ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ПОИСКА НИЖНИХ ОЦЕНОК ОПТИМАЛЬНОГО ЗНАЧЕНИЯ В ОДНОЙ ИЗ МОДЕЛЕЙ КЛАСТЕРНОГО АНАЛИЗА ГРАФОВ И СЕТЕЙ СВЯЗЕЙ, ВОЗНИКАЮЩИХ В ВЕБ-ПРОСТРАНСТВЕ, БИОЛОГИЧЕСКИХ И СОЦИАЛЬНЫХ СООБЩЕСТВАХ[1]

И.Л. Васильев, А.В. Ушаков

Институт динамики систем и теории управления СО РАН

e-mail: {vil, aushakov}@icc.ru

Аннотация

В статье рассматривается один из подходов к задаче кластерного анализа, основанный на применение модели задачи о p-медиане. Рассматривается известный алгоритм поиска нижних оценок для оптимального значения в задаче о p-медиане, базированный на построении релаксации Лагранжа, а также максимизации двойственной функции с помощью субградиентного метода. Предлагается эффективная схема распараллеливания такого алгоритма, включающая помимо прочего процедуру каскадной сборки данных между процессами. Разработанный алгоритм тестируется на искуственно сгенерированной задаче очень большой размерности.

1. Введение и постановка задачи

Рассмотрим постановку задачи о p-медиане. Пусть дано множество возможных пунктов размещения предприятий $I = \{1, \dots, m\}$, множество клиентов $J = \{1, \dots, n\}$, а также положительные величины d_{ij}, задающие затраты на обслуживание клиента $j \in J$ из предприятия расположенного в пункте $i \in I$. Задача о p-медиане состоит в размещении p предприятий из множества возможных пунктов I таким образом, чтобы суммарные затраты на обслуживание всех клиентов были минимальны. Отметим, что на практике часто полагают, что $I = J$, в этом случае задача может быть сформулирована на полном простом взвешенном ориентированном графе $G(I, A)$ с множеством вершин I и множеством дуг $A = \{ij : i \in I, j \in J, i \neq j\}$, причем каждой дуге приписывается вес d_{ij}, задающий расстояние между вершинами $i \in I$ и $j \in I$. Задача состоит в поиске p вершин графа $G(I, A)$, называемых медианами, таких, что сумма всех весов входящих дуг к немедианным вершинам от ближайшей медианы была минимальна. Задача о p-медиане может быть представлена в виде задачи целочисленного линейного программирования. С этой целью вводятся бинарные переменные y_i, x_{ij}, соответствующие вершинам и дугам графа. Переменная y_i принимает значение 1, если вершина i является медианой и 0 в противном случае; переменная x_{ij} равна 1, если вершина i является ближайшей медианой к j, 0 в противном случае. Дополнительно обозначим через $\delta^-(j) = \{i \in I | \ ij \in A\}$ множество

[1]Работа выполнена при финансовой поддержке интеграционного проекта СО РАН № 21, а также РФФИ в рамках проектов № 12-07-31198 мол-а, № 12-07-33045 мол_а_вед.

вершин, соединенных с j выходящей из них дугой, а через $\delta^+(i) = \{j \in I |\ ij \in A\}$ множество вершин соединенных с i выходящими из нее дугами. С использованием введенных переменных и обозначений задача о p-медиане может быть записана в следующей форме:

$$\min_{(x,y)} \sum_{(i,j)\in A} d_{ij}x_{ij} \tag{1}$$

$$\sum_{i\in\delta^-(j)} x_{ij} + y_j = 1 \qquad \forall\, j \in I, \tag{2}$$

$$x_{ij} \le y_i \qquad \forall\, (i,j) \in A, \tag{3}$$

$$\sum_{i\in I} y_i = p, \tag{4}$$

$$y_i \in \{0,\ 1\} \qquad \forall\, i \in I, \tag{5}$$

$$x_{ij} \in \{0, 1\} \qquad \forall\, (i,j) \in A. \tag{6}$$

Ограничения (2) гарантируют, что каждая вершина j является либо медианой, либо имеет одну входящую дугу из медианной вершины. Неравенства (3) исключают существование выходящих дуг из немедианных вершин. Количество медиан определяется уравнением (4). Условия на целочисленность переменных заданы ограничениями (5) и (6).

