Санкт-Петербургский государственный университет

Лунев Артем Евгеньевич

Выпускная квалификационная работа

Интеллектуальный помощник цифрового двойника сейсморазведки

Уровень образования: бакалавриат

Направление *09.03.04 «Программная инженерия»*

Основная образовательная программа *СВ.5080.2018 «Программная инженерия»*

Научный руководитель:
доц. кафедры СП, к. т. н., Ю.В. Литвинов

Рецензент:
Руководитель группы разработки ООО «Ланит-Терком», к.ф.-м.н., О.О. Якушкин

Санкт-Петербург
2022

Оглавление

1. Введение

Разведка и добыча нефти и газа является сложной и затратной деятельностью. Она нередко производится в сложных природных условиях, вдалеке от мест планирования и организации работ. Из-за природных условий работы в среднем длятся около полугода (с осени по весну), и зачастую это время надо использовать с максимальной эффективностью — мобилизовать технику, построить необходимые сооружения, а также произвести все необходимые сейсморазведочные работы.

Цифровой двойник сейсморазведки — это информационная система, которая позволяет сделать процесс планирования и подготовки к полевому сезону более быстрым и качественным, а также отслеживать прогресс выполнения геологоразведочных работ. Она представляет собой единую базу проектов сейсморазведки, разделенных на три этапа — планирование, мобилизация техники/персонала и полевой этап. Системой пользуются все участники процесса: кураторы и администраторы, которые планируют и отслеживают весь цикл работ, а также подрядчики, которые являются непосредственными исполнителями работ.

На этапе планирования утверждаются исследуемый контур работ, методика проведения работ, количество и численность бригад, а также планируемый объем работ. На этапе мобилизации указывается техника и персонал, необходимые для проведения работ. На этапе полевых работ исполнители работ заполняют ежедневные сводки, в которых указаны объемы проделанных работ, данные о погодных условиях, причины сниженной производительности или нерабочих дней, а также могут посмотреть аналитику по своим работам.

Главной задачей сейсморазведки является сбор геологических данных о месторождениях на местах проведения работ. Для сейсмической обработки местности строится сетка из профилей — дорог, которые прокладываются для будущего движения техники (рис. 1). При прокладке дороги для техники обычно расчищают снег и вырубают деревья, для проезда транспорта. По горизонтальным профилям с некоторым шагом стоят пункты возбуждения — места, где будет производиться

взрыв, вибрация или импульс для получения информации о залежах. По вертикальным профилям аналогично стоят пункты приема, где будут собираться сейсмические данные.

На рис. 1 продемонстрировано эталонное расположение профилей, но как видно из картинки, некоторые точки попадают в водные объекты. Также они могут попадать в овраги, места сильной залесенности, болота и так далее. Из-за этого сейсмикам приходится перестраивать каждый профиль, но при этом каждая точка приема или возбуждения должна остаться в пределах некоторого радиуса, в зависимости от метода исследования, иначе сейсмические данные будут неточными. Чем ближе точка к эталонной, тем лучше результаты исследования. Также, разумеется, от любой точки профиля до следующей точки должен существовать маршрут, по которому сможет пройти техника. Недавно компания-заказчик с помощью дронов произвела высокоточные съемки местности месторождений, и вследствие этого появилась возможность автоматизации построения профилей для техники.

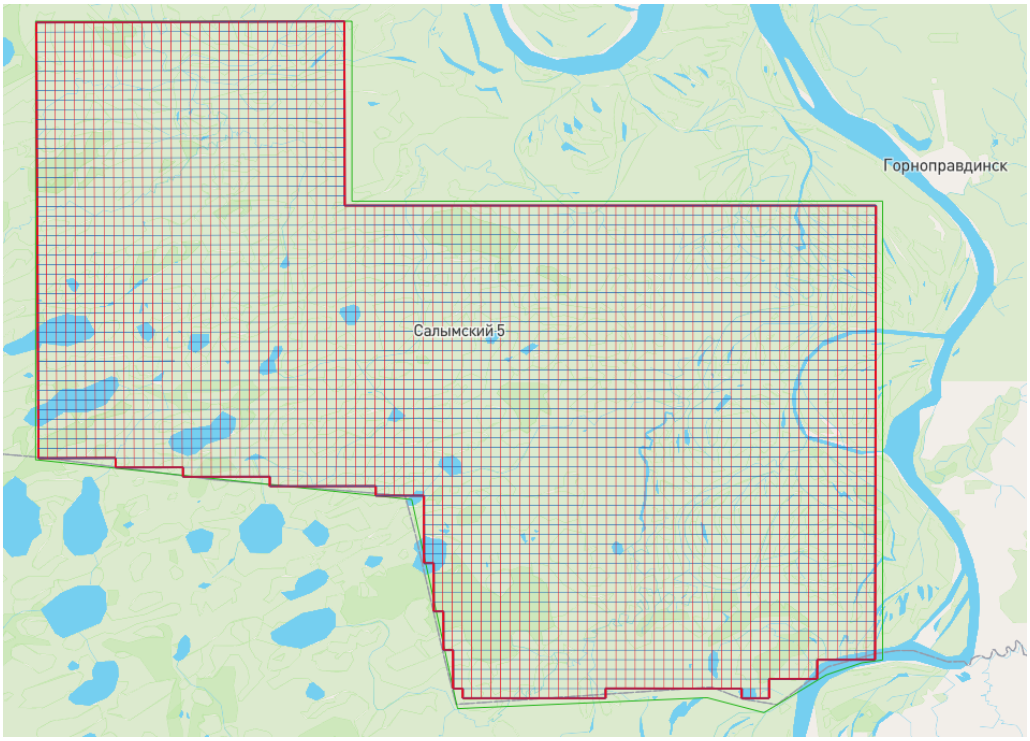

Рис. 1: Исходное отображение сетки профилей в системе

В данной работе представлена подсистема Цифрового двойника сей-

сморазведки, которая позволяет оптимизировать построение профилей на сетке, накрывающей исследуемую область работ. Она позволит снизить риски, возникающие при работе над полевыми проектами, цену и время, затрачиваемые на проект, а также минимизирует количество вырубки леса. При планировании проекта сейсмики смогут более качественно оценить необходимые ресурсы и количество бригад для проекта. Для визуализации результатов пользователю будет использоваться интерактивная карта.

