

附錄 一

附錄表 7.1 查核對象非航空公司之查核工作任務表

	查核工作任務編號	查核工作任務名稱	查核對象
航 務 查 核	JOB FUNCTION 11	機場檢查	機場
	JOB FUNCTION 27	航空器駕駛員訓練機構檢查	駕駛員訓練機構
	附錄 J	模擬機委託檢定檢查	模擬機委託檢定機構

資料來源：交通部民用航空局 航務檢查員手冊，民 91

附錄表 7.2 執行條件為認證及特殊事件之查核工作任務表

	查核工作任務	查核工作任務名稱	查核原因
航 務 查 核	JOB FUNCTION 17	最低裝備需求手冊核准	認證
	JOB FUNCTION 21	航空器租賃契約評估	租賃事件
	JOB FUNCTION 22	第二 / 三 / 類儀降作業檢查	認證
	JOB FUNCTION 23	雙渦輪引擎延展航程作業(包含加速核准流程)	認證
	JOB FUNCTION 24	航空公司申請開闢新航線、現有航線變更機種飛航審核	認證
	JOB FUNCTION 25	執行運渡飛航之持續授權特種許可	認證
	附錄 B	訓練核准	認證
	附錄 C	使用人手冊及程序	認證
	附錄 D	首次機型驗證飛航檢查	認證
	附錄 E	航空器適航試飛檢查	認證
	附錄 H	航空人員檢定給證作業程序	認證
	附錄 I	航空器事故（失事/重大意外）調查處理	失事事件
	附錄 M	飛安事件調查檢查	失事事件
	附錄 N	垂直高度隔離縮減檢查	認證
	附錄 P	全球定位系統（GPS）作業核准	認證
	附錄 Q	航空公司合併作業檢查	合併事件
	附錄 R	導航性能需求（RNP）作業核准	認證
	附錄 S	極地飛航運作檢查	認證

資料來源：交通部民用航空局 航務檢查員手冊，民 91

附錄表 7.3 其他本研究暫不考慮之查核工作任務表

	查核工作 任務	查核工作任務名稱	查核內容
航 務 查 核	附錄 F	正式申請函範本及符合之陳述範本	此章節提供正式申請函與符合之陳述的範本格式。
	附錄 G	檢查員之行為	此章節提供檢查員在執行查核工作時，需注意自身行為的細節。
	附錄 K	飛安檢查業務督導檢查	本章提供檢查員準備簡報資料之指引俾於局長飛安檢查業務督導會議中提報，討論提供飛安改善建議或心得分享每位檢查員，以增進飛安。

資料來源：交通部民用航空局 航務檢查員手冊，民 91

附錄表 7.4 各問題向度對應之數學式編號參照表

問題向度			數學式編號
查 核 工 作 排 程	效率	工作合併	前置作業
		團隊合作	模式 ()：式 (1) \ 式 (2)
	合理	任務順序關係	模式 ()：式 (7)
		頻次及時間安排	模式 ()：式 (6) \ 式 (8)
檢 查 員 指 派	技能分類	檢查員替代性	模式 ()：式 (10) \ 式 (11) \ 式 (12) \ 式 (13) \ 式 (15) \ 式 (17)
	彈性	多人指派與固定指派之平衡點	模式 ()：式 (4)
	公平	薪資公平性	模式 ()：式 (21)
	作業品質	年度複訓	模式 ()：式 (20)
		額外工作負荷	模式 ()：式 (5) \ 式 (22) \ 式 (23)
		工作量均分	模式 ()：式 (10) \ 式 (11) \ 式 (12) \ 式 (13) \ 式 (14)
其 他	人力資源是否符合工作需求		模式 ()：式 (3)
	工作時間不可切割執行		模式 ()：式 (9)
	確保檢查員在同一時間僅執行一項工作		模式 ()：式 (16) \ 式 (19)
	確保各項作業人力足夠		模式 ()：式 (18)
	各目標式權重		模式 ()：式 (24)