Задача о p-медиане является широко известной и одной из базовых задач в области дискретных задач размещения, впервые сформулированной в [1]. Отметим, что одним из важнейших приложений задачи о p-медиане является область интелектуальной обработки информации, а именно кластеризация и группирование данных, например в количественной психологии [2,3]. Одной из основных трудностей в разработке эффективных алгоритмов решения задачи о p-медиане является тот факт, что она NP-трудна [4]. Вопреки этому к сегодняшнему моменту был предложен ряд методов и подходов к ее решению как точных, так и эвристических, наиболее полный обзор которых представлен в работах [5,6]. Отметим, что самые большие задачи, которыми способны оперировать современные алгоритмы, имеют размерность порядка 90000 вершин. Результаты для таких примеров были получены, например, в работах [7,8,9].

2. Релаксация Лагранжа и субградиентный метод

Метод поиска нижних оценок, основанный на релаксации Лагранжа, довольно часто применяется к задаче о p-медиане. В работе рассматривается наиболее популярный тип релаксации исходной задачи, получающийся за счет ослабления ограничений (2). Данные ограничения добавляются в целевую функцию с некоторыми весами $\lambda \in \mathbb{R}^m$, называемыми множителями Лагранжа.

$$\mathcal{L}(\lambda) = \min_{(x,y)} \left\{ \sum_{(i,j)\in A} d_{ij}x_{ij} - \sum_{j\in I} \lambda_j \left(\sum_{i\in\delta^-(j)} x_{ij} + y_j - 1 \right) : \text{при (3)-(6)} \right\}.$$

Отметим, что функцию $\mathcal{L}(\lambda)$ называют двойственной функцией Лагранжа. При любом фиксированном наборе множителей $\lambda = (\lambda_1, \ldots, \lambda_m)$ значение двойственной функции является нижней оценкой оптимального значения исходной задачи.

Далее, для каждой вершины $i \in I$ вводятся величины

$$\rho_i(\lambda) = \sum_{j \in \delta^+(i)} \min\{0, d_{ij} - \lambda_j\} - \lambda_i,$$

называемые оценками Лагранжа и предполагается, что они упорядочены по возрастанию, т.е.

$$\rho_{i_1}(\lambda) \leq \cdots \leq \rho_{i_n}(\lambda).$$

Тогда оптимальное решение $(x_{ij}(\lambda), y_i(\lambda))$ релаксированной задачи записывается как

$$y_i(\lambda) = \begin{cases} 1, & i \in T(\lambda) \\ 0, & \text{в противном случае;} \end{cases}$$

$$x_{ij}(\lambda) = \begin{cases} 1, & y_i = 1 \wedge d_{ij} - \lambda_j < 0 \\ 0, & d_{ij} - \lambda_j \geq 0; \end{cases}$$

а значение двойственной функции Лагранжа может быть подсчитано в явном виде по формуле

$$\mathcal{L}(\lambda) = \sum_{i \in T(\lambda)} \rho_i(\lambda) + \sum_{i \in I} \lambda_i,$$

где $T(\lambda)$ — множество, состоящее из p вершин с минимальной оценкой Лагранжа.

