алгоритмическая математика и математическое моделирование

Компьютерные инструменты в образовании, 2017
№ 2: 16–24
УДК: 519.6
http://ipo.spb.ru/journal

ОБ ОПТИМИЗАЦИИ АЛГОРИТМА ПОСТРОЕНИЯ СТАЦИОНАРНОГО ПОТОКА НА ОРИЕНТИРОВАННОМ ГРАФЕ

Сергеев В.Д.[1]

[1]СПбГУ, Санкт-Петербург, Россия

Аннотация

В работе рассмотрен метод получения классификационного признака изображений, основанный на построении стационарного потока на графе, построенном по данному изображению. На основании начального и стационарного потоков вычисляется взвешенная энтропия, которая рассматривается как классификационный признак. В работе рассмотрены различные подходы к оптимизации данного алгоритма. Один из предложенных подходов состоит в делении исходного изображения на некоторое число областей, вычисления для которых можно производить независимо друг от друга на разных ядрах процессора. Второй подход основан на разбиении изображения на ячейки заданного размера и построении графа, вершинам которого сопоставлены эти ячейки. Третий подход состоит в использовании в представлении данных так называемых неизменяемых коллекций, позволяющих проводить распараллеливание без синхронизации. Приведены сравнительные результаты численных экспериментов.

Ключевые слова: *анализ изображений, стационарный поток на графе, взвешенная энтропия.*

Цитирование: Сергеев В.Д. Об оптимизации алгоритма построения стационарного потока на ориентированном графе // Компьютерные инструменты в образовании. 2017. № 2. С. 16–24.

1. ВВЕДЕНИЕ

Многие цифровые изображения, связанные с функционированием сложных систем, можно рассматривать как их фазовые портреты. Определение основных характеристик фазовых портретов помогает выявлять свойства изучаемых процессов. Классы изучаемых изображений определяют выбор методов их анализа и классификации — морфологических, статистических, фрактальных, текстурных.

К изображениям, связанным с процессом распространения вещества в среде, можно применять метод моделирования, основанный на построении марковской цепи на специальном графе и нахождении ее стационарных состояний [1], существование которых означает, что у исходной системы есть инвариантное множество.

К построению стационарных процессов на графе могут быть сведены задачи из разных областей математики, таких как линейное программирование (транспортные зада-

чи) или теория динамических систем (построение инвариантной меры на графе символического образа).

Для изучения динамики процесса можно оценить разницу между начальным состоянием и полученным стационарным. Поскольку эти состояния задаются в форме вероятностных распределений, целесообразно использовать расхождения Реньи, в частности расхождение Кульбака-Лейблера. В задачах, связанных с потоками на графах, величина этого расхождения, взятая с противоположным знаком, называется взвешенной энтропией. Фактически эту величину можно интерпретировать, как «расстояние» между распределением начальным и стационарным. В работах [1, 2] было показано, что взвешенная энтропия может использоваться как классификационный признак при анализе изображений. Один из методов построения стационарного потока на ориентированном графе был предложен ленинградским архитектором Г.В. Шелейховским [8], который решал задачу прогнозирования устойчивости расселения жителей города по районам — транспортную задачу с линейными ограничениями. Этот метод получил название метода балансировки (метода Шелейховского). Он заключается в последовательном преобразовании строк и столбцов матрицы смежности этого графа умножением и делением на подходящие коэффициенты (коэффициенты балансировки) так, чтобы построенное решение удовлетворяло заданным ограничениям. Позже Л.М. Брэгман [4] доказал сходимость этого метода, независимость результата от порядка выбора строк и столбцов, а также обобщил метод Шелейховского на класс задач, ограничения которых не являются линейными [5]. И.В. Романовский [5] показал, что построенный стационарный поток максимизирует взвешенную энтропию.

Алгоритм балансировки может сходиться медленно, если соответствующие коэффициенты оказываются близки к 1. Эту ситуацию трудно предусмотреть заранее, так как она обусловлена выбором начального распределения. Поэтому для практического применения, особенно в случае графов достаточно больших размеров, нужна оптимизация.