附錄 二

```
#include <ilsolver/ilosolverint.h>
#include <fstream>
ILOSTLBEGIN

typedef IloArray<IloIntArray> IntMatrix;
typedef IloArray<IntMatrix> _3DIntArr;
typedef IloArray<IloIntVarArray> _2DIntVarArr;
typedef IloArray<_2DIntVarArr> _3DIntVarArr;
typedef IloArray<_3DIntVarArr> _4DIntVarArr;

int main () {
IloEnv env;

//開一個叫"inspector.txt"的輸出檔
ofstream out_file("inspector.txt");
if(!out_file){
    cerr << "Can't creat the file yor wish to write to!!!" << endl;
    return -1;
}

IloInt a, j, l, n, d, j1,l1,k,m,a1,m1,i;

try {
IloModel model(env);

IloInt nbAirlines = 3;           //航空公司數
IloInt nbJobs = 8;              //工作任務數
IloInt nbSeciJobs = 3;
IloInt nbDays =21;              //每年作業時間天數
IloInt nbHours = 8;             //每日作業時間長度
IloInt nbInspector1 = 3;        //主任檢查員人數
IloInt nbInspector2 = 4;        //一般檢查員人數
IloInt nbSection = 3;
IloInt SectionDays = 7;

IloIntArray AirlineInspector(env, nbAirlines, 3, 2, 2);    //各航空公司檢查員人數
IloIntArray JobInspector1up(env, nbJobs+nbSeciJobs, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1);
                                                                //各工作所需主任檢查員之下限值
IloIntArray nbTrainDays1(env, nbInspector1, 1, 0, 0 );      //各主任檢查員之複訓天數
IloIntArray nbTrainDays2(env, nbInspector2, 1, 0, 1, 0);    //各一般檢查員之複訓天數
IloIntArray TrainstDay1(env, nbInspector1, 3, 0, 0 );       //各主任檢查員之複訓天數
IloIntArray TrainstDay2(env, nbInspector2, 1, 0, 1, 0);     //各一般檢查員之複訓天數

IntMatrix Inspector1P(env, nbInspector1);                  //各主任檢查員工作偏好度
IntMatrix Inspector2P(env, nbInspector2);                  //各一般檢查員工作偏好度
```

```

Inspector1P[0]=IloIntArray(env, nbAirlines, 1, 2, 2);
Inspector1P[1]=IloIntArray(env, nbAirlines, 2, 1, 2);
Inspector1P[2]=IloIntArray(env, nbAirlines, 2, 2, 1);

Inspector2P[0]=IloIntArray(env, nbAirlines, 1, 2, 3);
Inspector2P[1]=IloIntArray(env, nbAirlines, 1, 3, 2);
Inspector2P[2]=IloIntArray(env, nbAirlines, 2, 1, 3);
Inspector2P[3]=IloIntArray(env, nbAirlines, 2, 3, 1);

IntMatrix  frequency(env, nbAirlines);           //工作頻次
IntMatrix  JobHours(env, nbAirlines);             //各工作所需時間
IntMatrix  priority(env, nbJobs+nbSeciJobs);      //工作之順序關係
IntMatrix  JobManpower(env, nbAirlines);          //各工作所需人力
IntMatrix  Place(env, nbAirlines);                //各工作作業地點

frequency[0]=IloIntArray(env, nbJobs+nbSeciJobs, 1, 1, 2, 2, 4, 4, 1, 1, 1, 4, 1);
frequency[1]=IloIntArray(env, nbJobs+nbSeciJobs, 1, 1, 3, 1, 2, 2, 1, 1, 3, 1, 1);
frequency[2]=IloIntArray(env, nbJobs+nbSeciJobs, 1, 1, 3, 1, 2, 2, 1, 1, 3, 3, 1);

JobHours[0]=IloIntArray(env, nbJobs+nbSeciJobs, 8, 5, 2, 5, 4, 4, 4, 8, 5, 4, 4 );
JobHours[1]=IloIntArray(env, nbJobs+nbSeciJobs, 14, 3, 2, 3, 4, 4, 4, 8, 2, 4, 4);
JobHours[2]=IloIntArray(env, nbJobs+nbSeciJobs, 14, 3, 2, 3, 4, 4, 4, 7, 2, 4, 4);

priority[0]=IloIntArray(env, nbJobs+nbSeciJobs, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
priority[1]=IloIntArray(env, nbJobs+nbSeciJobs, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
priority[2]=IloIntArray(env, nbJobs+nbSeciJobs, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
priority[3]=IloIntArray(env, nbJobs+nbSeciJobs, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1);
priority[4]=IloIntArray(env, nbJobs+nbSeciJobs, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
priority[5]=IloIntArray(env, nbJobs+nbSeciJobs, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0);
priority[6]=IloIntArray(env, nbJobs+nbSeciJobs, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1);
priority[7]=IloIntArray(env, nbJobs+nbSeciJobs, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1);
priority[8]=IloIntArray(env, nbJobs+nbSeciJobs, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0);
priority[9]=IloIntArray(env, nbJobs+nbSeciJobs, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0);
priority[10]=IloIntArray(env, nbJobs+nbSeciJobs, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0);

JobManpower[0]=IloIntArray(env, nbJobs+nbSeciJobs, 2, 3, 2, 4, 2, 2, 3, 3, 1, 1, 1);
JobManpower[1]=IloIntArray(env, nbJobs+nbSeciJobs, 2, 3, 2, 4, 2, 2, 3, 3, 1, 1, 1);
JobManpower[2]=IloIntArray(env, nbJobs+nbSeciJobs, 2, 3, 2, 4, 2, 2, 3, 3, 1, 1, 1);

Place[0]=IloIntArray(env, nbJobs+nbSeciJobs, 1, 2, 2, 1, 2, 1, 1, 1, 2, 1, 1);
Place[1]=IloIntArray(env, nbJobs+nbSeciJobs, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);
Place[2]=IloIntArray(env, nbJobs+nbSeciJobs, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1);

IloIntArray AirlineTHours(env, nbAirlines);      //各航空公司檢查總時數
for(a=1; a<=nbAirlines; a++){
    for(j=1; j<=nbJobs+nbSeciJobs; j++){
        AirlineTHours[a-1] += JobHours[a-1][j-1]*JobManpower[a-1][j-1]*frequency[a-1][j-1];
    }
}

