ПРИЛОЖЕНИЕ Б

(справочное)

АППАРАТНО-ПРОГРАММНЫЙ КОМПЛЕКС «ЛЕСНОЙ ПОЖАР» ЭСКИЗНАЯ ТЕХНИЧЕСКАЯ ДОКУМЕНТАЦИЯ МАКЕТА

МОДУЛЬ «КЛАСТЕРИЗАЦИЯ ОЧАГОВ ПОЖАРОВ» ДЛЯ ВИЗУАЛИЗАЦИИ РЕЗУЛЬТАТОВ ПРОГНОЗА НЕГАТИВНЫХ ПОСЛЕДСТВИЙ ЛЕСНЫХ И ТОРФЯНЫХ ПОЖАРОВ

ОПИСАНИЕ ПРОГРАММЫ

046. 02068568. 00001-01 13 04

Листов 18

2012

Подп. и дата | Инв. № дубл. | Взамен № | Подп. и дата | Инв. № подл.

СОДЕРЖАНИЕ:

1 АННОТАЦИЯ

В данном программном документе приведено описание программы «Klaster.exe», предназначенной для кластеризации данных обработки космических снимков очагов пожаров и очагов в прямоугольные области, для последующего отображения в ГИС и использования в моделях расчета факелов лесных и торфяных пожаров. Исходным языком программы «Klaster.exe» является C++. Среда разработки – CBuilder 6 фирмы Borland.

Основной функцией программы «Klaster.exe» является создание кластеров пожаров – объединения очагов пожаров, рассматриваемых как единый пожар. В результате создается файл с данными о координатах центра и размерами пожаров для обеспечения исходными данными расчетного модуля и визуализации полученных результатов.

Оформление программного документа «Описание программы» произведено по требованиям ЕСПД (ГОСТ 19.101-77 [1)], ГОСТ 19.103-77 [2)], ГОСТ 19.104-78* [3)], ГОСТ 19.105-78* [4)], ГОСТ 19.106-78* [5)], ГОСТ 19.402-78* [6)], ГОСТ 19.604-78* [7)]).

[1)] ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов
[2)] ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов
[3)] ГОСТ 19.104-78* ЕСПД. Основные надписи
[4)] ГОСТ 19.105-78* ЕСПД. Общие требования к программным документам
[5)] ГОСТ 19.106-78* ЕСПД. Общие требования к программным документам, выполненным печатным способом
[6)] ГОСТ 19.402-78* ЕСПД. Описание программы
[7)] ГОСТ 19.604-78* ЕСПД. Правила внесения изменений в программные документы, выполненные печатным способом

2 ОБЩИЕ СВЕДЕНИЯ

2.1 Обозначение и наименование программы

Программа “Кластеризация очагов пожаров”» имеет следующие атрибуты:

- Наименование исполняемого файла - Klaster.exe
- Размер исполняемого файла - 230 Кбайт
- Версия файла - 1.1.0.0
- Версия продукта - 1.01.0003
- Внутреннее имя - Klaster
- Исходное имя файла - Klaster.exe
- Название продукта - Кластеризатор очагов
- Описание версии файла - 1.01.0003
- Производитель - РГГМУ
- Язык - Русский

2.2 Программное обеспечение, необходимое для функционирования программы

Программа работает под управлением ОС Windows 2000/XP/Vista/7.

2.3 Языки программирования, на которых написана программа

- Исходным языком программирования является C++. Среда разработки, компилятор – BorlandC v.6 .

3 ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

3.1 Назначение программы

Программа предназначена для группирования пикселей, отображающих очаги пожаров на космических снимках, в кластеры прямоугольной формы. Построение кластеров выполняется в географической и декартовой системах координат по текстовым данным о центрах очагов и размерам пикселей (снимков космического зондирования).

Программа решает следующие задачи:

– Переводит координаты пикселей, описывающих очаги пожаров из географической системы в прямоугольную систему координат;

– вычисляет расстояния между центрами очагов пожаров в специальной метрике, влючающей как пространственные координаты, так и меру термического влияния пожаров друг на друга;

– обединяет очаги пожаров в группы - кластеры, исходя из принципа близости в метрике пространственно-термических измерений. Критерием близости является эмпирически подобранная величина, позволяющая так группировать очаги, чтобы с одной стороны, выделенные области (кластеры очагов горения) не были слишком велики и допускали последующее 3D моделирование с высоким пространственным разрешением, а с другой – не взаимодействовали динамически и термодинамически друг с другом.

– создает файл параметров кластеров, содержащий также и основные данные входной информации, включая координаты центра каждого очага пожара, площадь пикселя, исходную температуру и выброс из очага пожара.

3.2 Сведения о функциональных ограничениях на применение

Программа позволяет обрабатывать только исходные текстовые файлы определенного формата. Размеры входного и выходных файлов ограничены 20 000 очагов пожаров.