Одним из способов сокращения времени вычислений является распараллеливание алгоритма построения стационарного потока. Изображение разбивается на части, на каждой из которых построение стационарного потока происходит независимо от других. Описание и реализация алгоритма были выполнены в работах [3, 4].

В работе [9] был предложен вариант оптимизации, основанный на замене пикселя ячейкой определенного размера, что позволяет значительно сократить размеры графа. Существенным моментом этого метода является выбор размера ячейки так, чтобы минимизировать искажения, вносимые в исходное изображение. Результат также зависит от выбора меры ячейки. Предложенный способ оптимизации позволяет сократить время вычислений в среднем в 2–3 раза без использования распараллеливания.

В данной работе мы рассматриваем варианты оптимизации, основанные на модификации способа представления данных для перебора вершин графа и распараллеливания отдельных частей алгоритма.

В базовой реализации алгоритма в качестве перебора вершин используется приоритетная очередь [7], что оказывается ресурсоемким решением, так как при каждом получении элемента из очереди (содержащей все вершины графа) она перестраивается. Мы заменяем приоритетную очередь на список вершин, в который складываем вершины, на дугах которых необходимо произвести балансировку. Для увеличения скорости работы алгоритма также были распараллелены операция выбора вершин и вычисление взвешенной энтропии. Проведенные эксперименты показали, что время работы алгоритма сокращается в среднем в 2–3 раза.

2. БАЗОВАЯ МОДЕЛЬ

Представим изображение как прямоугольную решетку пикселей (i, j). По этой решетке строится ориентированный граф, каждому узлу (вершине) которого сопоставляется значение интенсивности соответствующего пикселя $0 < I(i, j) \leqslant 255$.

Сразу отметим, что в процессе обработки изображений нулевые значения яркостей мты будем заменять некоторым малым значением, чтобы избегать нулевых значений мер на дугах графа. Рассмотрим окрестность ближайших соседей пикселя, состоящую из 4 пикселей, расположенных по вертикали и горизонтали. У пикселей, расположенных на границе решетки, количество соседей соответственно уменьшается. Всем вершинам графа приписываем веса, равные значениям интенсивностей соответствующих пикселей. Всем выходящим из вершины дугам приписываем значение интенсивности узла, деленное на число соседей. Значения в вершинах и дугах графа нормируем таким образом, чтобы сумма мер (весов) всех вершин (а тем самым и дуг) была равна 1. Обозначим полученный поток p_{ij}. По построению мера каждой вершины равна сумме мер выходящих дуг. Получаем марковскую цепь на графе.

Мы стремимся получить такое распределение $u = \{u_{ij}\}$ значений на дугах, при котором построенная цепь будет стационарной, то есть для каждой вершины графа сумма весов входящих в нее дуг равна сумме весов выходящих. Стационарное состояние может быть охарактеризовано с помощью энтропии стационарного процесса [9].

Величина, которая оценивает разницу между начальным состоянием и стационарным, называется взвешенной энтропией и определяется следующим образом

$$v = -\sum_{ij} u_{ij} \ln\left(\frac{p_{ij}}{u_{ij}}\right). \tag{1}$$

Рассмотрим задачу поиска максимума функции

$$f(u) = -\sum_{ij} u_{ij} \ln\left(\frac{p_{ij}}{u_{ij}}\right) \tag{2}$$

при условиях

$$\sum_i u_{ik} = \sum_j u_{kj}, \quad \sum_{ij} u_{ij} = 1. \tag{3}$$

Начальный поток p_{ij} рассматривают как приближенное решение этой задачи, хотя он и не удовлетворяет ограничениям. Построение стационарного потока с помощью метода балансировки дает решение этой задачи с некоторой заданной точностью ε, при этом полученный поток доставляет максимум взвешенной энтропии; решение существует, если на графе есть циклы [5].

