# Anexo

## **Código heurística IDCOA**

#include <stdio.h>

#include <stdlib.h>

#include <C:\Users\Salvador\Desktop\CUCKOO\schedule\_lib.h>

void creosecuencias (int filas, int trabajos,MAT\_INT secuencias);

void vectorFO(MAT\_INT secuencias,VECTOR\_INT Ttrabajos,VECTOR\_INT Ttransporte,VECTOR\_INT Tassembly,int maquinas,int trabajos,int filas,VECTOR\_INT Tegg);

void Hcuckoo (VECTOR\_INT Tegg,int filas,VECTOR\_INT Negg);

void CDELR(float alfamax,int f,float alfamin,int Tite,int trabajos,int filas,VECTOR\_INT Negg,int huevos,VECTOR\_INT DELR);

void calculoCSC(MAT\_INT secuencias,int trabajos,int filas,MAT\_INT CSC);

void intercambios(MAT\_INT secuencias,VECTOR\_INT DELR,int trabajos,int minCSC,int filas);

void busquedalocal(MAT\_INT secuelites,VECTOR\_INT Ttrabajos,VECTOR\_INT Ttransporte,VECTOR\_INT Tassembly,int maquinas,int trabajos,int contador,MAT\_INT BEST,VECTOR\_INT Tcuckoo);

int main()

{

 srand(time(NULL));

 int maquinas;

 int trabajos;

 maquinas=2;

 trabajos=20;

 int huevos;

 float alfamin=0.5;

 float alfamax=2;

 int indice=0;

 int Tite=400;

 int probabilidad=0;

 int contador;

 int mejortiempo=100000;

 VECTOR\_INT Ttrabajos=DIM\_VECTOR\_INT(trabajos);

 VECTOR\_INT Ttransporte=DIM\_VECTOR\_INT(trabajos);

 VECTOR\_INT Tassembly=DIM\_VECTOR\_INT(trabajos);

 VECTOR\_INT TFG=DIM\_VECTOR\_INT(trabajos);

 int i,f,k,a,j,h,x;

 for(a=0;a<trabajos;a++)

 TFG[a]=0;

 for(i=0; i<trabajos; i++)

 {

 Ttrabajos[i]=rand()%100;

 Ttransporte[i]=rand()%10;

 Tassembly[i]=rand()%100;

 }

 ////////////////PRIMERA ITERACION//////////////////////

 int filas=2;

 MAT\_INT secuencias=DIM\_MAT\_INT(filas,trabajos);

 creosecuencias(filas,trabajos,secuencias);

 VECTOR\_INT Tegg=DIM\_VECTOR\_INT(filas);

 vectorFO(secuencias,Ttrabajos,Ttransporte,Tassembly,maquinas,trabajos,filas,Tegg);

 free(secuencias);

 ///////////iteraciones//////////////////////////

 for(f=1;f<=400;f++)

 {

 x=rand()%100;

 srand(x);

 int minCSC=0;

 printf("iteracion %d :\n",f);

 VECTOR\_INT Negg=DIM\_VECTOR\_INT(filas);

 printf("empiezo con %d filas\n",filas);

 Hcuckoo(Tegg,filas,Negg);

 huevos=0;

 for(k=0;k<filas;k++)

 {

 huevos=huevos+Negg[k];

 }

 VECTOR\_INT DELR=DIM\_VECTOR\_INT(filas);

 CDELR(alfamax,f,alfamin,Tite,trabajos,filas,Negg,huevos,DELR);

 for(a=0;a<huevos;a++)

 {

 probabilidad=rand()%99+1;

 if(probabilidad<=10)

 {

 huevos=huevos-1;

 }

 probabilidad=0;

 }

 VECTOR\_INT DELReal=DIM\_VECTOR\_INT(huevos);

 a=0;

 j=1;

 for(h=0;h<huevos;h++)

 {

 DELReal[h]=DELR[a];

 if(j<Negg[a])

 {

 j++;

 }

 else

 {

 j=1;

 a++;

 }

 }

 free(Negg);

 filas=huevos;

 secuencias=DIM\_MAT\_INT(filas,trabajos);

 creosecuencias(filas,trabajos,secuencias);

 free(Tegg);

