

МИЭТ

Работа допущена к защите
зав. кафедрой

Прокофьев Александр Александрович

«_____» _____ 2018 г.

ВЫПУСКНАЯ РАБОТА БАКАЛАВРА

Тема: **Упрощение топологии объектов на цифровых
географических картах**

Направление: 01.03.04 – Прикладная математика

Выполнил студент гр. МП-40

_____ Березин Александр Андреевич

Научный руководитель,

ст. преподаватель

_____ Назаров Максим Николаевич

Москва – 2018

Оглавление

1.	Введение	3
2.	Постановка задачи	3
3.	Обзор предметной области	4
3.1.	Существующие решения	4
3.2.	Теоретические сведения	4
4.	Общая методология	4
4.1.	Требование 1	4
4.2.	Требование 2	6
4.3.	Требования 3 и 4	7
4.4.	Проверка допустимости соединений	8
5.	Реализация алгоритма	11
5.1.	Интерфейс	11
5.2.	Объектная модель	13
5.3.	Пример работы	14
6.	Заключение	14
	Список литературы	17

1. Введение

В работе описывается решение задачи преобразования многосвязных геометрических областей в односвязные в контексте кодирования географических карт. Полученный алгоритм работает в общем случае и может использоваться в других предметных областях.

Речь, конечно, идет о схематических картах, а не о спутниковых фотографиях. Такие карты кодируются в векторном формате. Но конкретная форма представления данных может отличаться. По условию поставленной задачи имеются две системы с различными представлениями. Одна из них поддерживает в составе карты объекты любой топологии, другая – только односвязные. Требуется разработать алгоритм для преобразования данных между этими системами.

2. Постановка задачи

По условию поставленной задачи имеются две геоинформационные системы, которые используют различные представления данных. Первая поддерживает в составе карты объекты любого размера и топологии. Вторая поддерживает только односвязные, а также имеет ограничение на количество точек в одном объекте. Понятно, что первая система автоматически поймет данные второй, но обратное неверно. Задача состоит в том, чтобы разработать алгоритм для преобразования данных из первой системы во вторую.

В качестве входных данных имеется географическая карта, закодированная по стандарту Geography Markup Language (GML). Этот стандарт предназначен для кодирования областей различной природы – к примеру, лесных массивов, водоемов или островов – в виде вложенных многоугольников при помощи XML-грамматики [1].

Требуется преобразовать входные данные в новый GML-файл таким обра-

зом, чтобы одновременно выполнялись следующие условия:

1. Отсутствуют вложенные области, т.е. все области односвязны.
2. Число точек, задающих границу каждой области, не превышает заданной константы N .
3. Количество добавочных соединений и их длины минимальны.
4. Создание новых вершин запрещено.

Возможность работы в реальном времени не требуется, поэтому оптимизация алгоритма в данной работе не затрагивается.

3. Обзор предметной области

3.1. Существующие решения

3.2. Теоретические сведения

4. Общая методология

4.1. Требование 1

Формулировка требования: «Отсутствуют вложенные области, т.е. все области односвязны».

Многосвязная область состоит из внешней границы и произвольного числа внутренних границ, заданных многоугольниками. Задача снижения порядка связности области сводится к выбору соединений точек различных границ таким образом, чтобы все они оказались связаны. В простейшем случае с одной внутренней границей подойдет любой отрезок, один из концов которого принадлежит множеству вершин внешней границы, а другой — внутренней, и который не имеет с границами других общих точек. В случае трехсвязной области (с

двумя внутренними границами) можно также соединить внутренние границы, а затем одну из них — со внешней.

Чтобы найти требуемые соединения, преобразуем объектную модель карты следующим образом. Поставим в соответствие каждому многоугольнику, включая внешнюю границу, вершину графа и сделаем полученный граф полным. В полученной графовой модели ребра соответствуют соединениям между многоугольниками. В качестве весов присвоим им длины кратчайших соединений между соответствующими многоугольниками. Далее, нужно исключить из модели недопустимые соединения — такие, которые имеют более двух общих точек с многоугольниками. Для этого будем проверять соединение на пересечение со всеми сторонами многоугольников в рассматриваемой области. Наконец, к полученному графу применим алгоритм Прима [2] для выделения минимального остовного дерева.

