Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение высшего образования
РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ
(РГГМУ)

Институт Информационных систем и геотехнологий
КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

На тему Решение проблем сельскохозяйственной отрасли с использованием ГИС- технологий

Исполнитель Ёлдашов Рустам Закирович
(фамилия, имя, отчество)

Руководитель кандидат технических наук доцент
(ученая степень,
Нигматули Тагир Асюдулович
(фамилия, имя, отчество)

«К защите допускаю»
Заведующий кафедрой _____
(подпись)

доктор технических наук,профессор
(ученая степень, ученое звание)

Истомин Евгений Петрович
(фамилия, имя, отчество)

«___»_____20__г.

Санкт–Петербург 2022

СОДЕРЖАНИЕ

СПИСОК УСЛОВНЫХ ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

ASP	Active Server Pages
CSS	Cascading Style Sheet
CSV	Comma Separated Values
GUI	Graphics User Interface
HTML	HyperText Markup Language
JSP	Java Server Pages
MTBF	Mean Time Before Failure
PC	Personal Computer
PHP	Personal Home Page, **P**HP **H**ypertext **P**reprocessor
SQL	Structured Queries Language
WWW	World Wide Web
XML	eXtensible Markup Language
БД	База данных
ЗИ	Защита информации
ИТ	Информационные технологии
НИПТ	Несанкционированное использование подвижной техники
ОО	Объектно-ориентированный (-ая, -ое)
ООП	Объектно-ориентированное программирование
ПО	Программное обеспечение
ПК	Персональный компьютер
СВН	Система видеонаблюдения
СКТМ	Система контроля текущего местонахождения
СУБД	Система управления базами данных

ВВЕДЕНИЕ

Развитие компьютерной техники и сопутствующих информационно-коммуникационных технологий внесло многочисленные изменения во многие сферы жизни как отдельного человека, так и общества в целом. В частности, значительно изменились средства и подходы к обеспечению безопасности самых разнообразных систем и сфер нашей жизни. Чего стоит только повсеместное использование систем видеонаблюдения (далее – СВН), которые на сегодняшний день стали одним из основных средств раскрытия преступлений тяжкого характера и предотвращения легких. Однако, растущее количество камер делает их работу все более автоматизированной: всю информацию они пишут в архивы, а текущую картинку обычно никто не просматривает, и, только при возникновении такой необходимости, нужные участки видео истребуют из этих молчаливых цифровых архивов. Соответственно, возникает проблема сохранения огромных объемов цифровых данных, производимых десятками, сотнями и тысячами камер в городе, десятками тысяч камер в стране и т.д. С абсолютно аналогичной проблемой сталкиваются и другие системы безопасности, такие как, например, системы контроля текущего местоположения (далее – СКТМ) движущихся объектов, например таких как сельскохозяйственная техника.

Эти СКТМ, как и СВН, имеют определенные настройки, которые могут способствовать уменьшению объемов фиксируемых данных, но это становится возможным только за счет уменьшения «качества» сведений, хранящихся для данной предметной отрасли. Некоторые «ухудшения» информации вполне приемлемы, некоторые наоборот могут изменять содержание сведений, содержащихся в такой информации, на десятки процентов или более (что недопустимо). Соответственно, важной и **актуальной** задачей является выработка методов одновременно «безопасной» и эффективной минимизации данных о маршрутах движений контролируемого объекта (например, комбайна, работающего на поле).

Таким образом, **целью** работы можно считать эффективное уменьшение объемов маршрутных данных, хранящихся в соответствующем архиве для обеспечения всесторонней безопасности процесса перемещения движущихся наземных объектов, в частности относящихся к сельскохозяйственной технике.

Для достижения поставленных целей необходимо решить следующие задачи исследования:

- проанализировать существующие способы снятия, преобразования и хранения маршрутной информации с учетом использования возможностей уменьшения ее объемов;

- разработать алгоритм минимизации маршрутных данных на основе известных алгоритмов, выполняя их оптимизацию, а также необходимые сопутствующие инженерные решения (например, формат хранения данных и др.);

- осуществить программную реализацию разработанного алгоритма и его комплексное исследование;

- использовать особенности процесса минимизации с учетом различных зависимостей (в частности, от скорости движения объекта), требований надежности хранения, защиты соответствующей маршрутной информации.

Объектом исследования является процесс накопления и минимизации маршрутных данных движущихся объектов.

Предметом исследования являются методы и модели минимизации маршрутных данных.

В работе использованы общенаучные **методы** анализа (для обзора существующих решений и принципов их работы) и синтеза (для создания решений). Для описания кривых маршрутной информации использованы методы высшей математики, для разработки программной реализации и всех сопутствующих элементов – методы отрасли информационных технологий (программирования).

В работе получены новые научно-технические результаты:

- разработана процедура моделирования для получения маршрутной информации;

- проанализированы возможности применения методов высшей математики (сплайн-интерполяции, интегральных преобразований) для сжатия именно маршрутной информации сельскохозяйственной техники (с учетом специфики такой информации);

- на основе научного анализа предложен новый комплексный алгоритм сжатия маршрутной информации;

- создан собственный формат хранения маршрутной информации.

Работа проведена вплоть до рабочего программного продукта, поэтому может иметь реальное практическое значение, а именно применяться в системах минимизации маршрутной информации в централизованных файловых архивах. В перспективе данное исследование может быть расширено для учета специфики не только сельскохозяйственной, но и коммунальной спецтехники, маршрутного транспорта и т.п.

1 АНАЛИЗ ПРОБЛЕМЫ МОНИТОРИНГА МЕСТОПОЛОЖЕНИЯ ОБЪЕКТОВ СЕЛЬСКОХОЗЯЙСТВЕННОЙ ТЕХНИКИ

1.1 Существующие методы и средства мониторинга местоположения объектов сельскохозяйственной техники с применением ГИС. Онлайн-мониторинг объектов

Итак, во введении в общих чертах была обозначена целесообразность хранения маршрутной информации и ее широкое, разноплановое использование в современных системах защиты, контроля и управления. Рассмотрим подробнее, где, с помощью каких средств и методов, а также с какими целями может применяться указанная маршрутная информация.

Во-первых, и это самое главное, системы контроля текущего местоположения (СКТМ) применяются на специализированной подвижной технике для ее мониторинга при условии, что владелец техники и ее оператор (водитель) являются разными лицами. При таком условии вполне естественно желание собственника (или его представителя, или другого уполномоченного лица) иметь возможность осуществлять проверку использования его технических средств строго в соответствии с их назначением. В первую очередь, это относится к местонахождению подвижного состава техники, например, как в следующих трех случаях:

а) текущее расположение автобусов коммунальных предприятий может проверяться любым человеком как членом общества, которому принадлежит техника; это на сегодняшний день реализуется в виде выделенной веб-страницы, куда может свободно заходить кто-либо из пользователей сети Интернет, и просматривать местонахождение транспорта, которое отслеживается по GPS-трекерам или датчикам системы ГЛОНАСС;

б) также часто во время зимней непогоды обывателей интересует местонахождение специальной снегоуборочной техники, на которую, как и на обычный транспорт, навешивают GPS-модули, и кто-либо на сайтах служб

автодорог (или аналогичных организаций) может посмотреть положение всех ее транспортных средств, оценить эффективность их использования, пожаловаться или поблагодарить руководство организации;

в) чрезвычайно актуальной для России, как для государства, имеющего мощный аграрный потенциал, является задача отслеживания положений сельскохозяйственной техники ее владельцем (или, чаще всего на практике, представителями собственника – уполномоченными лицами). Действительно, к сожалению, в реальности нередка ситуация, при которой водители транспортных средств не выполняют свою работу, топливо могут сливать и продавать, заявляя, что работа выполнена и топливо потрачено по назначению. При этом поле остается не политым, или не перепаханным, и т.п. Конечно, эта ситуация крайне негативно влияет на производительность и другие экономические показатели всего процесса сельскохозяйственного производства и, следовательно, недопустима.

Для корректности дальнейшего изложения материала будем называть ситуацию, когда подвижная техника по воле оператора (водителя) не двигается по заданному маршруту, а используется (или не используется) другим способом, словами «несанкционированное использование подвижной техники» (НИПТ).

Рассмотрим последний случай подробнее, ведь он имеет существенные особенности, по отношению к другим описанным выше ситуациям: здесь маршрутная информация не нужна немедленно, а может пересматриваться отсроченно, через несколько часов, дней или даже недель. В то же время, например, текущая информация о существующем положении общественного транспорта требуется пассажирам непосредственно в данный момент времени, при ожидании на остановке, а после совершения поездки или даже после посадки в автобус моментально теряет свою актуальность. Аналогично и информация о положении снегоуборочной техники важна в основном для текущего момента времени, ведь при планировании маршрута, какими улицами возвращаться домой, или если человек застрял в заносе, важно где

спасательная техника работает в данный момент времени, но после выезда субъекта из заснеженных участков эта информация резко теряет для него свою ценность.

Следовательно, маршрутная информация именно о перемещении сельскохозяйственной техники отличается тем, что должна сохраняться определенный (не малый, не менее величин порядка месяца) интервал времени, а значит, требуются соответствующие накопители информации (архивы), которые, конечно, стоят определенных денег. Очевидно, что увеличение интервала времени ΔT, в течение которого сохраняются маршрутные данные, повышает стоимость системы и уменьшает ее общую экономическую эффективность. С другой стороны, бездумное уменьшение этого параметра может способствовать повышению вероятности возникновения незамеченного НИПТ, что будет нести для собственника ущерб, описанный выше.

Таким образом, очевидно для этого случая наличие конфликтной ситуации: увеличение времени хранения приводит к увеличению стоимости цифровых носителей для ведения архива, но уменьшение этого времени ΔT приводит к увеличению потерь на НИПТ. Очевидно, что выбор наилучшего значения ΔT представляет собой одну из разновидностей математической задачи оптимизации, которая, тем не менее, имеет стохастическую составляющую, поскольку трудно предположить ΔT для произвольного водителя (вероятность даже может отличаться для одного и того же водителя, в зависимости от его текущих жизненных обстоятельств, поэтому обычно лучше минимизировать эту возможность и регулярно сообщать/напоминать водителям о непрерывном мониторинге текущего местонахождения их единицы подвижной техники). Обоснование выбора конкретных значений ΔT будет производиться ниже в соответствующем разделе данной работы.

Если местность имеет сложный рельеф, то использование маршрутной информации может носить и справочный или ориентировочный характер.

Так, один раз проложенный удобный и безопасный маршрут для выполнения определенных сельскохозяйственных работ (например, полива некоторого неровного участка) может повторяться другими водителями в следующие периоды на основе сохраненной маршрутной информации «первопроходца», который мог затратить определенное время на поиск этого наилучшего пути в условиях пересеченной местности. Хотя в целом, этот вариант применения маршрутных сведений является гораздо более экзотическим (хотя и возможным на практике), а главной целью мониторинга остается все же контроль адекватности, санкционированности использования подвижной сельскохозяйственной техники в соответствии с общими потребностями ее владельца, а не ситуативных желаний оператора (водителя).

1.2 Способы снятия маршрутной информации перемещений сельскохозяйственной техники и особенности процесса накопления данных маршрутах

Таким образом, в предыдущем подразделе установлено, что маршрутная информация имеет широкое применение с целью контроля, поэтому следующим шагом должна быть оценка объемов такой информации (причем с выбором конкретных оптимальных значений ΔT), но для этого следует сначала разобраться со способом снятия маршрутной информации, понять ее суть.

В первую очередь, при рассмотрении любых объектов на поверхности земли следует упомянуть о том, что наша планета имеет приближенную форму шара (хотя и несколько сжатого, что не принципиально для целей данной работы), а следовательно, любая точка на ее поверхности может характеризоваться двумя числами: широтой и долготой. Эти величины являются следующими углами:

- который образует радиус-вектор точки на поверхности Земли с плоскостью экватора – это географическая широта;

- который образует диаметральная плоскость, проходящая через данную точку и ось вращения Земли, с одной стороны, и специальной меридиональной плоскостью, также проходящей через ось вращения Земли и точку, долгота которой принята равной нулю (Гринвичский меридиан) – это географическая долгота.

Точность определения широты и долготы в современных системах навигации достаточно высока и позволяет различать точки, лежащие на поверхности Земли друг от друга на расстояниях порядка метра и даже менее (что уже не является существенным в данной работе).

Типичный пример задания координат точки местности, предоставляемой системой Google Maps, приведен на рис. 1.1.

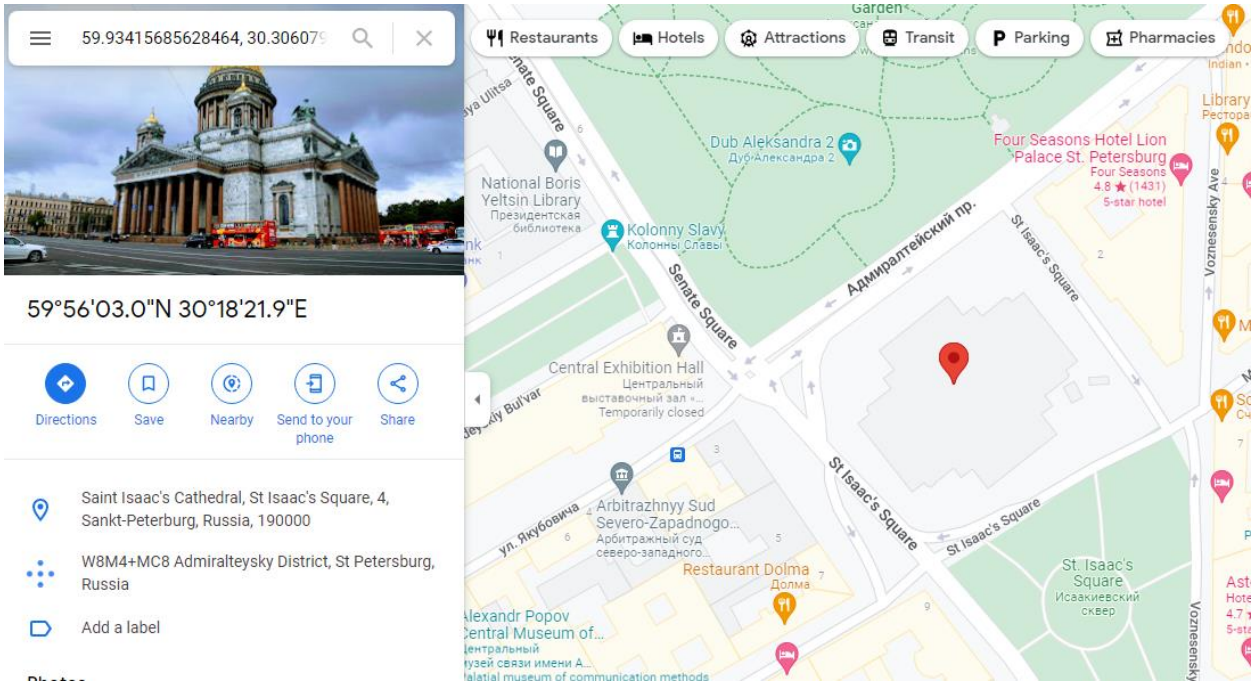


Рис. 1.1 Пример задания координат точки на карте Google

Как видим, координаты представляют собой записи вида дробных чисел с внедрением дополнительных символов типа апострофа, отметки градуса и т.п. (рис. 1.1, в левой части рисунка посередине):

$$59°56’03,0’’N\ 30°18’21,9’’E \qquad (1.1)$$

Указанные координаты читаются следующим образом: 59 градусов, 56 минут, 03.0 секунды северной широты и 30 градусов, 18 минут, 21.9 секунды восточной долготы.