4 ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

4.1 Режим работы программы

Программа работает в автоматическом режиме (основной вид использования в составе комплекса ГИС «Лесной пожар»). Для отладки и тестирования может выполняться в режиме автономного запуска для всей области исследования очагов пожаров в файле с исходными данными.

Результаты работы программы сохраняются в виде двух текстовых файлов фиксированной структуры, длиной не более 20000 записей (очагов).

4.2 Структура программы

Программа состоит из одного модуля, включающего основную функцию (main) и ряд вспомогательных функций. Имя программы Klaster, перечень входящих в программу функций приведен в таблице 1.

Таблица 1. Перечень подпрограмм-функций программы Klaster

Имя	Тип рез-та	Назначение
main	int	Основная процедура группировки очагов в кластеры
geographicalToRectangular	void	Преобразование географических координат в декартовые
f00	double	Расчет коэффициента геоида пересчета координат
dist	double	Расчет расстояния между двумя очагами в метрике пространство-температура

4.3 Описание Алгоритма работы модуля

Поскольку данные ИСЗ дают нам весьма разнообразные по яркостной температуре пиксели, как активному горению, так и угасающим участкам очагов пожара, то при кластеризации возникает необходимость использовать не только координаты точек, но и зафиксированную в них температуру, поскольку интерес представляют, в первую очередь, именно участки активного горения. Для этого

при кластеризации целесообразно применять не обычную евклидову метрику расстояния между пикселями, а метрику, использующую помимо взаимного положения пикселей еще и разность зафиксированных в них температур. В рамках данной работы использовалась следующая метрика взаимного положения пикселей $p = (p_x, p_y, p_t)$ и $q = (q_x, q_y, q_t)$:

$$|p,q| = \left((p_x - q_x)^2 + (p_y - q_y)^2\right)^{1/2} + k|p_t - q_t| \qquad \textbf{(4.1)}$$

где k – размерный коэффициент увязки влияния температуры точек, индексами x и y обозначены координаты пикселя (прямоугольные или географические), а индексом t – температура.

Существует значительное количество алгоритмов для решения задач такого типа кластеризации, которые ищут оптимальное разбиение точек на кластеры по разным нормам и разным показателям оптимальности (например, [2,4,5]). Некоторые из них накладывают дополнительные ограничения на применяемую норму. В большинстве случаев - это требование выполнения неравенства треугольника. Нетрудно видеть, что в нашем случае это условие выполнено.

В качестве критерия оптимальности может применяться диаметр кластера, средняя удаленность от центра кластера, среднее по-парное расстояние, размеры окаймляющего кластер прямоугольника, количество кластеров. Для целей решаемой нами задачи критериями оптимальности будут отклонение от желаемого количества кластеров и размеры окаймляющего прямоугольника, поскольку основные ограничения задачи моделирования – это целостность очага пожара и ограниченность области моделирования.

Наиболее подходящим для нашей задачи следует признать алгоритм [6], основанный на выделении ровно m центров кластеров и дальнейшем отнесении оставшихся точек к ближайшему центру. Задача выделения m центров является NP-полной, т.е. построение точного алгоритма, который находил бы оптимальное решение за полиномиальное от количества точек время, является в настоящее время неразрешимой задачей. Таким образом, для значительных по размерам

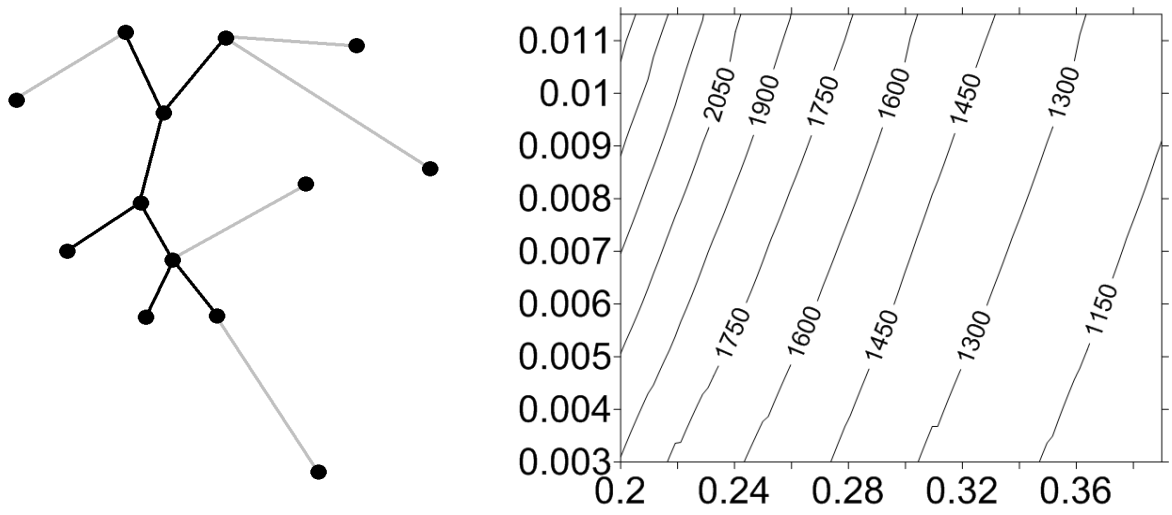
входных данных возможно использование лишь приближенных алгоритмов. Предложенный приближенный алгоритм находит решение не более чем в 2 раза худшее, чем оптимальное. Суть самого алгоритма заключается в последовательном нахождении очередной точки, которая будет добавлена последней при применении алгоритма Дейкстры [3] из уже найденных точек.

