Matematički fakultet u Beogradu

Domaći rad iz predmeta matematičko programiranje i optimizacija

Problem br:4

Ime studenta:
Nikola Stanojević, 1064/2012

Profesor:
Zorica Stanimirović

1 Opis problema

U pitanju je problem rasporedjivanja fabrika (capacitated facility location problem) koji možemo opisati na sledeći način:

Ako sa I obeležimo niz potencijalnih lokacija za izgradnju fabrika, od kojih je svaka sa kapacitetom Q, i ako sa J obeležimo skup klijenata ili zahteva. Za svaki zahtev $j \in J$ neka d_j odredjuje odgovarajući zahtev. Takodje neka f_i predstavlja fiksnu cenu uspostavljanja fabrike na lokaciji i. Pretpostavimo da nekoliko modula, različitih veličina, moze biti u okviru bilo koje veze fabrika-klijent. Označimo sa $\{1, \ldots, L\}$ niz različitih tipova modula. Neka C^l predstavlja kapacitet modula tipa l i neka je g^l odgovarajuća cena ($l \in \{1, \ldots, L\}$). Konačno sa c_{ij} ćemo označiti jedinstvenu cenu distribucije na vezi (i, j).
Na kraju potrebno je da garantujemo da su svi zahtevi zadovoljeni uz minimalnu cenu (cena rešenja je zbir cena uspostavljanja fabrike i fiksnih i promenljivih cena veza izmedju klijenata i fabrika).

Kako bi smo formulisali ovaj lokacijski problem potrebne su nam sledeće odlučujuce promenljive:
(i) x_{ij} - celobrojne promenljive koje predstavljaju broj jedinica količine robe koju fabrika i šalje klijentu j
(ii) u_{ij}^l - celobrojne promenljive koje predstavljaju broj modula tipa l koji su instalirani na vezi (i, j)
(iii) y_i - binarne promenljive koje pokazuju da li je fabrika uspostavljena na lokaciji i ($i \in I$).

$$y_i = \begin{cases} 1, & ako\ je\ fabrika\ uspostavljena\ na\ lokaciji\ i \\ 0, & inace \end{cases}$$

2 Matematička formulacija problema

$$Min \sum_{i=0}^{I} f_i y_i + \sum_{i=0}^{I} \sum_{j=0}^{J} \sum_{l=1}^{L} g^l u_{ij}^l + \sum_{i=0}^{I} \sum_{j=0}^{J} c_{ij} x_{ij} \tag{1}$$

$$ograniceja: \quad \sum_{i=0}^{I} x_{ij} = d_j \qquad j \in J \tag{2}$$

$$\sum_{j=0}^{J} x_{ij} \leq Q y_i \qquad i \in I \tag{3}$$

$$x_{ij} \leq \sum_{l=1}^{L} C^l u_{ij}^l \qquad i \in I, j \in J \tag{4}$$

$$u_{ij}^l \geq 0 \ \in Z \qquad i \in I, j \in J, l \in \{1, \ldots, L\} \tag{5}$$

$$x_{ij} \geq 0 \ \in Z \qquad i \in I, j \in J \tag{6}$$

$$y_i \in \{0, 1\} \qquad i \in I \tag{7}$$

Značenje uslova:

Funkcija cilja **(1)** minimizuje ukupnu cenu, koja uključuje cenu uspostavljanja fabrike i cene koje se odnose na veze izmedju klijenata i fabrika.
Ograničenje **(2)** garantuje da će svi zahtevi biti zadovoljeni.
Ograničenje **(3)** predstavlja ograničenje kapaciteta fabrika.
Ograničenje **(4)** obezbedjuje da moduli instalirani na vezi (i, j) imaju dovoljan kapacitet da bi podržali saobraćaj na toj vezi.
Ograničenja **(5), (6)** i **(7)** definišu domen korišćenih promenljivih.

