多波次导弹火力打击任务研究

季青梅* 辛文芳
JI Qing-mei XIN Wen-fang

摘 要 本文根据当前导弹部队的任务特点，研究分析了导弹火力打击任务分配的问题。本文采用 Dijkstra 算法计算了任意两节点的距离，提出了优化模型，确定出最优的发射点安排，并合理运用规划模型为发射点安排打击目标。本文将导弹多波次发射的任务要求转化为多阶段决策过程的优化过程，发挥诸火力单位的多波次整体协调优势，寻求对敌方最大的打击效果，并尽可能减少己方暴露时间，提高打击目标毁伤效果，给出合理可行的解决方案。

关键词 Dijkstra 算法；0-1 规划；路径规划；多阶段决策优化

doi：10.3969/j.issn.1672-9528.2017.01-02.035

1 引言

高技术条件下的现代化战争，突发性急骤增强，对导弹部队的机动能力提出了更高的要求。机动路线制定的好坏直接决定着导弹暴露时间长短。要实现机动快、暴露时间短，就必须要有合理的机动方案，其中机动路线如何选择是机动作战决策的一个重要课题。在一定的作战意图下根据给定的打击目标和现有的武器条件，将可用的弹型、弹量、火力发射单位最优配置到各个目标上去，确定打击各目标所使用的弹型和数量，确定从哪个火力发射单位进行打击，以追求更快和最优的打击效果。

本文假设某部现有 12 套车载发射装置，平均部署在 2 个待机地域，可携带甲、乙、丙三种类型导弹，分别可以对 A、B、C 三个目标进行打击。其所属作战区域内有 30 个发射点位，5 个转载地域，38 个道路节点。在此假设基础上，本文分别研究了如何进行 1 个波次、3 个波次对 3 个目标各 4 发弹的火力打击任务，并构建 2 波次的一般性的任务分配模型。

2 问题分析

2.1 导弹一波次打击任务分配与机动方案

1 个波次对 3 个目标各 4 发弹的火力打击任务的问题中，因从待机区域出发的车已经配备导弹，无需去转载地域进行装弹，所以，安排待机区域的距离最近的发射点发射导弹即是本问题的最优解。此外，因为要求同一波次的导弹齐射，为了减少整体暴露时间，安排较短路径上的装载车较晚发射，从而避免其在发射区等待。

由上小节中的分析可以得出，最小暴露时间可以用如下数学模型来表示：

$$t_{min} = min \sum_{j=1}^{K_D} \sum_{i=1}^{N_{D_j}} d\left(D_j, F_i\right) / v$$

其中：K_D 待机区域的总数，N_{Di} 为计划在从第 i 个待机区域出发的装载车的数量，分别为第 j 个等待区域和第 1 个发射点的坐标，$d\left(D_j, F_i\right)$ 表示 D_j, F_i 两点之间的最小行驶距离（并非空间上的直线距离）。

$d\left(D_j, F_i\right)$ 可以采用 Dijkstra 算法进行求解。Dijkstra 算法是图论中求解距离最为经典的一种解法。针对本问题具体的做法是将所有待机区域、转载地域、道路节点和发射点作为节点构造网络图，然后将任何相互连接的节点的距离记为其欧式距离，不相邻的节点的距离记为无穷大，使用邻接矩阵表示该网络图，应用 Dijkstra 算法即可求解任何两点的距离。另外值得注意的是，在上式中求解过程中需要避免某个发射点 F_i 被采用了多次。

假设需要花费最多时间的装载车需要 t_{max} 的时间可以到达指定发射节点，而其他装载车可以晚一些再从待机地域出发，其最大延迟的时间为 $t_{max} - t_j$，t_j 为其到达指定发射点所需的时间。那么任意一辆装载车经过某节点 J_i 的时间为：

$$t_{J_i} = d(D_j, J_i)/\mathrm{v} + (t_{max} - t_j)$$

当确定好发射点之后，对发射点进行打击目标的安排。考虑到任何有存在相交的发射安排，所有发射点到对应的目标点之间的距离和肯定不是最小的。理由如下：如图所示，假设除了 A 其他的发射路线的地面投影都不存在交点，那么交换当前 B 和 C 的打击目标图中就不存在任何交点，同时很