Распространение очага пожара зависит от многих факторов, поэтому является неравномерным по направлениям. Вследствие этого очаги пожара зачастую могут иметь неправильную форму. По этой причине применение описанного алгоритма кластеризации не дает желаемых результатов, поскольку строит выпуклые кластеры наименьшего диаметра, а не в форме очагов пожаров. Для решения этой проблемы предлагается алгоритм кластеризации на основе построения скелета (остовного дерева) точек с последующим его разделением на кластеры.

Рассмотрим полный граф на множестве всех точек зондирования с весами на ребрах, равными расстоянию между точками в метрике (1). Используя алгоритм Прима [3], выделим основное дерево, т.е. минимальный по сумме длин набор ребер при котором граф остается связным.

Алгоритм заключается в следующем. Будем строить дерево, последовательно добавляя в него ребра. На каждом следующем шаге выбираем из вершин еще не принадлежащих дереву ту вершину, у которой длина кратчайшего ребра до вершин уже построенного дерева минимальна. Добавим ее в дерево с ее кратчайшим ребром. Поскольку изначальный граф связный (в нашем случае он полный), то данную операцию возможно проводить до тех пор пока в дерево не будут добавлены все вершины. Также следует заметить, что поиск вершины для добавления можно производить не перебором всех вершин и всех ребер из нее, а сохранив для каждой вершины не принадлежащей дереву кратчайшее ребро до вершин дерева. Блок-схема алгоритма показана на рисунке 4.1.

Рисунок 4.1 – Блок схема алгоритма кластеризации.



а) б)

Рисунок 4.2 - Пример построенного фрагмента остовного дерева и кратчайшие ребра до него (а) и расчетная зависимость количества кластеров от порогового значения $D_{\max}$ и коэффициента K (б).

При добавлении очередной вершины в дерево кратчайшее ребро для какой-то другой вершины может измениться только в том случае, если ребро до вновь добавленной вершины короче сохраненного. Время обновления массива кратчайших ребер пропорционально количеству вершин, в то время как непосредственное его вычисление – квадрату количества вершин. Работа алгоритма проиллюстрирована на рисунке 4.2а. Текущее дерево выделено черными ребрами, кратчайшие ребра для остальных вершин – серыми. На очередном шаге будет взято наименьшее серое ребро и добавлено в дерево.

Как показали проведенные эксперименты, построенный таким образом остов содержит в основном ребра, лежащие внутри очагов пожара, соединяя кроме этого лишь ближайшие очаги единичными ребрами большой длины. После исключения ребер с длиной большей порогового значения $D_{\max}$, граф распадается на компоненты связности, которые и соответствуют очагам пожаров (или группам близкорасположенных очагов).

Следует заметить, что поскольку данный алгоритм использует в метрике

расстояние между точками на всем наборе точек зондирования, т.е. на поверхности всего земного шара, то логичнее было бы в формуле (4.1) использовать вместо выражения $\left((p_x - q_x)^2 + (p_y - q_y)^2\right)^{1/2}$, которое соответствует расстоянию в проекции Меркатора, расстояние между точками на эллипсоиде. Однако в этом нет необходимости, поскольку результат работы алгоритма Прима существенно зависят только от соотношения расстояний близкорасположенных точек, которое несущественно различается из-за достаточной густоты расположения точек зондирования в районах пожаров.

