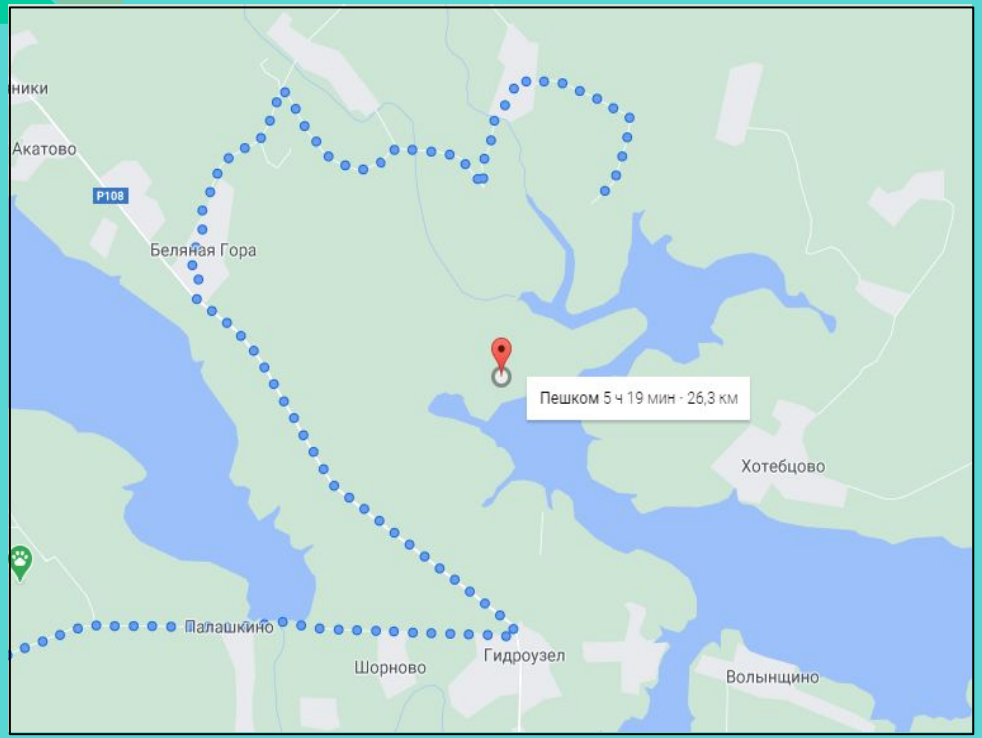
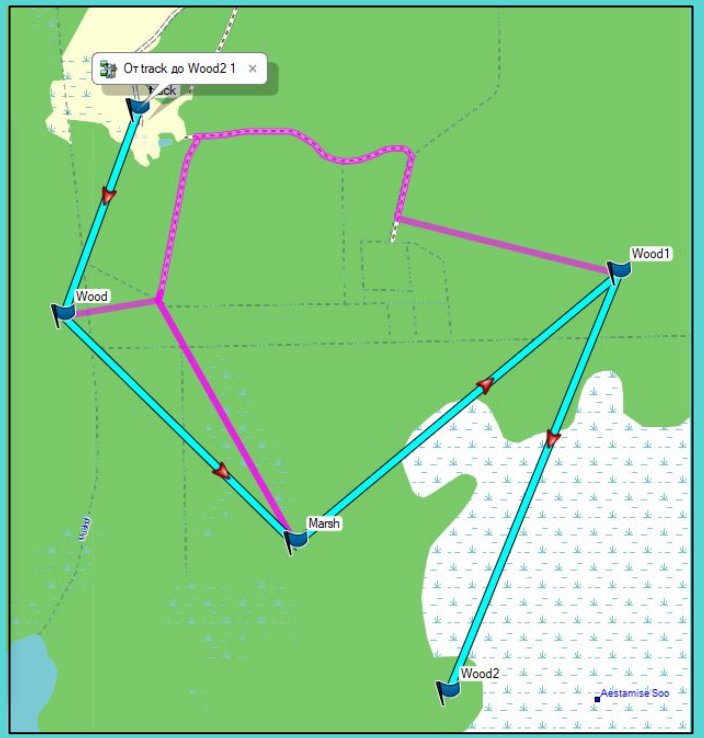
Методы и алгоритмы построения маршрутов на пересеченной местности