Объектная модель

Графовая модель

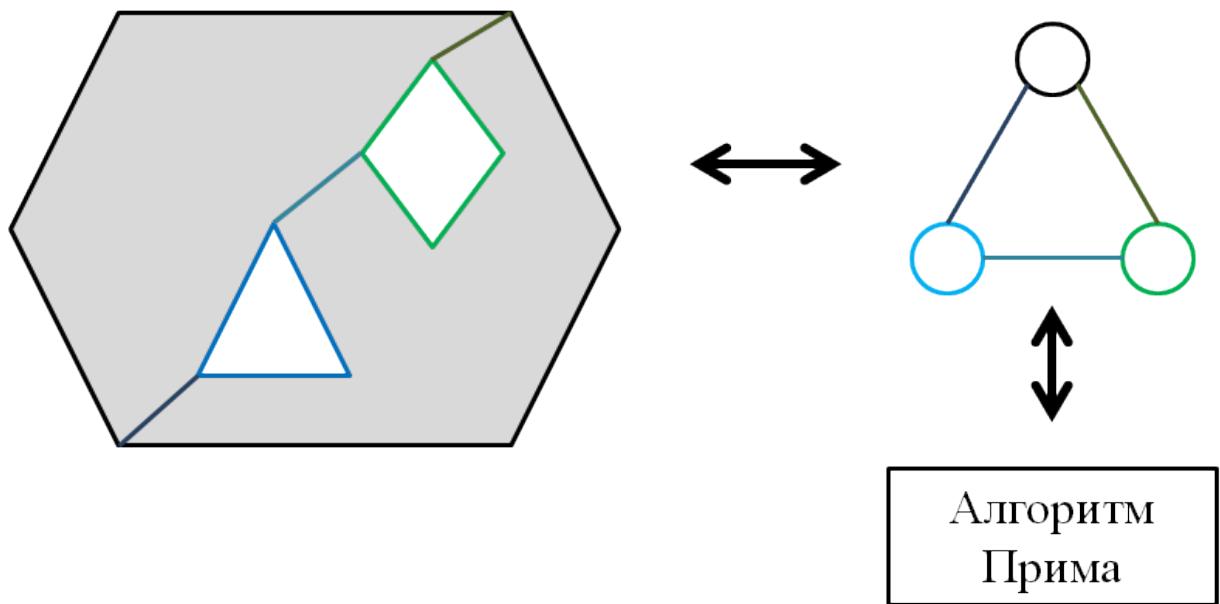


Рис. 1. Использование алгоритма Прима для удовлетворения требованию 1

Восстановление объектной модели из графовой происходит следующим образом. На внешней границе произвольно выбирается начальная точка. Если с

ней не связано ни одно из соединений в графовой модели, происходит переход к следующей вершине в том же многоугольнике; в противном случае эта вершина сохраняется в памяти, а следующей выбирается точка, находящаяся на другом конце найденного соединения, для которой та же процедура повторяется рекурсивно. Когда процесс возвращается в исходную точку, он возвращается по тому же соединению к предыдущему многоугольнику. Алгоритм завершает работу, когда возвращаться становится некуда. В результате вместо многосвязной области получается односвязная, в которой соединения становятся вырожденными туннелями.

4.2. Требование 2

Формулировка требования: «Число точек, задающих границу каждой области, не превышает заданной константы N ». На данном этапе все области уже односвязны.

Рассмотрим вначале область, задаваемую выпуклым многоугольником с более чем N вершинами. Из определения выпуклости очевидно, что ее можно разделить в произвольном отношении, т.к. любая линия, соединяющая несмежные точки, будет лежать внутри области. С учетом требования 3 следует выбрать кратчайшую линию, разделяющую многоугольник таким образом, чтобы в одной из полученных частей было ровно N вершин. Если во второй части их больше N , повторяем ту же процедуру для нее рекурсивно.

Отметим, что при разбиении выпуклого многоугольника получаются также выпуклые многоугольники, поскольку все вершины полученных многоугольников лежат по одну и ту же сторону от новой грани.

Рассмотрим теперь невыпуклые многоугольники. Согласно [3], любой многоугольник можно разбить на смежные треугольники. Очевидно, эти треугольники можно объединять по смежным сторонам, получая таким образом разбивку на части с произвольным количеством вершин. Следовательно, существует и такая разбивка, при которой одна из частей имеет ровно N вершин. Значит,

описанный алгоритм можно распространить и на невыпуклые области.