Эта же точка может иметь представление в десятичных долях градуса (на том же рисунке слева сверху мелким шрифтом):

59,934157 30,306079 (1.2)

Здесь первое число – широта – может меняться в пределах от –90 до 90 градусов (положительные значения соответствуют слову «северная»), а второе число – долгота – от –180 до 180 градусов (положительные значения соответствуют восточной долготе, а отрицательные – западной). Переход между двумя формами записи координат легко выполняется, если помнить, что в градусе 60 минут, а в одной угловой минуте содержится 60 угловых секунд, например от (1.1) к (1.2) переход будет следующим:

$$59 + 56/60 + 03{,}0/3600 \approx 59{,}934167,$$

$$30 + 18/60 + 21{,}9/3600 \approx 30{,}306083,$$

что с высокой точностью совпадает с числовыми значениями (1.2).

Обратный переход происходит за два шага (сначала вычисляются минуты, затем – секунды):

[0,934157/(1/60)] = 56’

[(0,934157 – 56/60)/(1/3600)] = 3,0’’

[0,306079/(1/60)] = 18’

[(0,306079 – 18/60)/(1/3600)] = 21,9’’

Здесь, в отличие от предыдущего перевода, когда было получено лишь приблизительное совпадение, имеется точное совпадение со значениями (1.1), а это говорит о том, что формат координат вида (1.2) является первичным для системы Google, а в вид (1.1) они переводятся только в редких случаях для удобного отображения человеку. Форма (1.2) является предпочтительной еще и из соображений объемов памяти, необходимых для ее хранения, ведь для записи координат именно в этой форме необходимо 20 символов (включая раздельную запятую и пробел или символ новой строки),

а для записи в формате «градусы-минуты-секунды» требуется больше (26 символов – с учетом всех разделительных и уточняющих символов). При большом количестве записей разница даже в один символ существенна, поэтому при одинаковых всех других условиях нет смысла выбирать более длинную форму записи, и поэтому в дальнейшей работе будем рассматривать текстовую запись координат только в виде (1.2).

Следует отметить, что самым простым вариантом сохранения любой информации является ее непосредственная запись строкой в текстовом файле, представляющем собой некий протокол (лог). Сам лог будет при этом выглядеть так, как на рис. 1.2.

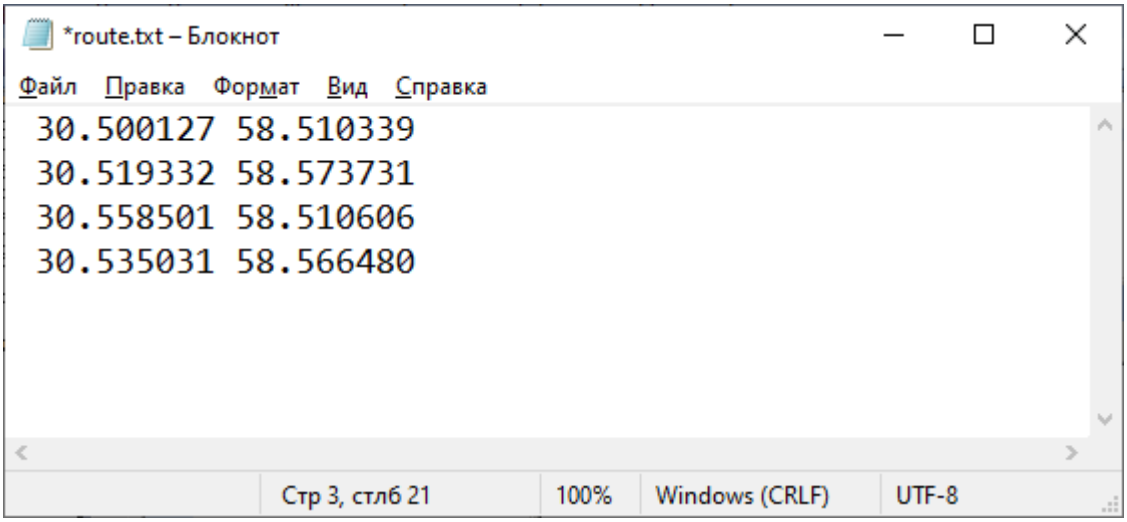


Рис. 1.2 Пример простейшего задания маршрута перемещения

Самый простой способ хранения информации, к сожалению, является самым избыточным, и поскольку информация хранится на цифровых носителях, не представляет никаких трудностей внедрение в систему некоторого подходящего кодирования, которое могло бы значительно уменьшить объем информации, подлежащей хранению. Этот подход аналогичен сжатию информации без потерь и будет подробнее рассмотрен в следующем изложении. Также, аналогично методам сжатия информации с потерями при обработке мультимедийных данных, при сохранении

маршрутной информации можно выбросить часть информации без потерь качества процесса контроля и предотвращения НИПТ. Этот вариант уменьшения объемов архивных данных будет рассматриваться в следующих подразделах. На данный момент же актуально определение объемов такой маршрутной информации в наиболее избыточном, строковом представлении.

Итак, исходя из анализа рис. 1.2 и выражения (1.2) следует, что для сохранения пары координат одной точки на карте, необходимо 20 символов, то есть при использовании кодировки Unicode (2 байта на символ), общее количество байтов будет составлять:

$$\Delta b = 40 \text{ Б}. \qquad (1.3)$$

Теперь следует определиться, насколько часто следует фиксировать текущие координаты. Генерировать маршрутную информацию можно двумя способами с точки зрения интервала выдачи:

- через равные промежутки времени;
- через равные участки пути.

Первый вариант намного проще в технической реализации, ведь без проверки всяческих дополнительных условий (в частности, без знания и учета скорости объекта) через заданный промежуток времени происходит определение текущих координат и передача их в архив.

Важную роль при этом подходе играет выбор интервала времени Δt между фиксациями координат. Оценить его целесообразные значения можно по следующим соображениям:

- скорость комбайна при сборке составляет от 5 км/ч до 15 км/ч (например, при сборе подсолнечника);
- скорость комбайна при перегонах максимально может составлять около 30 км/ч;
- расстояние между последовательными фиксациями координат для целей противодействия НИПТ может составлять порядка 10 м;

- расстояние между последовательными фиксациями координат для целей записи правильного маршрута на частично пересеченной (или неровной и т.п.) местности должно составлять порядка 2 м.

Таким образом, можно выполнить две следующие оценки для определения целесообразных значений Δt.

а) на перегонах:

Δt_1 = 10 м / 30 км/ч = 10 м / 8,3 м/с ≈ 1,2 с.

б) при движении по участкам сложной геометрии:

Δt_2 = 2 м / 5 км/ч = 2 м / 1,4 м/с ≈ 1,4 с.

Как видим, в кардинально разных ситуациях действительно получаем достаточно близкие необходимые временные интервалы между снятиями маршрутной информации. Вводя небольшой запас в 20-40%, что вполне оправдано для инженерных задач, мы можем принять удобную (психологически и для расчетов) цифру для Δt на уровне:

$$\Delta t = 1 \text{ с.} \qquad (1.4)$$

Таким образом, для дальнейшего анализа способа фиксации положения через равные промежутки времени принимаем значение этого промежутка согласно (1.4).

При продолжении анализа становится очевидным принципиальный недостаток данного подхода, чрезвычайно ярко проявляющийся при записи местоположения объекта, который неподвижно стоит в течение длительного интервала времени: если, как принято выше, положение записывать через каждые Δt = 1 секунду, то за один час (1 ч = 3600 с) простоя наберется $n_1 = 3600/\Delta t$ = 3600 одинаковых (т.е. фактически избыточных) записей. Учитывая (1.3), количество чрезмерной информации в час составит:

$$B_1 = n_1 \cdot \Delta b = 3600 \cdot \Delta b = 3600 \cdot 40 = 144000 \text{ Б} = 140 \text{ кБ}.$$

При альтернативном подходе, т.е. использовании данных о скорости объекта, в том числе обстоятельства (правила), что при нулевой скорости

координаты в архиве можно не дублировать, количество записанных байтов за все время простоя составит просто величину Δb (1.3).

В течение рабочего дня (считаем продолжительность смены 12 ч) при первом подходе будет сгенерировано:

$$B_{см} = 12B_1 = 12 \cdot 140 \text{ кБ} = 1680 \text{ кБ} = 1{,}65 \text{ МБ}. \quad (1.5)$$

С одной стороны, эта цифра не очень большая, но за сезон общее количество информации для одного комбайна будет достигать сотен мегабайт, а при условии использования соответствующих методов она может быть уменьшена в несколько раз. Соответственно и объемы, а следовательно и количество устройств для хранения информации может уменьшаться, повышая экономическую эффективность всей системы. Таким образом, целесообразно проанализировать весь процесс генерации и сохранения маршрутной информации на предмет применения специальных методов, направленных на уменьшение объемов такой информации, и перейдем к их рассмотрению.

1.3 Существующие способы оптимизации маршрутных данных, которые могут быть применены в работе с учетом статистики задержек измерений

В предыдущем подразделе в общих чертах описан процесс генерации и хранения маршрутных данных, выполнены оценки объемов информации, которая при этом возникает, указаны возможные пути усовершенствования всего процесса, в частности применение методов уменьшения объема архивных данных без потерь, с потерями, а также метод генерации маршрутных данных , параметры которого зависят от скорости движущегося объекта.

В первую очередь (и это основной способ, применяемый на практике), маршрутные данные, в каком бы формате они ни хранились, могут быть сжаты без потерь с помощью обычных программ-архиваторов. При этом уже

можно добиться значительной экономии дискового пространства, которую можно оценить с помощью следующих выкладок.

Сначала нужно получить большой объем маршрутной информации, которую при отсутствии реальных DATA-файлов можно сделать путем интеллектуального моделирования. Вспомним, что длина дуги (которой, по сути является прямолинейный путь на поверхности земли) определяется как простое произведение радиуса на центральный угол, который должен быть выражен в радианах:

$$s = \alpha \cdot R \qquad (1.6)$$

Оценивая радиус Земли на уровне 6370 км, по (1.6) можно оценить, что одной миллионной градуса на поверхности земли соответствует расстояние:

$$(0{,}000001° \cdot \pi/180) \cdot 6370000 \approx 0{,}1 \text{ м}, \qquad (1.7)$$

Соответственно, одной стотысячной соответствует расстояние около одного метра:

$$(0{,}00001° \cdot \pi/180) \cdot 6370000 \approx 1 \text{ м}, \qquad (1.8)$$

Расстоянию 1 км будет соответствовать угол около одной сотой доли градуса:

$$(0{,}01° \cdot \pi/180) \cdot 6370000 \approx 1000 \text{ м}, \qquad (1.9)$$

Расстоянию 10 км будет соответствовать угол около одной десятой доли градуса:

$$(0{,}1° \cdot \pi/180) \cdot 6370000 \approx 10 \text{ км}, \qquad (1.10)$$

Следовательно, если размеры среднего поля принять порядка километра, то разница между угловыми координатами для крайних противоположных точек будет составлять сотые доли градуса. Опираясь на эту информацию, можно смоделировать большой объем маршрутной информации, например, в среде MathCad по следующим зависимостям:

$$i := 0..32767$$

$$Coords_{i,0} := 30.5 + \frac{rnd(10^5)}{10^6}$$

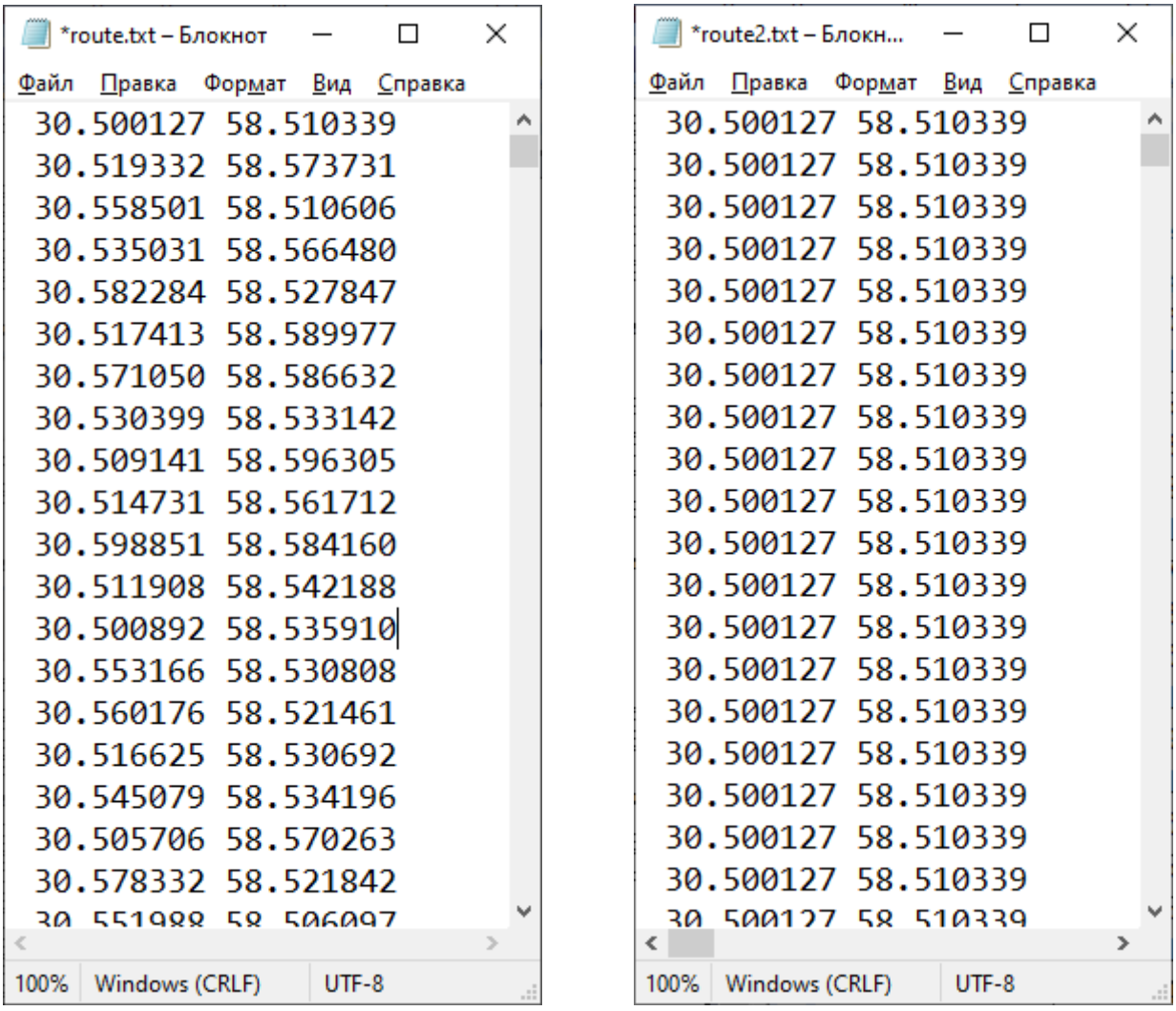
$$Coords_{i,1} := 58.5 + \frac{rnd(10^5)}{10^6}$$

$$out := WRITEPRN("route.txt", Coords)$$

Здесь координаты 30,5° восточной долготы и 58,5° северной широты приняты за крайнюю (левую нижнюю) точку поля.

