УДК 004.93'1

АППРОКСИМАЦИЯ ДИАГРАММЫ ВОРОНОГО *K*-ГО ПОРЯДКА

С. А. Хальзов, В. В. Фертиков

Воронежский государственный университет

Поступила в редакцию 13.04.2019 г.

Аннотация. Представлен алгоритм построения обобщенной дискретной диаграммы Вороного *k*-го порядка на сетке с переменным шагом, основанный на идее рекурсивного разбиения пространства. В алгоритме в качестве структуры данных используется дерево разбиения пространства (2^d-дерево), также известное как квадродерево и октодерево для 2-х и 3-х мерных пространств. Алгоритм рекурсивно уточняет границы ячеек Вороного (бисекторы), используя только локальные свойства в каждой точке области построения. В результате вычисления сосредотачиваются в области границ ячеек (глубина 2^d-дерева больше в области границ ячеек), и достигается существенный выигрыш производительности по сравнению с наивным алгоритмом (полным перебором). Алгоритм позволяет обобщать форму сайтов, используемую метрику, порядок диаграммы и число измерений.
Ключевые слова: аппроксимация диаграммы Вороного, диаграмма Вороного *k*-го порядка, диаграмма минимизации, квадродерево, октодерево, предикат уточнения.

ВВЕДЕНИЕ

Диаграммы Вороного являются фундаментальными структурами данных, которые хорошо изучены в вычислительной геометрии. Для данного множества сайтов, соответствующая диаграмма Вороного разбивает d-мерное пространство на ячейки, состоящие из точек, расположенных ближе всего к одному и тому же сайту. Под сайтом подразумевается некоторый объект (точка, отрезок, окружность, и т. п.), до которого можно вычислить расстояние. Хотя диаграммы Вороного в большинстве случаев заданы в метрическом пространстве, их можно определить более абстрактно. В данной работе диаграммы Вороного будут рассматриваться как диаграммы минимизации любого конечного множества функций. Обычно каждая из этих функций интерпретируется как функция расстояния до объекта.

Можно определить много вариантов диаграмм Вороного в зависимости от класса объектов, функции расстояния и пространства разбиения. Диаграммы Вороного также могут быть классифицированы в соответствии с природой бисекторов, например, как аффинные (affine) и не аффинные (non-affine, curved) [1]. Для конечного множества объектов и соответствующих функций расстояния бисектором является место точек, равноудаленных от двух объектов.

Диаграмма Вороного имеет множество приложений, например, в распознавании образов (получение срединных осей или «скелетов»), робототехнике (планирование маршрута при наличии препятствий), компьютерной графике, кристаллографии, метеорологии и во многих других областях.

АНАЛИЗ ПРЕДШЕСТВУЮЩИХ РАБОТ

Диаграммы Вороного для точек и отрезков на плоскости хорошо изучены, и существуют эффективные алгоритмы, которые точно их вычисляют [2–5], но точные алгоритмы для обобщенных диаграмм Вороного ограничены небольшим набором особых случаев [1, 6].

Обобщенные диаграммы Вороного сложно вычислять аналитически в целом [1, 6]. Например, построение обобщенной диаграммы Вороного в 3-мерном пространстве включает манипулирование алгебраическими поверхностями высокой степени и их пересечени-

ями [7]. Аналитическое построение обобщенной диаграммы создает много проблем с точки зрения вычислительной устойчивости и процессорного времени из-за большого количества точных расчетов, которые должны быть сделаны. В результате получаются алгоритмы, которые являются ненадежными, трудными для реализации и доказательства правильности. Из-за перечисленных выше проблем в большинстве практических приложений вместо точной обобщенной диаграммы Вороного вычисляют приближение.

Существует ряд адаптивных подходов к построению аппроксимированной диаграммы Вороного [8–17], которые используют для аппроксимации неоднородные сетки и различные техники, что позволяет значительно сократить вычислительную сложность и затраты памяти.

Lavender at al. [8] предложили адаптивный подход для вычисления аппроксимированной диаграммы Вороного множества обобщенных сайтов в пространстве произвольной размерности. Сайты и сама диаграмма представлены с помощью двух 2^d-деревьев. Узлы 2^d-дерева представляют собой d-мерные кубы (d-кубы). В качестве предиката уточнения (условия дробления) узлов 2^d-дерева используется количество сайтов-кандидатов (потенциально ближайших сайтов). Список сайтов-кандидатов строится на основе сравнения минимальных и максимальных расстояний от аппроксимированных сайтов до заданного узла дерева. Непрерывность границы приближенной диаграммы не гарантировано каким-либо теоретическим исследованием. Точность локализации бисектора уменьшается по мере удаления от образующих его сайтов, бисектор принимает форму «галстука-бабочки» вместо «тонкой» кривой.

Vleugels et al. [9, 10] представляют адаптивный подход, ограниченный выпуклыми сайтами. Предложенный ими алгоритм строит 2^d-дерево и аппроксимирует диаграмму Вороного с заданной точностью. Для каждой вершины узла 2^d-дерева (для каждого угла d-куба) ищется ближайший сайт среди всего множества сайтов. В качестве предиката уточнения используется размер множества ближайших сайтов узла. Если хотя бы две вершины d-куба принадлежат разным ячейкам Вороного, то через него проходит бисектор. Связность диаграммы гарантируется принудительным дроблением узлов дерева, смежных с узлами, через которые проходят бисекторы, если их размеры различаются более чем в два раза.

Teichmann et al. [11] представляют адаптивный подход, ограниченный треугольными сайтами в 3-мерном пространстве. Алгоритм разбивает пространство на тетраэдрические ячейки. Сайты вставляются в стандартное октодерево, а полигональная аппроксимация диаграммы Вороного вычисляется по стратегии распространения волнового фронта.

В [12] приводится подход, позволяющий ускорить поиск ближайшего сайта. Подход заключается в адаптивном разбиении пространства на зоны влияния сайтов (составляются списки сайтов-кандидатов). Алгоритм строит квадродерево, в качестве предиката уточнения используется длина списка сайтов-кандидатов.

Boada et al. [13–15] представляют адаптивный подход для полигональной аппроксимации диаграмм Вороного в 2-х и 3-х мерных пространствах. Алгоритм поддерживает различные формы сайтов и функции расстояния. Расстояния вычисляются с помощью стратегии распространения волнового фронта. Использование этой стратегии ограничивает применимость алгоритма диаграммами со связанными ячейками Вороного, но значительно сокращает стоимость поиска ближайшего сайта.

В [16] представлен адаптивный подход для аппроксимации диаграммы Вороного множества многоугольников, и дан краткий обзор подобных алгоритмов. Основной идеей алгоритма является использование чисто числовых примитивов и мягких предикатов. Также в этой работе исследовано влияние фильтров (различных комбинаций условий в предикате уточнения) на процесс построения 2^d-дерева, аппроксимирующего диаграмму.

Edwards et al. [17] представляют адаптивный подход для вычисления аппроксимации диаграммы Вороного произвольного набора

2-х и 3-х мерных геометрических объектов в эвклидовой метрике. В частности, авторы акцентируют внимание на наборах данных с близко расположенными объектами. Алгоритм строит 2^d-дерево, которое хранится неявно в виде структуры смежности вершин d-кубов. Поле расстояний вычисляется адаптивно, в зависимости от расположения объектов относительно друг друга, с помощью стратегии распространения волнового фронта.