3. БАЗОВЫЙ АЛГОРИТМ

Алгоритм построения стационарного потока на графе основан на методе балансировки Шелейховского-Брэгмана [5, 6, 8]. Для организации перебора вершин в работах [2, 3] использовалась приоритетная очередь, приоритетом в которой является дисбаланс вершины

$$q(n) = |\mu_{out}(n) - \mu_{in}(n)|, \tag{4}$$

где $\mu_{out}(n)$— сумма мер исходящих из вершины n дуг, а $\mu_{in}(n)$ — сумма мер входящих. На каждом шаге для вершины n с максимальным приоритетом вычисляется коэффициент:

$$\gamma = \sqrt{\frac{\mu_{out}(n)}{\mu_{in}(n)}}. \tag{5}$$

Для каждой вершины n меры всех дуг, исходящих из n, делятся на этот коэффициент, а меры входящих умножаются. После этого производится нормировка потока. Критерием остановки алгоритма является то, что дисбаланс вершин становится меньше некоторой заранее заданной величины. В работах [3, 4] этот метод был использован для получения значений взвешенной энтропии для класса изображений растворов ионов серебра различной концентрации. Было показано, что это значение может быть использовано как классификационный признак.

4. ОПТИМИЗАЦИЯ АЛГОРИТМА

4.1. Распараллеливание по частям изображения

В работе [4] был предложен метод распараллеливания базового алгоритма, который состоит в том, чтобы разделить исходные данные на некоторое число подмножеств, вычисления для которых можно производить независимо друг от друга. Простейшим является случай разбиения изображения на четыре решетки так, что изображение делится на две равные части по каждому из измерений. Пиксели, находящиеся на границах частей (граничные точки), входят в каждую. На каждой из таких решеток строится свой собственный граф, на котором все вычисления производятся независимо от вычислений на других графах. Полученные распределения объединяются в одно общее, так что в качестве значения дугам, входящим и выходящим из граничных точек и принадлежащим нескольким частям, присваивается вес, равный сумме весов соответственно входящих и исходящих ребер каждой из частей. Принцип объединения проиллюстрирован ниже.

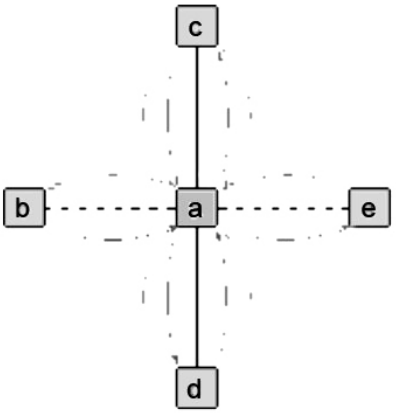


Рис. 1. Принцип вычисления мер граничных точек

Эксперименты показали [4], что при вычислениях по модифицированному алгоритму процент точности падает с 96% до 92% при разбиении изображения на 4 части и до

76% при аналогичном разбиении на 16 частей. При этом скорость вычислений по модифицированному алгоритму с изображением разбитым на 4 части даже при использовании 1 ядра процессора выше в 2,4 раза, а при использовании 4 ядер выше почти в 8 раз. При разбиении на 16 частей скорость работы алгоритма на 4 ядрах практически в 10 раз превосходит скорость работы базового алгоритма.

4.2. Сегментация изображения

В работе [9] была предложена модификации алгоритма, состоящая в разделении исходного изображения на ячейки заданного размера и построении графа, вершинам которого сопоставлены не пиксели, а ячейки. Вес вершины этого графа равен мере соответствующей ячейки. Мера ячейки определяется как сумма интенсивностей пикселей ячейки, деленная на сумму мер всех ячеек. Выбор размера ячейки зависит от изображения и определяется эмпирически. Понятно, что выбор небольшого размера ячейки позволит построить более адекватное представление изображения, так как в этом случае ячейка имеет более однородную по интенсивности структуру. Данный вариант оптимизации приводит к сокращению времени в среднем в 2–3 раза без использования параллельных вычислений. Эксперименты, подтверждающие эти выводы, были проведены в различных цветовых палитрах: grayscale, RGB, HSV. Хорошие результаты были получены на компонентах пространства HSV и grayscale палитре, в то время как при расчетах в пространстве RGB не всегда просматривалась закономерность результатов.

4.3. Изменение метода перебора вершин

Опишем в данном разделе предлагаемый нами вариант оптимизации поиска стационарного потока на графе.

