เอกสารอ้างอิง

ศิริจันทร์ ทองประเสริฐ. การจำลองแบบปัญหา. โรงพิมพ์จุฬาลงกรณ์มหาวิทยาลัย. กรุงเทพมหานคร, 2535.

Arrow, K.J., Chenery, H.B., Minhas, B.S., and Solow, R.M. "Capital-Labor Substitution and Economic Efficiency." Review of Economics and Statistics. Vol XLIII, 1961.

Box, G.E.P., Hunter, W.G., and Hunter, J.S. Statistics for Experimenters. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons. Inc. New York, 1978.

Blair, R.D., and Kenny, L.W. Microeconomics for Managerial Decision Making. McGraw-Hill. New York, 1987.

Chuvej Chansa-ngavej. Decision Criteria Under Uncertainties in Multiperiod Capital Budgeting. Ph D. Dissertation. Department of Industrial and Systems Engineering. The Ohio State University. Columbus. OH, 1989.

Degarmo, E.P., Sullivan, W.G., and Canada, J.R. Engineering Economy. Seventh Edition. Macmillan Publishing Company, 1984.

de Neufville, R. Applied Systems Analysis Engineering Planning and Technology Management. McGraw-Hill. New York, 1990.

Domar, E.D., Eddie, S.M., Herrick, B.H., Hohenberg, M., Intriligator, M.D., and Miyamoto, I. "Economic Growth and Productivity in the United States, Canada, United Kingdom, Germany and Japan in the Post-War Period." Review of Economics and Statistics. Vol 46, 1964.

Dhrymes, P.J. "A Comparison of Productivity Behavior in Manufacturing and Service Industries." Review of Economics and Statistics Vol. 45, 1963.

Hax, A.C., and Wiig, K.M. "The Use of Decision Analysis in Capital Investment Problems" Chapter 13 in Bell, D.E., Keeney, R.L.,and Raiffa, H. Conflicting Objectives in Decision. Wiley Chichester, UK. pp. 277-297, 1977.

Hammer, M. "Reengineering Work : Don't Automate, Obliterate. " Harvard Business Review. Vol. 68. No. 4 . Jul - Aug, 1990.

Kemp, M.C., Sheshinski, E., and Chi Thanh, P. "Economic Growth and Factor Substitution." International Economic Review. Vol 8, 1967.

Kmenta, J."On Estimation of the CES Production Function." International Economic Review. Vol 8, 1967.

Law, A.M., and Kelton, W.D. Simulation Modeling and Analysis. McGraw-Hill. New York, 1982.

Montgomery, D.C. Design and Analysis of Experiments. Third Edition. John Wiley & Sons, 1991.

Nerlove, M. "Notes on the Production and Derived Demand Relations Included in Macro-Econometric Models." International Economic Review. Vol 8, 1967.

Oakford, R.V.,Salazar, A., and Digiulio, H.A. "Factors that Effect the Growth Rate of Equity Capital". IIE Transaction. Vol. 17. No.2. June, 1985.

Oakford, R.V.,Salazar, A., and Lohmann, J.R. "Mathematics, Computation, and Practical Capital Budgeting Decisions". Institute of Industrial Engineer. Annual Industrial Engineering Conference Proceeding, 1982.

———, R.V., and Salazar, A. "The Long Term Effectiveness of "Exact" and Approximate Capital Rationing Procedures Under Uncertainty and Incomplete Information". Journal of Business Finance & Accounting 8.1, 1981.

Park, C.S., and Sharp-Bette, G.P. Advanced Engineering Economics. John Wiley & Sons. New York, 1990.

Salvatore, D. Microeconomic Theory . Schaum's Outline Series. McGraw-Hill. New York, 1974.

Samuelson, P.A. Economics Tenth Edition, McGraw-Hill, 1976.

Solow, R.M. "Technical Change and the Aggregate Production Function". Review of Economics and Statistics. Vol. 39, 1957.

Sumanth, D.J. Productivity Engineering and Management. McGraw-Hill. New York, 1985.

Thompson, A.A. Economics of the Firm Theory and Practice. Fifth Edition. Prentice - Hall, 1989.

Wilson, T.A., Eckstein, O. "Short-Run Productivity Behavior in U.S. Manufacturing." Review of Economics and Statistics Vol. 46, 1964.

