DOI 10.37882/2223-2966.2025.07.11

О КРИТЕРИЯХ ВЫБОРА ОЧЕРЕДНОГО СИМПЛЕКСА В АЛГОРИТМАХ ТРИАНГУЛЯЦИИ НА ОСНОВЕ ДВИЖУЩЕГОСЯ ФРОНТА

ON THE SELECTION CRITERIA FOR THE NEXT SIMPLEX IN TRIANGULATION ALGORITHMS BASED ON ADVANCING FRONT METHOD

A. Dronov

Summary. The purpose of this study is to develop criteria for evaluating the quality of a newly added simplex in the context of the advancing front triangulation algorithm. The quality assessment is mainly based on checking for intersections between simplices of different dimensions, as well as calculating the distances between them. The advancing front algorithm is well suited for complex geometric regions, but requires that criteria be satisfied when adding another simplex at each step. An analysis of the existing capabilities was carried out to monitor compliance with the necessary properties and criteria, on the basis of which a set of rules was formulated that formed the basis of the mesh generator.

Keywords: computational algorithms, mesh generation, triangulation, advancing front method, simplex, selection criteria, collision detection.

Дронов Антон Геннадьевич
аспирант, ФГАОУ ВО «МГТУ «СТАНКИН» (г. Москва)
chesston1t@yandex.ru

Аннотация. Цель данного исследования состоит в разработке критериев для оценки качества вновь добавляемого симплекса в рамках алгоритма триангуляции методом движущегося фронта. Оценка качества производится в основном на основе проверки наличия пересечений между симплексами разной размерности, а также на основе вычисления расстояний между ними. Алгоритм движущегося фронта хорошо подходит для сложных геометрических областей, но требует соблюдения критериев при добавлении очередного симплекса на каждом шаге. Был произведен анализ существующих возможностей, позволяющих контролировать соблюдение необходимых свойств и критериев, на основе которых был сформулирован набор правил, которые легли в основу генератора сеток.

Ключевые слова: вычислительные алгоритмы, генерация сеток, триангуляция, метод движущегося фронта, симплекс, критерии выбора, проверка пересечений.

Введение

При решении физических задач зачастую возникают системы дифференциальных и интегральных уравнений, которые зачастую решаются с помощью численных методов с использованием расчетных сеток, которые позволяют дискретизировать сложную расчетную область для её представления в конечномерном пространстве.

Несмотря на большое количество различий в них как по форме и структуре, так и по способу их построения, список предъявляемых к соблюдению свойств сеток достаточно часто является идентичным. В частности, достаточно популярным свойством, помимо отсутствия самопересечений, является свойство конформности. Сетка считается конформной, если любые два элемента либо не имеют общих узлов, либо ровно одну общую сущность более высокого уровня (ребро или грань). Соблюдение и поддержание этих свойств требует применения и адаптации специальных методик и правил.

В статье описываются сформированные критерии, которые используются при разработке последовательной и параллельной версиях генератора симплициальных сеток [1, 2], который идет в дополнение к разработанному в ИПМ имени Келдыша РАН комплексу MARPLE3D [3]. В комплексе для расчетов используются расчетные сетки нерегулярной структуры, в своем основании которые состоят из фигур произвольной формы.

В качестве алгоритма построения таких сеток выбран метод движущегося фронта, который заполняет расчетную область, стартуя от её границ по направлению к «материалу». Алгоритм может применяться как для триангуляции поверхностей, так и для тетраэдризации трехмерных областей, хотя и существуют некоторые особенности этих случаев; без ограничения общности сформулируем главную его идею.

Основополагающим является понятие фронта, куда включаются ребра, на основании которых нужно построить новую грань. В случае триангуляции на начальном этапе для каждой грани исходной геометрии во фронт добавляются ребра, полученные в ходе дискретизации её сторон. На каждом шаге из фронта выбирается активное ребро, наиболее подходящее в зависимости от выбранного критерия, зачастую — ребро с наименьшей

длиной. Вводится новая вершина, позволяющая построить на его основании новый треугольник, или же выбирается уже существующая в случае, если положение рассчитанной вершины не удовлетворяет требованиям, предъявляемым к сетке. После этого из фронта удаляется как текущее активное ребро, так и все уже существующие стороны нового треугольника (если такие были), только добавленные стороны при этом добавляются во фронт.

