Eric Inui RA138923 MC918/MO829 – Teoria dos Jogos Algorítmica

Resumo Seminário

Jogos Gráficos

Capítulo 7 do livro Algorithmic Game Theory,

Noam Nisan, Tim Roughgarden, Eva Tardos, e Vijay V. Vazirani,
editores. Cambridge University Press, 2007.

Abstract

Neste documento examinaremos os aspectos de um modelo de representação para jogos multijogadores. Conhecidos como jogos gráficos, estes modelos especificam restrições nas influências diretas de *payoffs* entre os jogadores de um jogo. Com o intuito de encontrar equilíbrios em jogos com vários jogadores, são apresentados algoritmos que funcionam em grafos na forma de uma árvore. Utilizando-os, conseguimos calcular todos os equilíbrios de Nash para um dado jogo, equilíbrios que podem ser aproximados, calculados em tempo polinomial em relação ao número de jogadores, ou exatos, calculados em tempo polinomial. Ainda, é estabelecida uma relação entre um jogo gráfico e redes de Markov para representar equilíbrios correlacionados desse jogo.

Introdução

Representar jogos de múltiplos jogadores com um número muito grande de tais na forma normal se torna um problema: primeiro pela sua representação crescer exponencialmente com o tamanho da população, segundo, a forma normal pode não representar estruturas presentes nas interações entre os jogadores que podem ajudar a encontrar equilíbrios. Para contornar essas dificuldades, jogos gráficos são utilizados. Eles são apropriados quando há muitos jogadores e a utilidade de cada um depende de uma pequena parcela da população.

Jogos gráficos adotam um modelo simples de grafos. Os jogadores são representados por vértices no grafo e seus *payoffs* dependem apenas dos vértices vizinhos a cada um. Cada jogador tem suas próprias especificações de utilidade, representadas por matrizes $M_{i,}$ para todo jogador $i \in$ ao grafo.

Dito isso, seguimos com algumas definições úteis.

Jogo com múltiplos jogadores:

- há n jogadores com conjunto finito de estratégias puras;
- $a_i \in \{0, 1\}$ é a ação escolhida pelo jogador *;*

- *c*ada jogador tem as suas especificações de *payoffs, que estão no intervalo* [0, 1] e são dados nas matrizes *M,* indexadas por $\vec{a} \in \{0, 1\}^{n;}$
- estratégias mistas: $p_i \in [0, 1]$ é a probabilidade do jogador *i* jogar 0;
- $\vec{p}$ é o vetor das probabilidades de todos os jogadores;
- *payoff* esperado para o jogador *i segue a equação:*

$$M_i(\vec{p}) = E_{\vec{a}\sim\vec{p}}[Mi(\vec{a})]$$

- $\vec{p}[i : p_i']$ *i*-ésimo componente de $\vec{p}$ muda para p_i';

Jogo gráfico:

- Cada jogador *i* é um nó no grafo não direcionado G;
- $N(i) \subseteq \{1, ..., n\}$, é a vizinhança do jogador *i* no grafo G (inclusive *i*);
- se $\vec{a}$ é uma ação conjunta, $\vec{a}i$ é a projeção de $\vec{a}$ apenas nos jogadores em *N*(*i*)

Portanto um jogo gráfico é definido como um par (G, *M*), onde G é um grafo não orientado e *M* um conjunto de *n* matrizes de *payoff*

<u>Equilíbrios:</u>

Temos 3 equilíbrios que serão computados pelos algoritmos:

- **Equilíbrio de Nash:** será o vetor de estratégias mistas $\vec{p}$, tal que vale:

$$M_i(\vec{p}) \geq M_i\ (\vec{p}[i : p_i']),\ \forall\ i,\ \forall\ pi \in [0,\ 1]$$

Significa que todas as estratégias escolhidas pelos jogadores é igual ou melhor que qualquer outra estratégia que eles podem jogar.

- ϵ**-Equilíbrio de Nash:** será o vetor de estratégias mistas $\vec{p}$, tal que vale:

$$M_i(\vec{p}) + \epsilon \geq M_i\ (\vec{p}[i : p_i']),\ \forall\ i,\ \forall\ pi \in [0,\ 1]$$

Análogo ao equilíbrio de Nash, porém agora com uma folga ϵ para o equilíbrio.

- **Equilíbrio correlacionado**

$$E_{\vec{a}\sim Pai=b}[Mi(\vec{a})] \geq E_{\vec{a}\sim Pai=b}\left[Mi\left(\vec{a}[i : \neg b]\right)\right]$$

$$\forall\ i \in \{1, ..., n\}\ ,\ \forall\ b \in \{0,\ 1\}$$

Pode ser entendido como se em um jogo houvesse um coordenador que sorteia uma das possíveis ações conjuntas para os jogadores, e informa individualmente o que cada um deve fazer. Todos fazem o que foi dito a eles, acreditando que os outros jogadores também o farão. Um exemplo é o sinal de trânsito, no qual o coordenador é o farol, que dá as instruções de ir (verde) parar (vermelho).

Algoritmos

Agora vamos especificar alguns algoritmos usados para encontrar equilíbrios de Nash e posteriormente um para os equilíbrios correlacionados.

TreeNash

Este algoritmo calcula todos os equilíbrios de Nash presentes em um dado jogo gráfico representado como uma árvore. Porém essa árvore, diferente do modo convencional, possui a raiz em baixo e as folhas mais acima. Mas da mesma fora de costume, o nó abaixo de um nó U é o filho de U, e os vértice acima de U ligados por uma aresta a ele são seus pais. Neste modelo, para cada vértice, há 2 conjuntos:

Upstream (U): conjunto de todos os vértices que estão em um caminho de U a todas as folhas da árvore. Escrito como $UP_G(U)$

Dowstream (U): conjunto de todos os vértices que estão no único caminho de U até a raiz da árvore.

