Appendix C

ZDP 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 : Zoom Dynamic Programming for Constrained ED problem
/* Compile : gcc zdp.c -o ZDP -lm
/* Execute : ZDP
/*-------------------------------------------------------------*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "os.h"
#include "coops.c"
#define MaxUnit   10
#define MaxIter   10
#define MaxMember 5
#define MaxHour 500
#define INF pow(10,14)
FILE    *OUTP;
FILE    *Inp;

typedef unsigned short  Word;
char    MyFolder[10],Command[20],datafile[20],rsfile[40];
Word    piece[MaxUnit],P_flag[MaxUnit],NUnit,Step,MaxStep,Member,LstIter;
int     Dimension[MaxUnit];
/* Cost Coefficient */
        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];
        double  SumBestCost,Cost,BestCost;

/* B Matrix */
        double  B00;
        double  B0[MaxUnit];
        double  B[MaxUnit][MaxUnit];
/* Based Power */
        double  P_Base;

/* Power output variables */
        unsigned long combination;
        Word    i,j,*pt;
        double  P_ini[MaxUnit];
        double  Pmin[MaxUnit],Pmax[MaxUnit],UR[MaxUnit],DR[MaxUnit];
        double  Prmin[MaxUnit],Prmax[MaxUnit];
        double  Upperbound[MaxUnit],Lowerbound[MaxUnit];
        double  Pd[MaxHour];
        double  SumPrmin,SumPrmax,Demand_Actual;
        double  PLoss_Best[MaxIter];
        double  PowPlant_Best,SumPow_Best;
        int     Range;
        double  PF[MaxUnit],BC[MaxIter];
        double  PLoss_ini,Demand;
        double  **P_Best;

/* Parameters -*/
        int     NumStepSize,Iter;
        double  delta_1,K_delta,StepSize;

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


void SetBound(double *P)
{  Word i;
   /*------------- Set Bound -------------*/
        for (i=0; i < NUnit; i++)
        {  Lowerbound[i] = P[i] - (StepSize*Dimension[i]);
              if(Lowerbound[i]<=Prmin[i])Lowerbound[i] = Prmin[i];
              Upperbound[i] = Lowerbound[i] + (2*StepSize*Dimension[i]);
              if(Upperbound[i]>=Prmax[i])Upperbound[i] = Prmax[i];
}}


void TitleReport(void)
{       Word i;
        fprintf(OUTP,"\nData: %s  Result: %s\n",datafile,rsfile);
```

```
        fprintf(OUTP,"\nStep_ini %lf K_delta %lf LstIter %d",delta_1,K_delta,LstIter);
        fprintf(OUTP,"\nStep\tPowDemand\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,"DPCost");
        fprintf(OUTP,"\tTime(Sec)\tMicroSec\n");
}

void Report(void)
{  Word i;
   fprintf(OUTP,"\n%d",Step);
   fprintf(OUTP,"\t%lf\t%lf\t%lf\t%lf",Pd[Step],PowPlant_Best,PLoss_Best[LstIter-
1],SumPow_Best);
   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_Best[NUnit-1][i]);
   fprintf(OUTP,"\t%.4f",BestCost);
   fprintf(OUTP,"%d\t%d",diff_time.tv_sec,diff_time.tv_usec);
   fflush(OUTP);
}

void ShowScreen(void)
{  Word i;
   write_xy(1,1,"");
   printf("\nYou're running DP in case of %s",datafile);
   printf("\nPowerDemand = %10.4f Step %2d\n",Pd[Step],Step);
   for (i=0; i<NUnit; i++)
   {  if(i%8 == 0)printf("\n\n");
      printf("%3.4f  ",P_Best[NUnit-1][i]);
   }
   printf("\nPLoss = %4.4f",PLoss_Best[LstIter-1]);
   printf("\nSumPow = %lf",SumPow_Best);
   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);
}

double DP(int unit,double Demand)
{
        double COST;double DPCost;double P;
        P_flag[unit] = 0;
        DPCost = INF;
        P = Lowerbound[unit];
        do
        {       if(unit==1)
                {  if(((Demand-P)<=Prmax[0])&&((Demand-P)>=Prmin[0]))
                        {       COST =CostCal(1,P,PF[1])+CostCal(0,Demand-P,PF[0]);
                                if(COST<DPCost)
                                {       P_Best[unit][0] = Demand-P; P_Best[unit][1] = P;
                                        DPCost = COST;
                                }
                        }
                }
                else
                {
                        COST=CostCal(unit,P,PF[unit]) + DP(unit-1,Demand-P);
                        if(COST<DPCost)
                        {       DPCost = COST;
                                P_Best[unit][unit] = P;
                                for(i = 0; i< unit; i++)P_Best[unit][i] = P_Best[unit-1][i];
                        }
                }

