Unique Games

Lucas Daher

2018

1 Definição do problema

1.1 Definição informal

O problema dos jogos únicos (*unique games problem*) é um problema de satisfação de restrições (*constraints*) em que cada restrição é binária, ou seja, cada restrição pode ser entendida como uma função que recebe dois argumentos e retorna 0 ou 1. Além disso, para cada restrição, dado um dos argumentos, existe exatamente um valor para o outro argumento que satisfaz a restrição.

Um problema que faz parte da família do *unique games* é o *maxcut*: dado um grafo G, encontrar um conjunto de vértices tal que a quantidade de arestas entre o conjunto e seu complementar seja máximo. Podemos modelá-lo como cada vértice sendo uma variável e cada aresta, uma restrição, que só é satisfeita se os dois vértices adjacentes a ela pertencerem a partições diferentes. A solução é a partição que maximiza o número de restrições satisfeitas.

1.2 Definição formal

Uma instância do problema (U, V, C) pode ser definida como:

* U, universo finito de valores possíveis para as variáveis
* $V := \{x_1,\ x_2, \ldots,\ x_n\}$, conjunto de n variáveis em que $x_i \in U$, para $1 \leq i \leq n$
* $C := \{c_1,\ c_2, \ldots,\ c_n\}$, conjunto de m restrições (funções) $U^2 \to \{0,\ 1\}$

Para cada $c_i \in C$, $lhs,\ rhs \in U$, existem $lcomp \in U$ e $rcomp \in U$ tais que:

* $c_i(lhs,\ rcomp) = 1$

* $c_i(lhs,\ u) = 0$, se $u \neq rcomp$

* $c_i(lcomp,\ rhs) = 1$

* $c_i(u,\ rhs) = 0$, se $u \neq lcomp$

1.3 Problema de decisão

A versão de decisão dos problemas da família do *unique games* consiste em encontrar uma solução que satisfaz todas as restrições. Essa versão do problema pode ser resolvida em tempo polinomial, ou seja, é "fácil".

Vamos modelar uma instância do problema como um grafo: sejam G um grafo e $(U,\ V,\ C)$ uma instância do UGP tais que para cada $x_i \in V$ corresponde um $v_i \in V(G)$ e para cada restrição $c_i(x_j,\ x_k)$ corresponda $u_j u_k \in E(G)$.

Para resolver o problema, supomos sem perda de generalidade que G seja conexo. Se não for, basta dividí-lo em subgrafos conexos. Escolhemos um vértice qualquer de G, digamos v_i e para cada $u \in U$, fazemos $x_i = u$. Para cada vértice v_j adjacente a v_i, há apenas um valor em U para x_j que satisfaz $c(x_i,\ x_j)$. Assim, "propagamos" os valores a partir de u_i até completar o grafo ou chegar em uma contradição quanto ao valor de um vértice. Esse algoritmo pode ser executado em $\mathcal{O}((|V| + |C|) * |U|)$

Por exemplo, a versão de decisão do *maxcut* é encontrar um corte que cubra todas as arestas, ou seja, saber se um grafo é bipartido.

1.4 Problema de otimização

A versão de otimização consiste em encontrar uma solução que maximize o número de restrições satisfeitas e é NP-difícil. Veremos uma aproximação que, dado uma instância em que uma fração de pelo menos $1 - \epsilon$ das clausulas é satisfazível, retorna uma solução que satisfaz pelo menos $1 - O(\sqrt{\epsilon\ log\ n})$

2 Algoritmo de aproximação

2.1 Modelagem

Vamos utilizar a definição do problema como um grafo para essa sessão: a cada variável corresponde um vértice, a cada restrição, uma aresta. Definiremos também a permutação $\pi_{uv}(i) = j$, em que $i,\ j \in U$ e $uv \in E(G)$, tal que, se $r(u) = i$, uv é satisfeito apenas se $r(v) = j$.

Começaremos modelando uma instância como um programa quadrático inteiro. Para cada vértice $u \in V$ e cada rótulo $i \in U$, definimos uma variável $u_i \in \{0,\ 1\}$, em que:

$$u_i = \begin{cases} 1, & \text{se } u \text{ recebe o rótulo } i \\ 0, & \text{caso contrário} \end{cases}$$

Como cada vértice recebe apenas um rótulo, temos que $\Sigma_{i=1}^{k} u_i^2 = 1$ e $u_i u_j = 0$ para $i \neq j$.