ภาคผนวก

โปรแกรม คอมพิวเตอร์
INVESMENT SIMULATOR
(INVESTOR)

```
/*Program invesment Simulator (INVESTOR) was originally developed
and written in "SIMCRIPT II" by " Dr. Chuvej Chansa-ngavej." It
was then converted into "C" programming language in order  to run
on microcomputer by
               Kittirat Lelahuta
               Dept.   of  Industrial  Engineering,
               Chulalongkorn  University.
The basic hardware and software requirements are micro processor
80286 with at least 1 MB RAM and DOS ver 3.3 and above.*/


#include<stdio.h>
#include<stdlib.h>

#define MAX_STR 80
#define LENGTH 17
#define LIFES 6
#define N1 1
#define N2 2
#define N3 3
#define N4 4
#define N5 5
#define N6 6
#define N7 7
#define HORN 11
#define T1 3
#define T2 3
#define MAX_RVAL 32768.0

FILE *fp_out;
char f_out_name[MAX_STR];

FILE *fp_seed;/* file for random seed */

int study;

char title_1[MAX_STR];
char title_2[MAX_STR];
char title_3[MAX_STR];

unsigned seed[30];/* seed number for random number */
unsigned temp_seed ;
int horizon; /* simulation length */

float lb_pfi,mo_pfi,ub_pfi;
float lb_cpi,mo_cpi,ub_cpi; /* economic parameters */
float lb_wri,mo_wri,ub_wri;

int lb_pro_num,mo_pro_num,ub_pro_num; /* numbers of projects */

int lb_pro_life,mo_pro_life,ub_pro_life; /* project life */
float lb_k,mo_k,ub_k; /* value of K */

float lb_mo_IRR,mo_IRR,ub_mo_IRR;
float lb_IRR,ub_IRR;       /* distribution of IRR */
float lb_gr,mo_gr,ub_gr; /* growth rate */

float start_bdgt; /* starting budget of simulation */
float a,b; /* output elasticities */

float lb_cap,ub_cap; /* distribution of output capacity */

float ll_ep,ul_ep;
float lm_ep,um_ep; /* price elasticities */
float lu_ep,uu_ep;

float ll_ew,ul_ew;
float lm_ew,um_ew; /* wages elasticities */
float lu_ew,uu_ew;

float lb_q1,ub_q1;
float lb_q2,ub_q2;
float lb_q3,ub_q3;
float lb_q4,ub_q4;
int m1,m2;

float MARR,STRR,HORR;
float au,bu;
float equity;
float z_prior[HORN],dvdend[HORN];

float pfi[LENGTH];
char pfi_class[LENGTH];

float cpi[LENGTH];
char cpi_class[LENGTH];

float wri[LENGTH];
char wri_class[LENGTH];

int pro_num[LENGTH];
float ant_bdgt[LENGTH],gr[LENGTH];
int pro_life[N6][LENGTH];
float k[N6][LENGTH],cap[N6][LENGTH],c[N6][LENGTH];

float mod_IRR[N6][LENGTH],IRR[N6][LENGTH];
float l_ep[N6][LENGTH],m_ep[N6][LENGTH],u_ep[N6][LENGTH];
float l_ew[N6][LENGTH],m_ew[N6][LENGTH],u_ew[N6][LENGTH];
float q1[N6][LENGTH],q2[N6][LENGTH],q3[N6][LENGTH],q4[N6][LENGTH];

float fk[N6][LENGTH],scale_up[N6][LENGTH],sum_fz[N6][LENGTH];
float z[LIFES][N6][LENGTH],life[N6][LENGTH];
float d[LIFES][N6][LENGTH];
float dm_pfi[T1][LENGTH],dm_cpi[T1][LENGTH],dm_wri[T1][LENGTH];

float dm_z[T2][T1][LIFES][N6][LENGTH];

/* variables used by "cf_gen()" function */

float ep[LIFES][N6][LENGTH],ew[LIFES][N6][LENGTH];
float p[LIFES][N6][LENGTH],w[LIFES][N6][LENGTH];
float l[LIFES][N6][LENGTH],y[LIFES][N6][LENGTH];
float income[LIFES][N6][LENGTH],outgo[LIFES][N6][LENGTH];
float raw_z[LIFES][N6][LENGTH],fz[LIFES][N6][LENGTH];

/* variables used by "dm_cf_gen()" function */

float dm_w[T2][T1][LIFES][N6][LENGTH];
float dm_p[T2][T1][LIFES][N6][LENGTH];
float dm_y[T2][T1][LIFES][N6][LENGTH];
float dm_l[T2][T1][LIFES][N6][LENGTH];
float dm_income[T2][T1][LIFES][N6][LENGTH];
float dm_outgo[T2][T1][LIFES][N6][LENGTH];
float dm_ep[T2][LIFES][N6][LENGTH];
float dm_ew[T2][LIFES][N6][LENGTH];
float dm_raw_z[T2][T1][LIFES][N6][LENGTH];
float s_scale_up[T2][T1][N6][LENGTH];
float dm_fk[T2][T1][N6][LENGTH];
float fz_z[T2][T1][LIFES][N6][LENGTH];
float sum_fz_z[T2][T1][N6][LENGTH];
float dm_IRR[T2][T1][N6][LENGTH];

float one_bun_k[N6][HORN];
float two_bun_k[N6][N6][HORN];
float three_bun_k[N5][N5][N5][HORN];
float four_bun_k[N4][N4][N4][N4][HORN];
float five_bun_k[N3][N3][N3][N3][N3][HORN];

float one_bun_worth[N6][HORN];
float two_bun_worth[N6][N6][HORN];
float three_bun_worth[N5][N5][N5][HORN];
float four_bun_worth[N4][N4][N4][N4][HORN];
float five_bun_worth[N3][N3][N3][N3][N3][HORN];


float k_best[LENGTH];
float z_best[LIFES][LENGTH];
float d_best[LIFES][LENGTH];

int best_flag;
float best_worth;
float shot;

int j1_best,j2_best,j3_best,j4_best,j5_best,j6_best,j7_best;
float maxw;
float sum_k[LENGTH],k_min[LENGTH];
float sum_z,sum_d;

float budget[LENGTH];
float xequity,mult;

float uniform();
float triangle();
float max();
float min();


int iuniform();
int itriangle();
int imax();
int imin();
```

```
main()

{

/* start to read all the input data from file */

read_input(); /* call read function */

fp_seed = fopen("seed.dat","r");

write_input(); /* call write function to echo check the input
                                read in */
read_seed();

pfi_gen(); /* generate performance index of the firm */

cpi_gen(); /* generate consumer price index of the economy */

wri_gen(); /* generate wages rate index of the firm */

cf_gen(); /* generate net cash flow for each candidate project */

dm_pfi_gen(); /* DM simulate performance index of the firm */

dm_cpi_gen(); /* DM simulate consumer price index of the economy */

dm_wri_gen(); /* DM simulate wages rate index of the firm */

dm_cf_gen(); /* DM simulate net cash flow for each candidate project */

sum_z = 0.0;
sum_d = 0.0;

form_bundle();

npw();

}

/*==================================================================*/

read_seed()

{
int i;


for(i = 0 ; i < 28 ; i++)
fscanf(fp_seed,"%d",&seed[i]);

}
```

```
/* function read input from the files */

read_input()

{

/* declare input file pointer as well as input file name */

FILE *fp_in;
char f_in_name[MAX_STR];

/* get input file name for simulation run */

int i;

printf("Enter the input file name ");
gets(f_in_name);

fp_in = fopen(f_in_name,"r");

/* read title of each run */

fgets(title_1,MAX_STR,fp_in);
fgets(title_2,MAX_STR,fp_in);
fgets(title_3,MAX_STR,fp_in);

/* read data for all parameters used in the simulation run */

fscanf(fp_in,"%d",&horizon);

/* read economic indecies distribution */

fscanf(fp_in,"%f %f %f",&lb_pfi,&mo_pfi,&ub_pfi);
fscanf(fp_in,"%f %f %f",&lb_cpi,&mo_cpi,&ub_cpi);
fscanf(fp_in,"%f %f %f",&lb_wri,&mo_wri,&ub_wri);

/* read project life and number of project  distribution */

fscanf(fp_in,"%d %d %d",&lb_pro_num,&mo_pro_num,&ub_pro_num);
fscanf(fp_in,"%d %d %d",&lb_pro_life,&mo_pro_life,&ub_pro_life);

/* read value of K */

fscanf(fp_in,"%f %f %f",&lb_k,&mo_k,&ub_k);

/* read distribution of IRR */

fscanf(fp_in,"%f %f %f",&lb_mo_IRR,&mo_IRR,&ub_mo_IRR);
fscanf(fp_in,"%f %f",&lb_IRR,&ub_IRR);

fscanf(fp_in,"%f %f %f",&lb_gr,&mo_gr,&ub_gr); /* growth rate */

fscanf(fp_in,"%f",&start_bdgt); /* starting budget */
fscanf(fp_in,"%f %f",&a,&b); /* */
fscanf(fp_in,"%f %f",&lb_cap,&ub_cap);

fscanf(fp_in,"%f %f",&ll_ep,&ul_ep);
fscanf(fp_in,"%f %f",&lm_ep,&um_ep);
fscanf(fp_in,"%f %f",&lu_ep,&uu_ep);


fscanf(fp_in,"%f %f",&ll_ew,&ul_ew);
fscanf(fp_in,"%f %f",&lm_ew,&um_ew);
fscanf(fp_in,"%f %f",&lu_ew,&uu_ew);

fscanf(fp_in,"%f %f",&lb_q1,&ub_q1);
fscanf(fp_in,"%f %f",&lb_q2,&ub_q2);
fscanf(fp_in,"%f %f",&lb_q3,&ub_q3);
fscanf(fp_in,"%f %f",&lb_q4,&ub_q4);

fscanf(fp_in,"%d %d",&m1,&m2);
fscanf(fp_in,"%f %f %f",&MARR,&STRR,&HORR);
fscanf(fp_in,"%f %f",&au,&bu);
fscanf(fp_in,"%f",&equity);

for(i = 0;i < HORN; i++)
   fscanf(fp_in,"%f",&z_prior[i]);

for(i = 0;i < HORN; i++)
   fscanf(fp_in,"%f",&dvdend[i]);

}
```

```
/* function to write the data that are read in by " read_input" */

write_input()

{
int i;

printf("%s\n",title_1);
printf("%s\n",title_2);
printf("%s\n",title_3);

fprintf(fp_out,"%-50s\n",title_1);
fprintf(fp_out,"%-50s\n",title_2);
fprintf(fp_out,"%-50s\n",title_3);

/* read data for all parameters used in the simulation run */


printf("Length of study is : %d\n",horizon);

/* print economic indecies distribution */

printf("PFI : %f %f %f \n",lb_pfi,mo_pfi,ub_pfi);