Для поиска наилучшей нижней оценки оптимального значения необходимо максимизировать двойственную функцию Лагранжа, т.е. решить задачу $\max\limits_{\lambda \in \mathbb{R}^m} \mathcal{L}(\lambda)$. Поскольку двойственная функция Лагранжа является негладкой, то для ее максимизации используется субградиентный алгоритм, находящий максимум посредством итерационной формулы

$$\lambda^{k+1} = \lambda^k + \alpha_k g(\lambda^k),$$

где $g(\lambda^k)$ — вектор субградиента подсчитанный на k-ой итерации по формуле

$$g_j(\lambda^k) = 1 - \sum_{i \in \delta^-(j)} x_{ij}(\lambda^k) - y_j(\lambda^k).$$

Шаг метода α_k подсчитывается по следующему эвристическому правилу

$$\alpha_k = \frac{\phi \cdot (1.05 \cdot BUB - \mathcal{L}(\lambda^k))}{\|g(\lambda^k)\|_2^2}, \tag{7}$$

где BUB — верхняя оценка оптимального значения, $\|\cdot\|_2$ — евклидова норма, а ϕ — параметр.

3. Распараллеливание алгоритма

Идея параллельного субградиентного алгоритма состоит в следующем. Фактически на каждой итерации алгоритма подсчет новых значений множителей Лагранжа, вектора

оценок Лагранжа и вектора субградиента может производиться частями разными процессами независимо друг от друга, а следовательно процессам необходимо оперировать лишь некоторыми непересекающимися подгруппами столбцов матрицы расстояний. Таким образом итерационный процесс может быть успешно распараллелен.

При реализации параллельного алгоритма использовался интерфейс обмена сообщениями MPI. В качестве схемы взаимодействия между процессами использовалась модель Master-Slave, в рамках которой выделяется один процесс — Master, управляющий работой всех остальных, называемых Slave-или вычислительные процессы.

Перед началом основного цикла управляющий процесс считывает входные данные, состоящие из набора координат вершин графа, и устанавливает значения параметров алгоритма, таких как: количество вершин m, медиан p, начальное количество активных элементов в каждом столбце n_0^j, а также диапазон индексов столбцов, т.е. размер блока, который каждый из Slave-процессов будет обрабатывать, в форме (j_q, n_q), где j_q — индекс первого столбца в блоке для данного процесса, n_q — размер блока, а $q = 1, \ldots, s$ — номер процесса. По завершению он формирует и отправляет всем вычислительным процессам сообщения, содержащие не только все считанные координаты вершин, но и индивидуальный набор параметров. На основании полученных данных каждый Slave-процесс вычисляет свою собственную часть матрицы расстояний $D = \{d(i,j)\}$, $i = u_1(j), \ldots, u_{n_0^j}(j)$; $j = j_q, \ldots, j_q + n_q$, а также устанавливает начальные значения множителей Лагранжа $\lambda_j^0 := d(u_1(j), j)$.

Основной цикл алгоритма начинается с подсчета Slave-процессами части вектора оценок Лагранжа $\rho_i(\lambda^k)$, $i \in I$ на основании столбцов из своего блока. Поскольку каждый столбец матрицы расстояний отсортирован по возрастанию и хранится в памяти не полностью, то затруднительно определить заранее какие компоненты вектора оценок Лагранжа и на каком процессе будут отличны от нуля (т.е. заполнены), что вынуждает каждый Slave-процесс передавать управляющему два вектора, один из которых содержит полученные значения оценок Лагранжа, а другой — соответствующие им индексы. Другой информацией передаваемой на этом этапе является сумма множителей $\sum_{i=j_q}^{j_q+n_q} \lambda_i^k$ необходимая для подсчета значения двойственной функции Лагранжа $\mathcal{L}(\lambda^k)$.

Получив эти данные Master-процесс собирает полный вектор оценок, сортирует его, подсчитывает $\mathcal{L}(\lambda^k)$ и текущую нижнюю оценку BLB, а также находит множество $T(\lambda^k)$, проверяя затем критерий остановки (шаг 2 и 3). При выполнении критерия всем вычислительным процессам отправляется сообщение о завершении работы, в противном случае каждый Slave-процесс получает сообщение, содержащее множество индексов $T(\lambda^k)$.

