Konrad Lisiecki 291649

Zadanie Zaliczeniowe

Obliczeniowa Teoria Wyboru Społecznego

Semestr zimowy 2014/15

Warszawa, 2015-02-12

Rozwiązanie

Celem zadania jest znalezienie algorytmu na znalezienia wartości :

$$contribution(\alpha_i, VBS(A)) \tag{1}$$

Zgodnie ze wzorem jest rowne:

$$contribution(\alpha_i, VBS(A)) = \phi_i((A, v)) \tag{2}$$

Przypomnijmy, że zgodnie z definicja z zadania wartość shapleya dla gracza $\alpha_i \in A$ wynosi

$$\phi_i((A, v)) = \frac{1}{n!} \sum \Delta_\pi^G(\alpha_i) \tag{3}$$

czyli gdy podstawimy do wzour zamiast $v\ performance$ otrzymamy:

$$\phi_i((A, v)) = \frac{1}{n!} \sum \Delta_\pi^G(\alpha_i) \tag{4}$$

Co mozna rozpisac rowniez w nastepujacy sposob :

$$\phi_i((A, v)) = \frac{1}{n!} \sum_{\pi \in \Pi^A} (performance(VBS(C_i^\pi \cup \{a_i\})) - performance(VBS((C_i^\pi))) \tag{5}$$

$$\phi_i((A, v)) = \frac{1}{n!} \frac{1}{|X|} \sum_{x \in X} \sum_{\pi \in \Pi^A} (min_{a_i \in C_i^\pi \cup \{a_i\}} time(a_i, x) - min_{a_i \in C_i^\pi} time(a_i, x)) \tag{6}$$

Co ten wzor oznacza? Oznacza on, ze aby otrzymac wartosc Shapleya dla algorytmu a_i mozemy rozpatrywac kazde zadanie z osobna. Czyli teraz glownym problemem jest to jak efektywnie obliczyc przyrost efektywnosci (zbioru algorytmu) z przyjsciem algorytmu a_i dla kazdej permutacji dla danego zadania, czyli jak efektywnie obliczyc ten czlon:

$$\sum_{\pi \in \Pi^A} (min_{a_i \in C_i^\pi \cup \{a_i\}} time(a_i, x) - min_{a_i \in C_i^\pi} time(a_i, x)) \tag{7}$$

W tym celu (dla danego) zadania uszeregujmy algorytmy wedlug ich wydajnosci (szybkosci dojscia do rozwiazania) np: dla zadania x_3 mamy nastepujaca kolejnosc:

$$x_3 : a_1, a_2, a_3, a_4, a_5, a_6, a_7$$

Zauwazmy, ze jezeli algorytm a_i jest najwolniejszy ze wszystkich to nie wniesie zadnego przyrosu efektywnosci do jakiegokolwiek podzbioru algorytmu.

Zauwazmy, tez ze jezeli algorytm a_i dochodzi do zbioru w ktorym juz jest szybszy od niego algorytmy to tez nie wniesie zadnej poprawy efektywnosci.

Jak zatem bedziemy obliczac wartosc Shapley dla algorytmu a_i dla konkretnego zadania. Przesledzmy to na krok po kroku. Załóżmy, że a_i to a_4 z przykładu wyżej.

Algorytm będzie wnosił przyrost efektywności tylko, gdy koalicja do której dochodzi nie zawiera szybszego algorytmu, a więc składnik wartości Shapleya będzie niezerowy dla:

1. wszystkich permutacjei w ktorych a_5 jest najszybszy, a tych jest (i to mnozymy razy przyrost efektywnosci pomiedzy a_4, a a_5):

- $(2-1)!\binom{2}{2} = 2$ dla zbiorow 3-elementowych
- $(1-1)!\binom{2}{1} = 2$ dla zbiorow 2-elementowych
- (1 dla zbiorow 1-elementowych

2. wszystkich permutacjei w ktorych a_6 jest najszybszy, a tych jest (i to mnozymy razy przyrost efektywnosci pomiedzy a_4, a a_6):

- $(1-1)!\binom{1}{1} = 1$ dla zbiorow 2-elementowych
- (1 dla zbiorow 1-elementowych

3. wszystkich permutacjei w ktorych a_7 jest najszybszy, a tych jest(i to mnozymy razy przyrost efektywnosci pomiedzy a_4, a a_6):

- 1 dla zbiorow 1-elementowych

Na podstawie tego mozna tego przykladu mozna zapisac szkic algorytmu:

Algorytm Poniżej przedstawiony został algorytm z rozwiązania:

```

def
shapley = 0
l_x = 0; // liczba X-ow, ktora pozostala
count_x = 0; // biezaca liczba nieuzytych X-ow
bal = 0; // roznica miedzy nawiasami otwierajacymi i zamykajacymi

for x in X: //sprawdzamy wszystkie znaki w slowie
    p = performancePosition();
    for algoritmh in
```