printf("CPI : %f %f %f \n",lb_cpi,mo_cpi,ub_cpi);
printf("WRI : %f %f %f \n",lb_wri,mo_wri,ub_wri);

/* print project life and number of project  distribution */

printf("PROJECT # : %d %d %d \n",lb_pro_num,mo_pro_num,ub_pro_num);
printf("PROJECT LIFE : %d %d %d \n",lb_pro_life,mo_pro_life,ub_pro_life);

/* read value of K */

printf("K VALUE : %f %f %f\n",lb_k,mo_k,ub_k);

/* read distribution of IRR */

printf("MODE IRR : %f %f %f\n",lb_mo_IRR,mo_IRR,ub_mo_IRR);
printf("IRR : %f %f\n",lb_IRR,ub_IRR);

printf("GROWTH RATE : %f %f %f\n",lb_gr,mo_gr,ub_gr); /* growth rate */

printf("STARTING BUDGET : %f\n",start_bdgt); /* starting budget */
printf("VALUE OF a & b : %f %f\n",a,b); /* */

printf("CAPACITIES : %f %f\n",lb_cap,ub_cap);

printf("LOW EP : %f %f\n",ll_ep,ul_ep);
printf("MODE EP : %f %f\n",lm_ep,um_ep);
printf("UPPER EP : %f %f\n",lu_ep,uu_ep);


printf("LOW EW : %f %f\n",ll_ew,ul_ew);
printf("MODE EW : %f %f\n",lm_ew,um_ew);
printf("UPPER EW : %f %f\n",lu_ew,uu_ew);
printf("Q1 : %f %f\n",lb_q1,ub_q1);
printf("Q2 : %f %f\n",lb_q2,ub_q2);
printf("Q3 : %f %f\n",lb_q3,ub_q3);
printf("Q4 : %f %f\n",lb_q4,ub_q4);

printf("M1 & M2 : %d %d\n",m1,m2);
printf("MARR,STRR,HORR : %f %f %f\n",MARR,STRR,HORR);
printf("AU BU : %f %f\n",au,bu);
printf("EQUITY : %f\n",equity);

for(i = 0;i < HORN; i++)
   printf("Z_PRIOR (%d) : %f\n",i+1,z_prior[i]);
   printf("\n");

for(i = 0;i < HORN; i++)
   printf("DIVIDENT (%d) : %f\n",i+1,dvdend[i]);

}
```

```
/* function uniform distribution to calculate value of simulated
   parameters
*/

float uniform(lower,upper,seed)

float lower,upper;
unsigned seed;
{
        float x_value,rnd;

        srand(seed);

    rnd = rand();

        rnd = rnd/MAX_RVAL;


        x_value = lower + rnd*(upper - lower);

        temp_seed = rnd*MAX_RVAL;

        return(x_value);

}
```

```
/* function uniform distribution to calculate value of simulated
  parameters
*/

int iuniform(lower,upper,seed)

int lower,upper;
unsigned seed;
{
      int x_value;
      float rnd,result;

      srand (seed);
   rnd = rand();

      rnd = rnd/MAX_RVAL;

      result = lower + rnd*(upper - lower);
      x_value  = result;

      if(result - x_value > 0.5)
      {
        x_value = x_value + 1;

      temp_seed = rnd*MAX_RVAL;


        return(x_value);
       }
      else
      {

      temp_seed = rnd*MAX_RVAL;

      return(x_value);
      }
}
```

```
/* function triangular distribution to generate values that have
   this kind of distribution*/

#include<math.h>
#include<errno.h>

float triangle(lower,mode,upper,seed)

float lower,upper,mode;
unsigned seed;
{
float rnd,rnd_var_x,c_prime;


        c_prime = (mode - lower)/(upper - lower);

        srand(seed);
        rnd = rand();
        rnd = rnd/MAX_RVAL;

        if(rnd <= c_prime)
        {

           rnd_var_x = sqrt(c_prime*rnd);
        }
        else
        {

           rnd_var_x = 1 - sqrt((1 - c_prime)*(1 - rnd));
        }

        temp_seed = rnd*MAX_RVAL;

    rnd_var_x = lower + (upper - lower)*(rnd_var_x) ;
    return(rnd_var_x);

}
```

```
/* function triangular distribution to generate values that have
   this kind of distribution*/

int itriangle(lower,mode,upper,seed)

int lower,upper,mode;
unsigned seed;

{

float rnd,c_prime,rnd_var_x, result;
int rnd_var_num;

        c_prime = 1.*(mode - lower)/(upper - lower);

        srand(seed);
        rnd = rand();

        rnd = rnd/MAX_RVAL;

        if(rnd <= c_prime)
        {
           rnd_var_x = sqrt(c_prime*rnd);
        }
        else
        {
           rnd_var_x = 1 - sqrt((1 - c_prime)*(1 - rnd));
        }

   result = lower + (upper - lower)*(rnd_var_x) ;
   rnd_var_num = result;

        if(result - rnd_var_num > 0.5)
        {
         rnd_var_num = rnd_var_num + 1;

         temp_seed = rnd*MAX_RVAL;


         return(rnd_var_num);
         }
        else
        {

        temp_seed = rnd*MAX_RVAL;

   return(rnd_var_num);
    }
}
```

```
/* function that returns maximum value between a & b */

float max(a,b)

float a,b;

{

        if (a > b)
            return (a);
    else
            return(b);

}

/*===================================================================*/
/* function that returns maximum value between a & b */

int imax(a,b)

int a,b;

{
        if (a > b)
            return (a);
    else
            return(b);

}
```

```
/* function that returns minimum value between a & b */

float min(a,b)

float a,b;

{
        if (a < b)
            return (a);
    else
            return(b);

}

/*=================================================================*/
/* function that returns minimum value between a & b */

int imin(a,b)

int a,b;

{
        if (a < b)
            return (a);
    else
            return(b);

}
```

```
/* function to generate the performance index of the firm at each
period using the intial distribution function and the current
transition classes*/

pfi_gen()
{
int lmax,t;
float third,one_third,two_third;
float one_six,five_six,one_half;

        lmax = ub_pro_life;
        third = (ub_pfi - lb_pfi)/3 ;
        one_third = lb_pfi + third ;
        two_third = ub_pfi - third ;
        one_six = lb_pfi + third/2;
        five_six = ub_pfi - third/2;
        one_half = (lb_pfi + ub_pfi)/2;

/* set the level of pfi */


        pfi[0] = triangle(lb_pfi,mo_pfi,ub_pfi,seed[0]);
        seed[0] = temp_seed;

        if (pfi[0] < one_third)
                pfi_class[0] = 'l';
        else if(pfi[0] >= one_third && pfi[0] < two_third)
                 pfi_class[0] = 'm';
        else
                 pfi_class[0] = 'h';

        t = 0;

        while(t < horizon + lmax - 1)
        {


          {
            if (pfi_class[t] == 'l')
                   pfi[t+1] = triangle(lb_pfi,one_six,ub_pfi,seed[0]);
                else if(pfi_class[t] == 'm')
                   pfi[t+1] = triangle(lb_pfi,one_half,ub_pfi,seed[0]);
                else
                   pfi[t+1] = triangle(lb_pfi,five_six,ub_pfi,seed[0]);

          }
        seed[0] = temp_seed;

        /* generate next pfi for the next period which affected
           from t - 1 period (auto coreration
        */
          {
          if (pfi[t+1] < one_third)
               pfi_class[t+1] = 'l';
          else if(pfi[t+1] >= one_third && pfi[0] < two_third)
```

```
                        pfi_class[t+1] = 'm';
                else
                        pfi_class[t+1] = 'h';
                }

                t++;
        }
}
```

```
/* DM simulation function to generate the performance index of the
firm at each period using the intial distribution function and the
current transition classes*/

dm_pfi_gen()
{
int lmax,t,tr1;
float third;
float one_six,five_six,one_half;


     lmax = ub_pro_life;
     third = (ub_pfi - lb_pfi)/3 ;
     one_six = lb_pfi + third/2;
     five_six = ub_pfi - third/2;
     one_half = (lb_pfi + ub_pfi)/2;


/* set the level of pfi */


     tr1 = 0;
     while (tr1 < m1 )
     {
       dm_pfi[tr1][0] = triangle(lb_pfi,mo_pfi,ub_pfi,seed[1]);
       seed[1] = temp_seed;
       tr1++;
      }
     t = 0;

     while(t < horizon + lmax - 1)
     {
       tr1 = 0;
       while(tr1 < m1)
       {
        if (pfi_class[t] == 'l')
          dm_pfi[tr1][t+1] = triangle(lb_pfi,one_six,ub_pfi,seed[1]);
        else if(pfi_class[t] == 'm')
          dm_pfi[tr1][t+1] = triangle(lb_pfi,one_half,ub_pfi,seed[1]);
        else
          dm_pfi[tr1][t+1] = triangle(lb_pfi,five_six,ub_pfi,seed[1]);

           ++tr1;
        seed[1] = temp_seed;
       }
       ++t;
     }

}
```

```
/* function to generate the consumer price index  at each period using
the intial distribution function and the current transition classes*/

cpi_gen()
{
int lmax,t;
float third,one_third,two_third;