Срединная ось или скелет многогранника можно рассматривать как подмножество диаграммы Вороного для его вершин, ребер и граней (в 3-мерном случае). В [18–22] исследуются приближенные и точные практические вычисления срединной оси многогранника. Очевидным недостатком этих методов является ограничение формы сайтов и используемой метрики.

Существует ряд алгоритмов построения диаграмм Вороного, использующие преимущества быстродействующего графического оборудования [23–30]. Эти алгоритмы эффективны, но большинство из них ограничено подмножеством типов сайтов, также все они используют однородные сетки и требуют большого количества вокселей (в 3-мерном случае) для разделения близко расположенных сайтов.

Диаграмму Вороного можно представить, как множество критических точек поля расстояний (distance field) [5]. В этом случае построение диаграммы Вороного является этапом постобработки для любого метода, который вычисляет поле расстояния. Преобразования расстояния (distance transformation) вычисляют поле расстояния, но большинство из них имеют равномерную сетку [31].

Все рассмотренные алгоритмы имеют ограничения на форму сайтов и используемую метрику, а также в большинстве случаев строят диаграмму 1-го порядка. Поэтому целью данной работы является синтез алгоритма аппроксимации диаграммы Вороного k-го порядка, который сохраняет гибкость наивного алгоритма (полного перебора) и при этом является адаптивным, значительно сокращая вычислительные затраты и расход памяти.

2. МЕТОДЫ ИССЛЕДОВАНИЯ

2.1. Определения

Для начала определим термины, которые будут использоваться далее в работе.

Неупорядоченная диаграмма Вороного k-го порядка – разбиение пространства на ячейки, состоящие из точек, для которых ближайшими являются одни и те же k сайтов. При построении упорядоченной диаграммы ближайшие k сайтов упорядочиваются согласно расстоянию до каждого из них, и этот порядок учитывается при разбиении пространства на ячейки (точки принадлежат разным ячейкам, если различается порядок ближайших k сайтов). Более подробное описание обоих типов диаграмм можно найти в [6].

Диаграмма Вороного k-го ближайшего сайта – разбиение пространства на ячейки, для которых k-м ближайшим сайтом является один и тот же сайт.

Farthest-point диаграмма Вороного – разбиение пространства на ячейки, для которых самым дальним является один и тот же сайт.

Сектор n-го порядка или n-сектор – множество точек, принадлежащих n и более ячейкам, то есть n-сектор представляет собой пересечение или место соприкосновения n и более ячеек Вороного.

Бисектор и трисектор – секторы 2-го и 3-го порядков соответственно.

Коридор n-го порядка или n-коридор – минимально необходимое (при заданном шаге дискретной сетки) множество дискретных ячеек, которое содержит в себе n-сектор.

Граничный коридор – коридор 2-го порядка.

2.2. Описание предлагаемого алгоритма

Очевидный, но вычислительно сложный способ построения дискретной диаграммы Вороного – дискретизировать область построения и определить для каждой ее точки принадлежность ячейкам по результатам вычисления расстояния до сайтов. Области, в которых соседние точки принадлежат разным ячейкам, являются границами ячеек Вороного – бисекторами. Этот способ гарантирует, что все ячейки, размеры которых больше шага дискретной сетки, будут найдены.

Очевидно, что при таком подходе производится много потенциально лишних вычислений. Области, через которые проходят бисекторы, рассчитываются с такой же точностью, как и все остальные, но при этом занимают относительно небольшую долю области построения. Также для каждой точки поиск k ближайших сайтов производится среди всего множества сайтов, тогда как очевидно, что следует рассматривать только те сайты, которые попали в некоторую окрестность точки. Такие сайты в дальнейшем будут называться сайтами-кандидатами.

Учитывая перечисленные выше замечания, можно улучшить наивный алгоритм. Предлагаемый далее алгоритм рекурсивно пересчитывает области, через которые проходят бисекторы, постепенно сужая граничный коридор и уменьшая длины списков сайтов-кандидатов. Процесс уточнения продолжается до тех пор, пока не будет достигнута требуемая точность.

Вычислительная процедура сопровождается специальной структурой данных, деревом разбиения пространства (2^d-деревом). Для 2-х и 3-х мерных пространств это дерево известно, как дерево квадрантов (квадродерево, region quadtree) и дерево октантов (октодерево, region octree) соответственно. Каждый узел дерева соответствует определенному d-мерному кубическому фрагменту области построения и сохраняет список соответствующих сайтов-кандидатов, а также ссылки на смежные в пространстве фрагменты, узлы-соседи.

Предлагаемый алгоритм допускает следующее обобщение классической постановки задачи: вместо традиционных геометрических сайтов-объектов (точки, отрезки, дуги и т. п.) и метрик используются сайты-функции, то есть пара понятий сайт-метрика заменяется функцией от точки d-мерного пространства. При таком подходе значение функции можно интерпретировать как расстояние до некоторого сайта в некоторой метрике, что позволяет снять ограничение на форму сайта, продиктованное в классической постановке конечным набором геометрических фигур. Аналогично обобщается способ измерения расстояния, который в классической постановке ограничен использованием фиксированной метрики для всех сайтов.

В конкретных приложениях можно рекомендовать представление сайтов в виде множества функций (подсайтов) и при поиске ближайшего сайта выбирать наименьшее значение среди всех подсайтов одного сайта. Такой подход позволит, например, аппроксимировать сайты сложной формы множеством подсайтов простой формы (точками, отрезками и дугами), измерение расстояния до которых имеет относительно невысокую вычислительную сложность. В предлагаемом алгоритме данный подход реализован при помощи преобразования индексов подсайтов в индексы сайтов через массив, после чего удаляются дубликаты индексов сайтов. Преобразование индексов происходит при поиске k ближайших сайтов после вычисления расстояний и упорядочивания списка согласно этому расстоянию.

На рис. 1 показана общая блок-схема алгоритма, детальное описание каждого пункта которой приводится далее по тексту.

Новые узлы дерева создаются на этапе затравки при принудительном выращивании ветвей дерева и на этапе уточнения при дроблении узлов, удовлетворяющих предикату уточнения. После создания нового узла дерева, с ним необходимо произвести следующие действия:

1) найти k ближайших сайтов среди сайтов-кандидатов родительского узла;

2) создать список сайтов-кандидатов, определяемый типом строящейся диаграммы;

3) вычислить ссылки на соседей, используя ссылки на соседей родительского узла.

2.3. Построение начального приближения

В классическом варианте бисекторы диаграммы Вороного представляют связный граф, и поэтому благодаря использованию процедуры прослеживания, описанной ниже, будут найдены все бисекторы, если будет найдена хотя бы одна точка граничного коридора. Но при обобщении сайтов бисекторы уже не обязательно образуют связный граф,

Рис. 1. Блок-схема алгоритма

а, следовательно, и не гарантируется нахождение всех точек граничного коридора.