4.3. Требования 3 и 4

Формулировка требования: «Количество добавочных соединений и их длины минимальны». В процессе разработки к этому условию было добавлено важное уточнение: если минимизация обоих параметров одновременно невозможна или невыполнима за полиномиальное время, приоритет имеет количество соединений.

Докажем, что после выполнения описанных выше шагов алгоритма это условие уже выполнено.

Вначале рассмотрим добавочные соединения, образовавшиеся в процессе упрощения топологии. Вспомним, что эти соединения представляют собой оствовное дерево в графе всевозможных соединений, причем за вес ребра в этом графе принимается длина соединения.

По свойству дерева [4], количество его ребер фиксировано: $B - P = 1$, где B — число вершин, P — число рёбер графа. Следовательно, удаление любого из оставшихся соединений приведет к тому, что объект карты перестанет быть односвязным. Значит, условие минимальности количества дополнительных соединений выполняется.

Для выбора минимальных соединений из графа всех возможных соединений использовался алгоритм Прима. Из его корректности [2] следует, что выбранное оствовное дерево графа является минимальным. Значит, условие минимальности длин дополнительных соединений выполняется. Таким образом, требование 3 удовлетворено.

Теперь рассмотрим добавочные соединения, образовавшиеся в процессе уменьшения числа вершин.

Пусть дан многоугольник с V вершинами. Тогда описанный выше алгоритм разобьет его на M частей, причем $(M - 1) * N < V + 2 * M \leq M * N$ (слагаемое $2M$ добавляется за счет того, что вершины, через которые прово-

дится разделяющая линия, дублируются в результирующих многоугольниках). Очевидно, что при уменьшении M на единицу количество вершин как минимум одного полученного после разбиения многоугольника будет превышать M ; следовательно, условие минимальности количества дополнительных соединений выполняется.

Одновременная минимизация добавочных соединений и их длины теоретически возможна, но также требует полиномиального времени выполнения. С учетом примечания к требованию 3, решено было ограничиться результатами разработанного алгоритма, который дает оптимальное или близкое к оптимальному решение в большинстве практических случаев.

Выполнение требования 4 очевидно, т.к. ни один из описанных выше алгоритмов не создавал новых вершин.

4.4. Проверка допустимости соединений

Выше упоминалась проверка проводимых линий на допустимость, т.е. на пересечение с уже существующими линиями. Эта процедура является важной частью алгоритма, т.к. даже одно пропущенное пересечение делает непригодным весь выходной файл.

Единственный способ убедиться в отсутствии недопустимых соединений — проверять каждое проводимое соединение на пересечение с каждой из линий, составляющих границы области. Для этого используется алгоритм, описанный в [5].

Этот алгоритм основывается на понятии ориентации. Ориентация упорядоченного списка из трех точек в зависимости от их взаимного расположения может принимать три значения:

- по часовой
- против часовой
- лежат на одной прямой

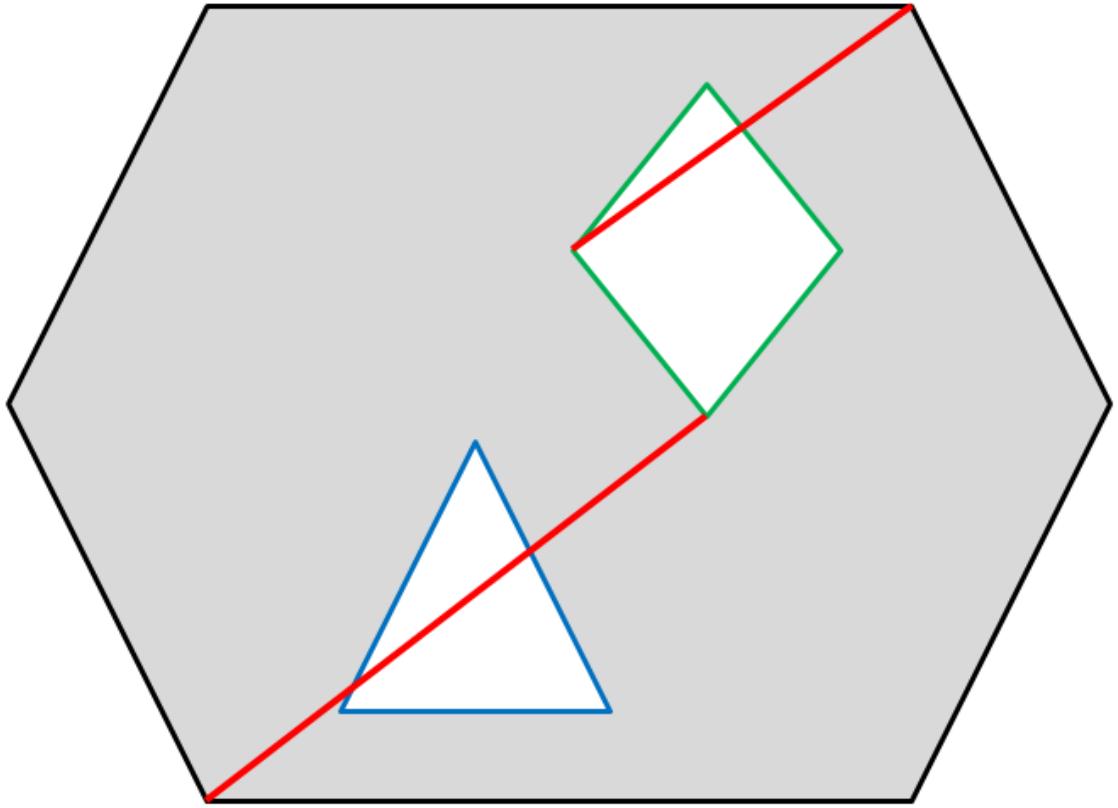


Рис. 2. Примеры недопустимых соединений (выделены красным)

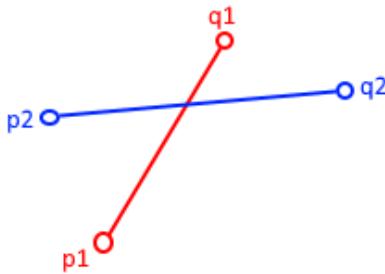
Два отрезка, (p_1, q_1) и (p_2, q_2) , пересекаются тогда и только тогда, когда верно одно из условий (см. рис. 3):

- (p_1, q_1, p_2) и (p_1, q_1, q_2) имеют разную ориентацию
- (p_2, q_2, p_1) и (p_2, q_2, q_1) имеют разную ориентацию

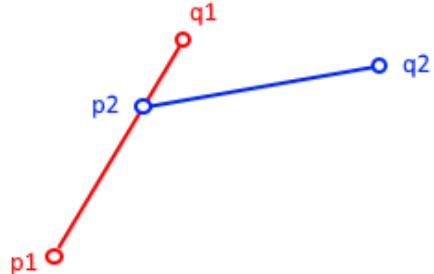
Отдельно следует рассмотреть случай коллинеарных отрезков. Они пересекаются, если одновременно выполняются условия (см. рис. 4):

- (p_1, q_1, p_2) , (p_1, q_1, q_2) , (p_2, q_2, p_1) , и (p_2, q_2, q_1) лежат на одной прямой
- проекции на ось x отрезков (p_1, q_1) и (p_2, q_2) пересекаются
- проекции на ось y отрезков (p_1, q_1) и (p_2, q_2) пересекаются

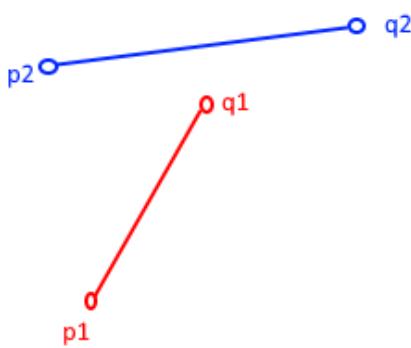
Докажем теперь, что проверять сами соединения на пересечения друг с другом не требуется. Для 4.2 утверждение очевидно: описанный алгоритм на каждом шаге исследует лишь одно соединение.



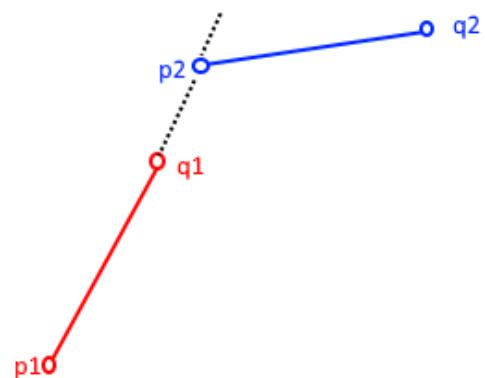
Example : Orientations of (p_1, q_1, p_2) and (p_1, q_1, q_2) are different. Orientations of (p_2, q_2, p_1) and (p_2, q_2, q_1) also differnet.



