РЕФЕРАТ

Михайлов А. В. РАЗРАБОТКА ПРОЦЕДУРЫ ПРОГНОЗИРОВАНИЯ ПРОМЕЖУТКА ВАРИАЦИИ МОМЕНТА ПРИБЫТИЯ ВОЗДУШНОГО СУДНА, выпускная квалификационная работа: стр. 31, рис. 8, библ. 9 назв.

Ключевые слова: УПРАВЛЕНИЕ ВОЗДУШНЫМ ДВИЖЕНИЕМ, ВОЗДУШНЫЕ ТРАССЫ, СХЕМЫ ЗАДЕРЖКИ, ПОТОКИ ВОЗДУШНЫХ СУДОВ, АЛГОРИТМЫ БЕСКОНФЛИКТНОГО СЛИЯНИЯ, ПРОГНОЗИРОВАНИЕ МОМЕНТА ПРИБЫТИЯ.

Целью работы является разработка и реализация алгоритма прогнозирования прибытия воздушного судна (ВС) в контрольные точки его движения по трассе. Входной информацией для алгоритма служит топографическая конфигурация системы воздушных трасс в том или ином районе управления воздушным движением (УВД). Такая конфигурация включает в себя описание контрольных точек трасс (координаты и допустимый диапазон скоростей ВС в точке), схемы их соединения (порядок прохода ВС по этим точкам), наличие вариантов трасс и схем задержек (которые подразумевают для некоторой контрольной точки несколько возможных следующих точек), участков возможного спрямления движения. В результате работы алгоритма для каждой точки каждой трассы вычисляется множество моментов возможного прибытия ВС в эту точку (в предположении что в начальный момент времени ВС вошло на контроль на первую точку своей трассы). Такая информация необходима для работы алгоритмов безопасного слияния потоков ВС в точках соединения их трасс, например, в аэропортовой зоне, куда приходят несколько потоков ВС с различных направлений. Создан формат представления воздушных трасс. Разработанный алгоритм реализован в виде программы на языке C++.

Содержание

Введение

В настоящее время движение воздушных судов (ВС) происходит по воздушным трассам, состоящим из коридоров в горизонтальной плоскости и эшелонов в вертикальной. При этом трассы могут разветвляться или соединяться (рис. 1). В точке соединения трасс возникает задача слияния потоков самолётов в единую посадочную очередь. Такая задача особенно актуальна в зонах подхода и зонах аэродромов, где плотность воздушного движения высока. Основным требованием при слиянии потоков ВС является наличие минимального безопасного временного интервала между моментами прибытия судов в точку слияния.

Поэтому имеется ряд задач:

1. на основе топографического описания воздушных трасс спрогнозировать возможные моменты прибытия ВС в контрольные точки его трассы;
2. на основе этой информации, а также с использованием номинального расписания прибытия ВС, разрабатывается фактическое расписание прибытия ВС, обеспечивающее безопасность в точках слияния потоков ВС;
3. на основе построенного расписания и топографического описания воздушных трасс вырабатываются рекомендации диспетчерам управления воздушным движением (УВД) по регулированию движения каждого планируемого ВС для выдерживания им режима движения, требуемого для прибытия в контрольные точки в требуемые моменты времени.

В рамках данной работы предлагается алгоритм решения первой из этих задач, то есть построение множеств возможных моментов прибытия ВС в те или иные контрольные точки его воздушной трассы, которые могут быть реализованы с использованием задержек и ускорений, связанных с вариацией скорости движения ВС, задержек, обеспечиваемых схемами задержки, ускорениями, обеспечиваемыми участками спрямления движения.

1 Постановка задачи

В рамках работы были поставлены следующие задачи:

1. изучить типовые элементы воздушных трасс и схем задержек;
2. придумать формальное описание этих элементов;
3. предложить формат представления данных о воздушной трассе в файле;
4. разработать структуры для хранения этих данных в памяти компьютера;
5. создать процедуры чтения этих данных из файла;
6. разработать математическую модель движения ВС по тем или иным элементам воздушной трассы;

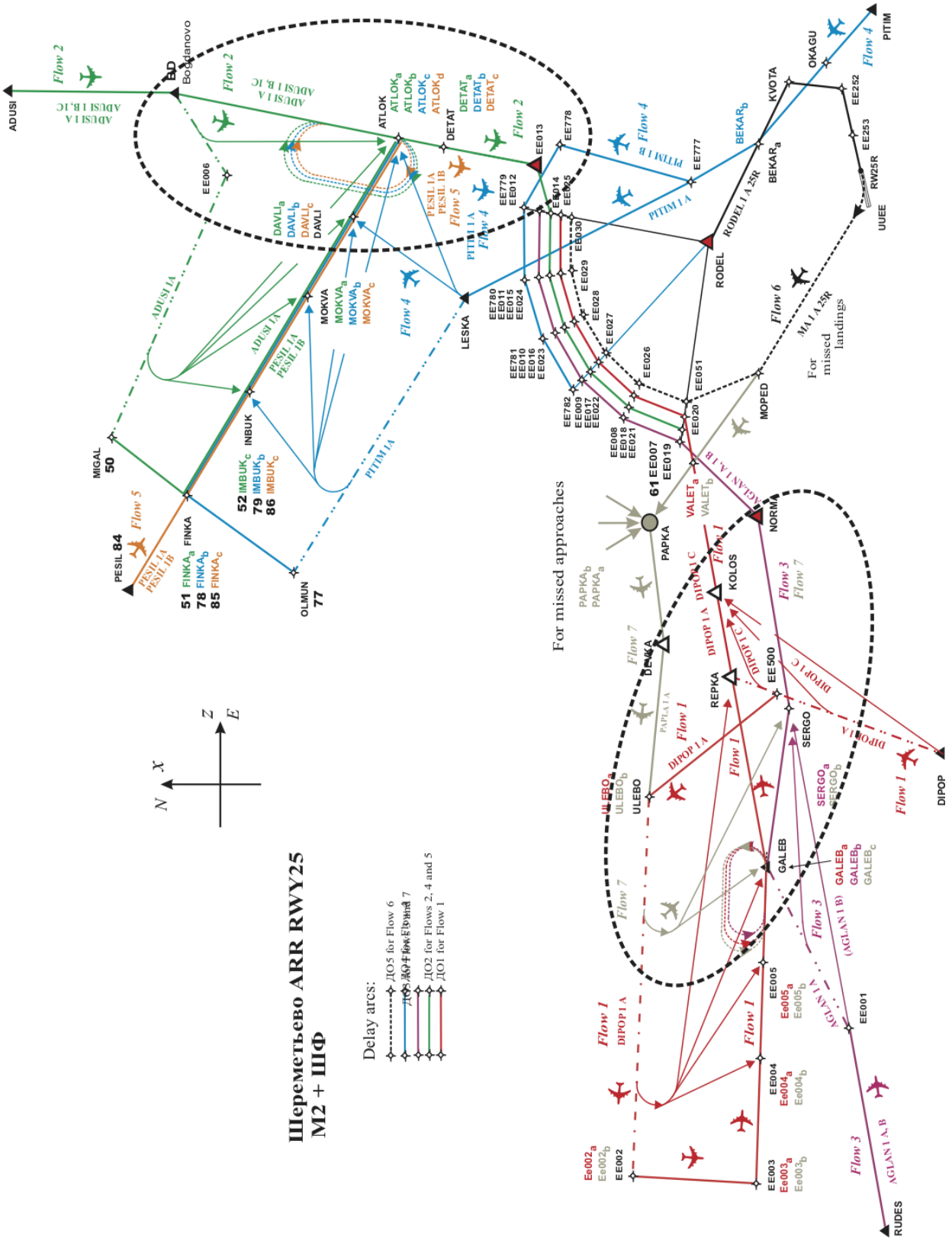

Рис. 1. Схема части воздушных трасс в аэропортовой зоне а/п Шереметьево

7. разработать процедуру, использующую модель движения ВС и структуру воздушной трассы, для формирования множества возможных моментов прибытия ВС в контрольные точки трассы.

2 Типовые элементы воздушных трасс и схем задержек

При рассмотрении схемы трасс, приведенной на рис. 1, можно выделить следующие типовые элементы трассы.

1. Линейный участок трассы, включающий две или более контрольных точек, по которым ВС движется последовательно.