 MAT\_INT CSC=DIM\_MAT\_INT(filas,filas);

 calculoCSC(secuencias,trabajos,filas,CSC);

 for(i=0;i<filas;i++)

 {

 for(k=0;k<filas;k++)

 {

 if(CSC[i][k]<minCSC)

 minCSC=CSC[i][k];

 }

 }

 free(CSC);

 intercambios(secuencias,DELR,trabajos,minCSC,filas);

 Tegg=DIM\_VECTOR\_INT(filas);

 vectorFO(secuencias,Ttrabajos,Ttransporte,Tassembly,maquinas,trabajos,filas,Tegg);

 VECTOR\_INT clasi=DIM\_VECTOR\_INT(filas);

 VECTOR\_INT copia=DIM\_VECTOR\_INT(filas);

 copyIVector(Tegg,copia,filas);

 sortLVector(copia,clasi,filas,'A');

 free(Tegg);

 MAT\_INT clasisecuencia=DIM\_MAT\_INT(filas,trabajos);

 for(i=0;i<filas;i++)

 {

 for(j=0;j<trabajos;j++)

 {

 clasisecuencia[i][j]=secuencias[clasi[i]][j];

 }

 }

 free(clasi);

 free(copia);

 Tegg=DIM\_VECTOR\_INT(filas);

 vectorFO(clasisecuencia,Ttrabajos,Ttransporte,Tassembly,maquinas,trabajos,filas,Tegg);

 contador=filas\*0.03;

 if(contador==0)

 {

 contador=1;

 }

 MAT\_INT secuelites=DIM\_MAT\_INT(contador,trabajos);

 for(k=0;k<contador;k++)

 {

 for(i=0;i<trabajos;i++)

 {

 secuelites[k][i]=clasisecuencia[k][i];

 }

 }

 MAT\_INT BEST=DIM\_MAT\_INT(contador,trabajos);

 VECTOR\_INT Tcuckoo=DIM\_VECTOR\_INT(contador);

 busquedalocal(secuelites,Ttrabajos,Ttransporte,Tassembly,maquinas,trabajos,contador,BEST,Tcuckoo);

 print\_int\_vector(Tcuckoo,contador);

 print\_int\_matrix(BEST,contador,trabajos);

 for(i=0;i<contador;i++)

 {

 if(Tcuckoo[i]>=mejortiempo)

 {

 indice++;

 }

 if(Tcuckoo[i]<mejortiempo)

 {

 free(TFG);

 TFG=DIM\_VECTOR\_INT(trabajos);

 mejortiempo=Tcuckoo[i];

 for(k=0;k<trabajos;k++)

 {

 TFG[k]=BEST[i][k];

 }

 indice=0;

 }

 if(indice>=3)

 {

 f=400;

 }

 }

 free(Tcuckoo);

 free(BEST);

 if(filas>20)

 {

 filas=20;

 VECTOR\_INT arreglar=DIM\_VECTOR\_INT(filas);

 for(i=0;i<filas;i++)

 {

 arreglar[i]=Tegg[i];

 }

 free(Tegg);

 Tegg=DIM\_VECTOR\_INT(filas);

 for(i=0;i<filas;i++)

 {

 Tegg[i]=arreglar[i];

 }

 free(arreglar);

 }

 }

 print\_int\_vector(TFG,trabajos);

 printf("mejor tiempo %d \n",mejortiempo);

}

void creosecuencias (int filas, int trabajos,MAT\_INT secuencias)

{

 int j,k;

 VECTOR\_INT huevopri=DIM\_VECTOR\_INT(trabajos);

 for(j=0;j<filas;j++)

 {

 randSequence(huevopri,trabajos);

 for(k=0; k<trabajos; k++)

 {

 secuencias[j][k]=huevopri[k];

 }

 }

 free(huevopri);

}

void vectorFO(MAT\_INT secuencias,VECTOR\_INT Ttrabajos,VECTOR\_INT Ttransporte,VECTOR\_INT Tassembly,int maquinas,int trabajos,int filas,VECTOR\_INT Tegg)

{

 int m;

 int i,k,j;