Example: Orientations of (p_1, q_1, p_2) and (p_1, q_1, q_2) are different. Orientations of (p_2, q_2, p_1) and (p_2, q_2, q_1) also different

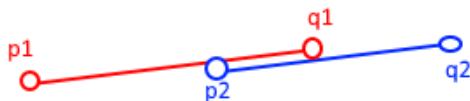


Example: Orientations of (p_1, q_1, p_2) and (p_1, q_1, q_2) are different. Orientations of (p_2, q_2, p_1) and (p_2, q_2, q_1) are same

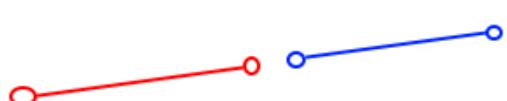


Example: Orientations of (p_1, q_1, p_2) and (p_1, q_1, q_2) are different. Orientations of (p_2, q_2, p_1) and (p_2, q_2, q_1) are same.

Рис. 3. Возможные ориентации отрезков в общем случае



Example: All points are collinear. The x-projections pf (p_1, q_1) and (p_2, q_2) intersect. The y- projection of (p_1, q_1) and (p_2, q_2) also intersect



Example: All points are collinear .The x-projections of (p_1, q_1) and (p_2, q_2) not intersect. The y-projection of (p_1, q_1) and (p_2, q_2) do not intersect

Рис. 4. Возможные ориентации коллинеарных отрезков

В 4.1 после построения полного графа допустимых соединений возникают пересечения. Однако, после выполнения алгоритма Прима они исчезнут. Почему? Предположим противное: в минимальном оствовном дереве имеются ребра, соответствующие пересекающимся соединениям. Поменяем концы этих соединений таким образом, чтобы не нарушить связность графа (это всегда можно

сделать, т.к. количество ребер графа не изменяется). Окажется, что длина каждого из полученных соединений меньше, чем у первоначальных. Но это противоречит корректности алгоритма Прима. Тем самым, утверждение доказано.

5. Реализация алгоритма

Требования, предъявленные к разрабатываемому исполняемому файлу:

1. Время исполнения для предоставленных файлов-примеров имеет порядок секунд или минут. Возможность работы в реальном времени не требуется.
2. Целевая платформа — Microsoft Windows. Доступны инструменты .NET Framework.
3. Графический интерфейс не требуется, т.к. взаимодействие программы с пользователем минимально.

Кроме того, для работы с большими массивами структур данных требуются инструменты высокого уровня. Исходя из этих требований, для реализации алгоритма был выбран язык C#: он полностью поддерживается платформой .NET Framework и обладает высокоуровневым инструментарием LINQ. В качестве среды разработки выбрана Microsoft Visual Studio 2013.

5.1. Интерфейс

Выбор файлов для обработки реализован при помощи вызова системного диалога Windows через класс OpenFileDialog .NET Framework. После этого вывод отладочной информации осуществляется через текстовый интерфейс.