* 武警特警学院 北京 102211

明显所有发射点到目标点总的距离将会变小（三角形两边之和大于第三边）。由此可见，当图中所有发射点到对应的目标点之间的距离和最小时一定可以得到一个可行的目标安排打击。

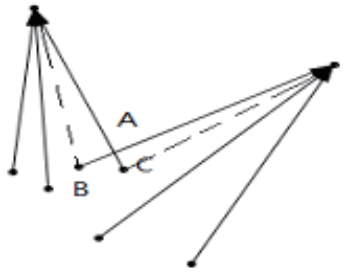

图 1 距离目标点最近的可行安排示意图

可以利用这个性质来构建一个通用的数学模型来求解可行的目标打击安排。将确定好的发射节点集为$\{f_i\}$，目标集为$\{t_i\}$，可以构建一个从发射节点集为$\{f_i\}$到目标集为$\{t_i\}$的有向图，图中的节点为全部发射节点集和目标集中的点，边为所有发射节点f_i到所有目标t_j构成一条边d_{ij}，边的长度为两点间的欧式距离。

若发射节点f_i发射导弹到目标t_j，则记$f_{ij}=1$，否则记$f_{ij}=0$，$R(t_j)$表示t_j安排的导弹数，那么一个通用的可行的目标打击安排可由如下 0-1 规划模型表示：

$$min\sum_j\sum_i f_{ij}\,|\,d_{ij}\,|$$

$$s.t.\begin{cases}\sum_j f_{ij}=1\\ t_{ji}=f_{ij}\\ \sum_i t_{ji}=R(t_j)\\ f_{ij}(f_{ij}-1)=0\end{cases}$$

本文采用 Dijkstra 算法计算任意两点的距离，并通过 C++ 和 MATLAB 实现了求解这个规划问题。为使得总体上导弹到目标的时间达到了最小化，选择发射任务的合理分配方案。

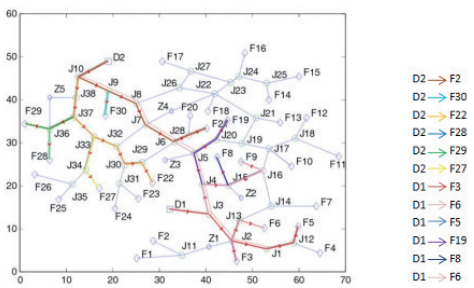

图 2 一波次发射任务的最优分配和机动方案

2.2 3 波次火力打击任务分配和机动方案

本问题是一个多波次的打击任务，我们针对多波次任务，提出分阶段的问题解决方案。这种分阶段的解决方案可以将整个任务按照波次分开。下面针对 3 个阶段进行讨论。

（1）转载地域选择：地域选择是为了下波次的打击任务做提前规划，现给出两种选择策略，只需要对两种策略进行具体建模即可。选择两种策略之下暴露时间最小的作为问题的解答结果。

策略 1 要求转载地域的导弹尽量一次用完，也就是说有多少辆装载车来该转载地域进行装弹已经确定。将等待时间加上装载车在路上所需要的时间进行求和作为预估的消耗时间，再寻找预估的消耗时间最小的方案。

策略 2 优先考虑当前距离到转载地域最近的 k 个发射点，这里的距离可以包含等待时间的造成影响，甚至可以考虑加入对下一次波次的影响。

（2）发射节点选择：发射节点的随转载地域的确定而确定。

（3）打击目标安排：打击目标的安排在问题 2.1 的解决中已经给出了具体的数学模型。在每波次打击任务的安排中该模型依然适用。

按波次迭代求解发射点、转载地域、打击目标安排，可以得到整体每波次的安排。再根据整体的任务安排通过模拟求解时间，有了具体时间后，就得到整个任务的详解安排。但是，由于图中的道路大多都是单向行驶的，转载地域还需要考虑等待时间，使得具体的模拟过程如何进行变成了一个十分棘手的问题。可以考虑从以下几个角度入手简化模拟的过程：

