

Использования Shapley Value для многокритериального ранжирования ребер графа

Ильгам Магданович Латыпов

Эксперт: Ю. В. Дорн

14 декабря 2023 г.

Цели исследования

Задачи:

- ▶ Дана функция на графе. Например: максимальный поток из вершины А в вершину Б. Нужно научиться выделять подграф наименьшей стоимости, который минимизирует/максимизирует эту функцию.
- ▶ Дан набор функций на графе. Нужно научиться выделять подграф, который обеспечивает "оптимальность" по всему набору и при этом имеет небольшую стоимость.
"Оптимальность" определим потом.

Идея: Ранжировать ребра по важности с помощью Shapley Value и применить жадный алгоритм для отбор ребер.

Проблемы:

- ▶ Точный подсчет Shapley Value – NP задача.
- ▶ Жадные алгоритмы типа рюкзака выдают не оптимальный подграф.

про Shapley Value

Введем обозначения: \mathcal{N} – множество игроков. $v : 2^{\mathcal{N}} \rightarrow \mathbf{R}$ – функция выигрыша игры. Она показывает какой выигрыш будет, если в игре участвует некоторый набор игроков.

SV(shapley Value) – способ оптимального распределения выигрыша между игроками в коалиционной игре.

$$\Phi_i(v) = \sum_{S \subseteq \mathcal{N} \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!} (v(S \cup \{i\}) - v(S))$$

То есть каждый игрок получает матожидание изменения выигрыша при добавлении этого игрока в игру.

Пример: пусть игроки не взаимодействуют между собой, то есть $v(S) = \sum_{i \in S} v(\{i\})$. тогда $\Phi_i(v) = v(\{i\})$.

Как считать SV

Есть множество работ посвященных подсчету SV . Найденные нами работы можно отнести к двум типам:

- ▶ Использование вида $v(S)$. В таких работах исследуется функция v и показывается, например, что ненулевых слагаемых полиномиальное количество. И на этом строится полиномиальный алгоритм.
- ▶ Методы Монте-Карло. Функция представляется суммой большого числа слагаемых. Так что можно семплировать некоторые слагаемые и получить приближенное значение функции.
- В некоторых работах отмечается, что аппроксимация SV зачастую слишком сильно отличается от реальных значений.

Approximating the SV without Marginal Contributions

Использовали результаты работы **Approximating the SV without Marginal Contributions** для приближенного вычисления SV.

Авторы исследовали вычисление SV для n игроков, при условии, что вычисление v можно запускать не более чем фиксированное число T раз.

В предложенном методе дисперсия SV будет равна $O(\ln(n)/(T - n))$. Это значение лучше чем в других рассмотренных работах.

Агрегация результатов: рюкзак

SV может быть проинтерпретирован как важность ребер для данного v . Из этого соображения появляется желание использовать рюкзак как способ выбрать подграф.

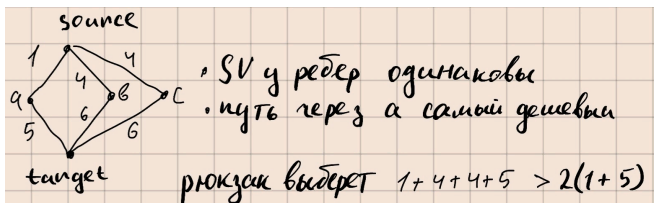
Однако этот метод может выдавать решения слишком далекие от оптимальных:

Обозначим $G(V, E)$ – граф

$$f_G(i, j) = \{\text{len of shortest path from } i \text{ to } j\}$$

и рассмотрим

$$v_{i,j}(S) = [f_G(i, j) == f_{G(V, S)}(i, j)]$$



Алгоритмы отбора ребер

Algorithm 1 greedy deletion

```
1: Input :  $\mathcal{N}, v$ 
2: deleted  $\leftarrow \emptyset$  ▷ list of deleted players
3:  $S \leftarrow \mathcal{N}$  ▷ non deleted players
4: while  $v(S) \neq 0$  and  $S \neq \emptyset$  do
5:    $i \leftarrow \arg \max_i v(S \setminus \{i\})$ 
6:   deleted  $\leftarrow$  deleted  $\cup \{i\}$ 
7:    $S \leftarrow S \setminus \{i\}$ ;
8: end while
9: return deleted
```