В результате получен набор координат, близкий к реальному – рис. 1.3, *а*. Проведенная процедура моделирования необходима для правильной оценки возможной степени сжатия файлов с координатами, ведь эта степень зависит от структуры сжимаемой информации (как пример, на рис. 1.3, *б* показан файл примерно того же размера, но полученный путем простого копирования одинаковых координат). Как наглядно показано ниже, степень сжатия двух наборов данных будет принципиально различной, поэтому для получения реального показателя возможной степени сжатия и нужно было провести моделирование.

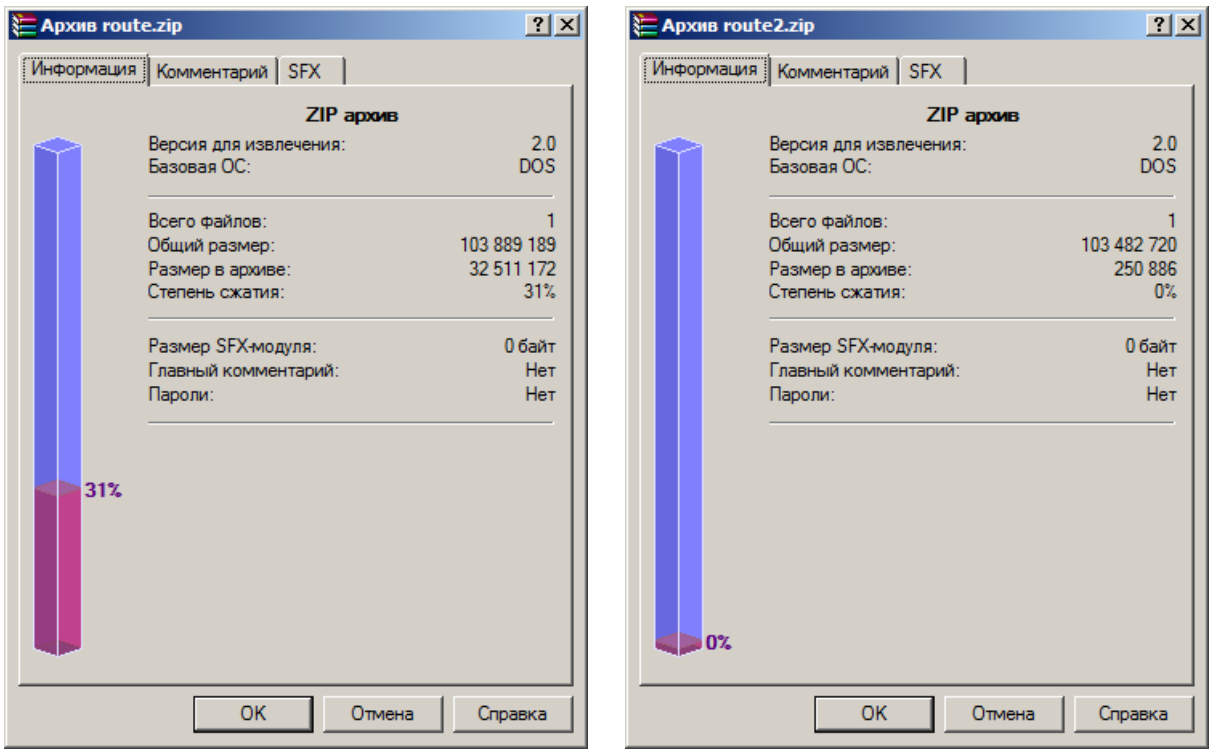
Файл с маршрутной информацией route.txt был направлен на вход архиватора, результаты которого можно видеть на рис. 1.4, *а*. Видно, что маршрутная информация, записанная в виде рис. 1.3, *а* достаточно хорошо подлежит сжатию до величины порядка 30% от первоначального объема (т.е. становится на 70% меньше). В то же время на рис. 1.4, *б* показано для сравнения, что сжатие файла вида 1.3, *б* дает нереальную степень сжатия близкую к 100%, а это и подтверждает необходимость проведения специальной процедуры моделирования маршрутной информации для ее дальнейшего исследования.



а) *б*)

Рис. 1.3 Примеры маршрутной информации: *а* – смоделированная информация, параметры которой близки к характеристикам реальной маршрутной информации при перемещении комбайна по полю; *б* – файл, полученный простым копированием одного набора координат

Следует обратить внимание, что рассматриваемое преобразование информации представляет собой сжатие без потерь (или другими словами без изменений) – т.н. архивирование, потому что в результате обратных операций можно получить первоначальную информацию абсолютно в той же форме, что и перед сжатием.



а) *б*)

Рис. 1.4 Примеры сжатия маршрутной информации простым архивированием: *а* – для смоделированной информации, параметры которой близки к реальным значениям; *б* – для файла, полученного простым копированием одного и того же набора координат

1.4 Анализ методов и моделей минимизации объемов маршрутной информации в зависимости от особенностей перемещения с/х техники

В предыдущем подразделе рассмотрены возможные способы уменьшения объемов маршрутной информации, не привязываясь к временным характеристикам «поездки» и другим особенностям процесса; предполагается, что координаты местоположения объекта снимаются через одинаковые промежутки времени, независимо от того, движется ли объект вообще, или говоря более обще, какова его скорость. Однако важным аспектом этого процесса является зависимость параметров сохранения данных от скорости. Действительно, очевидна избыточность сохранения

координат через каждую секунду, или даже минуту, если эти координаты вообще не изменяются (комбайн припаркован возле леска).

Таким образом, необходимо ввести определенную зависимость времени обновления координат Δt от особенностей движения средства, в частности, от его скорости движения, которая по существу является скоростью изменения координат объекта. Следовательно, записывать координаты следует только в том случае, если они достаточно изменились. Этап, на котором выполняется уменьшение информации ввиду разной скорости движения, может быть разным:

- непосредственно при записи;
- при помещении маршрутной информации в архив.

Оба подхода имеют свои плюсы и минусы. Если для сжатия применяется более или менее интеллектуальный алгоритм, который требует значительных вычислений, то его внедрение на стороне подвижного устройства (непосредственно на комбайне) возможно только при значительном усложнении соответствующей электронной базы, а значит и ее удорожание. Плюсом является то, что для хранения данных на движущемся объекте (т.е. еще до внесения в централизованный архив) требуется примерно вдвое меньше объемов дискового пространства (флэш-памяти). Экономия памяти будет при этом составлять не более 50%. Таким образом, при введении зависимости параметров маршрутной информации от скорости объекта непосредственно на нем (еще до внесения в архив) экономия достигается за счет использования накопителей вдвое меньшего объема, но большие затраты обусловлены необходимостью установки более полнофункциональных вычислительных устройств (например, процессоров общего назначения вместо микроконтроллеров). Учитывая, что для установки обычных процессоров необходим еще целый комплекс аппаратного обеспечения (материнская плата, оперативная память, платы расширения), в таком решении дополнительные затраты значительно превысят экономию.

Таким образом, очевидна целесообразность выполнения любой интеллектуальной обработки данных на стороне сервера, непосредственно перед помещением в архив. На движущихся же объектах информация будет храниться в несжатом во времени виде, что совершенно несущественно, учитывая огромные емкости современных флэш-карт, которые целесообразно применять для временного локального хранения информации.

Так, стоимость простейшей SD-карты объемом 16 Гб на момент написания работы составляет около 3 у.е., а 32 Гб версии – около 5 у.е. Относительная разница может показаться большой (почти в два раза), но абсолютная величина разницы 2 у.е. абсолютно несущественна и на порядки (т.е. до 100 раз) меньше стоимости дополнительного оборудования, которое нужно установить на каждый комбайн, чтобы проводить непосредственно на нем интеллектуальную обработку данных (сжатие) перед записью в локальную флеш-память малого объема. Таким образом, ввиду низкой стоимости флэш-памяти, на комбайне достаточно проводить простую фиксацию (запись без сжатия) во флэш-памяти текущих координат с помощью дешевого микроконтроллера, а затем сбрасывать их на центральный сервер, где его вычислительными ресурсами следует производить сжатие и записывать результат в файловый архив.

Таким образом, даже при ежесекундной фиксации координат в течение суток будет получено максимум вдвое больше данных, чем при (1.5), то есть около 3,3 Мб. Даже при объеме флэш-карты 16 Гб, она вместит 5 тыс. дней записи, то есть около 14 лет непрерывной записи. И повторимся, при перенесении этой информации в архив, где будут содержаться данные от сотен (а то и тысяч) комбайнов, вопросы максимального сжатия становятся чрезвычайно актуальными и подлежат более эффективному решению.

Возвращаясь к вопросу зависимости координат от скорости, можно сказать, что она будет внедряться в конечный результат косвенным образом, причем предложение конкретного метода будет ниже. На данный момент следует лишь констатировать, что такая зависимость должна быть учтена и

может придать существенную экономию необходимого объема дискового пространства.

2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ МИНИМИЗАЦИИ МАРШРУТНЫХ ДАННЫХ О ПЕРЕМЕЩЕНИИ СЕЛЬСКОХОЗЯЙСТВЕННОЙ ТЕХНИКИ

2.1 Обоснование выбора базового способа минимизации маршрутных данных

Для рассматриваемой системы могут быть применены достаточно разнообразные методы минимизации маршрутных данных. Рассмотрим главные особенности основных из них.

В первую очередь, возможно применение общих методов сжатия информации без потерь – архивирования. Анализ его особенностей и путей совершенствования в методе архивирования выходит далеко за рамки данной работы и представляет собой огромный пласт знаний из области теории информации. Таким образом, в данном исследовании архивирование будем применять в соответствии со стандартными методами (например, по алгоритму ZIP), причем в последнюю очередь, когда все другие методы сжатия уже использованы.

Во-вторых, можно рассматривать методы, принимающие во внимание физическую сущность данных, подлежащих сжатию. А именно практически все (кроме некоторых служебных сведений) числа, которые имеются в файле с маршрутной информацией, являются парами координат точек земной поверхности, причем точек соседних. Из отдельных последовательно расположенных точек формируется ломаная линия, которая, при достаточной плотности узловых точек, может достаточно точно представляться кривой, сплайном. Следовательно, возникает задача: по набору последовательно размещенных точек получить коэффициенты сплайна соответствующей степени, причем набор этих коэффициентов должен занимать меньшее количество байтов, чем набор координат точек, по которым построен сплайн.

Напомним, что координаты двух точек точно «ложатся» на прямую линию, уравнение которой состоит из двух коэффициентов ($y = ax + b$, два

коэффициента a и b). Три точки точно размещаются на кривой второго порядка – обычной параболе, имеющей три коэффициента ($y = ax^2 + bx + c$, три коэффициента a, b, c). Так и далее: для точного размещения n+1 точек на кривой соответствующего n-ного порядка требуется ровно n+1 коэффициентов многочлена n-ой степени:

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0 = \sum_{k=0}^{n} a_k x^k \qquad (2.1)$$

Таким образом, если использовать уравнение многочлена n-ного порядка, то сохранять необходимо будет только координаты x_i, а все значения y_i (которых всего n штук) будут рассчитываться по формуле (2.1) и их сохранять больше не нужно. Но, кроме всех x, хранению подлежат все коэффициенты a_k выражения (2.1), которых всего также n штук. Таким образом, если точность задания коэффициентов a_k не ниже точности задания y_i, то никакая экономия от такой замены не будет наблюдаться. В противном случае (при уменьшении точности хранения коэффициентов a_k), учитывая, что на основе этих коэффициентов арифметическим путем вычисляются значения y_i, то и эти результирующие значения будут определяться неточно (поскольку в (2.1) коэффициенты входят в первых степенях, то и неточность определение разрядов будет сохраняться: например, если какой-то коэффициент a_k имеет неточность во втором разряде после запятой, то и общий результат, т.е. y_i, будет иметь такую же неточность во втором разряде). Учитывая общее большое количество коэффициентов a_k и возможность существования неточностей достаточно больших порядков у каждого из них, общие погрешности определения координат y_i по (2.1) могут исказить маршрут так, что пользоваться им (просматривать, проверять и т.п.) будет совершенно невозможно.

Таким образом, переходить к сплайн-апроксимации в задачи хранения маршрутной информации нецелесообразно. Рассмотрим другой, более универсальный способ преобразования информации, который может быть полезен для целей данного исследования.

Как известно, кроме множества действительных чисел, которые могут использоваться для измерения физических величин, существует целый класс комплексных чисел, в которых участвует специальная комплексная единица. Из-за мнимого характера этого символа и, комплексные числа нельзя использовать для описания реальных явлений. Однако такие числа нашли широкое применение в задачах преобразования сигналов и в более широком смысле информации.

Суть многочисленных «комплексных» преобразований (их называют преобразованиями Фурье, хотя исторически первое название «преобразования Фурье» было использовано для преобразования на основе тригонометрических функций синус и косинус) заключается в наличии двух правил:

- как на основе набора действительных чисел получить соответствующий набор комплексных чисел;

- как на основе набора комплексных чисел вернуться в исходный набор действительных чисел.

Преобразование Фурье целесообразно выполнять из-за того обстоятельства, что некоторые действия, которые очень сложно выполняются над действительными числами (например, такой операцией является процесс решения дифференциального уравнения), намного легче проводятся с их комплексными образами (дифференциальное уравнение заменяется алгебраическим, что всегда решается значительно проще). Следовательно, полезная схема применения интегральных преобразований типа Фурье может быть следующей:

а) выполнить комплексное преобразование восходящего действительного сигнала $f_1(t)$ (информации), или, как говорится, оригинала, и получить соответствующее ему комплексное выражение $F_1(p)$ или изображение:

$$F_1(p) = \int_T K(p,t)\, f_1(t) dt, \qquad (2.2)$$

где $K(p, t)$ называют ядром интегрального преобразования;

T – область интегрирования в реальной области, принятая для данного ядра $K(p, t)$.

б) выполнить над комплексным образом $F_1(p)$ преобразованного входного сигнала необходимые действия, направленные на решение начальной задачи по обработке оригинального входного сигнала и получить измененный комплексный образ $F_2(p)$;

в) выполнить обратное комплексное преобразование, которое по измененному комплексному образу $F_2(p)$ воспроизведет нужный выходной сигнал $f_2(t)$ в обычной, действительной форме:

$$f_2(t) = \int_P K^{-1}(p,t) F_2(p) dp. \qquad (2.3)$$

Таким образом, путь выполнения операций

$$f_1(t) \to F_1(p) \to F_2(p) \to f_2(t) \qquad (2.4)$$

при применении интегральных преобразований типа Фурье значительно легче, чем прямая переработка информации $f_1(t) \to f_2(t)$.