2. Стандартная схема задержки (рис. 2). На стандартной схеме задержки ВС сходит с трассы, разворачиваясь на 180°, некоторое время движется в обратном направлении, после чего снова разворачивается и идёт по той же трассе или параллельно ей на более низком эшелоне.

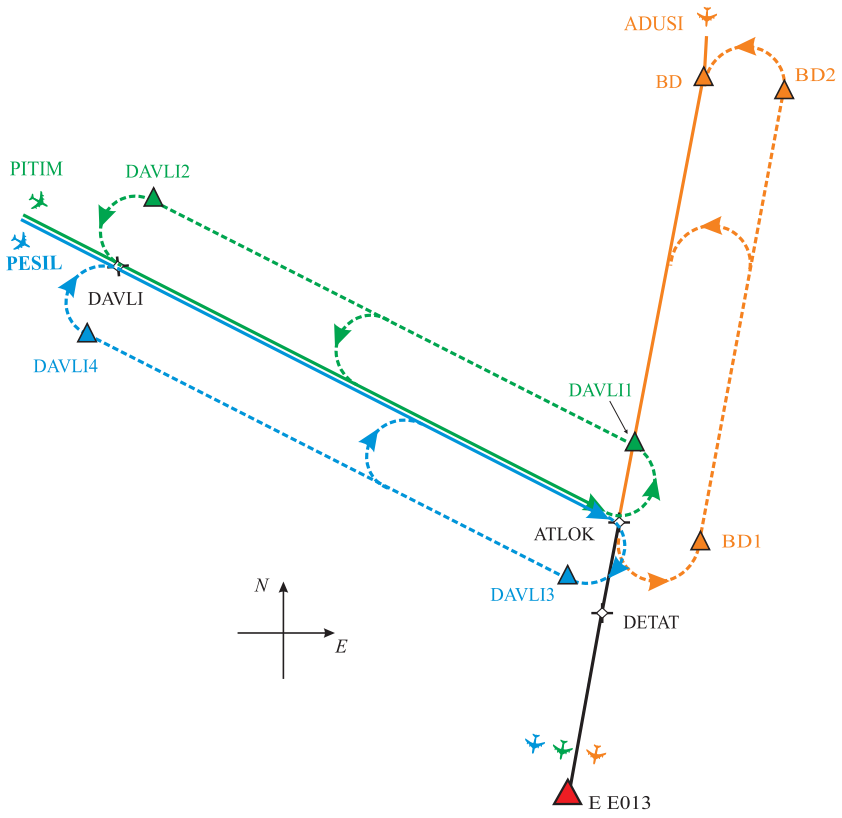

Рис. 2. Стандартная схема задержки

Стандартная схема характеризуется тремя точками:

1. точкой начала обратного разворота;
2. точкой конца дуги разворота;
3. предельной точкой возможного обратного движения;

Четвертая точки, точка конца дуги разворота, возвращающего на трассу, подразумевается. Важно отметить, что ВС может сделать разворот, возвращающий на трассу в любой точке прямолинейного отрезка 2–3 обратного движения и выйти на соответствующую точку отрезка 4–1. Четвертая точка не включается в о писание схемы, хотя и отмечена в схемах на рис 2, поскольку в вычислениях неважно, на какую именно точку вернулось ВС. Время повторного движения в прямом направлении считается равным времени обратного движения.

3. Схема задержки веерного типа (рис. 3. На схеме задержки веерного типа ВС выполняет поворот, уходя с маршрута движения, и двигается по дуге окружности с центром в следующем контрольном пункте, сходя с этой дуги на точку слияния при достижении нужной задержки.

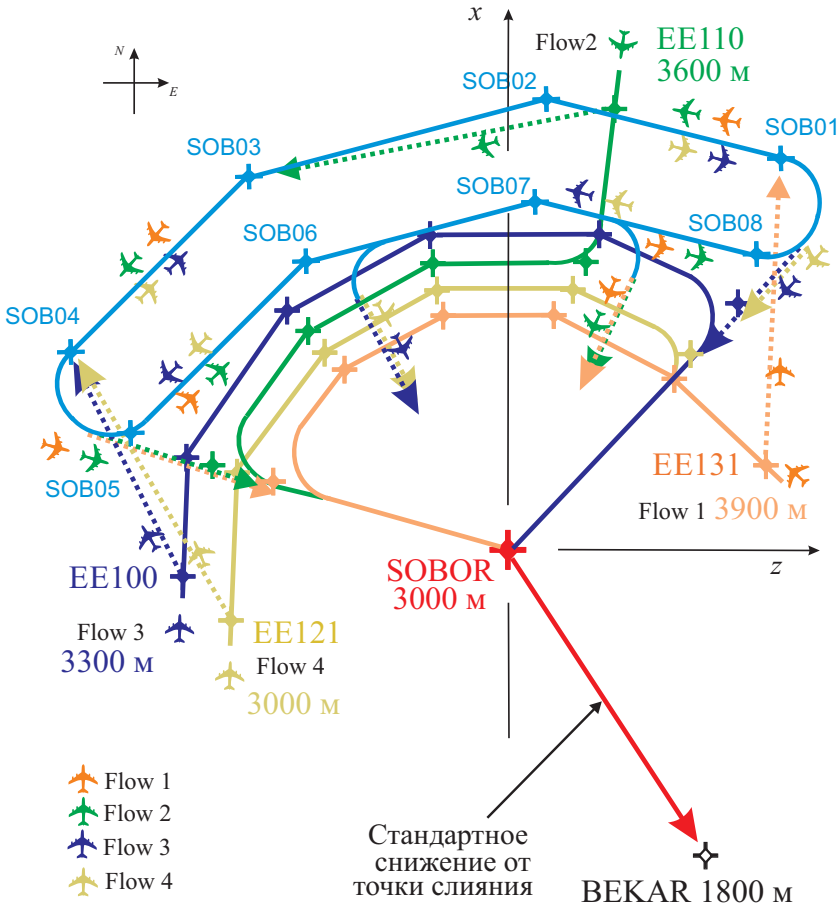

Рис. 3. Схема задержки веерного типа

Веерная схема характеризуется точкой слияния, а также точками, расположенными на дуге ожидания. Сход на точку слияния возможен с любой точки отрезков, соединяющих точки, расположенные на дуге ожидания.

4. «Тромбон» (рис. 4). Эта схема является объединением веерной схемы и стандартной схемы задержки. Данная схема состоит из стандартной схемы задержки, развёрнутой под углом к основному маршруту движения. При необходимости выполнения задержки ВС сходит с основного маршрута движения на данную схему, продолжа-

ет движение на том же эшелоне, удаляясь от основного маршрута движения. Затем, при необходимости выполнения длительной задержки, выполняет разворот на 180° со снижением и продолжает движение к основному маршруту движения. Выработав необходимую задержку, ВС выполняет сход (разворот) со снижением с плеча тромбона в точку слияния.

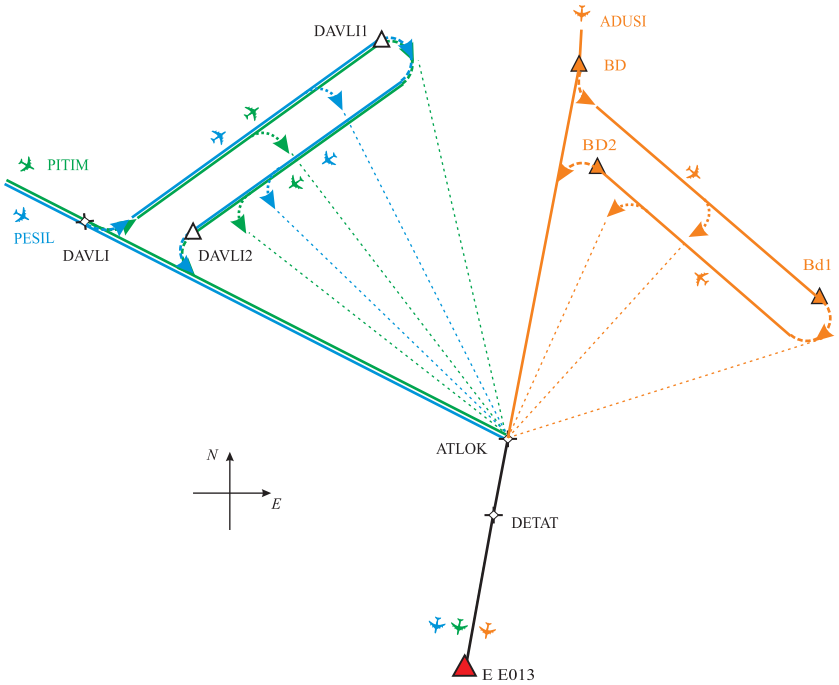

Рис. 4. Схема задержки типа «тромбон»

Соответственно, «тромбон» задаётся

1. точкой схода на боковое плечо;
2. точкой конца бокового плеча и начала обратного разворота;
3. точкой окончания обратного движения к трассе;
4. точкой нацеливания при сходе с «тромбона», которая также является и точкой трассы, по которой ВС идет в случае отсутствия задержки.

