МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
УРАЛЬСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
имени первого Президента России Б. Н. Ельцина

ИНСТИТУТ ЕСТЕСТВЕННЫХ НАУК И МАТЕМАТИКИ

Кафедра вычислительной математики и компьютерных наук

МНОГОМЕРНЫЙ АЛГОРИТМ ОВЫПУКЛЕНИЯ РОЯ ТОЧЕК, НАХОДЯЩИХСЯ В НЕОБЩЕМ ПОЛОЖЕНИИ

Направление
09.04.03 «Прикладная информатика»

Допустить к защите:	Магистерская диссертация
Зав. кафедрой:	**Корабельников Алексей Алексеевич**
______________________	______________________
Нормоконтролер:	Научный руководитель: к.ф.-м.н. доц. С. С. Кумков
______________________	______________________

Екатеринбург
2020

РЕФЕРАТ

Корабельников А. А. МНОГОМЕРНЫЙ АЛГОРИТМ ОВЫПУКЛЕНИЯ РОЯ ТОЧЕК, НАХОДЯЩИХСЯ В НЕОБЩЕМ ПОЛОЖЕНИИ, выпускная квалификационная работа: стр. 1.4.1, рис. ???, библ. ??? назв.

Ключевые слова: ВЫПУКЛАЯ ОБОЛОЧКА, МНОГОМЕРНОЕ ЕВКЛИДОВО ПРОСТРАНСТВО, НЕОБЩЕЕ ПОЛОЖЕНИЕ ТОЧЕК.

Целью работы является разработка и реализация алгоритма построения выпуклой оболочки многомерного роя точек в случае их необщего положения. Под необщим положением точек подразумевается ситуация, когда в $(d-1)$-мерной гиперплоскости пространства R^d измерения лежит больше, чем d точек. Процедура реализует метод заворачивания подарка. Особенностью предлагаемой версии алгоритма является то, что после нахождения плоскости очередной гиперграни проводится построение выпуклой оболочки подроя точек, попавших в аффинное пространство этой гиперплоскости. Этот рекурсивный переход в пространство меньшей размерности проводится тогда, когда точек слишком много, чтобы образовать симплициальную гипергрань. В процессе построения гиперграни накапливается информация о её гиперрёбрах, необходимая для перехода к соседним граням. В целом, алгоритм заворачивания подарка реализует поиск в ширину в графе, в котором вершины соответствуют граням конструируемой выпуклой оболочки, а рёбра соединят вершины, соответствующие граням, имеющим общее гиперребро. Предложенный алгоритм реализован в виде программы на языке C#.

СОДЕРЖАНИЕ

Введение

Одной из важных проблем вычислительной геометрии является построение выпуклой оболочки конечного набора точек. При этом, очевидно, выпуклая оболочка является многогранником соответствующей размерности. На текущий момент существует множество алгоритмов, весьма эффективно решающих эту проблему на плоскости (см., например, [1, 2, 3, 4]). Эти алгоритмы широко применяются для решения задач в геометрическом моделировании, распознавании образов, экономике, статистики, теории управления, дифференциальных играх и во многих других областях.

Однако переход в пространства более высокой размерности резко усложняет ситуации. Основная проблема при этом — необщее положение точек в овыпукляемом рое. Если в $(d-1)$-мерной гиперплоскости пространства R^d лежит больше, чем d точек обрабатываемого роя, то положение называют необщим.

При необщем положении точек роя может получиться так, что в гиперплоскости очередной гиперграни находится слишком много точек. Соответственно, требуется выяснение того, какие из этих точек являются вершинами выпуклой оболочки, а какие нет. Кроме того, важной информацией при работе алгоритмов овыпукления является соседство граней, которое выражается через информацию о гиперрёбрах, общих для тех или иных пар граней. Соответственно, в случае, когда грань не является симплексом соответствующего пространства, нахождение её гиперрёбер также требует дополнительных усилий.

В случае плоскости эти проблемы не возникают. В этом случае размерность гиперграни равна 1, грани являются отрезками, и несимплициальные грани возникнуть в принципе не могут. Если на грани — на отрезке — оказывается несколько точек, то очень легко выяснить, какие из них являются концами отрезка, а какие нет. Также многоугольник на плоскости имеет весьма эффективное представление в виде списка вершин, перечисленных в порядке обхода многоугольника по или против часовой стрелки. Эти обстоятельства и обеспечивают эффективность алгоритмов, производящих овыпукление на плоскости [1, 2, 3].