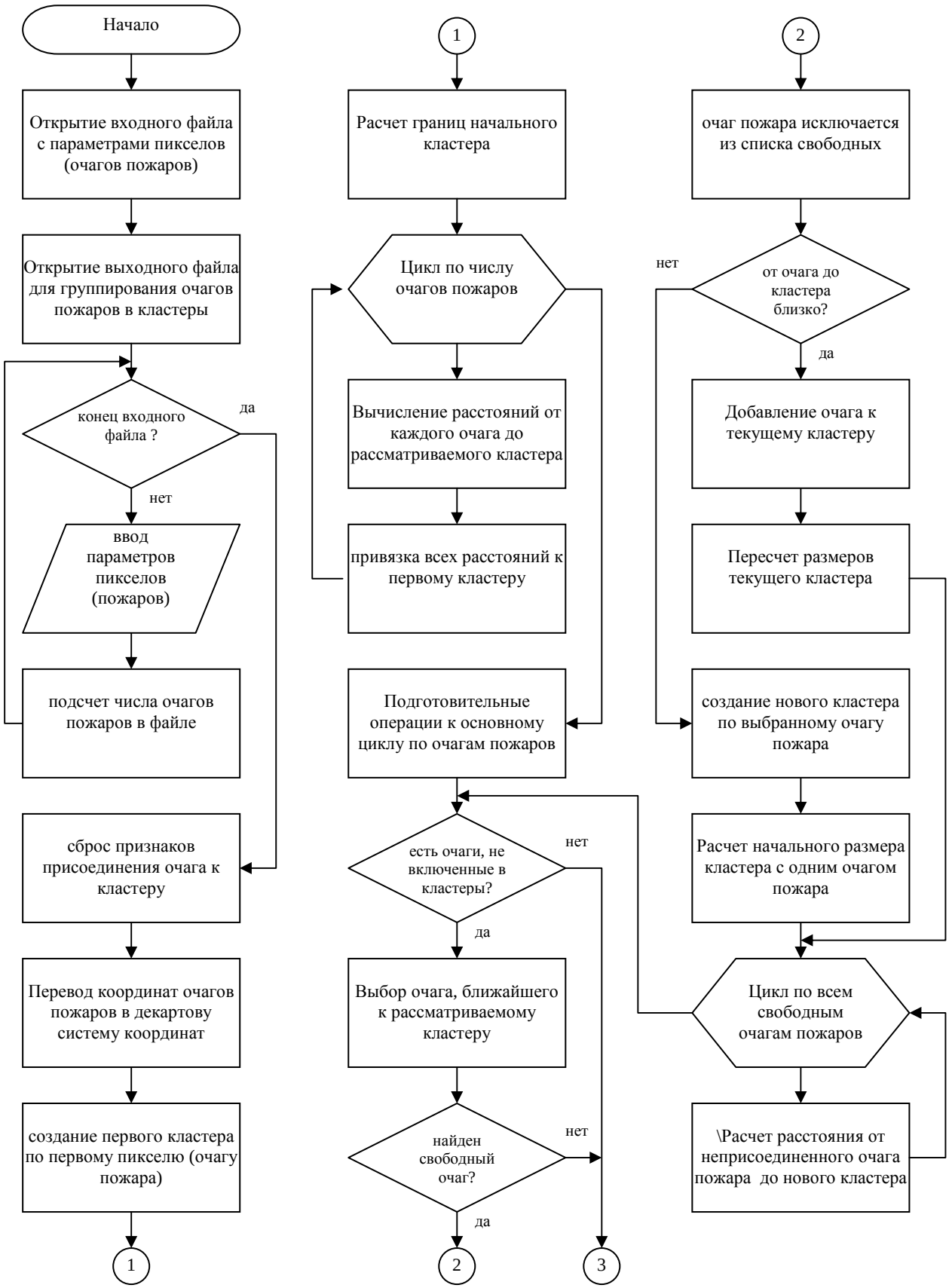


Рисунок П.Б.1. Блок-схема алгоритма основной процедуры модуля (начало)