2. Постановка задачи

Целью данной ВКР является создание в рамках системы “Цифровой двойник сейсморазведки” подсистемы, автоматизирующей прокладку профилей с учетом особенностей рельефа. Для достижения этой цели были поставлены следующие задачи.

1. Обзор алгоритмов, решающих задачу прокладки маршрутов.
2. Сбор требований к подсистеме и согласование входных данных.
3. Создание алгоритма прокладки профилей.
4. Реализация подсистемы на основе созданного алгоритма.
5. Проведение пользовательского тестирования.

3. Обзор

3.1. Архитектура системы

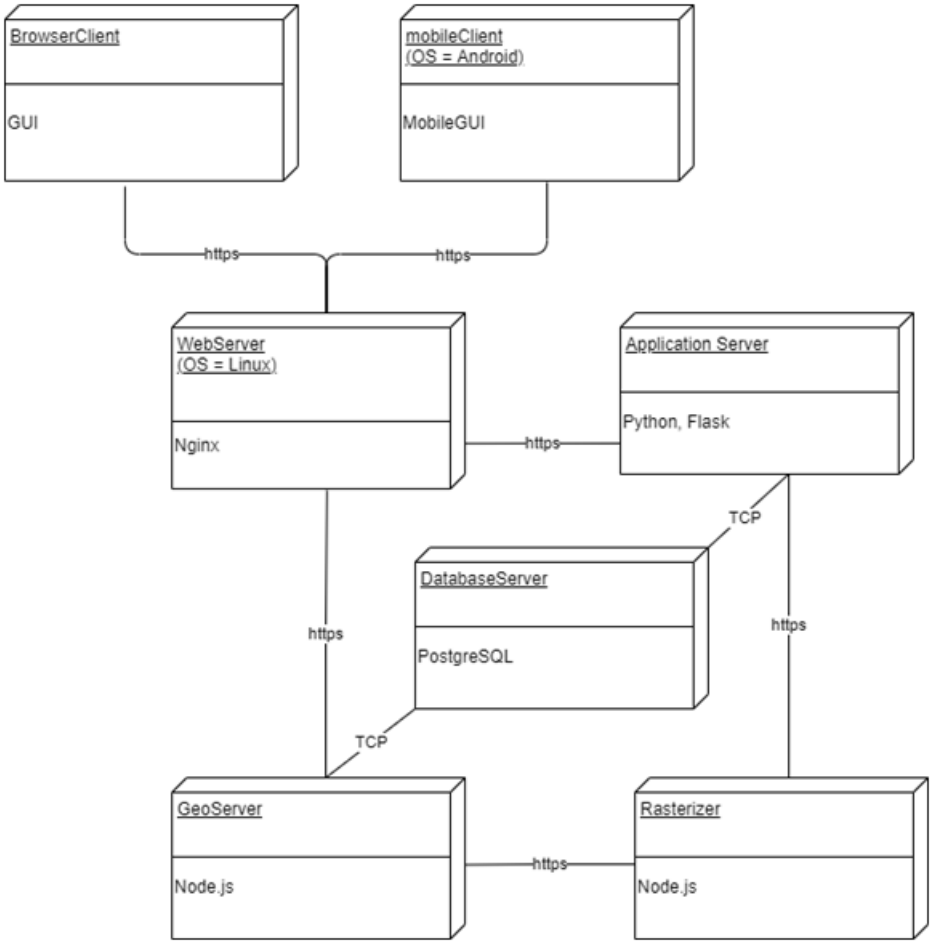

Рис. 2: Архитектура системы

На рис. 2 представлена архитектура системы «Цифровой двойник сейсморазведки». В системе есть два клиента — браузерный и мобильный, которые взаимодействуют с серверным приложением, через веб сервер nginx. В качестве СУБД используется PostgreSQL.

Важной задачей системы является работа с картами, на которых отображаются как географические объекты, так и объекты, необходимые для работы заказчика — контуры, сетки профилей и т.д. Для работы с картами используется геосервер. Работать с геопространственными данными в вычислительных системах принято по слоям, где каждый слой предоставляет информацию о каком-то объекте местности, напри-

мер, деревьях, реках, болотах, уклонах. Каждый слой может быть представлен в векторном или растровом формате. Векторные слои представляют из себя набор геометрических примитивов, а растровые — набор пикселей с географической привязкой. Эти слои хранятся в бинарном виде на сервере базы данных. Для работы с ними используется расширение PostgreSQL под названием PostGIS [12]. При интерактивной работе клиента с картами геосервер получает запросы, содержащие информацию о требуемых слоях, координаты участка карты, уровень приближения и параметры для фильтрации объектов на карте. Геосервер накладывает слои друг на друга, упрощает представление геометрических примитивов, если возможно, и задает стили отображения слоев. В ответ геосервер отправляет клиенту векторные тайлы – небольшие участки карты в векторной форме.

Но серверное приложение не обладает возможностью обрабатывать векторные тайлы и строить по ним изображение. Это часто бывает необходимо для использования изображений карт в автоматически генерируемых документах. Для получения таких изображений и передачи их серверному приложению используется растерайзер, который растеризует векторные изображения.