В трёхмерном пространстве ситуация несколько ухудшается, поскольку появляются описанные проблемы. Однако на помощь приходит естественная идея: после получения плоскости очередной грани, в которой лежит больше трёх точек, можно произвести овыпукление подроя точек, попавшего в эту плоскость. При этом используются алгоритмы для работы на плоскости, поэтому, в целом, процедура получается достаточно простой и по-прежнему эффективной [4].

Однако если и дальше увеличивать размерность, то гиперграни уже в свою очередь имеют размерность 3 или выше и не допускают удобную работу с ними.

Если наложить на овыпукляемый рой условие общности положения, это снимет указаненые проблемы: все гиперграни выпуклой оболочки гарантированно будут гиперсимплексами, не нужны будут никакие дополнительные вычисления для их получения и получения их гиперрёбер. Такие алгоритмы реализованы в современных популярных библиотеках алгоритмов вычислительной геометрии CGAL (бесплатная) и LEDA (платная). Библиотеки реализованы для языка C++. Но эти процедуры неприменимы в случае, когда в плоскости какой-либо гиперграни находится слишком много точек. В этом случае работа алгоритмов происходит некорректно, даже если грани реально являются гиперсимплексами.

Вследствие этого актуальной задачей является разработка эффективной процедуры построения выпуклой оболочки роя точек многомерного пространства, находящихся в необщем положении.

Традиционно для реализации алгоритмов, нацеленных на работу в многомерном пространстве, выбирается алгоритм Джарвиса, алгоритм заворачивания подарка, как наименее использующий специфику плоскости. Однако его нужно адаптировать для случая необщего положения точек, добавив определение ситации потенциально несимплициальной гиперграни и рекурсивный вызов в подпространстве меньшей размерности для обработки таких ситуаций. Кроме того, в случае наличия несимплициальных граней, которые в свою очередь могут иметь несимплициальные гиперрёбра и т.д., нужны нетривиальные структуры данных для хранения все информации об этих гипергранях и их соседстве.

В рамках выпускной квалификационной работы была поставлена цель разработать алгоритм построения выпуклой оболочки конечного набора точек в многомерном пространстве, находящихся в необщем положении, а также сопутствующие структуры данных для хранения результатов его работы. Также была запланирована программная реализация созданной процедуры.

1 Основная часть

1.1 Базовые определения и обозначения

Определение 1 ([5]) *Выпуклой оболочкой* множества $P \subset R^d$ называется пересечение всех выпуклых подмножеств пространства R^d, содержащих P. Выпуклая оболочка множества P будет обозначаться $CH(P)$ (от английского «convex hull», «выпуклая оболочка»)

Пусть $P = \{p_1, p_2, \ldots, p_n\} \subset R^d$, $d \geqslant 2$ — конечное множество точек в d-мерном евклидовом пространстве. Несложно понять, что выпуклой оболочкой $CH(P)$ будет замкнутый многогранник в пространстве R^d или, если P вложено в какое-то аффинное подпространство, в этом подпространстве.

Определение 2 Из двух точек пространства R^d точка p_1 считается меньше точки p_2 *в лексикографическом порядке*, если найдется индекс $1 \leqslant i^* \leqslant d$ такой, что $p_{1,i^*} < p_{2,i^*}$ и для всех $1 \leqslant i < i^*$ имеем $p_{1,i} = p_{2,i}$. Здесь и далее символ $p_{k,i}$ означает i-ю координату точки p_k.

Символом $\langle \cdot, \cdot \rangle$ будет обозначаться скалярное произведение в евклидовом пространстве R^d. Символом $|\vec{n}|$ будет обозначаться длина вектора $\vec{n}$: $|\vec{n}| = \sqrt{\langle \vec{n}, \vec{n} \rangle}$.

Для ненулевого вектора $\vec{n}$ обозначим через $\vec{n}_0$ вектор, сонаправленный с $\vec{n}$ и имеющий единичную длину: $\vec{n}_0 = \vec{n}/|\vec{n}|$.

