В.А. Зубарев, В.Н. Терещенко

Киевский национальный университет им. Т.Г. Шевченко

ОБОБЩЕННЫЙ ПАРАЛЛЕЛЬНО-РЕКУРСИВНЫЙ АЛГОРИТМ РЕШЕНИЯ НАБОРА ЗАДАЧ ГЕОМЕТРИЧЕСКОГО D-МОДЕЛИРОВАНИЯ

Работа посвящена созданию эффективных подходов решения задач геометрического моделирования в пространствах E^d. Предлагается новая парадигма создания алгоритмической среды для решения комплекса взаимосвязанных геометрических задач – обобщенный параллельно-рекурсивный алгоритм, использующий набор эффективных параллельных процедур решения. За основу обобщенного алгоритма взята стратегия «разделяй и властвуй», что позволяет эффективно использовать параллельные вычисления на каждом шаге алгоритма. Получено эффективное решение как отдельных задач, так и их совокупности. Установлены оценки сложности разработанных процедур и осуществлена их практическая реализация, подтвердившая их эффективность.

Графическое моделирование является одним из эффективных современных способов исследования и анализа сложных динамических процессов и явлений в науке и технике. В частности, это актуально для моделирования физических и механических процессов, которые происходят при высоких температурах нагревания твердого и жидкого тела, графического рендеринга, моделирования критических процессов реального времени и других сложных динамических процессов и явлений. Современный высокотехнологический уровень науки и техники ставит новые проблемы и определяет направления исследований относительно разработки новых и модификации существующих эффективных методов и алгоритмов. В основе этого лежат новые парадигмы и направления разработки, такие как параллельные и распределенные вычисления, а также архитектурные особенности новейших вычислительных систем.

Предметом исследования являются проблемы, задачи и алгоритмы геометрического моделирования в пространствах большой размерности.

Объектом исследования является *применение параллельных вычислений в задачах геометрического моделирования и решения набора взаимосвязанных задач вычислительной геометрии на одном и том же множестве входных данных*.

Целью исследования является поиск подходов, которые позволяют разработать простую и в то же время эффективную алгоритмическую среду для одновременного решения целого комплекса взаимосвязанных задач вычислительной геометрии в евклидовых *d*-мерных пространствах. Для решения этой проблемы необходимо разработать обобщенный параллельно-рекурсивный алгоритм, использующий общие алгоритмические инструменты и структуры данных, а также соответствующий набор процедур реализации.

К задачам геометрического моделирования можно отнести исследование целевых объектов познания по их модели, построение и изучение моделей реально существующих предметов, процессов или явлений, а также создание моделей новых и еще не изученных явлений. Геометрическая модель – определенный объект, природа которого отображает главные с точки зрения поставленной задачи пространственно-геометрические свойства его моделирования. К таким свойствам можно отнести расположение объекта в пространстве, его форма, внутренняя структура, и др. При построении модели для описания всего комплекса свойств необходимо решать одновременно целую совокупность взаимосвязанных геометрических задач.

Достаточно полно изучен двумерный случай. Но более сложными и важными на практике являются задачи для больших измерений. И здесь существует много открытых и нерешенных проблем, одной из которых является построение единого общего решения для целого комплекса взаимосвязанных задач. В большинстве случаев, даже при том, что каждая отдельная задача имеет эффективный алгоритм, в совокупности

это не всегда дает желаемую эффективную реализацию. В основном это связано с несоответствием структур данных и отсутствием связи между входными данными и результатами вычислений задач, что затрудняет возможность взаимодействия между процедурами алгоритмов и не дает минимального времени решения такого класса задач. Поэтому необходима разработка универсального инструмента, который имел бы общие средства для эффективного решения всего комплекса взаимосвязанных геометрических и прикладных задач. Другими словами, мы должны выбрать стратегию, которая использует общие средства реализации: структуры данных, отдельные этапы алгоритмов, процедуры и некоторые их шаги, а также способы представления результатов. И одной из таких стратегий может быть использование параллельных алгоритмов.