```

```

3DIntArr SeciJobst(env,nbAirlines);          //特殊工作起始時間
for(a = 1; a <= nbAirlines; a++)
SeciJobst[a-1] = IntMatrix(env,nbSeciJobs);
SeciJobst[0][0]=IloIntArray(env, frequency[0][nbJobs+0], 57);
SeciJobst[0][1]=IloIntArray(env, frequency[0][nbJobs+1], 45, 61, 105,145);
SeciJobst[0][2]=IloIntArray(env, frequency[0][nbJobs+2], 37);
SeciJobst[1][0]=IloIntArray(env, frequency[1][nbJobs+0], 65, 90, 125 );
SeciJobst[1][1]=IloIntArray(env, frequency[1][nbJobs+1], 49);
SeciJobst[1][2]=IloIntArray(env, frequency[1][nbJobs+2], 35);
SeciJobst[2][0]=IloIntArray(env, frequency[2][nbJobs+0], 61, 93, 133 );
SeciJobst[2][1]=IloIntArray(env, frequency[2][nbJobs+1], 42, 82, 153 );
SeciJobst[2][2]=IloIntArray(env, frequency[2][nbJobs+2], 33);

//build model
2DIntVarArr  SectionHours(env, nbAirlines);          //各階段工作總時
for(a = 1; a <= nbAirlines; a++)
SectionHours[a-1] =IloIntVarArray(env,nbSection, 0 , AirlineTHours[a-1]);

3DIntVarArr Jobst(env,nbAirlines);          //各工作起始時間變數
for(a = 1; a <= nbAirlines; a++)
Jobst[a-1] = _2DIntVarArr(env,nbJobs+nbSeciJobs);
for(a = 1; a <= nbAirlines; a++){
    for(j = 1; j <= nbJobs+nbSeciJobs; j++){
        if(frequency[a-1][j-1]>=1)
            Jobst[a-1][j-1]=IloIntVarArray(env,frequency[a-1][j-1],1,nbDays*nbHours-JobHours[a-1][j-1]+1);
    }
}

_4DIntVarArr jobman1(env, nbAirlines);          //執行各工作與檢查員狀態變數
for(a=1; a<=nbAirlines; a++)
jobman1[a-1] = _3DIntVarArr(env,nbJobs+nbSeciJobs);
for(a=1; a<=nbAirlines; a++){
    for(j=1; j<=nbJobs+nbSeciJobs; j++){
        if(frequency[a-1][j-1]>=1)
            jobman1[a-1][j-1] = _2DIntVarArr(env,frequency[a-1][j-1]);
    }
}
for(a=1; a<=nbAirlines; a++){
    for(j=1; j<=nbJobs+nbSeciJobs; j++){
        if(frequency[a-1][j-1]>=1){
            for(l=1; l<=frequency[a-1][j-1]; l++)
                jobman1[a-1][j-1][l-1] = IloIntVarArray(env,nbInspector1, 0, 1);
        }
    }
}

_4DIntVarArr jobman2(env, nbAirlines);          //執行各工作與各一般檢查員狀態變數
for(a=1; a<=nbAirlines; a++)
jobman2[a-1] = _3DIntVarArr(env,nbJobs+nbSeciJobs);
for(a=1; a<=nbAirlines; a++){