float one_six,five_six,one_half;

        lmax = ub_pro_life;

        third = (ub_cpi - lb_cpi)/3 ;
        one_third = lb_cpi + third ;
        two_third = ub_cpi - third ;
        one_six = lb_cpi + third/2;
        five_six = ub_cpi - third/2;
        one_half = (lb_cpi + ub_cpi)/2;

/* set the level of cpi */

        cpi[0] = triangle(lb_cpi,mo_cpi,ub_cpi,seed[2]);
          seed[2] = temp_seed;

        if (cpi[0] < one_third)
                cpi_class[0] = 'l';
        else if(cpi[0] >= one_third && cpi[0] < two_third)
                cpi_class[0] = 'm';
        else
                cpi_class[0] = 'h';

        t = 0;

        while(t < horizon + lmax - 1)
        {

          {
            if (cpi_class[t] == 'l')
                cpi[t+1] = triangle(lb_cpi,one_six,ub_cpi,seed[2]);
            else if(cpi_class[t] == 'm')
                cpi[t+1] = triangle(lb_cpi,one_half,ub_cpi,seed[2]);
            else
                cpi[t+1] = triangle(lb_cpi,five_six,ub_cpi,seed[2]);
          }
                seed[2] = temp_seed;

/*  generate next cpi for the next period which affected from t - 1
    period (auto coreration) */
          {
          if (cpi[t+1] < one_third)
              cpi_class[t+1] = 'l';
          else if(cpi[t+1] >= one_third && cpi[0] < two_third)
              cpi_class[t+1] = 'm';
          else
              cpi_class[t+1] = 'h';
          }
          t++;
        }
}
```

```
/* DM simulation function to generate the consumer price index  at
each period using the intial distribution function and the current
transition classes*/

dm_cpi_gen()
{
int lmax,t,tr1;
float third;
float one_six,five_six,one_half;

        lmax = ub_pro_life;

        third = (ub_cpi - lb_cpi)/3 ;
        one_six = lb_cpi + third/2;
        five_six = ub_cpi - third/2;
        one_half = (lb_cpi + ub_cpi)/2;

/* set the level of cpi */

        tr1 = 0;
        while (tr1 < m1 )
        {
           dm_cpi[tr1][0] = triangle(lb_cpi,mo_cpi,ub_cpi,seed[3]);
           seed[3] = temp_seed;
            tr1++;
         }

        t = 0;

        while(t < horizon + lmax - 1)
        {
          tr1 = 0;
          while( tr1 < m1 )
          {
            if (cpi_class[t] == 'l')
               dm_cpi[tr1][t+1] = triangle(lb_cpi,one_six,ub_cpi,seed[3]);
            else if(cpi_class[t] == 'm')
               dm_cpi[tr1][t+1] = triangle(lb_cpi,one_half,ub_cpi,seed[3]);
            else
               dm_cpi[tr1][t+1] = triangle(lb_cpi,five_six,ub_cpi,seed[3]);

                ++tr1;
            seed[3] = temp_seed;
          }

          t++;
        }
}
```

```
/* function to generate the wage rate index of the economics at each
period using the intial distribution function and the current
transition classes*/


wri_gen()
{

int lmax,t;
float third,one_third,two_third;
float one_six,five_six,one_half;

        lmax = ub_pro_life;
        third = (ub_wri - lb_wri)/3 ;
        one_third = lb_wri + third ;
        two_third = ub_wri - third ;
        one_six = lb_wri + third/2;
        five_six = ub_wri - third/2;
        one_half = (lb_wri + ub_wri)/2;

/* set the level of pfi */

        wri[0] = triangle(lb_wri,mo_wri,ub_wri,seed[4]);
          seed[4] = temp_seed;

        if (wri[0] < one_third)
                wri_class[0] = 'l';
        else if(wri[0] >= one_third && wri[0] < two_third)
                 wri_class[0] = 'm';
        else
                 wri_class[0] = 'h';

        t = 0;

        while(t < horizon + lmax - 1)
        {

          {
            if (wri_class[t] == 'l')
               wri[t+1] = triangle(lb_wri,one_six,ub_wri,seed[4]);
            else if(wri_class[t] == 'm')
               wri[t+1] = triangle(lb_wri,one_half,ub_wri,seed[4]);
            else
               wri[t+1] = triangle(lb_wri,five_six,ub_wri,seed[4]);
          }
            seed[4] = temp_seed;
/* generate next wri for the next period which affecte from t - 1 period
(auto coreration)*/

         {
          if (wri[t+1] < one_third)
                wri_class[t+1] = 'l';
          else if(wri[t+1] >= one_third && wri[0] < two_third)
                wri_class[t+1] = 'm';
          else
```

```
                wri_class[t+1] = 'h';
            }

          t++;
        }
}
```

```
/* DM simulation function to generate the wage rate index of the
economy at each period using the intial distribution function and
the current transition classes*/

dm_wri_gen()
{
int lmax,t,tr1;
float third;
float one_six,five_six,one_half;

        lmax = ub_pro_life;
        third = (ub_wri - lb_wri)/3 ;
        one_six = lb_wri + third/2;
        five_six = ub_wri - third/2;
        one_half = (lb_wri + ub_wri)/2;

/* set the level of pfi */

        tr1 = 0;
        while (tr1 < m1 )
        {
           dm_wri[tr1][0] = triangle(lb_wri,mo_wri,ub_wri,seed[5]);
           seed[5] = temp_seed;
            tr1++;
         }


        t = 0;

        while(t < horizon + lmax - 1)
        {
          tr1 = 0;
          while(tr1 < m1)
          {
            if (wri_class[t] == 'l')
               dm_wri[tr1][t+1] = triangle(lb_wri,one_six,ub_wri,seed[5]);
            else if(wri_class[t] == 'm')
               dm_wri[tr1][t+1] = triangle(lb_wri,one_half,ub_wri, seed[5]);
            else
               dm_wri[tr1][t+1] = triangle(lb_wri,five_six,ub_wri, seed[5]);

                tr1++;
                seed[5] = temp_seed;
          }
          t++;
        }
}
```

```
/* function to generate cash flow for the simulation run */

cf_gen()

{
        int n,t,j,i;
        float temp1,temp3,temp2,swp,ratio,labor;

        n = ub_pro_num;
        ant_bdgt[0]      = start_bdgt;

        t = 0;

        while(t < horizon )
        {
          pro_num[t] = itriangle(lb_pro_num,mo_pro_num,ub_pro_num,seed[6]);
          seed[6] = temp_seed;

          gr[t] = triangle(lb_gr,mo_gr,ub_gr,seed[7]);
          seed[7] = temp_seed;

          if(t > 0)
            ant_bdgt[t] = ant_bdgt[t-1]*(1. + gr[t]);

/* for each available project determine :the project life as a skewed-to-
the-left triangular distribution*/

          j = 0;

          while(j < pro_num[t])
          {
            pro_life[j][t] = itriangle(lb_pro_life,mo_pro_life,ub_pro_life,
                                seed[8]);
            seed[8] = temp_seed;

/* determine the project first cost (K) based on triangular distribution*/

            k[j][t] = ant_bdgt[t]*triangle(lb_k,mo_k,ub_k,seed[9]);
            seed[9] = temp_seed;

/* determine the IRR as a triangular distribution         */

            mod_IRR[j][t] = triangle(lb_mo_IRR,mo_IRR,ub_mo_IRR,seed[10]);
            seed[10] = temp_seed;

            IRR[j][t] = triangle(lb_IRR,mod_IRR[j][t],ub_IRR,seed[11]);
            seed[11] = temp_seed;

/* determine other parameters needed in the cash flow function */

            l_ep[j][t] = uniform(ll_ep,ul_ep,seed[12]);
            seed[12] = temp_seed;

            m_ep[j][t] = uniform(lm_ep,um_ep,seed[13]);
            seed[13] = temp_seed;
            u_ep[j][t] = uniform(lu_ep,uu_ep,seed[14]);
            seed[14] = temp_seed;

            l_ew[j][t] = uniform(ll_ew,ul_ew,seed[15]);
            seed[15] = temp_seed;

            m_ew[j][t] = uniform(lm_ew,um_ew,seed[16]);
            seed[16] = temp_seed;

            u_ew[j][t] = uniform(lu_ew,uu_ew,seed[17]);
            seed[17] = temp_seed;

            c[j][t] = uniform(lb_cap,ub_cap,seed[18]);
            seed[18] = temp_seed;

            q1[j][t] = uniform(lb_q1,ub_q1,seed[19]);
            seed[19] = temp_seed;

            q2[j][t] = uniform(lb_q2,ub_q2,seed[20]);
            seed[20] = temp_seed;

            q3[j][t] = uniform(lb_q3,ub_q3,seed[21]);
            seed[21] = temp_seed;

            q4[j][t] = uniform(lb_q4,ub_q4,seed[22]);