В базовом алгоритме (ячейки из одного пикселя) для перебора вершин используется так называемая приоритетная очередь, где приоритет вершины рассчитывается по формуле (4). Очередь с приоритетом — абстрактный тип данных в программировании, поддерживающий две обязательные операции — добавить элемент и извлечь максимум. Очередь с приоритетами может быть реализована на основе различных структур данных. Простейшие реализации могут использовать упорядоченный массив, связный список, подходящие в основном для небольших очередей. При этом вычисления могут быть как «ленивыми» (сложность вычислений переносится на извлечение элемента), так и ранними, когда вставка элемента сложнее его извлечения. Другими словами, одна из операций может быть произведена за время $O(1)$, а другая — за $O(N)$ или наоборот в зависимости от реализации (ленивая или ранняя), где N — длина очереди. Более эффективными являются реализации на основе кучи, где обе операции можно производить в худшем случае за время $O(\ln N)$.

Особенностью приоритетной очереди является ее регулярное перестроение, на что уходит большая часть времени работы алгоритма. В предлагаемой нами оптимизации реализован вариант перебора вершин с хранением их в виде списка, в который вершины складываются по критерию (4), таким образом операция добавления осуществляется за время $O(1)$.

Приоритетная очередь строится из всех вершин графа, и при удалении или добавлении вершин эта очередь перестраивается, что дает накладные расходы по памяти, так как мы храним очередь всех вершин графа. Таким образом, сложность такого представления по памяти есть $O(M)$, где M — количество вершин в графе. Временная слож-

ность алгоритма при использовании приоритетной очереди зависит от ее реализации: при неэффективной реализации (связный список, упорядоченный массив) получение элемента с высшим приоритетом получается за $O(M)$, а при более эффективной (куча) $O(\ln M)$.

Использование для перебора вершин списка, в который добавляются вершины по критерию (4), дает существенный прирост в производительности, так как мы не храним в списке данные для всех вершин, что дает нам оценку по памяти $O(M)$ в худшем случае и $O(1)$ в среднем, в то время, как в случае с приоритетной очередью сложность по памяти всегда $O(M)$. Также мы получаем ускорение по времени работы алгоритма, так как за один проход по всем вершинам мы строим список за время $O(M)$, причем этот список содержит необходимые вершины, и нам не надо больше следить за приоритетами, потому что в результирующем списке будут только те вершины, по которым надо произвести балансировку.

4.4. Распараллеливание операций алгоритма

В последнее время стали популярны подходы с использованием неизменяемых (или immutable) структур данных, в частности коллекций неизменяемых структур данных. Это связано с удешевлением оперативной памяти, необходимой для таких коллекций. Неизменяемость позволяет отследить каждое состояние объекта в процессе его обработки. Обработку коллекций неизменяемых данных можно распараллелить, так как для работы с ними из любого числа потоков не нужна синхронизация.

Синхронизация необходима для сохранения целостности данных, когда в нескольких потоках изменяется состояние ресурса. Обработка коллекций неизменяемых объектов подразумевает построение цепочки таких преобразований данных, что после каждого преобразования из данной цепочки мы получаем новую неизменяемую коллекцию данных, что позволяет отслеживать и наблюдать изменение состояния данных в процессе обработки. Этот подход дает возможность распараллеливать работу с данными без синхронизации, хотя требует дополнительное выделение памяти на новые неизменяемые коллекции.

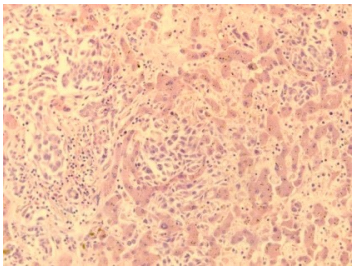
В процессе работы алгоритма, когда состояния вершин и ребер графа не изменяются, можно преобразовать коллекции вершин и ребер в иммутабельные. Приведение коллекций к подобному состоянию позволяет организовать распараллеливание на нескольких виртуальных ядрах, но в то же время приводит к затратам по памяти. Мы реализовали приведение к иммутабельной коллекции средствами Stream API языка Java [10].