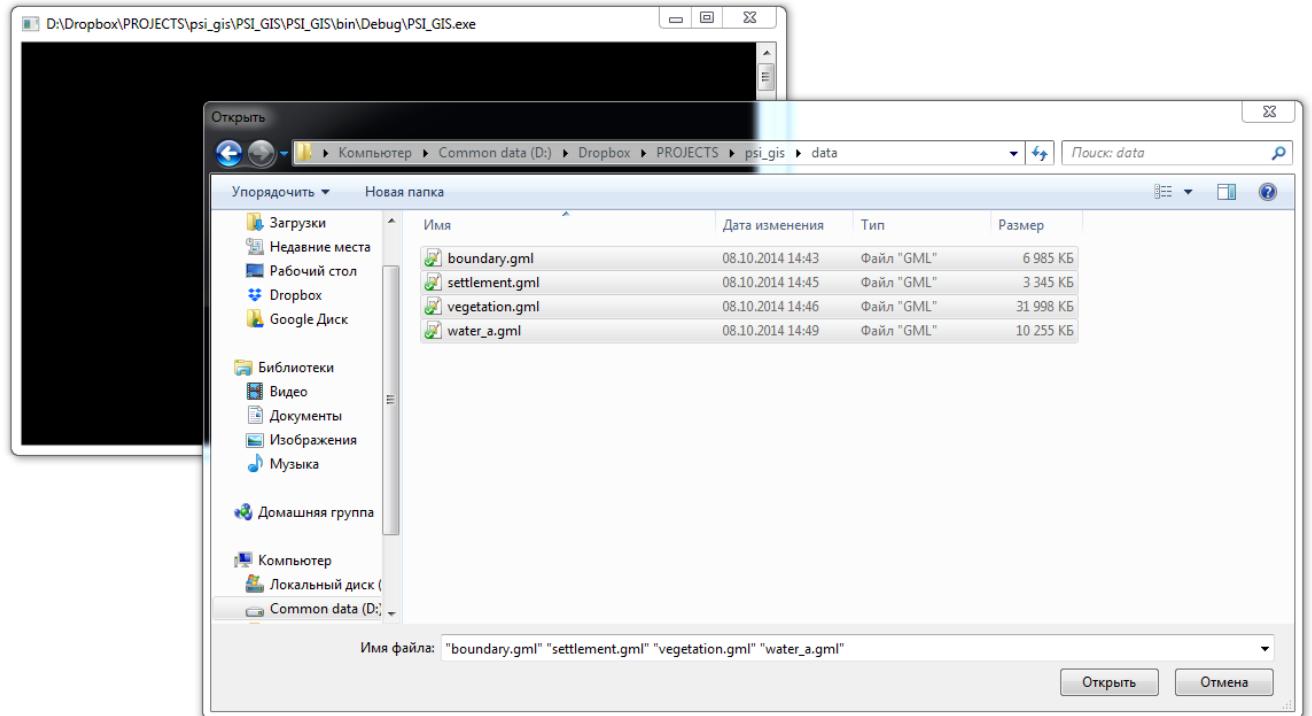
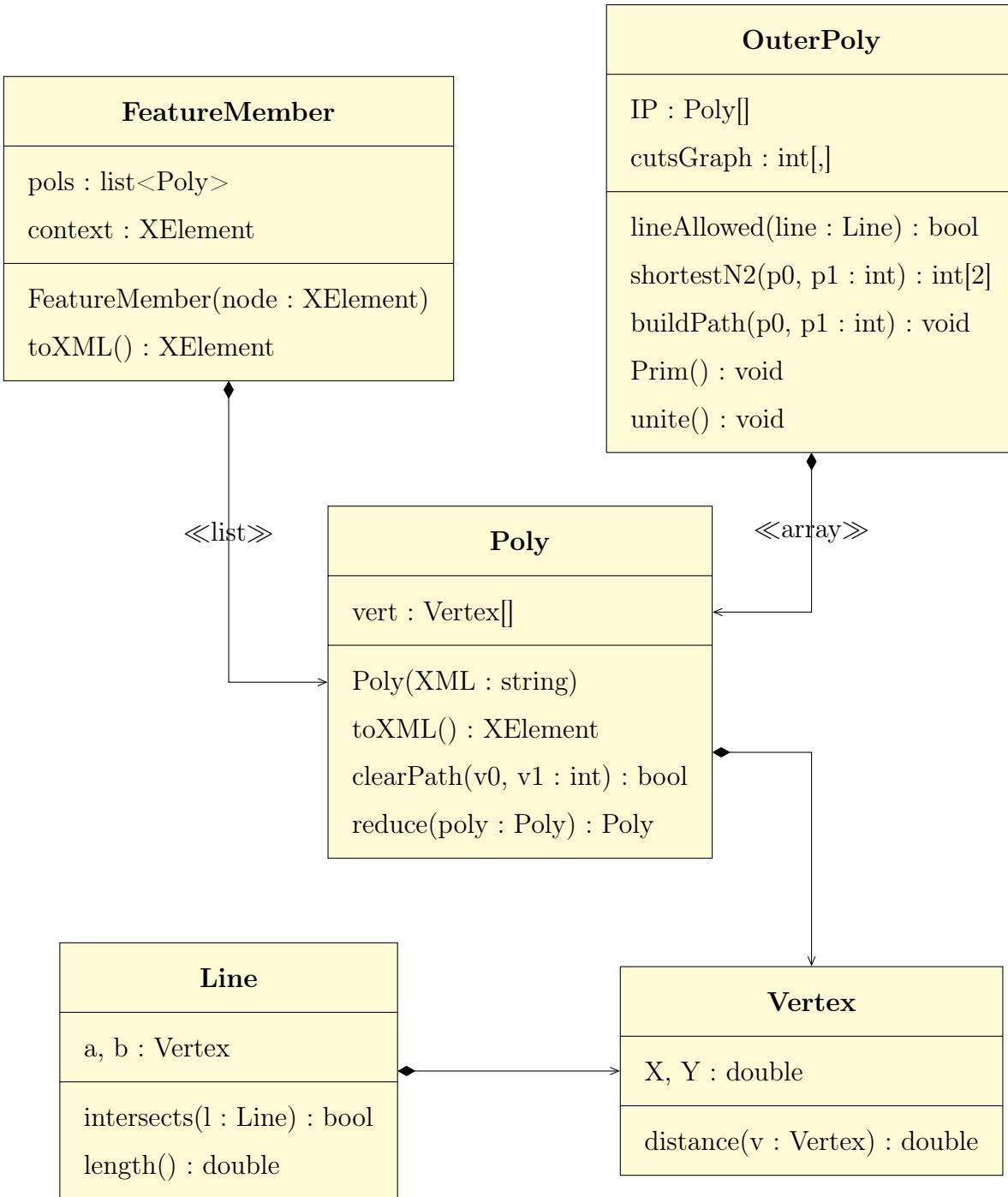