5. Участок спрямления (рис. 1, красная трасса на точках DIPOP–EE500–REPAK–KOLOS). Участок спрямления в каком-то смысле является схемой ускорения, обратной веерной схеме задержки. Если в веерной схеме нормальным является прохождение мимо схемы без схода на дугу ожидания, то на участке спрямления нормальным является движение по всем точкам трассы с возможным сходом на конечную точку при необходимости. Соответственно, с точки зрения своего описания участок спрямления полностью эквивалентен веерной схемы задержки.

Таким образом, можно видеть, что имеется три основных элемента, из которых формируются воздушные трассы: линейный участок движения, участок спрямления и стандартная схема задержки. Особенностью третьей схемы является то, что при возврате

на трассу ВС выходит не на точку нацеливания, а на отрезок между двумя контрольными точками. Кроме того, в отличие от других схем стандартная схема допускает свое неоднократное использование.

В итоге имеем иерархическую систему описания трасс. В основе описания лежит описание контрольных точек с информацией об их координатах и диапазоне допустимых скоростей их прохождения.

Далее из точек формируются элементы трасс. При этом линейный участок — это просто список точек, которые ВС проходит последовательно, в участок спрямления дополнительно включает точку или точки, на которые можно уходит из промежуточных положений. Описание стандартной схемы включает в себя четыре точки и информацию о возможном количестве кругов, которые может по ней пройти ВС.

Наконец, трасса — это последовательность отдельных элементов. Нужно отметить, что трасса, в целом, может иметь нелинейный характер, разветвляться на две или более ветвей, имеющих разные длительности движения ВС по ним. Например, на рис. 1 красная трасса имеет прямой путь EE500–REPKA–KOLOS, а кроме того длинный маршрут EE500–ULEBO–EE002–EE003–EE004–EE005–GALEB–REPKA–KOLOS. Впрочем, стыковка отдельных элементов в единую трассу производится по понятному признаку: конечная точка предыдущего элемента совпадает с начальной точкой следующего.

3 Работа с данными о трассах

Приняв описание отдельных элементов трассы, предложенное в предыдущем разделе, мы приходим к заключению о необходимости задания, чтения и хранения следующих описаний:

1. данные контрольных точек;
2. данные о схемах (об элементах трассы);
3. данные о потоках (о трассах).

Предполагается, что для удобства подготовки и просмотра все входные файлы являются текстовыми.

3.1 Данные о контрольных точках

3.1.1 Формат файла

Данные по контрольным точкам всех трасс хранятся в файле следующего формата. В первой строке указывается целое число — количество контрольных точек в описываемом районе.

Далее перечисляются контрольные точки по одной на строке в следующем формате:

ИМЯ_ТОЧКИ x y z V_{min} V_{max} флаг_посадочной_полосы

Отдельные элементы строки разделяются пробелами.

ИМЯ_ТОЧКИ — последовательность символов, начинается с буквы.

Величины x, y, z являются вещественными числами и описывают координаты точки в неподвижной системе координат, связанной с поверхностью Земли. Поскольку рассматривается относительно небольшой участок поверхности Земли, то ее кривизной пренебрегаем. Измеряются в метрах.

Величины V_{min}, V_{max} определяют допустимый диапазон скорости ВС при прохождении данной точки: V_{min} — минимально допустимая скорость, а V_{max} — максимально допустимая скорость. Измеряются в метрах в секунду.

Наличие величины флаг_посадочной_полосы означает, что эта точка связана с торцом взлетно-посадочной полосы (ВВП), где нужно завершить построение трассы. Точка считается финальной, если в конце строки имеется слово `LAND`. Если в конце строки этого элемента нет или стоит какой-либо другой токен, точка финальной не считается.

Следует отметить, что внутреннее хранение величин длины, скорости и времени организовано не в вещественных переменных, а в экземплярах классов `Coordinate`, `Velocity` и `Time`. При этом инициализировать и получать значения длин можно в метрах, километрах и морских милях, времен — в секундах, минутах и часах, а скоростей — в произвольных комбинациях единиц длин и времен. Но в текущей версии в формате и, соответственно, в процедуре чтения жестко прописаны метры и метры в секунду.

Примеры:

- `SS025 0 0 900 70 90`
- `RW08L -21.5 18431.5 10 65 85 LAND`

3.1.2 Структуры хранения

Для хранения информации об одной контрольной точке предусмотрен следующий тип записи:

```
struct CheckPoint {
  string name;
  Coordinate x = Coordinate::createMs(0);
  Coordinate y = Coordinate::createMs(0);
  Coordinate z = Coordinate::createMs(0);
  Velocity Vmin = Velocity::createVkm_h(0);
  Velocity Vmax = Velocity::createVkm_h(0);
  bool landing_flag = false;
};
```

При чтении точки укладываются в массив структур указанного типа. Однако такой подход затрудняет доступ к точкам, которые во входных файлах упоминаются по имени. Поэтому при чтении также заполняется словарь

```
map<string, int> pointNameToID;,
```

в котором ключом является имя точки, а значением — ее индекс в массиве, хранящим записи точек. В дальнейшем при чтении данных о схемах имена контрольных точек заменяются на индексы с использованием этого словаря.

3.1.3 Процедура чтения

Чтение файла с данными по контрольным точкам происходит следующим образом. Вначале читается количество точек.

Затем на каждой итерации цикла `for` читается по одной строке, которая разбирается в соответствующую структуру. В принципе, разбор строки можно было бы осуществить прямолинейно, по токенам. Однако для чтения схем (см. раздел 3.2.3) применяется механизм регулярных выражений. Поэтому было принято решение и чтение точек организовать с использованием этого инструмента.

Регулярное выражение для разбора строки с информацией о контрольной точки:

```
([a-z,A-Z]\w*)\s+
([-+]?[0-9]*\.?[0-9]+)\s+
([-+]?[0-9]*\.?[0-9]+)\s+
([-+]?[0-9]*\.?[0-9]+)\s+
([0-9]*\.?[0-9]+)\s+
([0-9]*\.?[0-9]+)
\s*(LAND|0)?\s*
```

Первая группа захвата отвечает за имя точки. Согласно формату, имя начинается с буквы. Далее может быть любые комбинации букв и цифр.

Вторая, третья и четвёртая группы захвата одинаковые и предназначены для распознания вещественных чисел, представляющих собой координаты точки.

Пятая и шестая почти аналогичны предыдущим трём, но они захватывают не все действительные числа, а только неотрицательные, представляющие собой скорости.

И, наконец, седьмая проверяет наличие флага посадки `LAND`, нуля, означающего непосадочную точку, либо же проверяет отсутствие токена.

Выигрыш от использования регулярных выражений в сравнении с ручным разбором строки заключается в автоматической проверке корректности формата строки.

3.2 Данные о схемах

3.2.1 Формат файла

Как говорилось ранее, фактически имеется три типа схем (элементов трассы): линейный участок, участок со спрямлением и стандартная схема задержки. При этом различие линейного участка и участка со спрямлением заключается именно в наличии или отсутствии этого спрямления.

Поэтому файл с данными о схемах разбит на две части: первая часть хранит данные по линейным участкам и участкам со спрямлением, которые задаются унифицировано; вторая часть хранит данные по стандартным схемам задержки. Формат файла следующий:

- первая строка хранит количество схем, соответствующих линейным участкам и участкам со спрямлением;
- вторая строка хранит количество стандартных схем;
- следующий блок хранит данные по схемам, соответствующих линейным участкам и участкам со спрямлением, по одной на строке;
- заключительный блок хранит данные по стандартным схемам по одной на строке.