Козуб Денис

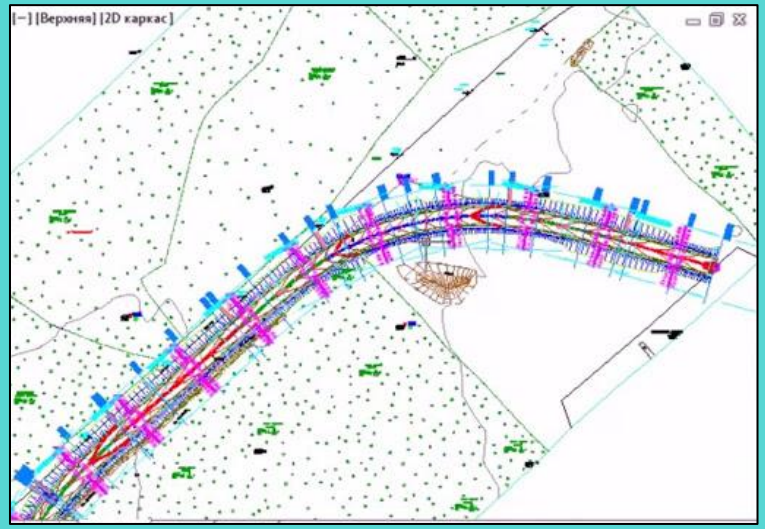
Как появилась идея?





Область применения

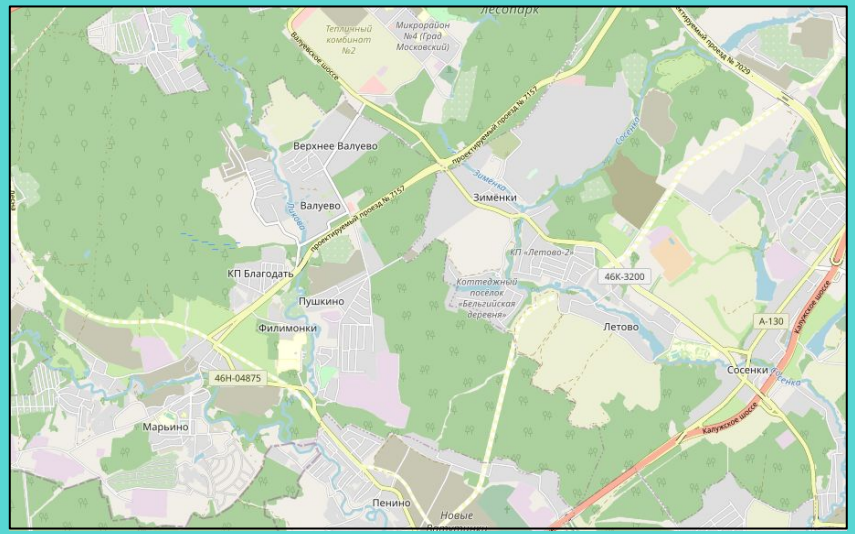
- Расширение функционала существующих систем
- Прокладывание дорог
- Планирование спасательных и военных операций
- Планирование туристических маршрутов



Основные требования:

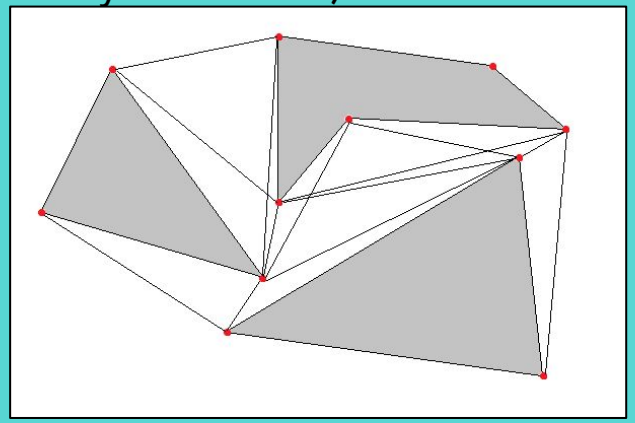
1) Скорость построения маршрута
2) Минимизация хранимых данных
3) Регулируемая точность маршрута
4) Динамическое изменение характеристик

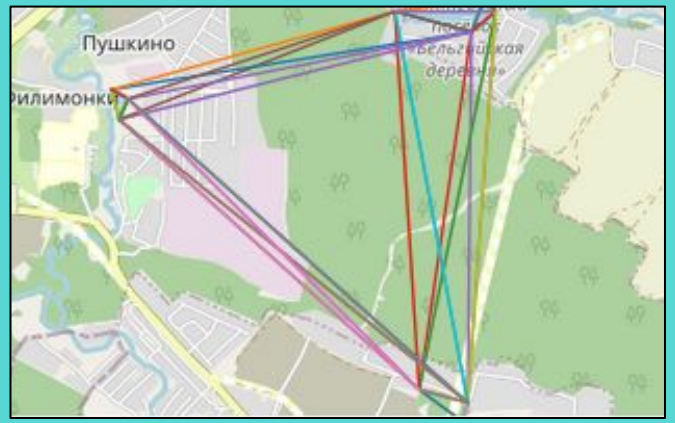
Дискретизация мира



Граф видимости

Граф видимости для множества многоугольников на плоскости - граф, в вершинах которого находятся вершины многоугольников, а ребра соединяют вершины, являющиеся видимыми друг для друга (никакое ребро графа видимости не пересекает ни один из данных многоугольников).





Определение веса ребра

- ➔ Различные типы поверхностей
- ➔ Погодные условия
- ➔ Время года
- ➔ Время суток

1) Составлена база данных туристических походов по пересеченной местности

2) Составлена база данных сельских дорог, проложенных через различные типы поверхностей

3) Используя полученную обучающую выборку, в будущем будут определены веса для каждого из типов поверхностей

Проблема скорости

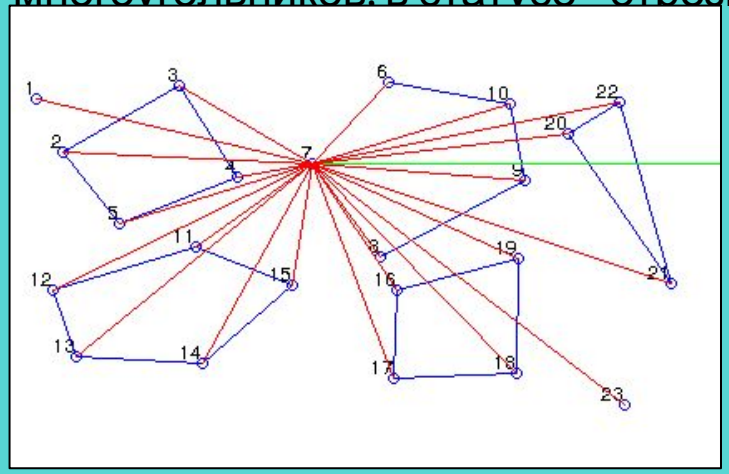
Время построения маршрута не должно зависеть от размеров области и количества объектов на ней. Иерархический подход:

- Построение маршрута до дороги -> маршрут по дороге -> маршрут до точки назначения
- Рассмотрение только элементов, размеры которых сопоставимы с размерами области, на которой строится маршрут

Вывод: не получится заранее построить граф видимости. Встает задача эффективной обработки массовых запросов на построение всех ребер видимости для данной точки q и множества многоугольников P = {Pi, i=1...h}. Это задача нахождения ребер графа видимости, инцидентных данной вершине без построения графа видимости для on-line алгоритма поиска пути.

