УДК 519.688

Д.А. Петренко, А.В. Скворцов, Р.О. Куленов

СРАВНЕНИЕ ТРИАНГУЛЯЦИЙ С ПОМОЩЬЮ ХЕШ-ФУНКЦИЙ

Ставится задача сравнения структуры и геометрии двух заданных триангуляций. Для решения поставленной задачи предлагается использовать хеширующие функции. Предлагается несколько вариантов хеш-функций на основе известных криптографических методов типа RSA. Проводится анализ полученных алгоритмов хеширования.

Триангуляция часто применяется в различных задачах машинной графики, вычислительной геометрии, методах конечных элементов, геоинформатике, различных инженерных задачах.

По модели триангуляции часто выполняется построение изолиний, профилей по поверхности, расчёт объёмов земляных работ и пр. При больших объёмах исходных данных некоторые алгоритмы решения указанных задач являются достаточно трудоёмкими [1 – 3]. Во многих программных продуктах после изменения исходных данных вызывается соответствующий алгоритм для повторного вычисления изолиний, объёмов работ и т.д. Однако в некоторых случаях этого можно избежать, например, если вновь построенная триангуляция в области наших интересов (в месте, где строятся изолинии и считаются объёмы работ) эквивалентна предыдущей.

Таким образом, возникает задача сравнения двух триангуляций. При этом возможны два класса вопросов: 1) совпадают ли две данные триангуляции по ряду параметров; 2) различаются ли две данные триангуляции по ряду параметров. Внешне эти два вопроса являются дополнительными друг к другу, однако они имеют существенно различный смысл в контексте предлагаемого в настоящей работе вероятностного метода решения.

Для сравнения триангуляций можно произвести точное побайтовое сравнение структур триангуляций. Однако в ряде случаев это сравнение не даст правильного результата. Например, в случае, когда две триангуляции совпадают как геометрическое место точек, но отличаются только порядком задания треугольников или узлов.

Для решения поставленной задачи в настоящей работе предлагается использовать метод хеширования [4,5]. При этом нам требуется построить такие хеш-функции, которые были бы устойчивыми к изменению порядка задания элементов триангуляции, чтобы избежать выдачи ложного сообщения о различии триангуляций, но в то же время позволяли различать триангуляции с минимальным их изменением как геометрических множеств точек.

ЗАДАЧА СРАВНЕНИЯ ТРИАНГУЛЯЦИЙ

Прежде всего, отметим, что в дальнейшем в настоящей работе мы будем предполагать, что наша триангуляция определена на структуре данных «Узлы и треугольники» [1]. В этой структуре узлы определены только как набор соответствующих координат (двумерных или трехмерных), а треугольники – как тройки ссылок на образующие узлы и соседние треугольники.

Попробуем проанализировать параметры триангуляции, по которым их следует различать. В первую очередь, это координаты узлов триангуляции. Во-вторых, топологические связи треугольников: при одних и тех же узлах треугольники могут быть связаны по-разному, что в конечном итоге даст различную форму триангуляции. Кроме того, в ряде случаев, возможно, следует дополнительно учесть некоторые дополнительные признаки треугольников. Например, при сравнении не следует учитывать треугольники, помеченные как невидимые или же находящиеся вне области наших интересов; в то же время, следует различать треугольники двух разных триангуляций, имеющие разные значения некоторых дополнительных признаков.

В связи с большим разнообразием возможных постановок задачи сравнения вначале рассмотрим задачу сравнения геометрической формы триангуляций.

Задача 1. Пусть даны две триангуляции T_1 и T_2, при этом для каждого узла заданы его координаты, а для каждого треугольника – ссылки на образующие узлы. Требуется установить, различаются ли две триангуляции как геометрическое место точек на плоскости или в пространстве.

Для решения данной задачи нам потребуется учесть координаты узлов триангуляции и её топологию в виде номеров ссылок треугольников на образующие узлы. Заметим, что учитывать номера смежных треугольников нет необходимости, так как эта информация является избыточной, поскольку она может быть однозначно восстановлена по номерам образующих узлов и координатам узлов.

В данной работе предлагается сохранить структуру триангуляций в некоторый бинарный поток данных, после чего применить к потоку в целом какую-либо хеш-функцию. Поскольку результаты хеш-функции, вычисленные по одним и тем же входным данным, равны, то задачу можно переформулировать следующим образом. Требуется найти алгоритм сохранения триангуляции в поток таким образом, чтобы для двух эквивалентных триангуляций содержимое потоков было бы одинаковым, а для отличающихся – различным.