```

```

        for(j=1; j<=nbJobs+nbSeciJobs; j++){
            if(frequency[a-1][j-1]>=1)
                jobman2[a-1][j-1] = _2DIntVarArr(env,frequency[a-1][j-1]);
        }
    }
    for(a=1; a<=nbAirlines; a++){
        for(j=1; j<=nbJobs+nbSeciJobs; j++){
            if(frequency[a-1][j-1]>=1){
                for(l=1;l<=frequency[a-1][j-1];l++){
                    jobman2[a-1][j-1][l-1] = IloIntArray(env,nbInspector2, 0, 1);
                }
            }
        }
    }

    //construct constraints
    for(a=1; a<=nbAirlines; a++){
        for(j=nbJobs+1; j<=nbJobs+nbSeciJobs; j++){
            for(l=1; l<=frequency[a-1][j-1]; l++){
                model.add(Jobst[a-1][j-1][l-1]==SeciJobst[a-1][j-1-nbJobs][l-1]);
            }
        }
    }

    // 單一工作開始時間限制
    for(a=1; a<=nbAirlines; a++){
        for(j=1; j<=nbJobs; j++){
            for(m=1; m<=3; m++){
                if(JobHours[a-1][j-1]>8*(m-1)+1){
                    if(JobHours[a-1][j-1]<=8*m){
                        if(frequency[a-1][j-1]>=1){
                            for(l=1; l<=frequency[a-1][j-1]; l++){
                                for(d=1; d<=nbDays ; d++) {
                                    for(k=1; k<JobHours[a-1][j-1]-8*(m-1) ; k++)
                                        model.add(Jobst[a-1][j-1][l-1]!=nbHours*(d-1)+nbHours+1-k);
                                }
                            }
                        }
                    }
                }
            }
        }
    }

    //不同工作時間優先關係
    for(a=1; a<=nbAirlines; a++){
        for(j=1; j<=nbJobs+nbSeciJobs; j++){
            for(n=1; n <= nbJobs+nbSeciJobs; n++) {
                if(priority[j-1][n-1]==1){
                    if(frequency[a-1][j-1]>=1){
                        if(frequency[a-1][n-1]>=1)
                            model.add( Jobst[a-1][j-1][0]+JobHours[a-1][j-1]-1 < Jobst[a-1][n-1][0] );
                    }
                }
            }
        }
    }

```

```

    }
    }
}
}
//單一工作時間先後關係
for(a=1; a<=nbAirlines; a++){
    for(j=1; j<=nbJobs; j++){
        if(frequency[a-1][j-1]>1){
            for( l=1; l<frequency[a-1][j-1]; l++)
                model.add( Jobst[a-1][j-1][l]-Jobst[a-1][j-1][l-1]>=0.5*(nbHours*nbDays/frequency[a-1][j-1]) );
        }
    }
}

```

//各航空公司作業均分與主任檢查員

```

for(a=1; a<=nbAirlines; a++){
    for(m=1; m<=nbSection; m++){
        IloExpr zeee(env), zeee2(env);
        try{
            for(j=1; j<=nbJobs+nbSeciJobs; j++){
                if(frequency[a-1][j-1]>=1){
                    for( l=1; l<=frequency[a-1][j-1]; l++){
                        zeee += ( 1+SectionDays*(m-1)*nbHours <= Jobst[a-1][j-1][l-1] && Jobst[a-1][j-1][l-1] <=
                            SectionDays*m*nbHours ) * JobManpower[a-1][j-1] * JobHours[a-1][j-1];
                        zeee2 += ( 1+SectionDays*(m-1)*nbHours <= Jobst[a-1][j-1][l-1] && Jobst[a-1][j-1][l-1] <=
                            SectionDays*m*nbHours ) * JobInspector1up[j-1] * JobHours[a-1][j-1];
                    }
                }
            }
            model.add( zeee >= 0.85*AirlineTHours[a-1]/nbSection);
            model.add( zeee <= 1.2*AirlineTHours[a-1]/nbSection);
            model.add( zeee == SectionHours[a-1][m-1]);
            model.add( zeee2 <= 0.5*nbDays*nbHours/nbSection);
        }
        catch(...){
            zeee.end();
            zeee2.end();
            throw;
        }
        zeee.end();
        zeee2.end();
    }
}

```

//單一航空公司同時刻作業人數上限

```

for(a=1; a<=nbAirlines; a++){
    for(j=1; j<=nbJobs+nbSeciJobs; j++){
        if(JobManpower[a-1][j-1]>AirlineInspector[a-1]){
            if(frequency[a-1][j-1]>=1){