Agora é possível se obter as seguintes definições:

- G^U é o subgrafo induzido em $UP_G(U)$, com raiz em *U;*
- Se V é filho de U e sua estratégia mista $v \in [0, 1]$, $M^U_{V=v}$ é o subconjunto de matrizes de *M* correspondente aos vértices em $UP_G(U)$;
- T(*v, u*) é uma tabela binária indexada pelos possíveis valores $v, u \in [0, 1]$ (sim, ela seria infinita!);
- T(*v, u*) é igual a
    - 1, see existe EN em ($G^{U'}$, $M^U_{V=v}$) e U=u é resposta ótima para V=v
    - 0, caso contrário
- Testemunhas (*witnesses): vetor* $\vec{u} = (u_{1, \dots,} u_k)$ de estratégias mistas para os pais $\vec{U} = (U_{1, \dots,} U_k)$ de um algum V, onde:
    - $T(v, u_i) = 1$, para todo $1 \leq i \leq k$

O algoritmo possui duas partes: a *downstream pass* e a *upstream.pass*

Na *downstream pass,* a partir das folhas (que estão no topo da árvore) são computados os T(u, v), sendo u a estratégia da folha U e v a do seu único filho V. O valor desse T será 1 se u for resposta ótima para v. Então U passa a tabela computada para o seu filho V, que recursivamente calculará valores de estratégia que façam com que ele esteja em equilíbrio tanto com $UP_G(V)$ e com o filho de V (W). Essas tabelas e os vetores de *witnesses* vão sendo computadas e passadas até chegar na raiz do grafo, que procura por suas estratégias que sejam ótimas para as entradas em T iguais a 1 (nas quais o equilíbrio existia até os pais da raiz), formando (ou mais) um equilíbrio(s) para o jogo (*G, M*).

Na parte do upstream pass é feito o contrário da downstream. A partir da raiz, que já possui as tabelas calculadas na downtream, é escolhida uma estratégia para a raiz que faça parte de um equilíbrio. Então para os pais da raiz são passadas as estratégias que eles devem jogar para manter o equilíbrio no grafo.

Note que se T(u, v) = 1 para algum par (u, v), existe(m) equilíbrio(s) para o jogo em

($G^{U'}$, $M^{U}_{V=v}$)

ApproximateTreeNash

Modificação do algoritmo TreeNash, recebendo uma entrada ϵ a mais ϵ que será a folga para o equilíbrio de Nash. E também, as estratégias dos jogadores são restringidas a apenas múltiplos de um fator τ.

Se τ = O(ϵ/d) o algoritmo encontra um ϵ-Equilíbrio de Nash e tem tempo de execução polinomial em 1/ ϵ, n e 2^d

ExactTreeNash

Simplesmente uma abstração do TreeNash. Se baseia no fato de que em qualquer jogo gráfico em forma de árvore, para qualquer das tabelas T(u, v) definidas pelo TreeNash, a região $\{(u, v) \in [0, 1]^2 : T(u, v) = 1\}$ pode ser representada por uma união finita de regiões retangulares em $[0, 1]^2$. Conforme o algoritmo avança, essas regiões vão se multiplicando, o que o torna exponencial em relação ao número de jogadores.

Equilíbrio correlacionado

Por ser complexo e poderem existir correlações de alta ordem, a sua representação pode se tornar muito difícil. Através de estudos concluiu-se que as redes de Markov podem ser relacionadas com os jogos gráficos e eles podem representar qualquer equilíbrio correlacionado existente no jogo.

Estabelecida essa relação, o calculo dos equilíbrios correlacionados são feito através de um PLI (programação linear inteira). Da seguinte forma:

Variables: For every player i and assignment $\vec{a}^{\,i}$, there is a variable $P(\vec{a}^{\,i})$.
LP Constraints:

(i) *CE Constraints:* For all players i and actions a, a',

$$\sum_{\vec{a}^{\,i}:a_i^i=a} P(\vec{a}^{\,i})M_i(\vec{a}^{\,i}) \geq \sum_{\vec{a}^{\,i}:a_i^i=a} P(\vec{a}^{\,i})M_i([\vec{a}^{\,i}[i:a']]).$$

(ii) *Neighborhood Marginal Constraints:* For all players i,

$$\forall \vec{a}^{\,i}: \quad P(\vec{a}^{\,i}) \geq 0; \quad \sum_{\vec{a}^{\,i}} P(\vec{a}^{\,i}) = 1.$$

(iii) *Intersection Consistency Constraints:* For all players i and j, and for any assignment $\vec{y}^{\,ij}$ to the *intersection neighborhood* $N(i) \cap N(j)$,

$$\begin{aligned} P(\vec{a}^{\,ij}) &\equiv \sum_{\vec{a}^{\,i}:\vec{a}^{\,ij}=\vec{y}^{\,ij}} P(\vec{a}^{\,i}) \\ &= \sum_{\vec{a}^{\,j}:\vec{a}^{\,ij}=\vec{y}^{\,ij}} P_j(\vec{a}^{\,j}) \\ &\equiv P_j(\vec{a}^{\,ij}). \end{aligned}$$

E função objetivo: F ({$P_i(\vec{a}^{i})$}) = $\sum_i \sum_{\vec{a}i} Pi(\vec{a}i)\, Mi(\vec{a}i)$