1.2 Алгоритм Джавриса на плоскости

Пусть дано множество точек $P = \{p_1, p_2, \ldots, p_n\} \subset R^2$. Требуется указать точки роя P, являющиеся вершинами многоугольника $CH(P)$.

Идея алгоритма Джарвиса (алгоритма заворачивания подарка) заключается в следующем:

1) находится какое-нибудь ребро многоугольника выпуклой оболочки и объявляется текущим;

2) с текущего ребра переходим к следующему, пока не будет достигнуто начальное ребро. После этого $CH(p)$ считается построенной.

Для построения начального ребра проделываются следующие действия. Находится точка из роя P, минимальная в лексикографическом порядке.

Пусть это точка p_1. Через неё проводится прямая, параллельная оси Oy. Иными словами, прямая, перпендикулярная оси Ox. Затем эта прямая поворачивается вокруг точки p_1 против часовой стрелки, пока она коснется одной или нескольких других точек. Самая дальняя от p_1 точка из тех, которых коснулась прямая, назначается вторым концом начального ребра. Пусть это точка p_2.

Пусть имеется текущее ребро с концами в точках p_i и $pi + 1$, причём точка p_{i+1} была найдена после точки p_i. Тогда поворачиваем прямую $(p_i p_{i+1})$ вокруг точки p_{i+1} против часовой стрелки, пока она не коснётся одной или нескольких других точек. Самая дальняя от p_{i+1} точка из тех, которых коснулась прямая, назначается вторым концом нового ребра. Пусть это точка p_{i+2}.

Если ребро $[p_{i+1} p_{i+2}]$ — это начальное ребро $[p_1 p_2]$, алгоритм заканчивает работу, а найденные точки p_1, p_2, $\ldots$, p_i являются вершинами многоугольника выпуклой оболочки $CH(P)$, перечисленными против часовой стрелки.

Видно, что центральным местом данного алгоритма является процедура поворота прямой вокруг заданной её точки до тех пора, пока она не коснётся других точек из набора. Формализуем эту процедуру.

Пусть вектор l_i — это единичный вектор, противоположно направленный вектору $\overrightarrow{p_i p_{i+1}}$ (то есть $l_i = (\overrightarrow{p_{i+1} p_i})_0$). Тогда нужные точки $\{p_1^i$, p_2^i, $\ldots$, $p_{k_i}^i\}$ роя P — это точки, на которых достигается минимум скалярного произведения $\langle l_i, (\overrightarrow{p_i, p_j})_0$, то есть такие точки, что угол между l_i и $\overrightarrow{p_i, p_j}$ максимален среди всех точек роя.

При построении начального ребра считаем, что l_0 — это орт положительного направления оси Oy.

Тогда выбор первого ребра можно схематично представить шагами, изображёнными на рис. 1–4. Красная точка на рис. 1 — это p_1, лексикографически минимальная точка роя P. На рис. 2 проведена прямая, параллельная оси Oy, и выбран вектор l_0 (красная стрелка). yf hbc/ **??** схематичнсо показан перебор точек роя в поисках второй вершины исходного ребра. Рис. 4 демонстрирует окончание этого процесса.

Действия, направленные на поиск следующего ребра выпукло оболочки схематично показаны на рис. 5, 6. На рис. 5 левая красная точка — это

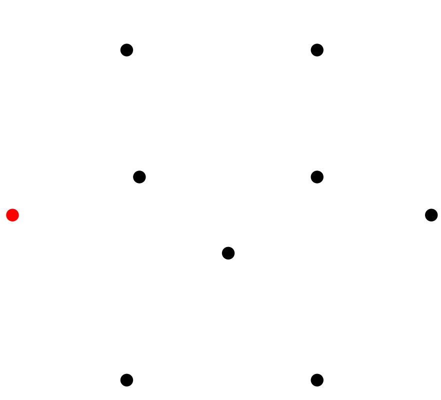

Рис. 1: Поиск начального ребра: выбор лексикорграфически минимальной точки овыпукляемого роя; красным выделена точка p_1