1）由于发射点位置不同，每辆车行驶的时间是也不同。可以考虑如何根据时间不同尽可能的给出一个最佳安排。

2）道路上可能会出现冲突，可以考虑如何尽量降低冲突时间。

根据分析，拆分成两个大的方面：

（1）整体任务规划：求整体任务每波次的发射点安排、转载地域安排，打击目标安排。

（2）机动调度规划：求解每辆装载车具体的路线规划、时间安排、每波次发射时间安排及整体暴露时间等。

整体任务规划：

针对策略 1，首先考虑下为了一波次的任务可以顺利进行至少需要的转载地域的个数N_d，其等于下一波任务需要发射的导弹总数k_j与转载地域最大的导弹容量数M_d的商（向上保留整数）。

$$N_d=ceil(\frac{k_j}{M_d})$$

计算N_d后，下一步找出最佳的N_d个转载地域。计算当前位置到所有发射点的距离和所有发射点到所有转载地域的

距离，对连接某发射点的上述两段距离求和，找出对于某个转载地域来说最优的发射点分配。优化模型如下：

$$\min\max\left\{\max\left\{\left(d\left(C_k-F_k\right)+d\left(F_k-D_i\right)+M_d\left(\overline{L}_i+vt_{wait}\right)/dgree\left(D_i\right)\right)\right\}\right\}$$

其中 $\overline{L}_i$ 表示与转载区 i 相连的节点的平均距离，v 是行驶速度，t_{wait} 表示装载导弹的等待时间，$dgree$ 表示转载地域的度。

上式的意义为找出驶向某转载地域 D_i 的装载车中走的最长距离，假设称其为关键距离。然后，对选中的所有转载的区域关键距离求最大值，这个值决定着本次装弹的暴露时间。求解上式就可以得到一个局部最优解，包括了本波次的发射点安排与下完成本波次打击后的转载地域的选择。这样，就可以完成本波次的打击任务，同时确定了下波次的打击任务的出发点，即模型求解得到的转载地域。那么下一轮打击任务只需要迭代上面的优化过程，又可以得到相应的打击任务与转载地域，一直迭代此过程。通过这种方式就可以得到一个整体规划。

针对策略 2，首先考虑如何去衡量当前距离转载地域的距离。同样这里需要考虑等待时间的影响，另外本文还考虑当前决定对下波次任务的影响，将这个影响用距离来衡量记为 L_{extra}，将由于实际距离和等待产生的影响也用距离来衡量记为 L_{main}。对于第 i 个转载地域，用如下的方式计算 L_{extra} 与 L_{main}：

$$L^i_{main}=d\left(C_k-F_k\right)+d\left(F_k-D_i\right)+M_d\left(\overline{L}_i+vt_{wait}\right)/dgree\left(D_i\right)$$

其中，m_i 为选择 D_i 的装载车的总数。

$$L^i_{extra}=\max\left\{\min-m_i\left(J_{D_i}\right)\right\}+k\overline{L}_i$$

类比策略 1，这里提出的优化方程：

$$\min\max\left\{L^i_{main}+L^i_{extra}\right\}$$

解上述方程，则可以得到局部最优的发射点与转载地域。类比策略 1，迭代直到最后一波次。

机动调度规划：

在问题分析中提出了利用模拟进行求解。其输入为一个整体任务规划，输出为对应的机动调度方案。本文采取的具体做法是将所有的装载车与除待机地域的所有节点视为可以产生碰撞的刚体。每当有车辆经过每个节点时，产生一个碰撞事件，捕捉碰撞事件计算是否会存在潜在的行驶冲突，如果有冲突则选择使某装载车进行避让。

机动调度整体上的执行流程如下：

（1）所有装载车辆指定唯一的编号。根据本波次的任务安排为每个辆装载车选择发射节点。

（2）装载车按照分配的发射节点选择适当的延迟，然后从待机区域出发驶向目标发射点。

（3）每辆装载车经过某个节点时产生碰撞事件，根据此时车辆的位置判断是否本波次的所有车辆都在发射点就位。如果就位，执行发射任务，根据整体规划来选择转载地域与下波次的发射点或者终止任务；如果没有就位，判断是否存在潜在的冲突进行相应的处理。