На основе полученного множества $T(\lambda^k)$ вычислительные процессы находят вектор $y(\lambda^k)$. Подсчитав вектор $y(\lambda^k)$ и используя свой блок столбцов, каждый Slave-процесс вычисляет часть вектора субградиента $g_j(\lambda^k)$, $j = j_q, \ldots, j_q + n_q$, а также норму $||g(\lambda^k)||_2$, которую отсылает управляющему процессу.

Просуммировав части нормы вектора субградиента Master-процесс вычисляет параметр ϕ, а также проверяет критерий остановки (шаг 6). Если критерий выполняется, то каждому Slave-процессу отправляется сообщение о завершении работы алгоритма. В противном случае управляющий процесс подсчитывает и отправляет всем Slave-процессам

шаг α_k.

Наконец, каждый процесс, используя полученный шаг, может подсчитать новые значения своей части множителей Лагранжа $\lambda_j^{k+1} = \lambda_j^k + \alpha_k\, g(\lambda^k)$, $j = j_q, \ldots, j_q + n_q$ и переходит на следующую итерацию алгоритма.

4. Результаты вычислительного эксперимента

Представленная выше схема параллельного субградиентного алгоритма была реализована на языке C++ с использованием библиотеки MPICH 1.2.7 и протестирована на вычислительном кластере Blackford (ИДСТУ СО РАН), имеющим следующие характеристики: 20 вычислительных узлов, 40 процессоров Intel Xeon Quad-Core EM64T, 160 ядер; пиковая производительность 1.493 TFlops; суммарный объем оперативной памяти на узлах — 160 GB. В качестве тестового примера использовалась задача очень большой размерности с количеством вершин равным 1000000 и $p = 255$, сгенерированный по правилу, предложенному в работе [10]. Отметим, что количество переменных в такой задаче составляет 10^{12}.

Вычислительный эксперимент проводился на 80-ти процессорных ядрах вычислительного кластера. Относительная погрешность известной верхней оценки относительно нижней составило порядка 0.346 процента. Общее время вычисления, включая подсчет матрицы расстояний, составило 22101.57 секунд (чуть более шести часов).

ЛИТЕРАТУРА

[1] Hakimi S.L. Optimum distribution of switching centers in a communication network and some related graph theoretic problems // Operations Research. 1964. V. 13, № 3. 462–475.

[2] Brusco M.J., Köhn. H.F. Optimal partitioning of a data set based on the p-median model // Psychometrika. 2008. V. 73, № 1. P. 89–105.

[3] Köhn H.F., Steinley D., Brusco M.J. The p-median model as a tool for clustering psychological data // Psychological Methods. 2010. V. 15, № 1. P. 87–95.

[4] Kariv O., Hakimi S.L. An algorithmic approach to network location problems; part 2. The p-medians // SIAM Journal on Applied Mathematics. 1979. V. 37, № 3. 539–560.

[5] Mladenović N., Brimberg J., Hansen P., Moreno-Pérez J.A. The p-median problem: A survey of metaheuristic approaches // EJOR. 2007. V. 179, № 3. 927–939.

[6] Reese J. Solution Methods for the p-Median Problem: An Annotated Bibliography // Networks. 2006. V. 28, № 3. 125–142.

[7] Avella P., Boccia M., Salerno S., Vasilyev I. An aggregation heuristic for large scale p-median problem // Computers & Operations Research. 2012. V. 39, № 7. 1625–1632.

[8] Garcia S., Labbé M., Marin A. Solving large p-median problems with a radius formulation // INFORMS Journal on Computing. 2011. V. 23, № 4. 546–556.

[9] Hansen P., Brimberg J., Urosević D., Mladenović N. Solving large p-median clustering problems by primal–dual variable neighborhood search // Data Mining and Knowledge Discovery. 2009. V. 19, № 3. 351-375.

[10] Zhang T., Ramakrishnan R., Livny M. BIRCH: an efficient data clustering method for very large databases // Journal of the American Statistical Association. 1996. V. 98, № 463. 103–114.