На сегодняшний день разработано много эффективных параллельных алгоритмов решения отдельных задач вычислительной геометрии. Так, в работе [1] описаны эффективные алгоритмы с использованием схемы «разделяй и властвуй», решения задач построения выпуклой оболочки для 2- и 3-мерных пространств с оценкой сложности $O(logN)$ и $O(log^3N)$, соответственно, а также построения диаграммы Вороного ($O(log^2N)$). В этой же работе представлен достаточно глубокий анализ других эффективных алгоритмов решения задач пересечение отрезков, триангуляции полигонов, оптимизации ($O(logN)$). В работах М.Т. Гудриха, М.Дж. Атталаха, Р. Коле, Н.М. Амато, Д.З. Чена и других авторов [2–7] получены улучшенные результаты решения вышеуказанных задач для различных моделей вычислений (CREW, EREW) и для высших размерностей ($d \geq 4$). Ряд статей посвящен общим параллельным методам, которые применимы к решению отдельных задач вычислительной геометрии [8–11]. Но после анализа непосредственной реализации многих из этих алгоритмов были выявлены некоторые проблемы, которые не были авторами учтены. В частности, проблемы возникают при анализе границ выпуклых оболочек, когда звенья границ имеют количество точек большее, чем размерность целевого пространства. В то же время идеи, предложеные в работах [1, 4–6, 12, 13],

дают нам представление о структуре необходимой геометрической модели в многомерных пространствах и пути развития необходимых стратегий построения моделей. Особенно следует отметить роботу М. Шеймоса и Д. Хоея, которые впервые в своей работе [14] использовали стратегию «разделяй и властвуй» при построении выпуклой оболочки для случая трехмерного пространства. Они получили наилучшую оценку *O(NlogN)* для однопроцессорной машины. Также необходимо упомянуть работу Т. Чена [15], в которой он элегантно реализовал их идею.

Но при решении целого комплекса задач на одном и том же множестве данных использование отдельных эффективных алгоритмов в общем случае не приносит желаемой эффективности. Дело в том, что каждый алгоритм нуждается в собственной предварительной обработке и создании структуры данных, а также процедур реализации, что не дает минимального времени решения такого класса задач. Поэтому необходимо было выбрать такую стратегию, которая бы позволяла получить самую эффективную реализацию. Учитывая отмеченные особенности и то, что большинство задач вычислительной геометрии имеет внутренний параллелизм и предусматривает рекурсивную природу реализации, наиболее подходящей стратегией, на наш взгляд, может быть та, в основе которой лежит параллельно-рекурсивный алгоритм, использующий технику «разделяй и властвуй». Здесь этапы предварительной обработки и разбиения множества будут общими для всего комплекса задач, а на этапе слияния предлагается использовать общую для всех задач структуру данных – *взвешенную сцепляемую очередь*. Кроме того, результаты отдельных этапов некоторых процедур будут использоваться другими процедурами, что обеспечивает высокую эффективность.

Параллельно-рекурсивный алгоритм. Поскольку нашей целью является создание алгоритмической среды для реализации комплекса задач, в качестве общей стратегии построения решения мы взяли схему «разделяй и властвуй». Эта стратегия позволяет нам построить эффективный обобщенный параллель-

но-рекурсивный алгоритм на уровне управления и использования общей структуры данных, что открывает возможность эффективной разработки не только на машинах с общей памятью, но и с распределенной. Кроме того, выделение и решение задач на подзадачах позволит при необходимости эффективно обмениваться уже полученными результатами между отдельными задачами, что уменьшает алгоритмическую нагрузку на них.

Общая схема. Весь процесс вычисления можно изобразить следующим образом: *предобработка, разбиение множества точек на равномощные подмножества (рекурсивный спуск), решение комплекса задач на подмножествах, начиная с элементарных, и слияние результатов при возвращении к исходному множеству.*

Поскольку предварительная обработка и шаг разбиения общие для всего комплекса задач, мы опишем лишь шаг слияния на примере задачи построения выпуклой оболочки. Процедуры слияния для задач построения триангуляции и диаграммы Вороного аналогичные и отличаются лишь деталями.

