| = 1$, coloque mais um vértice novo e ligue com o vértice em S_i através de uma aresta de peso v_i. Para cada lance (S_i, v_i) com $|S_i| = 2$, coloque uma aresta ligando os dois vértices em S_i através de uma aresta de peso v_i. Seja G o grafo ponderado obtido desta forma.

Note que com isso, uma alocação no jogo nos define um emparelhamento de mesmo peso em G (um emparelhamento é um conjunto de arestas que não tem extremos em comum). O problema de se obter um emparelhamento de peso total máximo pode ser resolvido eficientemente [27] e consequentemente, há uma rotina eficiente para obter uma alocação socialmente eficiente neste caso. Portanto, obtemos um mecanismo incentivo-compatível (usando esta rotina no mecanismo CA) para este caso.

6.2 Leilões Combinatoriais com Objetivo Único

Uma tentativa de se obter mecanismos eficientes no mecanismo VCG é substituir os algoritmos exatos de tempo exponencial por algoritmos eficientes (como algoritmos aproximados) na esperança que ainda continuem incentivo-compatíveis. Infelizmente não podemos simplesmente embutir uma rotina direta, pois nem sempre há garantias de que o mecanismo continue incentivo-compatível.

Para apresentar a utilização de um mecanismo aproximado neste caso, vamos considerar uma versão mais restrita do jogo, onde o problema de alocação associado ainda é intratável. Neste problema, que chamaremos de leilão combinatorial de objetivo único (*Single-Minded Case*), cada jogador tem interesse em obter um determinado conjunto de itens. O texto que segue foi baseado no artigo [4].

Uma valoração v_i, do jogador i, é chamada de valoração de *conjunto único* para o conjunto de itens I se existe um conjunto $S_i \subseteq I$ e valor $\phi_i > 0$ tal que $v_i(S_i) = \phi_i$ para todo conjunto S que contém S_i e $v_i(S_i) = 0$ caso contrário. Denotaremos um lance de conjunto único por um par (S_i, v_i) e o valor $v_i(S_i)$ simplesmente por v_i. Chamaremos apenas de *problema de alocação*, o problema de alocação para o jogo de leilão combinatorial com objetivo único.

Apesar de ser um caso particular dos leilões combinatoriais, o problema de se encontrar uma alocação socialmente eficiente para o caso de conjunto único ainda é NP-difícil.

Teorema 23 *Dado um inteiro k e uma instância do jogo de leilão combinatorial com objetivo único, o problema de se decidir se existe uma alocação socialmente eficiente de valor k é um problema* NP*-completo.*

Prova. A prova que este problema está em NP é direta. Assim, vamos mostrar que é NP-difícil. Para isso, vamos reduzir o problema do conjunto independente, enunciado como problema NP-completo no Teorema 2, para o problema de alocação de valor k.

Dados grafo $G = (V, E)$ e inteiro k, instância do problema do conjunto independente, defina o conjunto de itens como sendo o conjunto E e o conjunto de jogadores como o conjunto V. Para cada jogador $i \in V$, defina $S_i = \{e \in E : e \text{ é incidente a } i\}$, $v_i = 1$ e seu lance como o par (S_i, v_i). A prova segue do fato que um conjunto independente de tamanho k em G nos dá uma alocação de valor k no jogo, e vice-versa. □

Note que a prova de redução do problema do conjunto independente para o problema de alocação socialmente eficiente preserva aproximação, na versão de otimização destes dois problemas. Como há um resultado de inaproximabilidade para o problema do conjunto independente, este se transfere para o problema da alocação por valoração de conjunto único. O seguinte teorema mostra a dificuldade de se obter soluções aproximadas para o problema de conjunto independente [42, 20].

Teorema 24 *Não existe um algoritmo de aproximação com fator $n^{1-\epsilon}$, para qualquer $\epsilon > 0$, para o problema do conjunto independente, a menos que* P = NP.