Операциями, в которых не происходит изменения элементов коллекций являются: выбор вершин по критерию дисбаланса, расчет взвешенной энтропии, расчет потоков p_{ij} и u_{ij}.

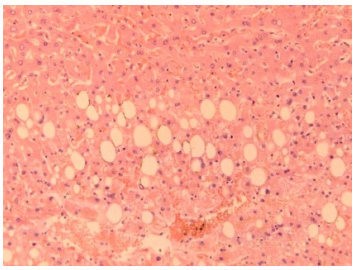
5. РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТОВ

Мы выполнили экспериментальную проверку выбранного метода оптимизации. В проведенных экспериментах использовались базовая модель алгоритма, его оптимизация с использованием списка и оптимизация с распараллеливанием некоторых шагов алгоритма. Расчеты проводились на классах изображений биомедицинских препаратов в палитрах RGB и grayscale (изображения были предоставлены отделом патологической анатомии Мариинской больницы Санкт-Петербурга). Были выбраны три класса изобра-

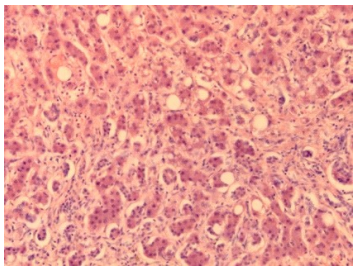
жений: дистрофия, метастазы и цирроз печени. Для экспериментов было выбрано 22 изображения, имеющих размеры 2584 × 1936 пикселей.



(а) метастазы печени



(б) жировая дистрофия печени



(в) цирроз печени

Рис. 2. Примеры из 3 классов изображений

Расчеты проводились на 4-х ядерной машине с операционной системой CentOS6.7.

Таблица 1. Расчеты в палитре grayscale

Изображение	Время работы с использованием кучи (сек.)	Время работы с использованием списка (сек.)	Время работы с использованием списка и распараллеливания операций алгоритма (сек.)
дистрофия_1.jpg	17,2	7,9	6,3
дистрофия_2.jpg	32,4	21,1	19,8
дистрофия_3.jpg	22,8	19,9	17,8
дистрофия_4.jpg	26,6	21,9	20,3
дистрофия_5.jpg	19,4	10,9	10,5
дистрофия_6.jpg	28,4	8,4	7,5
дистрофия_7.jpg	15,7	10,8	10,0
дистрофия_8.jpg	16,1	12,9	7,2
дистрофия_9.jpg	15,6	10,5	9,6
дистрофия/10.jpg	16,7	14,8	7,0
метастазы_1.jpg	14,4	6,0	5,8
метастазы_2.jpg	33,9	22,0	6,5
метастазы_3.jpg	18,0	8,8	8,5
метастазы_4.jpg	17,4	8,9	8,2
метастазы_5.jpg	16,2	10,6	10,2
метастазы_6.jpg	28,4	8,5	7,7
метастазы_7.jpg	24,8	12,8	10,3
метастазы_9.jpg	28,7	9,4	7,5
метастазы_10.jpg	15,1	9,0	8,0
цирроз_1.jpg	23,0	7,6	6,9
цирроз_2.jpg	21,2	11,3	9,7

Во всех экспериментах значения взвешенной энтропии совпадали. Хорошие результаты по времени были получены в палитре grayscale (табл. 1) и компоненте blue палитры RGB. Нужно отметить, что иногда время работы алгоритма с распараллеливанием операций больше, чем время работы алгоритма с перебором вершин с использованием списка. Это связано с тем, что операционная система может предоставлять для парал-

лельной обработки меньше виртуальных ядер, и в этих случаях выделяется одно ядро и параллельная обработка превращается в псевдопараллельную на одном ядре.

Как следует из приведенных данных, основной выигрыш по времени получается за счет изменения структуры данных, а добавление распараллеливания хорошо показывает себя при большом количестве свободных виртуальных ядер.

