Algorithms for Destructive Shift Bribery

Wojciech Grabis

Andrzej Kaczmarczyk, Piotr Faliszewski

January 9, 2020

Definicje

- Election $E = (C, V)$, gdzie $C = \{c_1, c_2, ..., c_m\}$ to zbiór kandydatów i $V = \{v_1, v_2, ..., v_n\}$ to multizbiór wyborców.
- Wyborcy sa zwiazani z pewnym porzadkiem kandydatów, opisujacych preferencje w wyborach.
- Poprzez $pos_v(c)$ opisujemy pozycje danego kandydata na liście wyborcy.
- Dodatkowo dla dwóch różnych kandydatów c i c' przez $N_E(c, c')$ oznaczamy ilość wyborców, którzy preferuja c do c'.

Reguła wyborów

- Election rule R to funkcja, która dla danego $E = (C, V)$ zwraca zbiór $W \subseteq C$ zwyciezców danych wyborów,
- W trakcie pracy rozważany jest model *unikalnego zwyciezcy*, czyli kandydat jest zwyciezca, tylko jeśli jest jedynym elementem zbioru $R(E)$.

Reguła wyborów c.d

- W pracy rozważana dla każdej reguły przedstawiona jest zasada przyznawania punktów, kandydaci z najwieksza ilościa punktów sa uznawani za zwyciezców.

Scoring protocols

- Protokół punktujacy określony jest przez wektor $\alpha = (\alpha_1, ..., \alpha_m)$ nierosnacych, nieujemnych liczb całkowitych.
- α-score danego $c \in C$ opisywane jest poprzez $\sum_v \alpha_{pos_v(c)}$
- Wśród najważniejszych protokołów sa: k-Approval, gdzie wektor składa sie z k jedynek i pozostałych zer, oraz Borda rule gdzie wektorem jest $(m-1, m-2, ..., 0)$

Bucklin i simplified Bucklin

- W celu wprowadzenia tych reguł wprowadzamy dodatkowa definicje: zwycieskiej rundy Bucklina która jest najmniejszym l takie, że istnieje kandydat, które $l-$Approval score jest wiekszy lub równy $\frac{|V|}{2} + 1$
- Zwyciezca reguły Simplified Bucklin sa wszyscy kandydaci posiadajacy powyższa wartość $l-$Approval score.
- Zwyciezca reguły Bucklin sa wszyscy kandydaci z najwieksza $l-$Approval score, gdzie l jest zwycieska runda Bucklina.

Maximin

Wartościa Maximin score kandydata c jest $min_{d \in C \setminus \{c\}} N_E(c, d)$

Nasz problem

Destructive Shift Bribery
Mamy dana reguł wyborów R elekcje $E = (C, V)$, kandydata $d \in C$ (zazwyczaj unikalny zwyciezca wyborów), budżet B, oraz funkcje p ceny przesuniecia d dla każdego wyborcy.

Funkcja ceny dla danego wyborcy v, gdzie dodatkowo $j = pos_v(d)$ zdefiniowana jest nastepujaco:

1. $p(0) = 0$
2. Dla każdych $i, i', i < i' \leq m - j$
3. $p(i) = +\infty$

Jest to funkcja opisujaca koszt przesuniecia kandydata d o i pozycji do tyłu.

Przykład

Przykładowo mamy problem *Destructive Shift Bribery* dla zasady Bordy, gdzie funkcja kosztu przesuniecia o i pozycji to i:

$$v_1 : b \succ a \succ c \succ d$$

$$v_1 : d \succ b \succ a \succ c$$

$$v_1 : d \succ c \succ a \succ b$$

$$v_1 : d \succ a \succ b \succ c$$

Poczatkowo kandydat d wygrywa z 9 punktami, jeśli przesuniemy d o 2 pozycje w preferencji v_4, to a i d maja po 7 punktów, wiec jeśli $B = 2$ to mamy rozwiazanie problemu *DSB*.

K-Approval

Istnieje algorytm wielomianowy dla problemu Destructive Shift Bribery dla reguły K-Approval. Dla danego $E = (C, V)$ oraz $C = \{c_1, c_2, ..., c_m\}$ oraz $d = c_1$ mamy:

- Sprawdzamy dla każdego kandydata $c \in C \setminus \{d\}$, czy może on w danym budżecie pokonać naszego nielubianego.
- Dzielimy głosujacych na 3 grupy $V_{d,c}, V_d, V'$, w taki sposób że pierwszy zbiór zawiera wszystkich, którzy maja d na pierwszy k pozycjach oraz c na $k+1$ pozycji, drugi gdzie sa pozostali wyborcy majacy d na pierwszych k pozycjach, oraz V' pozostali.

K-Approval c.d

- Sprawdzamy dla każdego a i b takiego, że $a \leq |V_{d,c}$ oraz $b \leq |V_d|$.
- Liczba a oznacza ilu bedziemy przesuwać z $V_{d,c}$, natomiast b ilu bedziemy przesuwać z V_d, wybieramy dla każdego zbioru najtańszy koszt dla przesuniecia d na $k+1$ pozycje.
- Sprawdzajac te kombinacje dla każdego kandydata, jeśli któraś kombinacja zapewnia że d nie wygrywa i mieści sie w budżecie to akceptujemy, jak żadna to odrzucamy.