Através deste teorema, obtemos o mesmo resultado de inaproximabilidade para o número de jogadores. Para adaptar o problema para número de itens, note que em um grafo, o número de arestas (que na redução são os itens) é no máximo n^2. Portanto, temos $m \leq n^2$ e o seguinte corolário segue do teorema anterior.

Corolário 25 *Não existe um algoritmo de aproximação com fator $m^{1/2-\epsilon}$, para qualquer $\epsilon > 0$, para o problema de alocação socialmente eficiente, a menos que* P = NP.

Agora, vamos considerar um mecanismo de aproximação incentivo-compatível para o jogo de leilão combinatorial com objetivo único. Este é um mecanismo guloso que privilegia conjuntos que tenham um equilíbrio entre quantidade de elementos e valor do conjunto. Posteriormente, mostraremos que este mecanismo é uma $\sqrt{m}$-aproximação para o problema de alocação, e de certo modo é o melhor possível, considerando o Corolário 25.

Mecanismo Guloso
Entrada: Conjunto de itens I a serem leiloados.

1. Cada jogador i submete um lance (S_i, v_i), onde $S_i \subseteq I$.
2. Reordene os lances tal que $\frac{v_1}{\sqrt{|S_1|}} \geq \frac{v_2}{\sqrt{|S_2|}} \geq \ldots \geq \frac{v_n}{\sqrt{|S_n|}}$.
3. $W \leftarrow \emptyset$
4. Para $i \leftarrow 1$ até n faça
5. se $S_i \cap (\cup_{j \in W} S_j) = \emptyset$ então $W \leftarrow W \cup \{i\}$.
6. Para $i \leftarrow 1$ até n faça
7. $p_i \leftarrow \frac{v_j}{\sqrt{|S_j|/|S_i|}}$, onde j é o menor índice tal que $S_i \cap S_j \neq \emptyset$ e para
8. todo $k < j, k \neq i$, $S_k \cap S_j = \emptyset$. Se não existir tal j então $p_i \leftarrow 0$.
9. Devolva a alocação $(T_1, \ldots, T_n)$, onde $T_i = S_i$ se $i \in W$ e $T_i = \emptyset$ caso

10. contrário, e pagamentos $(p_1, \ldots, p_n)$.

É fácil ver que este mecanismo é eficiente (de tempo polinomial). Para provar que este mecanismo é incentivo-compatível, vamos usar as seguintes propriedades para um mecanismo:

Definição 11 *Um mecanismo para o leilão combinatorial de objetivo único satisfaz a propriedade de* Monotonicidade *se todo jogador i que ganha com um lance (S_i, v_i) continua ganhando para qualquer $v_i' > v_i$ e para qualquer conjunto $S_i' \subset S_i$ (fixando os lances dos outros jogadores).*

Definição 12 *Um mecanismo para o leilão combinatorial de objetivo único satisfaz a propriedade de* Pagamento Crítico *se todo jogador i que vence, paga o menor valor necessário para vencer. Isto é, o ínfimo de todos os valores v_i' tal que (S_i, v_i') continua vencendo.*

Lema 26 *Um mecanismo para leilão combinatorial de objetivo único no qual os perdedores pagam 0 e que satisfaz as propriedades de monotonicidade e de pagamento crítico é incentivo-compatível.*

Prova. Note que nas condições dadas, um jogador que declara seu valor privado nunca irá receber uma utilidade negativa: sua utilidade é zero enquanto for perdedor (perdedores não pagam), e se vencer, seu valor deve ser pelo menos o valor crítico, que é exatamente seu pagamento.

Vamos mostrar que cada jogador i nunca melhora sua utilidade ao declarar um lance (S_i', v_i') em vez dos valores verdadeiros (S_i, v_i). Se (S_i', v_i') é um lance perdedor ou se S_i' não contém S_i, então claramente declarando (S_i, v_i) só pode ajudar (no sentido de não ser pior). Portanto, assumiremos que (S_i', v_i') é um lance vencedor e que $S_i' \supseteq S_i$.