        P+=StepSize;
        if(P>=Upperbound[unit]){P=Upperbound[unit];P_flag[unit]++;}
        }while(P_flag[unit]<2);

        return DPCost;
}


void CalPF(double *PF,double *P)
{
   int q,r;
```

```
    Word i,unit;
    double A0,A1,dF,F;
    for(unit = 0; unit<NUnit; unit++)
        {       A0=0;A1=0;
                for(i=0;i<NUnit;i++)
                {       if(i==unit){q=2;r=0;}else {q=r=1;}
                        A0 = A0+(q*B[unit][i]*P[i]/P_Base);
                        A1 = A1+(r*B[i][unit]*P[i]/P_Base);
                }
                dF = A0+A1+B0[unit];
                F = 1.0/(1.0-dF);
                PF[unit] = F;
    }
}


int main ()
{
    Word  nbytes;
    /* Reserve memory for the best power output */
    nbytes = MaxUnit*sizeof(double);
    if((P_Best=(double**)malloc(nbytes))==NULL)nomemory("P_Best");
    for(i=0; i<MaxUnit;i++)
    if((P_Best[i]=(double*)malloc(nbytes))==NULL)nomemory("P_Best");

    /* Input Filename */
    printf("\nData Filename(.txt)?"); scanf("%s",datafile);

    /* Parameter Selection */
    printf("\nInitial Step Size ?");scanf("%lf",&delta_1);
    printf("\nK_delta (5)?");scanf("%lf",&K_delta);
    printf("\nNumbers of Iteration ?");scanf("%d",&LstIter);
    printf("\nInit Step %lf, K_delta %lf Iterations %d",delta_1,K_delta,LstIter);
    printf("\nDP run %s",datafile);


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

    ReadData();
    BuildCoef();
    CalDim();

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

        sprintf(rsfile,"Result%d/DP%s",NUnit,datafile);
        if((OUTP = fopen(rsfile,"w"))==NULL){
                printf("cannot open file output\n");
                exit(0);
                }
    TitleReport();
    /*--------- end of openning the Result file ----------*/

    /*-------- Set P_Best, Reset TotalBestCost and CPU time------*/
        for(i=0; i<NUnit; i++)P_Best[NUnit-1][i] = Pmin[i];
         ClrScr();
         SumBestCost = 0;
         SumTime.tv_sec = 0; SumTime.tv_usec = 0;



        for(Step = 0; Step < MaxStep; Step++)
        {       Demand_Actual = Pd[Step];
                Start_Time();
                /*-------- Set bestcost to infinity and set ramp limit --------*/
                BestCost = INF;
                SetRampLimit(P_Best[NUnit-1],Prmax,Prmin,&SumPrmax,&SumPrmin);
                for(i=0;i<NUnit;i++)PF[i] = 1;
                PLoss_ini = 0;

                /*------------------- Iteration LOOP ----------------------*/
```

```
            for(Iter = 0; Iter < LstIter; Iter++)
            {       if(Iter < 2)
                    {       StepSize = delta_1;
                            for (i=0; i < NUnit; i++)
                            {Upperbound[i] = Prmax[i]; Lowerbound[i] = Prmin[i];}
                    }
                    else
                    {       StepSize /= K_delta;
                            SetBound(P_Best[NUnit-1]);
                    }

                    Demand = Demand_Actual+PLoss_ini;
                    PLoss_Best[Iter] = PLoss_ini;

                    BC[Iter] = DP(NUnit-1,Demand);

                    if(Iter>=LstIter-2)for (i=0; i < NUnit; i++)PF[i]=1;
                    else CalPF(PF,P_Best[NUnit-1]);

                    for(i=0;i<NUnit;i++)P_ini[i]= P_Best[NUnit-1][i];
                    PLoss_ini = B_Matrix(P_ini);
            }/* end of vary iteration */

            PowPlant_Best = 0;
            for (i=0; i<NUnit; i++)PowPlant_Best += P_Best[NUnit-1][i];
            PLoss_Best[LstIter-1]= B_Matrix(P_Best[NUnit-1]);
            SumPow_Best = PowPlant_Best- PLoss_Best[LstIter-1];
            BestCost = BC[LstIter-1];
            SumBestCost += BC[LstIter-1];

            Stop_Time();
            ReadTime(start_time,stop_time,&diff_time);
            Sum_Time();
             ShowScreen();
            Report();
        } /*---------- end of vary step ---------*/

        fprintf(OUTP,"\nSumCost = %lf",SumBestCost);
        fprintf(OUTP,"\nSumCPU Time %6i seconds %6i microseconds",SumTime.tv_sec,
SumTime.tv_usec);
        fclose(OUTP);
        return 0;
}
```