```

```

for( l=1; l<=frequency[a-1][j-1]; l++){
  lloExpr z1(env),z2(env);
  try{
    for(j1=1; j1<=nbJobs+nbSeciJobs; j1++){
      if(j!=j1){
        if(frequency[a-1][j1-1]>=1){
          for( l1=1; l1<=frequency[a-1][j1-1]; l1++){
            z1 += ( Jobst[a-1][j1-1][l1-1] >= Jobst[a-1][j-1][l-1] && Jobst[a-1][j1-1][l1-1] <=
              Jobst[a-1][j-1][l-1]+JobHours[a-1][j-1]-1);
            z2 += ( Jobst[a-1][j1-1][l1-1]+JobHours[a-1][j1-1]-1 >= Jobst[a-1][j-1][l-1] &&
              Jobst[a-1][j1-1][l1-1]+JobHours[a-1][j1-1]-1 <=
              Jobst[a-1][j-1][l-1]+JobHours[a-1][j-1]-1);
          }
        }
      }
    }
    model.add(z1 == 0);
    model.add(z2 == 0);
  }
  catch(...){
    z1.end();
    z2.end();
    throw;
  }
  z1.end();
  z2.end();
}
}
}
}
for(d=1; d<=nbDays ; d++) {
  for(k=1; k<=nbHours ; k++) {
    lloExpr z3(env),z4(env);
    try{
      for(j=1; j<=nbJobs+nbSeciJobs; j++){
        if(JobManpower[a-1][j-1]<=AirlineInspector[a-1]){
          if(frequency[a-1][j-1]>=1){
            for( l=1; l<=frequency[a-1][j-1]; l++){
              z3 +=(Jobst[a-1][j-1][l-1]<= nbHours*(d-1)+k-1 && Jobst[a-1][j-1][l-1]+JobHours[a-1][j-1]-1
                >= nbHours*(d-1)+k-1 )*JobManpower[a-1][j-1];
              z4 +=(Jobst[a-1][j-1][l-1]<= nbHours*(d-1)+k-1 && Jobst[a-1][j-1][l-1]+JobHours[a-1][j-1]-1
                >= nbHours*(d-1)+k-1 )*JobInspector1up[j-1];
            }
          }
        }
      }
    }
    model.add( z3 <= AirlineInspector[a-1] );
    model.add( z4 <= 1 );
  }
  catch(...){

```



```

        z3.end();
        z4.end();
        throw;
    }
    z3.end();
    z4.end();
}
}

//主任檢查員要求
for(a=1; a<=nbAirlines; a++){
    for(j=1; j<=nbJobs+nbSeciJobs; j++){
        if(JobInspector1up[j-1] >=1){
            if(frequency[a-1][j-1]>=1){
                for( l=1; l<=frequency[a-1][j-1]; l++)
                    model.add(jobman1[a-1][j-1][l-1][a-1]==1);
            }
        }
    }
}

//單一時間作業人數上限
for(d=1; d<=nbDays ; d++) {
    for(k=1; k<=nbHours ; k++) {
        IloExpr z5(env);
        try{
            for(a=1; a<=nbAirlines; a++){
                for(j=1; j<=nbJobs+nbSeciJobs; j++){
                    if(frequency[a-1][j-1]>=1){
                        for( l=1; l<=frequency[a-1][j-1]; l++)
                            z5 += (Jobst[a-1][j-1][l-1]<=nbHours*(d-1)+k-1 && Jobst[a-1][j-1][l-1]+JobHours[a-1][j-1]-1
                                >= nbHours*(d-1)+k-1 ) * JobManpower[a-1][j-1];
                    }
                }
            }
            model.add(z5 <= nbInspector1+nbInspector2);
        }
        catch(...){
            z5.end();
            throw;
        }
        z5.end();
    }
}

//工作人數限制
for(a=1; a<=nbAirlines; a++){
    for(j=1; j<=nbJobs+nbSeciJobs; j++){
        if(frequency[a-1][j-1]>=1){