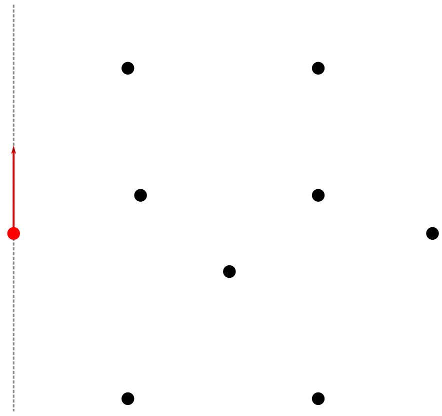

Рис. 2: Поиск начального ребра: приближение к прямой начального ребра — вертикальная прямая

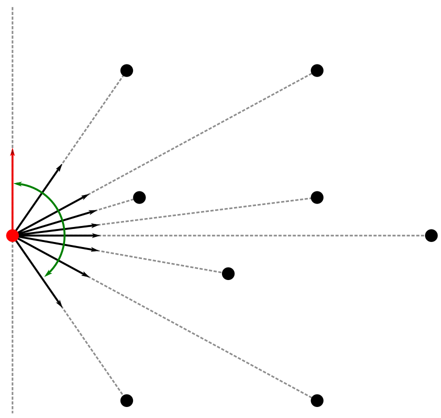

Рис. 3: Поиск начального ребра: перебор точек в поисках тех, направление на которые от p_1 даёт максимальный угол с l_0

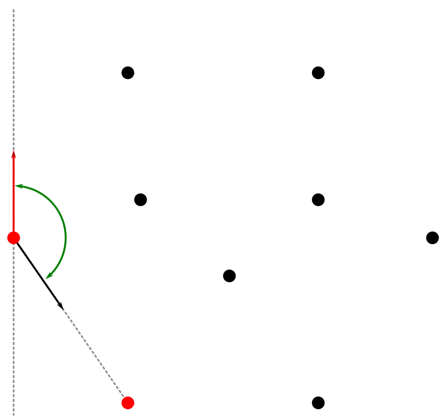

Рис. 4: Поиск начального ребра: вторая красная точка — p_2, второй конец начального ребра

начало p_i текущего ребра, правая красная точка — конец p_{i+1} текущего ребра. Перебираются точки роя, в посках тех, направление на которые от точки p_{i+1} даёт максимальный угол с вектором l_i (красная стрелка). Рис. 6 показывает окончание этого процесса.

На рис. 7 показано окончание процесса построения выпуклой оболочки.

Сложность данного алгоритма, очевидно, равна $O(hn)$, где n — количе-

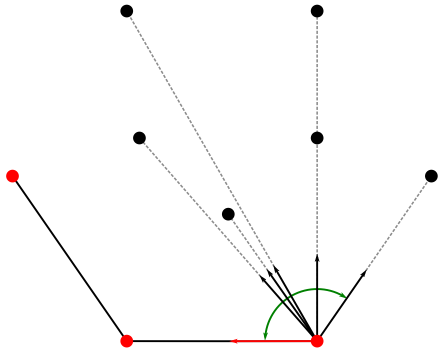

Рис. 5: Поиск очередного ребра: перебор точек в поисках тех, направление на которые от p_{i+1} даёт максимальный угол с l_i

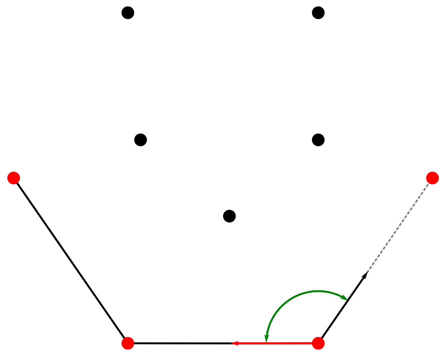

Рис. 6: Поиск очередного ребра: выбор точки p_{i+2}, второй вершины очередного ребра