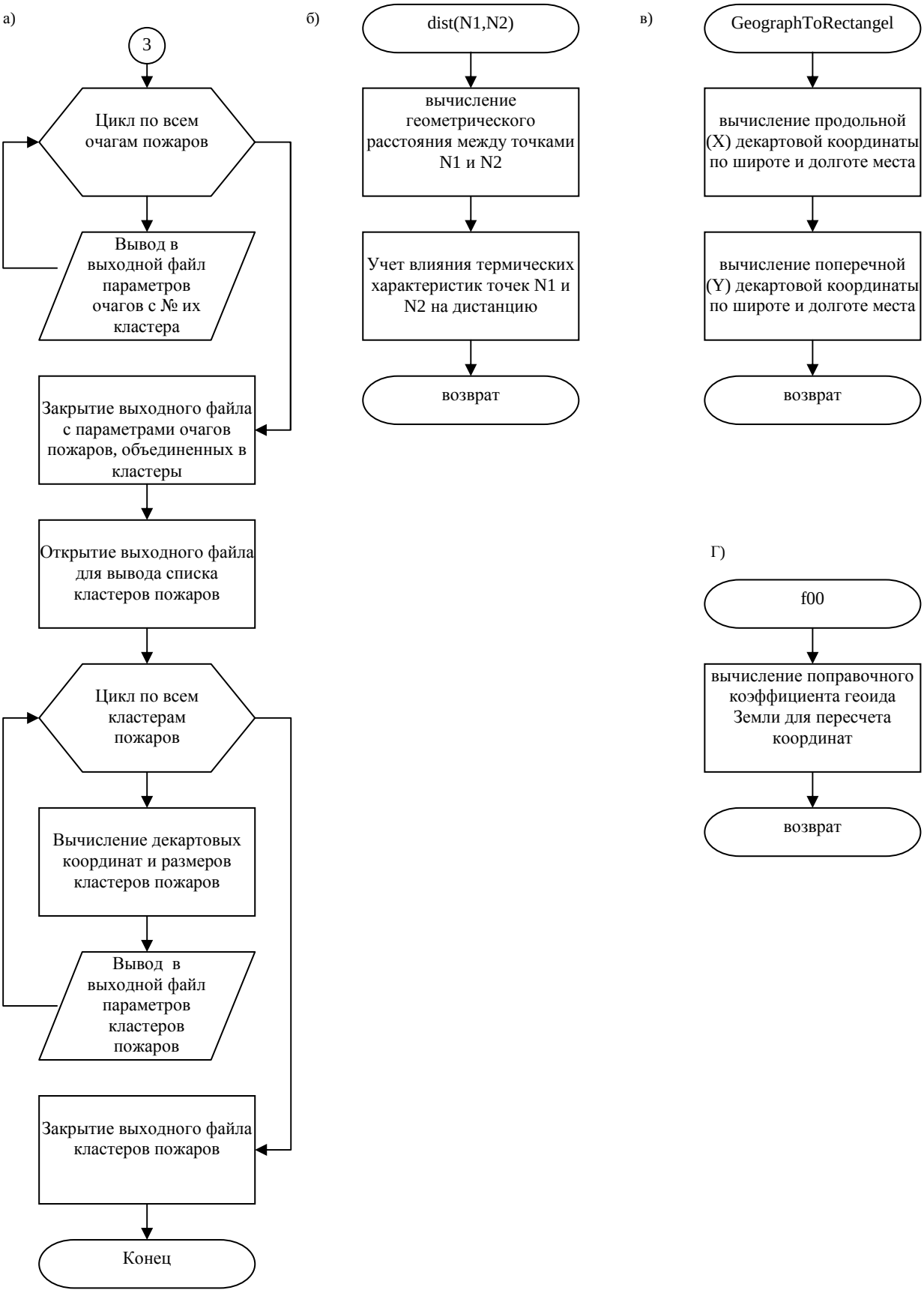


Рисунок П.Б.2 а) - Блок схема алгоритма основной процедуры (окончание); б) – алгоритм функции disat(), в)- алгоритм функции geographicalToRectangular(), г) – алгоритм функции f00().

4.4 Связи между составными частями программы

Связи между основной процедурой и функциями программы выполняются в виде стандартных вызовов подпрограмм.

4.5 Связи программы с другими программами

Программа вызывается из программы ГИС Лесной пожар (модуль ZWF) на выполнение после выбора области исследования очагов пожаров перед визуализацией рассчитанных кластеров очагов пожаров. Вызов выполняется с параметрами, задающими имя файла с данными, котрые необходимо кластеризовать и имена файлов очагов пожаров и кластеров, создаваемых программой.

5 ИСПОЛЬЗУЕМЫЕ ТЕХНИЧЕСКИЕ СРЕДСТВА

Программа эксплуатируется на персональном компьютере (ПК) типа IBM. Режим работы – в форме закрытого консольного приложения, без общения с оператором. Для работы в режиме отладки используется экран дисплея, клавиатура и манипулятор типа "мышь". Входные и выходные данные хранятся на жестком диске.

Требования к ПК:

- ПЭВМ с процессорами типа Intel Pentium под управлением ОС Windows 2000/XP/Vista/7.
- Свободное дисковое пространство не менее 1 ГБ.
- Оперативная память не менее 1ГБ.

6 ВЫЗОВ И ЗАГРУЗКА

После инсталляции программы на жесткий диск, ее вызов осуществляется стандартным способом по имени, с указанием трех параметров – имени файла с исходными данными, имени файла с характеристиками очагов пожаров, дополненных номером кластера, в который входит каждый очаг и имя файла с характеристиками каждого кластера. Имя программы- Klaster.exe.

7 ВХОДНЫЕ ДАННЫЕ

В качестве входных данных используется следующая информация:

– Текстовый файл с данными космического зондирования очагов пожаров в форме координат пикселей и параметров очагов пожаров.

Макет входного файла (последовательность данных) представлен ниже. Разделители чисел – пробелы, дробная часть отделяется «.» (точкой).

Таблица 2. Макет записи входного текстового файла.

N п.п	Параметр	Размерность	Формат
1	долгота центра пикселя	градусы	XXX.XXXX
2	широта центра пикселя	градусы	XXX.XXXX
3	размер пикселя по долготе	км	XX.XXXX
4	размер пикселя по широте	км	XX.XXXX
5	время (гринвич)	час	XX
6	день	-	XX
7	месяц	-	XX
8	год	-	XXXX
9	температура	C	XXX.XXX
10	выброс	кг/с	XX.XXX

8 ВЫХОДНЫЕ ДАННЫЕ

В качестве выхродных данных используется следующая информация:

– текстовый файл с исходной информацией, дополненный номером кластера, в который входит каждый очаг пожара;

– текстовый файл с характеристиками каждого кластера очагов пожаров.

Информация записывается в ссответствующих подкаталогах каталога с установленной программой.