 VECTOR\_INT Tmaquina=DIM\_VECTOR\_INT(maquinas);

 for(i=0; i<filas; i++)

 {

 int Tmaqtrans=0;

 int Tmaqasse=0;

 int menos=0;

 int indice;

 for(m=0; m<maquinas; m++)

 {

 Tmaquina[m]=0;

 }

 for(k=0; k<trabajos; k++)

 {

 for(j=0; j<maquinas; j++)

 {

 if(Tmaquina[j]<=menos)

 {

 menos=Tmaquina[j];

 indice=j;

 }

 }

 Tmaquina[indice]=Tmaquina[indice]+Ttrabajos[secuencias[i][k]];

 menos= Tmaquina[indice];

 if(Tmaquina[indice]<Tmaqtrans)

 {

 Tmaqtrans=Tmaqtrans+Ttransporte[secuencias[i][k]];

 }

 else

 {

 Tmaqtrans=Tmaquina[indice];

 Tmaqtrans=Tmaqtrans+Ttransporte[secuencias[i][k]];

 }

 if(Tmaqtrans<Tmaqasse)

 {

 Tmaqasse=Tmaqasse+Tassembly[secuencias[i][k]];

 }

 else

 {

 Tmaqasse=Tmaqtrans;

 Tmaqasse=Tmaqasse+Tassembly[secuencias[i][k]];

 }

 }

 int Thuevo=0;

 int x;

 for(x=0; x<maquinas; x++)

 {

 if(Thuevo<Tmaquina[x])

 {

 Thuevo=Tmaquina[x];

 }

 }

 Tegg[i]=Thuevo+Tmaqtrans+Tmaqasse;

 }

 free(Tmaquina);

}

void Hcuckoo (VECTOR\_INT Tegg,int filas,VECTOR\_INT Negg)

{

 int FOmejor=Tegg[0];

 int i,k;

 int w;

 for(i=0;i<filas;i++)

 {

 if(Tegg[i]<FOmejor)

 FOmejor=Tegg[i];

 }

 for(k=0;k<filas;k++)

 {

 w=rand()%5;

 Negg[k]=w\*(Tegg[k]/FOmejor);

 }

}

void CDELR(float alfamax,int f,float alfamin,int Tite,int trabajos,int filas,VECTOR\_INT Negg,int huevos,VECTOR\_INT DELR)

{

 int alfaite;

 int x;

 alfaite=alfamax-f\*((alfamax-alfamin)/Tite);

 if(alfaite>alfamax)

 alfamax=alfaite;

 if(alfaite<alfamin)

 alfamin=alfaite;

 for(x=0;x<filas;x++)

 {

 DELR[x]=alfaite\*trabajos\*Negg[x]/(huevos);

 }

}

void calculoCSC(MAT\_INT secuencias,int trabajos,int filas,MAT\_INT CSC)

{

 int S,Z;

 int a,b,i,j,k;

 Z=0;

 S=0;

for(i=0; i<filas; i++)

 {

 for(j=0; j<filas; j++)

 {

 for(k=0; k<trabajos; k++)

 {

 if(secuencias[i][k]==secuencias[j][k])

 Z++;

 }

 for(b=0; b<trabajos-1; b++)

 {

 for(a=0;a<trabajos-1;a++)

 {

 if(secuencias[i][b]==secuencias[j][a] && secuencias[i][b+1]==secuencias[j][a+1])

 S++;

 }

 }

 CSC[i][j]=(S+(2\*Z))/trabajos;

 S=0;

 Z=0;

 }

 }

 }

void intercambios(MAT\_INT secuencias,VECTOR\_INT DELR,int trabajos,int minCSC,int filas)

{

 int i,j;

 VECTOR\_INT Ncambios=DIM\_VECTOR\_INT(filas);

 for(i=0;i<filas;i++)

 {

 if(minCSC<=DELR[i]);

 {

 Ncambios[i]=minCSC;

 }

 if(minCSC>DELR[i])

 {

 Ncambios[i]=DELR[i];

 }

 }

 int limite;

 int z,x,y,w;

 srand(20);

 for(i=0;i<filas;i++)

 {

 limite=Ncambios[i];

 for(j=0;j<limite;j++)