3 Instance, opis instanci i primer jedne manje instance

Program za generisanje instanci radi tako što korisnik unosi vrednosti veličine skupova I i J, vrednost veličine skupa tipova modula L i konstantu Q i na osnovu vrednosti I, J i L dobijamo dimenzije matrica za generisanje koeficijenata $f_i, g_{ij}^l, c_{ij}, d_j, C^l$. Sve vrednosti se upisuju u tekstualni fajl, koji se otvara na samom početku programa za generisanje instanci, koji se potom koristi u glavnom programu kako bi se učitala instanca i dobio optimalni rezultat i vreme izvršavanja programa za datu instancu. Vrednosti u matricama su generisane korišćenjem funkcije rand(), tako da su vrednosti nasumično izgenerisane i ograničene na interval od 1 do 100. Na sam početak fajla upisujemo vrednosti koje je uneo korisnik za I, J, L i Q, zatim generisemo nasumične vrednosti nizova (i praznine izmedju vrednosti) dimenzija I, I*J*L, I*J, J, L koristeci for ciklus i random funkciju za generisanje nasumičnih brojeva kako bi dobili vrednosti za koeficijente $f_i, g_{ij}^l, c_{ij}, d_j, C^l$. Sve vrednosti nizova upisujemo u nastavku tekstualnog fajla. Vrednosti različitih nizova razdvajeni su novim redom radi bolje preglednosti. Na kraju programa zatvaramo tekstualni fajl.

Instance su generisane koristeći sledeci kod:

```
#include <iostream>
#include <fstream>
#include <ctime>
#include <cstdlib>
#include <stdio.h>
using namespace std;

int main(){

  // sve vrednosti upisujemo u datoteku "ulaz1.txt"
  ofstream myfile;
  myfile.open ("ulaz1.txt");

  srand(time(0));
  int I, J, L, Q;
      int i, min=1, max=100;

  printf("Unesite vrednost za I=");
  scanf("%d", &I);
  printf("Unesite vrednost za J=");
  scanf("%d", &J);
  printf("Unesite vrednost za L=");
  scanf("%d", &L);
```

```
printf("Unesite vrednost za Q=");
scanf("%d", &Q);
 // unosimo vrednosti I, J, L i Q u instancu
 myfile << I << " " << J << " " << L << " " << Q << "\n\n";

// I za f_i, generisemo vrednosti f_i nizom dimenzije I
 int* arr1 = (int*) malloc(I * sizeof(int));

   for(i = 0; i < I; i++)
     // nasumicni brojevi od min do max:
     arr1[i] = rand() % (max-min) + min + 1;

   for(i = 0; i < I; i++)
     myfile << arr1[i] << " ";

 myfile << "\n\n";

// I * J * L za g_ij^l, generisemo vrednosti g_ij^l nizom dimenzije I * J * L
 int* arr2 = (int*) malloc((I * J * L) * sizeof(int));

  for(i = 0; i < I*J*L; i++)
    // nasumicni brojevi od min do max:
    arr2[i] = rand() % (max-min) + min + 1;

  for(i = 0; i < I*J*L; i++)
    myfile << arr2[i] << " ";

 myfile << "\n\n";

// I * J za c_ij, generisemo vrednosti c_ij nizom dimenzije I * J
 int* arr3 = (int*) malloc((I * J) * sizeof(int));

  for(i = 0; i < I * J; i++)
      // nasumicni brojevi od min do max:
    arr3[i] = rand() % (max-min) + min + 1;

  for(i = 0; i < I * J; i++)
    myfile << arr3[i] << " ";

 myfile << "\n\n";

// J za d_j, generisemo vrednosti d_j nizom dimenzije J
 int* arr4 = (int*) malloc(J * sizeof(int));

  for(i = 0; i < J; i++)
    // nasumicni brojevi od min do max:
    arr4[i] = rand() % (max-min) + min + 1;

  for(i = 0; i < J; i++)
      myfile << arr4[i] << " ";

 myfile << "\n\n";

// L za C_l, generisemo vrednosti C_l nizom dimenzije L
 int* arr5 = (int*) malloc(L * sizeof(int));
```

```
    for(i = 0; i < L; i++)
      // nasumicni brojevi od min do max:
      arr5[i] = rand() % (max-min) + min + 1;

    for(i = 0; i < L; i++)
      myfile << arr5[i] << " ";

   // zatvaramo datoteku u koju smo upisivali rezultat
   myfile.close();

   return 0;
}
```