6. ЗАКЛЮЧЕНИЕ

Предложенные в данной статье варианты оптимизации алгоритма построения стационарного потока на ориентированном графе методом балансировки позволяют получить ускорение работы базового алгоритма в среднем в 2–3 раза. Наши рассмотрения показывают, что алгоритмическая сложность описанных оптимизаций алгоритма меньше алгоритмической сложности базового, что подтверждают проведенные эксперименты. Преимуществом данных оптимизаций алгоритма является возможность их эффективного применения на многоядерных вычислителях.

Список литературы

1. *Ампилова Н.Б.* Стационарные процессы на графах и анализ изображений // Компьютерные инструменты в образовании. 2013. № 2. С. 24–29.
2. *Ампилова Н.Б., Романовский И.В., Петренко Е.И.* О максимизации энтропии при линейных ограничениях / Труды Международной научной конференции «Космос, астрономия и программирование». 2008. С. 181–185.
3. *Батюков А.* Анализ цифровых изображений, основанный на построении стационарного потока на графе // Вестник Санкт-Петербургского университета. Серия 10. Прикладная математика, информатика, процессы управления. 2015. № 2. С. 115–122.
4. *Батюков А.М.* Классификация изображений биомедицинских препаратов адаптированным для параллельных вычислений алгоритмом построения стационарного потока // Сборник трудов Международ ной конференции «Актуальные проблемы прикладной математики, информатики и механики». М.: ФИЗМАТЛИТ, 2015. С. 39–43.
5. *Брэгман Л.М.* Доказательство сходимости метода Г.В. Шелейховского для задачи с транспортными ограничениями // Журнал вычислительной математики и математической физики. 1967. Т. 7, № 1. С. 147–156.
6. *Брэгман Л.М.* Релаксационный метод нахождения общей точки выпуклых множеств и его применение для решения задач выпуклого программирования // Журнал вычислительной математики и математической физики. 1967. Т. 7, № 3. С. 620–631.
7. *Кормен Т.Х.* Алгоритмы: построение и анализ, 2-е изд. М.: Вильямс, 2006.
8. *Шелейховский Г.В.* Композиция городского плана как проблема транспорта. М.: Гипрогор, 1946.
9. *Ampilova N.B., Sergeev V.D., Soloviev I.P.* On the method of digital image analysis based on the construction of a stationary flow on graph // Университетский научный журнал 22 (2016), p. 29–36, http://submit.uni-journal.ru/article/11818 (дата обращения 20.05.2016).
10. https://docs.oracle.com/javase/8/docs/api/java/util/stream/package-summary.html (дата обращения 20.05.2016).

Поступила в редакцию 20.02.2017, окончательный вариант — 30.03.2017.

Computer tools in education, 2017
№ 2: 16–24
http://ipo.spb.ru/journal

ON THE ALGORITHM OPTIMIZATION FOR CALCULATION A STATIONARY FLOW ON AN ORIENTED GRAPH

Sergeev V.D.[1]

[1]SPbSU, Saint-Petersburg, Russia

Abstract

A method for obtaining the classification attribute of images based on the construction of a stationary flow on a graph constructed from a given image is considered. Weighted entropy, which is considered as a classification attribute, is calculated by the initial and stationary flows. Various approaches for optimizing this algorithm are considered in the paper. One of the proposed approaches is to divide the original image into a number of areas, the calculations for which can be performed independently from each other on different processor cores. The second approach is based on splitting the image into cells of a given size and constructing a graph whose vertices are associated with these cells. The third approach is to use in the data representation the so-called immutable collections that allow parallelization without synchronization. Comparative results of numerical experiments are presented.

Keywords: *image analysis, stationary flow on graph, weighted entropy.*

Citation: V.D. Sergeev, “Ob optimizatsii algoritma postroeniya statsionarnogo potoka na orientirovannom grafe” [On the Algorithm Optimization for Calculation a Stationary Flow on an Oriented Graph], *Computer tools in education*, no. 2, pp. 16–24, 2017 (in Russian).

Received 20.02.2017, the final version — 30.03.2017.

Vladislav D. Sergeev, Postgraduate student at the Department of Informatics in Saint-Petersburg State University, vlad.sergeev.spbgu@gmail.com

**Сергеев Владислав Дмитриевич,
аспирант кафедры информатики СПбГУ,
vlad.sergeev.spbgu@gmail.com**