（4）重复过程（3）直到结束整个任务。由于，要求导弹齐发，所以决定各个装载车的暴露时间的长短主要由装载车从待机区域出发的时间，在进行计算机模拟的时候，安排车辆从出发地域按路径长短分批出发，从而降低潜在的等待时间。

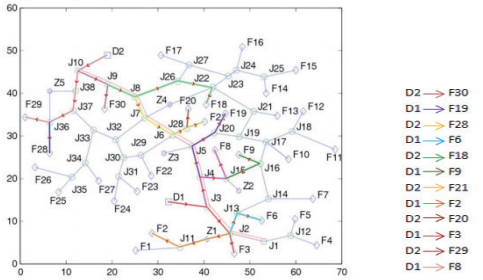

图 3 第一波导弹火力分配示意图

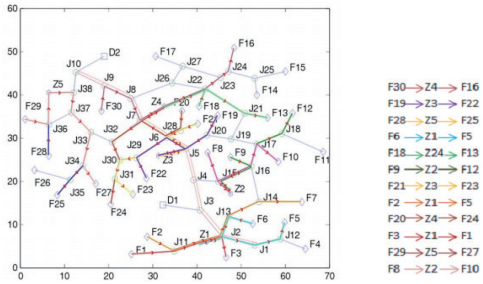

图 4 第二波导弹火力分配示意图

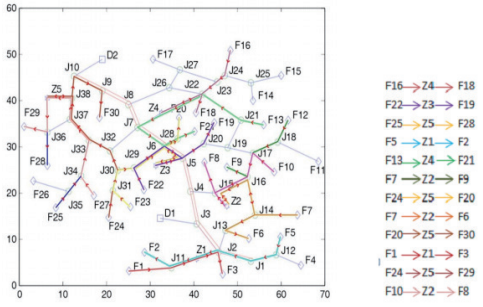

图 5 第三波导弹火力分配示意图

2.3 一般性火力打击查询系统设计

前面给出了一个多波次任务的解空间大小为 $C_n^k(C_{n-k}^k)^{m-1}$，本问 m 为 2 所以解空间为 $C_n^kC_{n-k}^k$，可以看到针对两波次任务而言，当 k,n 较小的时候 $C_n^kC_{n-k}^k$ 的值可以考虑采用暴力求解再适当剪枝的方式去搜索最佳方案。可以对 $C_n^kC_{n-k}^k$ 进行预估，如果 $C_n^kC_{n-k}^k$ 小于某个值的时候可以考虑使用暴力求解的方式。

更具体地，可以采用使用如下的方式进行求解，设定三个参数 $k_1,k_2,t_{critical}$：

（1）若 $C_n^k C_{n-k}^k < k_1$，使用暴力的方式求解全局最优解。

（2）若 $k_1 < C_n^k C_{n-k}^k < k_2$，尝试采用暴力求解。对求解过程计时，若当前时间大于 $t_{critical}$，则终止当前计算，选择上一节提出的模型求解。