Описание линейной схемы и схемы со спрямлением:

```
NAME (START)(FINAL LIST) : POINTS [STRAIGHTENING] POINTS
```

Здесь:

`NAME` — название схемы, последовательность букв и цифр;

`START` — имя начальной точки схемы;

`FINAL LIST` — список точек, которые считаются конечными точками схемы; используется для соединения схем между собой;

`POINTS` — список имен точек, составляющих линейный участок; может быть пустым;

`STRAIGHTENING` — описание участка, с которого возможно спрямление; если данный элемент отсутствует, то схема является линейной, если присутствует, то схема соответствует участку со спрямлением.

Описание участка спрямления имеет следующий вид:

```
Str(POINT( POINT)*) POINT POINT( POINT)*/Str
```

Токен `Str` является сигналом начала описания участка со спрямлением, токен `/Str` — сигналом окончания описания. Список имен точек в круглых скобках после токена `Str` — это список тех точек, на которые можно спрямляться. Этот список должен содержать хотя бы одну точку. Дальнейший список имен точек до токена `/Str` — это концы тех отрезков, с которых возможно спрямление на указанные точки. Этот список содержит хотя бы две точки, то есть хотя бы один отрезок. Подразумевается, что

последняя точка из списка до описания участка спрямления, если этот список непуст, связана с первой точкой участка спрямления. Также считается, что последняя точка участка спрямления связана с первой точкой из списка после описания спрямления, если этот список непуст.

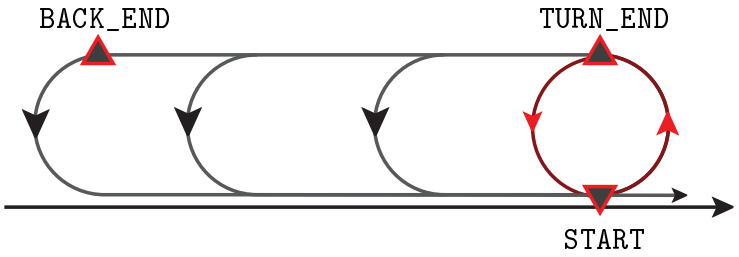

Рис. 5. Стандартная схема задержки

Формат описания стандартных схем:

```
NAME (START)(REPEAT) : START TURN_END BACK_END
```

Здесь:

`NAME` — название схемы, последовательность букв и цифр;

`START` — имя начальной точки схемы, точки начала дуги обратного разворота (см. рис. 5); эта же точка является первой из трех в списке точек, определяющих схему;

`TURN_END` — точка конца дуги обратного разворота (см. рис. 5);

`BACK_END` — точка, ограничивающая отрезок обратного движения ВС (см. рис. 5);

`REPEAT` — натуральное число, означающее максимальное количество проходов одного ВС по схеме.

По итогам тестирования программы на реальных трассах, выяснились некоторые типовые ситуации. После чего в процедуру чтения был встроен некоторый «синтаксический сахар»:

1. в описании стандартной схемы можно опустить часть «`(START)`»; начальная точка будет взята из списка;
2. если линейная схема состоит только из двух точек

   «`NAME (START)(FINISH) : START FINISH`»,

   то две точки после двоеточия можно опустить: «`NAME (START)(FINISH) :` ».

Примеры:

- `NameA (DIPOP)(EE500 KOLOS): Str(KOLOS) DIPOP EE500 /Str`
- `NameB (KOLOS)(BEKAR) : KOLOS VALET Str(RODEL) EE020 EE021 EE022 EE023 EE024 EE025 /Str RODEL BEKAR`

- `NameC (RODEL)(RW25R): RODEL BEKAR KVOTA EE252 EE253 RW25R`
- `NameStScheme (GALEB)(5): GALEB N1 N2`

3.2.2 Структуры хранения

Для хранения линейных схем и схем со спрямлением используется следующая структура

```
struct Scheme {
  string name;
  int start;
  vector<int> path;
  vector<int> end;
  vector<int> straighteningFrom;
  vector<int> straighteningWhere;
};
```

Поле `name` хранит имя схемы, по которому она будет упоминаться при построении потоков, поле `start` — индекс начальной точки.

Массив `path` хранит индексы всех точек линейной части схемы. В случае схемы со спрямлением это точки идущие до токена `Str`, между токенами `Str` и `/Str`, и после токена `/Str`.

Массив `end` хранит индексы всех точек, считающимися концевым для данной схемы.

Массив `straighteningFrom` хранит индексы точек, описывающих участок с которого можно спрямляться. Соответственно, массив `straighteningFrom` хранит индексы точек, на которые можно спрямляться.

Для хранения стандартных схем предназначена структура

```
struct StandardScheme {
  string name;
  int start;
  int second;
  int third;
  int repeat;
  Time Tmin = Time::createTsec(0);
  Time Tmax = Time::createTsec(0);
};
```

Поле `name` хранит имя схемы. Поля `start`, `second` и `third` хранят индексы точек `START`, `TURN_END` и `BACK_END`, соответственно (см. рис. 5). Поле `repeat` хранит максимальное число проходов по схеме.

Как будет сказано ниже, обработка стандартных схем происходит специальным образом, поэтому поля `Tmin` и `Tmax` хранят минимальное и максимальное время прохождения по схеме.

В файле данных схема так же, как и точки, идентифицируются своими именами. Однако при работе с ними важной является их не идентификация по имени, как это происходит с точками, а идентификация по первой точке — для соединения схем в поток. Для хранения этой информации предназначен словарь

```
map<int, int> startPointIDtoStSchemeID;,
```

который заполняется при чтении схем.

3.2.3 Процедура чтения

Текстовое представление схем является самым сложным среди всех объектов, используемых в программе. Сложность обуславливается переменным размером данных — разные схемы состоят из разного количества точек, а также опциональностью отдельных элементов представления. Линейная схема и схема со спрямлением разнятся отсутствие или наличием участка спрямления. В схеме со спрямлением могут быть, а могут отсутствовать линейные участки предшествующие и последующие участку спрямления.

Были рассмотрены три возможности организации чтения этих сильно варьирующихся данных:

1. ручной разбор строк;
2. использование библиотеки реализации грамматик;
3. использование регулярных выражений.

Первый и второй подходы показались чрезмерно трудоемкими. Первый — из-за весьма значительного объема кода, который следует написать для разбора всех возможных вариантов входных данных. Второй — из-за высокого порога входа, из-за значительного объема материала, необходимого для изучения, чтобы использовать этот инструмент.

Поэтому в текущей версии программы для разбора строк с информацией о схемах используются регулярные выражение.

Чтение линейных схем и схем со спрямлением.

Для разбора строки с информацией о линейной схеме и схеме со спрямлением используется следующее регулярное выражение:

```
(\w+)\s*
(?:\((\w+)\))?\s*
\(([\w\s]*|\d+)\)\s*
:
(?:
    \s*([\w\s]*)\s+
```

```
    (?:Str
        \(([\w\s]+)\)\s*
        ([\w\s]+)
    \/Str)?\s*
    ([\w\s]*)?
)?
```

Первая группа захвата — это имя схемы. Представляет из себя любой набор букв и цифр.

Вторая группа — это имя точки, с которой начинается схема.

Третья — это список имен конечных точек схемы.

Далее после двоеточия следует основное тело схемы.

Четвертая группа — это точки схемы, предшествующие спрямлению. Если оно отсутствует, то эта группа представляет из себя путь. Однако, в связи с тем, что после этой группы обязательно должен быть хотя бы один пробел, и если после последней точки схемы пробела не будет, то эта точка не попадёт в эту группу захвата. Её нужно искать в седьмой группе.

Пятая группа — это точки, на которые можно спрямляться с точек из шестой группы. Если спрямление в схеме отсутствует, то эти группы пустые.

Седьмая группа — это точки, идущие после спрямления. Или, если нет спрямления в схеме и нет пробела после последней точки, тогда тут находится та самая последняя точка.

Также для удобства записи схем состоящих из двух точек, было введено упрощение записи. Можно вводить только имя схемы, точку начала, точку конца и двоеточие. Что регулируется большой не захватывающей скобкой после двоеточия с вопросительным знаком на окончании.

Если спрямление присутствует, то путь состоит из четвёртой, шестой и седьмой групп захвата.

