Гибридный эволюционный алгоритм для задачи 2-CORRELATION CLUSTERING

Моршинин Александр
Институт Математики им. С.Л. Соболева СО РАН
Исследование выполнено за счет гранта Российского научного фонда
№ 22-71-10015-П

Задачи кластеризации

Задачи кластеризации являются важной частью анализа данных. В них необходимо разбить заданное множество объектов на несколько подмножеств (кластеров) основываясь на схожести объектов между собой.

В задаче CORRELATION CLUSTERING (КЛАСТЕРИЗАЦИЯ ВЕРШИН ГРАФА) требуется разбить вершины графа на кластеры, опираясь на реберную структуру этого графа.

Основные определения

Обыкновенный граф называется кластерным графом, если каждая компонента связности этого графа является полным графом. Компоненты связности называются кластерами.

Обозначим через $C(X,Y)$ кластерный граф, содержащий не более 2 кластеров (возможно $Y=\varnothing$).

Расстоянием $\rho(G_1,G_2)$ между помеченными графами $G_1=(V,E_1)$ и $G_2=(V,E_2)$ называется величина

$$\rho(G_1,G_2)=|E_1\Delta E_2|=|E_1\setminus E_2|+|E_2\setminus E_1|.$$

CORRELATION CLUSTERING

$CC_{\leq 2}$. Для произвольного графа $G = (V, E)$ найти ближайший к G кластерный граф C^* с не более чем 2 кластерами.

Эта задача *NP*-трудна.

3-приближенный алгоритм

Бансал, Блюм и Чаула (2004) предложили следующий 3-приближенный алгоритм для задачи $\mathbf{CC}_{\leq 2}$.

Алгоритм **N** (Neighborhood)
Вход: граф $G = (V, E)$. **Выход:** кластерный граф C_N. **Шаг 0.** Положить $F = \emptyset$. **Шаг 1.** Для каждой вершины $v \in V$ определить кластерный граф $C_v = C(X, Y)$, где $X = \{v\} \cup N_G(v)$, $Y = V \setminus X$. Положить $F = F \cup \{C_v\}$. **Шаг 2.** Среди всех графов из множества F выбрать ближайший к G кластерный граф C_N.

Трудоемкость алгоритма – $O(|V|^2)$.

Локальный поиск

Колман, Саундерсон и Вирт (2008) предложили следующую процедуру локального поиска.

Процедура **LS** (Local Search)
Итерация *k*. 1. Для каждой вершины вычислить величину уменьшения целевой функции при её переносе в противоположный кластер. 2. Если для всех вершин вычисленная величина неположительна ($\Delta \leq 0$), **завершить** алгоритм и вернуть текущее решение. 3. Иначе перенести вершину с максимальным положительным уменьшением в противоположный кластер и **перейти к итерации *k* + 1.**

2-приближенный алгоритм

Они также предложили следующий 2-приближенный алгоритм.

Алгоритм **N+mLS** (Neighborhood + multiple Local Search)
Вход: граф $G = (V, E)$. **Выход:** кластерный граф C_{NmLS}. **Шаг 1.** Пусть F – множество всех кластерных графов, построенных алгоритмом **N** на шаге 1. Применить процедуру **LS** к каждому графу из F. **Шаг 2.** Среди всех локальных оптимумов выбрать ближайший к G кластерный граф C_{NmLS}.

Трудоемкость процедуры **LS** – $O(|V|^3)$, а алгоритма **N+mLS** – $O(|V|^4)$.

Еще один 3-приближенный алгоритм

Легко получить еще один 3-приближенный алгоритм, использующий процедуру **LS** однократно.

Алгоритм **N+1LS** (Neighborhood + single Local Search)
Вход: граф $G = (V, E)$. **Выход:** кластерный граф C_{N1LS}. **Шаг 1.** Пусть F – множество всех кластерных графов, построенных алгоритмом **N** на шаге 1. Применить процедуру **LS** к ближайшему к G кластерному графу из F. Обозначить этот граф через C_{N1LS}.

Трудоемкость этого алгоритма – $O(|V|^3)$.

Гибридный эволюционный алгоритм

Алгоритм **N+mLS** обладает хорошей оценкой точности, но характеризуется высокой трудоёмкостью. Перспективной представляется идея сохранить преимущества процедуры **LS**, применяя её к ограниченному, не зависящему от размера графа множеству допустимых решений.

Гибридный эволюционный алгоритм

Алгоритм **E** (Evolution)
Вход: граф $G = (V, E)$. **Выход:** кластерный граф C_E. **Шаг 0.** Положить начальную популяцию $П^0$ равной множеству всех кластерных графов, построенных алгоритмом **N** на шаге 1. Положить $C_E = C_{N1LS}$, где C_{N1LS} – решение, построенное алгоритмом **N+1LS**. **Шаг 1.** Пока не **выполнен критерий остановки**, повторять следующее: **Итерация *k*.** Положить $П^k = \emptyset$. Пока в популяции не M особей, повторять: 1. Выбрать p_1, p_2 из $П^{(k-1)}$ оператором селекции. 2. Применить к p_1, p_2 оператор кроссинговера, получить c. 3. Применить оператор мутации к c, получить c'. 4. Добавить c' в популяцию $П^k$. Если в популяции есть особь C, лучшая, чем C_E, то положить $C_E = C$. Если **выполнен критерий встряски**, выполнить встряску популяции.

