Appendix D

SA Program

Program Description

Input:

- The generating unit system data which consists of the heat input-output curves, ramp rate limits, unit operating limits, fuel cost of each unit, initial power outputs at time t=0, forecast load demands, and transmission loss B-matrix coefficients.

Output:

- The result of the program consists of the power output of each generating units and their highest and lowest possible power output, the power loss, the generator fuel cost and the CPU time of each time period.

```
/*-------------------------------------------------------------*/
/* Program : Simulated Annealing for Constrained ED problem
/* Compile : gcc sa.c -o SA -lm
/* Execute : SA
/*-------------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "os.h"
#include "coops.c"

#define  MaxUnit 80
#define  MaxMember 5
#define  MaxHour 500
#define  INF pow(10,14)
#define  Gamma 1
#define  r 0.95
#define  Type  ".txt"
FILE     *OUTP;
FILE     *Inp;
typedef  unsigned short  Word;
char     Command[30],rsfile[50];
char     MyFolder[15],datafile[20];
/* Cost Coefficient */
Word     piece[MaxUnit];
double   SeedBestCost;
int      mseed;/*,BestFile,Fmseed,Lmseed;*/
double   SumPrmin,SumPrmax,BestCost;
double   PowPlant,SumPow,Demand_Actual,PLoss,SumBestCost;
double   Fuel_Cost[MaxUnit],F[MaxUnit];
double   P_int[MaxUnit][MaxMember];
double   Ah[MaxUnit][MaxMember],Bh[MaxUnit][MaxMember],Ch[MaxUnit][MaxMember];
double   aC[MaxUnit][MaxMember],bC[MaxUnit][MaxMember],cC[MaxUnit][MaxMember];

/* B Matrix */
double   B00;
double   B0[MaxUnit];
double   B[MaxUnit][MaxUnit];
double   P_Base;
Word     Step,MaxStep;
Word     Member;

/* Power Output variables */
double   Pmin[MaxUnit],Pmax[MaxUnit],UR[MaxUnit],DR[MaxUnit];
double   Prmin[MaxUnit],Prmax[MaxUnit],P[MaxUnit];
double   upperbound[MaxUnit],lowerbound[MaxUnit],N[MaxUnit];
double   Pd[MaxHour];
Word     NUnit;

/* Time */
struct   timeval   start_time,stop_time,diff_time,SumTime;
struct   timezone  timz;

double Initial(double SumPrmax,double Demand_Actual,double *P,double *PLossBest)
{       double P_ini[MaxUnit];
        double PowPlant,PLoss,SumPow,BestCost,Cost,SumPow_Best;
        Word   ref,i,ineq,varyref;
        BestCost = INF;

        /*------------- for divider --------------------*/
        for(i = 0; i < NUnit; i++)
        {       P[i] = (Prmax[i]*Demand_Actual)/(SumPrmax);
                if(P[i]<Prmin[i])P[i] = Prmin[i];
                if(P[i]>Prmax[i])P[i] = Prmax[i];
        }
        /*------------------- Pref and PLoss Calculation  ----------------------*/
        /*for initialize there is a problem with ref P in order to meet the load demand
        /*The ref P should be vary to get the best fit  */
        varyref=0;
        /*Don't want P_next_m to change during finding the best initial P*/
        for (i = 0; i < NUnit; i++)P_ini[i] = P[i];
        do{
                for(ref = 0; ref < NUnit; ref++)
                {
                        PLoss = B_Matrix1(ref,Demand_Actual,P_ini,&PowPlant,&SumPow);
                        /*----- if all initial P are in the ramp---------*/
```

```
                    ineq = 0;
                    for (i=0; i < NUnit; i++)
                    if((P_ini[i]<Prmin[i])||(P_ini[i]>Prmax[i]))ineq++;
                    /*---------- cost calculation ----------------*/
                    if ((fabs(SumPow-Demand_Actual)<0.00001)&&(ineq==0))