Заполнение структуры.

В общем случае первая группа сразу же присваивается в первое поле структуры. Второе поле — это результат отображения Имя точки $\mapsto$ идентификатор точки, где в качестве имени выступает вторая группа захвата.

Далее необходимо заполнить массив конечных точек. Для этого используется вспомогательная функция, которая получает строку на вход и массив, который требуется заполнить. Строка читается поэлементно, и с помощью отображения Имя точки $\mapsto$ идентификатор точки заполняет данный массив. В случае, если точки не оказалось среди *checkPoints*, тогда выдаётся ошибка и программа заканчивает свою работу с предупреждением о том, что в такой-то строке такая-та точка не найдена среди контрольных точек.

Затем абсолютно аналогично заполняются поля пять и шесть. Теперь необходимо собрать путь схемы. Для этого используя ту же вспомогательную функцию с четвёртой группой и полем путь, мы собираем точки до спрямления. Затем записываем в путь все элементы из поля пять, и, наконец, используя вспомогательную функцию с седьмой группой и полем путь, мы собираем точки после спрямления.

Если четвёртая и пятая группы пустые, то это означает, что мы имеем дело со схемой, состоящей из двух точек. В этом случае, имя и точка начала схемы заполняются как в общем случае, но также нам известно, что массив конечных точек состоит из одного элемента, поэтому мы можем его сразу же записывать в массив конечных точек. И тогда путь, также легко определяется, как точка начала схемы плюс первый, он же и единственный, элемент массива конечных точек.

Чтение стандартных схем.

Для разбора данных о стандартной схеме используется следующее регулярное выражение:

```
(\w+)\s*
(?:\((\w+)\))?\s*
\(([\w\s]*|\d+)\)\s*
:
(?:
    \s*([\w\s]*)\s+
    (?:Str
      \(([\w\s]+)\)\s*
      ([\w\s]+)
    \/Str)?\s*
    ([\w\s]*)?
)?
```

Первая группа захвата — это имя стандартной схемы. Так как начало и конец совпадают, то вторая группа захвата это второе поле структуры. Третья группа — это количество повторений схемы.

В стандартной схеме нет спрямлений, поэтому пятая и шестая группы всегда пустые. Четвёртая и седьмая вместе дают все три точки. Поэтому с помощью вспомогательной функции из обычных схем, заполняем временный массив *path*, аналогичный одноименному полю из обычных схем. И, затем, заполняем второе, третье и четвёртые поля структуры стандартной схемы как — первый, второй и третий элемент массива *path*. Если же длина этого массива получилась больше трёх, то программа завершает работу и выдаёт ошибку.

Далее заполняется отображение идентификатор точки начала стандартной схемы $\mapsto$ идентификатор стандартной схемы.

3.3 Данные о потоках

Это самый простой файл.

В первой строке хранится количество потоков. Следующие строки хранят информацию о потоках в виде

ИМЯ_ПОТОКА ИМЯ_ПЕРВОЙ_ТОЧКИ

Пример: 2
Flow1 DIPOP
Flow2 AKERA

Однако несмотря на простоту входных данных, структура хранения данных по потоку достаточно содержательна:

```
string name;
int start_point;
map<int, vector<int>> graph_of_descendants;
map<int, vector<int>> graph_of_ancestors;
vector<int> keys;
map<int, vector<pair<Time, Time>>> times;
```

Поле `name` — имя потока, `start_point` — индекс начальной точки потока. Эти поля заполняются непосредственно при чтении из файла.

Остальные поля хранят информацию об устройстве потока и вычисляются после чтения потока. Поля `graph_of_descendants` и `graph_of_ancestors` хранят информацию о связи точек в графе потока (соответственно, индексы потомков и предков каждой точки, входящей в поток и задаваемой своим индексом).

Поле `keys` вспомогательное поле, заполняемое после топологической сортировки графа потока (см. раздел 4.4). Содержит набор индексов точек потока в порядке топологической сортировки. Нужно для обхода и обработки контрольных точек потока в порядке, гарантирующем обработку всех предков точки до обработки ее самой.

Поле `times` хранит результат работы программы: множество возможных моментов прибытия ВС в контрольную точку, индекс которой является ключом в словаре. Множество задается как список (`vector`) непересекающихся интервалов времени (структура `pair<Time, Time>`), упорядоченных по возрастанию левых концов.

3.4 Данные по всей зоне

Как видно из предыдущей части текста, данные, описывающие структуру воздушных трасс, весьма многочисленны и разнообразны. Поэтому они собраны в объект структуры типа

```
struct Zone {
  vector<CheckPoint> checkPoints;
  vector<Scheme> schemes;
  vector<StandardScheme> standardSchemes;
  vector<Flow> flows;
  vector<vector<int>> graph_of_descendants;
};
```

Первые четыре поля — это массивы данных по контрольным точкам в зоне; схемам, линейным/со спрямлением и стандартным; Поле `graph_of_descendants` представляет собой полное описание графа точек во всей зоне. Из него черпаются данные для графов каждого из потоков (см. раздел 3.3).

В целом, данные программы это три глобальных объекта:

- объект типа `Zone`, хранящий информацию о трассах зоны;
- объект `pointNameToID`, хранящий сопоставление имен контрольных точек и их индексов в массиве `checkPoints` (см. раздел 3.1.2);
- объект `startPointIDtoStSchemeID`, хранящий сопоставление имен схем и индексов их начальных точек в массиве `checkPoints` (см. раздел 3.2.2).

4 Работа программы

4.1 Чтение данных

Чтение данных, в целом, представляет собой поочередное чтение данных из файлов и заполнение этих трёх объектов. Сначала считываются точки, потом схемы, затем потоки. Каждая читается с помощью регулярных выражений в соответствии с форматом.

Основная идея одинакова для всех трёх файлов. Процедура, соответствующая файлу, получает на вход путь к нему и массив структур, который требуется заполнить. Далее объявляется счётчик, который отвечает за движение по массиву. Затем делается попытка открыть файл. В случае успеха выводится сообщение об открытии файла, иначе выводится сообщение об ошибке, и программа останавливает свою работу. Если файл был открыт, то первым делом происходит считывание длины массива и проверка на не отрицательность этого числа. В случае, если полученная длина массива не положительна, то происходит остановка программы. Иначе длина массива становится равной полученному числу. Далее объявляется регулярное выражение для работы с данными из файла.

Начинается основной цикл, в котором происходит заполнение массива. Файл читается построчно и построчно обрабатывается регулярным выражением до тех пор, пока не

закончится файл. Первым делом делается проверка на успешность сопоставления регулярного выражения и строки. В случае неуспеха выводится сообщение о несоответствии формату. Также проверяется уникальность имён входных данных. Далее делается проверка на возможность доступа к текущей координате массива. В случае невозможности осуществить это программа заканчивает работу и предупреждает о нехватке выделенного места под элементы массива. Затем идёт заполнение, подробности которого можно найти ниже. И, наконец, после успешного наполнения массива, делается проверка и правка, чтобы действительная длина массива совпадала с декларируемой, если это не так, то производится коррекция, после чего файл закрывается.

4.2 Предобработка считанных данных

После того, как все данные считаны производится их предварительная обработка.

Сначала заполняем поля времён у стандартных схем. Минимальное время — пролёт по окружности с максимальной скоростью, а максимальное — пролёт по всей схеме с минимальной скоростью.

Обозначим точки стандартной схемы $\nu_0 = (x_0, y_0, z_0),\ \nu_1 = (x_1, y_1, z_1),\ \nu_2 = (x_2, y_2, z_2)$ Вычислим радиус поворота как половину от расстояния между ν_0 и ν_1:

$$R = \frac{1}{2} * \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2}$$

Длину плеча находится как расстояние между ν_1 и ν_2:

$$S = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Тогда минимальное и максимальное времена выражаются следующим образом:

$$T_{min} = \frac{2\pi R}{v_{max}}, \qquad T_{max} = \frac{2(S + \pi R)}{v_{min}},$$

где v_{max} и v_{min} — это максимально и минимально возможные скорости первой точки схемы.