3.2. Особенности прокладки профилей

Зачастую сейсморазведка проводится в необработанной человеком местности, например тайге. Перед проездом техники на ее предполагаемом проезде подготавливается путь. Часто из-за больших габаритов техники и ее малого угла поворота она оказывается не в состоянии пройти по намеченному пути. Также техника имеет малые предельные углы наклона, из-за чего часто оказывается не в состоянии произвести намеченный подъем или спуск.

Зачастую техника не может пройти свой маршрут без вырубки деревьев, но при этом количество срубленных деревьев должно быть минимально. В случаях, когда сруб дерева дает значительную выгоду в минимизации длины маршрута, принято рубить дерево. При этом, для

разных регионов и разных лесов критерии сруба дерева могут отличаться.

При движении техники часто на ее пути возникают ямы, овраги и холмы. Если техника сможет проехать по ним без превышения своих предельных углов наклона, но объезд препятствий не доставляет много трудностей и не приводит к большому смещению точек приема и возбуждения, зачастую принимается решение объехать данные препятствия.

3.3. Алгоритмы прокладки маршрутов

Поставленную задачу можно рассматривать как задачу нахождения пути от одной точки до другой с большим количеством нюансов. При этом стоит строить путь не от одной точки профиля до другой, а сразу от первой до последней, иначе оптимальность пути не будет глобальной. Например, могут возникнуть ситуации, когда при движении от одной точки до другой алгоритм найдет оптимальный маршрут, но техника встает в такое положение, что чтобы пройти к следующей точке ей нужно развернуться из-за препятствий перед ней. И в итоге она могла бы пройти, отклонившись от нужной точки немного дальше, но не разворачиваясь и не проходя дополнительное расстояние. Эту задачу можно решать с помощью теории графов, либо векторной алгебры, либо с использованием обоих подходов.

Пути Дубинса

Для решения задач моделирования движения транспорта часто используются так называемые пути Дубинса [6]. Математик Л. Дубинс доказал, что для транспортного средства с минимальным радиусом поворота r, кратчайший путь от начальной точки к конечной будет состоять из прямых линий и двух или менее окружностей радиуса r. Причем если обозначить поворот налево L, поворот направо R, движение по прямой S, то оптимальный путь будет одним из шести комбинаций RSR, RSL, LSR, LSL, RLR, LRL. Пример пути RSL на рис. 3.

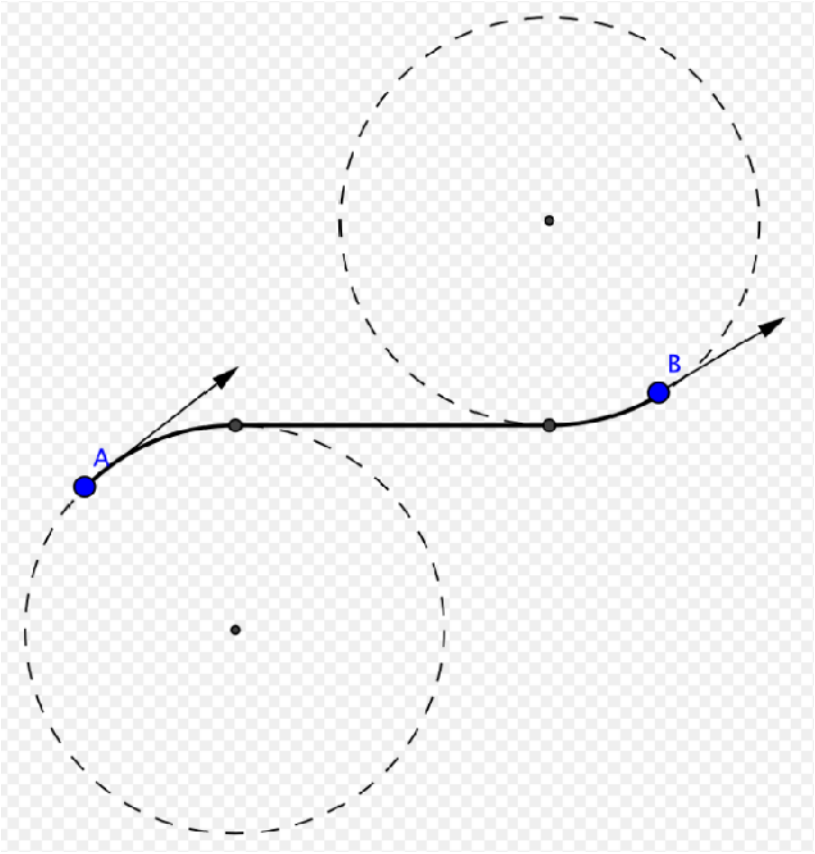

Рис. 3: Путь Дубинса

Существуют также вариации алгоритма, работающие с препятствиями [3], а также вариации, работающие с 3d картами [9]. Недостатки для поставленной задачи – невозможность задать условие на прохождение промежуточных точек в допустимом радиусе, невозможность гибко задать условия на обход препятствий. Одно из возможных применений — предрасчет пути от каждого препятствия до соседних. Так как при оптимальном движении техника всегда будет обходить препятствия, приблизившись к ним на минимальное расстояние. Это позволит оптимизировать алгоритм.

Алгоритм A*

Алгоритм A* [1] является оптимизацией алгоритма Дейкстры для поиска пути из начальной вершины до конечной, а не до всех вершин. Оптимизация заключается в том, что сначала просматриваются верши-

ны, находящие ближе к конечной по выбранной эвристике h(x) (обычно расстояние от текущей до конечной вершины). Приоритет просмотра вершины при очередной оптимизации цены складывается из текущей цены от начальной вершины до текущей и эвристики (f(x) = h(x) + cost(x)). Утверждается, что если эвристика никогда не больше истинной цены от текущей вершины до конечной, алгоритм найдет оптимальный путь. От выбора эвристики сильно зависит количество просмотренных вершин, и соответственно скорость работы алгоритма.