            seed[22] = temp_seed;

       {
            if(k[j][t] >= 85.0)
               cap[j][t] = uniform(27000.,35000.,seed[23]);
            else if(k[j][t] < 85.0 && k[j][t] >= 65.0)
               cap[j][t] = uniform(23000.,30000.,seed[23]);
            else if(k[j][t] < 65.0 && k[j][t] >= 45.0)
               cap[j][t] = uniform(16000.,21000.,seed[23]);
            else if(k[j][t] < 45.0 && k[j][t] >= 25.0)
               cap[j][t] = uniform(10000.,13000.,seed[23]);
            else if (k[j][t] < 25.0)
               cap[j][t] = uniform(4000.,6000.,seed[23]);
            else
                    ;
       }
            seed[23] = temp_seed;

/* calculate net operating cash flow based on Cubb-Douglass
   production function*/

             i = 0;

              while(i < pro_life[j][t])
              {

                ep[i][j][t] = triangle(l_ep[j][t],m_ep[j][t],u_ep[j][t],
                                            seed[24]);
                seed[24] = temp_seed;
```

```
         ew[i][j][t] = triangle(l_ew[j][t],m_ew[j][t],u_ew[j][t],
                                seed[25]);
         seed[25] = temp_seed;
         p[i][j][t]    =  q1[j][t]*pfi[i+t]  +  q2[j][t]*cpi[i+t]
                          + ep[i][j][t];
         w[i][j][t]    =  q3[j][t]*pfi[i+t]  +  q4[j][t]*wri[i+t]
                          + ew[i][j][t];

         temp1 = w[i][j][t]/(a*p[i][j][t]*c[j][t]*pow(k[j][t],b));

         l[i][j][t] = pow(temp1,(1./(a-1.)));

         temp3 = c[j][t]*pow(l[i][j][t],a)*pow(k[j][t],b);

         y[i][j][t] =  min(cap[j][t],temp3);

         if(y[i][j][t] == cap[j][t])
          l[i][j][t] = pow((cap[j][t]/(c[j][t]*pow(k[j][t],b))),
                           (1./a));

          income[i][j][t] = p[i][j][t]*y[i][j][t];

          outgo[i][j][t] = w[i][j][t]*l[i][j][t];

          raw_z[i][j][t] = income[i][j][t] - outgo[i][j][t];

          temp2 = pro_life[j][t];

          fz[i][j][t] = raw_z[i][j][t]*pow((1.+IRR[j][t]),
                         (temp2 - i - 1.));

          sum_fz[j][t] = sum_fz[j][t] + fz[i][j][t];

          i++;
      }

          fk[j][t] = k[j][t]*pow((1.+IRR[j][t]),temp2);
         scale_up[j][t] = fk[j][t]/sum_fz[j][t];

        j++;
        }
  t++;
}
t = 0;
while(t < horizon )
{
 j = 0;
 while(j < pro_num[t])
 {
  i = 0;
  while(i < pro_life[j][t])
  {
  z[i][j][t] = raw_z[i][j][t]*scale_up[j][t];

  i++;
```

```
         }
         j++;
        }
     t++;
    }
}
```

```
/*DM simulation function to generate cash flow based on DM simulation
  run */

dm_cf_gen()
{
int tr1,tr2,t,j,i;
float dummy_1,dummy_2,dummy_3,dummy_4;
float temp1;

t = 0;
while( t < horizon )
{
  tr1 = 0;
  while(tr1 < m1)
  {
   tr2 = 0;
   while(tr2 < m2)
   {
   j = 0;
   while(j < pro_num[t])
   {
   i = 0;
   while(i < pro_life[j][t])
   {
     dm_ep[tr2][i][j][t] = triangle(l_ep[j][t],m_ep[j][t],u_ep[j][t],
                                    seed[26]);
     seed[26] = temp_seed;
     dm_ew[tr2][i][j][t] = triangle(l_ew[j][t],m_ew[j][t],u_ew[j][t],
                                    seed[27]);
     seed[27] = temp_seed;

     dm_p[tr2][tr1][i][j][t] = q1[j][t]*dm_pfi[tr1][i+t]
                             + q2[j][t]*dm_cpi[tr1][i+t]
                             + dm_ep[tr2][i][j][t];

     dm_w[tr2][tr1][i][j][t] = q3[j][t]*dm_pfi[tr1][i+t]
                             + q4[j][t]*dm_wri[tr1][i+t]
                             + dm_ew[tr2][i][j][t];

     dummy_1 = pow(k[j][t],b);
     dummy_2 = a*dm_p[tr2][tr1][i][j][t]*c[j][t];
     dummy_3 = dm_w[tr2][tr1][i][j][t]/(dummy_1*dummy_2);
     dummy_4 = 1./(a - 1.);


     dm_l[tr2][tr1][i][j][t] = pow(dummy_3,dummy_4);

     dm_y[tr2][tr1][i][j][t] =  min(cap[j][t],c[j][t]
                                *pow(dm_l[tr2][tr1][i][j][t],a)
                                     *dummy_1);

     if(dm_y[tr2][tr1][i][j][t] == cap[j][t])
        dm_l[tr2][tr1][i][j][t] = pow((cap[j][t]/(c[j][t]
                                * pow(k[j][t],b))),(1.0/a));
     else
```

```
                    ;

     dm_income[tr2][tr1][i][j][t] = dm_p[tr2][tr1][i][j][t]
                                  * dm_y[tr2][tr1][i][j][t];

     dm_outgo[tr2][tr1][i][j][t]  = dm_w[tr2][tr1][i][j][t]
                                  * dm_l[tr2][tr1][i][j][t];

     dm_raw_z[tr2][tr1][i][j][t] = max(0.,(dm_income[tr2][tr1][i][j][t]
                                 - dm_outgo[tr2][tr1][i][j][t]));

     dm_IRR[tr2][tr1][j][t] = triangle(lb_IRR,mod_IRR[j][t],ub_IRR,
                                       seed[28]);
     seed[28] = temp_seed;
     temp1 = pro_life[j][t] ;

     fz_z[tr2][tr1][i][j][t] = dm_raw_z[tr2][tr1][i][j][t]
                             *pow((1.0+dm_IRR[tr2][tr1][j][t]),
                             (temp1 - i + 1.));

     sum_fz_z[tr2][tr1][j][t] = sum_fz_z[tr2][tr1][j][t]
                              + fz_z[tr2][tr1][i][j][t];

     dm_fk[tr2][tr1][j][t] = k[j][t]*pow((1.+ dm_IRR[tr2][tr1][j][t]),
                             temp1);

            i++;
   }

   if(sum_fz_z[tr2][tr1][j][t] == 0.0)
     s_scale_up[tr2][tr1][j][t] = 0.0 ;
   else
     s_scale_up[tr2][tr1][j][t] = dm_fk[tr2][tr1][j][t]
                                / sum_fz_z[tr2][tr1][j][t];

   j++;
   }
  tr2++;
  }
  tr1++;
 }
 t++;
}

 t = 0;
 while( t < horizon )
 {
  tr1 = 0;
  while(tr1 < m1)
  {
   tr2 = 0;
   while(tr2 < m2)
   {
   j = 0;
   while(j < pro_num[t])
   {
   i = 0;
   while(i < pro_life[j][t])
   {
     dm_z[tr2][tr1][i][j][t] = dm_raw_z[tr2][tr1][i][j][t]
                                * s_scale_up[tr2][tr1][j][t];

             i++;
   }

   j++;
   }
  tr2++;
  }.
  tr1++;
 }
 t++;
}

}
```

```
form_bundle()
{

 int t;

 t = 0;

 while(t < horizon )
 {
         form_1p_bundle(t);
         form_2p_bundle(t);
         form_3p_bundle(t);
         form_4p_bundle(t);

         t++;

 }
}


form_1p_bundle(t)

int t;
{
int j1;

j1 = 0;
while(j1 < pro_num[t])
{
  one_bun_k[j1][t] = k[j1][t];
  j1++;
}
}

form_2p_bundle(t)

int t;
{
int j1,j2;

j1 = 0;
while(j1 < pro_num[t] - 1)
{
  j2 = j1 + 1;

  while(j2 < pro_num[t])
  {
  two_bun_k[j1][j2-j1-1][t] = k[j1][t]+ k[j2][t];

   j2++;
  }
 j1++;
}
}
```

```
form_3p_bundle(t)

int t;
{
int j1,j2,j3;

j1 = 0;
while(j1 < pro_num[t]-2)
{
 j2 = j1 + 1 ;
 while(j2 < pro_num[t] - 1)
{
 j3 = j2 + 1 ;
 while(j3 < pro_num[t])
 {
  three_bun_k[j1][j2-j1-1][j3-j2-1][t] = k[j1][t] + k[j2][t] + k[j3][t];

   j3++;
  }
  j2++;
 }
 j1++;
}
}
```

```
form_4p_bundle(t)

int t;
{
int j1,j2,j3,j4;


j1 = 0;
while(j1 < pro_num[t]-3)
{
 j2 = j1+1;
 while(j2 < pro_num[t] - 2)
{
 j3 = j2+1;
 while(j3 < pro_num[t] - 1)
 {
  j4 = j3+1;
  while(j4 < pro_num[t])
  {
  four_bun_k[j1][j2-j1-1][j3-j2-1][j4-j3-1][t] = k[j1][t] + k[j2][t]
                                           + k[j3][t] + k[j4][t];


   j4++;
  }
   j3++;
 }
  j2++;
}
 j1++;
}
}
```

```
npw()
{
int t;
shot = 0;
t = 0;
while(t < horizon )
{
  update(t);
  one_npw(t);
  two_npw(t);
  three_npw(t);
  four_npw(t);

  one_pro_best(t);
  two_pro_best(t);
  three_pro_best(t);
  four_pro_best(t);



  printf(" PERIOD %d ",t+1);

  printf("\n\nBUDGET = $ %f \nK EQUTY = $ %f \nK DIVIDEND = $ %f K",
           budget[t],equity,dvdend[t]);

  printf("\nOUT OF %d CANDIDATE PROJECTS,",pro_num[t]);
  printf("\nWITH TOTAL COST OF $ %f  K",sum_k[t]);
  printf("\nAND A MINIMUM FIRST COST OF $ %f  K \n ",k_min[t]);


  printf(" Best _flag is %d : \n", best_flag);
  switch(best_flag)
  {
      case 0:
               printf("NONE ARE FEASIBLE, NO BUNDLE IS SELECTED\n");

               break;

      case 1:
                printf("BUNDLE %d IS SELECTED\n", j1_best);

               break;

      case 2:
               printf("BUNDLE %d AND %d IS SELECTED\n", j1_best,j2_best);

               break;