Критерии для оценки качества добавляемого симплекса

Добавляемый новый треугольник (тетраэдр) должен пройти оценку качества. Для этого выделяются основные типы критериев, используемые для проверки соблюдения требуемых свойств сеток:

- на основе угловых величин;
- на основе определения геометрического пересечения между объектами;
- на основе расчета расстояний между объектами.

Критерии на основе угловых величин обычно используются для формирования более качественных треугольных объектов [4, 5, 6, 7], но могут быть использованы для того, чтобы отбросить лишние точки из перебора кандидатов [8, 9] еще до использования более тяжеловесных функций определения пересечения между геометрическими объектами. Для текущего активного ребра *AB* треугольник-кандидат *ABC* не рассматривается, если существует ребро *AD*, смежное *AB*, такое, что $\cos(\angle BAC) < \cos(\angle BAD)$. Это позволяет избегать конфигураций фронта, при которых луч *AC* хотя бы в некоторой окрестности точки *A* оказывается внутри фронта. При этом повторяется такая проверка для обеих точек активного ребра *AB* и для каждого смежного ему ребра. Та же идея применима и к случаю тетраэдризации, когда активной сущностью является грань. В этом случае нужно сравнивать двугранные углы, которые получаются между активной и смежными ей гранями по каждой стороне.

Следующим типом метрик оценивания являются критерии на основе определения геометрического пересечения. На текущем шаге итерации рассматривается пересечение между рассматриваемым треугольником (тетраэдром) с сущностями активного фронта (прямолинейными ребрами в виде отрезков при триангуляции, треугольными гранями при тетраэдризации). В случае триангуляции пересечение можно искать напрямую с помощью пар сущностей «треугольник-отрезок», или же в некоторых ситуациях с помощью пары «отрезок-отрезок». При тетраэдризации, учитывая, что у тетраэдра треугольные грани, отношение «тетраэдр-треугольник» для простоты может быть сведено к паре «треугольник-треугольник», которая, в свою очередь, с некоторыми дополнениями может быть трансформирована в пару «треугольник-отрезок». Важными вспомогательными операциями являются проверки принадлежности точки треугольнику и тетраэдру; наиболее известными методами определения принадлежности являются метод на основе барицентрических координат [10, 11], а также методы на основе полуплоскостей и сравнения площадей (объемов) [12, 13].

В двумерном случае при определении пересечения между парой «отрезок-отрезок» сначала анализируется, с какой стороны каждого из отрезков располагаются концы другого; если хотя бы для одного отрезка с одной, то пересечения нет. В ином случае линейные сегменты представляются в виде уравнения сегмента $S_i(t_i) = A_i + t_i d_i$, $d_i = (B_i - A_i)$,, где A_i, B_i — начало и конец отрезка, $t_i \in [0;1]$ — параметр, задающий точку на этом отрезке ($t_A = 0$, $t_B = 1$). Приравняв два уравнения, получаем параметры t_1 и t_2, задающие положение точки пересечения на первом и втором линейных сегментах соответственно. При этом точка пересечения лежит в пределах каждого из отрезков, т.е. $t_1, t_2 \in [0;1]$.. Если пересечения нет, то задача может быть трансформирована к поиску кратчайшего расстояния между отрезками. При этом для её решения недостаточно просто посчитать расстояние между соответствующими концами двух отрезков, а следует произвести дополнительные операции [13]. В частности, если точка пересечения принадлежит только второму отрезку, то параметрическое значение ближайшей точки на первом отрезке находится в виде $t_1 = (A_2 + t_2 d_2 - A_1) \cdot d_1 \,/\, d_1 \cdot d_1$.

В случае трехмерного пространства задача первоначально формулируется в терминах нахождения кратчайшего расстояния, поскольку в 3D отрезки, как и линии, в общем случае не пересекаются. При этом сама идея решения остается прежней.

Следующей является пара «треугольник-отрезок». Определение пересечения является достаточно важным для задач рейтрейсинга, поэтому зачастую ищется пересечения именно луча, заданного точкой-источником и направлением, и треугольника [14, 15, 16], хотя существуют и алгоритмы для исходной постановки задачи [17, 18].