Primeiro, vamos mostrar que o jogador nunca estará pior declarando (S_i, v_i') em vez de (S_i', v_i'). Denote o pagamento do jogador pelo lance (S_i', v_i') por p', e para o lance (S_i, v_i') por p. Para todo $x < p$, o lance (S_i, x) é um lance perdedor, uma vez que p é um valor crítico. Pela monotonicidade, (S', x) também será um lance perdedor para todo $x < p$, e portanto o valor crítico p' é pelo menos p. Segue que o lance (S_i, v_i'), em vez de (S_i', v_i'), também ganha e seu pagamento não aumenta.

Resta mostrar que o lance (S_i, v_i) não é pior que o lance vencedor (S_i, v_i'). Assuma primeiro que (S_i, v_i) é um lance vencedor com um pagamento (de valor crítico) $\tilde{p}$. Enquanto v_i' é maior que $\tilde{p}$, o jogador continuará ganhando com o mesmo pagamento; assim, declarando um valor diferente não terá maior benefício. Quando $v_i' < \tilde{p}$ o jogador irá perder, ganhando utilidade zero, e ele não se dará melhor neste caso também.

Se (S_i, v_i) é um lance perdedor, v_i deve ser um valor menor que o correspondente valor crítico. Assim, o pagamento para qualquer lance vencedor (S_i, v_i') será maior que v_i, fazendo este desvio não lucrativo. ∎

Note que o mecanismo guloso se encaixa nas condições do Lema 26, e portanto é incentivo-compatível. Para isso, vamos mostrar que satisfaz as condições de monotonicidade e de pagamento-crítico.

Lema 27 *O mecanismo guloso para o leilão combinatorial de objetivo único é incentivo-compatível.*

Prova. A monotonicidade é garantida, uma vez que aumentando v_i ou decrescendo S_i só pode mover o jogador i para o início da ordenação, e portanto melhora as chances de i ser ganhador. A condição de pagamento-crítico é satisfeita, uma vez que i ganha se aparecer na ordem antes de j (veja passo 7) que é o primeiro jogador a ser impedido pela entrada de i. O pagamento computado é exatamente o valor da transição entre i ocorrer antes e depois de j na ordenação do mecanismo guloso. □

A seguir, mostramos a razão de aproximação do mecanismo guloso.

Lema 28 *Seja O^* uma alocação com $\sum_{i\in O^*} v_i$ máximo e seja W a alocação gerada pelo mecanismo guloso. Então, $\sum_{i\in O^*} v_i \leq \sqrt{m}\sum_{i\in W} v_i$.*

Prova. Para cada $i \in W$, seja $O_i^* = \{j \in O^* : j \geq i \text{ e } |S_i \cap S_j| \neq \emptyset\}$ o conjunto de jogadores de O^* que não entraram em W por seus conjuntos interseptarem o conjunto de i, além de i, se $i \in O^*$. Claramente $O^* \subseteq \cup_{i\in W} O_i^*$ e portanto se provarmos a seguinte desigualdade, para cada $i \in W$,

$$\sum_{j\in O_i^*} v_j \leq \sqrt{m} v_i, \tag{13}$$

temos o resultado desejado:

$$\sum_{i\in O^*} v_i \leq \sum_{i\in W}\sum_{j\in O_i^*} v_j \leq \sum_{i\in W} \sqrt{m} v_i = \sqrt{m}\sum_{i\in W} v_i.$$

Para provar (13), note que todo $j \in O_i^*$ apareceu após i na ordem gulosa e portanto $v_j \leq \frac{v_i\sqrt{|S_j|}}{\sqrt{|S_i|}}$. Somando para todo $j \in O_i^*$, temos

$$\sum_{j\in O_i^*} v_j \leq \frac{v_i}{\sqrt{|S_i|}} \sum_{j\in O_i^*} \sqrt{|S_j|}. \tag{14}$$