 {

 z=rand()%trabajos;

 x=rand()%trabajos;

 y=secuencias[i][z];

 w=secuencias[i][x];

 secuencias[i][z]=w;

 secuencias[i][x]=y;

 }

 }

free(Ncambios);

}

void busquedalocal(MAT\_INT secuelites,VECTOR\_INT Ttrabajos,VECTOR\_INT Ttransporte,VECTOR\_INT Tassembly,int maquinas,int trabajos,int contador,MAT\_INT BEST,VECTOR\_INT Tcuckoo)

{

 int i,j,k,x,a,b;

 int aux;

 MAT\_INT orden=DIM\_MAT\_INT(contador,trabajos);

 for(i=0;i<contador;i++)

 {

 for(j=0;j<trabajos;j++)

 {

 orden[i][j]=secuelites[i][j];

 }

 }

 VECTOR\_INT Tbus=DIM\_VECTOR\_INT(contador);

 VECTOR\_INT Tbest=DIM\_VECTOR\_INT(contador);

 vectorFO(orden,Ttrabajos,Ttransporte,Tassembly,maquinas,trabajos,contador,Tbest);

 vectorFO(orden,Ttrabajos,Ttransporte,Tassembly,maquinas,trabajos,contador,Tcuckoo);

 for(x=0;x<50;x++)

 {

 for(i=0;i<contador;i++)

 {

 for(j=0;j<trabajos;j++)

 {

 for(k=j+1;k<trabajos;k++)

 {

 aux=orden[i][j];

 orden[i][j]=orden[i][k];

 orden[i][k]=aux;

 vectorFO(orden,Ttrabajos,Ttransporte,Tassembly,maquinas,trabajos,contador,Tbus);

 if(Tbus[i]<Tbest[i])

 {

 Tbest[i]=Tbus[i];

 Tcuckoo[i]=Tbest[i];

 for(a=0;a<trabajos;a++)

 {

 BEST[i][a]=orden[i][a];

 }

 }

 aux=orden[i][j];

 orden[i][j]=orden[i][k];

 orden[i][k]=aux;

 }

 }

 if(Tbest[i]<Tcuckoo[i])

 {

 for(b=0;b<trabajos;b++)

 orden[i][b]=BEST[i][b];

 }

 }

 }

 free(Tbus);

 free(orden);

 free(Tbest);

}

## **Código heurística GRASP**

#include <stdio.h>

#include <stdlib.h>

#include <C:\Users\Salvador\Desktop\CUCKOO\schedule\_lib.h>

int FOBJ(VECTOR\_INT Ttrabajos,VECTOR\_INT Ttransporte,VECTOR\_INT Tassembly,int maquinas,int trabajos,VECTOR\_INT ORDENMEJOR);

int busquedalocal(VECTOR\_INT Ttrabajos,VECTOR\_INT Ttransporte,VECTOR\_INT Tassembly,int maquinas,int trabajos,VECTOR\_INT ORDENMEJOR,int TBEST);

int main()

{

 int maquinas,trabajos;

 int i;

 srand(time(NULL));

 maquinas=4;

 trabajos=20;

 VECTOR\_INT Ttrabajos=DIM\_VECTOR\_INT(trabajos);

 VECTOR\_INT Ttransporte=DIM\_VECTOR\_INT(trabajos);

 VECTOR\_INT Tassembly=DIM\_VECTOR\_INT(trabajos);

 for(i=0; i<trabajos; i++)

 {

 Ttrabajos[i]=rand()%100;

 Ttransporte[i]=rand()%10;

 Tassembly[i]=rand()%100;

 }

 VECTOR\_INT ORDENMEJOR=DIM\_VECTOR\_INT(trabajos);

 VECTOR\_INT secuencia=DIM\_VECTOR\_INT(trabajos);

 copyIVector(Ttrabajos,ORDENMEJOR,trabajos);

 sortIVector(ORDENMEJOR,trabajos,'D');

 int TBEST;

 for(i=0;i<trabajos;i++)

 {

 secuencia[i]=i;

 }

 TBEST=FOBJ(Ttrabajos,Ttransporte,Tassembly,maquinas,trabajos,ORDENMEJOR);