Алгоритмическое решение

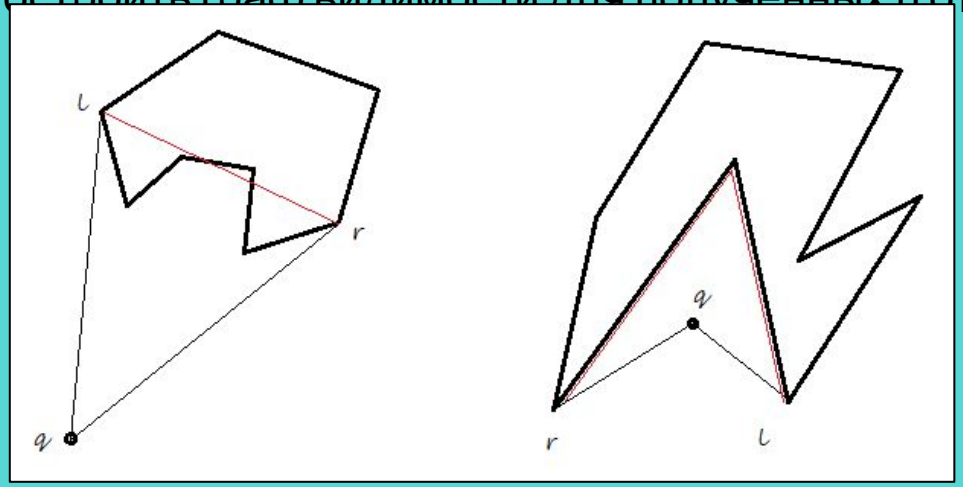
Пусть n = $\sum_{i=1}^{h}$ ni, ni - кол-во вершин в многоугольнике Pi. Тогда n - общее количество вершин. Алгоритм вращающейся заметающей прямой: O(n log n) В очереди событий находятся вершины многоугольников, в статусе - отрезки, которые пересекает заметающая прямая. Затраты памяти линейны. Проблемы: большая временная сложность, итоговый граф не оптимизирован. Ребра оптимизиро- ванного графа - опорные прямые к многоугольникам.



Алгоритмическое решение

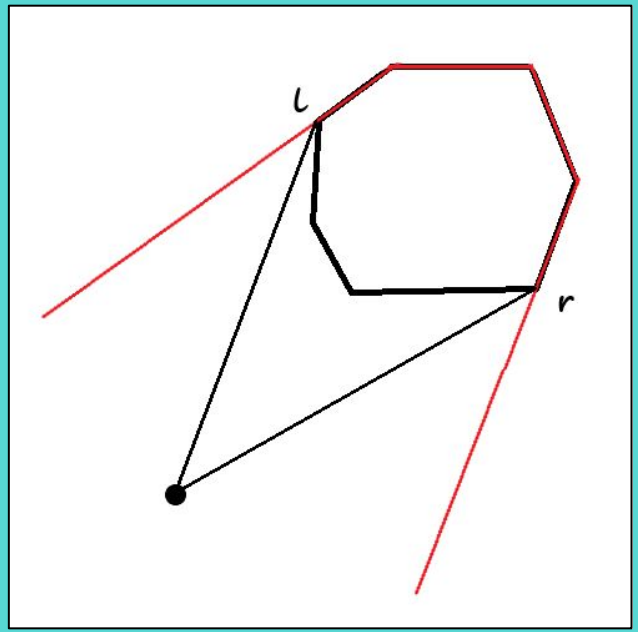
Рассмотрим точку запроса q и множество {Pi}. Имеем 2 подзадачи:

1) Найти 2 опорные точки l, r из q к каждому Pi
2) Построить граф видимости для полученных h отрезков



Опорные прямые к выпуклому многоугольнику за O(log ni)

В выпуклом многоугольнике Pi полуплоскости, пересечением которых он образован, монотонны относительно того, содержат ли они внешнюю точку q. Это делит полуплоскости на 2 подмножества. Для организации двоичного поиска вершин, разделяющих подмножества (опорные вершины) необходимо найти внутреннюю точку каждого из них. Прямая, проходящая через q и любую внутреннюю точку Pi, пересекает ребра Pi именно в таких точках.