Reguła Bordy

Istnieje algorytm wielomianowy dla problemu Destructive Shift Bribery dla reguły Bordy. Podobnie jak poprzednio dla danego $E = (C, V)$ oraz $C = \{c_1, c_2, ..., c_m\}$ oraz $d = c_1$ mamy:

- Bedziemy sprawdzali dla każdego kandydata $c \in C \setminus \{d\}$, czy może on w danym budżecie pokonać naszego nielubianego.
- Dodatkowo do algorytmu wprowadzamy

$$A(j,k) = \begin{cases} 1 & \text{gdy } \mathrm{v_j} \text{ rankinguje c wśród k pozycji za d} \\ 0 & \text{wpp} \end{cases}$$

oraz oznaczamy $s = score_E(d) - score_E(c)$

Reguła Bordy c.d

- Bedziemy chcieli policzyć dla każdego $j \in \{1, ..., n\}$ funkcje $f(j, k)$, która oznacza najmniejszy koszt przesuniecia d do tyłu dla zbioru głosujacych $\{v_1, v_2, ..., v_j\}$, tak, że w zmienionych wyborach E' $s - (score_{E'}(d) - score_{E'}(c)) \geq k$.
- Jeśli w $f(n, s)$ otrzymamy koszt nie wiekszy niż budżet to akceptujemy.
- Wartość f bedziemy liczyli w nastepujacy sposób:

$$f(j, k) = \min_{k' < k} f(j - 1, k - (k' + A(j, k'))) + p_j(k')$$

Uogólnienie dla Scoring Protocols

Zauważmy, że algorytm wykorzystany dla reguły Bordy nie korzysta z żadnej własności samej metody. W przypadku jeśli

- Różnica punktów pomiedzy kandydatem d oraz c może być ograniczona przez wielomian liczby głosujacych oraz liczby kandydatów

To przy pomocy funkcji $f(j, k)$ możemy policzyć rezultat dla danego kandydata dla każdego *Scoring protocol*.

Uogólnienie dla Scoring Protocols c.d

Istnieje algorytm wielomianowy, dla instancji problemu *Destructive Shift Bribery* w przypadku gdy scoring protocol α dany na wejściu może być zakodowany unarnie.

Uogólnienie dla Scoring Protocols c.d

W przypadku gdy funkcje kosztu sa pewna ograniczona wielomianowo liczba od ilości kandydatów oraz ilości głosujacych, to możemy nasza funkcje f zamienić na $f(j,t)$, która mówi o maksymalnym zwiekszeniu relatywnego wyniku kandydata c do d.

Istnieje algorytm wielomianowy, dla instancji problemu *Destructive Shift Bribery* oraz scoring protocol α, gdy funkcje ceny sa zakodowane unarnie.

Uogólnienie dla Scoring Protocols c.d

Problem *Destructive Shift Bribery* jest NP-zupełny, gdy scoring protocol α oraz funkcje kosztu sa zakodowane binarnie, oraz scoring protocol jest cześcia wejścia.

Dowód z redukcji problemu Partition do problemu Destructive Shift Bribery.

Uogólnienie dla Scoring Protocols c.d

Rozważmy instancje problemu Partition $S = (s_1, s_2, \ldots s_n)$, dodatkowo $s = \sum_{i=1}^{n} s_i$. Załóżmy że $\bigvee i \in \{1, \ldots n-1\} s_i \geq s_{i+1}$. Dodatkowo zakładamy, że s jest parzyste oraz $s_1 < \frac{s}{2}$. Nasza instancja problemu *DSB*:

1. Kandydaci $d, p_1, \ldots p_n$ oraz sztuczni dodatkowi $\{c_i^j | i, j \in \{1, \ldots, n\}$
2. Wektor do pnktowania $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{n^2+n+1})$, gdzie $\alpha_1 = \frac{s}{2}$ oraz $\bigvee i \in \{1, \ldots n\} \alpha_{i+1} = s_i$, pozostałe $\alpha_i = 0$
3. Nasze wybory składaja sie z n głosujacych, gdzie każdy reprezentuje element z S. Dla v_i umieszczamy na pierwszej pozycji p_i, natomiast d na $(i+1)$ pozycji. Pozostałe pierwsze $(n+2)$ pozycje wypełniamy naszymi sztucznymi kandydatami $\{c_i^j : j \in \{1, \ldots n\}$ pozostałe pozycje wypełniamy pozostałymi kandydatami p_k.
4. Ustawiamy budżet B na $\frac{s}{2}$. Natomiast funkcja kosztu przesuniecia $p_i(0) = 0$, dla każdego $t \in \{1, \ldots, n-i+1\}$ mamy $p_i(t) = s_i$, a pozostałe $p_i(t) = B + 1$.