Для решения данной проблемы алгоритм предусматривает затравку, суть которой заключается в том, чтобы для каждого сайта был найден хотя бы один узел, который принадлежит ячейке данного сайта. Успех данного стартового этапа лежит на ответственности пользователя, который учитывает специфику решаемой задачи. При обобщении сайтов ячейки не обязательно представлены связными множествами, т. е. могут состоять из нескольких «островов». В таких случаях, если их можно заранее предсказать, следует разместить дополнительные точки затравки для каждого такого «острова». Если предсказать их расположение нельзя, то следует использовать больше случайно размещенных точек затравки.

2.4. Предикат уточнения

Для экономии вычислительных ресурсов алгоритм определяет, какие области следует уточнять, а какие не представляют интереса на последующих этапах. Для этого используется предикат уточнения, который представляет собой составное условие, по которому принимается решение о дальнейшей обработке узла. В простейшем случае в предикат входит условие на достижение требуемой точности (глубины дерева), принадлежности узла граничному коридору (или n-коридору в общем случае) и ограничение области построения. При тестировании узла, ему назначаются метки (булевы флаги), которые соответствуют выполнению условий предиката уточнения. Эти метки используются при анализе результатов работы алгоритма, а также при отборе точек n-коридора.

В зависимости от специфики задачи дополнительно может быть выработано более сложное правило для определения областей, требующих уточнения. Конкретный вид такого правила вызовет необходимость относительно несложной модификации алгоритма.

Для определения принадлежности узла граничному коридору анализируется окрестность узла. Очевидно, что чем больше анализируемая окрестность, тем больше вычислительные затраты, но выше точность выявления точек граничного коридора. В алгоритме анализируются только смежные узлы вдоль осей координат, таких соседей всего $2d$, где d – число измерений. Если в окрестности присутствует узел, принадлежащий другой ячейке Вороного, то значит рассматриваемый узел вместе со своим соседом принадлежат граничному коридору.

Наряду с поиском бисекторов для определенных приложений может потребоваться поиск секторов большего порядка. Например,

трисекторы могут быть интересны в задаче обработки 3-мерных объектов, так как будут представлять собой множество связанных кривых в пространстве. В общем случае для поиска n-сектора необходимо искать узлы, принадлежащие n-коридору.

Коридор n-го порядка в большинстве случаев будет представлен фигурой размерности $\max(d-n+1,0)$, где d – размерность пространства. Очевидно, чем больше n, тем меньше соответствующий объем коридора в сравнении с общим объемом области построения диаграммы. Если требуется найти коридор n-го порядка, то выгодней получить его непосредственно, не строя всю диаграмму.

Необходимо отдельно рассмотреть два случая $n \le d$ и $n > d$. Если $n \le d$, то для построения коридора n-го порядка алгоритм производит поиск $(n-1)$-симплексов, все вершины которых являются попарно смежными узлами, принадлежащими разным ячейкам Вороного.

Необходимо учитывать, что чем выше порядок коридора, тем сложнее его обнаружить (выше вероятность ошибки). Эффективность поиска n-коридора резко снижается при значении $n > d$, так как n-коридор в большинстве случаев перестает быть связным множеством, и процедура прослеживания, описанная ниже, становится неэффективной. В этом случае алгоритм ищет $n-1$ узлов в некоторой окрестности, размер которой выбирается исходя из специфики задачи. Данные узлы вместе с рассматриваемым должны принадлежать n различным ячейкам Вороного. Для увеличения надежности алгоритм предусматривает расширение просматриваемой окрестности, что соответственно увеличивает вычислительную сложность.

Для анализа окрестности алгоритм реализует два варианта доступа к соседям узла. Самым простым из них является поиск по дереву. Такой способ не требует затрат дополнительной памяти, но обладает сложностью $O(L)$, где L – текущая глубина дерева. Другой подход заключается в хранении ссылок на соседей в самом узле. При таком подходе требуется дополнительная память, но поиск соседей будет иметь сложность $O(1)$. Достаточно хранить ссылки на соседей только вдоль осей координат, в таком случае будет требоваться $O(d)$ дополнительной памяти на один узел, где d – число измерений.

2.5. Прослеживание n-коридора

Как было упомянуто выше, при анализе недостаточно большой окрестности можно пропустить узлы, принадлежащие другим ячейкам Вороного, и ошибочно маркировать рассматриваемую ячейку как не принадлежащую к n-коридору. В результате в n-коридоре образуется разрыв. Можно решить данную проблему расширением просматриваемой окрестности, но это приведет к соответствующему росту вычислительной сложности. Алгоритм реализует альтернативный подход: применяется процедура прослеживания, которая будет такие разрывы достраивать. Неформально можно описать данную процедуру так: фрагментировать соседа по заданному направлению пока он принадлежит слою дерева с меньшим номером и удовлетворяет предикату уточнения. Формальное описание:

1) вычислить соседа по заданному направлению;

2) получить номер слоя (уровня дерева), которому принадлежит сосед;

3) если номер слоя соседа равен номеру слоя рассматриваемого узла, то завершить процедуру прослеживания;

4) если сосед удовлетворяет предикату уточнения (например, является вершиной $(n-1)$-симплекса, который принадлежит n-коридору), то фрагментировать его и перейти к пункту 1.

Процедуру следует применять во время просмотра окрестности узла для всех его соседей.

2.6. Формирование списка сайтов-кандидатов

Для корневого узла дерева в список сайтов-кандидатов включаются все сайты. На каждом шаге алгоритма список составляется для каждого узла-потомка на основе списка его родителя. Предлагаемый метод формиро-

вания такого списка, заключается в следующем: для точек, лежащих внутри узла дерева, производится оценка минимальных $\min_{est}^{(i)}$ и максимальных $\max_{est}^{(i)}$ расстояний до всех сайтов-кандидатов. Данная оценка не обязана быть точной, достаточно чтобы выполнялось неравенство:

$$\min_{est}^{(i)} \le \min_{real}^{(i)} \le dist_i \le \max_{real}^{(i)} \le \max_{est}^{(i)}, \quad (1)$$

где $\min_{real}^{(i)}$ и $\max_{real}^{(i)}$ – точные значения минимального и максимального расстояний для i-го сайта соответственно, $dist_i$ – расстояние от i-го сайта до центра узла (данное расстояние интерпретируется как расстояние от сайта до узла). После чего находится минимальное $\max_{\min}$ значение среди всех $\max_{est}^{(i)}$. Данное значение будет играть роль порога для отбора сайтов для нового списка кандидатов. В новый список следует включить только сайты, удовлетворяющие следующему неравенству:

$$\min_{est}^{(i)} \le \max_{\min}. \quad (2)$$

Неравенство (1) гарантирует, что в новый список сайтов-кандидатов попадет хотя бы один сайт. Функции для получения оценок $\min_{est}^{(i)}$ и $\max_{est}^{(i)}$ необходимо найти аналитически. В случае невозможности произвести какую-либо оценку заранее, можно принять, что:

$$\min_{est}^{(i)} = -\infty, \quad \max_{est}^{(i)} = +\infty. \quad (3)$$

В этом случае сайт будет всегда попадать в новый список сайтов-кандидатов и не будет влиять на отбор других сайтов. В качестве примера рассмотрим сайт-точку в эвклидовой метрике в d-мерном пространстве. Узел дерева разбиения пространства будет d-мерным кубом (d-кубом). Данный d-куб можно вписать в $(d-1)$-сферу с радиусом r:

$$r = \frac{a}{2}\sqrt{d}, \quad (4)$$

где a – длина стороны d-куба. $\min_{est}^{(i)}$ и $\max_{est}^{(i)}$ можно рассчитать на основе расстояния $dist_i$. Расстояние r_p от центра узла до любой точки, принадлежащей узлу дерева, не превосходит радиус r описанной $(d-1)$-сферы:

$$r_p \le r. \quad (5)$$

С учетом (5), согласно неравенству треугольника, получаем:

$$dist_i - r \le \min_{real}^{(i)}, \quad \max_{real}^{(i)} \le dist_i + r. \quad (6)$$

