

Algorithms for Destructive Shift Bribery

Wojciech Grabis

Andrzej Kaczmarczyk, Piotr Faliszewski

January 9, 2020

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

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

- W pracy rozważana dla każdej reguły przedstawiona jest zasada przyznawania punktów, kandydaci z największą ilością punktów są uznawani za zwycięzców.

Scoring protocols

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

Bucklin i simplified Bucklin

- W celu wprowadzenia tych reguł wprowadzamy dodatkowa definicje: zwycięskiej rundy Bucklina która jest najmniejszym l takie, że istnieje kandydat, które l -Approval score jest większy 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 największa l -Approval score, gdzie l jest zwycięska runda Bucklina.

Maximin

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

Destructive Shift Bribery

Mamy dane regułę wyborów R elekcje $E = (C, V)$, kandydata $d \in C$ (zazwyczaj unikalny zwycięzca wyborów), budżet B , oraz funkcję p ceny przesunięcia d dla każdego wyborcy.

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

- ❶ $p(0) = 0$
- ❷ Dla każdych $i, i', i < i' \leq m - j$
- ❸ $p(i) = +\infty$

Jest to funkcja opisująca koszt przesunięcia kandydata d o i pozycji do tyłu.

Przykład

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

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

$$v_2 : d \succ b \succ a \succ c$$

$$v_3 : d \succ c \succ a \succ b$$

$$v_4 : d \succ a \succ b \succ c$$

Początkowo kandydat d wygrywa z 9 punktami, jeśli przesuniemy d o 2 pozycje w preferencji v_4 , to a i d mają po 7 punktów, więc jeśli $B = 2$ to mamy rozwiązanie problemu *DSB*.

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

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

- Sprawdzamy dla każdego a i b takiego, że $a \leq |V_{d,c}|$ oraz $b \leq |V_d|$.
- Liczba a oznacza ilu będziemy przesuwać z $V_{d,c}$, natomiast b ilu będziemy przesuwać z V_d , wybieramy dla każdego zbioru najtańszy koszt dla przesunięcia d na $k + 1$ pozycje.
- Sprawdzając te kombinacje dla każdego kandydata, jeśli któraś kombinacja zapewnia że d nie wygrywa i mieści się 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 $C = \{c_1, c_2, \dots, c_m\}$ oraz $d = c_1$ mamy:

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

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

oraz oznaczamy $s = \text{score}_E(d) - \text{score}_E(c)$

- Bedziemy chcieli policzyć dla każdego $j \in \{1, \dots, n\}$ funkcje $f(j, k)$, która oznacza najmniejszy koszt przesunięcia d do tyłu dla zbioru głosujących $\{v_1, v_2, \dots, v_j\}$, tak, że w zmienionych wyborach E' $s - (\text{score}_{E'}(d) - \text{score}_{E'}(c)) \geq k$.
- Jeśli w $f(n, s)$ otrzymamy koszt nie większy niż budżet to akceptujemy.
- Wartość f bedziemy liczyli w następujący 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 pomiędzy kandydatem d oraz c może być ograniczona przez wielomian liczby głosujących 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 głosujących, to możemy naszą funkcję f zamienić na $f(j, t)$, która mówi o maksymalnym zwiększeniu 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.

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

Dowód z redukcji problemu Partition do problemu Destructive Shift Bribery.

Uogólnienie dla Scoring Protocols c.d

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

- 1 Kandydaci d, p_1, \dots, p_n oraz sztuczni dodatkowi $\{c_i^j | i, j \in \{1, \dots, n\}\}$
- 2 Wektor do punktowania $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_{n^2+n+1})$, gdzie $\alpha_1 = \frac{s}{2}$ oraz $\forall i \in \{1, \dots, n\} \alpha_{i+1} = s_i$, pozostałe $\alpha_i = 0$
- 3 Nasze wybory składają się z n głosujących, 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 $\{c_i^j : j \in \{1, \dots, n\}\}$ pozostałe pozycje wypełniamy pozostałymi kandydatami p_k .
- 4 Ustawiamy budżet B na $\frac{s}{2}$. Natomiast funkcja kosztu przesunięcia $p_i(0) = 0$, dla każdego $t \in \{1, \dots, n-i+1\}$ 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
 $C = \{d\} \cup \{p_1, \dots, p_5\} \cup \{c_i^j | i, j \in \{1, \dots, 5\}\}$, protokół punktujący
 $\alpha = (7, 5, 4, 2, 2, 1, 0, \dots, 0)$ oraz budżet $B = 7$, listy głosujących:

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

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

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

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

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

Maximin rule

W przypadku zasady Maximin, rozwiązanie naszego problemu DSB udaje się znaleźć w czasie wielomianowym.

Założmy $E = (C, V)$, oraz $d \in C$, budżet B oraz funkcje kosztu $\{p_1, \dots, p_n\}$. Rozwiązaniem naszego problemu będzie wektor $S = (s_1, \dots, s_n)$, który opisuje jakie przesunięcia mamy zrobić dla każdego głosującego.

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

Zauważmy że jedyne rozwiązania które nas interesują to takie, gdzie d występuje poniżej w lub tuż poniżej t . Nie opłaca się przesuwać kandydata na dalsze pozycje. Bedziemy to nazywać *tight solution*

W naszym algorytmie będzie dla każdej pary w, t szukać wśród powyższych rozwiązań. Dodatkowo jako $pref(c_1, c_2)$ oznaczamy zbiór wyborców preferujący c_1 nad c_2 , natomiast $price(v, c)$ to cena za przesunięcie 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 $\text{pref}(d, w) \cup \text{pref}(d, t) = \{v_1'', v_2'', \dots, v_l''\}$, czyli wyborcy którzy preferują d nad w lub t .

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

Nasze rozwiązanie będzie znajdować się wśród wartości $f_{w,t}(l, x, y)$ dla par $x, y \in \{0, 1, \dots, l\}$.

Maximin rule c.d

Wzór na liczenie $f_{w,t}$

Wartości początkowe: Dla każdego $j, x, y \in \{0, 1, \dots, l\}$ mamy $f_{w,t}(j, 0, 0) = 0$. Jeśli $x > 0$ lub $y > 0$ to $f_{w,t}(0, x, y) = \infty$.

Dla danego $j \in \{1, \dots, l\}$ oraz $x, y \in \{0, 1, \dots, l\}$, mamy 3 przypadki do rozważenia:

❶ Jeśli $v_j'' \in \text{pref}(d, w) \setminus \text{pref}(d, t)$:

$$f_{w,t}(j, x, y) = \min \begin{cases} f_{w,t}(j-1, x, y) \\ f_{w,t}(j-1, x-1, y) + \text{price}(v_j'', w) \end{cases}$$

❷ Jeśli $v_j'' \in \text{pref}(d, t) \setminus \text{pref}(d, w)$:

$$f_{w,t}(j, x, y) = \min \begin{cases} f_{w,t}(j-1, x, y) \\ f_{w,t}(j-1, x, y-1) + \text{price}(v_j'', t) \end{cases}$$

- 4 Jeśli $v_j'' \in \text{pref}(d, t) \cap \text{pref}(d, w)$. To mamy 2 sytuacje, albo $d \succ w \succ t$ i wartość:

$$f_{w,t}(j, x, y) = \min \begin{cases} f_{w,t}(j-1, x, y) \\ f_{w,t}(j-1, x-1, y) + \text{price}(v_j'', w) \\ f_{w,t}(j-1, x-1, y-1) + \text{price}(v_j'', t) \end{cases}$$

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

$$f_{w,t}(j, x, y) = \min \begin{cases} f_{w,t}(j-1, x, y) \\ f_{w,t}(j-1, x, y-1) + \text{price}(v_j'', t) \\ f_{w,t}(j-1, x-1, y-1) + \text{price}(v_j'', w) \end{cases}$$

Koniec