Граф зоны будем строить списками следующих. То есть соединяем ребрами данную вершину, со всеми вершинами, которые имеют связь с данной. Делать это будем следующим образом: идём по массиву линейных схем/схем со спрямлением, обрабатывая каждую схему в отдельности. На данном этапе нас интересует поле *path*. Так как в нём лежат вершины в порядке прохождения самолётом схемы, то мы будем соединять текущую точку со следующей. Далее нужно соединить вершины, с которых возможно спрямление со всеми вершинами на которые это спрямление доступно. После чего переходить к другой схеме. Но есть нюанс. Если из предпоследней точки схемы можно спрямиться на конечную, то возникает проблема с добавлением этого ребра дважды —

первый раз как элемент пути, второй как элемент спрямления. Это необходимо учесть.

4.3 Построение графов потоков

Теперь построим граф потока, как подграф графа зоны, также списками потомков. Для этого воспользуемся поиском в глубину. Заведём стек, на который будем складывать не посещённые вершины и массив меток, в котором будем отмечать посещённые вершины. Будем работать до тех пор, пока стек не опустеет. Сначала складываем на него точку начала потока. Используя граф зоны, соединяем эту точку со всеми сыновьями, и если сын ещё не посещён, то добавляем его на стек и ставим метку, что его посетили. В результате получим граф потока. После собираем граф потока, заданный списками предшественников, путём перебора графа, построенного списками потомков.

4.4 Топологическая сортировка графов потоков

К полученным ориентированным графам потоков применяется топологическая сортировка для получения порядка обработки точек потока, гарантирующего, что никакая точка не будет обработана ранее какого-либо из своих предков. Для хранения полученного порядка вершин используется поле *keys* записи потока. Это массив, хранящий идентификатор в топологически отсортированном порядке $\mapsto$ идентификатор в исходном порядке. Сортировка проводится методом полустепеней захода. Заведём переменную *number*, которая будет помогать отсортировывать вершины, инициализируем её нулём. Сначала сделаем размер массива ключей равным количеству вершин в потоке. Затем инициализируем отображение Точка $\mapsto$ полу степень захода следующим образом всем вершинам сопоставляется ноль. Заполняем его так: идя по спискам потомков, увеличиваем значение на единицу за каждого предка у вершины. Заводим стек и складываем на него все вершины с нулевой полустепенью захода. Пока стек не пуст, берём вершину со стека и на место number в массиве записываем идентификатор вершины. Увеличиваем *number* на единицу. И для всех потомков данной вершины уменьшаем на единицу полустепень захода, и, если она стала равна нулю — кладём этого потомка на стек. В результате получаем нужный массив.

4.5 Вспомогательная процедура объединения двух множеств моментов прибытия

На вход дается массив пар действительных чисел, упорядоченных по возрастанию первых элементов и задающих набор отрезков. Отрезки могут вкладываться и/или пересекаться. Необходимо конвертировать этот набор в эквивалентный набор непересекающихся отрезков, произведя объединение пересекающихся поднаборов. Будем делать с помощью отображения число $\mapsto$ действие. Для корректного сравнения действительных

чисел реализованы два компаратора для проверки что одно число больше или меньше другого с точностью ϵ, заданной в классе времени (по умолчанию $\epsilon = 10^{-4}$). Сначала проверим, что пары заданы корректно, то есть первый элемент меньше либо равен второго, иначе программа завершает работу и выдаёт сообщение об ошибке. Затем инициализируем нулевыми значениями отображение и заполняем его, если элемент стоит на первом месте в паре, тогда его значение увеличиваем на единицу, если на втором, то уменьшаем на единицу. Очистим массив времён. Заведём переменную, в которой будем считать сумму, инициализировав нулём. Вместе с тем, запомним первый элемент и назовём его начальным. Идём до конца ассоциативного массива, суммируя с накоплением значения отображения. Если в какой-то момент сумма стала равна нулю, то добавляем пару, состоящую из первого элемента, который мы запомнили, и текущего элемента. Проверяем, если мы не дошли до конца ассоциативного массива, то следующий элемент объявляем начальным и продолжаем цикл. В результате мы объединили вложенные и пересекающиеся интервалы, остались лишь непересекающиеся.

4.5.1 Вычисление возможных времен прибытия ВС в контрольные точки

Будем рассчитывать время для каждого потока по отдельности.

Так как поток топологически отсортирован, мы будем двигаться по возрастанию идентификатора вершин до тех пор, пока не дойдём до вершины с максимальным индексом, рассчитывая возможное время прибытия в следующие точки из данной. Сразу же выполним объединение времён, так как с прошлого шага мы получили набор интервалов, не обязательно не пересекающихся.

Требует отдельного рассмотрения стандартная схема. Если в данная точка является началом стандартной схемы, у которой остались повторения, то нужно к текущим временным интервалам данной точки добавить интервалы, полученные прибавлением T_{min} и T_{max} из полей стандартной схемы к каждому из имеющихся интервалов данной точки, и уменьшить количество повторений этой стандартной схемы на единицу.

Если же стандартной схемы нет или она исчерпала себя, то тогда для всех потомков время вычисляется как равнопеременное:

$$T_{min} = \frac{2S}{v^0_{max} + v^1_{max}}$$

где v^0_{max} — это максимально допустимая скорость на текущей контрольной точке, а v^1_{max} — это максимально допустимая скорость на данном потомке текущей контрольной точке

$$T_{max} = \frac{2S}{v^0_{min} + v^1_{min}}$$

где v^0_{min} — это минимально допустимая скорость на текущей контрольной точке, а v^1_{min} — это минимально допустимая скорость на данном потомке текущей контрольной точке,

а S — евклидово расстояние между точками:

$$S = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2}.$$

После чего добавляем к каждому интервалу полученные T_{min} и T_{max}, и этот новый интервал записываем в массив времён потомка. И увеличиваем счётчик на единицу. После выполнения цикла получаем набор временных интервалов для каждой точки из потока.

Выполняя данную процедуру для каждого из потоков, получим набор временных интервалов для каждой точки.

5 Примеры расчётов

5.1 Простейшие случаи

Рассмотрим работу процедуры в простейших случаях на данном примере:

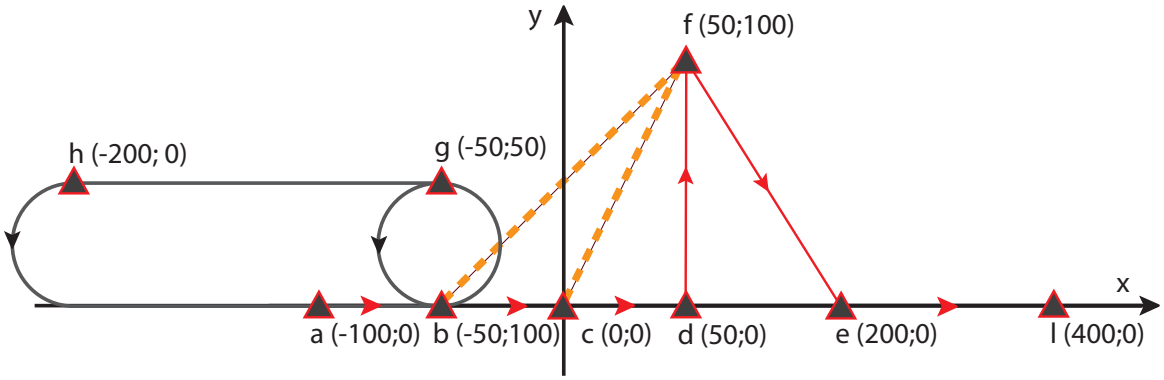

Рис. 6. Макетный пример

Пусть расстояние будет в метрах, а скорость в метрах в секунду.