```

```

for( l=1; l<=frequency[a-1][j-1]; l++){
  IloExpr z6(env),z7(env);
  try{
    for(i=1;i<=nbInspector1;i++){
      z6 +=jobman1[a-1][j-1][l-1][i-1];
      model.add( z6 >= JobInspector1up[j-1] ); //執行該工作之主任檢查員人數需大等於該工作所
                                              需主任檢查員人數

      for(i=1;i<=nbInspector2;i++){
        z7 +=jobman2[a-1][j-1][l-1][i-1];
        model.add( z6+z7 == JobManpower[a-1][j-1] ); //執行該工作之主任檢查員加一般檢查員人數
                                                    需等於該工作所需人數
      }
    }
  } catch(...){
    z6.end();
    z7.end();
    throw;
  }
  z6.end();
  z7.end();
}
}
}

//複訓時間不可接受工作
for(a=1; a<=nbAirlines; a++){
  for(j=1; j<=nbJobs+nbSeciJobs; j++){
    if(frequency[a-1][j-1]>=1){
      for( l=1; l<=frequency[a-1][j-1]; l++){
        for(k = 1; k <= nbHours ; k++) {
          for(i=1; i<= nbInspector1; i++){
            if(nbTrainDays1[i-1]>0){
              for(d = TrainstDay1[i-1]; d <=TrainstDay1[i-1]+nbTrainDays1[i-1]-1 ; d++)
                model.add( jobman1[a-1][j-1][l-1][i-1]*(Jobst[a-1][j-1][l-1] <= nbHours*(d-1)+k-1 &&
                    Jobst[a-1][j-1][l-1]+JobHours[a-1][j-1]-1 >= nbHours*(d-1)+k-1 )==0);
            }
          }
        }
        for(i=1; i<= nbInspector2; i++){
          if(nbTrainDays2[i-1]>0){
            for(d = TrainstDay2[i-1]; d <=TrainstDay2[i-1]+nbTrainDays2[i-1]-1 ; d++)
              model.add( jobman2[a-1][j-1][l-1][i-1]*(Jobst[a-1][j-1][l-1] <= nbHours*(d-1)+k-1
                  &&Jobst[a-1][j-1][l-1]+JobHours[a-1][j-1]- >= nbHours*(d-1)+k-1 )==0);
            }
          }
        }
      }
    }
  }
}
}

```

//時間人員限制

```

for(d = 1; d <= nbDays ; d++) {
    for(k = 1; k <= nbHours ; k++) {
        for(i = 1; i <= nbInspector1; i++){
            IloExpr z8(env);
            try{
                for(a=1; a<=nbAirlines; a++){
                    for(j=1; j<=nbJobs+nbSeciJobs; j++){
                        if(frequency[a-1][j-1]>=1){
                            for( l=1; l<=frequency[a-1][j-1]; l++)
                                z8 += jobman1[a-1][j-1][l-1][i-1]*(Jobst[a-1][j-1][l-1] <= nbHours*(d-1)+k-1  &&
                                    Jobst[a-1][j-1][l-1]+JobHours[a-1][j-1]-1 >= nbHours*(d-1)+k-1 );
                        }
                    }
                }
                model.add( z8 <= 1 );
            }
            catch(...){
                z8.end();
                throw;
            }
            z8.end();
        }
        for(i = 1; i <= nbInspector2; i++){
            IloExpr z9(env);
            try{
                for(a=1; a<=nbAirlines; a++){
                    for(j=1; j<=nbJobs+nbSeciJobs; j++){
                        if(frequency[a-1][j-1]>=1){
                            for( l=1; l<=frequency[a-1][j-1]; l++)
                                z9 += jobman2[a-1][j-1][l-1][i-1]*(Jobst[a-1][j-1][l-1] <= nbHours*(d-1)+k-1  &&
                                    Jobst[a-1][j-1][l-1]+JobHours[a-1][j-1]-1 >= nbHours*(d-1)+k-1 );
                        }
                    }
                }
                model.add( z9 <= 1 );
            }
            catch(...){
                z9.end();
                throw;
            }
            z9.end();
        }
    }
}

```

//單一檢查員工作時數限制

```

for(i=1; i<=nbInspector1; i++){
    IloExpr z10(env);
    try{

```

```

for(a=1; a<=nbAirlines; a++){
  for(j=1; j<=nbJobs+nbSeciJobs; j++){
    if(frequency[a-1][j-1]>=1){
      for( l=1; l<=frequency[a-1][j-1]; l++)
        z10 += jobman1[a-1][j-1][l-1][i-1]*JobHours[a-1][j-1];
    }
  }
}
model.add( z10 >=0.8*IloSum(AirlineTHours)/(nbInspector1+nbInspector2));
model.add( z10 <=1.2*IloSum(AirlineTHours)/(nbInspector1+nbInspector2));
}
catch(...){
  z10.end();
  throw;
}
z10.end();
}