Поскольку порядок задания вершин триангуляции не должен влиять на содержимое потока с данными об узлах, необходимо пронумеровать узлы триангуляции таким образом, чтобы на любой триангуляции с такими же координатами узлов алгоритм сравнения выдавал одинаковый результат. Тогда можно будет сохранять в поток количество узлов триангуляции и их координаты в соответствии с возрастанием порядковых номеров узлов.

Предположим, что мы пронумеровали узлы некоторым образом. Теперь рассмотрим вопрос сохранения топологии треугольников. В связи с тем, что в двух геометрически эквивалентных триангуляциях порядок задания узлов и треугольников может быть разным, то необходимо также некоторым способом пронумеровать треугольники, чтобы на любой триангуляции идентичной топологии алгоритм выдавал одинаковый результат. Тогда можно будет, сохранив в поток количество треугольников, для каждого треугольника сохранять номера вершин, образующих треугольник, в порядке возрастания их номеров.

Задача 2. Пусть даны две триангуляции T_1 и T_2, при этом для каждого узла заданы его координаты, а

для каждого треугольника – ссылки на образующие узлы. Кроме того, для всех узлов и треугольников задан некоторый набор вспомогательных признаков. Требуется установить, различаются ли две триангуляции как геометрическое место точек и по набору соответствующих признаков элементов триангуляции.

При решении этой задачи кроме топологии и координат узлов триангуляции нам потребуется учесть соответствующие признаки элементов. Поскольку порядок задания элементов в триангуляции T_1 и T_2 может отличаться, необходимо при сохранении в поток признаков элементов учитывать эту особенность. Для решения поставленной задачи предлагается использовать нумерацию элементов триангуляции, которая применялась при решении предыдущей задачи. В этом случае можно будет последовательно сохранять в поток интересующие признаки триангуляции. При сохранении признаков можно применять два подхода:

1. Сохранять в поток последовательно в порядке возрастания номера только тех элементов триангуляции, которые соответствуют требуемому признаку (эффективно, когда количество таких элементов невелико).

2. При сохранении координат (для узлов) или индексов образующих треугольник узлов (для треугольников) так же сохранять в поток и сам признак (эффективно, когда «вес» признака мал).

Для решения поставленных задач требуется алгоритм нумерации узлов и треугольников, не зависящий от их порядка в структуре триангуляции. Для этого предлагается отсортировать все узлы триангуляции по их X-координатам, а узлы, имеющие равные X-координаты, дополнительно отсортировать по Y-координатам. Аналогичным образом сортируются все треугольники, при этом в качестве сравнивающихся координат используются координаты центров треугольников. В результате для двух эквивалентных триангуляций, различающихся только порядком составляющих её элементов, мы получим одинаковые индексы элементов.

В целом поток, в который последовательно записаны координаты узлов триангуляции, индексы образующих треугольники узлов и, возможно, дополнительная информация об узлах и треугольниках, мог бы быть использован для однозначной идентификации триангуляции. То есть, в принципе, для сравнения двух триангуляций T_1 и T_2 можно было бы, сохранив их в соответствующие потоки S_1 и S_2, сравнить по байтам содержимое потоков и по результату сравнения потоков сделать выводы об эквивалентности триангуляций, но такой подход нерационален в смысле использования памяти и неудобен на практике.

Поэтому после того, как триангуляция была сохранена в поток, предлагается применить к потоку в целом какую-либо хеш-функцию и для сравнения триангуляций сравнивать только значения хеш-функций, вычисленных на данных потоках. Это позволяет хранить в триангуляции только однажды вычисленное хеш-значение, не генерируя поток данных триангуляции для каждой операции сравнения. В случае, если одна из триангуляций зафиксирована и меняется редко, то такой подход позволяет в 2 раза уменьшить объем временно используемой памяти для хранения потока данных триангуляций.

Кроме того, рассмотрим такой пример. Пусть сравниваемые триангуляции – это по сути одна и та же модель, но построенная в разное время. Например, набор исходных данных был изменен (изменены координаты существующих или добавлены дополнительные объекты), в результате, после построения новой триангуляции, необходимо принять решение о необходимости перестроения изолиний, трёхмерной модели и пр. Тогда для сравнения потоков двух триангуляций (бывшей и текущей) необходимо было бы хранить поток данных от старой триангуляции, что очень невыгодно по затратам памяти. В этом смысле сохранение только одного хеш-значения для старой триангуляции даёт существенный выигрыш.