Para cada aresta uv temos que:

$$\Sigma_{i=1}^{k} u_i v_{\pi_{uv}(i)} = \begin{cases} 1, & \text{se } uv \text{ é satisfeita} \\ 0, & \text{caso contrário} \end{cases}$$

Resultando o seguinte programa:

$$\begin{aligned} \max\ & \Sigma_{uv \in E(G)} \Sigma_{i=1}^{k} u_i v_{\pi_{uv}(i)} & \\ \text{sujeito a: } & \Sigma_{i=1}^{k} u_i^2 = 1, & u \in V(G), \\ & u_i u_j = 0, & u \in V(G), i, j \in U, i \neq j, \\ & u_i \in \{0,\ 1\}, & u \in V(G), i \in U \end{aligned}$$

Adicionaremos algumas restrições redundantes, mas úteis aos se relaxar o programa.

Para todo $u,\ v \in V(G)$ e $i,\ j \in U$, vale que $(v_j - u_i)^2 \geq v_j^2 - u_i^2$. E para todo $u,\ v,\ w \in V(G)$ e $i,\ j,\ h \in U$, vale que $(w_h - u_i)^2 \leq (w_h - v_j)^2 + (v_j - u_i)^2$. Prova: basta verificar todos os valores possíveis para as variáveis.

Adicionando essas restrições, obtemos o seguinte programa:

$$\begin{aligned} \max\ & \Sigma_{uv \in E(G)} \Sigma_{i=1}^{k} u_i v_{\pi_{uv}(i)} & \\ \text{sujeito a: } & \Sigma_{i=1}^{k} u_i^2 = 1, & u \in V(G), \\ & u_i u_j = 0, & u \in V(G), i, j \in U, i \neq j, \\ & (v_j - u_i)^2 \geq v_j^2 - u_i^2, & u,\ v \in V(G), i,\ j \in U, \\ & (w_h - u_i)^2 \leq (w_h - v_j)^2 + (v_j - u_i)^2, & u,\ v,\ w \in V(G), i,\ j,\ h \in U \\ & u_i \in \{0,\ 1\}, & u \in V(G), i \in U \end{aligned}$$

Relaxamos o programa para um programa vetorial substituindo cada variável escalar u_i por um vetor u_i e cada operação de multiplicação por produto interno.

Ainda, temos que $u_i \cdot u_i = ||u_i||^2$ e $(v_j - u_i) \cdot (v_j - u_i) = ||v_j - u_i||^2$. Finalmente, obtemos o seguinte programa:

$$
\begin{aligned}
\max\ & \Sigma_{uv \in E(G)} \Sigma_{i=1}^{k} u_i \cdot v_{\pi_{uv}(i)} & \\
\text{sujeito a: } & \Sigma_{i=1}^{k} ||u_i||^2 = 1, & u \in V(G), \\
& u_i \cdot u_j = 0, & u \in V(G), i, j \in U, i \neq j, \\
& ||v_j - u_i||^2 \geq ||v_j||^2 - ||u_i||^2, & u,\ v \in V(G), i,\ j \in U, \\
& ||w_h - u_i||^2 \leq ||w_h - v_j||^2 + ||v_j - u_i||^2, & u,\ v,\ w \in V(G), i,\ j,\ h \in U \\
& u_i \in \mathcal{R}^{kn}, & u \in V(G), i \in U
\end{aligned}
$$

É claro que esse programa é uma relaxação do anterior e, portanto, um *upper bound* do valor ótimo.

2.2 Intuição do algoritmo

O primeiro passo do algoritmo é resolver a relaxação vetorial. A partir desse resultado, iremos podar arestas de G, que serão ignoradas pelo resto da execução. O objetivo da poda é remover arestas que contribuem pouco para o resultado e quebrar o grafo em esferas de raio pequeno. Para o centro u de cada esfera, escolheremos aleatoriamente um rótulo, dando probabilidade $||u_i||^2$ para cada rótulo i (é válido, pois $\Sigma_{i=1}^{k} ||u_i||^2 = 1$). Para cada outro vértice na esfera, escolheremos o rótulo j que minimiza $||w_i - v_j||^2$. Para uma aresta xy com ambas as pontas em uma mesma esfera, teremos alta probabilidade de satisfazê-la.