Primer jedne manje instance (za I=5, J=5, L=2, Q=5):

```
5 5 2 5

90 40 14 25 55

57 59 100 2 69 39 7 65 88 16 10 18 9
98 72 89 60 16 86 87 5 71 34 71 19 33
66 30 31 49 82 5 72 47 2 96 100 83 90
25 53 46 64 65 87 40 10 20 76 60

39 74 15 10 53 78 82 36 91 97 51 55 63
89 62 51 38 47 71 88 73 59 75 60 92

25 28 87 48 85

59 86
```

4 Rezultati CPLEX-a

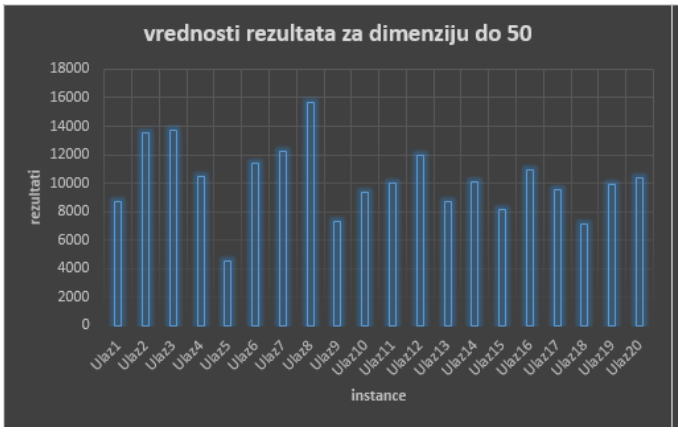
	ULAZNI PODACI					CPLEX	
	Ulaz	I	J	L	Q	Rezultat	Vreme
dimenzija do 50	Ulaz1	5	5	2	5	8680.661	0.1076
	Ulaz2	10	10	2	2	13581	0.2804
	Ulaz3	10	15	2	5	13686.15	0.3524
	Ulaz4	10	20	2	2	10499.156	0.4218
	Ulaz5	15	15	2	5	4499	0.4422
	Ulaz6	15	20	2	2	11417	0.555
	Ulaz7	15	25	2	2	12274.221	0.637
	Ulaz8	20	20	2	5	15659.5	0.6698
	Ulaz9	20	25	2	2	7281.52	0.7886
	Ulaz10	25	25	2	2	9380.25	0.9388
	Ulaz11	30	30	3	2	10022.755	1.6904
	Ulaz12	30	40	3	2	11953.164	2.1994
	Ulaz13	30	40	4	2	8688.579	2.7148
	Ulaz14	30	45	3	2	10117.365	2.4584
	Ulaz15	35	35	3	2	8143.346	2.2476
	Ulaz16	35	50	5	2	10899.364	4.6472
	Ulaz17	40	40	4	2	9553.127	3.5832
	Ulaz18	45	45	4	2	7093.583	4.488
	Ulaz19	50	50	4	2	9886.234	5.4898
	Ulaz20	50	50	5	2	10370.909	6.5606