4. Требования

4.1. Функциональные требования

Входными данными для работы подсистемы являются параметры техники, которая будет использоваться для проезда по профилям, а также слои с географическими объектами. Параметрами техники являются ее длина, ширина, предельный продольный и поперечный угол наклона, минимальный радиус поворота и клиренс (расстояние между опорой и нижней частью техники). Входными слоями являются слой древостоя (векторный слой с деревьями поштучно), слой высот рельефа (растровый слой, где значение каждого пикселя представляет собой высоту рельефа) и слой исходных профилей (векторный слой). При этом векторные слои приходят в формате shapefile [4] — одном из популярных форматов векторных географических файлов, а растровые в формате GeoTIFF [8]. Также пользователь должен иметь возможность задать баланс между вырубкой деревьев и получаемой от этого выгоды. Чаще всего этот параметр будет устанавливаться в зависимости от региона работ.

После получения входных данных подсистема должна перестроить исходные профили. Они должны проходить через все пункты приема и возбуждения в заданных радиусах если они достижимы и по возможности смещены ортогонально. Так как изначально не известно, с какой точки будет стартовать техника, необходимо найти оптимальную для старта точку в допустимом радиусе отклонения первого пункта профиля. Количество срубленных деревьев должно быть минимизировано в соответствии с параметром баланса пользователя. Места с большим перепадом высоты рельефа, такие как ямы, овраги и холмы, следует объезжать, если это не сказывается на результате. Также, очевидно, должна быть возможность проехать по построенному пути — перемещение должно быть плавным, не превышающим максимальный угол поворота техники, техника не должна превысить свои углы наклона и разность высот между точками опоры и дном техники не должна пре-

вышать ее клиренс. После перестроения профилей подсистема должна отобразить их на интерактивной карте, а также отобразить смещенные пункты приема и возбуждения.

4.2. Нефункциональные требования

Единственным значимым нефункциональным требованием является ограничение по времени. Обычно суммарная длина всех профилей составляет около 5000 километров, а географические данные могут достигать больших размеров, что значительно повышает время работы алгоритма. Из-за этого было решено, что это не будет мгновенный процесс для пользователя, и чаще расчеты данной подсистемы будут запускаться ночью. Желаемое ограничение на время работы подсистемы при запуске — 1 час. Но и этого времени работы добиться не просто, хотя следует отметить, что задача хорошо масштабируется, так как каждый профиль можно рассчитывать независимо друг от друга.

5. Алгоритм прокладки профилей

Из-за большей гибкости графовых алгоритмов в качестве базового алгоритма был выбран алгоритм A*. Изначально карта разбивается на сетку, состоящую из клеток со сторонами приблизительно 30 см (зависит от минимального радиуса поворота техники). Каждая клетка рассматривается как вершина графа с ребрами, соединенными с ее соседями. Основными проблемами, которые необходимо решить в рамках этого алгоритма, являются обеспечение плавности движения техники, необходимость прохождения через промежуточные точки, минимизация сруба деревьев и объезд ям, оврагов и холмов.

5.1. Учет ограниченной мобильности техники

В стандартном разбиении при применении алгоритма A* размер клетки обычно соответствует размеру перемещаемого объекта, но при таком разбиении нет возможности для отслеживания положения техники, и соответственно углов поворота. Техника могла бы переместиться из текущей клетки в любую соседнюю, например восточную, но в реальном мире техника могла быть повернута на север, и с такой легкостью не переместилась бы на восточную клетку. Важно понять, что поворот не происходит на месте, как в примере выше, а происходит при движении вперед или назад.

Из-за проблем, описанных выше, в реализации алгоритма используется пара для задания вершины графа – центр техники и ее угол поворота, по которому можно понять направление движения. Ребра задаются так, чтобы пройти по минимальному радиусу поворота или большему радиусу, для большей плавности движения. На рис. 4 показан пример поворота техники по минимальному радиусу. Переходы осуществляются между красными клетками, при этом при каждом переходе меняется текущий угол поворота. Оранжевыми линиями выделены ребра. Как можно видеть из картинки, при достаточно малом выборе размера клетки алгоритм построит маршрут вдоль минимального радиуса, а при размере клетки стремящейся к нулю совпадет с ним. Помимо пере-

ходов вдоль окружности, представленной на рисунке, также возможны переходы вдоль левоориентированной относительно техники окружности, аналогичные движения в обратную сторону вдоль окружностей и движение по прямой из каждого положения.

При этом возможные углы поворота, в которых может оказаться техника, фиксированы. В примере на рисунке 4 изображена четверть окружности и пять возможных углов поворота. Из каждого фиксированого угла можно перейти в «соседние» фиксированные углы, либо остаться с таким же углом поворота, при движении по прямой. При достаточном количестве углов поворота, в котором может оказаться техника, получающийся результат хорошо аппроксимирует реальный результат.

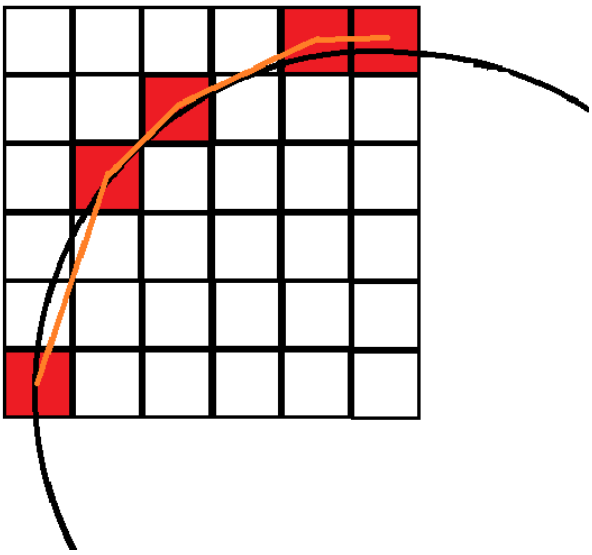

Рис. 4: Пример поворота

Таким образом, поворот происходит при движении, так как угол поворота может меняться только при перемещении от одной клетки к другой, и движение остается плавным. Также оно становится более реалистичным – например, при крутом повороте, если угол поворота техники слишком мал, алгоритм построит маршрут так, что техника отъедет сначала в противоположную повороту сторону, чтобы взять пространство для маневра и успеть поменять свое направление, как

это происходит при реальном движении.