2.3 Notação

Precisamos de notação adicional:

* $\delta := \sqrt{\epsilon \ln(n+1)}$
* seja Z^* a solução ótima da relaxação linear. Então: $Z^* \geq opt \geq (1 - \epsilon) * |E(G)|$
* seja uv uma aresta, definiremos seu comprimento $l(uv) = \frac{1}{2} \Sigma_{i=1}^{k} ||u_i - v_{\pi_{uv}(i)}||^2$

O comprimento de uma aresta uv representa quão longe os vetores de u estão

dos vetores de v que satisfazem uv. Temos que:

$$\begin{aligned} l(u,\ v) &= \frac{1}{2}\Sigma_{i=1}^{k}||u_i - v_{\pi_{uv}(i)}||^2 \\ &= \frac{1}{2}\Sigma_{i=1}^{k}(||u_i||^2 + ||v_{\pi_{uv}(i)}||^2 - 2u_i \cdot v_{\pi_{uv}(i)}) \\ &= \frac{1}{2}\left(2 - 2\Sigma_{i=1}^{k}u_i \cdot v_{\pi_{uv}(i)}\right) \\ &= 1 - \Sigma_{i=1}^{k}u_i \cdot v_{\pi_{uv}(i)} \end{aligned}$$

Ou seja, $l(uv)$ é um menos a contribuição de uv para Z^*. Seja L^* a soma do comprimento de todas as arestas, então:

$$L^* = \Sigma_{uv \in E(G)} l(uv) = |E(G)| - Z^* \leq |E(G)| - (1-\epsilon)|E(G)| = \epsilon|E(G)|$$

2.4 Algoritmo

Primeiramente, devemos descartar toda aresta uv tal que $l(uv) \geq \delta/4$. Como $L^* \leq \epsilon|E(G)|$, descartamos no máximo:

$$\frac{4\epsilon|E(G)|}{\delta} = \frac{4\epsilon|E(G)|}{\sqrt{\epsilon \ln(n+1)}} = \frac{4|E(G)|\sqrt{\epsilon \ln(n+1)}}{\ln(n+1)} = \frac{4|E(G)|\delta}{\ln(n+1)}$$

arestas dessa maneira.

Agora, devemos dividir o grafo em esferas de raio menor ou igual a $\delta/4$. Utilizaremos o lema 2.1, que será provado na sub sessão seguinte.

Lema 2.1 *Existe um algoritmo polinomial que remove no máximo* $8|E(G)|\delta$ *arestas do grafo* G *e divide os vértices em* t *esferas* $B_1, \ldots, B_t$*, tais que cada bola* B *é centrada em algum vértice* $c \in V(G)$ *e possui raio menor ou igual a* $\delta/4$*.*

Agora, para cada esfera $B \in G$, seja $c \in V(G)$ o centro de B. Atribuimos a cada rótulo $i \in U$ probabilidade $||c_i||^2$ e escolhemos aleatoriamente um rótulo para c. Para todos os outros vértices $u \in B$, atribuimos o rótulo j tal que $||c_i i u_j||^2$ seja mínimo. O lema 2.2 garante que para toda aresta $uv \in E(G)$ com $u,\ v \in B$, uv é satisfeito com probabilidade maior ou igual a $1 - 3\delta$.

Lema 2.2 *Para toda bola* B *com centro em* c *e toda aresta* uv *com* $u,\ v \in B$*, a probabilidade do algoritmo atribuir rótulos que satisfazem* uv *é de pelo menos* $1 - 3\delta$*.*

Ao remover as arestas longas, removemos no máximo $4|E(G)|\delta/\ln(n+1) \leq 4|E(G)|\delta$ arestas. Ao dividir o grafo em esferas, removemos no máximo $8|E(G)|\delta$ arestas e ao atribuir rótulos, deixamos de satisfazer no máximo $3|E(G)|\delta$ arestas. Assim, deixamos de satisfazer no máximo $(4+8+3)|E(G)|\delta = 15|E(G)|\delta$ arestas.

Concluimos que o algoritmo satisfaz uma fração de pelo menos $1 - 15\delta = 1 - \mathcal{O}(\sqrt{\epsilon \ln(n+1)})$ das restrições.

2.5 Provas adicionais

Provaremos lema 2.1 e lema 2.2 a seguir.

2.6 Resultados adicionais

Existem aproximações melhores, que também utilizam a mesma relaxação vetorial vista acima, como a enuncia no lema 2.3:

Lema 2.3 *Dado uma instância do UGP em que pelo menos* $1 - \epsilon$ *das restrições é satisfazível, existe um algoritmo polinomial que satisfaz pelo menos* $1 - \mathcal{O}(\sqrt{\epsilon log(k)})$.

Assumindo que a conjectura dos jogos únicos é verdade, lema 2.3 é assinoticamente o melhor resultado possível.