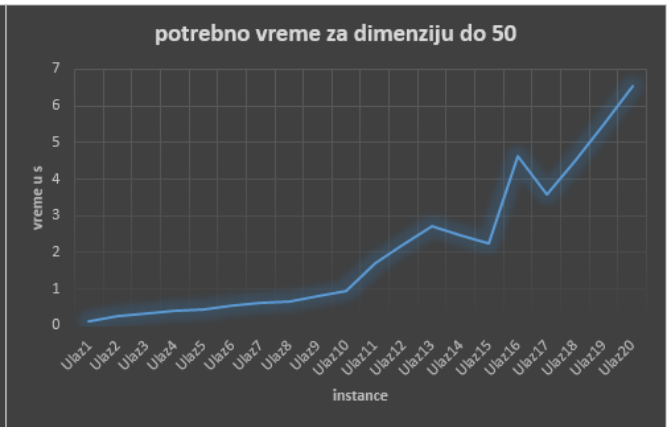
	ULAZNI PODACI					CPLEX	
	Ulaz	I	J	L	Q	Rezultat	Vreme
dimenzija do 100	Ulaz21	55	55	2	2	11795.221	3.9762
	Ulaz22	55	55	5	2	10972.556	7.9256
	Ulaz23	55	75	6	2	12158.582	12.5366
	Ulaz24	60	60	5	2	12085.188	9.4042
	Ulaz25	60	80	6	2	17022.604	14.5602
	Ulaz26	60	80	8	2	15862.5	18.6828
	Ulaz27	65	65	5	2	9277.953	10.997
	Ulaz28	70	95	7	2	16832.478	22.9784
	Ulaz29	70	95	9	2	15556.968	28.7366
	Ulaz30	75	75	6	2	13007.453	17.0478
	Ulaz31	75	100	8	2	15694.731	29.1718
	Ulaz32	75	100	10	2	15486.227	36.4765
	Ulaz33	80	80	6	2	11949.212	19.3946
	Ulaz34	80	80	8	2	12271.667	25.004
	Ulaz35	85	85	7	2	12716.665	25.0256
	Ulaz36	90	90	9	2	12829.667	34.9555
	Ulaz37	95	95	8	2	14319.627	35.048
	Ulaz38	95	95	9	2	14935.566	38.977
	Ulaz39	100	100	2	2	22346.545	12.8908
	Ulaz40	100	100	10	2	15417.805	48.669

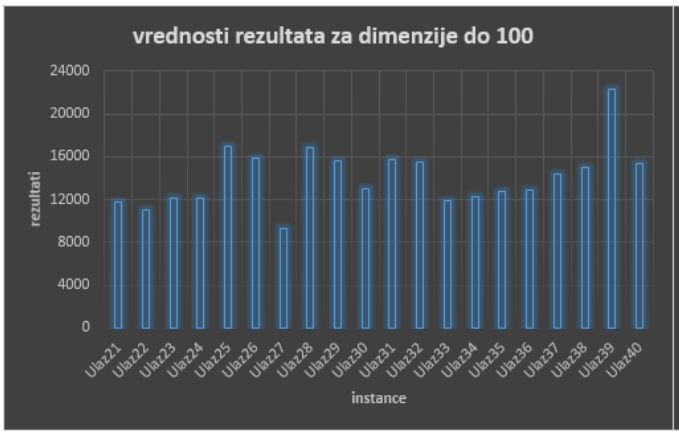
	ULAZNI PODACI					CPLEX	
	Ulaz	I	J	L	Q	Rezultat	Vreme
dimenzija preko 100	Ulaz41	105	105	10	2	14601.071	54.014
	Ulaz42	110	110	10	2	14039.761	59.298
	Ulaz43	115	115	11	2	17575.613	71.1
	Ulaz44	115	155	15	2	22464.992	125.7837
	Ulaz45	120	120	12	2	15409.995	81.6925
	Ulaz46	120	160	15	2	18285.279	135.421
	Ulaz47	130	130	13	2	16286.216	103.786
	Ulaz48	135	180	15	2	20974.095	172.293
	Ulaz49	135	180	18	2	23607.667	204.833
	Ulaz50	140	140	13	2	18155.215	123.693
	Ulaz51	140	140	14	2	20584.609	133.855
	Ulaz52	145	145	14	2	18411.925	139.397
	Ulaz53	150	150	15	2	20016.706	159.374
	Ulaz54	150	200	18	2	23956.923	254.483
	Ulaz55	150	200	20	2	26110.4	281.226
	Ulaz56	160	160	15	2	21825.613	182.020
	Ulaz57	170	170	17	2	19801.367	231.756
	Ulaz58	180	180	18	2	20963.347	275.572
	Ulaz59	190	190	19	2	21875.867	323.682
	Ulaz60	200	200	20	2	22106.423	377.797