С учетом (6), получаем:

$$\min_{est}^{(i)} = dist_i - r, \quad \max_{est}^{(i)} = dist_i + r. \quad (7)$$

Аналогично можно получить $\min_{est}^{(i)}$ и $\max_{est}^{(i)}$ для L_p метрики, где $1 \le p \le \infty$.

На рис. 2 проиллюстрирован процесс фильтрации списка сайтов-кандидатов. Квадрат $\square ABCD$ представляет собой узел дерева в 2-мерном пространстве, точка O – центр узла и цент описанной вокруг него окружности радиусом $r = |OD|$, P_i – сайты-точки из списка сайтов-кандидатов родительского узла, P_1 – ближайший сайт-точка, P_i' – образы точек P_i, смещенные на вектор: $\mathbf{r}_i = -r\dfrac{\overrightarrow{OP_i}}{|\overrightarrow{OP_i}|}$. Обозначения на рис. 2 соотносятся с формулами (1)–(7) следующим образом: $|HP_1| = \min_{real}^{(1)}$, $|DP_1| = \max_{real}^{(1)}$, $|OP_i'| = |OP_i| - r$, $|OP_i''| = |OP_i| + r$, $|OP_i| = dist_i$.

В новый список сайтов-кандидатов попадают все точки, смещенные образы P_i' которых попадают внутрь окружности с центром в точке O и радиусом $|OP_1''| = dist_1 + r$. На рис. 2 такими точками являются P_1 и P_2, точка P_3 в новый список сайтов-кандидатов не попадает.

2.7. Диаграмма Вороного k-го порядка

Обычно диаграмма Вороного k-го порядка строится последовательно, начиная с диаграммы 1-го порядка. Описанный в данной статье алгоритм получения дискретной диаграммы Вороного после небольшой модификации позволяет находить диаграмму k-го порядка (как упорядоченную, так и неупорядоченную) за один проход. Модификация заключается в следующем:

1) при поиске ближайшего сайта необходимо искать не один, а k ближайших сайтов;

2) метка принадлежности узла к ячейке Вороного будет уже не индексом сайта, а множеством (для неупорядоченной диаграммы) или списком (для упорядоченной диаграммы) из k индексов сайтов;

3) при составлении списка сайтов-кандидатов вместо минимальной из максимальных оценок расстояния $\max_{\min}$ необходимо использовать k-ю (нумерация с 1) оценку мак-

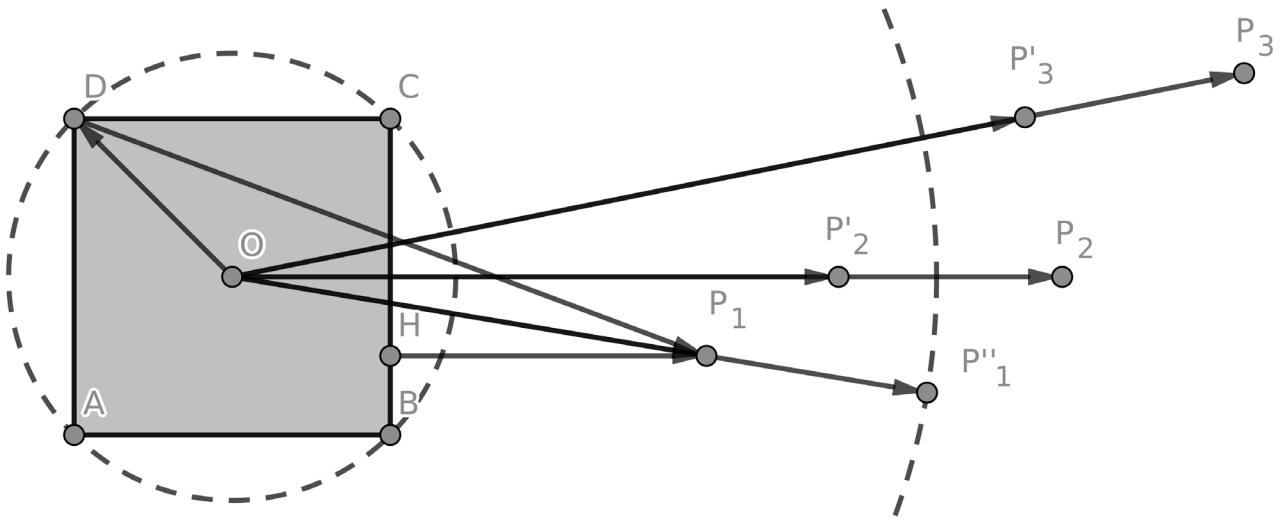


Рис. 2. Фильтрация списка сайтов-кандидатов

симального расстояния $\max_k$ в отсортированном по возрастанию списке оценок максимального расстояния.

Пункт 3 гарантирует, что в список сайтов-кандидатов попадет не менее k сайтов.

2.8. Диаграмма Вороного k-го ближайшего сайта

Аналогично диаграмме k-го порядка можно построить диаграмму k-го ближайшего сайта. Для этого достаточно в качестве метки принадлежности узла к ячейке Вороного использовать индекс k-го ближайшего сайта. В остальном процесс построения диаграммы не отличается от построения диаграммы k-го порядка.

2.9. Farthest-point диаграмма Вороного

Для построения farthest-point диаграммы соответствующая модификация алгоритма использует поиск максимумов вместо минимумов, то есть алгоритм вместо поиска ближайшего сайта находит самый дальний, а также меняет местами оценки минимума и максимума при фильтрации списка сайтов-кандидатов. При этом в новый список включаются только сайты, удовлетворяющие следующему неравенству:

$$\max_{est}^{(i)} \geq \min_{\max}, \qquad (8)$$

где $\max_{est}^{(i)}$ – оценка максимального расстояния от i-го сайта до узла, $\min_{\max}$ – максимальное значение среди всех минимальных оценок расстояний $\min_{est}^{(i)}$ для данного узла.

2.10. Вырожденные случаи и ошибки округления

Из-за обобщения метрик возможно возникновение ситуации, когда бисектор вместо размерности $d-1$ имеет размерность d (размерность пространства), то есть появляются области размерности d, в которых точки равноудалены от 2-х и более сайтов. Возможность появления такой ситуации зависит от свойств множества сайтов-функций, и ее обработка зависит от специфики решаемой задачи. Также стоит упомянуть, что в таких равноудаленных областях может наблюдаться некоторая неустойчивость, то есть сайты будут попеременно преобладать. Данная неустойчивость возникает из-за ошибок округления в операциях с плавающей точкой.