Algorithm 2 SV deletion

```
1: Input :  $\mathcal{N}, v$ 
2:  $S \leftarrow \mathcal{N}$  ▷ игроки которые остались
3: deleted  $\leftarrow \emptyset$  ▷ list of deleted players
4: arr  $\leftarrow \arg \text{sort}(SV(v))$  ▷  $SV(v)$  – массив значений SV. Сортируем игроков по увеличению SV
5: for  $i \in \text{arr}$  do
6:   if  $v(S \setminus \{i\}) > 0$  then
7:      $S \leftarrow S \setminus \{i\}$ ;
8:     deleted  $\leftarrow$  deleted  $\cup \{i\}$ ;
9:   end if
10: end for
11: return deleted
```

Рис.: proposed algorithms

Value Function

Нужно построить value function v . Предлагается ее строить в виде произведения функций от базовых метрик.

Рассмотрим задачу multicommodity flow. Хотим получить граф подешевле и чтобы стоимость потоков была не большой.

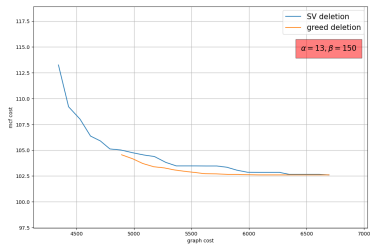
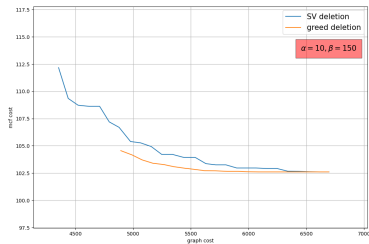
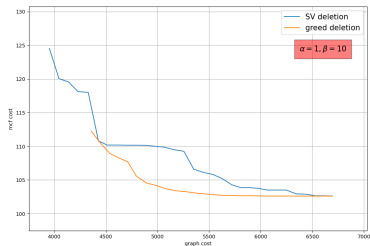
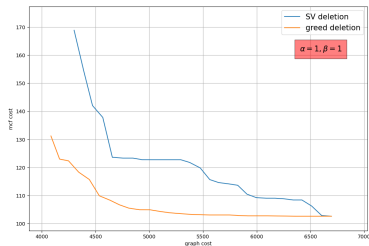
1. $\text{graph_cost} \in [0, \text{max_cost}]$. $g(c) = g(c) = (2 - \frac{c}{\text{max_cost}})$

2. $\text{flow_cost} \in [\text{min_flow_cost}, \infty]$. $h(f) = \frac{\text{min_flow_cost}}{f}$.

$$v(S) = \left(2 - \frac{c}{\text{max_cost}}\right)^\alpha * \left(\frac{\text{min_flow_cost}}{f}\right)^\beta \quad (1)$$

Рассмотрим результаты работы алгоритма при разных параметрах.

Эксперименты



Эксперименты

Теперь рассмотрим три метрики: cost, flow_cost, λ_2

Введем функции множители, которые представляют эти метрики в v :

