Построение рекурсивно-параллельных алгоритмов решения задач вычислительной геометрии на основе стратегии «распределяй и властвуй»

В.Н. Терещенко

В работе рассматривается один из подходов построения эффективных алгоритмов решения широкого класса задач вычислительной геометрии – рекурсивно-параллельные алгоритмы, в основе которых лежит стратегия «распределяй и властвуй». Вчастности, на примере задачи « О ближайшей паре» предложен рекурсивно-параллельный алгоритм ее решения, причем этап слияния результатов на каждом шаге рекурсивного подъема представлен в двух вариантах.

1. Введение

Одним из важных направлений построения эффективных алгоритмов решение задач вычислительной геометрии есть создание параллельных алгоритмов. Особенностью таких задач является то, что при наличии их внутреннего параллелизма, они мало исследованы относительно применения параллельных алгоритмов их решения. Поэтому возникает интерес к разработке эффективных параллельных алгоритмов решения некоторых из этих задач, и в частности, на основе общей рекурсивной стратегии "распределяй и властвуй". Для определенной части задач вычислительной геометрии существуют последовательные рекурсивные алгоритмы их решения, на основе выше упомянутой стратегии. Согласно этой стратегии входное множество точек плоскости на каждом шаге рекурсии разбивается на два подмножества, пока не сформируется поточечное разбиение, а потом осуществляется шаг слияния результатов и построения окончательного решения.

В работе, на примере задачи "о ближайшей паре", рассматривается теоретическая модель построения параллельного алгоритма ее решение с использованием идеи последовательного алгоритма "распределяй и властвуй", который имеет оценку сложности *O(NlogN)* [1].

2. Математическая модель алгоритма

2.1 Постановка задачи

На плоскости задано множество S из N точек. Найти две из них, расстояние между которыми наименьшее.

Последовательный алгоритм решения этой задачи со временем *O(NlogN)* можно построить, осуществив сначала предварительную обработку множества точек, а потом, построив бинарное дерево, выполнить два прохода по нему: разбиение заданного множества S на подмножества одинаковой мощности S_1, S_2 с определением медианы l и слияния результатов [1]. При построении параллельного алгоритма решения поставленной задачи будет использована эта идея.

2.2 Предварительная обработка

Пусть заданное множество *S* из *N* точек $S = \{ P_1, P_2, \ldots, P_N \}$ на плоскости (рис.1). И пусть заданы *O(N)* процессоров: $П_1, П_2, ..., П_{N/2}$.

1.Сначала построим отсортированный список точек по *x* координате $U_x = \{P_{1x}, P_{2x}, \ldots, P_{Nx}\}$, а потом отсортированный список точек по *y* координате $U_y = \{P_{1y}, P_{2y}, \ldots, P_{Ny}\}$. На рис.1 рассматривается случай для N=12.

2. Из списков U_x, U_y формируем массив точек U = {P_{ij}, i, j = 1, N}, где i - индекс, который указывает номер точки в упорядоченном списке U_x, а j –индекс в списке U_y.

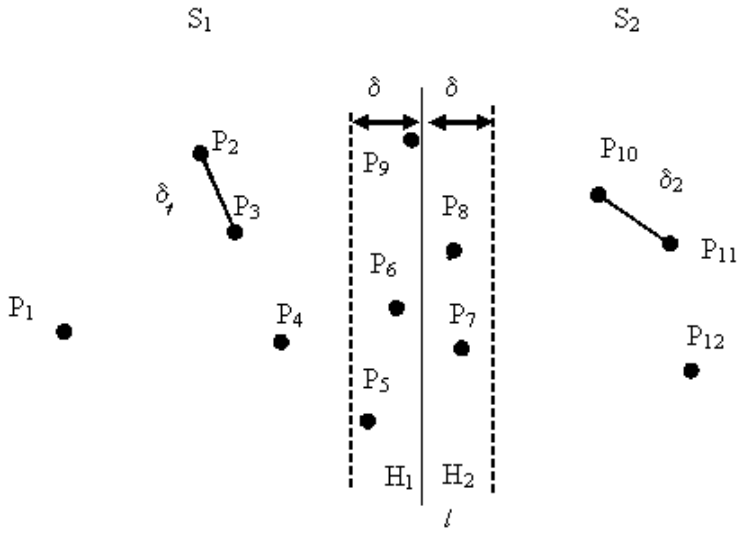

Рис.1. Упорядоченный список U={P_{51}, P_{122}, P_{13}, P_{74}, P_{45}, P_{66}, P_{87}, P_{118}, P_{39}, P_{1010}, P_{211}, P_{912}}

Сформированный таким образом список подается на вход алгоритма, граф которого имеет вид бинарного дерева (рис.2.). В этом дереве каждый узел обозначен целым числом k, относительно которого разбивается список точек в узлах на два равной мощности, относительно медианы, списка, после сравнения первых индексов точек массива P_{ij}. А каждый номер узла k определяется за один проход по дереву, если известное количество точек заданного множества, по формуле:

$$k = [\ (m+M)/2], \qquad (1)$$

где m -номер первого элемента списка, M - последний элемент списка.