      case 3:
               printf("BUNDLE %d AND %d AND %d IS SELECTED\n", j1_best,
                                  j2_best,j3_best);

               break;
      case 4:
               printf("BUNDLE %d AND %d AND %d AND %d IS SELECTED\n",
                              j1_best,j2_best,j3_best,j4_best);

               break;


   }

    t++;
  }

 performance();

}
```

```
performance()

{
int t,i;
float measure;

measure = 0.0;
t = 0;

while( t < horizon)
{
  measure = measure - k_best[t]*pow((1.+STRR),(horizon - t - 1.));

  i = 0;
  while(i < ub_pro_life)
  {
    if(t+i <= horizon)
   {

      measure = measure + z_best[i][t]*pow((1.+STRR),(horizon-t-1.-i));

    }
    else
    {

      measure = measure + z_best[i][t]*pow((1.+HORR),(horizon-t-1.-i));

    }

    i++;
   }
   t++;
}
printf("\n\n\nNET FUTURE WORTH OF CASH FLOW AT THE HORIZON = $ %f K",
        measure);
}
```

```
one_npw(t)

 int t;
 {
 int j1,i,tr1,tr2;
 float sum;

 j1 = 0;
while(j1 < pro_num[t])
 {
   sum = 0;
   tr1 = 0;
   while(tr1 < m1)
   {
     tr2 = 0;
     while(tr2 < m2)
     {
      i = 0;
      while(i < ub_pro_life)
      {
        sum  = sum + dm_z[tr2][tr1][i][j1][t]/pow((1+MARR),(i+1));

        i++;
      }
      tr2++;
     }
    tr1++;
   }
   one_bun_worth[j1][t] = (-1)*one_bun_k[j1][t] + sum/(m1*m2);

   one_bun_k[j1][t],sum);

  j1++;
 }


}
```

```
two_npw(t)
int t;
{
int j1,j2;

j1 = 0;
while(j1 < pro_num[t]-1)
{
  j2 = j1+1;

  while(j2 < pro_num[t])
  {
   two_bun_worth[j1][j2-j1-1][t] = one_bun_worth[j1][t]
                                  + one_bun_worth[j2][t];

   j2++;
  }
  j1++;
 }
}
```

```
three_npw(t)
int t;
{
int j1,j2,j3;

j1 = 0;
while(j1 < pro_num[t]-2)
{
  j2 = j1+1;

  while(j2 < pro_num[t]-1)
  {
   j3 = j2+1;
   while( j3 < pro_num[t])
   {
     three_bun_worth[j1][j2-j1-1][j3-j2-1][t] = one_bun_worth[j1][t]
                                              + one_bun_worth[j2][t]
                                              + one_bun_worth[j3][t];

         j3++;
        }
   j2++;
  }
  j1++;
 }
}
```

```
four_npw(t)
int t;
{
int j1,j2,j3,j4;

j1 = 0;
while(j1 < pro_num[t]-3)
{
  j2 = j1+1;

  while(j2 < pro_num[t]-2)
  {
   j3 = j2+1;
   while( j3 < pro_num[t]-1)
   {
    j4 = j3+1;
    while(j4 < pro_num[t])
    {
    four_bun_worth[j1][j2-j1-1][j3-j2-1][j4-j3-1][t] = one_bun_worth[j1][t]
                                                    + one_bun_worth[j2][t]
                                                    + one_bun_worth[j3][t]
                                                    + one_bun_worth[j4][t];

     j4++;
    }
        j3++;
       }
  j2++;
 }
  j1++;
 }
}
```

```
update(t)

 int t;
 {
   int kkk;

   kkk = 0;

   if( t > 0)
     while(kkk < t-1)
     {
            sum_z = sum_z + z_best[t-kkk][kkk];

            kkk++;

          }
  else
     printf("\n");


  budget[t] = z_prior[t]+sum_z+shot*(1+STRR)-dvdend[t];
  equity = equity+z_prior[t]+sum_z+shot*STRR-dvdend[t];


  if(budget[t]+equity <= 0)
  {
    printf("\n\n****** THE FIRM IS BANKRUPT AT PERIOD  %d *******\n\n",t+1);


    performance();
        exit(1);
  }
  else
         ;
}
```

```
one_pro_best(t)

int t;
{
 int i,j1,nw,maxj1;

 best_flag = 0;
 best_worth = 0;
 j1_best = 0;
 shot = budget[t];
 j1 = 0;
 nw = 0;

 while(j1 < pro_num[t])
 {
   if((one_bun_k[j1][t] <= budget[t]) &&
      (one_bun_worth[j1][t] > 0))
    {
       nw = nw + 1;

       if(nw == 1)
       {
         maxw = one_bun_worth[j1][t];
         maxj1 = j1;

       }
       else if(one_bun_worth[j1][t] > maxw)
       {
           maxw = one_bun_wortn[j1][t];
           maxj1 = j1;
       }
       else
               ;
    }
  else
           ;

  j1++;
 }

  if(nw > 0)

  {

   best_flag = 1;
   best_worth = maxw;
   k_best[t] = one_bun_k[maxj1][t];
   shot = budget[t] - k_best[t];
   j1_best = maxj1;

   i = 0;

   while(i < ub_pro_life)
   {
    z_best[i][t] = z[i][maxj1][t];



    i++;

   }
  }
  else
          ;
}
```

```
two_pro_best(t)
int t;
{
 int i,j1,j2,nw,maxj1,maxj2;

 j1 = 0;
 nw = 0;

 while(j1 < pro_num[t]-1)
 {
  j2 = j1+1;

  while(j2 < pro_num[t])
  {
   if(two_bun_k[j1][j2-j1-1][t] <= budget[t] &&
      two_bun_worth[j1][j2-j1-1][t] > 0
      && two_bun_worth[j1][j2-j1-1][t] > maxw)
      {
        nw = nw + 1;
        if(nw == 1)
        {
          maxw = two_bun_worth[j1][j2-j1-1][t];
          maxj1 = j1;
          maxj2 = j2;
        }
        else if(two_bun_worth[j1][j2-j1-1][t] > maxw)
        {
            maxw = two_bun_worth[j1][j2-j1-1][t];
            maxj1 = j1;
            maxj2 = j2;
        }
        else
            ;
       }
    else
          ;
         j2++;
   }
   j1++;
  }

  if(nw > 0)
  {
    best_flag = 2;
    best_worth = maxw;
    k_best[t] = two_bun_k[maxj1][maxj2-maxj1-1][t];
    shot = budget[t] - k_best[t];
    j1_best = maxj1;
    j2_best = maxj2 ;

    i = 0;

    while(i < ub_pro_life)
    {
     z_best[i][t] = z[i][maxj1][t]+z[i][maxj2][t];



     i++;
    }
   }
   else
          ;
 }
```

```
three_pro_best(t)
int t;
{
 int i,j1,j2,j3,nw,maxj1,maxj2,maxj3;

 j1 = 0;
 nw = 0;

 while(j1 < pro_num[t]-2)
 {
  j2 = j1+1;

  while(j2 < pro_num[t]-1)
  {
   j3 = j2+1;
   while(j3 < pro_num[t])
   {
   if(three_bun_k[j1][j2-j1-1][j3-j2-1][t] <= budget[t] &&
      three_bun_worth[j1][j2-j1-1][j3-j2-1][t] > 0
      && three_bun_worth[j1][j2-j1-1][j3-j2-1][t] > maxw)
      {
        nw = nw + 1;
        if(nw == 1)
        {
          maxw = three_bun_worth[j1][j2-j1-1][j3-j2-1][t];
          maxj1 = j1;
          maxj2 = j2;
          maxj3 = j3;
        }
        else if(three_bun_wortn[j1][j2-j1-1][j3-j2-1][t] > maxw)
        {
            maxw = three_bun_worth[j1][j2-j1-1][j3-j2-1][t];
            maxj1 = j1;
            maxj2 = j2;
            maxj3 = j3;

        }
        else
             ;
      }
     else
           ;

     j3++;
    }
         j2++;
   }
    j1++;
  }

  if(nw > 0)
  {
    best_flag = 3;
    best_worth = maxw;
    k_best[t] = three_bun_k[maxj1][maxj2-maxj1-1][maxj3-maxj2-1][t];
   shot = budget[t] - k_best[t];
   j1_best = maxj1;
   j2_best = maxj2 ;
   j3_best = maxj3 ;

   i = 0;

   while(i < ub_pro_life)
   {
    z_best[i][t] = z[i][maxj1][t]+z[i][maxj2][t]+z[i][maxj3][t];
    i++;
   }
  }
  else
      ;
}
```

```
four_pro_best(t)
int t;
{
 int i,j1,j2,j3,j4;
 int nw,maxj1,maxj2,maxj3,maxj4;

 j1 = 0;
 nw = 0;

 while(j1 < pro_num[t]-3)
 {
  j2 = j1+1;
  while(j2 < pro_num[t]-2)
  {
   j3 = j2+1;
   while(j3 < pro_num[t]-1)
   {
    j4 = j3+1;
    while(j4 < pro_num[t])
    {
      if(four_bun_k[j1][j2-j1-1][j3-j2-1][j4-j3-1][t] <= budget[t] &&
         four_bun_worth[j1][j2-j1-1][j3-j2-1][j4-j3-1][t] > 0
        && four_bun_worth[j1][j2-j1-1][j3-j2-1][j4-j3-1][t] > maxw)
      {
        nw = nw + 1;
        if(nw == 1)
        {
          maxw = four_bun_worth[j1][j2-j1-1][j3-j2-1][j4-j3-1][t];
          maxj1 = j1;
          maxj2 = j2;
          maxj3 = j3;
          maxj4 = j4;
        }
        else if(four_bun_worth[j1][j2-j1-1][j3-j2-1][j4-j3-1][t] > maxw)
        {
            maxw = four_bun_worth[j1][j2-j1-1][j3-j2-1][j4-j3-1][t];
            maxj1 = j1;
            maxj2 = j2;
            maxj3 = j3;
            maxj4 = j4;
          }
        }
        else
           ;
        j4++;
    }

     j3++;
   }
        j2++;
  }
   j1++;
 }

 if(nw > 0)
{
 best_flag = 4;
 best_worth = maxw;

 k_best[t]
    = four_bun_k[maxj1][maxj2-maxj1-1][maxj3-maxj2-1][maxj4-maxj3-1][t];

 shot = budget[t] - k_best[t];
 j1_best = maxj1;
 j2_best = maxj2 ;
 j3_best = maxj3 ;
 j4_best = maxj4 ;

 i = 0;

 while(i < up_pro_life)
 {
   z_best[i][t] = z[i][maxj1][t]+z[i][maxj2][t]+z[i][maxj3][t]
                          + z[i][maxj4][t];
   i++;
 }
}
 else
       ;
}
```