                    {  varyref++;
                        Cost = CostCal(P_ini);
                        if(Cost < BestCost)
                        {   BestCost = Cost;
                            SumPow_Best = SumPow;
                            *PLossBest = PLoss;
                            for(i=0;i<NUnit;i++)P[i] = P_ini[i];
                        }
                    }
                    else
                    {
                        for (i = 0; i < NUnit; i++)
                        {   if(P_ini[i]<Prmin[i])P_ini[i] = Prmin[i];
                            if(P_ini[i]>Prmax[i])P_ini[i] = Prmax[i];
                        }
                    }
                }/*end of vary ref loop*/
                if(varyref == 0)
                {
                    if((Demand_Actual-SumPow)>0)for (i=0; i<NUnit; i++)P_ini[i] *= 1.01;
                    else for (i=0; i<NUnit; i++)P_ini[i] *= 0.99;
                }
            }while(varyref == 0);

    return SumPow_Best;
}

Word rndint(Word low,Word high)
{
    Word    i;
    i = ((rand()*(high - low))/RAND_MAX) + low;
    return i;
}
double rndreal(double low,double high)
{   double    d;
    d = ((rand()*(high - low))/RAND_MAX) + low;
    return d;
}

double SA(double Demand_Actual,double *P_Best,Word Trial,double sigma_1,double
sigma_f,double *SumPow_Best,double *PLoss_Best,Word *ref_Best)
{       double  Pi[MaxUnit],P_best_m[MaxUnit],Pdat[MaxUnit];
        double  Nupper,Nlower;
        double  SumPow,PowPlant,PLoss,PL_best_m,SumPow_m;
        double  Cost,BestCost_m,BestCost,costdiff;
        double  Pop,sigma_k;
        Word    ref,ref_m,i,k,m,ineq;

        BestCost = pow(10,14);
        BestCost_m = CostCal(P_Best);
        for(i=0; i<NUnit; i++)P_best_m[i] = P_Best[i];
        SumPow_m = *SumPow_Best;
        PL_best_m = *PLoss_Best;
        ref_m = *ref_Best;

   k = 1;
   do{

        /*-----  Calculate sigma_k -----*/
        sigma_k = pow(r,k-1)*sigma_1;

        for(i=0; i<NUnit; i++)Pi[i] = P_best_m[i];

       for(m = 1; m <= Trial; m++)
        {
                for (i = 0; i<NUnit; i++)
                {       lowerbound[i] = -Gamma*sigma_k + Pi[i];
                        upperbound[i] =  Gamma*sigma_k + Pi[i];
                        if(lowerbound[i]<Prmin[i])lowerbound[i] = Prmin[i];
                        if(upperbound[i]>Prmax[i])upperbound[i] = Prmax[i];
                }
```

```
                do{
                        /*--- (a) ----*/
                        /*Random number between 0 and NUnit-1*/
                        ref = rndint(0,(NUnit-1));
                        /*---- (b) ----*/
                        for(i = 0; i < NUnit; i++)N[i] = 0;
                        ineq = 0;
                        for (i = 0; i<NUnit; i++)if (i != ref)
                        {  Nupper = upperbound[i]-Pi[i];
                           Nlower = lowerbound[i]-Pi[i];
                           if(Nlower>0)Nlower=0; if(Nupper<0)Nupper=0;
                           N[i] = rndreal(Nlower,Nupper);
                           /*N[i] = Perturb(upperbound[i],lowerbound[i],Pi[i]);*/
                           /*---- (c) ----*/
                           Pdat[i] = Pi[i]+N[i];
                           if((Pdat[i]>upperbound[i])||(Pdat[i]<lowerbound[i]))++ineq;
                        }
                } while(ineq != 0);