2.3 Разбиение множества точек (рекурсивный спуск)

Этот этап алгоритма состоит в разбиении на каждом шаге рекурсии заданного множества точек, в виде списка U, на подмножества U_1, U_2 равной мощности, поиска медианы l (вертикали) и передачи U_1, U_2 на следующий шаг рекурсии. Процесс разбиения завершается, если в подмножествах разбиения останется по одной точке. Структура данных, которая бы поддерживала этот процесс, представлена на рисунке 2. В листьях дерева будут отдельные точки с U.

В случае последовательного алгоритма общее время рекурсивного спуска будет $O(Nlog_2N)$, поскольку на каждом шаге рекурсии передача данных выполняется за время $O(M)$, а поиск медианы на упорядоченном по координате x индексированном массиве точек P_{ij} выполняется за константное время $O(1)$. Медиана определяется по формуле $l = (P_{kj}+P_{k+1j})/2$, где k определяется из соотношения (1), а M - количество точек в списке. Время, необходимое на рекурсивный спуск в случае параллельного алгоритма, определяется следующей леммой.

Лемма 1. *Этап рекурсивного разбиения множества S из N точек на равной мощности подмножества S_1 и S_2 плоскости, поиск медианы l и передачу подмножеств S_1 и S_2 с помощью N процессоров можно выполнить за время $O(log_2N)$, выполнив предварительную обработку за время $O(Nlog_2N)$.*

Доказательство. Пусть заданное множество точек в виде индексированного двумерного, упорядоченного массива U = {P_{ij}, i,j = 1, N}, где i- индекс, который указывает номер точки в упорядоченном списке U_x, а j -индекс, который указывает номер точки в упорядоченном списке U_y.

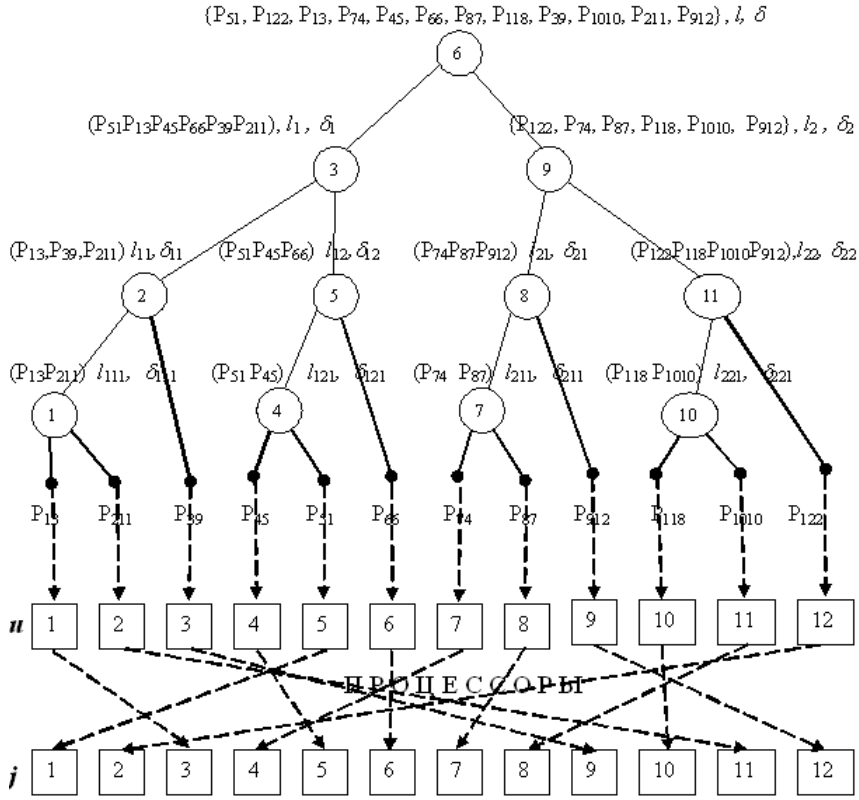

Рис.2. Дерево алгоритма для N=12.

Такое представление множества точек позволяет построить бинарное дерево разбиения, при известном количестве точек N в списке U. На рис.2 построено дерево для N=12 точек.