Алгоритмическое решение

Пусть CH(Pi) - выпуклая оболочка Pi. Есть 2 случая:

1) $q \notin CH(P_i)$ - тогда опорные прямые к Pi совпадают с опорными прямыми к CH(Pi), которые могут быть найдены за $O(\log n_i)$
2) $q \in CH(P_i)$ - тогда опорные прямые могут быть найдены за $O(n^2)$

Предобработка: построение выпуклых оболочек для всех Pi. Алгоритм Чена: $O(n_i \log h_i)$, где h_i - количество точек Pi, образующих замыкание CH(Pi). Суммарная временная сложность: $O(\sum_{i=1}^{h} n_i \log h_i)$

Проверка принадлежности точки выпуклому многоугольнику CH(Pi) - $O(\log n_i)$

Построение графа видимости для полученных h отрезков

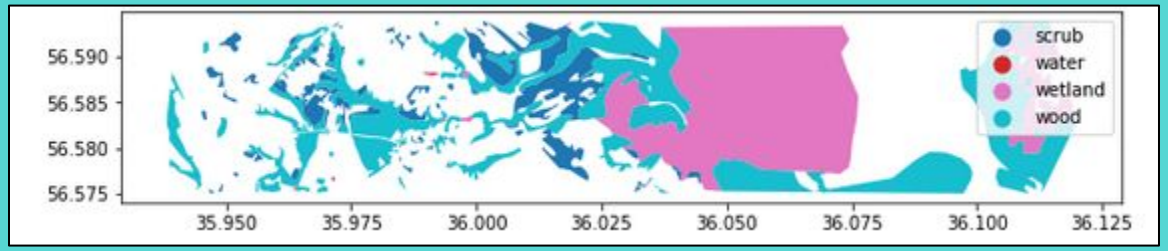
1) Алгоритм разделения окружности на num_angle углов и запоминание фиксированного количества расстояний отрезка, каждое из которых соответствует определенному углу наклона. Точность построения - π/num_angle. Погрешность ≤ 5° незаметна для человека. Данный параметр также используется для упрощения получаемого графа. Временная сложность - O(h), однако константа большая.
2) Алгоритм вращающейся заметающей прямой - O(h log h). На практике предпочтителен, так как работает быстрее.

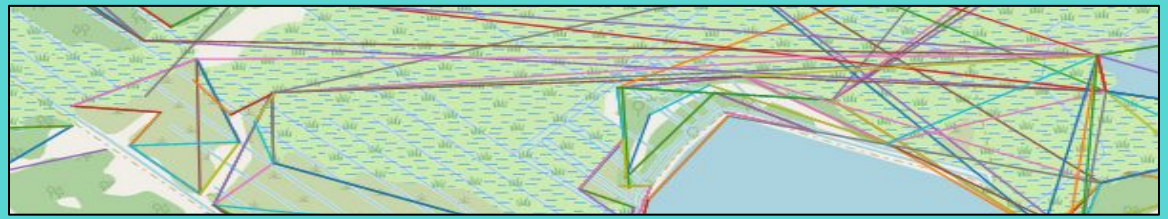
Итоговая сложность алгоритма

1) Нахождение 2h опорных точек: $O(\sum_{i=1}^{h} (\log ni + f(ni)))$, где $f(ni) = \log(ni)$, если $q \notin CH(Pi)$, $f(ni) = ni^2$, если $q \in CH(Pi)$. Из того, что $\sum_{i=1}^{h} (\log ni) < h * \log(\sum_{i=1}^{h} ni) = \log n$ следует, что минимальная сложность алгоритма - $O(h \log n)$, максимальная - $O(n^2)$. Память линейна. Однако не существует конфигурации, при которой $\forall q$, $\forall Pi$ -> $q \in CH(Pi)$. На практике количество таких точек $q << n$. Сложность $O(h \log n)$ достигается.
2) Построение графа видимости для отрезков - $O(h \log h)$