                /*---- (d) ----*/
                /*-------------------- Pref and PLoss Calculation  ----------------------*/
                PLoss = B_Matrix1(ref,Demand_Actual,Pdat,&PowPlant,&SumPow);

                /*---------- (f) -------------*/
                if (fabs(SumPow-Demand_Actual)<0.00001)
                {
                        /*---------- cost calculation ----------------*/
                        Cost = CostCal(Pdat);
                         /*-------- (g) -------------*/
                        costdiff = Cost - BestCost_m;
                        if(costdiff < 0)
                        {  BestCost_m = Cost;
                           for (i=0; i<NUnit; i++){P_best_m[i] = Pdat[i];Pi[i] = Pdat[i]; }
                           PL_best_m = PLoss;SumPow_m = SumPow;ref_m=ref;
                        }
                        else
                        {
                           if((costdiff/sigma_k)>100)Pop = 0;
                           else Pop = 1/(1+exp(costdiff/sigma_k));
                           if(Pop > (rand()/RAND_MAX))for (i=0; i<NUnit; i++) Pi[i] = Pdat
[i];
                        }
                }
        }

        if(BestCost_m < BestCost)
        {  BestCost = BestCost_m;
           for (i=0; i<NUnit; i++)P_Best[i] = P_best_m[i];
           *PLoss_Best    = PL_best_m;
            *SumPow_Best   = SumPow_m;
            *ref_Best = ref_m;
         }
        k++;}while(sigma_k >=sigma_f);

   return BestCost;
}

void TitleReport(Word Trial,double sigma_1,double sigma_f)
{     Word i;
                fprintf(OUTP,"\nData: %s  Result: %s\n",datafile,rsfile);
        fprintf(OUTP,"\nTrial %d Sigma1 %lf Sigmaf %lf",Trial,sigma_1,sigma_f);
                fprintf(OUTP,"\nStep\tDemand\tPowPlant\tP_Loss\tSumPow\t");
                for (i=0; i < NUnit; i++) fprintf(OUTP,"Prmax[%d]\t",i);
                for (i=0; i < NUnit; i++) fprintf(OUTP,"Prmin[%d]\t",i);
                for (i=0; i < NUnit; i++) fprintf(OUTP,"P[%d]\t",i);
                fprintf(OUTP,"SACost\n");
}

void Report()
{  Word i;
   PowPlant = 0;
   for(i=0; i<NUnit; i++)PowPlant += P[i];
       fprintf(OUTP,"\n%d",Step);
       fprintf(OUTP,"\t%lf\t%lf\t%lf\t%lf",Pd[Step],PowPlant,PLoss,SumPow);
   for (i=0; i<NUnit; i++)fprintf(OUTP,"\t%lf",Prmax[i]);
   for (i=0; i<NUnit; i++)fprintf(OUTP,"\t%lf",Prmin[i]);
```

```
        for (i=0; i<NUnit; i++)fprintf(OUTP,"\t%lf",P[i]);
        fprintf(OUTP,"\t%.4f",BestCost);
        fprintf(OUTP,"\t%.6i\t%.6i",diff_time.tv_sec,diff_time.tv_usec);
    fflush(OUTP);
}

void ShowScreen(void)
{  Word i;
        write_xy(1,1,"");
    printf("\nYou're running SA in case of %s",datafile);
    printf("\nFile No. %2d PowerDemand = %10.4f Step %2d\n",mseed,Pd[Step],Step);

    for (i=0; i<NUnit; i++)
    {  if(i%8 == 0)printf("\n\n");
       printf("%3.4f  ",P[i]);
    }
    printf("\nPLoss = %4.4f",PLoss);
    printf("\nSumPow = %lf PowPlant = %lf",SumPow,PowPlant);
    printf("\tBestCost = %.4f",BestCost);

    printf("\nSumCost = %.4f",SumBestCost);
    printf("\nCPU time = %6i seconds %6i micro_seconds",diff_time.tv_sec,diff_time.tv_usec);
    printf("\nSumCPU time = %6i seconds %6i
micro_seconds\n",SumTime.tv_sec,SumTime.tv_usec);


}

int main ()
{       FILE    *Summary;
        char    SummaryName[30];
        double  SummaryCost;
        Word    Trial,ref,i;