Алгоритм построения процедуры слияния выпуклых оболочек

Алгоритм слияния можно схематически разделить на 3 этапа: построение опорных граней моста, удаление лишних вершин и граней, окончательное построение граней моста.

Этап 1. *Построение опорных граней моста*, которые соединяют две оболочки.

1.Выбираем внутреннюю точку p. Пусть это будет точка, которая лежит в серединной плоскости, разделяющей взаимно выпуклые части двух выпуклых оболочек.

2. Находим первую опорную грань. Берем первую точку левой выпуклой оболочки и первую точку правой. На этих точках и векторах b_2, b_{n-1} базиса B пространства E^d строим плоскость. В качестве положительного направления вектора нормали выбираем тот, полуплоскость которого содержит точку p. Перебираем соседей точки первого выпуклого множества. Если новая точка вместе с правой образует плоскость, относительно которой предыдущая

точка вместе с точкой p будет лежать в одной полуплоскости, вместо первой точки берем вновь найденную. Повторяем этот шаг до тех пор, пока можно найти такую точку.

3. Аналогично для следующей грани.

4. Повторяем предыдущие два шага до тех пор, пока не сможем изменить ни одну из точек. Таким образом, мы имеем уже 2 точки, принадлежащие определенной плоскости на выпуклой оболочке. Дальше, нам необходимо найти другие d–2 точки. Для этого исключаем из базиса вспомогательной плоскости по одному вектору и перебираем соседей текущих точек. Если текущая точка лежит по разные стороны от текущего кандидата и центральной точки, переписываем текущего кандидата на новую точку. Первого кандидата выбираем автоматически в качестве первого соседа. В результате получаем грань выпуклой оболочки.

5. Для найденной грани перебираем все ее ребра и для каждого ребра, концы которого принадлежат разным оболочкам, из соседних точек ребра находим такую, относительно которой все другие точки лежат в одной полуплоскости с p:

а) если найденной грани нет в списке уже прибавленных граней к выпуклой оболочке, добавляем ее и выполняем пункт 5;

б) если найдено определенное подмножество точек, которые являются кандидатами на грань на плоскости, но для этого подмножества еще не выполняется пункт 5, то триангулируем это множество по условию принадлежности каждой грани триангуляции опорной грани (дальше «мост»). Каждую полученную грань добавляем к «мосту» и выполняем пункт 5 для нее. Берется начальное ребро из грани, которую проверяем, формируется плоскость с этой гранью и вектором нормали к плоскости, в которой это все строится. Среди точек текущего множества, находящихся по разные стороны относительно плоскости с точками, уже включенными в мост, находим точку с минимальным углом к этой плоскости. При этом рассматриваются только те точки, которые находятся слева и справа, а также смежные с текущими точками;

в) иначе поиск прекращаем. Когда поиск прекращен для всех граней, переходим к следующему пункту.

6. Найденное множество и является мостом между оболочками.

Этап 2. *Удаление вершин с гранями*, которые принадлежат «внутренней части» оболочек подмножеств и не принадлежат результирующей оболочке.

1. Возвращаемся к центральной точке p_1, которую использовали для объединения дочерних множеств с первой. Сначала перебираем точки моста на левой оболочке, а затем – все грани, которым принадлежит эта точка. Для каждой грани проверяем, лежит ли точка p_1 в одной полуплоскости с точками максимума и минимума по каждой координате второй оболочки. Если нет, и все точки грани принадлежат мосту, то просто удаляем эту грань. Если же какая-то точка не принадлежит мосту, то запускаем шаг 2 (такой вызов будет единственен). После проверки всех ребер, переходим к шагу 3.

2. Отмечаем точку как удаленную из оболочки и рекурсивно запускаем этот шаг для всех соседей, которые не принадлежат мосту. На шаге рекурсивного подъема удаляем все ребра и грани, которые выходят из этой точки.

3. Выполняем шаги 1–3 для другой оболочки.

Этап 3. Грани моста добавляем к выпуклым оболочкам и получаем выпуклую оболочку, общую для всего множества.