9 ПОИСК НАИЛУЧШИХ ЗНАЧЕНИЙ ПАРАМЕТРОВ И МЕТОДЫ ТЕСТИРОВАНИЯ

Для определения наилучших значений порогового значения $D_{\max}$ и коэффициента увязки K был проведен комплекс расчетов кластеров по архиву данных космического зондирования за лето 2011 г. для широкого диапазона значений параметров. Данные включают в себя следующие поля: географические координаты точки, размеры прямоугольника, на котором зафиксировано превышение температуры, дата, температура и выброс. Исследование показало, что наилучший результат в смысле решения поставленной в начале данной работы задачи достигается при количестве кластеров близком к 1700. Зависимость количества кластеров от порогового значения $D_{\max}$ и коэффициента K показана на на рисунке 4.2б.

Из рисунка видно, что значение 1700 задает нам лишь изолинию взаимозависимости параметров, однако по смыслу понятно, что ни расстояние, ни разность температур не может превалировать в определении метрики, поэтому следует использовать такое значение K, при котором расстояние и разность температур вносят одинаковый вклад в определение метрики. Взяв в качестве такого значения 0.0075°/°K получает значение $D_{\max}$ равным 0.15°.

В случае, если на каком-либо наборе данных эти значения дают неудовлетворительный результат, то возможно их определение с использованием экспертной оценки. При этом настроечным параметром будет количество кластеров, а оцениваться будет качество разбиения на кластеры при вычисленных значениях $D_{\max}$ и K.

Пример использования предложенного алгоритма применительно к данным на 7 октября 2011 года приведен на рисунке 4.3. Очаги представлены черными квадратами и обрамлены прямоугольниками кластеров.

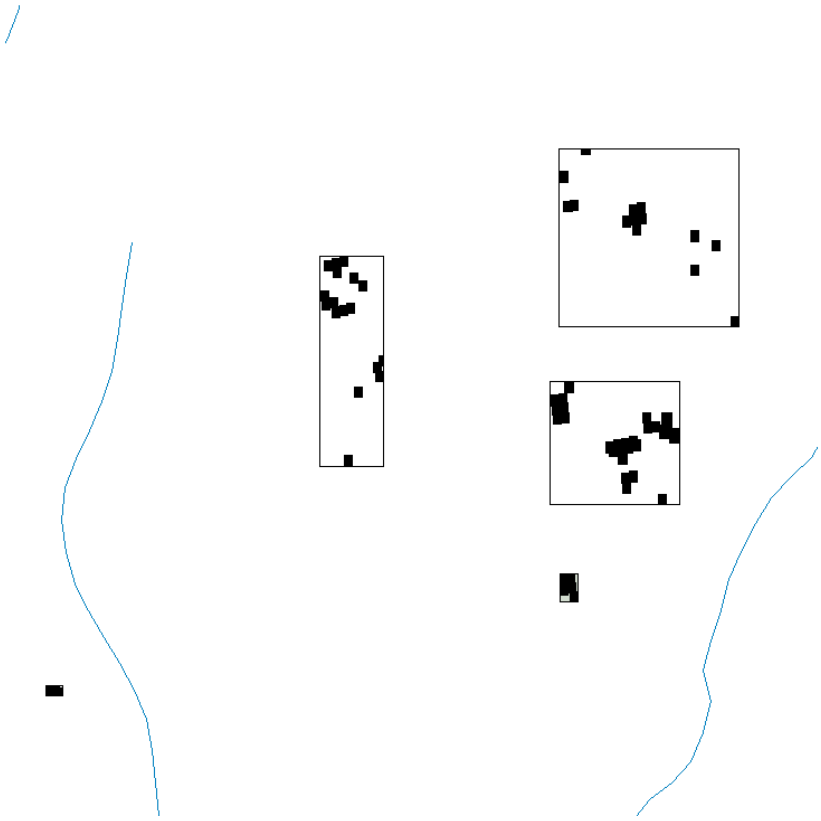


Рисунок 4.3 – Примеры очагов лесного пожара, сгруппированные при помощи предложенного алгоритма кластеризации.

Отдельные точки, выделенные алгоритмом в самостоятельные кластеры, являются точками с низкой температурой и интереса для моделирования не представляют.

Таким образом, предложенный алгоритм оказывается весьма надежным инструментом для автоматического выделения локальных областей территории с группами очагов лесных или торфяных пожаров для последующего численного моделирования переноса и рассеяния в атмосфере продуктов сгорания.

Тестирование программы проводилось методом экспертной оценки для данных дистанционного зондирования на следующие даты: 20 июня, 17 июля, 8 августа, 27 августа, 20 сентября, 7 октября 2011 года, которые отражают разные фазы развития интенсивности и конфигурации очагов горения.