（3） 若 $C_n^k C_{n-k}^k > k_2$，采用上一节提出的模型求解。

其中的三个参数 $k_1, k_2, t_{critical}$ 三个参数是根据计算机的计算能力设定的，可以适当调节以满足最适合当前计算机的值。

算法流程图如下：

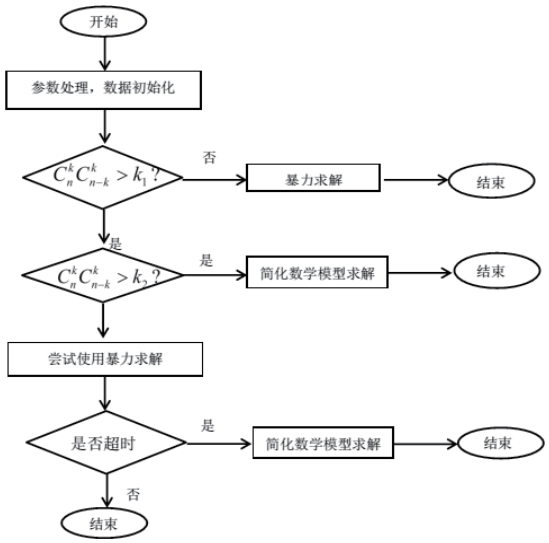

图 6 模型求解流程图

3 结果分析

本文对实际问题进行了分解与简化，模型简单明了，考虑问题比较全面，模型的求解也十分快速，更为重要的是模型的通用性很好，而且针对一个打击任务可以给出完整的解决方案。无退回优先策略、k 最近策略等两种不同的方案，可以针对具体情形采用更为合适的方案，或将两者结合使用。

在涉及多波次的火力打击问题时，模型将整个任务按照波次进行划分，并考虑当前波次对后续波次的影响，使结果更加准确。根据可行性方案的数量及其解空间的大小对问题进行区分，采用不同的求解方案，从而实现高效高质量求解。文章在求解过程中针对导弹齐射问题，为了减少整体暴露时间，安排车辆从出发地域按路径长短分批出发，从而减少了其在发射区等待，使结果相对更优。

当然，由于本文对于实际问题做了适当简化，如在涉及多波次打击任务时，主要考虑了当前波次对下一波次任务分配的影响，这样得出的结果一般不是全局的最优解。文章中主要针对最近距离进行考虑，可能存在车辆等待问题，或者途中会车问题考虑不够全面。针对模型的缺点，我们认为主要是每波次发射点不能重复使用和道路只能单向行驶导致的，使得最后整体暴露时间可能不是最优解，所以本文的改进方向是充分考虑所有车辆出发时间，以及在每波次往返途中会车等待问题。

参考文献

[1] 金宏，余跃，张如飞．常规导弹联合火力打击统一分配模型 [J]．火力与指挥控制．2014

[2] 卓金武．MATLAB 在数学建模中的应用 [M]．北京航空航天大学出版社 2011.4.