 TBEST=busquedalocal(Ttrabajos,Ttransporte,Tassembly,maquinas,trabajos,ORDENMEJOR,TBEST);

 printf("TBEST %d \n", TBEST);

}

int FOBJ(VECTOR\_INT Ttrabajos,VECTOR\_INT Ttransporte,VECTOR\_INT Tassembly,int maquinas,int trabajos,VECTOR\_INT secuencia)

{

 int m;

 int k,j;

 VECTOR\_INT Tmaquina=DIM\_VECTOR\_INT(maquinas);

 int TBEST;

 int Tmaqtrans=0;

 int Tmaqasse=0;

 int menos=0;

 int indice;

 for(m=0; m<maquinas; m++)

 {

 Tmaquina[m]=0;

 }

 for(k=0; k<trabajos; k++)

 {

 for(j=0; j<maquinas; j++)

 {

 if(Tmaquina[j]<=menos)

 {

 menos=Tmaquina[j];

 indice=j;

 }

 }

 Tmaquina[indice]=Tmaquina[indice]+Ttrabajos[secuencia[k]];

 menos= Tmaquina[indice];

 if(Tmaquina[indice]<Tmaqtrans)

 {

 Tmaqtrans=Tmaqtrans+Ttransporte[secuencia[k]];

 }

 else

 {

 Tmaqtrans=Tmaquina[indice];

 Tmaqtrans=Tmaqtrans+Ttransporte[secuencia[k]];

 }

 if(Tmaqtrans<Tmaqasse)

 {

 Tmaqasse=Tmaqasse+Tassembly[secuencia[k]];

 }

 else

 {

 Tmaqasse=Tmaqtrans;

 Tmaqasse=Tmaqasse+Tassembly[secuencia[k]];

 }

 }

 int Thuevo=0;

 int x;

 for(x=0; x<maquinas; x++)

 {

 if(Thuevo<Tmaquina[x])

 {

 Thuevo=Tmaquina[x];

 }

 }

 TBEST=Thuevo+Tmaqtrans+Tmaqasse;

 return TBEST;

}

int busquedalocal(VECTOR\_INT Ttrabajos,VECTOR\_INT Ttransporte,VECTOR\_INT Tassembly,int maquinas,int trabajos,VECTOR\_INT ORDENMEJOR,int TBEST)

{

 int i,j,k;

 int FO=0;

 int control;

 int itera=0;

 VECTOR\_INT FOite=DIM\_VECTOR\_INT(100);

 for(k=0;k<100;k++)

 {

 FOite[k]=0;

 }

 VECTOR\_INT ordenllegadaBEST=DIM\_VECTOR\_INT(trabajos);

 VECTOR\_INT mejorsecuencia=DIM\_VECTOR\_INT(trabajos);

 copyIVector(ORDENMEJOR,ordenllegadaBEST,trabajos);

 copyIVector(ORDENMEJOR,mejorsecuencia,trabajos);

 for (control=0;control<400;control++)

 {

 FOite[control]=TBEST;

 for(i=0;i<trabajos;i++)

 {

 for(j=i+1;j<trabajos;j++)

 {

 int aux=ordenllegadaBEST[i];

 ordenllegadaBEST[i]=ordenllegadaBEST[j];

 ordenllegadaBEST[j]=aux;

 FO=FOBJ(Ttrabajos,Ttransporte,Tassembly,maquinas,trabajos,ordenllegadaBEST);

 if(FO<FOite[control])

 {

 FOite[control]=FO;

 copyIVector(ordenllegadaBEST,mejorsecuencia,trabajos);

 }

 aux=ordenllegadaBEST[i];

 ordenllegadaBEST[i]=ordenllegadaBEST[j];

 ordenllegadaBEST[j]=aux;

 }

 }

 if(TBEST<=FOite[control])

 {

 itera++;

 if(itera==20)

 {

 randSequence (mejorsecuencia,trabajos);

 copyIVector(mejorsecuencia,ordenllegadaBEST,trabajos);

 itera=0;

 }

 }

 if(TBEST>FOite[control])

 {

 TBEST=FOite[control];

 copyIVector(mejorsecuencia,ordenllegadaBEST,trabajos);

 }

 }

 return TBEST;

}