10 ТЕКСТ ПРОГРАММЫ

```
#include<stdio.h>
#include<math.h>
#include <algorithm>

/*   Проект Пожары". Программа группировки очагов пожаров в кластеры
   с созданием файла харакеристик каждого кластера в виде текстового файла.
      Разработка на C в качестве отдельного EXE-модуля. Макетный вариант.
      Начало разработки:  22.02.12
      Оончание разработки:10.05.12
      Программист Чихачев КБ, РГГМУ
*/
#define min(X,Y) ((X) < (Y) ? (X) : (Y))
#define max(X,Y) ((X) > (Y) ? (X) : (Y))

// Константы для функции преобразования географических координат в прямоугольные
double A = 6378245.0;
double E2 = 0.0066934216;
double E12 = E2 / (1 - E2);
double E1 = (1 - sqrt(1 - E2)) / (1 + sqrt(1 - E2));
double FALSENORTHING = 0;
double FALSEEASTING = 500000;
double SCALEFACTOR = 1;
double zona;

double const pi=4*atan(1);

// Мавсимально допустимое количество пикселей
int const maxn=20000;

// Массив исходных данных - описание пиклелей пожаров
struct firepoint{
  double lat,lon,x,y,t,dx,dy;
  int h,d,m,year;
} data[maxn];
int n = 0;

// Массив для хранения кратчайшего расстояния по одному ребру до дерева для всех
вершин, не принадлежащих дереву
double d[maxn];
// Массив для хранения ребер, на которых это кратчайшее расстояние достигается
int to[maxn];
// Массив для хранения номера кластера по номеру пиксела
int comp[maxn];
// Массив для хранения границ кластера по номеру кластера: индексы 1 и 2 - левый
верхний угол, 3 и 4 - правый нижний
double compxy[maxn][4];
// Текущее количество кластеров
int compcount;
int compnum;
// Длина, начиная с которой ребра выкидываются из дерева
double Emax=0.15; //0.26
// Коэффициент увязки расстояния и температуры
double k=0.0075;
// Массив пометок принадлежности вершин дереву
bool used[maxn] = {0};
// Массив для хранения избыточной информации, прочитанной из входного файла
char s[maxn][100];

// Вспомогательная подфункция функции преобразования географических координат в
```

```
прямоугольные
double foo(double f0){
    double z;
    z = (1 - E2 / 4 - 3 * pow(E2, 2) / 64 - 5 * pow(E2, 3) / 256)* f0;
    z = z - (3 * E2 / 8 + 3 * pow(E2, 2) / 32 + 45 * pow(E2, 3) / 1024)* sin(2 *
f0);
    z = z + (15 * pow(E2, 2) / 256 + 45 * pow(E2, 3) / 1024) * sin(4 * f0);
    z = z - (35 * pow(E2, 3) / 3072) * sin(6 * f0);
    return z;
};

// Функция преобразования географических координат в прямоугольные
void geographicalToRectangular(double lat, double lon,double& x,double& y){
    double lon0,z,v1,a,T,C,M0,M,zona;

    lon0 = (zona * 6 - 3)*pi/180;
    lat = lat*pi/180;
    lon = lon*pi/180;
    z = sin(lat);
    z = pow(z, 2);
    z = 1 - E2 * z;
    v1 = A / sqrt(z);
    a = (lon - lon0) * cos(lat);
    T = pow(tan(lat), 2);
    C = E12 * pow(cos(lat), 2);
    M0 = A * foo(0);
    M = A * foo(lat);
    z = a+(1-T+C)*pow(a,3)/6+(5-18*T+pow(T,2)+72*C-58*E12)*pow(a,5)/120;
    y = FALSEEASTING + SCALEFACTOR * v1 * z + zona * 1000000;
    z = pow(a, 2) / 2 + (5 - T + 9 * C + 4 * pow(C, 2)) * pow(a, 4) / 24;
    z = z + (61 - 58 * T + pow(T, 2) + 600 * C - 330 * E12) * pow(a, 6)/ 720;
    x = FALSENORTHING + SCALEFACTOR * (M - M0 + v1 * tan(lat) * z);
    return;
}

// Функция вычисления расстояния между двумя кластерами по их номерам
double dist(int a, int b){
  return sqrt((data[a].lat-data[b].lat)*(data[a].lat-data[b].lat)+(data[a].lon-
data[b].lon)*(data[a].lon-data[b].lon))+k*fabs(data[a].t-data[b].t);
}

int main ( int argc, char *argv[] ) {

    freopen ( argv[1] , "r" , stdin );
    freopen ( argv[2] , "w" , stdout );

// Чтение исходных данных
    while (!feof(stdin)) {

scanf("%lf%lf%lf%lf%d%d%d%d%lf%s\n",&data[n].lon,&data[n].lat,&data[n].dy,&data[n].
dx,&data[n].h,&data[n].d,&data[n].m,&data[n].year,&data[n].t,s[n]);
      n++;
    }
// Обнуление массива пометок
    for (int i=0; i<n; ++i) used[i]=false;
// Преобразование географических координат в прямоугольные для всех пиклелей. Для
того, чтобы все координаты попали в одну
// зону, зона определяется по первому пикселу
    zona=ceil(data[0].lon / 6);
    for (int i=0; i<n; ++i){