Теоретически, схема (2.4) может пригодиться и для организации системы сжатия маршрутной информации, но только при выполнении следующих условий:

а) если преобразованный сигнал $F_1(p)$ можно каким-то способом уменьшить в объеме, получив измененное выражение $F_2(p)$;

б) если размер $F_2(p)$ в памяти ЭВМ стал меньше размера исходного действительного сигнала $f_1(t)$. Другими словами должно выполняться:

$$\text{sizeof}(F_2(p)) < \text{sizeof}(f_1(t)), \qquad (2.5)$$

где оператором sizeof() обозначено взятие размера в байтах. Если (2.5) не выполняется, то во всем преобразовании просто нет смысла, поскольку никакое сжатие не происходит;

в) если по измененному изображению $F_2(p)$ можно восстановить сигнал $f_2(t)$, который, вообще говоря, будет отличаться от исходного сигнала $f_1(t)$,

так как у них разные изображения $F_1(p)$ и $F_2(p)$. Тем не менее, эти два сигнала должны быть близки друг другу. Иными словами, должно быть:

$$\text{diff}\,(f_1(t), f_2(t)) < \Delta f, \qquad (2.6)$$

где оператором diff обозначена операция нахождения разницы между двумя сигналами, а Δf – пороговое допустимое значение этой разницы. В качестве конкретной операции нахождения разницы можно взять интегрирование арифметической разности двух функций по их области определения:

$$\text{diff}\,(f_1(t), f_2(t)) \overset{def}{=} \frac{1}{T}\int_T \left| f_1(t) - f_2(t) \right| dt. \qquad (2.7)$$

Для определения возможности применения этой схемы сжатия в реальной системе хранения маршрутной информации следует проанализировать ее потенциальную эффективность. Для этого зададимся стандартным преобразованием Фурье, в котором ядро имеет вид $K(p, t) = e^{-pt}$, и реализуем интегральное преобразование в уже используемой системе Mathcad.

Эта программная среда содержит функцию fft(), осуществляющую «быстрое» преобразование Фурье, особенностью которого является то, что входной вектор чисел должен содержать 2^m элементов (т.е. их количество должно быть степенью двойки). Учитывая, что поток координат маршрутной информации непрерывный, не представляет особой проблемы его разбиение на части по 2^m чисел. При этом на выходе функция fft() дает 2^{m-1} комплексных чисел той же точности, а поскольку каждое комплексное число является парой двух действительных чисел, общее количество информации при таком преобразовании ($f_1(t) \rightarrow F_1(p)$) не изменяется. В то же время изображение $F_1(p)$ можно сжать, реализовав преобразование $F_1(p) \rightarrow F_2(p)$. Например, это возможно, если оценивать модуль каждого комплексного числа и числа с малыми модулями, меньшими предельного значения ΔF (т.е. вносящие в общий результат очень малый вклад) просто обнулять (уменьшая таким образом, количество информации в $F_2(p)$) . После выполнения такой

процедуры сжатое изображение $F_2(p)$ можно подвергнуть обратному преобразованию Фурье и получить новый оригинал $f_2(t)$, который, как уже отмечалось выше, будет несколько отличаться от исходного сигнала $f_1(t)$. Задача здесь состоит в оценке, как выбор порога ΔF влияет на разницу (2.4) между двумя действительными сигналами, возникающую при определенном ΔF.

Для исследования указанных вопросов реализованы простые функции пользователя системы MathCad, код которых приводится ниже:

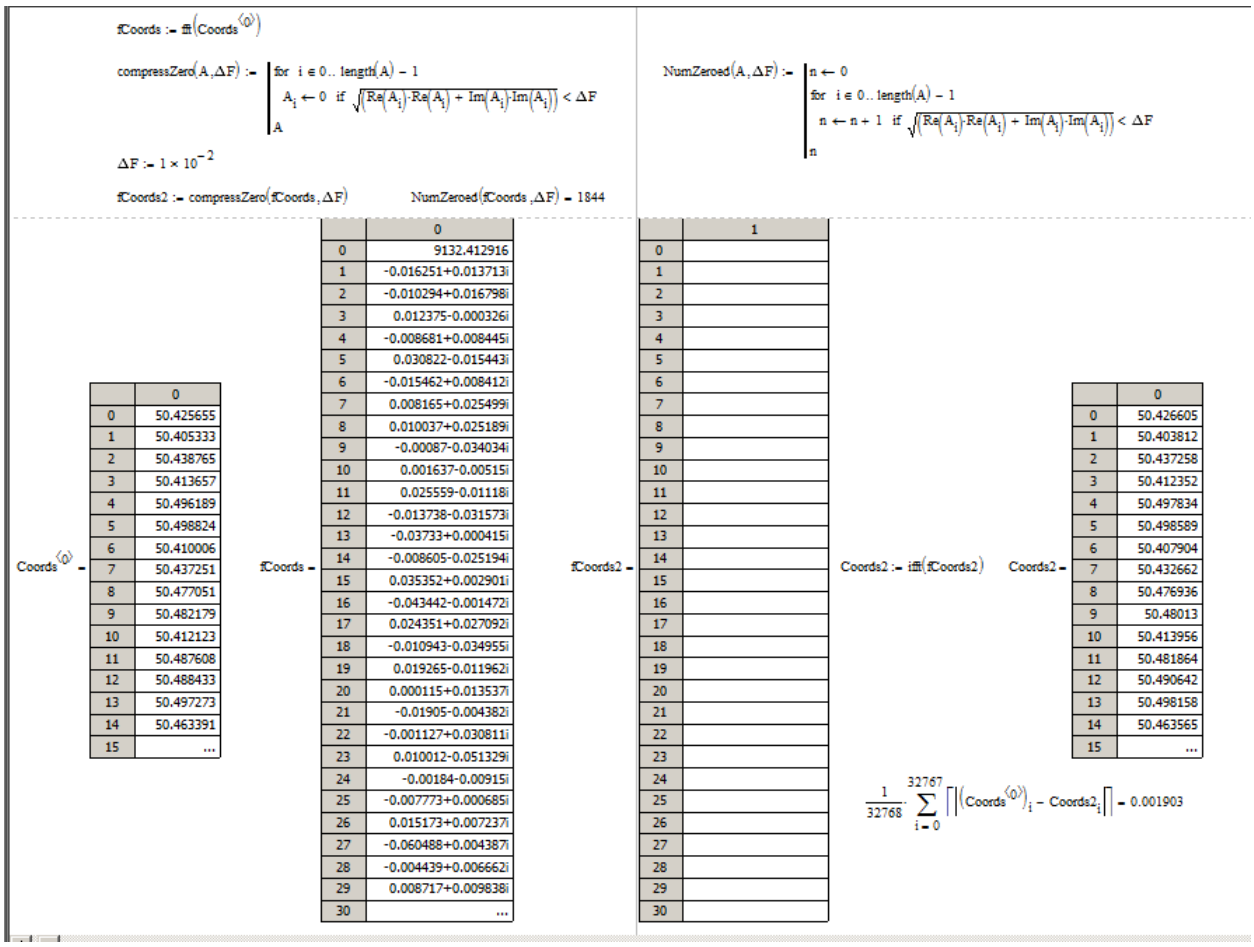
а) функция compressZero(fCoords, ΔF), обнуляющая все комплексные числа в матрице fCoords, модуль которых меньше заданного порога ΔF:

$$\text{compressZero}(A, \Delta F) := \left| \begin{array}{l} \text{for } i \in 0..\text{length}(A) - 1 \\ \quad A_i \leftarrow 0 \text{ if } \sqrt{\left(\text{Re}\left(A_i\right)\cdot\text{Re}\left(A_i\right) + \text{Im}\left(A_i\right)\cdot\text{Im}\left(A_i\right)\right)} < \Delta F \\ A \end{array} \right.$$

б) функция NumZeroed(fCoords, ΔF), которая считает количество комплексных чисел в матрице fCoords, модуль которых меньше заданного порога ΔF:

$$\text{NumZeroed}(A, \Delta F) := \left| \begin{array}{l} n \leftarrow 0 \\ \text{for } i \in 0..\text{length}(A) - 1 \\ \quad n \leftarrow n + 1 \text{ if } \sqrt{\left(\text{Re}\left(A_i\right)\cdot\text{Re}\left(A_i\right) + \text{Im}\left(A_i\right)\cdot\text{Im}\left(A_i\right)\right)} < \Delta F \\ n \end{array} \right.$$

После применения этих функций (пример применения приведен на рис. 2.1) получаем следующие результаты для разных величин порогов ΔF, которые сведены в табл. 2.1.



	0
0	50.425655
1	50.405333
2	50.438765
3	50.413657
4	50.496189
5	50.498824
6	50.410006
7	50.437251
8	50.477051
9	50.482179
10	50.412123
11	50.487608
12	50.488433
13	50.497273
14	50.463391
15	...

	0
0	9132.412916
1	-0.016251+0.013713i
2	-0.010294+0.016798i
3	0.012375-0.000326i
4	-0.008681+0.008445i
5	0.030822-0.015443i
6	-0.015462+0.008412i
7	0.008165+0.025499i
8	0.010037+0.025189i
9	-0.00087-0.034034i
10	0.001637-0.00515i
11	0.025559-0.01118i
12	-0.013738-0.031573i
13	-0.03733+0.000415i
14	-0.008605-0.025194i
15	0.035352+0.002901i
16	-0.043442-0.001472i
17	0.024351+0.027092i
18	-0.010943-0.034955i
19	0.019265-0.011962i
20	0.000115+0.013537i
21	-0.01905-0.004382i
22	-0.001127+0.030811i
23	0.010012-0.051329i
24	-0.00184-0.00915i
25	-0.007773+0.000685i
26	0.015173+0.007237i
27	-0.060488+0.004387i
28	-0.004439+0.006662i
29	0.008717+0.009838i
30	...

	1
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	

	0
0	50.426605
1	50.403812
2	50.437258
3	50.412352
4	50.497834
5	50.498589
6	50.407904
7	50.432662
8	50.476936
9	50.48013
10	50.413956
11	50.481864
12	50.490642
13	50.498158
14	50.463565
15	...

Рис. 2.1 Пример применения интегрального преобразования для сжатия маршрутной информации (для значения порога $\Delta F = 10^{-2}$)

Табл. 2.1 Параметры системы сжатия маршрутных данных в зависимости от значения порогового значения ΔF

№	ΔF	Объем фрейма, отсчетов	Обнуленных фреймов	На сколько уменьшается объем, %	Среднее отклонение, diff, градусов	Среднее отклонение, метров
1.	$0{,}5 \cdot 10^{-2}$	32768	460	1,4	0,0005	50
2.	10^{-2}	32768	1821	5,5	0,002	200
3.	$2 \cdot 10^{-2}$	32768	6238	19	0,007	700
4.	$3 \cdot 10^{-2}$	32768	10797	33	0,013	1300

Анализ табл. 2.1, к сожалению, показывает, что применять эффективное сжатие на основе интегральных преобразований по схеме (2.4) при условии выполнения (2.6)-(2.7), т.е. при обеспечении достаточного качества восстановленного сигнала, не представляется возможным.

Действительно, в первой строке показаны параметры варианта расчетов с порогом $\Delta F = 0{,}5 \cdot 10^{-2}$, при котором отбрасыванию подлежат 460 комплексных чисел, то есть около 1,4%. Среднее отклонение одной точки при обратном преобразовании, т.е. при воспроизведении $f_2(t)$, составляет около 50 м, что находится на грани точности задания маршрутной информации (даже немного выходит за эту грань, поскольку в условиях пересеченной местности случайное отклонение точки на величину 50 метров полностью искажает содержание маршрутной информации). Одним словом, среднее отклонение зафиксированных точек маршрута, возникающее при сжатии с помощью интегрального преобразования, должно быть меньше, то есть порог ΔF должен быть уменьшен. Однако, даже при малом рассматриваемом значении $\Delta F = 0{,}5 \cdot 10^{-2}$ объем информации уменьшается на всего лишь около 1%, а если порог уменьшить, то и объем отбрасываемой информации станет еще меньше одного процента, что абсолютно неэффективно для задач сжатия данных.

Подводя итог, можно сказать, что ни метод сплайн-апроксимации, ни методы интегрального преобразования Фурье не дают эффективных результатов для задачи сжатия маршрутной информации перед ее помещением в архивы. В то же время, учитывая сведения, приведенные в первой главе, можно заключить, что наиболее оптимальным способом обработки маршрутной информации является рассмотрение и учет самих ее свойств и сжатие на основе их анализа, без необходимости сложной математической обработки.

2.2 Анализ возможностей усовершенствования выбранного метода минимизации маршрутных данных

Итак, минимизацию будем проводить на основе анализа свойств маршрутной информации сравнительно простыми с математической точки зрения методами.

В первую очередь следует обратить внимание, что значения координат, предоставляемых GPS-датчиком, записаны в долях градуса и содержат миллионные его доли (см. напр. (1.2)). В расчете (1.7) было установлено, что одной миллионной доле градуса примерно соответствует расстояние 0,1 м, что совершенно не требуется для рассматриваемой предметной отрасли (точность слишком высокая). Это значит, что совершенно без потерь важной содержательной информации можно выбросить миллионные цифры (разряды), ограничившись стотысячными долями градуса, дающими расстояния между двумя последовательными положениями порядка метра. Такое уменьшение количества отображаемых разрядов для обоих координат предоставит уменьшение объемов хранящейся информации примерно на $1/8 \approx 12\%$, что совсем не мало. Дальнейший отброс разрядов уже не является обоснованным, поскольку будет способствовать возникновению ошибок в определении местоположения на целые метры в соответствии с (1.8), что может быть критическим для описания особенностей маршрута, например, на частично пересеченной местности.

Если посмотреть на координаты с другой стороны (со стороны высших разрядов), можно видеть повторение другого типа, когда повторяется значение собственно градусов, а также их высших разрядов десятичных дробей (например, на рис. 1.4, *а* все записи долготы начинаются с цифр 30.5 , а широты начинаются с цифр 50.4). Очевидно, что наличие таких повторов можно эффективно использовать для сжатия маршрутной информации. Например, используя тег BASE (или для экономии просто B), можно задавать базовые значения долготы и широты, к которым будут прибавляться те

значения, которые массово указываются в строках ниже. Следующий тег типа BASE будет встречаться при изменении той части координаты, которая прописана в этом теге.

Например, при работе в рамках одного сельскохозяйственного поля разница между угловыми перемещениями по долготе и широте не будет превышать 0,1°, поскольку согласно (1.10) такое угловое расстояние соответствует линейному перемещению около 10 км. Даже если изменение угловых координат и будет происходить в пределах 0,1 градуса, это будет происходить очень редко и будет перекрываться вводом тега типа BASE.