5.2. Учет слоя древостоя и слоя высот рельефа

Для решения проблемы объезда деревьев подходит увеличение цены ребра при переходе из одного положения в другое, если в новом положении техника пересекается с деревом. Если цена при этом увеличивается например на единицу, то это значит, что будет принято решение срубить дерево, если это уменьшит общую длину пути на одну клетку, а зная размер клетки можно понять сколько метров пути экономится при срубе. Таким образом, пользователь может задавать баланс рубки деревьев в терминах экономии пути проезда техники за сруб дерева. Также цена сруба зависит от диаметра дерева и от типа дерева — например березы или дуба. Эта информация содержится в атрибутах слоя древостоя. Так как срубленные деревья относятся не ко всем путям графа, а к каждому из всех просматриваемых алгоритмом необходимо хранить текущие срубленные деревья на уровне вершины графа, чтобы не посчитать их срубленными несколько раз. Таким образом вершина графа становится тройкой, а не парой.

Для вычисления углов наклона используется слой высот рельефа. Зная габариты техники, а также ее текущий угол поворота, и следовательно ее положение, можно определить, где находятся ее колеса, а по высоте, на которой они стоят, можно понять продольный и поперечный углы наклона. Если при переходе в очередное положение эти углы превышают предельные допустимые углы, то такой переход считается невозможным, т.е. в графе отсутствует ребро.

Для объезда рельефа с большим перепадом высот, таких как ямы и холмы, можно было бы применить тот же способ, что и с деревьями — увеличивать стоимость ребра за силу перепада углов наклона техники при прохождении по ребру, но такой способ имеет два недостатка. Во-первых цена за эти перепады достаточно мала, так как объезд происходит, только если это ничему не вредит. Но так как эта малая цена будет добавляться практически к каждому переходу, это сильно сни-

зит скорость работы алгоритма. Например, если алгоритм приблизится к финальной точке на одну клетку, но цена ребра составит 1.01, то из-за неидеального возможного приближения в соответствие с эвристикой, алгоритму нужно просмотреть также и другие соседние переходы, иначе путь может оказаться не оптимальным по цене. Таким образом значительно увеличивается количество просмотренных вершин графа. Во-вторых не учитывается скорость изменения угла наклона техники. То есть долгий спуск по пологому склону будет восприниматься так же, как и по крутому склону, так как суммарный перепад угла наклона будет одинаковый. Можно сказать, нужно смотреть на производную перепада угла наклона. Также, чтобы эти малые цены повлияли на результат, они должны копиться за несколько переходов по ребрам. Могут возникать случаи, когда при очередном переходе при не слишком большом перепаде высоты это играет роль и алгоритм неожиданно смещает путь. Другими словами алгоритм должен реагировать конкретно на ямы и холмы, а не на каждое, возможно малое, изменение высоты.

Из-за проблем, описанных выше, цена разделена на приоритеты. Цена первого приоритета — это обычная цена, без цены за перепад угла наклона техники, а цена второго приоритета накапливается за перепад угла наклона. Когда цена второго приоритета накапливается до некоторого заданного уровня, она переходит в единицу цены первого приоритета, до этого момента она ни на что не влияет. Также после каждого перехода по ребру она постепенно снижается на фиксированное значение вплоть до нуля. Таким образом, во-первых, решается проблема с просмотром большого количества вершин, так как перепады влияют только после достижения некоторого порога, то есть не часто. Во-вторых, из-за постоянного снижения цены второго приоритета для влияния на результат она должна за малое количество переходов дойти до своей границы, а значит теперь начинает учитываться скорость перепада. При достижении этой границы, рельеф будет восприниматься как препятствие, и не будет ситуаций, когда алгоритм неожиданно среагирует на небольшое изменение цены спустя некоторое время из-за ранее накопленной цены второго приоритета. То есть алгоритм либо

воспримет яму как яму, если она достаточно глубока для этого, либо не среагирует на нее.

5.3. Учет отклонения от промежуточных точек профиля

Как упоминалось выше, для глобально оптимального пути алгоритм не может двигаться от одного пункта профиля до другого и должен двигаться от первого до последнего. В связи с этим встает вопрос гарантии прохождения промежуточных точек.

Но для начала нужно разобраться с точкой старта. Следует заметить, что в требованиях было условие выбора оптимальной точки старта в пределах заданного радиуса. Конечно, можно запустить алгоритм несколько раз из каждой возможной вершины, но это бы замедлило его примерно в 60 раз, из-за 60 возможных точек старта. Поэтому все же точка старта остается одна, но из нее заданы нулевые ребра во все возможные другие точки старта, а значит такие переходы не влияют на итоговую цену и тоже воспринимаются как точки старта. Это позволяет алгоритму не просматривать одни и те же пути несколько раз, но все же замедляет его работу примерно в три раза.

Для учета отклонений от промежуточных точек к вершине добавляется новый компонент — список посещенных окрестностей, и вершина на самом деле является четверкой. При проходе по ребру, ведущему в одну из окрестностей, которые необходимо посетить, в четвертый компонент вершины добавляется посещенная окрестность. После этого, проход по любым ребрам, вне зависимости от того отдаляют они от конечной вершины или приближают, не сотрут из списка эту окрестность. Окрестности могут только добавляться в четвертую компоненту вершины. Конечная вершина задается как исходая конечная вершина, но содержащая все промежуточные вершины в четвертом компоненте. Таким образом, алгоритм закончит работу, только если найдет путь, проходящий через все промежуточные вершины. Очевидно, алгоритм бы работал слишком долго на графе, заданном таким образом, если бы

его работа не регулировалась эвристикой.