```

```

for(i=1; i<=nbInspector2; i++){
IloExpr z11(env);
try{
  for(a=1; a<=nbAirlines; a++){
    for(j=1; j<=nbJobs+nbSeciJobs; j++){
      if(frequency[a-1][j-1]>=1){
        for( l=1; l<=frequency[a-1][j-1]; l++)
          z11 += jobman2[a-1][j-1][l-1][i-1]*JobHours[a-1][j-1];
      }
    }
  }
  model.add( z11 >=0.8*IloSum(AirlineTHours)/(nbInspector1+nbInspector2));
  model.add( z11 <=1.2*IloSum(AirlineTHours)/(nbInspector1+nbInspector2));
}
catch(...){
  z11.end();
  throw;
}
z11.end();
}

```

//工作地點限制

```

for(d = 1; d <= nbDays ; d++) {
  for(i=1; i<=nbInspector1; i++){
    IloExpr z12(env),z13(env);
    try{
      for(a=1; a<=nbAirlines; a++){
        for(j=1; j<=nbJobs+nbSeciJobs; j++){
          if(Place[a-1][j-1]==1){
            if(frequency[a-1][j-1]>=1){
              for( l=1; l<=frequency[a-1][j-1]; l++)
                z12 += (Jobst[a-1][j-1][l-1]+JobHours[a-1][j-1]-1>nbHours*(d-1) && Jobst[a-1][j-1][l-1]

```

```

        <= nbHours*d ) * jobman1[a-1][j-1][l-1][i-1];
    }
}
}
}
for(a=1; a<=nbAirlines; a++){
    for(j=1; j<=nbJobs+nbSeciJobs; j++){
        if(Place[a-1][j-1]==2){
            if(frequency[a-1][j-1]>=1){
                for( l=1; l<=frequency[a-1][j-1]; l++){
                    z13 += (Jobst[a-1][j-1][l-1]+JobHours[a-1][j-1]-1>nbHours*(d-1) && Jobst[a-1][j-1][l-1]
                        <= nbHours*d ) * jobman1[a-1][j-1][l-1][i-1];
                }
            }
        }
    }
}
model.add((z12>=1)+(z13>=1)<=1);
}
catch(...){
    z12.end();
    z13.end();
    throw;
}
z12.end();
z13.end();
}
for(i=1; i<=nbInspector2; i++){
    IloExpr z14(env),z15(env);
    try{
        for(a=1; a<=nbAirlines; a++){
            for(j=1; j<=nbJobs+nbSeciJobs; j++){
                if(Place[a-1][j-1]==1){
                    if(frequency[a-1][j-1]>=1){
                        for( l=1; l<=frequency[a-1][j-1]; l++){
                            z14 += (Jobst[a-1][j-1][l-1]+JobHours[a-1][j-1]-1>nbHours*(d-1) && Jobst[a-1][j-1][l-1]
                                <=nbHours*d ) * jobman2[a-1][j-1][l-1][i-1];
                        }
                    }
                }
            }
        }
    }
    for(a=1; a<=nbAirlines; a++){
        for(j=1; j<=nbJobs+nbSeciJobs; j++){
            if(Place[a-1][j-1]==2){
                if(frequency[a-1][j-1]>=1){
                    for( l=1; l<=frequency[a-1][j-1]; l++){
                        z15 += (Jobst[a-1][j-1][l-1]+JobHours[a-1][j-1]-1>nbHours*(d-1) && Jobst[a-1][j-1][l-1]
                            <=nbHours*d ) * jobman2[a-1][j-1][l-1][i-1];
                    }
                }
            }
        }
    }
}

```

```

    }
    model.add((z14>=1)+(z15>=1)<=1);
}
catch(...){
    z14.end();
    z15.end();
    throw;
}
z14.end();
z15.end();
}
}
}

```

//目標式 1

```
IloIntVar tatalObj(env,0, 250);
```

```
IloExpr Obj1a(env),Obj1b(env);
```

```

try{
    for(a=1; a<=nbAirlines; a++){
        for(j=1; j<=nbJobs+nbSeciJobs; j++){
            if(frequency[a-1][j-1]>=1){
                for( l=1; l<=frequency[a-1][j-1]; l++){
                    for(i=1; i<=nbInspector1; i++){
                        Obj1a += jobman1[a-1][j-1][l-1][i-1]*Inspector1P[i-1][a-1];
                    }
                    for(i=1; i<=nbInspector2; i++){
                        Obj1b += jobman2[a-1][j-1][l-1][i-1]*Inspector2P[i-1][a-1];
                    }
                }
            }
        }
    }
    model.add(tatalObj == Obj1a+Obj1b);
}
catch(...){
    Obj1a.end();
    Obj1b.end();
    throw;
}
Obj1a.end();
Obj1b.end();