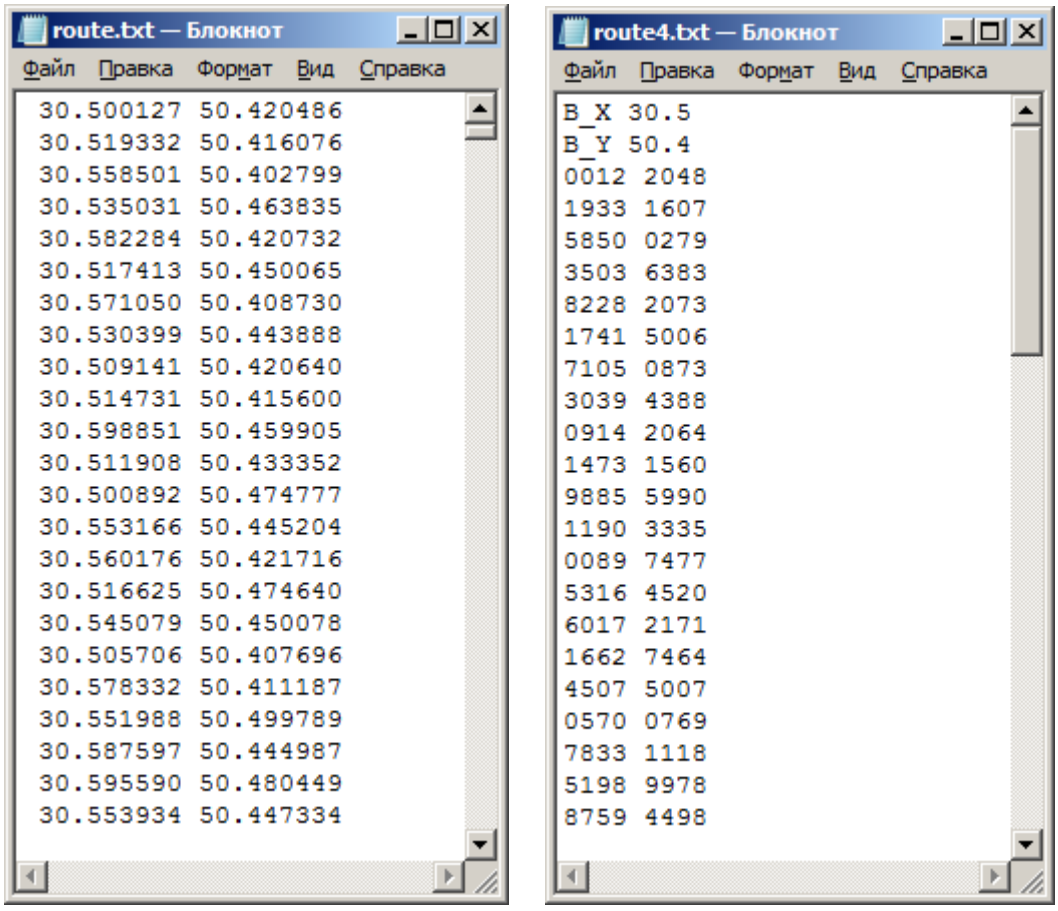
Еще более гибкий подход можно внедрить, рассматривая отдельно два базовых тэга:

а) B_X – для смены базовой части долготы;

б) B_Y – изменение базовой части широты.

При этом фрагмент маршрутной информации будет смотреться как на рис. 2.2, *б*. Для сравнения на рис. 2.2, *а* показан фрагмент того же файла до отбрасывания лишних миллионных долей градуса и кодирования базовой части координат.

При таком преобразовании уменьшение объемов информации происходит примерно вдвое, причем фактически без потерь, так как выполненный отброс миллионных долей градуса является несущественным для рассматриваемой предметной отрасли. Для полноты картины следует отметить, что степень сжатия при архивировании нового формата как на рис 2.2 *б* становится несколько меньше, чем на рис. 2.2, *а* (около 45-50%), но все равно небольшое уменьшение степени сжатия архиватором (с 45% до 30%) компенсируется значительно б*о*льшим уменьшением объема исходных данных (приблизительно в два раза) и в целом степень сжатия информации при применении обоих методов вместе больше, чем при применении каждого из них отдельно.



а) *б*)

Рис. 2.2 Преобразование координатной информации: *а* – оригинал, *б* – та же информация после отбрасывания миллионных долей и ввода базовой части каждой координаты

Кроме вышеуказанных действий, следующий шаг сжатия может быть реализован через перевод каждой четырехзначной координаты (рис. 2.2, *б*) в числовую форму. Поскольку цифр всего 4, то каждое такое число может помещаться в 2-байтовую переменную типа short int. Следовательно, вместо четырех байтов, отводимых в текстовом файле под хранение слова из 4 цифр, в конечном сжатом формате на каждую координату будет отводиться 2 байта, за счет чего весь объем информации уменьшается почти вдвое (за исключением служебной информации типа тегов BASE).

Здесь следует обратить внимание на то, почему в базу были включены десятичные доли координат (кроме самих координат, конечно), а сотые, тысячные, десятитысячные и стотысячные доли остаются и формируют

указанные четырехзначные числа. Задача выбора разрядов, которые бы включались в базу, является классической задачей оптимизации с конфликтующими требованиями.

Во-первых, следует стараться, чтобы служебные записи типа слова BASE (или B_X и B_Y) встречались как можно реже, поскольку большое количество будет увеличивать общий объем файла. Следовательно, в них следует пытаться включить как можно меньше разрядов, например сами координаты, без дробной части вообще. Однако, при этом для каждой записи местоположения объекта нужно будет хранить целых 5 цифр, то есть максимально – число 99999, для размещения которого в памяти ЭВМ нужно больше 2 байта. Поскольку создать переменную размером 3 байта стандартными способами невозможно (а использование такой структуры/записи будет крайне сложно технически), то реально для таких чисел следует использовать 4-байтовые переменные типа int. Таким образом, экономя на тэгах типа BASE, больше теряется на представлении каждой координаты. Если же в базу включить помимо целых значений, еще и десятичные разряды, то тег BASE будет встречаться чуть чаще, однако для каждой координаты останется 4 разряда, то есть максимальное число 9999, вмещающееся в 2-байтовую переменную типа short int. Таким образом, достигается максимальная экономия дискового пространства при хранении целого файла. Если далее увеличивать количество разрядов, которые будут включаться в базу, то при включении сотых долей градуса тег BASE будет встречаться еще чаще, однако на каждую координату все равно нужно будет отводить 2 байта (чтобы хранить в таких переменных числа от 000 до 999 не помещающиеся в один байт). Таким образом, этот вариант однозначно хуже предыдущего. Последним вариантом, который целесообразно рассмотреть, является включение в базу еще и тысячных долей градуса, при которых на каждую координату остается всего два разряда (т.е. это числа от 00 до 99), которые можно вместить в один байт. Однако, при изменении даже тысячной доли градуса следует будет вводить отдельный тег типа BASE, а 0,001°

соответствует линейному перемещению по поверхности Земли примерно на 100 м. Таким образом, если соседние точки маршрута размещаются гораздо ближе друг к другу, чем 100 м, то целесообразным является введение в тэг BASE еще и тысячных долей градуса, а если соседние точки размещены на расстояниях порядка десятков метров (т.е. тысячные доли градуса меняются довольно часто), то BASE целесообразно ограничивать десятыми долями градуса. Вся эта информация в сводном виде представлена в табл. 2.2. В таблице приведены половинчатые значения средних линейных расстояний, на которые следует сдвинуться с места, чтобы изменилась соответствующая база, поскольку движение возможно как в одну, так и в другую сторону.

Табл. 2.2 Анализ эффективности распределения разрядов координат между базой и отдельными записями (радиус Земли взят равным 6370 км)

№	Число дробных разрядов в базе	Расстояние, на которое следует сдвинуться, чтобы изменилась база, °, м	Число разрядов в каждой отдельной координате	Объем каждой отдельной координаты, байт	Целесообразность использования на практике
1.	0, числа вида XX	0,5°, 55589 м	5	4	–
2.	1, числа вида XX,X	0,05°, 5559 м	4	2	+
3.	2, числа вида XX,XX	0,005°, 556 м	3	2	–
4.	3, числа вида XX,XXX	0,0005°, 56 м	2	1	+
5.	4, числа вида XX,XXXX	0,00005°, 6 м	1	1	–

Таким образом, подводя итог распределения разрядов координат между базой и отдельными записями, можно сказать, что для целей фиксации маршрутной информации сельскохозяйственной техники лучше всего подходит вариант №4.

Все приведенные усовершенствования относительно независимы, поэтому могут быть применены все вместе, в комплексе, формируя соответствующий алгоритм, что и будет реализовано в данной работе, а подробно рассмотрено в следующем подразделе.

2.3 Разработка алгоритмов, связанных с минимизацией маршрутных данных

Таким образом, в предыдущем подразделе предложено несколько операций по сжатию маршрутной информации, предоставляемой GPS-датчиком, на основе которых может быть сформирован алгоритм, состоящий из следующих шагов:

1) отбрасывание миллионных долей (шестая цифра после десятичной запятой) во всех имеющихся координатах;

2) выбор количества десятичных разрядов (вариант 2 или 4 из табл. 2.2), которые будут включаться в базу, и, соответственно, других, которые будут оставаться в каждой координате (в зависимости от характера маршрутной информации; для случая движения сельскохозяйственной техники по полю оптимальным является вариант 4 из таблицы 2.2);

3) формирование меток типа BASE, в которых задается базовая (неизменная в рамках большого блока данных) начальная часть каждой координаты и отбрасывание этой базовой части в пределах одного блока во всех имеющихся в блоке координатах;

4) выполнение кодирования каждого четырехзначного числа, соответствующего одной из координат, одной числовой переменной типа short int – при условии, что в шаге 2 был выбран вариант №2 из табл. 2.2;

5) выполнение кодирования каждого двухзначного числа, соответствующего одной из координат, одной числовой переменной типа char/byte – при условии, что в шаге 2 был выбран вариант №4 из табл. 2.2;

6) архивирование полученного файла с координатами с помощью архиватора.

Схема этого метода приведена на рис. 2.3.

Полученный файл следует разместить в централизованном хранилище (на одном выделенном сервере), а при возникновении необходимости специализированное программное обеспечение, подлежащее разработке в рамках данной работы, должно предоставлять возможность пользователю осуществить обратное преобразование и получить маршрутную информацию в понятной (текстовой) форме, что может распознаваться системами типа Google Maps.

Для тестирования этого алгоритма на вход должна быть представлена реальная или приближенная к ней по своим параметрам информация. В подразделе 1.3 уже выполнялось моделирование маршрутной информации (однако с совсем другой целью – оценить степень ее сжимаемости обычными архиваторами) и приведенный там элементарный способ генерации был вполне удовлетворительным. Сейчас же необходимо, в том числе, визуализировать маршрут наглядно, поэтому приведенная ранее процедура создания случайных координат, где каждая из следующих не связана с предыдущими, уже не является удовлетворительной. Впрочем, довольно легко можно разработать усложненную процедуру генерации случайных маршрутных данных, причем параметры и внешний вид которых очень близки к реальным движениям комбайна по полю. Реализация по-прежнему создана в системе Mathcad и представляет собой функцию пользователя, которая называется GenerateWay() и принимает 5 входных параметров:

- широта центральной точки поля;
- долгота точки поля, которая может считаться центральной;

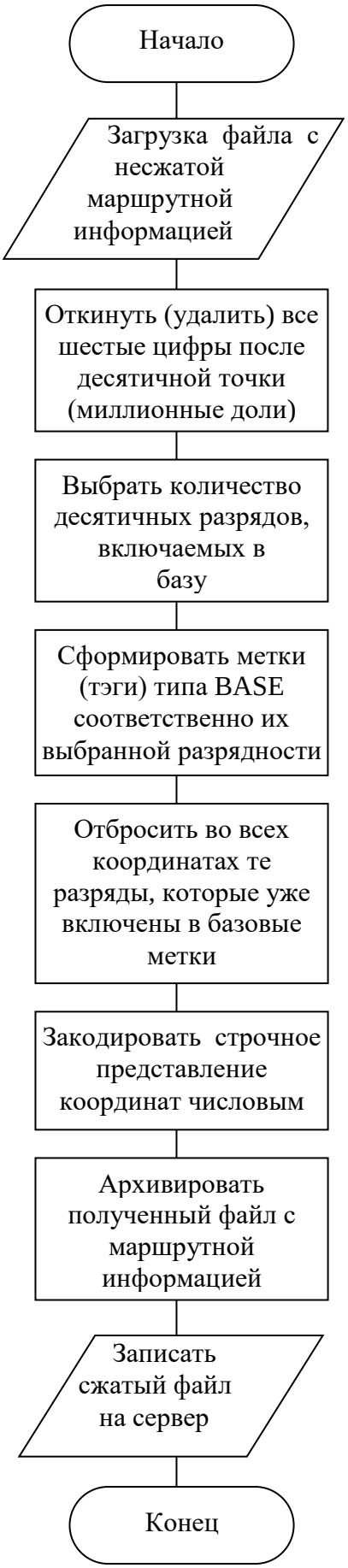


Рис. 2.3 Схема предлагаемого метода сжатия маршрутной информации

- допустимое отклонение движущегося объекта от центральной точки поля в градусах (когда оно достигается, комбайн резко меняет направление движения к центру);

- модуль скорости комбайна в градусах в секунду;

- направление скорости комбайна, задаваемого единым (полярным) углом α;

– количество точек для генерации.

Код разработанной функции можно видеть на рис. 2.4.

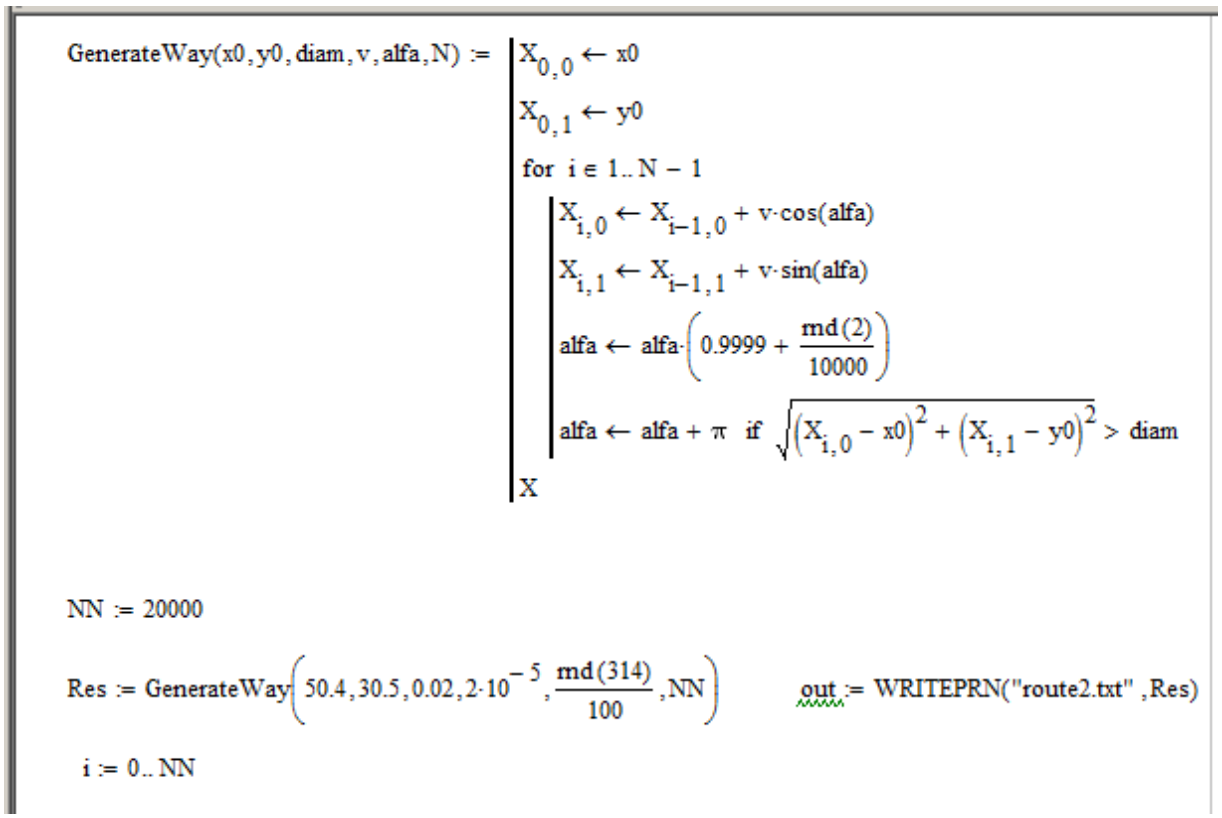


Рис. 2.4 Фрагмент окна среды Mathcad при моделировании маршрутной информации для сельхозтехники на поле

Результаты работы этой функции (при вызове GenerateWay(), показанном в нижней части рис. 2.4) приведены на рис. 2.5.