ชุดตาราง เลขสุ่มที่ใช้ในการทดลอง

ชุดตาราง เลขสุ่มที่ใช้ในการทดลอง

ชุดที่ 1

3513	6976	9847	1722	3874	2883	882
4311	2741	2020	5675	2501	7238	945
7257	8101	7066	4632	998	5356	7526
9890	2642	7480	6822	2251	5741	4166

ชุดที่ 2

5897	2726	5765	5434	1213	8665	4055
4334	928	9783	5313	5105	4112	6666
7778	9289	6567	5664	1641	8621	7013
2650	6793	5587	4368	7740	2544	1043

ชุดที่ 3

283	3170	7628	7272	3355	9564	3843
5352	8223	742	4834	4524	3463	3880
4256	2927	2555	2608	9328	8859	4052
4612	4477	962	4846	7966	4678	3327

ชุดที่ 4

1810	9342	1325	6555	25	2252	2434
6253	7228	9661	7136	526	1247	4679
1275	7556	5297	2749	7881	3774	3275
6238	5269	5173	1169	1644	9747	6867

ชุดที่ 5

108	3049	384	2344	3781	4089	7772
7509	6337	884	3558	9190	1596	6919
1270	4543	5186	1993	4481	9660	5234
8405	7590	4618	6531	515	3757	7600

ชุดที่ 6

7370	2243	5086	6896	7283	2662	9636
4353	627	8749	3644	3854	4503	9998
743	2297	6403	7840	4341	5716	979
3730	6510	6063	7602	5202	4748	8497

ชุดที่ 7

7311	4717	2397	4918	2388	9291	6603
7564	6778	7857	9951	2080	3808	6404
5648	6083	3047	4767	7413	3172	6935
9563	4330	6846	8196	6861	9952	3075

ชุดที่ 8

9475	1186	6565	6196	5667	9175	9497
1153	1679	4871	9776	6306	7834	4311
8221	6077	2359	6623	753	5485	8476
7398	9231	5720	8918	6164	3158	5779

ชุดที่ 9

7541	4439	698	2361	1589	3767	4576
8712	5744	7522	4733	199	5988	3214
829	4491	7	7513	7581	7277	4348
8252	9448	4063	4518	7392	1571	2418

ชุดที่ 10

3423	3687	4977	7234	9474	3974	1460
3333	9261	5075	3013	5569	5294	463
5187	2337	276	1228	9424	5440	6903
1533	537	2381	5955	382	6711	2158

ข้อมูลที่ใช้ในการทดลอง
กรณี "ผลตอบแทนเพิ่มขึ้นในอัตราส่วนเพิ่มขึ้น"

เงื่อนไข : "รุ่งเรืองและความเสี่ยงสูง/ เงินทุนสูง

```
10
1 1.5 1 1.61
1 1.5 1 1.61
0.93 1 1.33
3 4 4
3 5 6
0.25 0.35 0.4
0.22 0.34 0.38
-0.05 0.39
0.075 0.08 0.085
48
0.36 0.84
1800 2200
0.00114 0.00133
0.0018 0.002
0.002 0.0022
6.75 7
7 8
9.75 10.5
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
10 10 15 15 15 20 20 20 20 25 25
```

เงื่อนไข : "รุ่งเรืองและความเสี่ยงสูง" / แรงงานสูง

```
10
1 1.5 1 1.61
1 1.5 1 1.61
0.93 1 1.33
3 4 4
3 5 6
0.25 0.35 0.4
0.22 0.34 0.38
-0.05 0.39
0.075 0.08 0.085
48
0.84 0.36
1800 2200
0.00114 0.00133
0.0018 0.002
0.002 0.0022
6.75 7
7 8
9.75 10.5
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
10 10 15 15 15 20 20 20 20 25 25
```

เงื่อนไข : "รุ่งเรืองและความเสี่ยงต่ำ" / เงินทุนสูง

```
10
1.35 1.5  1.61
1.35 1.5  1.61
0.93 1 1.1
3 4 4
3 5 6
0.25 0.35 0.4
0.30 0.34 0.38
0.05 0.39
0.075 0.08 0.085
48
0.36 0.84
1800 2200
0.0016 0.0018
0.0018 0.002
0.002 0.0022
6.75 7
7 8
8 8.6
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
10 10 15 15 15 20 20 20 20 25 25
```

เงื่อนไข : "รุ่งเรืองและความเสี่ยงต่ำ" / แรงงานสูง

```
10
1.35 1.5  1.61
1.35 1.5  1.61
0.93 1 1.1
3 4 4
3 5 6
0.25 0.35 0.4
0.30 0.34 0.38
0.05 0.39
0.075 0.08 0.085
48
0.84 0.36
1800 2200
0.0016 0.0018
0.0018 0.002
0.002 0.0022
6.75 7
7 8
8 8.6
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
10 10 15 15 15 20 20 20 20 25 25
```

เงื่อนไข : "ถดถอยและความเสี่ยงสูง" / เงินทุนสูง

```
10
0.53 0.8 0.86
0.53 0.8 0.86
1.11 1.2 1.56
3 4 4
3 5 6
0.25 0.35 0.4
0.1 0.16 0.18
-0.05 0.19
0.075 0.08 0.085
48
0.36 0.84
1800 2200
0.000372 0.000434
0.000590 0.000650
0.000663 0.000682
11.25 11.625
12.0 13.0
16.25 17.5
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
4 4 4 4 4 4 4 4 4 4 4
```

เงื่อนไข : "ถดถอยและความเสี่ยงสูง" / แรงงานสูง

```
10
0.53 0.8 0.86
0.53 0.8 0.86
1.11 1.2 1.56
3 4 4
3 5 6
0.25 0.35 0.4
0.1 0.16 0.18
-0.05 0.19
0.075 0.08 0.085
48
0.84 0.36
1800 2200
0.000372 0.000434
0.000590 0.000650
0.000663 0.000682
11.25 11.625
12.0 13.0
16.25 17.5
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
4 4 4 4 4 4 4 4 4 4 4
```

เงื่อนไข : "ถดถอยและความเสี่ยงต่ำ" / เงินทุนสูง

```
10
0.72 0.80 0.86
0.72 0.80 0.86
1.11 1.20 1.32
3 4 4
3 5 6
0.25 0.35 0.4
0.14 0.16 0.18
0.05 0.19
0.075 0.08 0.085
48
0.36 0.84
1800 2200
0.000527 0.000589
0.000590 0.000650
0.000663 0.000682
11.25 11.625
12.  13.
13.125 14.375
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0
4 4 4 4 4 4 4 4 4 4
```

เงื่อนไข : "ถดถอยและความเสี่ยงต่ำ" / แรงงานสูง

```
10
0.72 0.80 0.86
0.72 0.80 0.86
1.11 1.20 1.32
3 4 4
3 5 6
0.25 0.35 0.4
0.14 0.16 0.18
0.05 0.19
0.075 0.08 0.085
48
0.84 0.36
1800 2200
0.000527 0.000589
0.000590 0.000650
0.000663 0.000682
11.25 11.625
12.  13.
13.125 14.375
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
4 4 4 4 4 4 4 4 4 4 4
```

ข้อมูลที่ใช้ในการทดลอง
กรณี "ผลตอบแทนเพิ่มขึ้นในอัตราส่วนคงที่"

เงื่อนไข : "รุ่งเรืองและความเสี่ยงสูง" / เงินทุนสูง

```
10
1 1.5 1 1.61
1 1.5 1 1.61
0.93 1 1.33
3 4 4
3 5 6
0.25 0.35 0.4
0.22 0.34 0.38
-0.05 0.39
0.075 0.08 0.085
48
0.3 0.7
1800 2200
0.00114 0.00133
0.0018 0.002
0.002 0.0022
6.75 7
7 8
9.75 10.5
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
10 10 15 15 15 20 20 20 20 25 25
```

เงื่อนไข : "รุ่งเรืองและความเสี่ยงสูง" / แรงงานสูง

```
10
1 1.5 1 1.61
1 1.5 1 1.61
0.93 1 1.33
3 4 4
3 5 6
0.25 0.35 0.4
0.22 0.34 0.38
-0.05 0.39
0.075 0.08 0.085
48
0.7 0.3
1800 2200
0.00114 0.00133
0.0018 0.002
0.002 0.0022
6.75 7
7 8
9.75 10.5
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
10 10 15 15 15 20 20 20 20 25 25
```