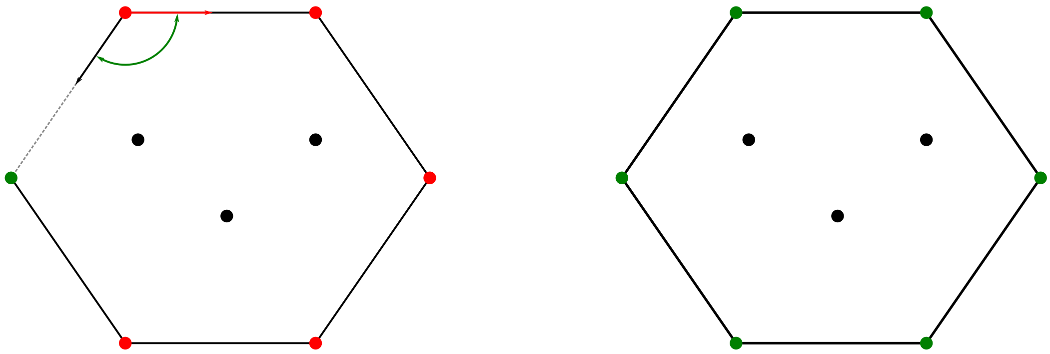

Рис. 7: Окончание процесса построения выпуклой оболочки

ство точек в рое P, а h — количество рёбер выпуклой оболочки. Сложность вытекает из того, что при построении каждого из h рёбер перебирается $n-1$ точка роя (кроме точки, вокруг которой поворачивает прямая текущего ребра).

1.2.1 Алгоритм заворачивания подарка в многомерном пространстве в случае общего положения точек

В дальнейшем приставка *гипер-* будет опускаться, когда из контекста будет понятно, об объекте какой размерности идёт речь. То есть под *плоскостью* будет подразумеваться $(d-1)$-мерная гиперплоскость в пространстве R^d, под *гранью* d-мерного многогранника будет подразумеваться $(d-1)$-мерная гипергрань, а под *ребром* — $(d-2)$-мерное гиперребро, или гипергрань (гипер)грани.

Пусть теперь базовое пространство, в котором находится рой P, имеет размерность $d > 2$. Кроме того, будем считать, что точки роя P находятся в общем положении.

Вследствие этого предположения можно утверждать, что выпуклая оболочка $CH(P)$ есть многранник с гранями, какждая из которых есть $(d-1)$-мерный симлекс и задается d точками. При этом других точек роя в плоскости грани нет.

Тогда в алгоритм Джарвиса произойдут следующие изменения.

На этапе построения начальной грани теперь надо совершить не один поворот исходной плоскости, перпендикулярной первой координатной оси, а $d-1$ поворот, чтобы плоскость «легла» на d точек роя.

Формально процедура поворота плоскости вокруг ребра теперь выглядит следующим образом. Пусть имеется плоскость α, задаваемая точками $p_1^\alpha, p_2^\alpha, \ldots, p_d^\alpha$. Пусть мы желаем повернуть её относительно ребра, задаваемого точками $p_2^\alpha, \ldots, p_d^\alpha$. Выполняются следующие действия:

1. Строим ортонормированный базис ребра, вокруг которого происходит поворот. Для этого проводим процедуру ортонормирования Грама–Шмидта (см., например, [6]) набора из $d-2$ векторов $\overrightarrow{p_2p_3}, \overrightarrow{p_2p_4}, \ldots, \overrightarrow{p_2p_d}$. Обозначим полученный ортонормированный набор векторов через $N = \{\vec{n}_i\}$, $i = 1, \ldots, d_2$.
2. Построим вектор $\vec{n}'$, дополняющий набор N до базиса плоскости α. Для этого сделаем шаг процедуры Грама–Шмидта для вектора $\overrightarrow{p_2p_1}$ на фоне набора N. Заметим, что в силу сути действий, производимых при шаге этой процедуры, вектор $\vec{n}'$ будет направлен внутрь грани, задаваемой точками p_k.
3. Перебираем точки роя P, за исключением точек $p_1^\alpha, \ldots, p_d^\alpha$. Для каж-

дой взятой точки p'_i строим вектор $\vec{n}_i$, дополняющий набор N до базиса плоскости, определяемой точками p'_i, p^α_2, $\ldots$, p^α_d. Для этого делаем шаг процедуры Грама–Шмидта для вектора $\overrightarrow{p_2 p'_i}$ на фоне набора N. Опять имеем, что полученный вектор $\vec{n}'_i$ направлен внутрь тетраэдра, задаваемого точками p'_i, p^α_2, $\ldots$, p^α_d.

4. Перебирая точки p'_i, выбираем ту, для которой угол между векторами $\vec{n}'$ и $\vec{n}'_i$ наибольший, то есть для которых скалярное произведение $\langle \vec{n}', \vec{n}'_i \rangle$ наименьшее. В силу предположения об общем положении роя P такая точка p^* будет единственной.
5. Точки $p^*, p^\alpha_2, \ldots, p^\alpha_d$ задают грань, имеющую общим ребро, задаваемое точками p^α_2, $\ldots$, p^α_d, с гранью, задаваемой точками p^α_1, p^α_2, $\ldots$, p^α_d.