[3] 舒长胜，孟庆德．舰炮武器系统应用工程基础 [M]．国防工业出版社．2014

[4] 谭浩强．C 程序语言设计 [M]．清华大学出版 社．2010.6

[5] 申卯兴，曹泽阳，周林．现代军事运筹 [M]．国防工业出版社．2014.7

附录

dijkstra.m 代码：

```
function [L,Z]=dijkstra(W,S,T)
N=length(W(:,1));% 顶点数
W(find(W==0))=inf;
L=Inf*ones(1,N);
L(S)=0;
C=S;
Q=1:N;% 未走访的顶点集
Z=S*ones(1,N);
Z(S)=0
for K=1:N % 更新 L 和 Z 的循环
Q=setdiff(Q,C); [L(Q),ind]=min([L(Q);L(C)+W(C
,Q)]);
Z(Q(find(ind==2)))=C;
if T&C==T
L=L(T); % 最短路径长度;
road=T;% 最短路径终点;
while T~=S% 追溯最短路径上的点
T=Z(T); % 从终点往前寻找其父亲结点
road=[road,T];
end
Z=road(length(road):-1:1); % 颠倒次序;
return;
end;
[null, mC]=min(L(Q));
if null==Inf
% disp(' 到值是 Inf 的点的路不通！ ');
```

```
Z(find(L==Inf))=nan;
return;
else
C=Q(mC);
C;
end
end
end
basic m 代码：
distense=zeros(75,75);
for i = 1:75
fprintf( ‘%g to (1-75)\n’ ,i);
for j = 1:75
distense(i,j)= dijkstra(ddd,j,i);
%fprintf(1,’ %g\n’ ,distense(i));
end
end
C++ 程序：
#include <iostream>
#include <fstream>
#include <cstdi o>
#include<stdlib.h>
#include<string.h>
#define MAX 1000000
using namespace std;
const char * split1 = “,” ;
const char * split2 = “ “;
char f[] =” D2 J10 7.4793,…,” ;
char *set[75]={“D1”,” D2”,// 0-1…};
double arcs[75][75];//邻接矩阵
void init_mubiao()
{for (int i = 0 ;i<75;i++)
{for(int j = 0;j<75;j++)
{arcs[i][j] = MAX;}}}
char pp[82][20];
char *a1[82];
char *a2[82];
char *a3[82];
void fenge()
{int i = 0,j=0;
int mm=0,nn=0;
char * p,* a;
char * s;
p = strtok (f,split1);
while(NULL!=p)
{strcpy(pp[i],p);
p = strtok(NULL,split1);
i++;}
for (i = 0;i<82;i++)
{a = strtok(pp[i],split2);
strcpy(a1[i], a);
a = strtok(NULL,split2);
strcpy(a2[i], a);
a = strtok(NULL,split2);
strcpy(a3[i], a);
a = strtok(NULL,split2);
while(NULL!=a)
{a = strtok(NULL,split2);}}}}
void pipei()
{int numberx[82];int numbery[82];
for (int i =0;i<82;i++)
{for(int j =0;j<75;j++)
if( strcmp(a1[i],set[j])==0)
{numberx[i] = j;
break;}}
for (int i =0;i<82;i++)
{for(int j =0;j<75;j++)
if( strcmp(a2[i],set[j])==0)
{numbery[i] = j;
break;}}
for (int i = 0;i<82;i++)
{arcs[numberx[i]][numbery[i]] = atof(a3[i]);
arcs[numbery[i]][numberx[i]] = atof(a3[i]);}
for(int i=0 ; i<75;i++)
arcs[i][i] = 0;}
int main()
{for ( i = 0 ;i<82;i++)
{a1[i]=new char[5];
a2[i]=new char[5];
a3[i]=new char[10];}
init_mubiao();
fenge();
pipei();
ofstream out(“asd.txt”);
for(int i = 0;i<75;i++)
{for(int j=0;j<75;j++)
```

```
out<<arcs[i][j]<<' \t' ;
out<<endl;}
out.close();
cout<<"矩阵已经保存在工程目录下 asd.txt";
char end;
cin>>end;
return 0;}
```