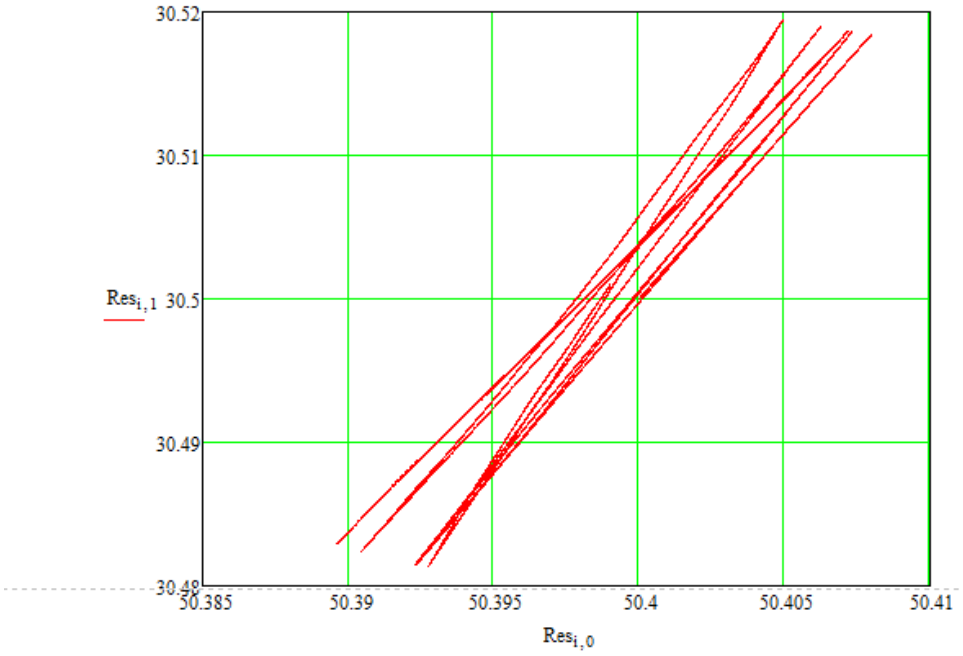


Рис. 2.5 Пример генерации модельных данных о маршруте сельскохозяйственной техники (в пределах одного поля)

Подходящий «большой» набор координат записывается в файл (route2.txt) и может быть использован для применения методов сжатия, в частности, в соответствии с алгоритмом рис. 2.3.

2.4 Учет зависимости параметров методов и моделей минимизации объема маршрутных данных от особенностей перемещения с/х техники

Как уже отмечалось в первой главе, параметры маршрутной информации в целом зависят от величины скорости движущегося объекта. В частности, наиболее ярким выражением этой зависимости является то, что при равенстве скорости нулю (т.е. когда происходит простой техники) координаты некоторое время просто повторяются. Конечно, нет никакого смысла сохранять данные, повторяющиеся много раз подряд, и самым очевидным подходом в этом случае является простой ввод в файл числа

повторов каждого набора координат с одновременным исключением повторяющихся координат. Это можно сделать с помощью специального тега или метки (который использовать в дополнение к BASE).

Следовательно, кроме ввода тегов типа BASE (B_X, B_Y) в файл следует ввести также тег типа REP (от REPetition – повторение), который означает повтор последних координат N раз, что является чрезвычайно эффективным и крайне необходимым способом сжатия маршрутных данных при банальном простое техники на обочине.

Отметим, что реализовывать разделение тега REP на две части в отдельности (типа R_X, R_Y) не имеет смысла, так как это не сделает систему более гибкой и эффективной: если объект не двигается, то неизменными являются одновременно обе его координаты, а если он перемещается, то меняться будут также обе координаты, хотя возможно в одной из них изменения будут только в самых младших разрядах координат (движение строго вдоль меридиана или параллели чрезвычайно маловероятно, поэтому этот случай в реальной практике не рассматриваем).

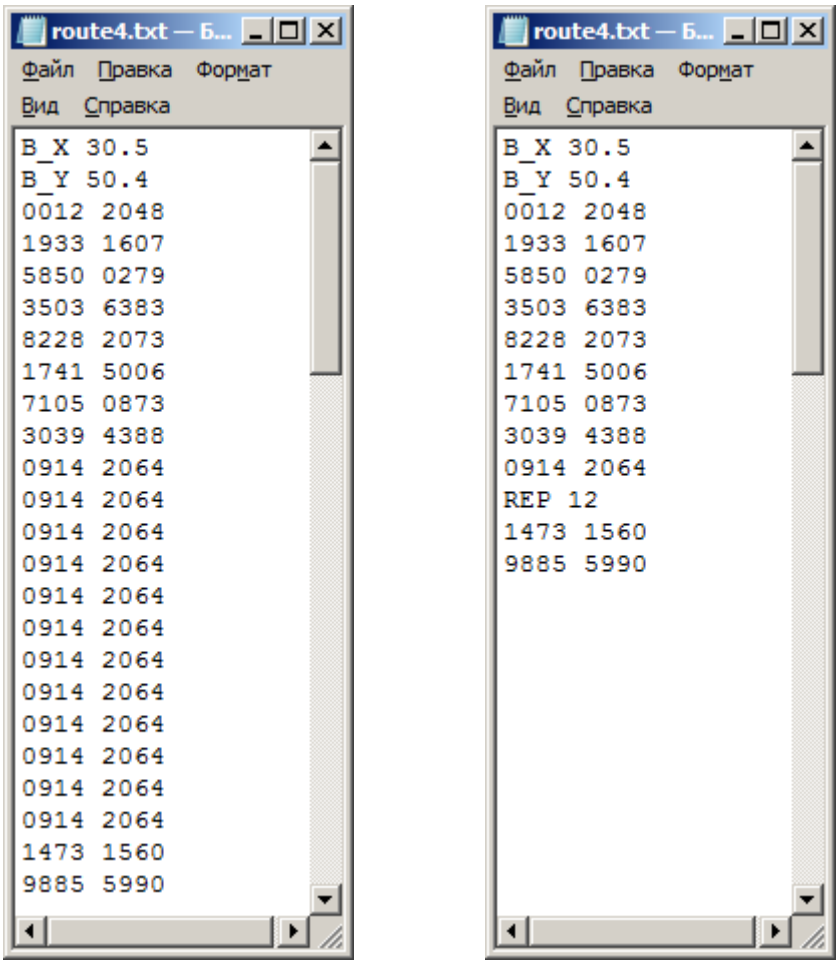
При таком подходе файл с координатами будет преобразован по примеру, приведенному на рис. 2.6. Очевидно, что при длительных простоях или многочисленных кратковременных остановках данный подход может предоставить значительную экономию объема места, необходимого для сохранения маршрутной информации.

При вводе описанного тэга REP его следует внедрить в алгоритм рис. 2.3 и последовательность соответствующих операций. Соответственно, получим новую схему алгоритма (рис. 2.7) и конечную последовательность действий при сжатии маршрутной информации:

1) отбрасывание миллионных долей (шестая цифра после десятичной запятой) во всех имеющихся координатах;

2) выбор количества десятичных разрядов (вариант 2 или 4 из табл. 2.2), которые будут включаться в базу, и, соответственно, других, которые будут оставаться в каждой координате (в зависимости от характера

маршрутной информации; для случая движения сельскохозяйственной техники по полю оптимальным является вариант 4 из таблицы 2.2);



а) *б*)

Рис. 2.6 Фрагмент файла с маршрутной информацией: *а* – до сжатия; *б* – после применения тега REP

3) формирование меток типа BASE, в которых задается базовая (неизменная в рамках большого блока данных) начальная часть каждой координаты и отбрасывание этой базовой части в пределах одного блока во всех имеющихся в блоке координатах;

4) формирование меток типа REP и отбрасывание повторяющихся координат, если хотя бы две, или более подряд расположенных точки имеют одинаковые координаты (при простое или остановке техники);

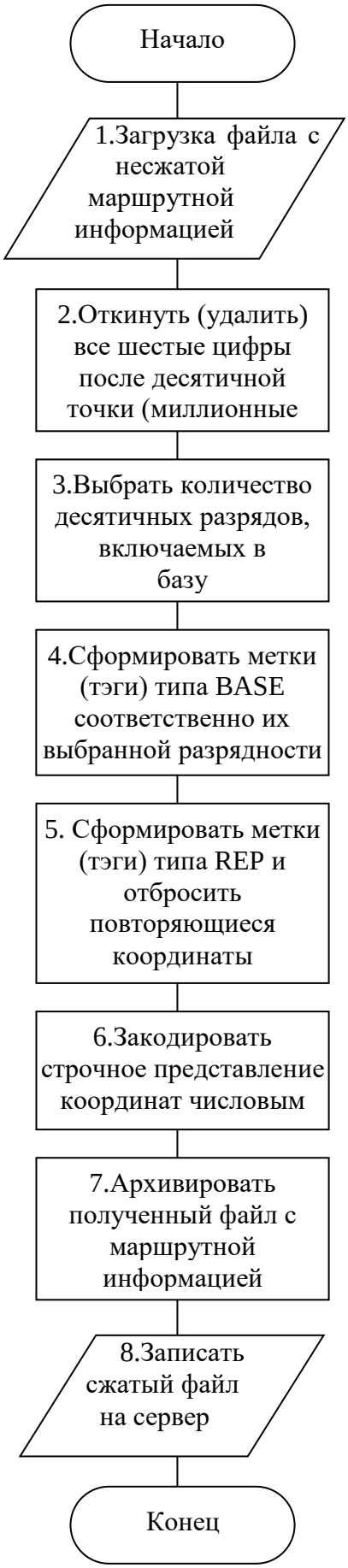


Рис. 2.7 Конечный вариант метода сжатия маршрутной информации (с учетом повторов отдельных координат)

5) выполнение кодирования каждого четырехзначного числа, соответствующего одной из координат, одной числовой переменной типа short int – при условии, что в шаге 2 был выбран вариант №2 из табл. 2.2;

6) выполнение кодирования каждого двухзначного числа, соответствующего одной из координат, одной числовой переменной типа char/byte – при условии, что в шаге 2 был выбран вариант №4 из табл. 2.2;

7) архивирование полученного файла с координатами с помощью архиватора.

Итак, в данном подразделе рассмотрен вариант внедрения в систему зависимости параметров сжатия маршрутной информации от скорости объекта на примере ситуации остановки или простоя, когда повторяющиеся координаты отвергаются, а остается только первое их вхождение с указанием количества повторений (предложен специальный тег REP, или кратко R).

Таким образом, в данном разделе предложены методы сжатия маршрутной информации по следующим принципам:

- отбрасывание наименьших (миллионных) долей градуса, соответствующих перемещениям на поверхности земли, близким к 0,1 м (согласно (1.7)) – из-за их ненужности в рассматриваемой системе;

- разбиение каждой координаты на базовую часть, которая прописывается один раз для сравнительно больших наборов подряд расположенных координат с помощью специального тега типа BASE (предложено отдельно рассматривать базу для долготы и широты: B_X и B_Y), и на вторую часть – значимые младшие разряды каждой координаты, не включенные в базу;

- исключение координат, повторяющихся несколько раз подряд (такая ситуация реализуется, если объект стоит на одном месте) путем внедрения специального тега REP (или коротко R) с указанием требуемого количества повторов;

- представление строковых выражений (которыми первоначально являются координаты точек и их части-составляющие) в числовой форме, то есть переход от символьного кодирования к двоичному;

- использование традиционных программ архиваторов перед окончательным помещением на файловый архивный сервер.

Все эти действия делают возможным эффективное сжатие маршрутной информации перед помещением ее в централизованный файловый архив. Следующим шагом в работе должна стать реализация окончательного решения посредством завершенного программного продукта, чему и посвящена следующая глава.

3 ОСОБЕННОСТИ РЕАЛИЗАЦИИ СИСТЕМЫ МИНИМИЗАЦИИ МАРШРУТНЫХ ДАННЫХ О ПЕРЕМЕЩЕНИИ СЕЛЬСКОХОЗЯЙСТВЕННОЙ ТЕХНИКИ

3.1 Обоснование выбора технологий и средств разработки

В предыдущем разделе выполнена разработка алгоритма эффективной минимизации маршрутных данных, подлежащего дальнейшему внедрению в программной реализации.

Перед выполнением любой программной реализации следует определиться с несколькими концептуальными вопросами, связанными с ней, а именно:

- какую технологию программирования лучше всего применить для реализации данного алгоритма при данном комплексе условий;

- какой язык программирования, поддерживающий выбранную технологию программирования, целесообразнее всего применить для программной реализации в существующих условиях;

– какую среду разработки (или комплекс простых отдельных средств разработки) лучше всего применить для данной программной реализации.

Только получив обоснованные ответы на эти вопросы можно переходить непосредственно к этапу программирования.

3.1.1 Выбор технологии программирования

Итак, первоочередным вопросом, стоящим перед разработчиками практически любого программного обеспечения, является выбор модели или технологии его разработки. Реально широко используемыми на сегодняшний день в производственной практике являются технологии структурного (процедурного) и объектно-ориентированного программирования. Каждая из них имеет свои особенности, преимущества и недостатки, которые мы рассмотрим подробнее.

Структурное, или как его еще называют, процедурное программирование основано на использовании отдельных структурных блоков – в первую очередь, подпрограмм (процедур и функций). Структурное программирование подразумевает построение программы в соответствии с тремя основными принципами: следование, ветвление, повторение.