      geographicalToRectangular(data[i].lat,data[i].lon,data[i].x,data[i].y);
    }
```

```
// Добавляем в дерево первую вершину и создаем первый кластер с этой вершиной
    used[0]=true;
    compcount=1;
    comp[0]=1;
// Задаем границы кластера
      compxy[1][1]=data[0].lat+data[0].dx*0.5*180/pi/6371.0;
      compxy[1][2]=data[0].lon-(acos((cos(data[0].dy*0.5/6371.0)-
1)/pow(cos(data[0].lat*pi/180),2)+1))*180/pi;
      compxy[1][3]=data[0].lat-data[0].dx*0.5*180/pi/6371.0;
      compxy[1][4]=data[0].lon+(acos((cos(data[0].dy*0.5/6371.0)-
1)/pow(cos(data[0].lat*pi/180),2)+1))*180/pi;
// Вычисяем расстояния по нее для всех остальных
    for (int i=0; i<n; ++i){
        d[i]=dist(i,0);
        to[i]=0;
    }
    double mindist;
    int mindistnum=1;
// Пока еще есть вершина не в дереве
    while (mindistnum!=-1){
        mindist=1e20;
        mindistnum=-1;
// Ищем вершину с минимальным расстоянием до дерева
        for (int i=0; i<n; ++i){
          if (!used[i]&&(d[i]<mindist)){
            mindist=d[i];
            mindistnum=i;
          }
        }
// Если она нашлась, что добавляем ее в дерево
        if (mindistnum!=-1){
// Для этого помечаем ее
                used[mindistnum]=true;
// Если расстояние от нее до дерева меньше максимально допустимого, то добавляем ее
тот же кластер, что и вершина, до которой
// это минимальное расстояние достигается
            if (mindist<=Emax){
                    compnum=comp[to[mindistnum]];
                comp[mindistnum]=compnum;
// И обновляем границы кластера

      compxy[compnum][1]=max(compxy[compnum][1],data[mindistnum].lat+data[mindistnu
m].dx*0.5*180/pi/6371.0);

      compxy[compnum][2]=min(compxy[compnum][2],data[mindistnum].lon-
(acos((cos(data[mindistnum].dy*0.5/6371.0)-
1)/pow(cos(data[mindistnum].lat*pi/180),2)+1))*180/pi);

      compxy[compnum][3]=min(compxy[compnum][3],data[mindistnum].lat-
data[mindistnum].dx*0.5*180/pi/6371.0);

      compxy[compnum][4]=max(compxy[compnum][4],data[mindistnum].lon+(acos((cos(dat
a[mindistnum].dy*0.5/6371.0)-
1)/pow(cos(data[mindistnum].lat*pi/180),2)+1))*180/pi);
// Иначе создаем новый кластер из одной вершины
            } else {
                compnum=++compcount;
                comp[mindistnum]=compnum;
// И задаем его границы

      compxy[compnum][1]=data[mindistnum].lat+data[mindistnum].dx*0.5*180/pi/6371.0
;
```

```
                        compxy[compnum][2]=data[mindistnum].lon-
(acos((cos(data[mindistnum].dy*0.5/6371.0)-
1)/pow(cos(data[mindistnum].lat*pi/180),2)+1))*180/pi;
                        compxy[compnum][3]=data[mindistnum].lat-
data[mindistnum].dx*0.5*180/pi/6371.0;

      compxy[compnum][4]=data[mindistnum].lon+(acos((cos(data[mindistnum].dy*0.5/63
71.0)-1)/pow(cos(data[mindistnum].lat*pi/180),2)+1))*180/pi;
            }
// Для всех непомеченных вершин обновляем расстояния до дерева, если расстояние до
вновь добавленной вершины меньше
                  for (int i=0; i<n; ++i){
                        double t=dist(mindistnum,i);
                        if (!used[i]&&(t<d[i])){
                              d[i]=t;
                              to[i]=mindistnum;
                        }
                  }
        }
    }

// Выводим пикселы в формате, аналогичном исходному с добавлением номера кластера
    for (int i=0; i<n; ++i){
        printf("%lf %lf %lf %lf %d %d %d %d %lf %s
%d\n",data[i].lon,data[i].lat,data[i].dy,data[i].dx,data[i].h,data[i].d,data[i].m,d
ata[i].year,data[i].t,s[i],comp[i]);
    }
    fclose(stdout);
    freopen ( argv[3] , "w" , stdout );
    for (int i=1; i<=compcount; ++i){
// Вычисляем размеры кластера в километрах
        double dlon =
6371.0*acos(1+pow(cos((compxy[i][1]+compxy[i][3])*0.5*pi/180),2)*(cos((compxy[i][4]
-compxy[i][2])*pi/180)-1));
        double dlat = 6371.0*(compxy[i][1]-compxy[i][3])*pi/180;
// Выводим информацию по кластерам в формате, аналогичном входным данным
      printf("%lf %lf %lf %lf 0 0 0 0 0 0
%d\n",0.5*(compxy[i][2]+compxy[i][4]),0.5*(compxy[i][1]+compxy[i][3]),dlon,dlat,i);
    }

      return 0;
}
```