Файл с данными точек:

9

a -100 0 0 5 10

b -50 0 0 5 10

c 0 0 0 5 10

d 50 0 0 5 10

e 200 0 0 4 6

f 50 100 0 4 8

g -50 50 0 5 10

h -200 50 0 5 10

r 400 0 0 2 5 LAND

Файл с данными схем:

3

1

Name1 (e)(r):

Name2 (a)(e): a Str(f) b c d /Str f e

NameSt1 (0): b g h

Файл с данными потоков:

3

Flow1 e

Flow2 a

5.1.1 Движение по прямой

Рассмотрим движение по прямой на примере первого потока:

$$T_{min} = \frac{2S}{v^e_{max} + v^r_{max}} = \frac{2 \cdot \sqrt{(400-200)^2}}{6+5} \approx 36.(36)\,\text{c}$$

$$T_{max} = \frac{2S}{v^e_{min} + v^r_{min}} = \frac{2 \cdot \sqrt{(400-200)^2}}{4+2} \approx 66.(66)\,\text{c}$$

И результат процедуры:

Flow1:

e $\rightarrow$ [0 c, 0 c]

r $\rightarrow$ [36.36 c, 66.67 c]

5.1.2 Движение по вееру и прямым участкам

Движение по вееру и прямым участкам рассмотрим на примере второго потока. Рассчитаем временной интервал для точки f:

$$T_{min} = \frac{2S_{a-b}}{v^b_{max} + v^a_{max}} + \frac{2S_{b-f}}{v^f_{max} + v^b_{max}} = \frac{2 \cdot \sqrt{(-50+100)^2}}{10+10} + \frac{2 \cdot \sqrt{(50+50)^2 + (100-0)^2}}{8+10} \approx$$

$$\approx 20.71348\,\text{c} \quad (1)$$

$$T_{max} = T^{a-b}_{max} + T^{b-c}_{max} + T^{c-d}_{max} + T^{d-f}_{max} = \frac{2S_{a-b}}{v^b_{min} + v^a_{min}} + \frac{2S_{b-c}}{v^c_{min} + v^b_{min}} + \frac{2S_{c-d}}{v^d_{min} + v^c_{min}} + \frac{2S_{d-f}}{v^f_{min} + v^d_{min}} =$$

$$= \frac{2 \cdot \sqrt{(-50+100)^2}}{5+5} + \frac{2 \cdot \sqrt{(0+50)^2}}{5+5} + \frac{2 \cdot \sqrt{(50-0)^2}}{5+5} + \frac{2 \cdot \sqrt{(50-50)^2 + (100-0)^2}}{5+4} =$$

$$= 10 + 10 + 10 + 22.(2) = 52.(2)\,\text{c} \quad (2)$$

Результат процедуры:

Flow2:

a $\rightarrow$ [0 c, 0 c]

b $\rightarrow$ [5 c, 10 c]

c $\rightarrow$ [10 c, 20 c]

d $\rightarrow$ [15 c, 30 c]

f $\rightarrow$ [20.71 c, 52.22 c]

e → [46.47 с, 97.29 с]
r → [82.83 с, 163.96 с]

5.1.3 Движение по вееру и прямым участкам со стандартной схемой

Теперь изменим файл схем следующим образом — добавим одно повторение к стандартной схеме:
3
1
Name1 (e)(r):
Name2 (a)(e): a Str(f) b c d /Str f e
NameSt1 (1): b g h

И запустим поток с точки a:
1
Flow1 a

Посчитаем время минимальной и максимальной задержки для этой стандартной схемы:

Вычислим радиус поворота, как половину от расстояния между b и g:

$$R = \frac{1}{2} \cdot \sqrt{(-50+50)^2 + (50-0)^2} = 25\,\text{м}$$

Длину плеча найдём, как расстояние между h и g:

$$S = \sqrt{(-200+50)^2 + (50-50)^2} = 150\,\text{м}$$

$$T_{min} = \frac{2\pi R}{v^b_{max}} = \frac{2\pi \cdot 25}{10} = 5\pi \approx 15.7\,\text{с}$$

$$T_{max} = \frac{2(S+\pi R)}{v^b_{min}} = \frac{2(150+\pi \cdot 25)}{5} = 60 + 10\pi \approx 91.4\,\text{с}$$

Результат процедуры:
Flow1:
a → [0 с, 0 с]
b → [5 с, 10 с] [20.71 с, 101.42 с]
c → [10 с, 20 с] [25.71 с, 111.42 с]
d → [15 с, 30 с] [30.71 с, 121.42 с]
f → [20.71 с, 143.64 с]
e → [46.47 с, 188.71 с]
r → [82.83 с, 255.37 с]

Как можно заметить, появился новый временной интервал, соответствующий проходу по стандартной схеме.

5.2 Расчёты для аэропорта «Кольцово»

Рассмотрим работу программы на примере аэропорта Кольцово.

Общий вид зоны Кольцово представлен на рис. 7. Вид внутренней зоны представлен на рис. 8

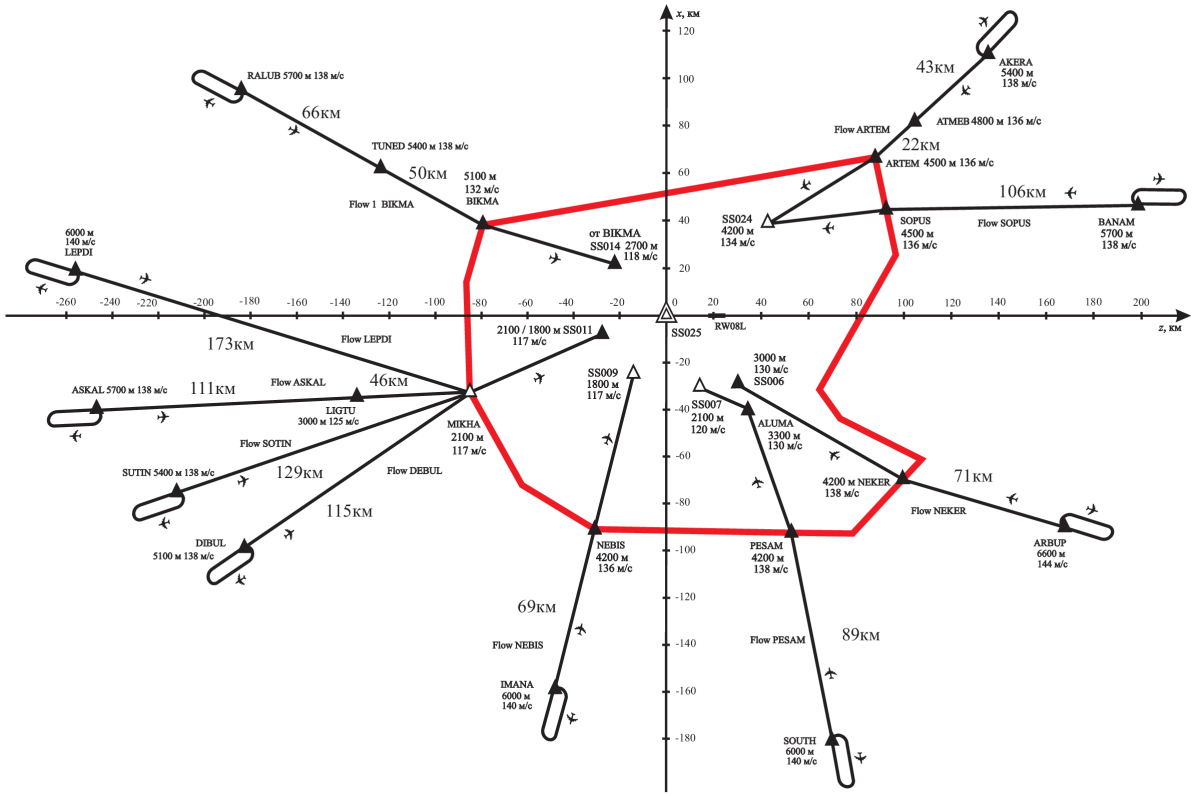

Рис. 7. Общий вид зоны Кольцово

Рассмотрим поток BIKMA начинающийся в точке RALUB.