```

//目標式 2

```
IloIntVar tatalObj2(env,0, 147);
```

```
IloExpr Obj2c(env),Obj2d(env);
```

```

try{
    for(d = 1; d <= nbDays ; d++) {
        for(i=1; i<=nbInspector1; i++){
            IloExpr Obj2a(env);
            try{
                for(a=1; a<=nbAirlines; a++){

```

```

        for(j=1; j<=nbJobs+nbSeciJobs; j++){
            if(frequency[a-1][j-1]>=1){
                for( l=1; l<=frequency[a-1][j-1]; l++)
                    Obj2a += jobman1[a-1][j-1][l-1]*(Jobst[a-1][j-1][l-1] <= nbHours*d &&
                        Jobst[a-1][j-1][l-1] + JobHours[a-1][j-1]-1 > nbHours*(d-1) );
            }
        }
    }
    Obj2c += (Obj2a >=1);
}
catch(...){
    Obj2a.end();
    throw;
}
Obj2a.end();
}
for(i=1; i<=nbInspector2; i++){
    IloExpr Obj2b(env);
    try{
        for(a=1; a<=nbAirlines; a++){
            for(j=1; j<=nbJobs+nbSeciJobs; j++){
                if(frequency[a-1][j-1]>=1){
                    for( l=1; l<=frequency[a-1][j-1]; l++)
                        Obj2b += jobman2[a-1][j-1][l-1]*(Jobst[a-1][j-1][l-1] <= nbHours*d &&
                            Jobst[a-1][j-1][l-1]+JobHours[a-1][j-1]-1 > nbHours*(d-1) );
                }
            }
        }
        Obj2d += ( Obj2b >=1 );
    }
    catch(...){
        Obj2b.end();
        throw;
    }
    Obj2b.end();
}
}
model.add(tatalObj2 == Obj2c+Obj2d);
}
catch(...){
    Obj2c.end();
    Obj2d.end();
    throw;
}
Obj2c.end();
Obj2d.end();
IloObjective Obj=IloMinimize(env, tatalObj+tatalObj2);

//solver model
IloSolver solver(model);

```

```

solver.solve();

for(m=1; m<=nbSection; m++){
out_file << " Month      " <<m<<" "<<1+SectionDays*(m-1)*nbHours<<"-"<<SectionDays*m*nbHours<<endl;
out_file << "Job        Workstarttime      hours      "<<endl;
    for( a = 1; a <=nbAirlines; a++){
        for( j = 1; j <=nbJobs+nbSeciJobs; j++){
            if(frequency[a-1][j-1]>=1){
                for( l = 1; l <=frequency[a-1][j-1]; l++){
                    if( solver.getValue(Jobst[a-1][j-1][l-1]) >= 1+SectionDays*(m-1)*nbHours){
                        if( solver.getValue(Jobst[a-1][j-1][l-1]) <= SectionDays*m*nbHours)
                            out_file << a <<"_"<<j<<"_"<<l<<"          "<< solver.getValue(Jobst[a-1][j-1][l-1]) <<"
                                "<<JobHours[a-1][j-1]<<endl;
                }
            }
        }
    }
}
out_file << endl;

out_file << "Airline      Section      WorkHours"<<endl;
for( a = 1; a <=nbAirlines; a++){
out_file <<  a <<"          "<<m<<"          "<<solver.getValue(SectionHours[a-1][m-1])<<endl;
}
out_file << endl;

out_file << "jobNumber      Worker"<<endl;
for(a=1; a<=nbAirlines; a++){
    for(j=1; j<=nbJobs+nbSeciJobs; j++){
        if(frequency[a-1][j-1]>=1){
            for(l=1;l<=frequency[a-1][j-1];l++) {
                for(i = 1; i <= nbInspector1; i++) {
                    if(solver.getValue(jobman1[a-1][j-1][l-1][i-1])==1)
                        out_file << "job      " <<  a <<"_"<<j<<"_"<<l<<"          1_"<<i<<endl;// ":" <<
                            solver.getValue(jobman1[j-1][i-1]) << endl;
                }
                for(i = 1; i <= nbInspector2; i++) {
                    if(solver.getValue(jobman2[a-1][j-1][l-1][i-1])==1)
                        out_file << "job      " <<  a <<"_"<<j<<"_"<<l<<"          2_"<<i<<endl;// ":" <<
                            solver.getValue(jobman2[j-1][i-1]) << endl;
                }
            }
        }
    }
}
}

out_file <<solver.getValue(tatalObj) <<endl;
out_file <<solver.getValue(tatalObj2)  <<endl;
}
catch (IloException& ex) {

```



```
cerr << "Error: " << ex << endl;  
}  
env.end();  
return 0;  
}
```