Теорема 1. *Выпуклую оболочку на множестве S из N точек в евклидовом d-мерном пространстве* E^d $(d > 2)$ *можно построить параллельно-рекурсивным алгоритмом с использованием q процессоров* $(q \le N)$ *за время* $O(\log(N)/(\max(2,\log(q)+1)d(K/2+dK/\max(1,q-\log(q)))))$, *где K – количество вершин выпуклой оболочки), с ускорением процедуры слияния* $R = \log(N) / \max(2,\log(q)+1)$.

Доказательство теоремы выводится из анализа сложности основных этапов алгоритма. Для двумерного случая эта оценка будет $O(\log N/\min(2,\log(q)+1)K)$, поскольку количество точек моста всегда 2 на каждом из подмножеств.

Особенности построения процедур слияния для триангуляции и диаграммы Вороного

Задача триангуляции на ограниченном множестве точек достаточно изучена и для ее решения существует ряд эффектив-

ных последовательных алгоритмов. Здесь следует отметить работы Скворцова, и в частности роботу [16], в которой предложено несколько быстрых алгоритмов триангуляции. В данной работе предлагается процедура триангуляции для рассматриваемого параллельного алгоритма в случае асимптотически больших множеств точек (или множеств точек, представляющих структурированные объекты).

Легко доказать, что триангуляция в евклидовом d-мерном пространстве – это то же самое, что выпуклая оболочка в пространстве $d + 1$. Для этого нужно расширить пространство размерности d до d+1, а точки заданного множества S необходимо поместить на d+1-мерный параболоид, где d+1 координата каждой точки определяется простым соотношением $x_{d+1} = x_1^2 + . + x_d^2$. Далее, необходимо построить выпуклую оболочку для полученных точек, и спроектировать обратно на d-мерное пространство. Полученная нижняя часть оболочки будет искомой триангуляцией.

Теорема 2. *Триангуляцию на множестве S из N точек в евклидовом d-мерном пространстве* E^d $(d > 2)$ *можно построить параллельно-рекурсивным алгоритмом с использованием q процессоров* $(q \le N)$ *за время* $O(\log(N)/\max(2\ \log(q)+1)\cdot d\ K/(\max(k,1q-\log(q))),$ *(K – количество вершин выпуклой оболочки), с ускорением процедуры слияния* $R = \log(N) / \max(2,\log(q)+1)$.

Предложенный метод позволяет значительно улучшить суммарную сложность, заменив в оценке для классического решения N на K. Кроме того, при кажущейся одинаковой эффективности мы выигрываем на значительно меньшем значении коэффициента комплексности оценки в последнем случае (то есть $O(N) = k_2 N$, k_2 = const. $k_2 << k_1$, где κ_2 – коэффициент триангуляции, k_1 – коэффициент построения оболочки). Этот факт очевиден, достаточно сравнить соответствующие шаги выполнения в обоих случаях. Еще одно преимущество такой реализации в том, что не нужно создавать новый массив точек для триангуляции, все можно реализовать на базе основных, на которых строим выпуклую оболочку.

Как и в случае с триангуляцией, для построения диаграммы Вороного используются на основных этапах построения

процедуры слияния результаты предыдущих задач: триангуляции и выпуклой оболочки. Нас здесь больше интересует решение, вытекающее из построения триангуляции. Здесь не нужна перестройка в полном объеме. Достаточно получить триангуляцию в пространстве той же самой размерности и на ее основе построить диаграмму Вороного. В таблице приведен общий анализ оценок сложности для рассмотренных задач.

Общий анализ эффективности алгоритма

Параллельность / Алгоритм	Один процессор	Q процессоров
Предобработка	$O(NlogN)$	$O(logN/max(log(Q)+1,2)N)$
Выпуклая оболочка	$O(Nd^2logK)$	$O(log(N)/(max(2,log(Q)+1)d(K/2+ + dK/max(K,1,Q-log(Q))))$
Триангуляция	$O(logN \cdot d \cdot K)$	$O(log(N)/max(2, log(Q)+1) \cdot d \cdot K/(max(k,1,Q-log(Q)))$
Диаграмма Вороного	$O(N \cdot d^2)$	$O(N \cdot d2/max(q,N))$
Суммарная	$O(N(d^2logK+logN)$	O (предобработка + построение оболочки)