Как показано на рисунке, с первым индексом i каждой точки P_{ij} связан номер процессора, а со вторым индексом j - номер ячейки памяти, в которую заносится точка согласно упорядоченности по y координате. На каждом шаге разбиения соответствующие процессоры синхронно сравнивают первые индексы со списка точек и рассылают точки в соответствующие узлы алгоритма, сохраняя при этом порядок расположения точек, который определяется их порядком в ячейках памяти. Учитывая четкую упорядоченность по обоим индексам точек P_{ij} массива U и взаимосвязь между процессорами и элементами памяти, время выполнения процесса слияния в каждом узле дерева не будет превышать константу $O(1)$.

Таким образом, общее время разбиения будет не больше, чем $O(\log_2 N)$ для наихудшего ввода данных. Что и надо было доказать.

Пусть n количество элементов списка в узлах алгоритма, которое определяется делением количества элементов родительского узла пополам; r - уровень дерева алгоритма, который отвечает шагу рекурсии; k - номер узла, который определяется соотношением (1), а i, j - индексы элементов массива $U=\{P_{ij}, i, j = 1, N\}$. Процессы, которые реализуют алгоритм на этапе разбиения, можно описать таким образом.

Пока n > 1, всем процессам:

1. Присвоить номер k каждому узлу дерева алгоритма, согласно (1).
2. На r-ом шаге алгоритма в узле k определить медиану l множества S (U) по формуле $l = (P_{kj}+P_{k+1j})/2$, где k - номер узла;
3. На r-ом шаге алгоритма в узле k необходимо сравнить значение i-го индекса P_{ij} элементов массива U с числом k.

а) если $i \le k$ – послать каждый элемент $P_{ij} \in$ U, индекс которого удовлетворяет этому условию, в узел " ЛЕВЫЙ СЫН" по *i-ому* каналу и записать в *j*- тую ячейку памяти узла.

б) если $i > k$ – послать каждый элемент $P_{ij} \in$ U, индекс которого удовлетворяет этому условию, в узел " ПРАВЫЙ СЫН" по *i-ому* каналу и записать в *j*- тую ячейку памяти узла.

4. $n = 1$. Завершение работы этого этапа алгоритма.

Таким образом, в результате работы алгоритма, на этом этапе будут сформированы упорядоченные по y координате подмножества точек равной мощности U_1, U_2 относительно медианы l. При этом: U_1 расположено слева, а U_2 -справа, относительно l.

2.4 Слияние результатов (рекурсивный подъем)

На каждом шаге (кроме первого) рекурсивного подъема, решается, собственно говоря, главная задача алгоритма - определение ближайшей пары. Процесс определения ближайшей пары состоит из таких шагов-процедур:

1. В родительском узле определяется наименьшее расстояние среди присланных от обоих сынов (δ_1, δ_2, соответственно): $\delta = min(\delta_1, \delta_2)$.
2. Определяются точки из соседних множеств S_1, S_2 относительно медианы l (вертикали), которые попали в $\delta(x)$- окрестность медианы, то есть у полосы $H_1=[l-\delta]$ и $H_2 = [l+\delta]$.
3. Пусть U_1, U_2 – списки точек, которые попали у полосы H_1, H_2 , соответственно. Для каждой точки P_i с полосы H_1 проверяется расстояние к точкам полосы H_2 , которые попадают в область близости точки P_i, то есть, прямоугольник $[y_i \pm\delta]\times\delta$ полосы H_2 . Максимальное количество точек, расстояние между которыми будет не меньше δ, и которые могут попасть в такой прямоугольник, не превышает шести [1].
4. Найденные на шаге 3 расстояния δ_i сравниваются с δ, и выбирается наименьшее $\delta' = min(\delta_i, \delta)$. Это расстояние передается на следующий уровень рекурсии.

Учитывая упорядоченность по координате y списков в узлах дерева алгоритма и то, что в наихудшем случае обе полосы H_1, H_2 могут содержать $O(N)$ точек, время слияния на каждом шаге рекурсивного подъема не будет превышать $O(N)$. Общее время рекурсивного подъема при последовательной реализации будет не больше $O(Nlog_2N)$.

Время, которое расходуется на рекурсивный подъем в случае параллельного алгоритма, определяется следующей леммой.