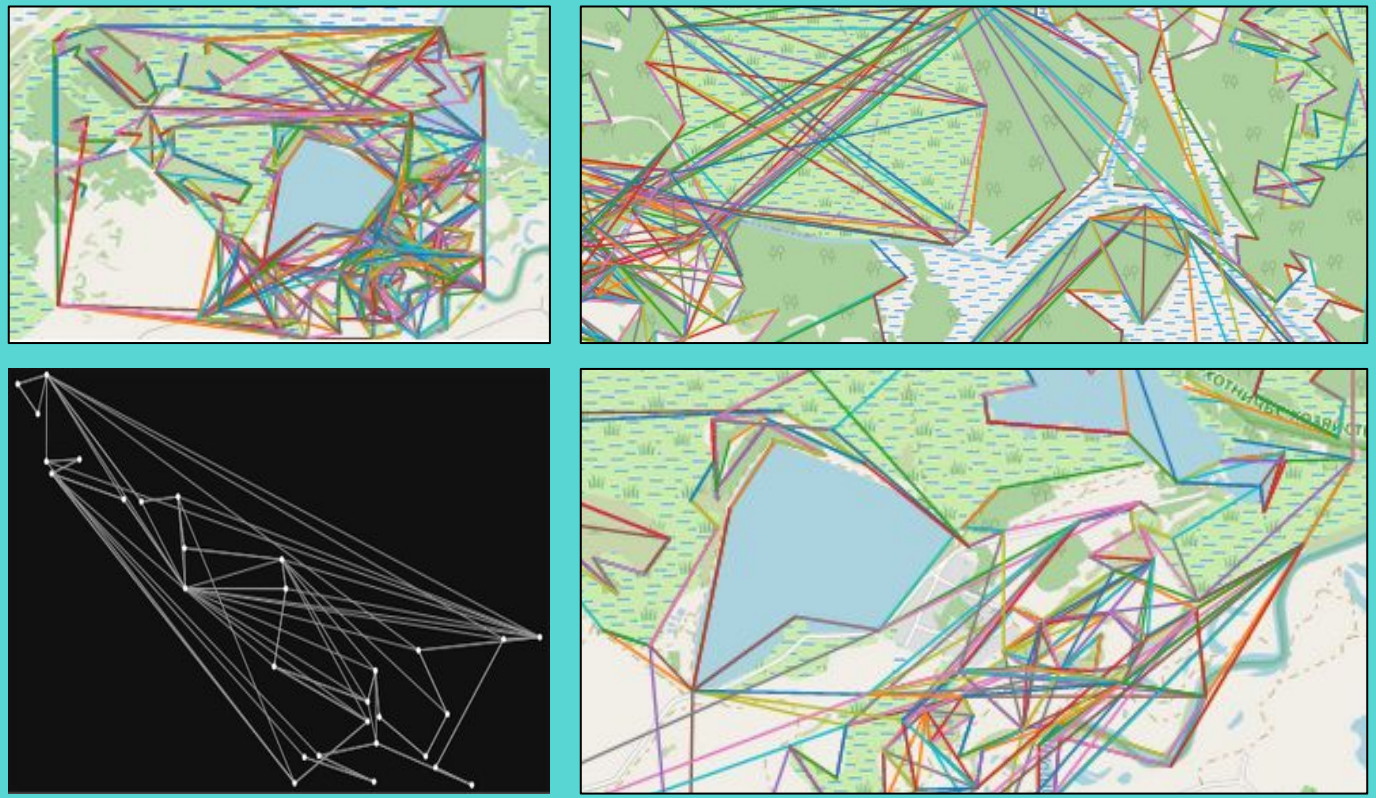
Итоговая сложность: $O(h \log nh)$. Граф является оптимизированным.

Результат работы





Результат работы



Спасибо за внимание!