Алгоритм учитывает эти вырожденные случаи для диаграмм 1-го порядка, что позволяет избежать возникновения в равноудаленных областях ложных бисекторов, а также фрагментирования всех узлов при рекурсивном уточнении граничного коридора. В таких случаях вместо построения всего граничного коридора, строятся только его края, что значительно повышает производительность алгоритма, так как уменьшает размерность граничного коридора на 1. Вырожденные случаи, возникающие при поиске секторов большего

порядка, обрабатываются аналогично. Для диаграмм 2-го и большего порядков обработка вырожденных случаев представляет собой более сложную задачу, эффективное решение которой зависит от специфики предметной области, и в данной работе не рассматривается.

Для обработки вырожденных случаев для диаграмм 1-го порядка вместо одного ближайшего сайта алгоритм ищет все сайты, расстояние до которых отличается от расстояния до ближайшего сайта не более чем на заданную величину ошибки ε, которая задается пользователем исходя из специфики задачи.

При фильтрации списка сайтов-кандидатов к $\max_{\min}$ прибавляется ε, согласно неравенству (1), это гарантирует, что в новый список сайтов кандидатов попадут все равноудаленные сайты. Вместо метки ближайшего сайта используется множество меток равноудаленных сайтов. При поиске точек n-коридора сравниваются множества меток равноудаленных сайтов.

На рис. 3 проиллюстрирован простой вырожденный случай. Два сайта-отрезка AC и

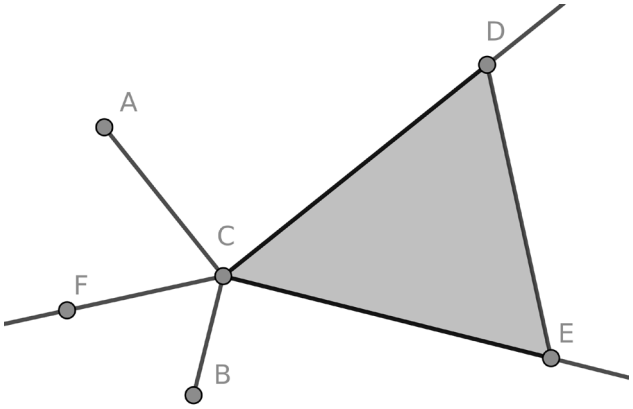


Рис. 3. Вырожденный случай

BC соприкасаются в точке C. В данном примере бисектор состоит из двух фигур: луча CF и плоского угла $\angle DCE$. Треугольник $\triangle DCE$ представляет собой часть плоского угла и является частью бисектора. Все точки плоского угла $\angle DCE$ равноудалены от сайтов-отрезков AC и BC. При обработке данного вырожденного случая алгоритм не будет уточнять область внутри плоского угла $\angle DCE$, пытаясь найти бисектор, а будет уточнять только лучи CD и CE, которые образуют данный угол.

2.11. Реализация параллелизма

Предлагаются следующие способы эффективного распараллеливания описанного алгоритма построения диаграммы Вороного:

1) параллельный поиск ближайшего сайта для узла;

2) параллельная обработка узлов в одном блоке (узлов с общим предком);

3) параллельное построение различных ветвей дерева.

Первые два способа относительно легко реализуются стандартными средствами в большинстве языков программирования. Третий способ позволяет строить разные части дерева асинхронно на разных ЭВМ. В данной работе предлагается подход, при котором отпадает необходимость в передаче данных между параллельно строящимися ветвями. Суть подхода в следующем: узлам добавляется метка, которая наследуется при дроблении узла. Узлы с меткой, отличной от текущей метки процесса (активной метки), не помещаются в очередь на обработку. В результате активно обрабатываются только те ветви, которым была назначена метка текущего процесса, а остальные ветви обрабатываются только при необходимости. В результате недостающие данные будут вычисляться благодаря процедуре прослеживания, описанной выше, и передача данных не требуется.

3. РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЙ И ИХ ОБСУЖДЕНИЕ

3.1. Оценки сложности алгоритма

Сложность алгоритма и отдельных его частей приведена в табл. 1. В таблице использованы следующие обозначения: k – порядок диаграммы, s – число сайтов, N – число разбиений по измерению, d – число измерений.

Итак, предложено два основных улучшения наивного алгоритма (полного перебора), а именно: рассчитывать только n-коридор и при поиске ближайшего сайта искать только среди сайтов из некоторой окрестности точки. Оба улучшения приводят к значительному приросту производительности по сравнению с полным перебором.

Таблица 1

Сравнение сложности алгоритмов

Критерий	Полный перебор (поиск бисекторов)	Адаптивный алгоритм	
		Поиск бисекторов	Поиск трисекторов
Количество обрабатываемых дискретных ячеек	$O\left(N^d\right)$	$O\left(sN^{d-1}\right)$	$O\left(sN^{d-2}\right)$
Вычисление расстояний	$O\left(sN^d\right)$	$O\left(ksN^{d-1}\right)$	$O\left(ksN^{d-2}\right)$
Поиск k ближайших сайтов	$O\left(sN^d\right)$	$O\left(sN^{d-1}\right)$	$O\left(sN^{d-2}\right)$
Сортировка меток k ближайших сайтов	$O\left(k\log(k)N^d\right)$	$O\left(k\log(k)sN^{d-1}\right)$	$O\left(k\log(k)sN^{d-2}\right)$
Анализ окрестности	$O\left(dkN^d\right)$	$O\left(dksN^{d-1}\right)$	$O\left(d^2ksN^{d-2}\right)$
Общая вычислительная сложность	$O\left(\left(s+k\log k+dk\right)N^d\right)$	$O\left(\left(d+\log k\right)ksN^{d-1}\right)$	$O\left(\left(d^2+\log k\right)ksN^{d-2}\right)$
Память для хранения координат	$O\left(dN^d\right)$	$O\left(dsN^{d-1}\right)$	$O\left(dsN^{d-2}\right)$
Память для списка(-ов) сайтов-кандидатов	$O\left(s\right)$	$O\left(s\right)$	$O\left(s\right)$
Память для меток k ближайших сайтов	$O\left(kN^d\right)$	$O\left(ksN^{d-1}\right)$	$O\left(ksN^{d-2}\right)$
Память для ссылок на соседей	–	$O\left(dsN^{d-1}\right)$	$O\left(dsN^{d-2}\right)$
Общая сложность по памяти	$O\left(s+kN^d\right)$	$O\left(\left(k+d\right)sN^{d-1}\right)$	$O\left(\left(k+d\right)sN^{d-2}\right)$

Максимальная эффективность будет достигнута только, если можно получить простые формулы для вычисления оценок $\min_{est}^{(i)}$ и $\max_{est}^{(i)}$ для всех сайтов-функций. В противном случае сайты, для которых невозможно получить $\min_{est}^{(i)}$, придется включать в каждый список сайтов-кандидатов, а при отсутствии адекватной $\max_{est}^{(i)}$ невозможно провести фильтрацию списка сайтов-кандидатов, что значительно снизит как вычислительную эффективность, так и эффективность по памяти. Для того, чтобы сократить расходы памяти, сайты, для которых невозможно получить $\min_{est}^{(i)}$, помещаются в отдельный список. При создании новых списков сайтов-кандидатов, в случае, если новый список не отличается от родительского списка, вместо создания нового просто копируется ссылка на старый.