Usando a desigualdade de Cauchy-Schwarz, podemos limitar a última somatória em (14) como

$$\sum_{j\in O_i^*} \sqrt{|S_j|} \leq \sqrt{|O_i^*|}\sqrt{\sum_{j\in O_i^*} |S_j|}.$$

Todo S_j para $j \in O_i^*$ intersepta S_i. Como O^* é uma alocação, estas interseções devem ser todas disjuntas, e portanto $|O_i^*| \leq |S_i|$. Como O^* é uma alocação, temos $\sum_{j\in O_i^*} |S_j| \leq m$. Portanto, obtemos $\sum_{j\in O_i^*} \sqrt{|S_j|} \leq \sqrt{|S_i|}\sqrt{m}$ e substituindo na desigualdade (14), temos a desigualdade (13). □

O próximo teorema segue direto dos lemas 27 e 28.

Teorema 29 *O mecanismo guloso é eficientemente computável, incentivo-compatível e produz um resultado que é uma $\sqrt{m}$-aproximação da alocação socialmente eficiente.*

7 Considerações Finais

Neste texto, apresentamos uma pequena introdução à Teoria dos Jogos Algorítmica, uma área com intensa atividade nos últimos anos. Apesar de dar uma pequena visão da área, apresentamos várias preocupações que se destacam quando os jogos são analisadas pelo lado algorítmico. O texto foi baseado principalmente no primeiro livro da área [32]. Sugerimos ao leitor ver este livro que além dos tópicos considerados no texto, também versa sobre a abordagem algorítmica de outros problemas em Teoria dos Jogos, como equilíbrio de mercados, criptografia, eleições e escolhas sociais, computação distribuída, compartilhamento de custos, mecanismos *online*, sistemas de reputação, leilões em motores de busca, entre outros.

Agradecimentos. Agradeço ao revisor do JAI e à Ana Bazzan pelas leituras e pelas sugestões que melhoraram a apresentação do texto. Agradeço também ao André Vignatti, que fez o mestrado e o doutorado nesta área e possibilitou termos valiosas discussões em vários problemas desta fascinante área. Por fim, agradeço ao CNPq pelo apoio financeiro.

Índice Remissivo

Referências bibliográficas

[1] E. Anshelevich, A. Dasgupta, J. Kleinberg, E. Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. *SIAM J. Comput.*, 38(4):1602–1623, 2008.

[2] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer, 1999.

[3] A. L. C. Bazzan. Coordenação de agentes com uso de técnicas de teoria dos jogos. In A. T. Martins and D. L. Borges, editors, *Jornada de Atualização em Inteligência Artificial*, pages 3–43. Sociedade Brasileira de Computação, 2001.

[4] L. Blumrosen and N. Nisan. Combinatorial auctions. In N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors, *Algorithmic Game Theory*, chapter 11, pages 267–299. Cambridge, 2007.

[5] M. H. Carvalho, M. R. Cerioli, R. Dahab, P. Feofiloff, C. G. Fernandes, C. E. Ferreira, K. S. Guimarães, F. K. Miyazawa, J. C. Pina Jr., J. Soares, and Y. Wakabayashi. *Uma Introdução Sucinta a Algoritmos de Aproximação*. Editora do IMPA, Rio de Janeiro, 2001. M. R. Cerioli and P. Feofiloff and C. G. Fernandes and F. K. Miyazawa (editores).

[6] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11(1), 1971.

[7] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing (STOC)*, pages 151–158, 1971.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2001.

[9] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a nash equilibrium. *SIAM J. Comput.*, 39(1):195–259, 2009.

[10] E. A. de Souza e Silva and D. R. Figueiredo. Uma breve introdução à teoria de jogos com aplicações a redes de computadores. In T. Kowaltowski and K. Breitman, editors, *Atualizações em Informática*, chapter 2, pages 57–114. Sociedade Brasileira de Computação, 2007.