Рис. 5. Интерфейс ввода данных

5.2. Объектная модель



- **Vertex**: базовый класс, представляет точку на плоскости.
- **Line**: представляет отрезок. Состоит из двух объектов Vertex. Содержит методы для измерения длины и проверки пересечения с другим отрезком.
- **Poly**: Общее представление многоугольника. Состоит из объектов Vertex. Реализует методы прямого и обратного преобразования в GML-элемент

(Poly, toXML). Метод ClearPath проверяет, можно ли соединить две вершины многоугольника, не пересекая при этом его сторон. Метод reduce реализует алгоритм, описанный в [4.2](#).

- **OuterPoly:** Представляет многосвязную область. Состоит из внешней границы, представляемой объектом Poly, и массива многоугольников (IP), представляющих «дыры». Метод lineAllowed проверяет, можно ли провести внутри области отрезок, не пересекающий сторон ни одного из составляющих ее многоугольников. Метод shortestN2 определяет кратчайший отрезок, соединяющий два многоугольника внутри области. Метод unite реализует алгоритм, описанный в [4.1](#) при помощи методов buildPath для построения полного графа соединений и Prim для алгоритма Прима.
- **FeatureMember:** хранит метаданные областей, не имеющие отношения к их геометрии. Реализует ввод и вывод в GML.

5.3. Пример работы

См. рис. [6](#) и [7](#).

6. Заключение

В рамках данной работы была разработана программа для программно-технического комплекса (ПТК) Автоматизированной системы диспетчерского управления (АСДУ) электрическими сетями PSIcontrol по оптимизации геоинформационных полигональных (замкнутых) структур данных на основе математической обработки входных XML-файлов по следующим критериям:

1. оптимальное перекрытие островных структур на случай их наличия в глобальной структуре без взаимных пересечений с использованием вырожденных туннелей;



Рис. 6. Визуализация фрагмента входного файла

2. оптимальное разрезание полигонов, содержащих значительное количество точек, на более мелкие полигоны, количество точек которых не превышает N.

Результаты работы позволили снять ряд существенных ограничений по внедрению ГИС-системы ПТК АСДУ PSIcontrol, что позволило существенно (на два-три порядка) сократить время предварительной обработки геоинформационных подложек для ПТК.

Результаты работы внедрены в практическую деятельность Департамента энергетики ООО «ПСИ» и используются во всех текущих проектах компании по созданию и модернизации АСДУ на предприятиях российской электроэнергетики.