Это также дает возможность восстанавливать пути, когда техника заходит в область промежуточного пункта, а потом из-за препятствий впереди выходит из нее тем же путем, что и пришла, и начинает объезжать препятствия, так как едет назад она, формально, по другим вершинам, что не дает зациклиться массиву для восстановления путей. В противном случае, для вершины x, массив бы указывал предыдущую вершину в найденном пути y, а для вершины y — x. Для восстановления пути необходимо пройти по данному массиву, начиная с конечной вершины.

Хотя в данной работе говорится о вершине как о четверке, следует понимать, что вершина — это обычная вершина графа, а ее компоненты всего лишь абстракция. Ограничения связанные с этими компанентами выполняются из-за верно заданных ребер. Каждая вершина по сути привязана к соответствующей клетке карты и к каждой клетке просто привязано несколько вершин, так как они могут отличаться другими своими компонентами, такими как угол поворота, срубленные на текущий момент деревья и посещенные промежуточные вершины.

5.4. Эвристика

Как было описано выше, эвристика задает приоритет просмотра для очередной вершины, который складывается из нее самой и цены до вершины. Она сильно влияет на время работы алгоритма. Есть несколько нюансов, связанных с эвристикой в данном алгоритме. Обычно она задается как расстояние от текущей точки до конечной, так как чем ближе мы к конечной точке, тем приоритетней пойти по данному пути. В данной эвристике это также берется за основу, но помимо этого она должна также учитывать промежуточные точки. Дойти до конечной точки, можно только пройдя через все промежуточные. Это значит, что расстояние, которое необходимо пройти алгоритму можно оценивать как сумму расстояния от текущей точки до первой не посещенной окрестности и расстояние от этой окрестности до финальной точ-

ки. При этом эвристика не становится больше истинной цены, так как чтобы дойти до финальной точки, необходимо пройти через промежуточные. Таким образом, если путь отдаляется от окрестности, которую еще не посетил, его приоритет просмотра снижается. Чем дальше, тем сильнее. Это позволяет избавиться от просмотра большого количества вершин в этом графе.

Также эвристика учитывает направление техники. Если, например, конечная точка лежит на севере, а техника направлена на восток, то приоритет просмотра данного положения уменьшается. В таком случае можно утверждать, что расстояние по прямой от текущей точки до конечной — не самое лучшее приближение цены, так как технике придется пройти расстояние при повороте. Это позволяет избежать просмотра бесполезных поворотов. При старте эвристика отдает предпочтение точкам, находящимся ближе к исходному пункту, чтобы при существовании одинаковых по цене путей, результатом был путь, лежащий ближе к промежуточным точкам.

6. Особенности реализации

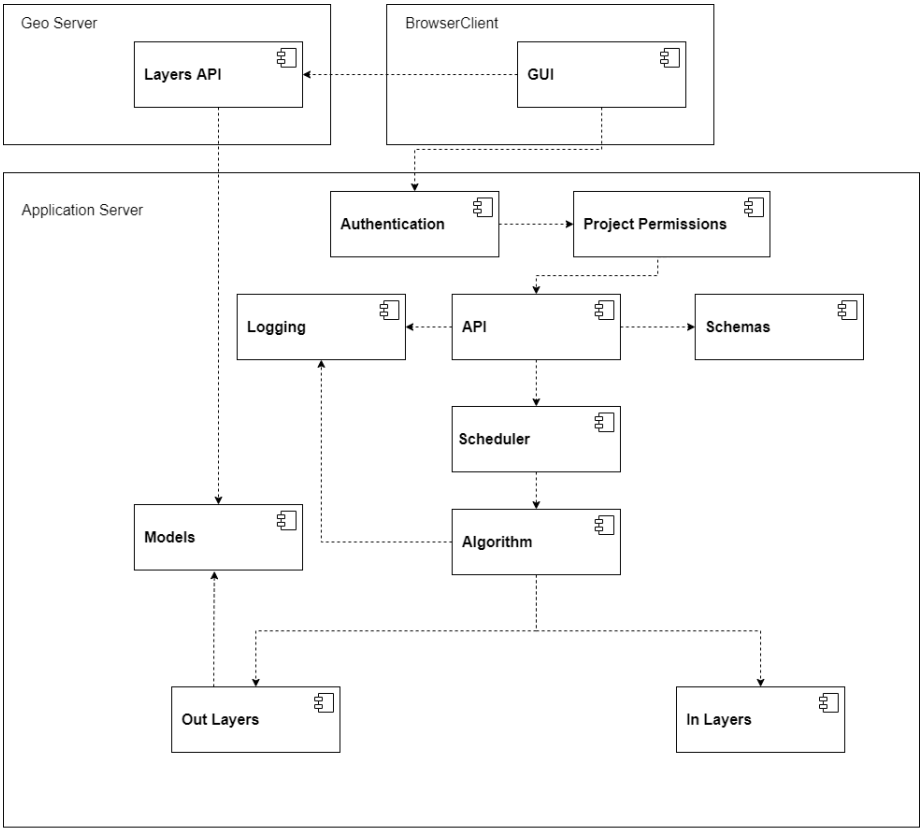

Рис. 5: Диаграмма компонентов подсистемы

На рис. 5 изображена диаграмма компонентов подсистемы. Любой запрос к подсистеме начинается с аутентификации (Authentication). Для этого используется протокол Kerberos. После этого проверяются права доступа на действие над проектом (Project Permissions). Они зависят от роли пользователя в системе и его связи с проектом. Подсистема работает с сейсмическими проектами, у которых планируется использование сетки профилей.