Параметры эволюционного алгоритма

Кодирование решений. Разбиение множества вершин графа $G = (V, E)$ на два кластера кодируется бинарным вектором длины $|V|$. При этом значение i-й координаты вектора определяет принадлежность вершины v_i: если оно равно 0, вершина относится к кластеру X, иначе – к кластеру Y. Без потери общности можно считать, что вершина v_1 всегда принадлежит кластеру X.

Размер популяции. Размер популяции M равен 128, за исключением нулевой популяции, которая формируется с помощью алгоритма **N**.

Критерий остановки. Если в течение последних 6 итераций не происходило обновление рекорда, вычисления останавливаются.

Параметры эволюционного алгоритма

Оператор кроссинговера. Для поддержания разнообразия популяции между итерациями применяется следующая процедура скрещивания. Сначала формируются кластеры, включающие вершины, принадлежащие одному кластеру у обоих родителей. На этом этапе может образоваться до 4 кластеров. Затем, пока количество кластеров превышает 2, на каждом шаге объединяются два кластера, чье слияние минимально увеличивает целевую функцию. Когда остается 2 кластера, выполняется проверка: если их объединение уменьшает целевую функцию, кластеры сливаются. В результате получается допустимое решение, содержащее не более двух кластеров.

Оператор мутации совпадает с процедурой **LS**.

Параметры эволюционного алгоритма

Пример работы оператора кроссинговера. Пусть граф содержит 4 вершины: $\{1, 2, 3, 4\}$.
Первая кластеризация: $X_1 = \{1, 2\}$, $Y_2 = \{3, 4\}$.
Вторая кластеризация: $X_2 = \{1, 2, 3\}$, $Y_2 = \{4\}$.
Попарно пересекаем все кластеры первой и второй кластеризации, получаем 3 непустых множества: $M_1 = \{1, 2\}$, $M_2 = \{3\}$, $M_3 = \{4\}$.
Далее, как минимум 2 кластера сливаются, после чего получившиеся кластеры снова могут быть слиты.

Параметры эволюционного алгоритма

Встряска и критерий встряски. В процессе эволюции популяция стремится к заполнению локальными оптимумами. Для выхода из них применяется процедура встряски, которая активируется каждую пятую итерацию. Процедура выполняется 5 раз для каждой особи: на каждом шаге каждая вершина с вероятностью 0.1 переносится в противоположный кластер независимо от изменения целевой функции.

Легко видеть, что алгоритм **E** является 3-приближенным алгоритмом решения задачи $\mathbf{CC}_{\leq 2}$ за счет начального рекорда, заданного алгоритмом **N+1LS**, а его трудоемкость – $O(|V|^3)$.

Вычислительный эксперимент

Для оценки эффективности представленных алгоритмов был проведен вычислительный эксперимент. Заметим, что для любого графа значение целевой функции решений, полученных алгоритмами **N+mLS**, **N+1LS** и **E**, не превышает значения для алгоритма **N**, последний был выбран в качестве бейслайна. Было вычислено среднее процентное улучшение (уменьшение) целевой функции остальных алгоритмов относительно этого бейслайна.

Вычислительный эксперимент

Эксперименты проводились на наборе данных вопросов и ответов с сайта Stack Overflow. Каждый объект в наборе данных помечен тегами (Java, JavaScript, ML и др.). Для адаптации данных к задаче кластеризации была введена мера схожести: для пары объектов вычисляется коэффициент Жаккара между множествами их тегов. Объекты считаются схожими, если этот коэффициент превышает 0.5. Для упрощения эксперимента были отобраны только вопросы с тегами из топ-10 по популярности на Stack Overflow.

Вычислительный эксперимент

Исследовались графы размером 1000, 2000, 3000, 4000 и 5000 вершин. Для генерации графа заданного размера из набора данных равномерно случайно выбиралось соответствующее количество объектов, после чего ребра формировались согласно описанной мере схожести. Для каждого размера графа было решено по 20 тестовых задач.

Вычислительный эксперимент

	N+1LS	**N+mLS**	**E**
1000	21.23	23.44	23.74
2000	20.31	23.15	23.32
3000	20.10	22.99	23.17
4000	20.56	23.70	23.84
5000	20.97	23.67	23.73

Среднее снижение значения целевой функции относительно бейслайна, %

Вычислительный эксперимент

	N	**N+1LS**	**N+mLS**	**E**
1000	0	0	1	3.25
2000	2.2	2.2	12	14.25
3000	9.4	9.45	46.5	41.1
4000	21.2	21.4	103.45	72.3
5000	39.75	40	191.95	113.75

Среднее время работы алгоритмов, сек

Вывод

1. Алгоритмы **N** и **N+1LS** самые быстрые из исследуемых, но их решения хуже, чем у алгоритмов **N+mLS** и **E**.
2. Алгоритм **E** в каждой решенной задаче дает наилучшее решение. Качество этих решений немного лучше, чем у алгоритма **N+mLS**.
3. С ростом числа вершин алгоритм **E** решает задачи быстрее, чем алгоритм **N+mLS**. Таким образом алгоритм **E** является оптимальным выбором при решении задачи $\mathbf{CC_{\leq 2}}$.

Спасибо за внимание.