Следование означает, что операторы и блоки программного кода следуют и выполняются один за другим. Ветвление реализуется разными условными операторами типа if, и позволяет выбирать один из нескольких последующих вариантов выполнения программы. Повторения обычно относят на счет циклов (многократных повторов одного и того же участка кода, идущих подряд), хотя этот же принцип можно отнести и к подпрограммам (функциям и процедурам).

Вообще же, структурное программирование иногда называют устаревшей методикой программирования, на смену которой вместе с оконным интерфейсом пришло объектно-ориентированное программирование (некоторые современные языки программирования общего назначения даже не позволяют создать структурную программу, только объектно-ориентированную – как, например, Visual C# или Java). Однако, при создании небольших программ (например, до 10000 строк кода и без предполагаемого расширения) применение этой методики программирования однозначно более оправдано ввиду ее простоты, а соответствующий код гораздо лучше воспринимается, чем его объектно-ориентированный вариант.

Суть же методологии объектно-ориентированного программирования заключается в том, что система рассматривается, как совокупность отдельных сущностей – объектов, имеющих набор каких-либо своих внутренних параметров – свойств, а также могущих взаимодействовать между собой с помощью некоторых действий – вызовов методов.

Если говорить о программном коде, то для того чтобы оперировать объектом, его сначала нужно создать. Объекты создаются как переменные, в которых типом выступает класс объекта. Класс – это просто описание, какие свойства могут иметь объекты данного типа (т.е. какую информацию они могут хранить), и какие у них есть методы (т.е. какие действия они могут выполнять). Объект – это набор значений, чему именно равны свойства данного экземпляра объекта (свои методы каждый объект получает от своего класса, то есть методы одинаковы у всех объектов, принадлежащих данному классу).

Кроме разбиения (декомпозиции) всей предметной области на объекты (классы) и соотношения между ними, также ОО-подход подразумевает соблюдение трех основных его принципов: инкапсуляция, наследование и полиморфизм.

Под инкапсуляцией подразумевается объединение данных (значения свойств класса у некоторого конкретного объекта) и средств их обработки (методы класса). Это опять же удобно психологически, так как позволяет реализовывать отдельные завершенные сущности – классы, которые сами обрабатывают свои данные. Обращение к объектам этих классов происходит посредством методов, образующих интерфейс класса.

Наследование весьма полезно, так как позволяет значительно сокращать объемы повторяющегося кода (к чему нужно всегда стремиться при разработке любого программного обеспечения). Согласно этому принципу выделяется класс, имеющий общий набор свойств и методов для нескольких более расширенных классов. Этот класс объявляется предком, базовым классом для нескольких производных от него (потомков, наследников). Все классы-потомки наследуют от базового все его свойства и методы, но кроме того еще имеют свои собственные оригинальные свойства и/или методы. При наследовании иногда методы родительского и производного класса имеют одинаковое предназначение, но реализуются по-

разному. Такие методы называются перегруженными. При этом еще раз подчеркнем, что назначение метода в обоих случаях одно и то же.

Полиморфизм является возможностью некоторой функции принимать объекты как родительского, так и производных классов, и уметь вызвать перегруженные методы именно того класса, объект которого был передан в функцию. Следует сказать, что это довольно специфическая возможность и многие программисты используют ОО-подход и без обращения к полиморфизму.

Важными понятиями в ООП также являются: статические члены класса, абстрактные методы и классы, дружба функций и классов и т.д. В целом можно утверждать, что ОО-подход значительно более сложен в применении, и должен использоваться только для крупных проектов, когда использование структурного подхода становится неэффективным.

Отметим, что часто помимо рассмотренных особенностей, также на выбор методики программирования влияют другие факторы, например, возможность будущего расширения функциональности, создания как можно более понятного кода (для работы над проектом целой команды, а не одного программиста), или просто пожелание заказчика применить наиболее современный подход к программированию (или наоборот, имеется разрешение на использование возрастных, проверенных временем технологий).

Основываясь на перечисленных особенностях двух существующих методов программирования, выбираем структурный подход как более простой, что соответствует небольшим (не промышленным) масштабам проектируемого ПО, а также требованиям к сложности (программа не может быть сложной из-за однотипности выполняемых ею операций: в основном это операции с строками).

3.1.2 Выбор языков программирования

После выбора технологии программирования следующим шагом является выбор языка программирования (или комплекса языков – при условии, например написания программного продукта в виде распределенного клиент-серверного приложения, а также при некоторых других условиях).

Наиболее укрупненно, по типу целевой платформы разрабатываемого программного продукта все языки программирования можно распределить на средства разработки настольных приложений, средства веб-разработки и мобильной разработки. Мобильные технологии в данной работе вообще не могут пригодиться, поскольку разрабатываемый программный продукт не нуждается в мобильности, а вычислительные ресурсы мобильных систем в разы меньше, чем у настольных, или тем более, клиент-серверных. Что касается альтернативы «веб-приложение» или «настольное программное обеспечение», первый вариант кажется более удобным по следующим причинам:

- полностью кроссплатформенный (качественные веб-приложения одинаково функционально работают в любых современных веб-браузерах, независимо от того, на какой операционной системе или даже на каком устройстве они запущены);

- позволяет использовать вычислительные ресурсы сервера, которые всегда значительно мощнее, чем у настольных ПК;

- могут иметь довольно быстрый и удобный интерфейс при условии использования в них современных веб-технологий (типа AJAX, кэширование, локальной доработки данных, предзагрузки страниц и т.д.).

Таким образом, выбираем веб-разработку и проанализируем средства, которые могут при таком случае использоваться.

Современные средства для разработки Интернет-ресурсов настолько обширны, что разобрать их все в рамках дипломной работы совершенно не представляется возможным. Поэтому в процессе выбора будем

ориентироваться как на объективные факторы (такие, как, например, поддержка современных технологий программирования), так и субъективные, но получившие широкое влияние (как-то склонность отдельных людей к определенным языкам, или даже стилей программирования, что в итоге влечет за собой популярность соответствующих средств разработки на массовом уровне).

Таким образом, рассмотрим наиболее распространенные средства разработки, которые популярны сейчас в соответствующей отрасли. В целом, весь процесс веб-разработки традиционно делят на две компоненты: front-end и back-end.

Упрощенно говоря, под «передней» частью – Front-end – в программировании имеется в виду то, что видит пользователь. Для настольных систем под этим подразумевают интерфейс, а под back-end'ом подразумевают логику работы программы, которая теоретически (в частности, так часто делают в Linux-системах) вообще может быть реализована отдельной программой с текстовым консольным интерфейсом. Задачей фронт-энда является удобный для широкого круга пользователей сбор всех входных данных, необходимых для выполнения содержательных операций консольной утилитой back-end’ом. Во многих случаях фронт и бэк соединены вместе в рамках одного программного обеспечения. Такова ситуация для настольных систем.

Если конкретнее говорить о веб-программировании, то здесь ситуация максимально поляризована (еще больше, чем в описанном выше случае написание фронт-энда одним программистом для бэк-энд-утилиты, созданной другим разработчиком). А именно, пользователь работает и «видит» систему через браузер на одной удаленной машине (на компьютере или другом устройстве, которое называется клиентом), а логика приложения и практически вся сопутствующая информация размещается на совсем другом компьютере-сервере.

Таким образом, все программные коды и исходные тексты, исполняемые в обозревателе, то есть на стороне клиента, относятся к front-end части. С другой стороны, все, что выполняется на сервере, относится к back-end составляющей.

Если глубоко не вдаваться в подробный анализ, то вкратце можно сказать, что набор инструментов для front-end разработки меньше, чем для бека. Среди них можно выделить:

- HTML – пятой версии (HTML5), самый современный стандарт языка гипертекстовой разметки (HyperText Markup Language), где традиционно под гипертекстом подразумевают текст, оснащенный гиперссылками на другие части документов и, возможно, сопровождаемый встроенными мультимедийными материалами: картинками, звуками, видео и т.д.;

- CSS – третья версия (CSS3) – правила записи стилей различных элементов веб-страницы, или, если говорить точно – каскадные таблицы стилей;

- JavaScript – самый мощный инструмент front-end разработки, позволяющий вывести html-страницы на новый уровень практически полноценных приложений, вроде настольных, активно реагирующих на действия пользователя, передающих и получающих информацию от него.

Эти технологии безальтернативны при разработке веб-приложений, поэтому применим их и в данной работе.

Как уже отмечалось выше, под Back-end'ом подразумевают программные компоненты, работающие на сервере, и выбор средств для реализации этих компонентов крайне широк.

Условно все средства для серверного веб-программирования можно разделить на два класса: отдельные приложения, работающие в соответствии с технологией CGI и языком, восходящие коды которых встраиваются прямо в веб-страницу и прогоняются препроцессором веб-сервера перед выдачей в браузер клиента. Второй подход более удобен, о чем свидетельствует все

большее упадок CGI-средств. В качестве альтернативы для них выступают такие серьезные продукты, как:

- серверные страницы Java – JSP, активно продвигаемые любителями этого открытого, современного языка программирования (общего назначения);

- активные серверные страницы ASP от корпорации Microsoft (соответственно степень поддержки этого решения очень высока);

- использование языка PHP, который и выбирает большинство разработчиков серверных приложений через целый комплекс положительных черт этого продукта, которые рассмотрим подробнее.

На сегодняшний день PHP является наиболее распространенным языком веб-программирования. Большинство сайтов и веб-сервисов в Интернете написано с помощью PHP. По некоторым оценкам PHP применяется более чем на 80% сайтов, среди которых такие сервисы как facebook.com, vk.com, baidu.com и другие. И такая популярность не кажется удивительной. Простота языка позволяет быстро и легко создавать сайты и порталы разной сложности.

Рассмотрим, какие же преимущества предоставляет PHP:

- для всех наиболее распространённых операционных систем (Windows, MacOS, Linux) есть свои версии пакетов разработки на PHP, а это значит, что можно создавать веб-сайты для любой из этих операционных систем;

- PHP может работать в связке с любыми веб-серверами: Apache, Nginx, IIS, и т.п.;

- простота и легкость освоения - особенность этого языка. Как правило, имея даже маленький опыт в программировании, на PHP можно создавать «средние» сайты;

- PHP похож на язык C, поэтому, зная C, или один из языков с C-подобным синтаксисом, будет проще овладевать PHP;

- PHP поддерживает работу с большим количеством систем баз данных (MySQL, MS SQL Server, Oracle, Postgres, MongoDB и другие);

– распространенность хостинговых услуг и их дешевизна. Как правило, хостинговые компании размещают веб-сайты, написанные на PHP, на веб-серверах Apache или Nginx, работающих на одной из операционных систем семейства Linux. И веб-серверы, и операционные системы на базе Linux бесплатны, что снижает общую стоимость использования хостинга;

- постоянное развитие, ведь PHP продолжает развиваться, выходят все более новые версии, которые несут новые функции, адаптируя язык программирования к новым вызовам. И, как правило, переход на новую версию не вызывает никакого труда;

- существует очень много качественных Интернет-ресурсов по поддержке процесса программирования на PHP, где можно задать собственный вопрос и получить квалифицированный ответ.

Учитывая все положительные качества, принимаем PHP в качестве основного средства Back-end разработки в данном проекте.

Таким образом, в качестве языков программирования для выбранного направления (веб-разработки) выбираем PHP и JavaScript.

Последним вопросом является выбор конкретной среды разработки, позволяющей писать код на выбранных языках. В качестве такой среды можно взять популярный на сегодняшний день среди студенческого сообщества Visual Studio Code или более профессиональный PHP Storm, или простой текстовый редактор типа Sublime Editor. Такая широта выбора средств обусловлена простотой задачи (в частности, отсутствием необходимости применения системы управления базами данных и т.п.) и стандартностью (распространенностью) выбранных языков программирования среди представителей современного программистского сообщества.

3.2 Особенности реализации элементов выбранных алгоритмов, связанных с минимизацией маршрутных данных

Рассмотрим особенности операций 1-8 из рис. 2.7.

Операция 1, заключающаяся в загрузке файла с несжатой маршрутной информацией, не имеет никаких особенностей, разве что файл следует открывать в текстовом режиме.

Операция 2 требует считывания всех чисел из файла и записи снова, но с меньшим количеством разрядов после десятичной запятой (5 вместо 6). Для форматированного вывода в файл можно использовать функцию типа fprintf.

Операцию 3 при работе с перемещениями сельскохозяйственной техники по полю можно выполнять, выбирая по умолчанию п. 4 из табл. 2.2.

Операция 4 будет выполняться следующим образом:

- считывается очередная координата X, и если ее базовые разряды не соответствуют текущей базе X, то в исходный файл записывается новый тег B_X;

– считывается очередная координата Y, и если ее базовые разряды не соответствуют текущей базе по Y, то в исходный файл записывается новый тег B_Y;

- разряды, относящиеся к базе, во всех координатах отбрасываются в любом случае.

Выполнение операции 5 следует вести по алгоритму, блок-схема которого приведена на рис. 3.1, текстовое описание этого алгоритма приводится ниже.

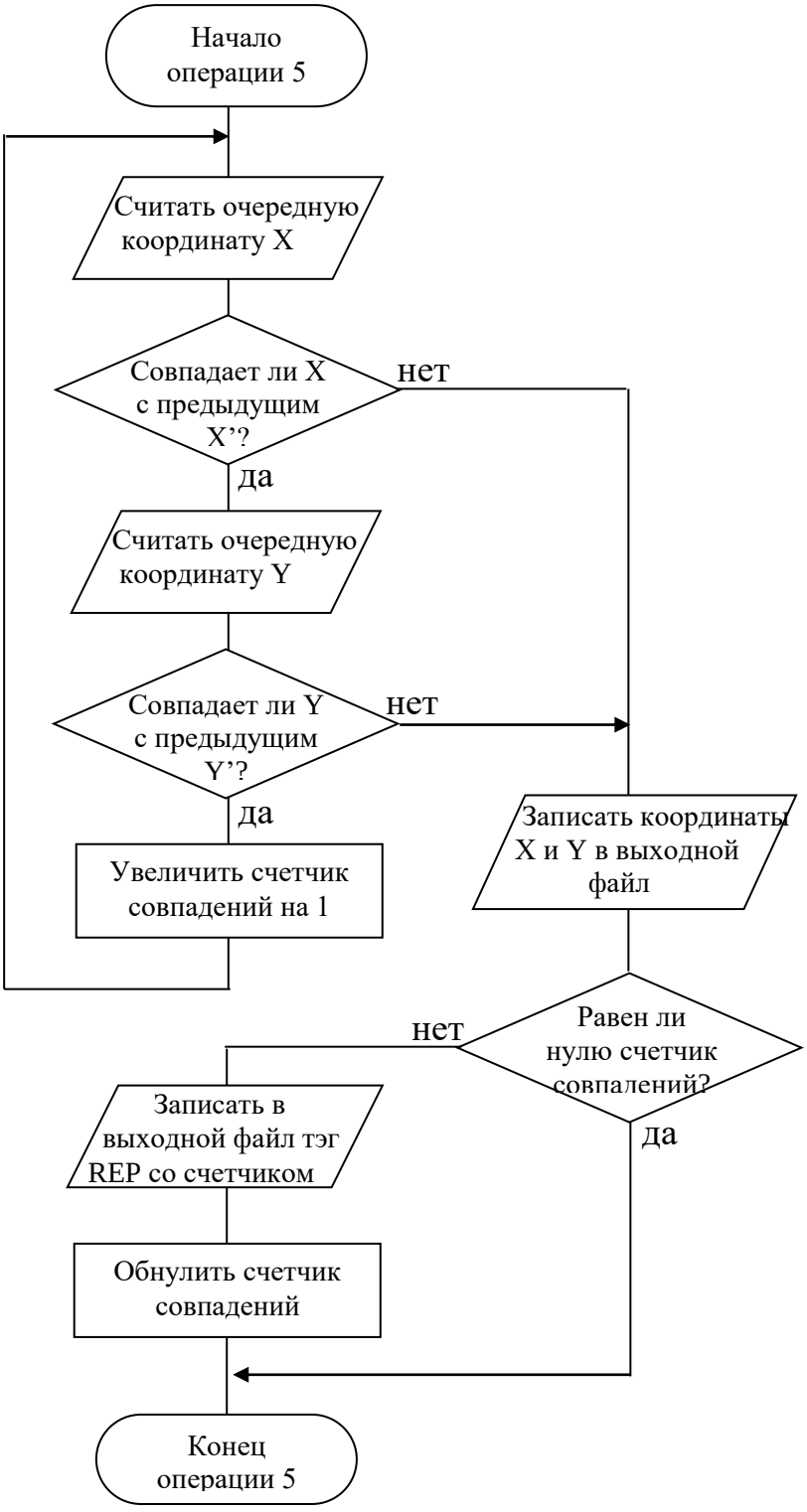


Рис. 3.1 Блок-схема выполнения операции 5 общего алгоритма сжатия маршрутных данных, заключающаяся в учете повторов (тег REP).