Лемма 2. *Этап рекурсивного слияния результатов определения ближайшей пары на множестве S из N точек плоскости и передачи наименьших расстояний на следующий уровень рекурсивного подъема с помощью N процессоров можно выполнить за время $O(log_2N)$, выполнив предварительную обработку за время $O(Nlog_2N)$.*

Доказательство. Учитывая то, что каждый процессор имеет доступ к упорядоченному по y массиву точек U и то, что в окрестность точки с полосы H_1 может попасть не большее шести точек из полосы H_2, вычисление наименьшего расстояния на $N/2$ процессорах между точками с полос H_1, H_2 будет выполняться за константное время $O(1)$. Нахождение точек, которые попали в δ - окрестность i - ой точки полосы H_1 из H_2 можно выполнить с помощью шести сравнений. Для этого необходимо взять соседние шесть точек i - ой точки полосы H_1 по y координате с полосы H_2 в основном массиве точек U и проверить, попадание их $2\delta\times\delta$ прямоугольник полосы H_2 . Процессы, которые реализуют алгоритм на *этапе слияния,* можно описать таким образом.

Всем процессам:

1. Передать элементы массива, расположенные в листьях дерева алгоритма.
2. Для узлов алгоритма, отличных от $r-2$, определить расстояния δ между парами элементов массива.

Пока $n \le N$ ($r < r - 2$)

3. Передать значения δ родительским узлам.
4. Для узлов r –го уровня алгоритма:

а) отложить слева и справа от медианы l каждого узла вертикальные полосы шириной δ;

б) определить множества точек, которые попадают в левую (H_1) и правую (H_2) полосы множеств U_1, U_2, соответственно; (если в узле множество точек представлено в виде списка, то поиск осуществляется по спискам H_1, H_2, упорядоченным по y, а если в узле множество точек представлено в виде бинарного дерева, то осуществляется бинарный поиск);

в) для каждой точки P_{sq} из H_1 ($1< s,q < N$) найти точки с H_2 , которые попадают в прямоугольник $[y_q \pm \delta]\times\delta$;

г) для каждой точки P_{sq} из H_1 ($1< s,q < N$) определить расстояния δ_m к m точкам из H_2, которые попали в прямоугольник $[y_q \pm \delta]\times\delta$;

д) найденные расстояния сравнить с δ, то есть найти $\delta = min(\delta, \delta_m)$, $\forall m$;

е) передать δ от сынов родительским узлам на следующий уровень алгоритма.

5. Процесс заканчивается, после завершения шага 4 для $n = N$.

Общее время рекурсивного алгоритма определения ближайшей пары для множества S из N точек определяется временем рекурсивного разбиения и рекурсивного слияния. А поэтому имеет место теорема.

Теорема 1. *Задачу о нахождении ближайшей пары на множестве S из N точек плоскости можно решить с помощью N процессоров со временем $O(\log_2 N)$, при использовании $O(N\log_2 N)$ памяти за два рекурсивных прохода, выполнив предварительно обработку входных данных за время $O(N\log_2 N)$.*

Доказательство. Задача о ближайшей паре решается рекурсивным методом ” распределяй и властвуй”. В основу этого метода положен рекурсивно-параллельный алгоритм, основными шагами которого есть рекурсивное разбиение заданного множества на подмножества равной мощности и шага слияния результатов определения ближайшей пары, которые выполняются согласно леммам 1,2 за время $O(\log_2 N)$.

Более эффективный алгоритм решения поставленной задачи, с использованием значительно меньшего количества процессоров и меньшего объема памяти, можно построить, если в узлах графа алгоритма (рис.2), для обработки на этапе разбиения и на этапе слияния, использовать вместо упорядоченных списков по в координате U = $\{P_{ij}, \ i, j = 1, N\}$, построенные на их основе бинарные деревья. В этом случае все операции, даже в случае последовательной работы, выполняются не большее, чем за $O(\log_2 N)$ времени. Поэтому имеет место такая теорема.

Теорема 2. *Задачу об определении ближайшей пары на множестве S из N точек плоскости можно решить с помощью N/2 процессора со временем $O(\log_2 N)$, при использовании O(N) памяти за два рекурсивных прохода, выполнив предварительно обработку входных данных за время $O(N\log_2 N)$.*

Все основные процедуры на шаге разбиения и на шаге слияния, в этом случае, будут выполняться аналогично случаю со списками, с той лишь разницей, что поисковые процедуры будут выполняться на дереве упорядоченных точек по y координате. Так, например, поиск ближайшей пары на этапе слияния в каждом узле алгоритма будет выполняться так: начиная с корня дерева, выбираем первую точку с U_1, которая попала в полосу H_1, и для нее ищем соседние на дереве точки с U_2, которые попадают в ее δ-окрестность (прямоугольник $2\delta \times \delta$) полосы H_2. Таких точек будет не большее шести. Учитывая то, что все операции по дереву выполняются не более, чем за $O(\log_2 N)$ времени на этапе слияния, получим общее время решения задачи $O(\log_2 N)$.

