Методы оптимизации для задач на графах

Першин Антон Юрьевич, Ph.D.

Программа «Большие данные и распределенная цифровая платформа»
Санкт-Петербургский государственный университет

20 апреля 2024 г.

Definition

Линейной программой называется задача оптимизация с линейной целевой функцией и линейными ограничениями (равенствами и неравенствами):

$$\begin{aligned} &\text{argmax}_{\boldsymbol{x}} \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.}\ &\boldsymbol{A}_{eq}\boldsymbol{x} = \boldsymbol{b}_{eq} \\ &\boldsymbol{A}_{ub}\boldsymbol{x} \leq \boldsymbol{b}_{ub} \\ &\boldsymbol{A}_{lb}\boldsymbol{x} \geq \boldsymbol{b}_{lb} \end{aligned}$$

Стандартная форма

$$\begin{aligned} &\text{argmax}_{\boldsymbol{x}} \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.}\ &\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \\ &\boldsymbol{x} \geq 0 \end{aligned}$$

Каноническая форма

$$\begin{aligned} &\text{argmax}_{\boldsymbol{x}} \boldsymbol{c}^T \boldsymbol{x} \\ \text{s.t.}\ &\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ &\boldsymbol{x} \geq 0 \end{aligned}$$

Кратчайший путь как линейная программа

Введем обозначения:

- $w(u,v)$ – вес ненаправленного или направленного ребра $u \to v$;
- s – начальный узел пути;
- t – конечный узел пути;
- $x(u,v)$ – бинарная переменная, указывающая, включено ли ребро $u \to v$ в путь.

Линейная программа поиска кратчайшего пути на графе:

$$\operatorname{argmin}_{\boldsymbol{x}} \sum_{(u,v)\in E} w(u,v)x(u,v)$$

$$\text{s.t.} \sum_{u\in V} x(u,t) - \sum_{w\in V} x(t,w) = 1$$

$$\sum_{u\in V} x(u,v) - \sum_{w\in V} x(v,w) = 0 \quad \forall v \in V\backslash\{s,t\}$$

Поиск восхождением к вершине (Hill Climbing)

Hill Climbing – стохастический итерационный алгоритм локального поиска.

```
n ← number of tweak desired to sample the gradient
S ← some initial candidate solution
repeat
    R ← Tweak(Copy(S))
    for n − 1 times do
        W ← Tweak(Copy(S))
        if Quality(W) > Quality(R) then
            R ← W
        end if
    end for
    if Quality(R) > Quality(S) then
        S ← R
    end if
until S is the ideal solution or we have run out of time
return S
```

Имитация отжига (Simulated Annealing)

Имитация отжига – стохастический итерационный алгоритм глобального поиска.

$t \leftarrow$ temperature
$S \leftarrow$ some initial candidate solution
$\xi \leftarrow$ random value generator yielding a random value from 0 to 1 each time it is accessed
$Best \leftarrow S$
repeat
    $R \leftarrow \text{Tweak}(\text{Copy}(S))$
    if $\text{Quality}(R) > \text{Quality}(S)$ or $\xi < \exp[\frac{\text{Quality}(R) - \text{Quality}(S)}{t}]$ **then**
        $S \leftarrow R$
    end if
    Decrease t
    if $\text{Quality}(S) > \text{Quality}(Best)$ **then**
        $Best \leftarrow S$
    end if
until $Best$ is the ideal solution or we have run out of time or $t \leq 0$
return $Best$

Раскраска графов

Раскраска графа предполагает присваивание одного из k цветов каждой вершине таким образом, чтобы у любых двух смежных вершин были несовпадающие цвета.

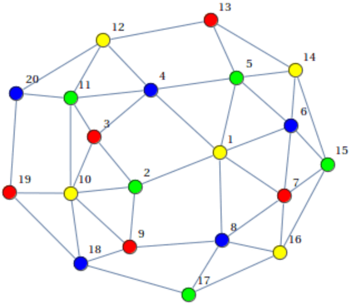

Рис.: 4-раскраска графа.

Зачастую количество цветов меньше хроматического числа, что требует поиска решения, минимизирующего количество конфликтов, то есть количество ребер с узлами одинаковых цветов. Кроме того, раскраска графов в общем случае является NP-сложной задачей.

Разбиение графов

Разбиение графа $G = (V, E)$ в k компонент предполагает такое разбиение $V = \bigcap_i V_i$, что сумма весов ребер, разделяющих компоненты, минимальна.

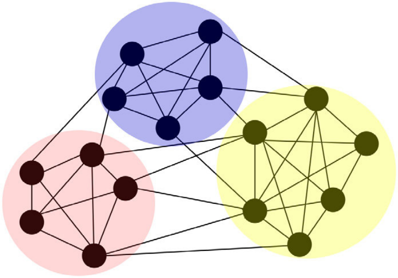

Рис.: Разбиение графа по 3 компонентам.

Эта задача находится множество применений: проектирование топологий сетей, балансировка наргазки в вычислительных системах, задачи теории расписаний и так далее. Разбиение графов в общем случае является NP-сложной задачей.

Спектральная кластеризация

Одним из способов решения задачи разбиения графа является спектральная кластеризация:

1. Построить матрицу смежности $\boldsymbol{A}$
2. Построить диагональную матрицу $\boldsymbol{D}$, где $D_{ii} = \sum_j A_{ij}$
3. Найти лапласиан графа: $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$
4. Найти разложение $\boldsymbol{L}$ на собственные числа и собственные вектора
5. Первый собственный вектор (предполагается сортировка от наименьшего собственного числа к наибольшему) содержит информацию о компонентах связности
6. Остальные вектора по порядку можно использовать как признаки для кластеризации (например, с помощью k-means)