Так, во-первых, при считывании очередной координаты X следует сравнивать ее с предыдущим значением X', и, если есть совпадение, сравнивать еще и координату Y с предыдущей. Если совпали обе

координаты, то эти координаты в исходный файл не следует записывать, а нужно увеличить на единицу счетчик совпадений; если же совпадения нет, то этот счетчик следует обнулить. Если при обнулении оказалось, что значение счетчика от предыдущих сравнений не равно нулю, в исходный файл следует внести тег REP с соответствующим количеством повторов.

При выполнении операции 6 достаточно всего лишь считывать координаты из входящего файла как строки, а записывать в выходной файл как числа соответствующего формата (целые числа, длиной 1 байт). Эти действия, как и в параграфе 2, можно выполнять с помощью функций типа fprintf.

Операция 7 может быть выполнена любым стандартным способом архивирования, например алгоритмом ZIP.

Операция 8 представляет собой простое сохранение получаемого файла в нужном месте на сервере.

Таким образом, реальная алгоритмическая составляющая имеется только в шагах 4 и (в большей степени) в шаге 5. В целом, в подразделе рассмотрены особенности реализации всех 8 элементов алгоритма минимизации маршрутных данных, разработанного в подразделе 2.4.

3.3 Определение формата хранения маршрутов перемещения с/х техники

Выше по тексту работы уже немало внимания было уделено особенностям хранения маршрутной информации, поэтому можно заключить, что разработан собственный формат хранения соответствующей информации. Рассмотрим его подробнее.

Во-первых, в конечном файле сначала отдельной строкой должна быть записана дата и отдельной строкой время начала записи маршрутной информации. После этого отдельной строкой должен быть указан интервал,

через который снимаются данные о местоположении отслеживаемого движущегося объекта – стандартно 1 секунда.

Далее в отдельных строках должны размещаться теги типа BASE, которых существует две разновидности:

- B_X – указание базы первой координаты X;

- B_Y – указание базы для второй координаты Y.

Справа от этих тегов должны размещаться числовые значения разрядов, входящих в соответствующую базу, например:

B_X50.382

B_Y38.405

Далее в отдельных строках должны размещаться пары координат X и Y, причем только их младшие разряды (не входящие в базы), закодированные в числовой форме.

Если какие-то координаты повторяются, в файле будет введен тег REP (точнее его сокращенная форма R), после которого указывается количество повторов. Здесь N одинаковых строк соответствуют N–1 повторам, например, трем подряд одинаковым строкам координат соответствует тег вида:

R2

Таким образом, реализуя процесс минимизации маршрутной информации указанным путем и сохраняя данные в описанном формате, можно достичь высоких показателей эффективности подсистемы сжатия, что в идеале должно исследоваться экспериментальным путем, на основе работающего программного продукта, к описанию реализации которого следует перейти.

3.4 Особенности внедрения разработанной системы минимизации маршрутных данных в конкретном техническом решении

Для возможности постепенной разработки (включая отладку частей программы), весь код писался на базе программного продукта типа All-in-

One, называемого XAMPP и включающего веб-сервер (Apache), СУБД (MySQL или MareaDB), интерпретатор языка PHP и другие компоненты, мене значимые для данной работы.

Реализуем все разработанные выше алгоритмы на языке PHP с использованием JavaScript и получаем результат, показанный в окне рис. 3.2.

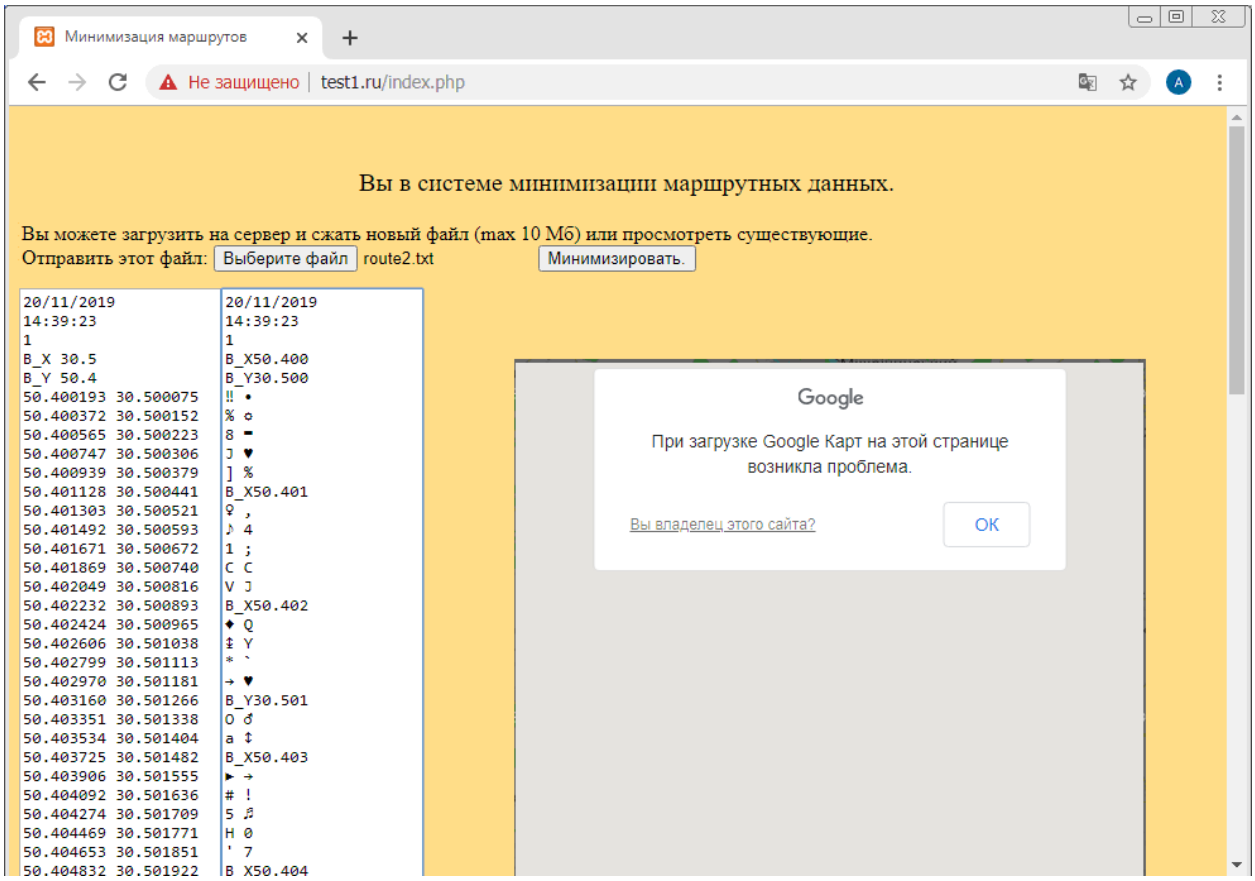


Рис. 3.2 Пример сжатия файла с маршрутной информацией с помощью разработанной системы

Видим, что в систему можно загружать файлы по стандартной кнопке «Выберите файл», после чего следует нажать кнопку «Минимизировать!». Выбранный файл с локального компьютера отправляется на сервер и там сжимается, результат чего отображается в правом вертикальном текстовом поле типа textarea, а главное передается в архивную папку для постоянного хранения.

В правой части окна программы мы видим попытку отображения маршрута на Гул картах, однако с середины 2018 года сервис предоставляется на платной основе. Хотя система и предоставляет 200 у.е. бесплатного доступа в месяц, однако все равно для регистрации требует ввода реальных данных кредитной карты и самое худшее то, что если каким-то образом по зарегистрированному акаунту будет осуществлено запросов больше, чем на 200 долларов в месяц (а это 20 тыс. запросов в месяц), то разница будет автоматически списана со счета подписчика. Таким образом, при создании атаки типа DDoS, сайт, использующий Google Maps API автоматически в конце месяца может стать должным компании Google тысячи долларов (и даже больше). Таким образом использовать эту систему для некоммерческих ресурсов на данном этапе ее развития опасно с финансовой точки зрения.

В целом же можно сказать, что система минимизации маршрутной информации реализована и готова к практическому использованию.

3.5 Анализ результатов работы системы

Очевиден факт сжатия маршрутной информации с помощью разработанной системы, однако эффективность этого процесса может быть разной, в первую очередь в зависимости от выбранного варианта представления базы: п.2 из табл. 2.2 или п.4, содержащийся в предыдущих разделах и принятый за основной вариант. На рис. 3.2 приведен пример использования базы с 3 цифрами после десятичной запятой и в результате этого файл с маршрутной информацией, имевший первоначальный размер 2149 Кб, был сжат до объема 901 Кб, что составляет около 42% от первоначального объема, а после применения архивации. объем уменьшился еще почти вдвое и составил 487 Кб. Общая степень сжатия составляет около 22% от первоначального объема, что примерно на 10% лучше применения простого архивирования.

Получается, что система, разработанная в рамках данной работы (т.е. без привлечения крупных финансовых инвестиций) показывает достаточный уровень экономической эффективности и позволяет экономить дисковое пространство не менее чем на 10% по сравнению с существующими решениями. При больших объемах хранящейся информации (как в централизованном файловом архиве, как и описано в данной работе) приводит к аналогичной экономии финансов на постоянные запоминающие устройства.

Таким образом, в разделе описано проектирование и реализацию системы минимизации маршрутной информации, выполненной в виде веб-приложения. Такой вариант удобен с организационной точки зрения, поскольку пользователь из любой точки планеты через Интернет может подключиться к разработанной системе и загрузить туда файл с маршрутной информацией, который автоматически будет сжат, результат показан пользователю, а сам сжатый файл помещается в папку с архивами. Созданная система дает экономию дискового пространства как минимум на 10% больше, чем применение простого архивирования, поэтому ее целесообразно применять в практической деятельности.

ЗАКЛЮЧЕНИЕ

В данной работе разработана система минимизации маршрутной информации на основе оригинального разработанного алгоритма. В работе проанализированы существующие способы снятия, преобразования и хранения маршрутной информации с учетом использования возможностей уменьшения ее объемов. После этого на основе известных подходов разработан комплексный оригинальный алгоритм минимизации маршрутных данных, а также проработаны сопутствующие научно-технические решения (как, например, моделирование маршрутной информации, создание собственного формата хранения данных и др.).

Также осуществлена программная реализация разработанного алгоритма и его исследование эффективности, что показало улучшение показателей сжимаемости минимум на 10% по сравнению с традиционными методами. В работе учтены разные зависимости (в частности, от скорости движения объекта), рассматриваются требования надежности хранения и защиты соответствующей маршрутной информации.

Разработанный программный продукт является полностью работоспособным и может быть использован в реальных практических задачах.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. JavaScript. Шаблоны / Под ред. Стоян Стефанов - СПб, Символ-Плюс, 2011. С. 120-123.

2. Выразительный Javascript / Под ред. Marijn Haverbeke - No Starch Press, 2015. С. 37-41.

3. Основное преимущество PHP, определяющее его популярность [Электронный ресурс]. Электрон. дан. www.dlya-mastera.ru, cop. 2007–2010. URL: http://dlya-mastera.ru/132-osnovnoe-preimushhestva-phpopredelyayushhie-ego-populyarnost.html/.

4. Преимущества PHP [Электронный ресурс]. Электрон. дан. Sitesprofi.ru, cop. 2008–2010. URL: http://sitesprofi.ru/osnovy_php/preimuwestva_php.

5. Преимущества PHP [Электронный ресурс]. Электрон. дан. PHP.SU, cop. 2006–2010. URL: http://php.su/php/?opport.

6. Воронцов Ю.А., Козинец А.В. Стандарты веб-сервисов для создания распределенных информационных систем / Ю.А.Воронцов, А.В.Козинец, // Век качества. 2015. №3.

7. Воронцов Ю.А., Козинец А.В. Пример построения распределенной информационной системы на AJAX с использованием PHP и IIS (Internet Information Services) / Ю.А.Воронцов, А.В.Козинец, // Век качества. 2016. №2.

8. Дюбуа П. MySQL.: Пер. с англ. М.: Издательский дом «Вильямс», 2004. 1056 с.: ил.

9. Хокинс С. Администрирование Web-сервера Apache и руководство по электронной коммерции.: Пер. с англ. М.: Издательский дом «Вильямс», 2001. 336 с.: ил.

10. Аргерих Л. и др. Профессиональное PHP программирование, 2-е издание. Пер. с англ. СПб: Символ-Плюс, 2005. 1048 с.: ил.

11. Росс В. С. Создание сайтов: HTML, CSS, PHP, MySQL. Учебное пособие, М.:МГДД(Ю)Т, 2010. Ч.1. 107 с.

12. Котеров Д.В. PHP 7 / Д.В. Котеров, И.В. Смидянов. – Санкт Петербург: БХВ-Петербург 2017. – 265 с.

13. Кузнецов М. В. Объектно-ориентированное программирование на PHP. – БХВ-Петербург, 2012.

14. Маклафлин Б. PHP и MySQL. Исчерпывающее руководство первое издание / Маклафлин Б. – Санкт Петербург 2004.

15. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript и CSS. 2-е изд //СПб.: Питер. – 2013. – 201 с.

16. JavaSpript. Подробное руководство / Под ред. Дэвид Флэнаган. СПб, СимволПлюс, 2008. – 923 с.