เงือนไข : "รุ่งเรืองและความเสี่ยงต่ำ" / เงินทุนสูง

```
10
1.35 1.5  1.61
1.35 1.5  1.61
0.93 1 1.1
3 4 4
3 5 6
0.25 0.35 0.4
0.30 0.34 0.38
0.05 0.39
0.075 0.08 0.085
48
0.3 0.7
1800 2200
0.0016 0.0018
0.0018 0.002
0.002 0.0022
6.75 7
7 8
8 8.6
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
10 10 15 15 15 20 20 20 20 25 25
```

เงือนไข : "รุ่งเรืองและความเสี่ยงต่ำ" / แรงงานสูง

```
10
1.35 1.5  1.61
1.35 1.5  1.61
0.93 1 1.1
3 4 4
3 5 6
0.25 0.35 0.4
0.30 0.34 0.38
0.05 0.39
0.075 0.08 0.085
48
0.7  0.3
1800 2200
0.0016 0.0018
0.0018 0.002
0.002 0.0022
6.75 7
7 8
8 8.6
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
10 10 15 15 15 20 20 20 20 25 25
```

เงื่อนไข : "ถดถอยและความเสี่ยงสูง" / เงินทุนสูง

```
10
0.53 0.8 0.86
0.53 0.8 0.86
1.11 1.2 1.56
3 4 4
3 5 6
0.25 0.35 0.4
0.1 0.16 0.18
-0.05 0.19
0.075 0.08 0.085
48
0.3  0.7
1800 2200
0.000372 0.000434
0.000590 0.000650
0.000663 0.000682
11.25 11.625
12.0 13.0
16.25 17.5
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
4 4 4 4 4 4 4 4 4 4 4
```

เงื่อนไข : "ถดถอยและความเสี่ยงสูง" / แรงงานสูง

```
10
0.53 0.8 0.86
0.53 0.8 0.86
1.11 1.2 1.56
3 4 4
3 5 6
0.25 0.35 0.4
0.1 0.16 0.18
-0.05 0.19
0.075 0.08 0.085
48
0.7  0.3
1800 2200
0.000372 0.000434
0.000590 0.000650
0.000663 0.000682
11.25 11.625
12.0 13.0
16.25 17.5
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
4 4 4 4 4 4 4 4 4 4 4
```

เงื่อนไข : "ถดถอยและความเสี่ยงต่ำ" / เงินทุนสูง

```
10
0.72 0.80 0.86
0.72 0.80 0.86
1.11 1.20 1.32
3 4 4
3 5 6
0.25 0.35 0.4
0.14 0.16 0.18
0.05 0.19
0.075 0.08 0.085
48
0.3  0.7
1800 2200
0.000527 0.000589
0.000590 0.000650
0.000663 0.000682
11.25 11.625
12.  13.
13.125 14.375
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
4 4 4 4 4 4 4 4 4 4 4
```

เงื่อนไข : "ถดถอยและความเสี่ยงสูง" / แรงงานสูง

```
10
0.72 0.80 0.86
0.72 0.80 0.86
1.11 1.20 1.32
3 4 4
3 5 6
0.25 0.35 0.4
0.14 0.16 0.18
0.05 0.19
0.075 0.08 0.085
48
0.7  0.3
1800 2200
0.000527 0.000589
0.000590 0.000650
0.000663 0.000682
11.25 11.625
12.  13.
13.125 14.375
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
4 4 4 4 4 4 4 4 4 4 4
```

ข้อมูลที่ใช้ในการทดลอง
กรณี "ผลตอบแทนเพิ่มขึ้นในอัตราส่วนลดลง"

เงื่อนไข : "รุ่งเรืองและความเสี่ยงสูง" / เงินทุนสูง

```
10
1 1.5 1 1.61
1 1.5 1 1.61
0.93 1 1.33
3 4 4
3 5 6
0.25 0.35 0.4
0.22 0.34 0.38
-0.05 0.39
0.075 0.08 0.085
48
0.24 0.56
1800 2200
0.00114 0.00133
0.0018 0.002
0.002 0.0022
6.75 7
7 8
9.75 10.5
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
10 10 15 15 15 20 20 20 20 25 25
```

เงื่อนไข : "รุ่งเรืองและความเสี่ยงสูง" / แรงงานสูง

```
10
1 1.5 1 1.61
1 1.5 1 1.61
0.93 1 1.33
3 4 4
3 5 6
0.25 0.35 0.4
0.22 0.34 0.38
-0.05 0.39
0.075 0.08 0.085
48
0.56 0.24
1800 2200
0.00114 0.00133
0.0018 0.002
0.002 0.0022
6.75 7
7 8
9.75 10.5
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
10 10 15 15 15 20 20 20 20 25 25
```

เงือนไข : "รุ่งเรืองและความเสี่ยงต่ำ" / เงินทุนสูง

```
10
1.35 1.5  1.61
1.35 1.5  1.61
0.93 1 1.1
3 4 4
3 5 6
0.25 0.35 0.4
0.30 0.34 0.38
0.05 0.39
0.075 0.08 0.085
48
0.24 0.56
1800 2200
0.0016 0.0018
0.0018 0.002
0.002 0.0022
6.75 7
7 8
8 8.6
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
10 10 15 15 15 20 20 20 20 25 25
```

เงือนไข : "รุ่งเรืองและความเสี่ยงต่ำ"/ แรงงานสูง

```
10
1.35 1.5  1.61
1.35 1.5  1.61
0.93 1 1.1
3 4 4
3 5 6
0.25 0.35 0.4
0.30 0.34 0.38
0.05 0.39
0.075 0.08 0.085
48
0.56  0.24
1800 2200
0.0016 0.0018
0.0018 0.002
0.002 0.0022
6.75 7
7 8
8 8.6
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
10 10 15 15 15 20 20 20 20 25 25
```

เงื่อนไข : "ถดถอยและความเสี่ยงสูง" / เงินทุนสูง

```
10
0.53 0.8 0.86
0.53 0.8 0.86
1.11 1.2 1.56
3 4 4
3 5 6
0.25 0.35 0.4
0.1 0.16 0.18
-0.05 0.19
0.075 0.08 0.085
48
0.24  0.56
1800 2200
0.000372 0.000434
0.000590 0.000650
0.000663 0.000682
11.25 11.625
12.0 13.0
16.25 17.5
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
4 4 4 4 4 4 4 4 4 4 4
```

เงื่อนไข : "ถดถอยและความเสี่ยงสูง" / แรงงานสูง

```
10
0.53 0.8 0.86
0.53 0.8 0.86
1.11 1.2 1.56
3 4 4
3 5 6
0.25 0.35 0.4
0.1 0.16 0.18
-0.05 0.19
0.075 0.08 0.085
48
0.56  0.24
1800 2200
0.000372 0.000434
0.000590 0.000650
0.000663 0.000682
11.25 11.625
12.0 13.0
16.25 17.5
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
4 4 4 4 4 4 4 4 4 4 4
```

เงื่อนไข : "ถดถอยและความเสี่ยงต่ำ" / เงินทุนสูง

```
10
0.72 0.80 0.86
0.72 0.80 0.86
1.11 1.20 1.32
3 4 4
3 5 6
0.25 0.35 0.4
0.14 0.16 0.18
0.05 0.19
0.075 0.08 0.085
48
0.24  0.59
1800 2200
0.000527 0.000589
0.000590 0.000650
0.000663 0.000682
11.25 11.625
12.  13.
13.125 14.375
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
4 4 4 4 4 4 4 4 4 4 4
```

เงื่อนไข : "ถดถอยและความเสี่ยงต่ำ" / แรงงานสูง

```
10
0.72 0.80 0.86
0.72 0.80 0.86
1.11 1.20 1.32
3 4 4
3 5 6
0.25 0.35 0.4
0.14 0.16 0.18
0.05 0.19
0.075 0.08 0.085
48
0.56  0.24
1800 2200
0.000527 0.000589
0.000590 0.000650
0.000663 0.000682
11.25 11.625
12.  13.
13.125 14.375
0.0016 0.002
0.0021 0.0025
-6 -2
8 12
3 3
0.08 0.08 0.08
100 4
20
30 25 0 0 0 0 0 0 0 0 0
4 4 4 4 4 4 4 4 4 4 4
```

ประวัติผู้เขียน

นาย กิตติรัตน์ ลีละหุต เกิดวันที่ 16 มิถุนายน พ.ศ. 2501 ที่กรุงเทพมหานคร ฯ สำเร็จการศึกษาปริญญาตรีวิทยาศาสตร์บัณฑิต สาขาวิศวกรรมอุตสาหการ ภาควิชา วิศวกรรมระบบและอุตสาหการ มหาวิทยาลัย อริโซนา สหรัฐอเมริกา เมื่อปีการศึกษา 1987 เข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตร์ (ภาคนอกเวลาราชการ) สาขาวิศวกรรมอุตสาหการ ที่จุฬาลงกรณ์มหาวิทยาลัย เมื่อ พ.ศ. 2532 ปัจจุบันมีตำแหน่งเป็น ผู้จัดการแผนกวางแผน บริษัท วอร์เนอ-แลมเบิท (ประเทศไทย) จำกัด