Из сравнительного анализа оценок следует, что для всех задач достигается высокая эффективность, за исключением предобработки. Важно отметить, что предложенный алгоритм в общем способен эффективно использовать количество процессоров даже большее, чем N. Это достигается благодаря, во-первых, использованию процедур слияния, построенных единым способом, что позволяет выполнять ряд расчетов независимо и, во-вторых, наличие нескольких задач позволяет получить дополнительное распараллеливание на межзадачном уровне (пока решается определенная стадия одной из текущих задач, другие могут выполнять следующие стадии).

Реализация алгоритма. Для распараллеливания алгоритма реализована система управления процессами, которая совместима как с системами, которые работают с общей памятью, так и с распределенной. Более сложным по структуре и реализации является второй случай. Там особого выбора нет, система использует MPI на низком уровне. Для первого случая было решено разработать свой механизм управления, но так, чтобы интерфейсы управления в обоих случаях были похожими для обеспечения максимальной простоты перехода от одной парадигмы к другой. Необходимо также отметить, что эффективность использования функциональности реализации лежит на разработчике в рамках используемой им парадигмы. Кроме этого, для «последовательной части» функциональности имеется возможность создавать виртуальные процессоры, которые будут псевдопараллельно нагружать определенный процессор.

На этапе предварительной обработки нужно выполнить начальную обработку входных данных. В нашем случае это лишь сортировка точек. В работах [17–19] предложены эффективные параллельные алгоритмы сортировки. Для реализации разработанного алгоритма мы выбрали сортировку слиянием. Этот алгоритм простой и показывает хорошие результаты на больших объемах данных. Он имеет меньшее ускорение на большом количестве процессоров, но не требователен к архитектуре памяти. Был проведен анализ производительности разработанного алгоритма в сравнении с существующими последовательными реализациями. Вычислительная система реализации алгоритма исследовалась для множества точек от 10^5 по 10^6 с шагом 10^5. На рис. 1 показаны графики временной зависимости работы для последовательного алгоритма (ПА), параллельно-рекурсивного алгоритма (П-РА) и предварительной обработки (ПО). На рис. 2 представлены результаты построения выпуклой оболочки и триангуляции с помощью предложенного алгоритма.

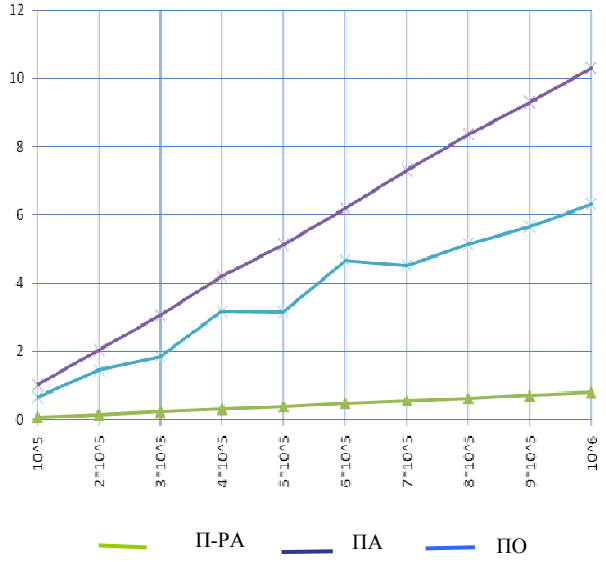

Рис. 1. График временной зависимости от количества точек

Рис. 2. Выпуклая оболочка и триангуляция для точек, равномерно распределенных в кубе

Таким образом, был разработан параллельно-рекурсивный алгоритм построения комплекса взаимосвязанных задач вычислительной геометрии (выпуклой оболочки, триангуляции, диаграммы Вороного и других задач) для пространств произвольной размерности. Получена улучшенная эффективность решения как

отдельных задач, так и их совокупности. Установлена эффективность алгоритма и предложена его практическая реализация для построения выпуклой оболочки и триангуляции в 3-мерном пространстве. Характерной чертой практической реализации предложенного подхода является то, что разработанный алгоритм позволяет одновременно решать как разные шаги одной процедуры задачи на многих процессорах, так и разные процедуры в одном узле алгоритма с помощью технологии MPI. Еще одна особенность этой модели алгоритма заключается в том, что предобработка и процесс разбиения множества точек являются общими для решения рассматриваемого множества задач; отличаться будут лишь процедуры на этапе слияния.