5 Komentar i analiza dobijenih rezultata

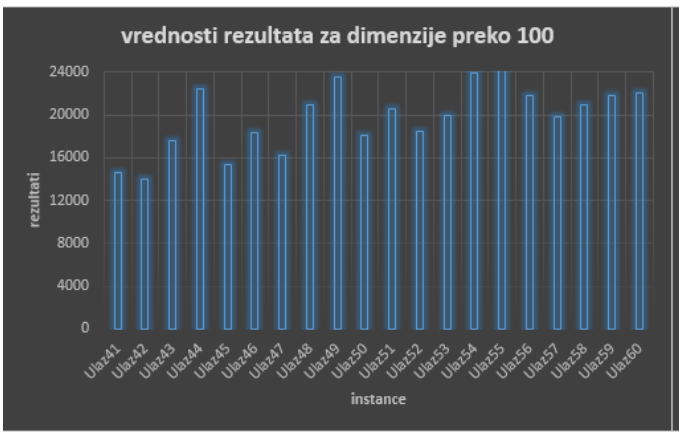
Problem je rešen korišćenjem CPLEX-a i testiran je na 60 različitih ulaznih podataka. Od kojih je 20 manjih dimenzija (I, J i L uzimaju vrednosti do 50), 20 srednjih dimenzija (I, J i L uzimaju vrednosti od 50 do 100) i 20 vecih dimenzija (I, J i L uzimaju vrednosti vece od 100).
Vreme izvršavanja je izraženo u sekundama, a svaka instanca je testirana 5 puta i uzet je prosek vremena izvršavanja. Merenje je izvršeno na PC računaru sa Intel Core 2 Duo procesorom sa radnim taktom 2GHz pod Windows 7 operativnim sistemom. Povećanjem granice L za samo 1 primećuje se da kod instanci manjih dimenzija imamo povećanje potrebnog vremena za dobijanje rešenja za oko 20% (ulaz19 i ulaz20 oko 1 sekunde), dok kod instanci srednjih dimenzija imamo povećanje za oko 10% (ulaz37 i ulaz38 oko 4 sekunde). Kod instanci većih dimenzija to povećanje se jos procentualno smanjuje pa izvosi oko 8% (ulaz50 i ulaz51), ali se vreme potrebno za dobijanje rešenja značajno povećava usled većih dimenzija problema i u ovom slučaju iznosi već oko 10 sekundi.