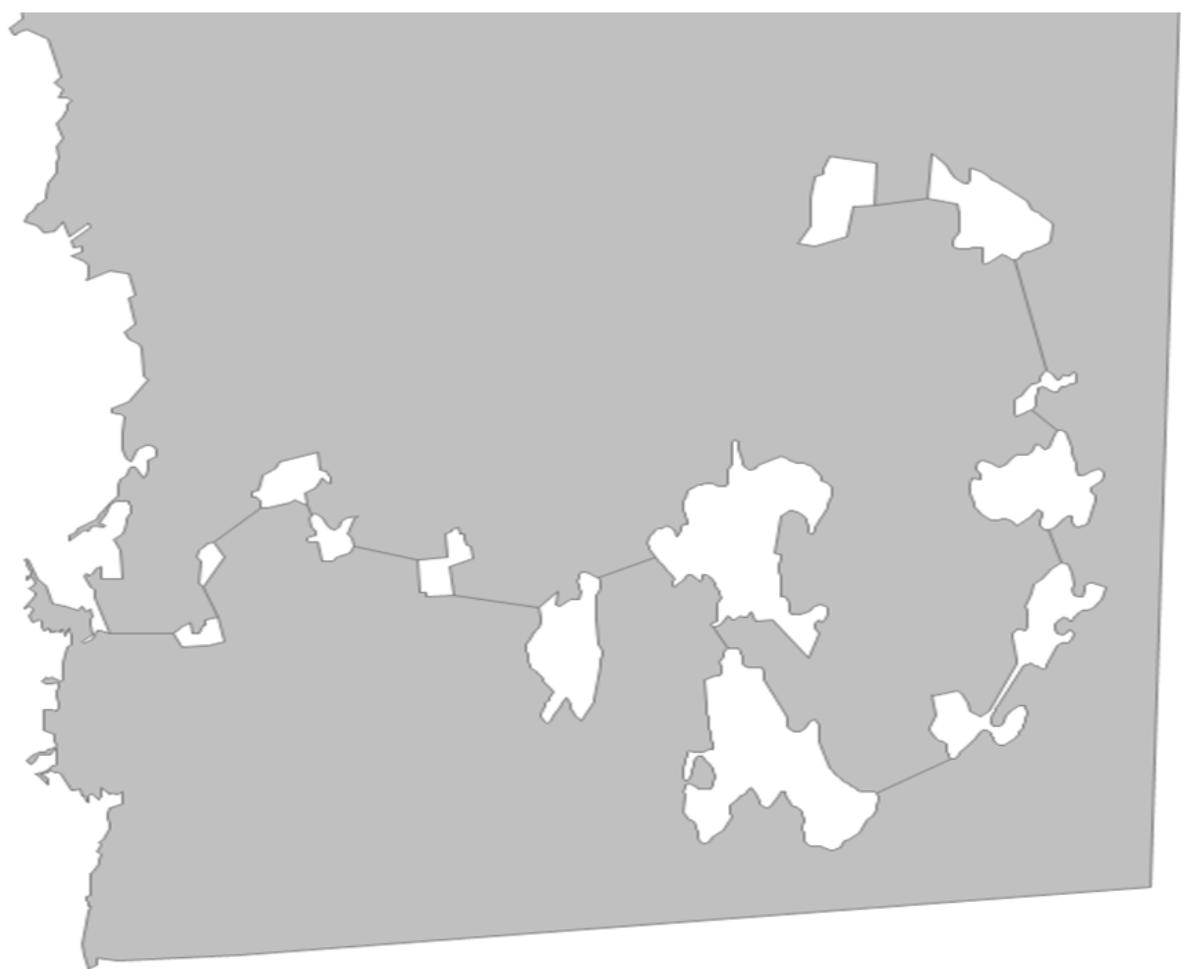


Рис. 7. Результат преобразования фрагмента файла

Список литературы

1. Geography Markup Language | OGC. — URL: <http://www.opengeospatial.org/standards/gml> (дата обр. 19.05.2018).
2. Prim R. C. Shortest connection networks And some generalizations // Bell System Technical Journal. — 1957. — 36 (6). — С. 1389—1401. — DOI: [10.1002/j.1538-7305.1957.tb01515.x](https://doi.org/10.1002/j.1538-7305.1957.tb01515.x).
3. Chapter 3: Polygon Triangulation / M. de Berg [и др.] // Computational Geometry (2nd revised ed.) — Springer-Verlag, 2000. — С. 45—61. — ISBN 3-540-65620-0.
4. Олейник Т. Деревья // Основы дискретной математики: теория и практика. — МИЭТ, 2010.
5. Introduction to algorithms. — Cambridge, Mass. : MIT Press, 2009. — ISBN 978-81-203-4007-7. — OCLC: 984411568.