ВЫБОР ХЕШ-ФУНКЦИИ

При выборе хеш-функции следует обратить внимание на её статистическую независимость. Поскольку для различных триангуляций результаты хеш-функции потенциально могут оказаться одинаковыми [5], то существует вероятность того, что после сравнения может быть выдан ошибочный результат о совпадении двух различающихся триангуляций. Поэтому важно (это скорее необходимое, но не достаточное условие), чтобы результаты хеш-функций были распределены равномерно на множестве их возможных значений для всех возможных триангуляций. В качестве хеш-функций в данной работе предлагается использовать следующие варианты:

А. Применение операции «Исключающее ИЛИ» для всех данных из полученного потока, который рассматривается как последовательность 64-битных целых чисел. Далее в настоящей работе этот метод будем называть XOR.

Б. Применение операции «Исключающее ИЛИ» для всех данных из полученного потока, который рассматривается как последовательность 64-битных целых чисел, со сдвигом результата каждой операции влево на n бит. Следует заметить, что для получения более равномерного распределения значений хеш-функции в качестве n лучше брать небольшое простое число, например 3, 5 или 7 (этот результат чисто эмпирический, не обоснованный теоретически, однако проверенный авторами на практике при экспериментальной оценке различных алгоритмов). Далее в работе этот метод будем называть XOR*.

В. Использование функций Microsoft CryptoAPI с применением одного из алгоритмов хеширования, стандартно встроенных в операционную систему Microsoft Windows 9x/Me/NT/2000/XP: MD2, MD4, MD5 или SHA [6 – 8]. Далее в работе эти методы будем называть соответственно MD2, MD4, MD5, SHA,

Сравнение алгоритмов MD2, MD4, MD5, SHA

В криптографии хеширование часто применяется для цифровой подписи сообщений [6,8,9], при этом в хеш-функцию в качестве входных параметров передаётся, помимо исходных данных, ключ (приватный или публичный). Тогда данные ключа используются при вычислении значения хеш-функции. В нашем случае можно обойтись алгоритмами, не требующими ключей для вычисления хеш-значений.

Ниже перечислены несколько алгоритмов, используемых для вычисления хеш-функций, не требующих для работы ключей. Все эти алгоритмы поддерживаются Microsoft RSA Base Cryptographic Provider, стандартно поставляемым со всеми операционными системами Microsoft.

Семейство алгоритмов RSA для вычисления хеширующих значений использует стандартную методику RSA при работе с публичными и приватными ключами. При этом заранее (при разработке программы хеширования) создаётся некоторый предопределенный ключ, с помощью которого производится генерация хеш-значения. Именно поэтому статистические параметры получаемого хеш-значения определяются свойствами самого алгоритма RSA, который в настоящее время достаточно полно исследован. В частности, количество различных получаемых хеш-значений пропорционально величине публичного или приватного ключа в алгоритме RSA. В различных реализациях алгоритма RSA обычно используются ключи длиной не менее 512 бит, при этом после усечения сгенерированного хеш-значения до 128 или 160 бит (в реализациях, поставляемых Microsoft) полученные хеш-значения оказываются распределены равномерно на всем диапазоне возможных 128 или 160-битных значений соответственно.

Кратко опишем конкретные поставляемые Microsoft реализации метода RSA.

MD2 – хеширующий алгоритм, генерирующий 128-битное хеш-значение. Алгоритм оптимизирован для 8-битных компьютеров, поэтому на практике работает достаточно медленно.

MD4 и MD5 – хеширующие алгоритмы, генерирующие 128-битное хеш-значение. Эти два алгоритма оптимизированы для 32-битных компьютеров. Скорости и качество работы этих алгоритмов практически не отличаются.

SHA – хеширующий алгоритм, генерирующий 160-битное хеш-значение. Разработан совместно Национальным институтом стандартов и технологий (NIST, США) и Агентством национальной безопасности (NSA, США).

Все перечисленные криптографические хеш-функции должны выдавать статистически независимые результаты на различных входных данных, даже если они отличаются на такую ничтожно малую величину, как один бит.