В худшем случае, когда для всех сайтов невозможно получить оценку $\min_{est}^{(i)}$ и $\max_{est}^{(i)}$, сложность по памяти благодаря приему, описанному выше, не изменяется, но общая вычислительная сложность значительно увеличивается, например, при поиске бисекторов сложность будет равна:

$$O\left(\left(s\log s+dk\right)sN^{d-1}\right), \qquad (9)$$

где k – порядок диаграммы, s – число сайтов, N – число разбиений по измерению, d – число измерений.

Как можно заметить из формулы сложности, представленной в табл. 1, алгоритм наиболее эффективен в пространстве малой размерности из-за уменьшения показателя степени при множителе N, то есть из-за уменьшения размерности задачи на 1 при поиске бисекторов, и на 2 при поиске трисекторов. Дальнейшее увеличение производительности возможно при использовании большего количества априорной информации о сайтах.

3.2. Экспериментальна проверка алгоритма

Была проведена серия экспериментов для сравнения эффективности предложенного в данной работе алгоритма с наивным. Использовался оптимальный по времени вариант алгоритма для 100 точек в 2-мерном пространстве. Для тестирования использовался Matlab r2015a на ЭВМ со следующими характеристиками: центральный процессор Intel Core i7-4790 с частотой 3,6 ГГц, 16 Гбайт оперативной памяти. Результаты представлены в табл. 2. Под точностью в таблице подразумевается число разбиений вдоль оси координат, время – затраченное на работу алгоритма время, память – занятая оперативная память.

Для того, чтобы результаты экспериментов были сопоставимы, оба алгоритма выполнены в последовательном варианте, расстояние до сайтов вычисляется с помощью анонимных скалярных функций, то есть эмулируется ситуация вычисления сложных функций для измерения расстояний. Как видно из табл. 2, представленный в работе алгоритм начинает превосходить наивный (выделено жирным) по времени работы и затрачиваемой памяти, если точность превышает некоторую пороговую величину, конкретное значение которой зависит от набора сайтов, а также от свойств конкретной реализации алгоритма. Из-за простоты реализации и малых дополнительных накладных расходов наивный алгоритм преобладает над адаптивным при малой точности.

3.3. Демонстрация работы алгоритма

На рис. 4 (а–г) продемонстрированы результаты работы алгоритма. На рис. 4а показана обобщенная диаграмма Вороного для 6-ти сайтов: три сайта-точки обозначены маркером «*», составной сайт из трех точек, обозначенных маркером «x», сайт-отрезок и