В работе [14] авторы фокусируются на алгоритме, не требующем предварительного расчета нормали плоскости треугольника. Нахождение точки пересечения происходит с помощью системы $O + tD = (1 - u - v)p_0 + up_1 + vp_2$, где слева представлено уравнение луча с источником в точке *O* и направлением *D*, *t* — параметр, определяющий положение точки на нем, а справа — уравнение треугольника с помощью барицентрических координат, которые позволяют описать любую точку внутри него. При этом не рассматрива-

ется случай, когда треугольник и луч лежат в одной или параллельных плоскостях.

В статье [15] точка пересечения также представляется с помощью барицентрических координат треугольника, при этом предварительно рассчитываются 3 дополнительные плоскости. Для более быстрого расчета используются параллельные инструкции из набора команд SSE4. В работе [16] сохраняются в памяти нормали к плоскости треугольника, при этом производятся преобразования треугольника таким образом, чтобы каждая точка могла быть представлена в двумерных координатах. Кроме того, добавляются дополнительные тесты, такие, как пересечение луча с ограничивающим пространством треугольника, луча и описывающей сферы, луча и описанной вокруг треугольника окружности и т.д.

Алгоритм [17] определяет наличие пересечения с помощью знаков тетраэдров, получаемых соединением вершин треугольника и отрезка, но не позволяет в своей исходной форме найти саму точку пересечения. В статье [18] совмещаются идеи, связанные с определением знаков тетраэдров и нахождением точек внутри них с помощью барицентрических координат, что позволяет, при необходимости, определить точку пересечения, и все необходимые для этого вычисления производятся в конце.

Последней парой сущностей является пара «треугольник–треугольник». В работе [19] описывается алгоритм, в ходе которого вычисляются плоскости, в которых лежат треугольники. В случае, если все точки треугольника лежат с одной стороны от плоскости другого, пересечений нет. Принадлежность каждой точки единой плоскости обрабатывается отдельно по аналогии двумерному случаю. В оставшейся ситуации пересечение между плоскостями треугольников образует прямую линию, после чего она проецируется на одну из осей. Рассчитанные интервалы для каждого треугольника на этой прямой пересекаются, что позволяет найти сегмент пересечения между треугольниками. В статье [20] описывается определение пересечений между плоскостями треугольников на основе специальных предикатов, определителей матриц, при этом алгоритм не подразумевает нахождение сегмента пересечения, а только сам факт его наличия или отсутствия. Описанный в [21] алгоритм оптимизирует вышеописанные подходы за счет уменьшения количества используемых операций и стоимости нахождения сегмента пересечения. В работе [22] оптимизируется один из шагов алгоритма [19] за счет использования дополнительной проекции на плоскость.

В статье [23] авторами предлагается подход, позволяющий классифицировать найденное пересечение (точка, отрезок или площадь) с помощью решения систем неравенств вида $m \le ax + by \le n$.. При этом классификация происходит скорее с помощью логических тестов, нежели геометрических.

Заключительной является метрика расстояния между объектами. Помимо расстояния между точками и кратчайшего расстояния между отрезками и линиями в трехмерном пространстве, нахождение которого описано выше, можно выделить также расстояние от точки до отрезка, треугольника и тетраэдра.

Для определения расстояния между точкой и прямой линией необходимо найти расстояние между точкой и её проекцией на эту прямую. Если прямая линия задана точкам *A* и *B*, то параметр, соответствующий проекции точки *C* на прямую может быть найден в виде $t' = (C - A) \cdot n \,/\, \|B - A\|$, где $n = (B - A) \,/\, \|B - A\|$ есть единичный вектор в направлении AB [13]. При решении задачи о поиске расстояния между точкой и отрезком нужно определить, какое значение принимает параметр t'; если $t' \in [0;1]$, то расстояние ищется с этой же точкой, в ином случае нужно искать расстояние с ближайшим к найденной проекции концом отрезка *AB*.