Хеширующие алгоритмы MD2, MD4 и MD5 были разработаны в корпорации RSA Data Security, Inc. Среди перечисленных алгоритмов Microsoft рекомендует использовать MD5, как наиболее новый и быстрый алгоритм хеширования.

ЭКСПЕРИМЕНТ

Авторами было проведено экспериментальное моделирование работы предложенных алгоритмов на примерах сгенерированных триангуляций на различных наборах случайных точек. Для всех триангуляций было вычислено значение хеш-функций по предложенным алгоритмам. Для оценки распределения результатов хеш-функций в эксперименте использовались упрощённые функции, возвращающие 8-битное значение, поскольку для проверки гипотезы о равномерном распределении 128-битных значений на практике потребуется огромное количество тестов (не менее $5 \cdot 2^{128}$) [10], выполнить которые за разумное время невозможно. Для сведения 64-, 128- и 160-битных значений к 8-битным к ним была применена побайтно операция «исключающее ИЛИ».

На рис. 1 представлены графики распределения результатов хеш-функций, вычисленных на 100000 триангуляций, построенных на 5000 равномерно распределённых в единичном квадрате точках.

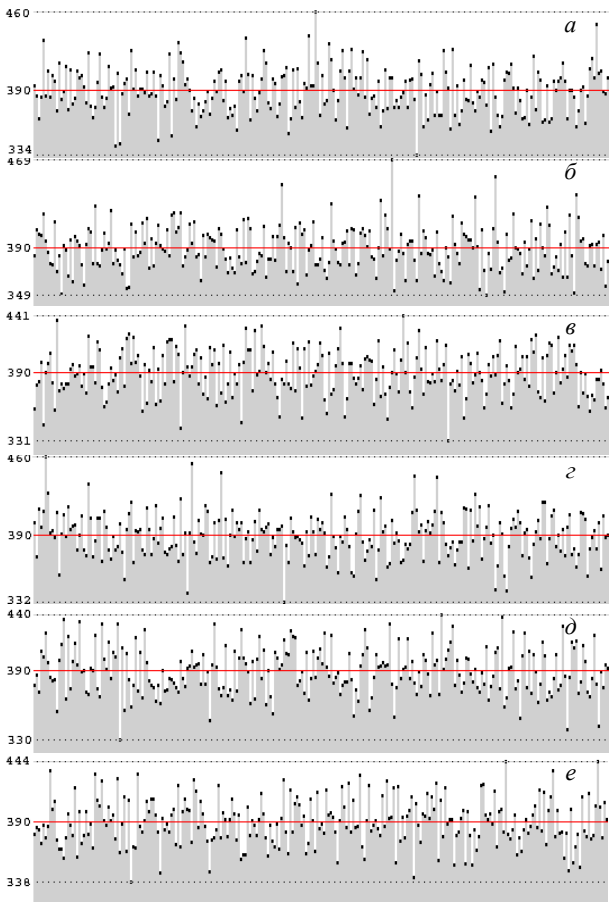

Рис. 1. Графики распределения результатов различных хеш-функций: *а*) алгоритм MD2; *б*) алгоритм MD4; *в*) алгоритм MD5; *г*) алгоритм SHA; *д*) алгоритм XOR; *е*) алгоритм XOR*

Приведённые результаты были оценены на соответствие равномерному распределению по χ^2-критерию [10]. При уровне значимости $\alpha = 0{,}05$ все меры отклонения χ^2 истинного распределения от равномерного меньше критического $\chi^2_\alpha = 292{,}97$, вычисленного для данного случая, что не даёт повода сомневаться в правильности предположения о равномерном распределении результатов хеш-функций на множестве её возможных значений:

$$\chi^2 = \sum_{i=0}^{255} \frac{(M_i - np_i)}{np_i},$$

где M_i – число значений хеш-функций, равных $i = \overline{0,255}$, n – объём выборки значений хеш-функций (количество испытаний), p_i – «теоретическая вероят-

ность» того, что результат хеш-функции будет равен i (при 8-битном значении хеш-функции все $p_i = 1/256$).

Следует, однако, отметить, что при использовании простой 64-битной хеш-функции (не криптографической), такой, как XOR, распределение её результатов может несколько ухудшиться по сравнению с равномерным распределением. Такая ситуация может возникать при вычислении значения хеш-функции $XOR(T)$ по триангуляции, построенной на регулярном множестве точек. Поскольку в поток сохраняются координаты узлов триангуляции и целочисленные индексы её элементов, в некоторых случаях $XOR(T)$ будет содержать большое количество нулевых бит. При использовании криптографических алгоритмов хеширования данный эффект проявляться не будет.