Заметим, что в случае построения начальной грани, плоскость, которую мы начинаем вращать, задаётся не набором точек. Действительно, мы проводим эту плоскость через точку p_1, лексикографически минимальную в наборе P, перпендикулярно первой координатной оси. То есть эта плоскость задаётся одной точкой p_1 и своим базисом, который состоит координатных ортов e_2, e_3, $\ldots$, e_d.

В этом случае мы поочередно выкидываем первый вектор из этого набора, второй, третий и т.д., рассматривая оставшиеся как базис ребра, вокруг которого поворачиваем, а после нахождения новой плоскости, задаваемой точкой p^*, вставляем в набор вектор $\overrightarrow{p_1 p^*}$ взамен исключенного. Соответственно, перед каждым очередным поворотом требуется провести ортонормирование имеющегося базиса ребра.

Процедура перехода от начальной грани к соседним в отличие от случая на плоскости не имеет линейного характера, поскольку каждая грань уже имеет d соседей (поскольку $(d-1)$-мерный симплекс, задаваемый d точками, имеет d рёбер, которые задаются каждым из d наборов из $(d-1)$-й точки).

Для описания этой процедуры уместна графовая формализация.

Рассмотрим неориентированный граф, в котором вершины — это грани многогранника $CH(P)$. Две вершины v_i и v_j соединены ребром, если соответствующие им грани α_i и α_j имеют общее ребро. Наша задача — перебрать все вершины этого графа.

Этот перебор можно осуществить при помощи того или иного алгоритма поиска по графу. Однако при этом следует иметь в виду, что традиционно используемая рекурсивная реализация поиска в глубину, использующая системный стек, может быть чересчур накладной, так как многогранник $CH(P)$ может иметь достаточно большое количество граней. При «ручной» же организации сопутствующих структур данных более кдобным оказывается поиск в ширину. Именно он и был выбран для реализации.

При работе алгоритма поиска на графе важной структурой данных является структура, определяющая, была ли ранее посещена та или иная вершина. В случае, когда вершины графа — натуральные числа от 1 до максимального номера вершины, в качестве такой структуры выступает массив логических значений. Однако в рассматриваемой ситуации вершины графа имеют более сложную природу. Поэтому вместо массива разумно рассматривать хэш-словарь, хранящий логические выражения.

Такой подход, в свою очередь, приводит к необходимости определения разумной хэш-функции грани. Рассматривались следующие варианты вычисления хэша:

1. упорядочиваем в лексикографическом порядке точки, определяющие грань, считаем их хэши и на основе полученного набора целых чисел считаем хэш грани;
2. упорядочиваем индексы точек, определяющих грань, и на основе полученного набора целых чисел считаем хэш грани;
3. уравнение плоскости, содержащей грань, и на основе коэффициентов уравнения строим хэш грани.

Первые два способа хорошо работают в случае общего положения точек. Однако, если рой P не общего положения, то после получения плоскости очередной грани и отсева точек, попавших в эту плоскость, непонятно, какие из них являются вершинами грани и на основе каких точек считать хэш.

Поэтому в рамках предлагаемой реализации используется именно третий подход. После выполнения шага процедуры заворачивания подарка в распоряжении имеется базис полученной плоскости. Поэтому нормаль к этой плоскости можно получить как внешнее произведение с имеющимся $(d-1)$-м вектором.

Для вычисления внешнего произведения нужно решить недоопределенную систему линейных уравнений ранга $d-1$. Решение системы, например, методом Гаусса или методом Гаусса с поиском главного элемента, даст какой-то ненулевой вектор, ортогональный плоскости. Его после этого надо нормализовать и направить так, чтобы он был вектором внешней нормали, то есть давал бы положительное скалярное произведение с вектором от опорной точки плоскости к какой-нибудь точке роя, не лежащей в плоскости. Опорная точка плоскости выбирается из тех точек, которые определяют ребро, при «перекатывании» через которое плоскость была получена.