问题二所对应的仿真模型：

```
#include "stdafx.h"
#include <iostream>
#include <hash_map>
#include <fstream>
#include <stdio.h>
#include<string>
using namespace std;
#define MSIZE 75
#define DSIZE 2
#define FSIZE 30
#define ZSIZE 5
#define JSIZE 38
#define ZBSIZE 5
#define INIBSIZE 6
#define LSIZE 12
#define BACTH 3
#define LMEAN 10
#define VLUNCHER 50
#define TWAIT 0.1666667
#define MFPATH "distances.txt"
#define LFPATH "labels.txt"
#define OFPATH "output.txt"
double **distances;
hash_map<string,int> labelToIndex;
string Dset[2]={"D1","D2"};
string
Fset[30]={"F01",…,"F30"};
bool F_ON_COOL_set[32];
bool F_Selected_set[32];
string Zset[5]={"Z1","Z2","Z3","Z4","Z5"};
int ZDset[5]={2,1,1,2,2};
string
Jset[38]={"J01",…,"J38"};
struct ZNode
{string label;
const int degree;
int cur_select_num;
int left_bomb_num;
ZNode(stringlabel,inttrait,intcur_select_num,intleft_bomb_num):label(label),degree(trait),cur_select_num(cur_select_num),left_bomb_num(left_bomb_num){}
ZNode(int trait):degree(trait){}};
struct Luncher
{string curlabel;
strbool loaded;
Luncher(string curlabel, string targetZ,bool
loade):curlabel(curlabel),targetZ(targetZ),loaded(loaded){}
Luncher(){}};
ZNode * znodes[ZSIZE];
Luncher *lunchers[LSIZE];
hash_map<string,string> nearest_FZPair;
hash_map<string,ZNode*> Zs_map;
void globalInitialization(){
int len=MSIZE;
string mfpath=MFPATH;
string lfpath=LFPATH;
ifstream mf(mfpath);
ifstream lf(lfpath);
string * labels=new string[len];
distances=new double *[len];
for (int i=0;i<len;i++)// 初始化距离矩阵
{distances[i]=new double [len];
for (int j=0;j<len;j++){
mf>>distances[i][j];}
lf>>labels[i];
labelToIndex.insert(pair<string,int>(labels[i],i ));}
for (int i=0;i<ZSIZE;i++)
{znodes[i]=new ZNode(Zset[i],ZDset[i],0,ZBSIZE);
Zs_map.insert(pair <string,ZNode*>(Zset[i], znodes[i]));}
for (int i=0;i<LSIZE/DSIZE;i++)
{for (int j=0;j<DSIZE;j++)
{lunchers[i*DSIZE+j]=new Luncher(Dset[i],"",true); }}
```

```
int index=0;
for (int i=0;i<FSIZE;i++)
{index=0;
for (int j=1;j<ZSIZE;j++)
{if(distances[labelToIndex[Fset[i]]][labelToIndex[Zset[j]]]<distances[labelToIndex[Fset[i]]][labelToIndex[Zset[index]]]){
index =j;}}
nearest_FZPair.insert(pair<string,string>(Fset[i],Zset[index]));}
#ifdef DEBUG
for (int i=0;i<len;i++)
{for (int j=0;j<len;j++){
cout<<distances[i][j]<<" ";}}
for (int i=0;i<len;i++)
{cout<<labels[i];}
for (hash_map<string,int>::iterator
it=labelToIndex.begin();it!=labelToIndex.end();it++)
{cout<<"key: "<<it->first <<" value: "<<it->second<<endl;}
#endif // DEBUG}
void onMission(){
ofstream out(OFPATH,ios::app);
int indexF,indexZ;
double estimatedDis,cur_L;
for (int i=0;i<LSIZE;i++)
{indexF=indexZ=0;
cur_L=0.0;
estimatedDis=INT_MAX;
for (int j=0;j<FSIZE;j++)// 遍历
{if (F_ON_COOL_set[j]||F_Selected_set[j])
{continue;}
for (int k=0;k<ZSIZE;k++)// 遍历
{if (znodes[k]->left_bomb_num<=0)
{continue;}
cur_L=distances[labelToIndex[lunchers[i]->curlabel]][labelToIndex[Fset[j]]]+distances[labelToIndex[Fset[j]]][labelToIndex[Zset[k]]]+znodes[k]->cur_select_num*LMEAN/znodes[k]->degree+znodes[k]->cur_select_num*VLUNCHER*TWAIT;
if(cur_L<estimatedDis)// 比较预估时间
{estimatedDis=cur_L;
indexZ=k;
indexF=j;}}}
out<<Fset[indexF]<<","<<Zset[indexZ]<<endl;
znodes[indexZ]->cur_select_num++;
znodes[indexZ]->left_bomb_num--;
lunchers[i]->curlabel=Zset[indexZ];
F_Selected_set[indexF]=true;}
for (int i=0;i<FSIZE;i++){
F_ON_COOL_set[i]=F_Selected_set[i];
F_Selected_set[i]=false;}
out<<"------------------------"<<endl;
out.close();}
void onLastMission(){
ofstream out(OFPATH,ios::app);
int indexF;
double estimatedDis;
for (int i=0;i<LSIZE;i++)
{indexF=0;
estimatedDis=INT_MAX;
for (int j=0;j<FSIZE;j++)// 遍历所有的发射点
{if (F_ON_COOL_set[j]||F_Selected_set[j])
{continue;}
if(distances[labelToIndex[lunchers[i]->curlabel]][labelToIndex[Fset[j]]]<estimatedDis)
{estimatedDis=distances[labelToIndex[lunchers[i]->curlabel]][labelToIndex[Fset[j]]];
indexF=j;}}
out<<Fset[indexF]<<","<<lunchers[i]->curlabel<<endl;
F_Selected_set[indexF]=true;}
out.close();}
int main()
{globalInitialization();
for (int i=0;i<BACTH-1;i++)
{onMission();}
onLastMission();
system("pause");
return 0;}
```

【作者简介】季青梅，女，武警特警学院数理教研室主任、教授，硕士，研究方向：计算机技术与应用、应用数学。

（收稿日期：2016-12-09）