В табл. 1 приведены результаты тестирования предложенных в данной работе хеширующих функций. Из таблицы видно, что все предложенные хеш-функции имеют равномерное распределение результатов на множестве их возможных значений. Среднее время работы алгоритмов указано в абстрактных единицах времени и может использоваться для качественного сравнения их скорости работы. Сравнивая алгоритмы по скорости, можно сделать вывод, что наиболее выгодным с точки зрения оценки скорости работы функций представляется использование хеш-функции, работающей по алгоритму MD5.

Таблица 1

Результаты тестирования хеширующих алгоритмов

Хеширующий алгоритм	Значение χ^2	Среднее время работы, у.е.
MD2	262,75	637
MD4	248,14	32
MD5	256,29	31
SHA	255,90	50
XOR	273,76	90
XOR*	270,57	93

Также был проведён следующий эксперимент: была построена триангуляция T_1 на 10000 случайных точек, равномерно распределённых в единичном квадрате. По триангуляции T_1 было вычислено 128-битное значения хеш-функции H_1, работающей по алгоритму MD5. После этого триангуляция T_1 была зафиксирована, и с неё была получена её точная копия T_2. Было проведено 100000 незначительных модификаций триангуляции T_2, изменяющих её по одному из следующих параметров: 1) незначительное изменение координат одного из узлов триангуляции; 2) изменение значения одного признака случайного узла триангуляции; 3) перестроение пары соседних треугольников (переброска ребра); 4) изменение одного признака случайного треугольника. Для каждой модификации по такому же алгоритму вычислялось 128-битное значение хеш-функции H_2. Ни одно значение H_2 не совпало с исходным значением H_1, что даёт основание сделать вывод о достаточной надёжности предложенного метода.

ЗАКЛЮЧЕНИЕ

В заключение вернемся к поставленным в начале работы и возникающим при сравнении триангуляций двум вопросам. С помощью предложенного метода хеширующих функций мы можем уверенно сделать вывод только о различности двух исходных триангуляций, если мы получили различные хеш-значения. В случае, если хеш-значения совпали, то мы можем лишь утверждать, что, скорее всего, триангуляции также совпадают, причём с большой степенью уверенности (вероятность ошибки составляет $\theta(2^{-n})$, где n – длина сгенерированного хеш-значения в битах).

Предложенный в данной работе метод сравнения триангуляций реализован на практике и внедрен в систему автоматизированного проектирования IndorCAD и геоинформационную систему IndorGIS, разработанные в ООО «ИндорСофт», г. Томск.

ЛИТЕРАТУРА

1. *Скворцов А.В.* Триангуляция Делоне и её применение. Томск: Изд-во Том. ун-та, 2002. 127 с.
2. *Скворцов А.В., Костюк Ю.Л.* Применение триангуляции для решения задач вычислительной геометрии // Геоинформатика. Теория и практика. Вып. 1. Томск: Изд-во Том. ун-та, 1998. С. 22–47.
3. *Скворцов А.В.* Особенности реализации алгоритмов построения триангуляции Делоне с ограничениями // Вестник ТГУ. 2002. № 275. С.90–94.
4. *Ахо А., Хопкрофт Д., Ульман Д.* Построение и анализ вычислительных алгоритмов: Пер. с англ. М.: Мир, 1979. 536 с.
5. *Кнут Д.* Искусство программирования, том 3. Сортировка и поиск. М.: Вильямс, 2000. 832 с.
6. *Щербаков А., Домашев А.* Прикладная криптография. М.: Русская редакция, 2003. 416 с.
7. *Schieber R.* Site Security Using Microsoft's CryptoAPI. N.Y.: Wrox, 2001. 49 p.
8. *Bondi R.* Cryptography for Visual Basic: A Programmer's Guide to the Microsoft CryptoAPI. N.Y.: John Weiley & Sons, 2000. 480 p.
9. *Шнайер Б.* Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. М.: Триумф, 2002. 816 с.
10. *Бронштейн И.Н., Семендяев К.А.* Справочник по математике для инженеров и учащихся втузов. М.: Наука, 1986. 544 с.

Статья представлена кафедрой теоретических основ информатики факультета информатики Томского государственного университета, поступила в научную редакцию 15 мая 2003 г.