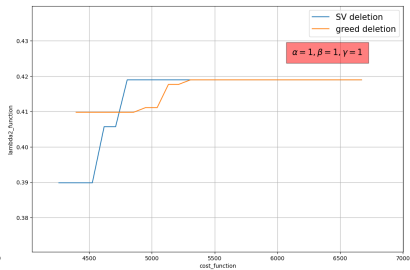
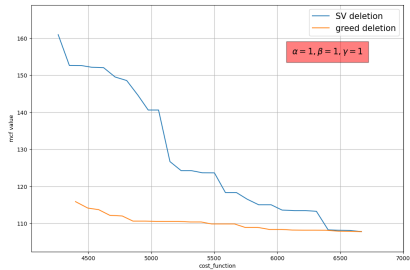
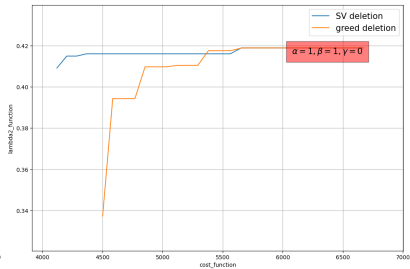
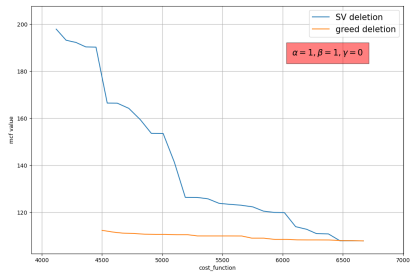
1. $\text{graph_cost} \in [0, \text{max_cost}]$. $g(c) = g(c) = 1 + (1 - \frac{c}{\text{max_cost}})$
2. $\text{flow_cost} \in [\text{min_flow_cost}, \infty]$. $h(f) = \frac{\text{min_flow_cost}}{f}$.
3. $\lambda_2 \in [0, 1]$. $r(\lambda_2) = \frac{1+\lambda_2}{2}$

И получим функцию качества:

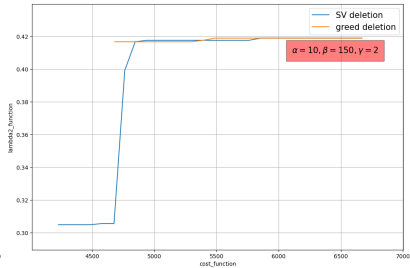
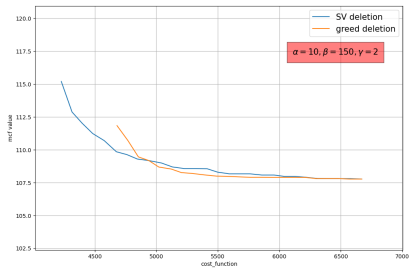
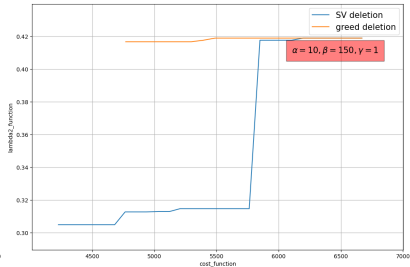
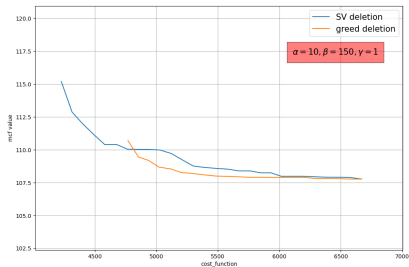
$$v_2(S) = \left(\frac{1}{2} \left(2 - \frac{c}{\text{max_cost}} \right) \right)^\alpha * \left(\frac{\text{min_flow_cost}}{f} \right)^\beta * \left(\frac{1 + \lambda_2}{2} \right)^\gamma \quad (2)$$

И снова рассмотрим работу алгоритмов при разных параметрах

Эксперименты



Эксперименты



При возведении в степень функция растягивается по оси u , из-за чего дифференцирующие свойства составляющих value function возрастают. То есть если взять две топологии, такие что $v_1 > v_2$, то при возведении компонент в степень отношение $\frac{v_1}{v_2}$ увеличится, а значит первая топология становится еще более предпочтительной.

1. results

- ▶ Предложен способ построения value function, который работает в примерах.
- ▶ Результаты оформлены в виде статьи. Код написан с примером применения и закинут на gitlab.

2. next

- ▶ Сделать автоматический подбор коэффициентов.
- ▶ Обосновать почему работает метод.
- ▶ Научиться использовать аддитивность SV по value function.

3. fails

- ▶ были рассмотрены способы аппроксимации SV с помощью нейронок, но там не получили каких-то результатов.
- ▶ Был рассмотрен способ агрегирования результатов построением путей. Но путей может быть слишком много и также другие проблемы.