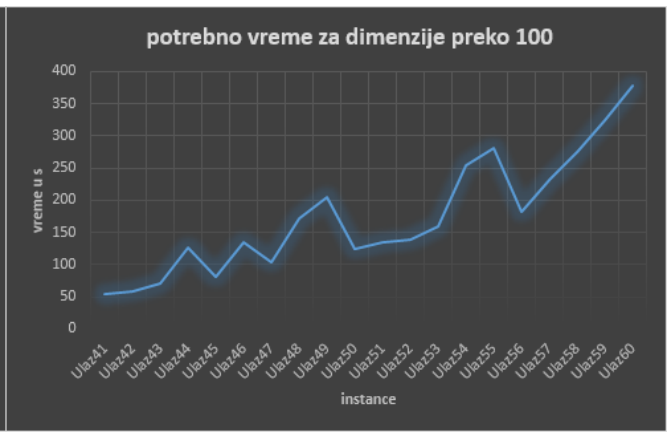

vrednosti rezultata za dimenziju do 50
rezultati
instance


potrebno vreme za dimenziju do 50
vreme u s
instance


vrednosti rezultata za dimenzije do 100
rezultati
instance


potrebno vreme za dimenziju do 100
vreme u s
instance


vrednosti rezultata za dimenzije preko 100
rezultati
instance


potrebno vreme za dimenzije preko 100
vreme u s
instance

6 Kod programa

```
#include <ilcplex/cplex.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <time.h>
#include <conio.h>
#include <windows.h>
#include <winbase.h>

int main()
{
  int startTime = clock();
  FILE* fin = fopen("input.txt", "r");
  if (fin == NULL)
    exit(-1);

  int Q;
  int I, J, L;
  int i,j,l;

  // ucitavamo podatke iz datoteke input.txt i upisujemo vrednosti u promenljive
  fscanf(fin, "%d %d %d %d", &I, &J, &L, &Q);
    printf("I=%d, J=%d, L=%d, Q=%d\n\n", I, J, L, Q);


    //deklarisemo i alociramo prostor za promenljive koje koristimo i upisujemo
        vrednosti
  double* f = (double*) malloc(I * sizeof(double));
  for(i = 0; i < I; i++){
    fscanf(fin, "%lf", &f[i]);

  printf("f[%d]=%lf\n", i, f[i]);
  }
  double* g = (double*) malloc(I * J * (L + 1) * sizeof(double));
  for (i = 0; i < I; i++)
    for (j = 0; j < J; j++)
        for (l = 1; l < L + 1; l++){
          fscanf(fin, "%lf", &g[l]);
          printf("\ng[%d][%d][%d]=%lf\n", i,j,l, g[l]);
        }
  double* c = (double*) malloc(I * J * sizeof(double));
  for (i = 0; i < I; i++)
    for (j = 0; j < J; j++){
      fscanf(fin, "%lf", &c[i * J + j]);
      printf("\nc[%d][%d]=%lf\n", i,j, c[i * J + j]);
        }

  double* d = (double*) malloc(J * sizeof(double));
  for (j = 0; j < J; j++){
    fscanf(fin, "%lf", &d[j]);
    printf("\nd[%d]=%lf\n", j, d[j]);
        }
```

```
  double* C = (double*) malloc((L + 1)* sizeof(double));
  for (l = 1; l < L+1; l++){
    fscanf(fin, "%lf", &C[l]);
  printf("\nC[%d]=%lf\n", l, C[l]);
        }
  fclose(fin);

  int status;
  CPXENVptr env = CPXopenCPLEX(&status);
  CPXLPptr lp = CPXcreateprob(env, &status, "CFLP");
  CPXchgobjsen(env, lp, CPX_MIN);
  // pravimo funkciju cilja
  double* coefs = (double*) malloc((I + I*J*L + I*J) * sizeof(double));
  double* lb = (double*) malloc((I + I*J*L + I*J) * sizeof(double));
  double* ub = (double*) malloc((I + I*J*L + I*J) * sizeof(double));
  char* type = (char*) malloc((I + I*J*L + I*J) * sizeof(char));

  // prvi deo: jedna suma
  for(i = 0; i < I; i++)
  {
      coefs[i] = f[i];
      lb[i] = 0;
      ub[i] = 1;
      type[i] = 'B';
  }

  // drugi deo: tri sume
  for(i = 0; i < I; i++)
    for (j = 0; j < J; j++)
        for (l = 1; l < L; l++)
        {
          coefs[I + i*J + j*L + l] = g[l];
          lb[I + i*J + j*L + l] = 0;
          ub[I + i*J + j*L + l] = CPX_INFBOUND;
          type[I + i*J + j*L + l] = 'C';
        }