        int      Fmseed,Lmseed,BestFile;
        double  sigma_1,sigma_f;

    printf("\nData File?"); scanf("%s",datafile);
    printf("\nTrial ?");scanf("%d",&Trial);
    printf("\nSigmal ?");scanf("%lf",&sigma_1);
    printf("\nSigmaf ?");scanf("%lf",&sigma_f);
    printf("\nTrial %d, Sigma_1 %lf Sigma_f %lf",Trial,sigma_1,sigma_f);
    printf("\nFirst mseed start from 1:"); scanf("%d",&Fmseed);
    printf("\nLast mseed end at 20:"); scanf("%d",&Lmseed);
    printf("\nSA run between mseed %d - %d",Fmseed,Lmseed);
    BestFile = Fmseed;


    if((Inp     = fopen(datafile,"r"))==NULL){
        printf("cannot open datafile %s input\n",datafile);
        exit(0);
        }

    sprintf(SummaryName,"Summaryof%s",datafile);
    if((Summary = fopen(SummaryName,"w"))==NULL){
        printf("cannot open summaryfile %s \n",SummaryName);
        exit(0);
        }

    ReadData();
    BuildCoef();

    /* Make Folder Report */
    sprintf(MyFolder,"Result%i",NUnit);
    sprintf(Command,"mkdir %s",MyFolder);
    system(Command);

     /*--------------Vary the Seed No.-------------*/
     SeedBestCost = INF;
     SummaryCost = 0;
     for(mseed = Fmseed; mseed <= Lmseed; mseed++)
     {

         srand(mseed);

         /*------ Make LoopFile ----------*/
         sprintf(rsfile,"Result%d/SA%d%s",NUnit,mseed,datafile);
```

```
        if((OUTP = fopen(rsfile,"w"))==NULL){
        printf("cannot open file output in folder Result!!\n");
        exit(0);
        }

        TitleReport(Trial,sigma_1,sigma_f);

        fflush(OUTP);
        ClrScr();
        SumBestCost = 0;
        SumTime.tv_sec = 0;
        SumTime.tv_usec = 0;

        for(i=0;i<NUnit;i++)P[i]=Pmin[i];

        for(Step = 0; Step < MaxStep; Step++)
        {
                Start_Time();
                Demand_Actual = Pd[Step];
                BestCost = INF;
                SetRampLimit(P,Prmax,Prmin,&SumPrmax,&SumPrmin);
                SumPow = Initial(SumPrmax,Demand_Actual,P,&PLoss);
                BestCost = SA(Demand_Actual,P,Trial,sigma_1,sigma_f,&SumPow,&PLoss,&ref);
                SumBestCost += BestCost;

                Stop_Time();
                ReadTime(start_time,stop_time,&diff_time);
                Sum_Time();

         Report();
         ShowScreen();
         fflush(OUTP);

        }/*------------ end of Step Hour loop ---------*/

        fprintf(OUTP,"\nSumCost = %lf",SumBestCost);
        fprintf(OUTP,"\nSumCPU Time %6i seconds %6i microseconds",SumTime.tv_sec,
SumTime.tv_usec);

        if((SumBestCost<SeedBestCost) || (mseed==Fmseed))
        {   SeedBestCost = SumBestCost;   BestFile =mseed; }
        fprintf(OUTP,"\n\nTotal Operating Cost = %.4f",SumBestCost);
        fprintf(OUTP,"\nBest File No.          = %d",BestFile);
        fprintf(OUTP,"\nBest Total Cost        = %.4f",SeedBestCost);
        fflush(OUTP);

        /*---- Summary ----*/
        SumBestCost /= 4;
        SummaryCost += SumBestCost;
        fprintf(Summary,"\nSeed No. %2d %.41f",mseed,SumBestCost);

        } /*------------ end of mseed loop ------------*/
        printf("\nFile No. %d",mseed);
        printf("\nBest File No. = %d",BestFile);
        printf("\nBest Total Cost = %.4f\n",SeedBestCost);

        fprintf(Summary,"\nAvg Cost %lf",(SummaryCost/Lmseed));
        fprintf(Summary,"\nBestCost %.41f",(SeedBestCost/4));
        fclose(OUTP);

        return 0;
}
```