Uogólnienie dla Scoring Protocols c.d

Przykład dla $S = (5, 4, 2, 2, 1)$, nasza stworzona elekcja $C = \{d\} \bigcup \{p_1, \ldots, p_5\} \bigcup \{c_i^j | i, j \in \{1, \ldots, 5\}\}$, protokół punktujacy $\alpha = (7, 5, 4, 2, 2, 1, 0, \ldots, 0)$ oraz budżet $B = 7$, listy głosujacych:

$$v_1 : p_1 \succ d \succ c_1^1 \succ c_1^2 \succ c_1^3 \succ c_1^4 \succ c_1^5 \succ \ldots$$

$$v_2 : p_2 \succ c_2^1 \succ d \succ c_2^2 \succ c_2^3 \succ c_2^4 \succ c_2^5 \succ \ldots$$

$$v_3 : p_3 \succ c_3^1 \succ c_3^2 \succ d \succ c_3^3 \succ c_3^4 \succ c_3^5 \succ \ldots$$

$$v_4 : p_4 \succ c_4^1 \succ c_4^2 \succ c_4^3 \succ d \succ c_4^4 \succ c_4^5 \succ \ldots$$

$$v_5 : p_5 \succ c_5^1 \succ c_5^2 \succ c_5^3 \succ c_5^4 \succ d \succ c_5^5 \succ \ldots$$

Maximin rule

W przypadku zasady Maximin, rozwiazanie naszego problemu DSB udaje sie znaleźć w czasie wielomianowym.

Załóżmy $E = (C, V)$, oraz $d \in C$, budżet B oraz funkcje kosztu $\{p_1, \ldots, p_n\}$. Rozwiazaniem naszego problemu bedzie wektor $S = (s_1, \ldots, s_n)$, który opisuje jakie przesuniecia mamy zrobić dla każdego głosujacego.

Ważnymi dla rozwiazania kandydatami sa w i t, gdzie w jest zwycieżajacym kandydatem, natomiast t implementuje wynik d ($score_{E'} = N_{E'}(d, t)$.

Maximin rule c.d

Zauważmy że jedyne rozwiazania które nas interesuja to takie, gdzie d wystepuje poniżej w lub tuż poniżej t. Nie opłaca sie przesuwać kandydata na dalsze pozycje. Bedziemy to nazywać *tight solution*

W naszym algorytmie bedzie dla każdej pary w, t szukać wśród powyższych rozwiazań. Dodatkowo jako $pref(c_1, c_2)$ oznaczamy zbiór wyborców preferujacy c_1 nad c_2, natomiast $price(v, c)$ to cena za przesuniecie d poniżej c w kolejności wyborcy v.

Maximin rule c.d

Bedziemy korzystać z programowania dynamicznego, zauważmy że jedynym zbiorem wyborców który nas interesuje to zbiór $pref(d, w) \bigcup pref(d, t) = \{v_1", v_2", \ldots, v_l"\}$, czyli wyborcy którzy preferuja d nad w lub t.

Nasza funkcja $f_{w,t}$ bedzie taka funkcja, że $f_{w,t}(j, x, y)$ to najmniejszy koszt przesuniecia d do tyłu x razy poniżej w oraz y razy poniżej t, czyli $j, x, y \in \{0, 1, \ldots, l\}$.

Nasze rozwiazanie bedzie znajdować sie wśród wartości $f_{w,t}(l, x, y)$ dla par $x, y \in \{0, 1, \ldots, l\}$.

Maximin rule c.d

Wzór na liczenie $f_{w,t}$

Wartości poczatkowe: Dla każdego $j, x, y \in \{0, 1, \ldots, l\}$ mamy $f_{w,t}(j, 0, 0) = 0$. Jeśli $x > 0$ lub $y > 0$ to $f_{w,t}(0, x, y) = \infty$.
Dla danego $j \in \{1, \ldots, l\}$ oraz $x, y \in \{0, 1, \ldots, l\}$, mamy 3 przypadki do rozważenia:

1. Jeśli $v_j" \in pref(d, w) \setminus pref(d, t)$:

$$f_{w,t}(j, x, y) = min \begin{cases} f_{w,t}(j-1, x, y) \\ f_{w,t}(j-1, x-1, y) + price(v_j", w) \end{cases}$$

2. Jeśli $v_j" \in pref(d, t) \setminus pref(d, w)$:

$$f_{w,t}(j, x, y) = min \begin{cases} f_{w,t}(j-1, x, y) \\ f_{w,t}(j-1, x, y-1) + price(v_j", t) \end{cases}$$

Maximin rule c.d

4. Jeśli $v_j" \in pref(d,t) \bigcap pref(d,w)$. To mamy 2 sytuacje, albo $d \succ w \succ t$ i wartość:

$$f_{w,t}(j,x,y) = min \begin{cases} f_{w,t}(j-1,x,y) \\ f_{w,t}(j-1,x-1,y) + price(v_j", w) \\ f_{w,t}(j-1,x-1,y-1) + price(v_j", t) \end{cases}$$

Lub sytuacje $d \succ t \succ w$ i wartość:

$$f_{w,t}(j,x,y) = min \begin{cases} f_{w,t}(j-1,x,y) \\ f_{w,t}(j-1,x,y-1) + price(v_j", t) \\ f_{w,t}(j-1,x-1,y-1) + price(v_j", w) \end{cases}$$

Koniec