Таблица 2

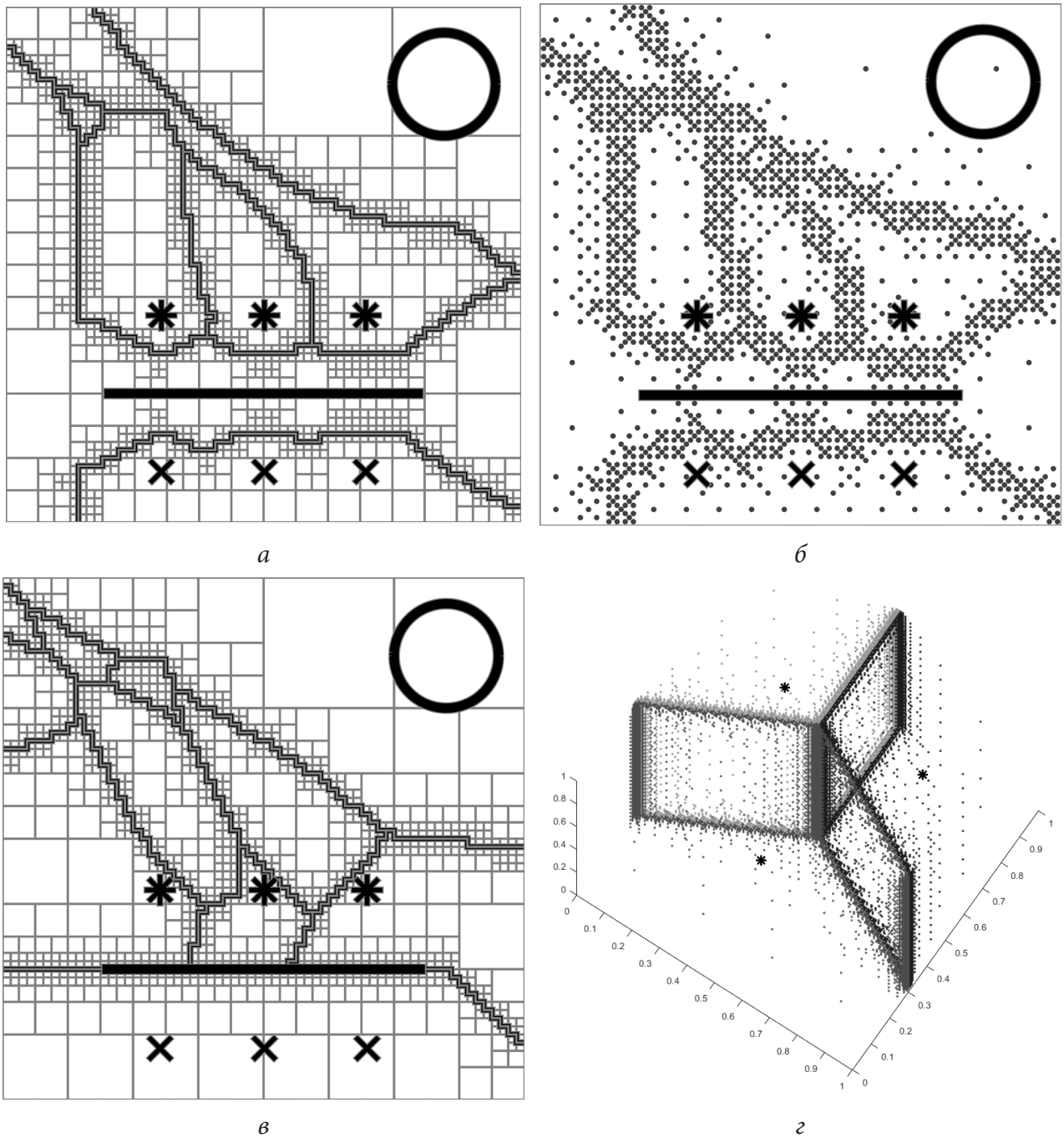
Экспериментальное сравнение алгоритмов

Точность	Наивный алгоритм		Адаптивный алгоритм			
			Поиск бисекторов		Поиск трисекторов	
	Время	Память	Время	Память	Время	Память
2	3,85 мс	84 байт	5,85 мс	2,10 Кбайт	37,23 мс	2,10 Кбайт
4	11,94 мс	0,33 Кбайт	**7,83 мс**	3,86 Кбайт	41,86 мс	3,86 Кбайт
8	46,96 мс	1,31 Кбайт	**14,33 мс**	8,38 Кбайт	59,82 мс	8,38 Кбайт
16	207 мс	5,25 Кбайт	**62,84 мс**	24,97 Кбайт	**139,72 мс**	24,97 Кбайт
32	756 мс	21 Кбайт	**345 мс**	85,67 Кбайт	**459,72 мс**	82,01 Кбайт
64	3,04 с	84 Кбайт	**1,17 с**	270 Кбайт	**1,05 с**	202 Кбайт
128	12,11 с	336 Кбайт	**2,94 с**	687 Кбайт	**1,73 с**	353 Кбайт
256	48,92 с	1,31 Мбайт	**6,47 с**	1,51 Мбайт	**2,45 с**	**512 Кбайт**
512	194,6 с	5,25 Мбайт	**13,51 с**	**3,21 Мбайт**	**3,17 с**	**676 Кбайт**
1024	776,1 с	21 Мбайт	**27,51 с**	**6,62 Мбайт**	**3,87 с**	**837 Кбайт**
2048	3088 с	84 Мбайт	**55,51 с**	**13,44 Мбайт**	**4,54 с**	**997 Кбайт**
4096	12233 с	336 Мбайт	**111,08 с**	**27,08 Мбайт**	**5,31 с**	**1159 Кбайт**

сайт-окружность. Для отрезка и окружности использовалась эвклидова метрика (L_2), для левой верхней и левой нижней точек – манхэттенская (L_1), для средней верхней и средней нижней точек – эвклидова (L_2), а для правой верхней и правой нижней точек – расстояние Чебышева (L_∞).

При использовании неэвклидовой метрики сайты уже необязательно лежат внутри своих ячеек и сами ячейки могут состоять из нескольких несвязанных между собой частей. На рис. 4а в левом верхнем углу видна треугольная ячейка, которая не содержит в себе ни одного сайта. Данная ячейка порождена сайтом-точкой с эвклидовой метрикой.

На рис. 4б показаны точки, выбранные алгоритмом для вычисления расстояний при построении диаграммы, изображенной на рис. 4а. Видно, что вычисления сосредоточены в области бисекторов. На рис. 4в показана диаграмма 2-го порядка для того же самого набора сайтов, что и на рис. 4 (а, б). На рис. 4г



а *б* *в* *г*

Рис. 4. Демонстрация работы алгоритма

показан пример для 3-мерного пространства. Сайтами являются точки, расстояние измеряется в эвклидовой метрике. На рис. 4г показаны точки, выбранные алгоритмом для вычисления расстояний при поиске трисекторов и пересечений бисекторов с границей области построения.

ЗАКЛЮЧЕНИЕ

В данной работе описан алгоритм для эффективной аппроксимации диаграммы Вороного для произвольных сайтов и метрик, или в более общем смысле, диаграммы минимизации конечного множества функций в пространстве произвольной размерности. Алгоритм наиболее эффективен в пространстве малой размерности, что следует из формулы асимптотической оценки сложности, которая экспоненциально зависит от размерности пространства.

Для работы алгоритма достаточно иметь только набор функций, способных вычислять расстояние между точкой и сайтом. Без привлечения какой-либо дополнительной информации о сайтах, алгоритм уже превосходит наивный алгоритм по производительности, так как уменьшает количество рассчитываемых точек. При увеличении количества априорной информации (при добавлении функций оценки минимального и максимального расстояний между сайтом и узлом 2^d-дерева) также увеличивается эффективность алгоритма, так как уменьшается количество сайтов, среди которых ищется ближайший.

Алгоритм позволяет применять простую аппроксимацию сайтов примитивами (точками, отрезками, дугами и т. п.), до которых относительно легко вычислять расстояние, прост в реализации и обладает относительно невысокой вычислительной сложностью, сохраняя при этом гибкость наивного алгоритма, не страдает от ошибок округления, так как для своей работы требует только вычисление расстояний между точкой и сайтом, а также содержит защиту от подобных ошибок. При построении диаграммы k-го порядка создается структура, пригодная для выполнения запросов k-ближайших соседей за время $O(d \log N)$, где d – число измерений, N – число разбиений вдоль измерения.

Дальнейшая работа направлена на изучение применения алгоритма, исследование различных путей его ускорения и реализацию параллельной версии алгоритма, в частности версии для GPU.

СПИСОК ЛИТЕРАТУРЫ

1. *Boissonnat, J. D.* Effective Computational Geometry for Curves and Surfaces / J. D. Boissonnat [et al.] ; Editors J. D. Boissonnat, M. Teillaud. – Springer-Verlag Berlin Heidelberg, 2006. – 343 p. – DOI: 10.1007/978-3-540-33259-6.

2. *Lee, D. T.* Medial axis transformation of a planar shape / D. T. Lee // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 1982. – Vol. 4(4). – P. 363-369. – DOI: 10.1109/TPAMI.1982.4767267.

3. *Karavelas, M. I.* A robust and efficient implementation for the segment Voronoi diagram / M. I. Karavelas // International Symposium on Voronoi Diagrams in Science and Engineering. – 2004. – P. 51–62.

4. *Berg, M.* Computational Geometry: Algorithms and Applications / M. Berg [at al.]. – 3rd ed. – Springer-Verlag Berlin Heidelberg, 2008. – 386 p. – DOI: 10.1007/978-3-540-77974-2.

5. *Местецкий, Л. М.* Непрерывная морфология бинарных изображений. Фигуры, скелеты, циркуляры / Л. М. Местецкий. – Москва : Физматлит, 2009. – 288 с.

6. *Okabe, A.* Spatial Tessellations: Concepts and Applications of Voronoi Diagrams / A. Okabe [et al.] ; with a foreword by D. G. Kendall. – 2nd ed. p. cm. – New York : John Wiley & Sons Ltd., 2008. – 671 p.

7. *Everett, H.* The Voronoi Diagram of Three Lines / H. Everett [at al.] // Discrete & Computational Geometry. – 2009. – Vol. 42(1). – P. 94–130. – DOI: 10.1007/s00454-009-9173-3.

8. *Lavender, D.* Voronoi Diagrams of Set-Theoretic solid Models / D. Lavender [et al.] // IEEE Computer Graphics and Applications. – 1992. – Vol. 12(5). – P. 69-77. – DOI: 10.1109/38.156016.

9. *Vleugels, J.* Approximating Generalized Voronoi Diagrams in Any Dimension / J. Vleugels, M. Overmars // Technical report 14, Depart-

ment of Computer Science, Utrecht University. – 1995. – 21 p.

10. *Vleugels, J.* Approximating Voronoi Diagrams of Convex Sites in Any Dimension / J. Vleugels, M. Overmars // International Journal of Computational Geometry & Applications. – 1998. – Vol. 8(2). – P. 201–221. – DOI: 10.1142/S0218195998000114.

11. *Teichmann, M.* Polygonal approximation of Voronoi diagrams of a set of triangles in three dimensions / M. Teichmann, S. Teller // Technical Report 766, Laboratory of Computer science, MIT. – 1997. – 12 p.