В компоненте Schemas содержатся схемы для сериализации, десериализации и валидации запросов и ответов API. При сериализации объекты json, используемые в теле запроса, конвертируются в типы данных языка Python, а при десериализации наоборот. Для упрощения и универсализации этого процесса используется библиотека marshmallow [10].

Для решения трудоемких или отложенных задач на сервере суще-

ствует отдельный компонент «Планировщик» (Scheduler). Для его реализации используется библиотека APScheduler [2]. Так как задача поиска профилей занимает много времени, для нее создается отдельная задача, которой присваивается свой uuid. По этому uuid FrontEnd может получить информацию о статусе задачи.

Из планировщика запускается алгоритм по построению профилей (компонент Algorithm). В нем содержится все, что нужно для выполнения алгоритма описанного выше, кроме непосредственной работы с геослоями.

Компонент In Layers работает с входными слоями. Эти слои ограничевают перемещение по построенному нами графу. Они предоставляют интерфейс для увеличения ребра графа, который на вход принимает соответствующую информацию о перемещении по графу и возвращает новую цену ребра. Также для упрощения работы со всеми профилями, с ними ведется работа как с горизонтальными профилями, из-за чего в начале осуществляется поворот профиля, при необходимости. Поэтому входящие слои также умеют совершать поворот.

Компонент Out Layers принимает результаты, полученные после выполнения алгоритма. Слои в этом компоненте имеют интерфейс для генерации их в бинарном виде и сохранении в базу данных.

Для работы с географическими данными используется библиотека GDAL (Geospatial Data Abstraction Library) [7]. Она предоставляет широкую функциональность по работе с векторными и растровыми данными, позволяет генерировать новые файлы, конвертировать слои из одной географической системы координат в другую, выполнять операции с геометрическими данными и т.д.

В компоненте Models содержатся классы для работы с базой данных. В системе используется ORM Flask-SQLAlchemy [5], которая позволяет работать с базой данных в терминах классов. Для поддержания работы подсистемы были созданы классы (соответствующие таблицам базы данных), которые представляют выходные слои со связью с соответствующим сейсмическим проектом в системе. На гео сервере был создан новый слой, работающий с этими таблицами базы данных. Он

принимает один параметр — uuid проекта, слои которого хочет получить клиент и производит по нему фильтрацию.

Из-за реализации сервера на языке Python, в целях упрощения разворачивания системы и ее целостности, было принято решение не выходить за рамки данного языка, хоть он и не слишком хорошо подходит для вычислительно сложных операций. По этой причине была использована библиотека Numba [11], которая позволяет компилировать код на языке Python в машинный код. Это снизило время работы алгоритма примерно в 2 раза. На текущий момент один профиль в 500 метров строится приблизительно за 12 секунд со среднеквадратичным отклонением в 5.1 секунд, если нет необходимости искать оптимальную стартовую точку, и в 4 раза дольше, если такая необходимость есть. Время работы сильно зависит от количества деревьев и рельефа на пути.

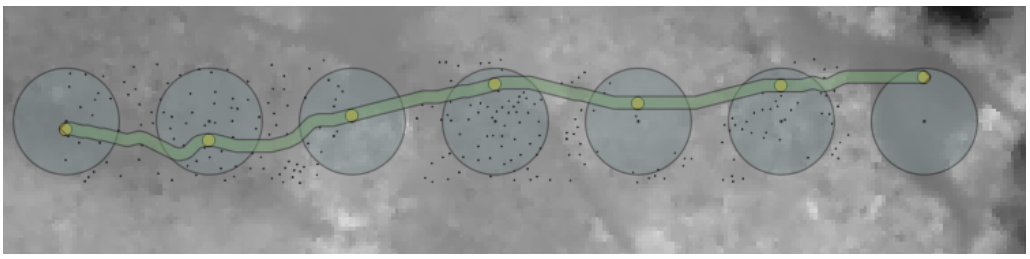

Рис. 6: Пример построения профиля

На рис. 6 изображен пример построения профиля. На нем зеленым цветом обозначен сам профиль, маленькие точки — деревья, желтые точки — смещенные пункты приема, голубые круги — допустимый радиус смещения. Каждый пиксель серого слоя высот соответствует высоте рельефа — чем он темнее, тем высота меньше.

7. Пользовательское тестирование

Для тестирования работы алгоритма, пользователям был предоставлен плагин системы QGIS, которая предназначена для работы с географическими слоями. В ней пользователи могли задать необходимые параметры, и получить сетку профилей. Пример такой сетки представлен на рис. 7.