  // treci deo: dve sume
  for (i = 0; i < I; i++)
    for (j = 0; j < J; j++)
    {
      coefs[I + I * J * L + i * J + j] = c[i * J + j];
      lb[I + I * J * L + i * J + j] = 0;
      ub[I + I * J * L + i * J + j] = CPX_INFBOUND;
      type[I + I * J * L + i * J + j] = 'C';
    }

  CPXnewcols(env, lp, I + I*J*L + I*J, coefs, lb, ub, type, NULL);

// uslovi

  // prvi uslov
  int* rmatind = (int*) malloc(I * sizeof(int));
  double* rmatval = (double*) malloc(I * sizeof(double));
  double* rhs = (double*) malloc(sizeof(double));
```

```
int* rmatbeg = (int*) malloc(sizeof(int));
char* sense = (char*) malloc(sizeof(char));

for (j = 0; j < J; j++)
{
  rhs[0] = d[j];
  sense[0] = 'E';
  rmatbeg[0] = 0;
  for (i = 0; i < I; i++)
  {
    rmatind[i] = I + I*J*L + i*J+j;
    rmatval[i] = 1;
  }
  CPXaddrows(env, lp, 0, 1, I, rhs, sense, rmatbeg, rmatind, rmatval, NULL, NULL)
     ;
}
free(rhs);
free(sense);
free(rmatind);
free(rmatbeg);
free(rmatval);

// drugi uslov
rmatind = (int*) malloc((J + 1) * sizeof(int));
rmatval = (double*) malloc((J + 1) * sizeof(double));
rhs = (double*) malloc(sizeof(double));
rmatbeg = (int*) malloc(sizeof(int));
sense = (char*) malloc(sizeof(char));
for (i = 0; i < I; i++)
{
  rhs[0] = 0;
  sense[0] = 'G';
  rmatbeg[0] = 0;
  for (j = 0; j < J; j++)
  {
    rmatind[j] = I + I*J*L + i*J+j;
    rmatval[j] = 1;
  }
  rmatind[J] = i;
  rmatval[J] = -Q;
  CPXaddrows(env, lp, 0, 1, J + 1, rhs, sense, rmatbeg, rmatind, rmatval, NULL,
     NULL);
}
free(rhs);
free(sense);
free(rmatind);
free(rmatbeg);
free(rmatval);

// treci uslov
rmatind = (int*) malloc( (L + 1) * sizeof(int));
rmatval = (double*) malloc( (L + 1) * sizeof(double));
rhs = (double*) malloc(sizeof(double));
rmatbeg = (int*) malloc(sizeof(int));
sense = (char*) malloc(sizeof(char));
for (i = 0; i < I; i++)
```

```
    for(j = 0; j < J; j++)
  {
    rhs[0] = 0;
    sense[0] = 'G';
    rmatbeg[0] = 1;
    for (int l = 1; l < L+1; l++)
    {
      rmatind[l] = I + i * J + j * L + l;
      rmatval[l] = C[l];
    }
    rmatind[L] = I + I*J*L + i*J+j;
    rmatval[L] = -1;

    CPXaddrows(env, lp, 0, 1, L + 1, rhs, sense, rmatbeg, rmatind, rmatval, NULL,
       NULL);
  }
  free(rhs);
  free(sense);
  free(rmatind);
  free(rmatbeg);
  free(rmatval);

  status = CPXsetdblparam (env, CPX_PARAM_TILIM, 14400); //ogranicava se rad na 4h
  int cvorova = CPXgetnodecnt(env, lp);
  int iteracija = CPXgetmipitcnt(env, lp);

  CPXchgprobtype(env, lp, CPXPROB_MILP);
  CPXmipopt(env, lp);
  double objval;
  CPXgetobjval(env, lp, &objval);
  printf("%.3lf, t = %.3lf\n br_cvorova:%d, iteracija:%d\n", objval, (clock() -
     startTime) / 1000.0, cvorova, iteracija);
  CPXfreeprob(env, &lp);
  CPXcloseCPLEX(&env);
  system("PAUSE");
  return 0;
}
```

7 Link za skidanje zadatka

http://alas.matf.bg.ac.rs/ mi09018/projects/MPIO/