Как известно, координаты нормали являются коэффициентами линейного уравнения плоскости. Свободный член уравнения находится из скалярного произведения нормали и радиус-вектора опорной точки плоскости.

Однако числа с плавающей точкой, которыми являются полученные коэффициенты уравнения плоскости, сами по себе не очень пригодны для получения хэша, поскольку при разных путях вычисления одной и той же величины из-за округлений могут получаться вещественные числа, чуть-чуть отличающиеся на малую величину. Поэтому была взята следующая идея: все полученные вещественные коэффициенты умножаются на 10^6, у результат отбрасывается дробная часть, и хэш вычисляется на основе полученного набора целых чисел. Есть надежда, что в этих знаках после запятой числа совпадают.

1.2.2 Алгоритм заворачивания подарка в многомерном пространстве в случае общего положения точек

Как говорилось выше, две основных проблемы, которые могут возникнуть при необщем положении точек роя — появление в плоскости грани дополнительных точек, не являющихся вершинами выпуклой оболочки, и появление у грани большого количества рёбер. И как тоже было сказано выше, естественным решением обеих проблем является рекурсивный запуск алгоритма овыпукления в аффинном подпространстве грани.

В результате этого запуска и отсеиваются лишние точки, и получается информация о рёбрах грани. При этом лишние точки можно пометить как несущественные и вообще не рассматривать в дальнейших операциях. Информация о рёбрах грани получается при соответствующем оформлении алгоритма, поскольку она необходима для процедура перехода через ребро и, как следствие, генерируется алгоритмом; нужно лишь запоминать её.

Таким образом, модификация алгоритма Джарвиса в многомерном пространстве для случая необщего положения точек выглядит следующим образом.

Во-первых, в процедуре поворота плоскости может получиться так, что часть точек из тех, которые не рассматриваются как задающие ребро, всё-таки лежат в подпространстве этого ребра или совпадают с какими-то из точек, задающих ребро. В такой ситуации при ортонормировании очередного вектора $\overrightarrow{p_1p}^*$ получается нулевой вектор. Соответственно, такие точки должны пропускаться при переборе.

Во-вторых, может получиться так, что весь рой лежит в каком-то аффинном подпространстве меньшей размерности. В этом случае при построении начальной грани в какой-то момент не найдется ни одной точки, не лежащей вне очередной плоскости. В этом случае процесс построения начальной грани останваливается, точки роя пересчитываются в координаты этой грани, и алгоритм запускается в пространстве меньшей размерности.

В-третьих, после того, как установлена плоскость очередной грани, нужно выделить подрой точек, попавших в полученную плоскость. Если количество этих точек равно размерности пространства, то грань может быть лишь симплициальной и никаких дополнительных действий не требуется, вся нужная информация о грани получается тривиально. Если же количество точек этого подроя больше размерности пространства, то грань может быть несимплициальной, и требуется дополнительная его обработка для получения информации о её вершинах и рёбрах.

К этому моменту имеется ортонормированный аффинный базис плоскости: опорная точка и ортонормированный линейный базис. Точки подроя пересчитываются в координаты этого базиса, и алгоритм запускается рекурсивно на полученном рое в подпространстве размерности на единицу меньше. К концу работы алгоритма получаем выпуклую оболочку этого роя и полную информацию о её устройстве. Но при этом все точки и векторы выражены в координатах подпространства, а не исходного пространства. Та же проблема имеется и в ситуации, когда весь рой содержится в подпространстве меньше размерности.

Как следствие, требуется обратный пересчёт из координат подпространства в координаты исходного пространства. Пересчёт векторов проблемы не составляет, поскольку вектора базиса подпространства выражены в координатах исходного простанства. В принципе, таким же образом можно пересчитывать и точки, однако после пересчёта мы получим новые объекты, в то время как требуется привязаться к объектам точек, хранящимся в структуре роя.

Поэтому в первой и в третьей ситуация при проецировании точек в подпространство требуется запоминать ссылки на объекты исходных точек. При обратном пересчёте объекты точек в координатах подпространства заменяются на иходные объекты точки, получаемые по запомненным ссылкам.

Также алгоритм возвращает информацию о рёбрах полученной грани: это грани выпуклой оболочки, построенной в подпространстве.