Список литературы

1. Parallel computational geometry / A. Aggarwal [et al.] // Algorithmica. – 1988. – No. 3. – P. 293–327.

2. Atallah M.J., Cole R., Goodrich M.T. Cascading divide-and-conquer: A technique for designing parallel algorithms // SIAM J. Comput. – 1989. – No. 18. – P. 499–532.

3. Cole R., Goodrich M.T. Optimal parallel algorithms for polygon and point-set problems // Algorithmica. – 1992. – No. 7. – P. 3–23.

4. Amato N.M., Goodrich M.T., Ramos E.A. Parallel algorithms for higher-dimensional convex hulls // In Proc. 35th Annu. IEEE Sympos. Found. Comput. Sci. – 1994. – P. 683–694.

5. Chen D. Efficient geometric algorithms on the EREW PRAM // IEEE Trans. Parallel Distrib. Syst. – 1995. – No. 6. – P. 41–47.

6. Berkman O., Schieber B., Vishkin U. A fast parallel algorithm for finding the convex hull of a sorted point set // Internat. J. Comput. Geom. Appl. – 1996. – No. 6. – P. 231–242.

7. Goodman J.E., O'Rourke J. Handbook of Discrete and Computational Geometry. – N.Y.: Chapman and Hall/CRC Press, 2004. – 1497 p.

8. Akl S.G., Lyons K.A. Parallel Computational Geometry. – Englewood Cliffs: Prentice-Hall, 1993. – 224 p.

9. JaJa J. An Introduction to Parallel Algorithms. – Amsterdam: Addison-Wesley, 1992. – 566 p.

10. Van Leeuwen J. Handbook of Theoretical Computer Science. – Amsterdam: Elsevier/The MIT Press, 1990. – 1294 p.

11. Reif J.H. Synthesis of Parallel Algorithms. – San Mateo: Morgan Kaufmann, 1993. – 1011 p.

12. Ghouse M. and Goodrich M. Fast randomized parallel methods for planar convex hull construction. Comput. Geom. Theory Appl., 7: 219{236, 1997.

13. Шеймос М., Препарата Ф. Вычислительная геометрия: Введение. – М.: Мир,1989. – 478 с.

14. Shamos M.I., Hoey D. Closest-point problems. In Proc. 16th IEEE Sympos. Found. Comput. Sci pages 151–162, 1975.

15. Chen T.M. A minimalist's Implementation 3-d Divide-and-Conquer Convex Hull Algorithm. School Computer Science University Waterloo. – 2003. – 12 с.

16. Скворцов А.В. Триангуляция Делоне и ее применение. – Томск: Изд-во Томск. ун-та, 2002. – 128 с.

17. Cole R. and Goodrich M.T. Optimal parallel algorithms for polygon and point-set problems. Algorithmica, 7:3–23, 1992.

18. Reif J.H. and Sen S. Optimal parallel randomized algorithms for three-dimensional convex hulls and related problems. SIAM J. Comput., 21:466–485, 1992.

19. Akl S.G., Lyons K.A. Parallel Computational Geometry. Prentice-Hall, Englewood Cliffs, 1993.

Г.Г. Исламов, А. Г. Исламов

Удмуртский государственный университет, г. Ижевск

ОБ ОДНОМ КЛАССЕ ОРТОГОНАЛЬНЫХ ПРОЕКТОРОВ

Пусть $B = H_n$ и H_μ – конечномерные вещественные гильбертовы пространства размерности соответственно ν и μ, причём $\nu << \mu$. Пусть, далее, $\Lambda: B \to H_\mu$ – линейное инъективное отображение и $\Lambda^*: H_\mu \to B$ – сопряжённый с ним оператор. Следующее утверждение очевидно.