В [13] предлагается эффективный подход для нахождения кратчайшего расстояния от точки *P* до треугольника *ABC* на основе диаграммы Вороного. Диаграмма разбивается на 6 регионов, по одной вокруг каждой вершины треугольника и его сторон. Чтобы проверить, лежит ли ближайшая точка в области позади вершины *A*, анализируются скалярные произведения $d_1 = \left(\overrightarrow{AB} \cdot \overrightarrow{AP}\right)$, $d_2 = \left(\overrightarrow{AC} \cdot \overrightarrow{AP}\right)$. Если d_1 и d_2 не являются положительными величинами, то ближайшая точка к *P* в треугольнике *ABC* есть вершина *A*. Ближайшая точка лежит на ребре *AB*, если выполняется каждое из условий $vc \le 0$, $d_1 \ge 0$, $d_3 \le 0$, где $vc = d_1 * d_4 - d_3 * d_2$, $d_3 = \left(\overrightarrow{AB} \cdot \overrightarrow{BP}\right)$, $d_4 = \left(\overrightarrow{AC} \cdot \overrightarrow{BP}\right)$; кратчайшее расстояние в таком случае есть расстояние от точки *P* до её проекции на ребро *AB*. Для других вершин и ребер вычисления производятся схожим образом. В случае, если точка не принадлежит ни одному из регионов, считается ортогональная проекция точки *P* на треугольник *ABC* с помощью барицентрических координат.

Поиск расстояния от точки до тетраэдра, в случае, если точка не лежит внутри него, может решаться нахождением минимального расстояния от точки до каждой треугольной грани тетраэдра, при этом также может применяться идея с диаграммой Вороного, которая будет включать уже 14 областей [13].

Используемые в генераторе правила

Геометрическая модель поступает на вход программы в формате граничного представления, работа с ней

ведется через ядро OpenCascade [1]. OpenCascade позволяет работать с ребрами и гранями модели в параметрическом представлении кривой и поверхности соответственно. Пользователем задается идеальный шаг сетки h, а также максимально допустимое расстояние между соседними точками e. В программе определяется единый обход контура, т.е. определено, в какую сторону от текущей активной сущности должны строиться новые.

Генератор работает со структурами данных, используемыми в MARPLE3D [3]. Каждая добавляемая сущность получает уникальный индекс, отличающий её от уже существующих. Для определения близких точек и борьбы с дубликатами используется специальная структура на основе двоичного дерева поиска, позволяющая для каждого узла в декартовых координатах получать хэш-индекс. В памяти хранятся отношения инцидентности и смежности, позволяющие, например, получить номера вершин ребра по его индексу или ребра, смежные по узлу соответственно.

При триангуляции проверка качества сетки происходит как в декартовых, так и параметрических координатах. Для соблюдения угловых метрик для текущего активного ребра обрабатываются ему смежные по обеим его вершинам; при этом метрики рассчитываются только для ребер, второй конец который лежит с нужной стороны от текущего активного (определяется знаком косого произведения в параметрическом пространстве).

Допустимость построения нового треугольника проверяется с помощью параметрических координат и сущностей «отрезок–отрезок», что позволяет ускорить процесс проверки геометрических пересечений. Поскольку на данном этапе идет работа с прямолинейными ребрами, то в параметрическом представлении они представлены в виде прямолинейных ребер. В полученном двумерном пространстве треугольник и отрезок фронта пересекаются тогда и только тогда, когда отрезок фронта пересекает одну из сторон треугольника, при этом текущее активное ребро уже находится во фронте, а значит, прошло все проверки. Таким образом, необходимо только проверить пересечение между двумя другими сторонами треугольника и близкими ребрами из активного фронта. В то же время, из этих ребер отбираются только не имеющие общих вершин с текущим активным. Помимо этого, проверяется, что ни одна из вершин фронта не лежит внутри рассматриваемого треугольника.

В качестве метрик расстояния используются расстояния от узлов фронта до ребер и граней рассматриваемого треугольника; рассматриваются только точки, не являющиеся вершинами треугольника. Минимально допустимое расстояние выбирается в диапазоне $0.15h$–$0.25h$. Показатель выбран эмпирически для регулирования качества сетки.

При тетраэдризации в качестве угловых метрик используются двугранные углы между смежными с текущей активной гранями, при этом для каждого ребра грани метрика рассчитывается отдельно. Идет перебор только среди таких граней, третья вершина которых находится в положительной полуплоскости относительно текущей активной с помощью одной из вершин треугольника и его нормали.

Геометрическое пересечение рассматриваемого тетраэдра с активным фронтом треугольных граней происходит с помощью пары сущностей «треугольник-отрезок». Алгоритмы «отрезок-отрезок» не позволяют определить пересечения внутри области, в то же время алгоритмы вида «треугольник-треугольник» имеют значительные трудности при определении всего сегмента пересечения в случае, например, если треугольники имеют общую сторону. В общем случае, тетраэдр *ABCD* проходит проверку с треугольником *PQR* из активного фронта, если соблюдаются все следующие условия:

- Грани тетраэдра *ABCD* не пересекаются с ребрами треугольника *PQR*;
- Треугольник *PQR* не пересекается с ребрами тетраэдра *ABCD*;
- Ни одна из точек треугольника *PQR* не лежит внутри тетраэдра *ABCD*.