В случае параллельной обработки, синхронно может работать меньшее количество процессоров, чем для узловых списков, в одних и тех же узлах алгоритма, а загруженность процессоров будет более высокая. Вопрос использования оптимального состава процессоров (компьютеров) в данной работе не рассматривался и требует дальнейших исследований. В то же время, известно [2, 3] что, если обозначить за k количество процессоров, за w- общее количество операций алгоритма, а за $O(F(N))$ – сложность алгоритма для неограниченного количества процессоров, то время выполнения алгоритма можно представить в виде:

$$O(f(N)) = ((k-1)O(F(N)) + w)/k \qquad (2)$$

В нашем случае, согласно теореме 2, $F(N) = \log_2 N$, а значит время выполнения алгоритма для k процессоров будет определяться следующим соотношением:

$$O(f(N)) = ((k-1)O(\log_2 N) + w)/k \qquad (3)$$

Так, в частности, для количества операций w порядка $O(N)$ из соотношения (3) вытекает: в случае $k = 2$ процессорам, время выполнения алгоритма будет равным $O(N/2)$, а при $k=N/2$ - $O(\log_2 N)$. Кроме этого, на наш взгляд, синхронная обработка данных алгоритма может быть заменена конвейерной, что расширяет возможности повышения общей эффективности алгоритма.

3. Реализация алгоритма

Для практической реализации разработанного параллельного алгоритма одним из эффективных подходов может быть подход на основе использования технологии ПАРУС (Параллельные Асинхронные Рекурсивно Управляемые Системы) [4], которые в ряде случаев дают возможность реализовать эффективные параллельные алгоритмы решения сложных задач. Об этом свидетельствуют известные результаты при решении задач вычислительной математики [4] и оптимизационных задач в экономике [5]. А использование системы ПАРУС-JAVA [6] дает возможность распараллеливать обычные рекурсивные алгоритмы на компьютерах, соединенных через локальную сеть, или сеть Internet, ускоряя, тем самым, процесс получения решения.

Система ПАРУС-JAVA реализуется на основе локальной или глобальной компьютерной сети обработки информации, предоставляя возможность пользователям маломощных компьютеров использовать ресурсы мультипроцессорных комплексов, или ресурсы компьютерной сети. Основные элементы системы ПАРУС и средства параллельного программирования для реализации ПАРУС - технологии, детально описанные в роботах [4-7].

4. Заключение

В данной работе разработана стратегия построения параллельного алгоритма для задачи о ближайшей паре на основе схемы “распределяй и властвуй” при использовании технологии ПАРУС.

Учитывая то, что большинство задач вычислительной геометрии связанны с обработкой множества точек, предложенная стратегия может быть общей методологией решения широкого класса задач вычислительной геометрии.

Этап разбиения, предложенный в этой работе, может быть общим не только для задач вычислительной геометрии, исходными данными которых есть точки, а также может быть обобщенным и для решения задач других классов [8], не связанных с вычислительной геометрией. Этого можно достичь, например, используя отношения сводимости задач.

Основные шаги этапа слияния тоже могут быть общими для других классов задач вычислительной геометрии. Будут изменяться лишь специфические процедуры слияния, которые являются сутью поставленной задачи.

Литература

1. Препарата Ф., Шеймос М. Вычислительная геометрия: Введение. Г.: Мир, 1989. – 478 с.
2. В. В. Воеводин. Матричные методы и алгоритмы. Сб. научных трудов/ РАН Инс. Выч. Мат., под ред. Воеводина, 1993, 171 с.

3. В. В. Воеводин, Вл. В. Воеводин. Параллельные вычисления, СПб., БХВ- Петербург, 2004, 608 с.

4. Анисимов А.В., Глушков В.М. Управляющие пространства в асинхронных параллельных вычислениях. // Кибернетика, №5, 1980. - С.1-9.

5. Анисимов А.В., Борейша Ю.Э., Карапетян М.С. . Параллельное программирование экономических систем на основе декомпозиционных методов. КИБЕРНЕТИКА, Киев, Научная мысль, 1990, №3, с. 105-110.

6. Анисимов А.В., Деревянченко А.В Построение виртуального параллельного пространства с использованием технологии ПАРУС-JAVA.// Материалы Международной научно-технической конференции ИМС ' 2003 ,Т.2. - С.18-19.

7. Анісімов А.В., Кулябко П.П., Терещенко В.М. Паралельні алгоритми в дослідженнях неперервних систем. К.: Видавництво “ЛОГОС”, 1999.- 50с.