Результаты работы процедуры для потока BIKMA с нулём повторений стандартной схемы:

Flow1BIKMA:

RALUB → [0 с, 0 с]

TUNED → [474.52 с, 548.67 с]

BIKMA → [818.27 с, 947.42 с]

SS014 → [1240.52 с, 1443.1 с]

SS015 → [1482.57 с, 1734.04 с]

SS025 → [1578.33 с, 1853.75 с]

SS003 → [1684.27 с, 1991.08 с]

RW08L → [1792.31 с, 2132.36 с]

Результаты работы процедуры для потока BIKMA с однократным повторением стан-

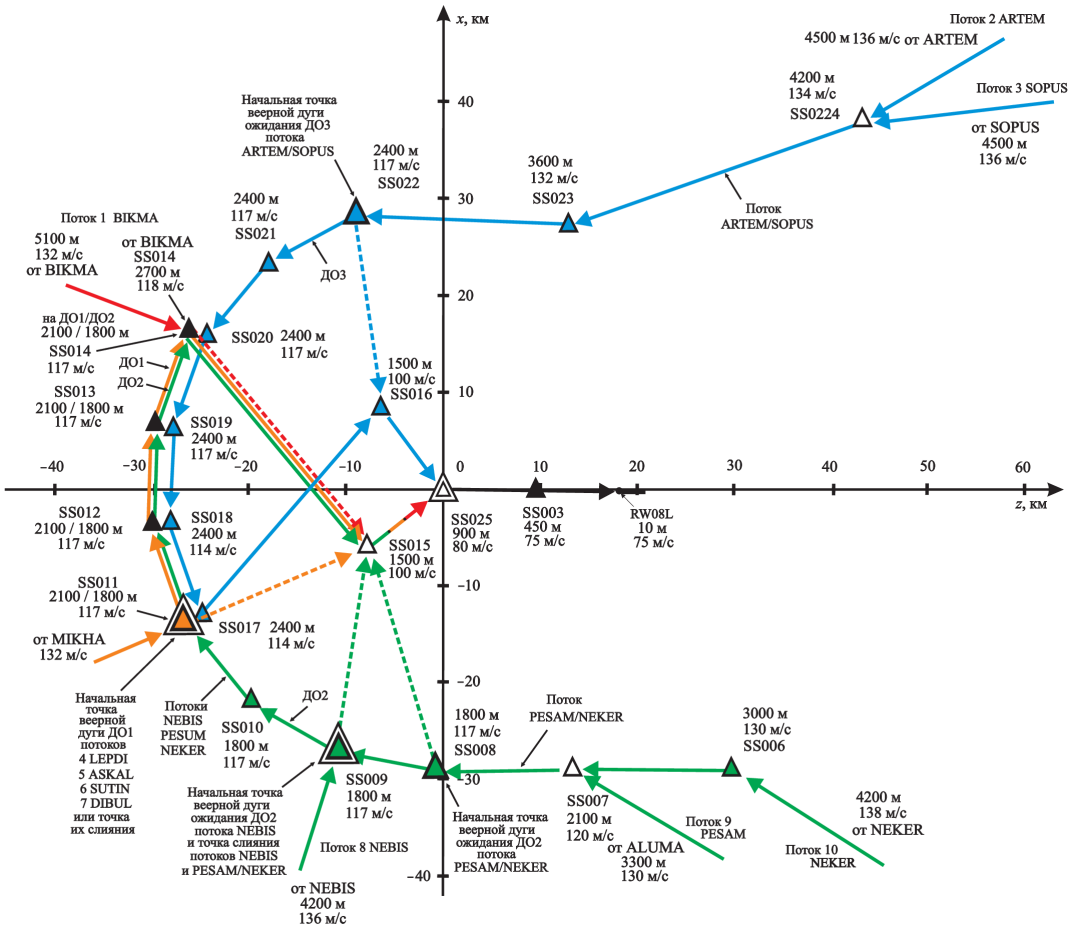

Рис. 8. Прибывающие потоки и веерные схемы их слияния.

дартной схемы:

Flow1BIKMA:

RALUB → [0 c, 0 c] [389.33 c, 992.17 c]

TUNED → [474.52 c, 548.67 c] [863.86 c, 1540.84 c]

BIKMA → [818.27 c, 947.42 c] [1207.61 c, 1939.58 c]

SS014 → [1240.52 c, 1443.1 c] [1629.85 c, 2435.26 c]

SS015 → [1482.57 c, 1734.04 c] [1871.9 c, 2726.21 c]

SS025 → [1578.33 c, 1853.75 c] [1967.66 c, 2845.91 c]

SS003 → [1684.27 c, 1991.08 c] [2073.6 c, 2983.24 c]

RW08L → [1792.31 c, 2132.36 c] [2181.64 c, 3124.52 c]

Рассмотрим поток NEKER начинающийся в точке ARBUP.

Результаты работы процедуры для потока NEKER с нулём повторений стандартной схемы:

Flow10NEKER:

ARBUP → [0 c, 0 c]

NEKER → [463.24 c, 533.96 c]

SS006 → [1015.68 c, 1175.5 c]
SS007 → [1140.14 c, 1321.61 c]
SS008 → [1249.54 c, 1451.18 c]
SS009 → [1331.5 c, 1548.45 c]
SS010 → [1413.66 c, 1645.98 c]
SS011a → [1496.1 c, 1743.82 c]
SS012a → [1578.68 c, 1841.84 c]
SS013a → [1661.3 c, 1939.91 c]
SS014aaa → [1743.95 c, 2038 c]
SS015 → [1455.79 c, 2330.19 c]
SS025 → [1551.56 c, 2449.89 c]
SS003 → [1657.5 c, 2587.22 c]
RW08L → [1765.54 c, 2728.5 c]

Результаты работы процедуры для потока NEKKER с однократным повторением стандартной схемы в точке ARBUP:
Flow10NEKER:
ARBUP → [0 c, 0 c] [234.91 c, 925.33 c]
NEKER → [463.24 c, 533.96 c] [698.15 c, 1459.29 c]
SS006 → [1015.68 c, 1175.5 c] [1250.59 c, 2100.83 c]
SS007 → [1140.14 c, 1321.61 c] [1375.04 c, 2246.93 c]
SS008 → [1249.54 c, 1451.18 c] [1484.45 c, 2376.51 c]
SS009 → [1331.5 c, 1548.45 c] [1566.4 c, 2473.78 c]
SS010 → [1413.66 c, 1645.98 c] [1648.57 c, 2571.31 c]
SS011a → [1496.1 c, 2669.15 c]
SS012a → [1578.68 c, 2767.16 c]
SS013a → [1661.3 c, 2865.24 c]
SS014aaa → [1743.95 c, 2963.33 c]
SS015 → [1455.79 c, 3255.51 c]
SS025 → [1551.56 c, 3375.21 c]
SS003 → [1657.5 c, 3512.54 c]
RW08L → [1765.54 c, 3653.83 c]

Заключение

В результате работы была решена поставленная задача о вычислении временных интервалов прибытия воздушного судна на контрольные точки. Процесс работы состоял в изучении задачи, выборе структур данных и алгоритмов. Реализации на языке C++ 14-го стандарта. В качестве примера была рассмотрена схема аэропорта «Кольцово» в городе Екатеринбурге.

Список литературы

[1] Асанов, М. О. Дискретная математика: графы, матроиды, алгоритмы : учебное пособие / М. О. Асанов, В. А. Баранский, В. В. Расин. — 2-е изд. испр. и доп. СПб.: Лань, 2010. 368 с.

[2] Спиридонов А.А. Кумков С.С. Разработка и исследование формализаций задачи бесконфликтного слияния потоков воздушных судов. Отчёт о научно-экспериментальной работе «Алгоритмы и программное обеспечение обработки информации в АС УВД» по Договору №29-19У. (Этап 1). Том 3, июнь 2019 г., 25 стр.

[3] Проект структуры воздушного пространства МУДР. Стандартные маршруты прибытия на аэродром Шереметьево. Версия 03 ПСВП МУДР 8.5.1 (таблицы). ГосНИИГА, Москва, 2016.

[4] Проект структуры воздушного пространства МУДР. Схемы захода на посадку на аэродром Шереметьево. Версия 03 ПСПВ. Стандартные маршруты прибытия на аэродром Шереметьево. Версия 04 ПСВП МУДР 8.5.1 (схемы). ГосНИИГА, Москва, 2016.

[5] Point Merge Integration of Arrival Flows Enabling Extensive RNAV Application and Continuous Descent. Operation Services and Environment Definition. Report, July 2010. Eurocontrol Experimental Center, Bretigny-sur-Orge.
`http://www.eurocontrol.int/eec/gallery/content/public..`

[6] Air Traffic Management Technology Demonstration–1 (ATD–1). NASA Report FS-2011-10-01-ARC. 2011.

[7] Пятко С.Г., Красов А.И. и др. Автоматизированные системы управления воздушным движением. Новые информационные технологии в авиации. СПб.: Изд. Политехника, 2004.

[8] Королев Е. Н. Технологии работы диспетчеров управления воздушным движением. М.: Воздушный транспорт, 2000.

[9] ГОСТ 20058-80. Динамика летательных аппаратов в атмосфере. Термины, определения и обозначения. М.: Госстандарт, 1980.