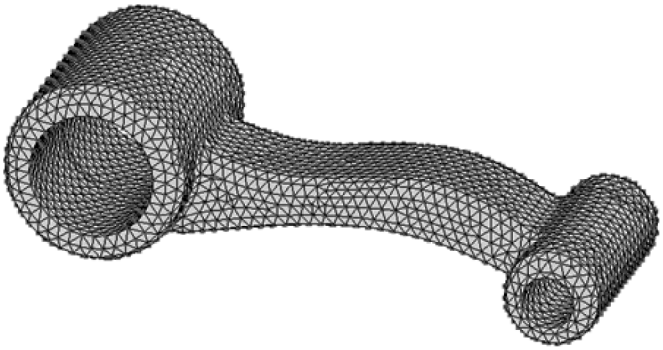

Рис. 1. Построенная сетка на модели linkrods

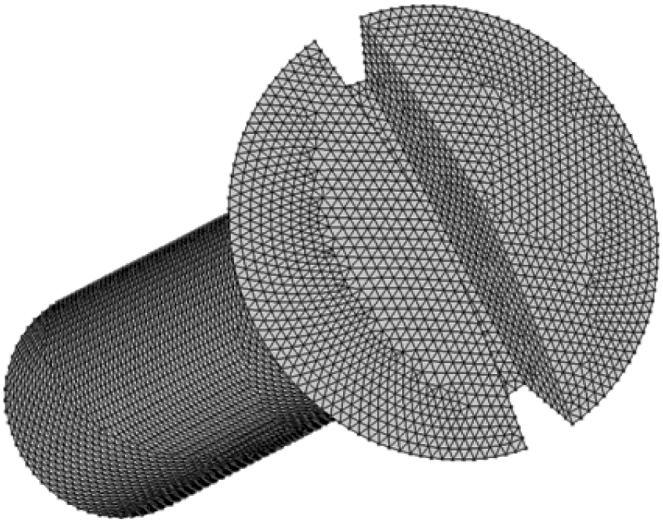

Рис. 2. Построенная сетка на модели screw

Для определения пересечения в настоящий момент используется описанный в [14] алгоритм (как самый проверенный и надежный) с некоторыми дополнениями. В частности, если отрезок является ребром треугольника, то это считается допустимым пересечением. Если у треугольника и отрезка одна вершина, то источник луча перемещается во второй конец ребра, и единственное допустимое пересечение между треугольником и ребром есть общая вершина. В случае, если общих вершин нет, в качестве источника выбирается любая вершина, и допустимых общих точек пересечения нет.

В качестве метрик расстояния используются расстояние между ребрами тетраэдра и ребрами активных граней фронта, расстояние от новой вершины тетраэдра до активных граней фронта, а также расстояние от узла фронта до рассматриваемого тетраэдра.

Результат работы последовательной версии генератора с используемыми правилами представлен на моделях linkrods и screw, распространяемыми вместе с ядром OpenCascade.

Заключение

В ходе исследования были сформулированы основные критерии, используемые для оценки качества очередного симплекса, состояния расчетной сетки в момент её построения и после. На основе проведенных исследований был сформирован и обоснован набор правил, используемых в разработке последовательной и параллельной версиях генератора симплициальных сеток.