[11] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[12] E. Even-Dar, A. Kesselman, and Y. Mansour. Convergence time to nash equilibrium in load balancing. *ACM Transactions on Algorithms*, 3(3):Article 32, 2007.

[13] A. Fabrikant, C. H. Papadimitriou, and K. Talwar. The complexity of pure nash equilibria. In *Proc. of the 36th annual ACM symposium on Theory of computing*, pages 604–612, New York, NY, USA, 2004. ACM.

[14] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[15] G. Finn and E. Horowitz. A linear time approximation algorithm for multiprocessor scheduling. *BIT*, 19:312–320, 1979.

[16] D. Fotakis, S. C. Kontogiannis, E. Koutsoupias, M. Mavronicolas, and P. G. Spirakis. The structure and complexity of nash equilibria for a selfish routing game. *Theoretical Computer Science*, 410:3305–3326, 2009.

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

[18] G. Gottlob, G. Greco, and F. Scarcello. Pure nash equilibria: Hard and easy games. *Journal of Artificial Intelligence Research*, 24:357–406, 2005.

[19] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[20] J. Hastad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182(1):105–142, 1999.

[21] J. Hershberger and S. Suri. Vickrey pricing in network routing: Fast payment computation. In *Proc. of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 252–259, 2001.

[22] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing lp. *J. ACM*, 50(6):795–824, 2003.

[23] D. S. Johnson, C. H. Papadimtriou, and M. Yannakakis. How easy is local search? *J. Comput. Syst. Sci.*, 37(1):79–100, 1988.

[24] E. Koutsoupias and C. H. Papadimitriou. Worst-case equilibria. In *STACS '99 : Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*, pages 404–413, 1999.

[25] T. Kowaltowski. Von Neumann: suas contribuições à computação. *Estudos Avançados*, 26(26):237–260, 1996.

[26] S. Lahaie, D. M. Pennock, A. Saberi, and R. V. Vohra. Sponsored search auctions. In N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors, *Algorithmic Game Theory*, chapter 28, pages 699–716. Cambridge, 2007.

[27] L. Lovász and M. D. Plummer. *Matching Theory*. North-Holland, 1986.

[28] F. K. Miyazawa. *XI Escola Regional de Informática*, chapter Programação Inteira, pages 49–90. SBC–Paraná, 2003.

[29] J. Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295, September 1951.

[30] N. Nisan. Introduction to mechanism design (for computer scientists). In N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors, *Algorithmic Game Theory*, chapter 9, pages 209–241. Cambridge, 2007.

[31] N. Nisan and A. Ronen. Algorithmic mechanism design. In *Proc. of the 31st Annual ACM Symposium on Theory of Computing*, 1999.

[32] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA, 2007.

[33] C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, Reading, Massachusetts, 1994.

[34] M. H. Rothkopf, R. M. Harstad, and A. Pekec. Computationally manageable combinational auctions. *Management Science*, 44(8):1131–1147, 1998.

[35] A. A. Schäffer and M. Yannakakis. Simple local search problems that are hard to solve. *SIAM J. Comput.*, 20(1):56–87, 1991.

[36] V. Syrgkanis. The complexity of equilibria in cost sharing games. In *Proc. of the 6th international conference on Internet and network economics*, WINE'10, pages 366–377, 2010.

[37] E. Tardos and V. V. Vazirani. Basic solution concepts and computational issues. In N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors, *Algorithmic Game Theory*, chapter 1, pages 3–28. Cambridge, 2007.

[38] E. Tardos and T. Wexler. Network formation games and the potential function method. In N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors, *Algorithmic Game Theory*, chapter 19, pages 487–516. Cambridge, 2007.

[39] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2001.

[40] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37, 1961.

[41] B. Vöcking. Selfish load balancing. In *Algorithmic Game Theory*, chapter 20, pages 517–542. Cambridge, 2007.

[42] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proc. 38th Annual ACM Symposium on Theory of computing*, pages 681–690, 2006.