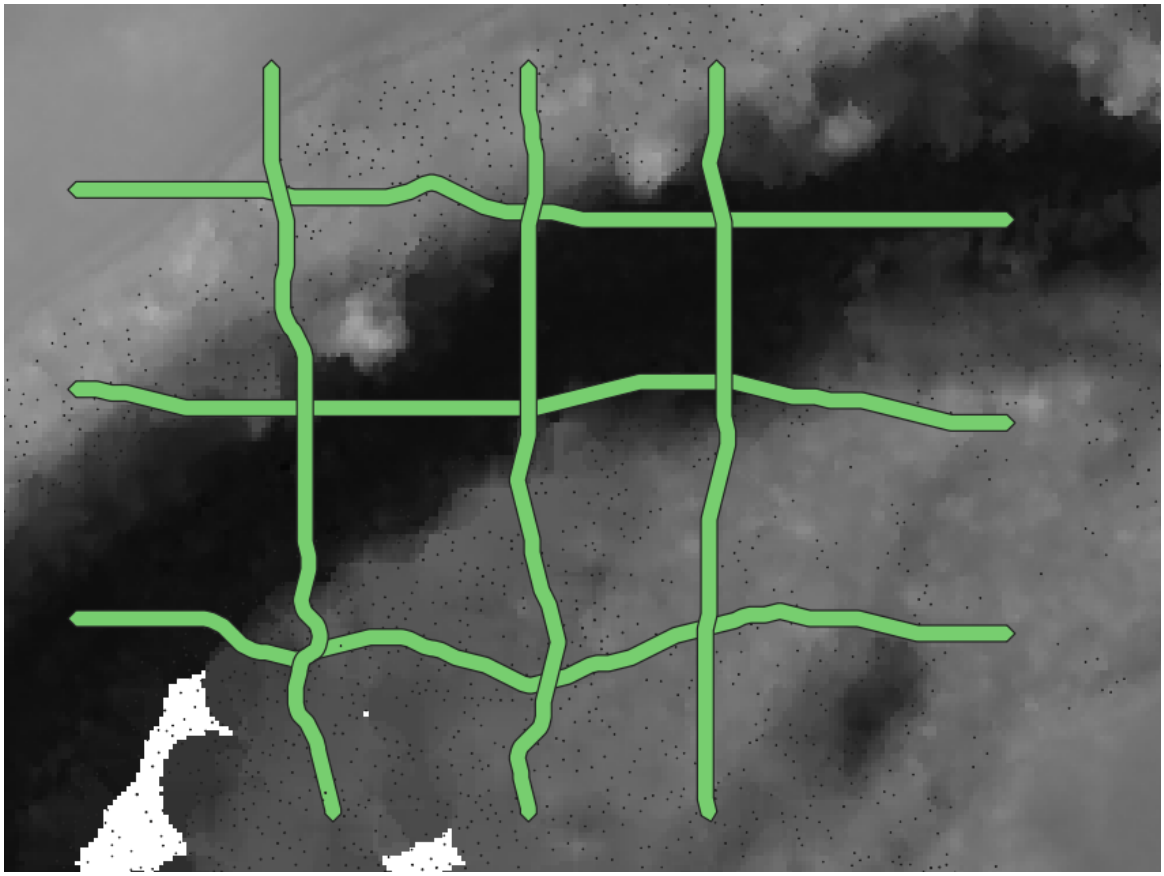

Рис. 7: Пример сетки профилей

На текущий момент основные проблемы, возникавшие у пользователей, решены. Оставшимися предложениями пользователей для улучшения работы алгоритма являются более продвинутое выставление цены за сруб дерева, в зависимости от размера ствола (чем меньше размер, тем меньше цена за сруб) и более сглаженное отображение профилей. По большей части претензий к качеству построения профилей нет, но есть пожелания по снижению времени работы алгоритма.

8. Заключение

В ходе работы были достигнуты следующие результаты.

1. Сделан обзор алгоритмов, решающих задачи прокладки маршрутов — пути Дубинса, алгоритм A*. Из-за большей гибкости в качестве базового алгоритма был выбран A*.

2. Был выполнен сбор требований к подсистеме: проработана задача, стоящая перед сейсмиками, согласованы входные данные алгоритма.

3. Разработан новый алгоритм на основе алгоритма A*, который позволяет учитывать следующие особенности местности – деревья, уклоны, водные объекты, а также низкую проходимость техники и промежуточные точки профиля.

4. Выполнена реализация подсистемы (язык Python, библиотеки GDAL, Numba).

5. Проведено пользовательское тестирование, которое показало удовлетворенность пользователей подсистемой.

В настоящее время для передачи заказчику подсистемы необходимо завершение работы команды, которая занимается связной задачей — автоматическим выделением входных для подсистемы географических слоев из исходных данных, полученных от дронов. Также в будущем планируется расширение подсистемы в виде обработки других полезных географических слоев.

Список литературы

[1] A* algorithm. — URL: https://en.wikipedia.org/wiki/A*_search_algorithm (online; accessed: 2022-01-17).

[2] APScheduler docs. — URL: https://apscheduler.readthedocs.io/en/3.x/ (online; accessed: 2021-10-15).

[3] Dongxiao Yang Didong Li Huafei Sun. 2D Dubins Path in Environments with Obstacle. — China : Hindawi Publishing Corporation, 2013. — P. 1–6. — URL: https://www.researchgate.net/publication/274941408_2D_Dubins_Path_in_Environments_with_Obstacle (online; accessed: 2021-12-16).

[4] ESRI Shapefile Technical Description. — 1998. — URL: https://www.esri.com/content/dam/esrisites/sitecore-archive/Files/Pdfs/library/whitepapers/pdfs/shapefile.pdf (online; accessed: 2021-09-11).

[5] Flask-SQLAlchemy homepage. — URL: https://flask-sqlalchemy.palletsprojects.com/en/2.x/ (online; accessed: 2021-09-06).

[6] Fuchs Satyanarayana G. Manyam David Casbeer Alexander L. Von Moll Zachariah. Shortest Dubins Path to a Circle. — 2018. — URL: https://arxiv.org/pdf/1804.07238.pdf (online; accessed: 2021-12-21).

[7] GDAL homepage. — URL: https://gdal.org (online; accessed: 2022-02-01).

[8] GeoTIFF Standard. — 2019. — URL: http://docs.opengeospatial.org/is/19-008r4/19-008r4.html (online; accessed: 2022-01-07).

[9] Hota Sikha, Ghose Debasish. Optimal Geometrical Path in 3D with Curvature Constraint. — 2010. — URL: http://sector3.imm.uran.ru/magistr/literat/05653663.pdf (online; accessed: 2021-12-19).

[10] Marshmallow docs. — URL: https://marshmallow.readthedocs.io/en/stable/ (online; accessed: 2021-09-12).

[11] Numba homepage. — URL: https://numba.pydata.org/ (online; accessed: 2022-04-10).

[12] PostGIS homepage. — URL: https://postgis.net (online; accessed: 2021-09-10).