ЛИТЕРАТУРА

1. Дронов А.Г. Об одном алгоритме генерации поверхностных сеток // Современная наука: актуальные проблемы теории и практики. Серия «Естественные и технические науки». — 2024. — №6. — С. 74–79.
2. Дронов А.Г., Болдарев А.С. О разработке параллельных алгоритмов построения расчетных сеток на системах с распределенной памятью // Перспективы науки. — 2024. — №6 (177). — С. 27–32.
3. MARPLE: программное обеспечение для мультифизического моделирования в задачах сплошных сред / В.А. Гасилов [и др.] // Препринты ИПМ им. М.В.Келдыша. 2023. № 37. 40 с. Режим доступа: https://doi.org/10.20948/prepr-2023-37, https://library.keldysh.ru/preprint.asp?id=2023-37.
4. Löhner, R. Progress in grid generation via the advancing front technique // Engineering with Computers. — 1996. — Vol. 12. — P. 186–210. https://link.springer.com/article/10.1007/BF01198734
5. Advancing front surface mesh generation in parametric space using a Riemannian surface definition / J.R. Tristano, S.J. Owen, S.A. Canann // In: International Meshing Roundtable. — 1998. — P. 429–445.
6. Shewchuk J.R. Delaunay refinement algorithms for triangular mesh generation // Computational geometry. — 2002. — Vol. 22. — P. 21–74. doi: 10.1016/S0925-7721(01)00047-5.
7. Mohammadi F., Shontz S.M. A direct method of generating quadratic curvilinear tetrahedral meshes using an advancing front approach // In: Proceedings of the 29th International Meshing Roundtable. — 2021. — P. 74–91. doi: 10.5281/zenodo.5559211.
8. Löhner, R. Recent advances in unstructured mesh and point generation // In: 1st Conference on Advances and Applications of GiD. — 20-22 February, Barcelona, Spain, 2002. — P. 5–22.
9. Lau T.S., Lo S.H. Finite element mesh generation over analytical curved surfaces // Computers & Structures. — 1996. — Vol. 59(2). — P. 301–309. doi:10.1016/0045-7949(95)00261-8.
10. Floater, M.S. Generalized barycentric coordinates and applications. // Acta Numerica. — Cambridge University Press, 2015. — Vol. 24. — P. 161–214. doi:10.1017/s0962492914000129.
11. Marschner S., Shirley P. Fundamentals of computer graphics, 5th edition — CRC Press, 2021. — 700 p.
12. Haines E. Point in polygon strategies // Graphics Gems IV. / Edited by P. Heckbert. — Morgan Kaufmann, 1994. — P. 24–46.
13. Ericson C. Real-time collision detection. — CRC Press, 2004. — 632 p.
14. Möller T., Trumbore B. Fast, Minimum Storage Ray-Triangle Intersection // Journal of Graphics Tools. — 1997. — Vol: 2. — P. 21–28. doi: 10.1080/10867651.1997.10487468.
15. Havel J., Herout A. Yet Faster Ray-Triangle Intersection (Using SSE4) // In: IEEE Transactions on Visualization and Computer Graphics — 2010. — Vol: 16(3). — P. 434–438. doi: 10.1109/TVCG.2009.73.
16. Pichler T.A. Fast CPU Ray-Triangle Intersection Method // Diplom-Ingenieur in Media Informatics — 2018.
17. Segura R.J., Feito F.R. Algorithms to test ray-triangle Intersection. Comparative study // Journal of WSCG — 2001. — Vol. 9(3). — P. 76–81.
18. A robust segment/triangle intersection algorithm for interference tests. Efficiency study / Juan J. Jiménez, Rafael J. Segura, Francisco R. Feito // Computational Geometry: Theory and Applications — Elsevier, 2010. — Vol: 43(5). — P. 474–492. doi: 10.1016/j.comgeo.2009.10.001.
19. Möller T. A Fast Triangle-Triangle Intersection Test// In: Journal of Graphics Tools. — 1997. — Vol: 2(2). — P. 25–30. doi: 10.1080/10867651.1997.10487472.
20. Guigue P., Devillers O. Fast and robust triangle-triangle overlap test using orientation predicates // Journals of graphics tools — 2003. — Vol. 8(1). — P. 25–42.
21. A fast triangle to triangle intersection test for collision detection / O. Tropp, A. Tal, I. Shimshoni // Computer Animation and Virtual Worlds — Wiley, 2006. — Vol: 17(5). — P. 527–535. doi: 10.1002/cav.115.
22. An improved algorithm for triangle to triangle intersection test / X. Ye et al. // In: 2015 IEEE International Conference on Information and Automation (ICIA). — IEEE, 2015. — P. 2689–2694. doi: 10.1109/ICInfA.2015.7279740.
23. Sabharwal C.L., Leopold J. L. An Implementation of Triangle-Triangle Intersection for Qualitative Spatial Reasoning // In: Seventh International Conference on Wireless & Mobile Network. — 2015. doi: 10.5121/csit.2015.51003.

Журнал «Современная наука: актуальные проблемы теории и практики»