12. *Telea, A.* Visualization of Generalized Voronoi Diagrams / A. Telea, J. J. van Wijk // Data Visualization 2001. – 2001. – P. 165–174. – DOI: 10.1007/978-3-7091-6215-6_18.

13. *Boada, I.* The Voronoi-Quadtree: construction and visualization / I. Boada, N. Coll, J. A. Sellares // Eurographics 2002. – 2002. – P. 349–355. – DOI: 10.2312/egs.20021044.

14. *Boada, I.* Approximations of 3D generalized Voronoi diagrams / I. Boada [et al.] // Proceedings of the 21st European Workshop on Computational Geometry. – 2005. – P. 163–166.

15. *Boada, I.* Approximations of 2D and 3D generalized Voronoi diagrams / I. Boada [et al.] // International Journal of Computer Mathematics. – 2008. – Vol. 85(7). – P. 1003–1022. – DOI: 10.1080/00207160701466362.

16. *Yap, C. K.* Towards exact numerical Voronoi diagrams / C. K. Yap, V. Sharma, J. M. Lien // Ninth International Symposium on Voronoi Diagrams in Science and Engineering. – 2012. – P. 2–16. – DOI: 10.1109/ISVD.2012.31.

17. *Edwards, J.* Approximating the Generalized Voronoi Diagram of Closely Spaced Objects / J. Edwards [et al.] // Computer Graphics Forum. – 2015. – Vol. 34(2). – P. 299–309. – DOI: 10.1111/cgf.12561.

18. *Etzion, M.* Computing Voronoi skeletons of a 3-D polyhedron by space subdivision / M. Etzion, A. Rappaport // Computational Geometry. – 2002. – Vol. 21(3). – P. 87–120. – DOI: 10.1016/S0925-7721(01)00056-6.

19. *Foskey, M.* Efficient computation of a simplified medial axis / M. Foskey, M. Lin, D. Manocha // Journal of Computing and Information Science in Engineering. – 2003. – Vol. 3(4). – P. 274–284. – DOI: 10.1115/1.1631582.

20. *Culver, T.* Exact computation of the medial axis of a polyhedron / T. Culver, J. Keyser, D. Manocha // Computer Aided Geometric Design. – 2004. – Vol. 21(1). – P. 65–98. – DOI: 10.1016/j.cagd.2003.07.008.

21. *Sud, A.* Homotopy-preserving medial axis simplification / A. Sud, M. Foskey, D. Manocha // International Journal of Computational Geometry & Applications. – 2007. – Vol. 17(5). – P. 423–451. – DOI: 10.1142/S0218195907002434.

22. *Stolpner, S.* Sampled Medial Loci for 3D Shape Representation / S. Stolpner, S. Whitesides, K. Siddiqi //Computer Vision and Image Understanding. – 2011. – Vol. 115(5). – P. 695–706. – DOI: 10.1016/j.cviu.2010.10.014.

23. *Hoff, K. E.* III. Fast computation of generalized Voronoi diagrams using graphics hardware / K. E. Hoff III [et al.] // Proceedings of the 26th annual conference on Computer graphics and interactive techniques. – 1999. – P. 277–286. – DOI: 10.1145/311535.311567.

24. *Hsieh, H.* A simple GPU-based approach for 3D Voronoi diagram construction and visualization / H. Hsieh, W. Tai // Simulation Modelling Practice and Theory. – 2005. – Vol. 13(8). – P. 681–692. – DOI: 10.1016/j.simpat.2005.08.003.

25. *Fischer, I.* Fast Approximation of High-Order Voronoi Diagrams and Distance Transforms on the GPU / I. Fischer, C. Gotsman // Journal of Graphics, GPU, and Game Tools. – 2006. –Vol. 11(4). – P. 39–60. – DOI: 10.1080/2151237X.2006.10129229.

26. *Sud, A.* Fast proximity computation among deformable models using discrete Voronoi diagrams / A. Sud [et al.] // ACM Transactions on Graphics (TOG). – 2006. – Vol. 25(3). – P. 1144–1153. – DOI: 10.1145/1141911.1142006.

27. *Sud, A.* Interactive 3d distance field computation using linear factorization / A. Sud [et al.] // Proceedings of the 2006 symposium on Interactive 3D graphics and games. – 2006. – P. 117–124. – DOI: 10.1145/1111411.1111432.

28. *Rong, G.* Variants of Jump Flooding Algorithm for Computing Discrete Voronoi Diagrams / G. Rong, T. S. Tan // 4th International Symposium on Voronoi Diagrams in Sci-

ence and Engineering. – 2007. – P. 176–181. – DOI: 10.1109/ISVD.2007.41.

29. *Wu, X.* GPU-Based Feature-Preserving Distance Field Computation / X. Wu [et al.] // International Conference on Cyberworlds. – 2008. – Vol. 1. – P. 203–208. – DOI: 10.1109/CW.2008.62.

30. *Cao, T. T.* Parallel Banding Algorithm to compute exact distance transform with the GPU / T. T. Cao [et al.] // Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games. – 2010. – P. 83–90. – DOI: 10.1145/1730804.1730818.

31. *Jones, M.* W. 3D distance fields: a survey of techniques and applications / M. W. Jones, J. A. Baerentzen, M. Sramek // IEEE Trans Vis Comput Graph. – 2006. – Vol. 12(4). – P. 581–599. – DOI: 10.1109/TVCG.2006.56.

Хальзов С. А. – аспирант кафедры технологий обработки и защиты информации, факультет компьютерных наук, Воронежский государственный университет.
E-mail: khalzov@vsu.ru

Фертиков В. В. – канд. физ.-мат. наук, доцент, ведущий научный сотрудник кафедры информационных систем, факультет компьютерных наук, Воронежский государственный университет.
E-mail: fvv@cs.vsu.ru

APPROXIMATION OF THE ORDER-K VORONOI DIAGRAM

S. A. Khalzov, V. V. Fertikov

Voronezh State University

Annotation. This paper presents an algorithm for constructing a generalized discrete order-*k* Voronoi diagram on the grid with variable step, based on the idea of recursive partitioning space. The algorithm uses a partitioning tree (2^d-tree) as a data structure, also known as quadtree and octree for 2 and 3 dimensional spaces. The algorithm recursively refines the boundaries of Voronoi cells (bisectors), using only local properties at each point of the construction area. As a result, the calculations are concentrated in the area of the cell boundaries (the depth of the 2^d-tree is larger in the region of the cell boundaries), and a significant performance gain is achieved compared with the naive algorithm (brute force). The algorithm allows to generalize the shape of sites, the metric used, the order of the diagram and the number of measurements.
Keywords: Voronoi diagram approximation, order-*k* Voronoi diagram, minimization diagram, quadtree, octree, refinement predicate.

Khalzov S. A. – Postgraduate Student, Department of Processing Technology and Information Security, Computer Sciences Faculty, Voronezh State University.
E-mail: khalzov@vsu.ru

Fertikov V. V. – Candidate of Physical and Mathematical Sciences, Senior Researcher of Department of Information Systems, Computer Sciences Faculty, Voronezh State University.
E-mail: fvv@cs.vsu.ru