1.3 Схема предлагаемого алгоритма

В результате был предложен алгоритм построения выпуклой оболочки, приведённый на листинге 1. В строке 1 происходит проверка, что точки являются двумерными. Если это так, то вызывается метод построения выпуклой оболочки в двухмерном пространстве и алгоритм заканчивает работу. Двумерный алгоритм описан в разделе 1.2.

В строке 8 производится поиск начальной гиперграни. Алгоритм описан в разделах 1.2.1 и 1.2.2. Начальная грань записывается в очередь граней на обработку, как того требует алгоритм поиска в ширину.

Затем реализуется традиционный обход графа в ширину. На каждом шаге цикла, если очередь обрабатываемых граней непуста, из неё выбирается очередная грань. В цикле по её рёбрам поризводится переход через соответствующее ребро и, если новая грань еще не создавалась, она создаётся и записывается в результат и в очеред на обработку.

В строке 19 проверяется, задают ли точки P_g, лежащие в новой плоскости, симплекс. Если задают, то вызывается отдельный метод построения симплициальной гиперграни. Иначе происходит пересчёт точек P_g в базис гиперплоскости и рекурсивный вызов метода для обработки множества точек P_g с последующим обратным пересчётом результата в координаты текущего пространства.

Листинг 1: Алгоритм построения выпуклой оболочки $CH(P)$

Входные данные: Множество точек $P = \{p_i\}_{i=1}^n \subset R^d$

Результат: Выпуклая оболочка $CH(P)$, заданная набором граней.

если $P \in R^2$ **тогда**
    Найти выпуклую оболочку CH_2 на плоскости при помощи метода Джарвиса
    Построить векторы рёбер полученного многоугольника
    возвратить CH_2
иначе
    Инициализировать пустую выпуклую оболочку CH
    Инициализировать пустое множество использованных ребер H
    Найти исходную грань F_0 выпуклой оболочки
    Добавить F_0 в очередь S
    Пометить F_0 как уже создававшуюся
    до тех пор, пока $S \neq \varnothing$ **выполнять**
        Изъять из головы S грань F
        цикл по рёбрам e из списка рёбер F **выполнять**
            Добавить e в H
            Найти гиперплоскость g, содержащую $CH(e \cup p)$, где $p \in P$, и все точки $p_i \in P$ принадлежат отрицательному полупространству плоскости g
            если плоскость g уже создавалась **тогда**
                Перейти к следующему ребру e
            Найти множество точек $P_g = \{p_i \in P \mid p_i \in g\}$
            если $|P_g| = n$ **тогда**
                Построить симплициальную грань F'
            иначе
                Пересчитать рой P_g в координаты плоскости g
                $F' = \mathrm{CH}(P_g)$
                Пересчитать элементы F' в координаты исходного пространства
            если $P = P_g$ **тогда**
                возвратить F'
            Добавить грань F' в CH
            Пометить грань F' как уже создававшуюся
    возвратить CH

1.4 Программная реализация

1.4.1 Тестирование реализации

Заключение

В данной работе был разработан алгоритм решение задачи построения выпуклой оболочки конечного набора точек R^n размерности, находящихся в необщем положении.

Алгоритм не является окончательным и требует дальнейшей доработки.

Оставшиеся задачи и алгоритм будут доработаны в будущей выпускной работе.

Список литературы

[1] Preparata F. P., Shamos M. I. Computational Geometry. An Introduction. — New York: Springer, 1985. — 400 с.

[2] Препарата Ф., Шеймос М. Вычислительная геометрия: Введение. пер. с англ. под ред. Банковского Ю. М. — М.: Мир, 1989. — 478 с.

[3] de Berg M., Cheong O., van Kreveld M., Overmars M. Computational Geometry. Algorithms and Applications, 3rd Ed. — Berlin: Springer-Verlag, 2008. — 398 с.

[4] Ивановский С. А., Преображенский А. С., Симончик С. К. Алгоритмы вычислительной геометрии. Выпуклые оболочки в трехмерном пространстве // Компьютерные инструменты в образовани. — 2007. — № 3. — С. 3–17.

[5] Kokichi S. Robust gift wrapping for the three-dimensional convex hull // Journal of Computer and System Sciences. — 1994. — Vol. 49, issue 2, pp. 391–407.

[6] Беклемишев Д. В. Курс аналитической геометрии и линейной алгебры. — М.: Наука, 2005. — 308 с.