

Variable Intensified Neighborhood Search za Rešavanje Problema Maksimalne Različitosti

seminarski zadatak iz matematičkog programiranja i optimizacije

Student: Petar Jovanović, 1135/2012, 5i
Problem: 15

Apstrakt

Problem maksimalne različitosti je predstavljen kao problem mešovitog celobrojnog programiranja, u kome se iz datog skupa tačaka u prostoru $\mathbb{R}^K$ bira podskup određene veličine takav da rastojanje, definisano datom metrikom, među tačkama koje se nađu u njemu bude maksimalno. U ovom radu prikazan je pristup rešavanju problema pomoću matheuristike radnog naziva Variable Intensified Neighborhood Search, koja je razvijena kao generalizacija Variable Neighborhood Decomposition Search matheuristike. Prikazani rezultati su dobijeni sa preliminarnim skupom definicija okolina i parametara algoritma (kao što su veličine okolina i vremena pretrage svake) i daju utisak da je potrebno dalje eksperimentisanje da bi se našle okoline i parametri optimalniji za ovaj problem.

ključne reči: matheuristike, optimizacija, mip, mdp.

1. Opis problema

Problem maksimalne različitosti (maximum diversity problem) se sastoji u biranju podskupa od m elementa iz skupa veličine n , tako da je suma rastojanja (razlika) između izabranih elemenata maksimalna. Svaki element se predstavlja kao vektor dužine K , ili tačka u K -dimenzionalnom prostoru $\mathbb{R}^K$. Neka je s_{ik} vrednost k -tog atributa i -tog elementa, gde $i=1,2,\dots,n$; $k=1,2,\dots,K$. Onda se udaljenost d_{ij} između dva elementa i i j može jednostavno definisati kao euklidsko rastojanje, prikazano sledećom formulom:

$$d_{ij} = \sqrt{\sum_{k=1}^K (s_{ik} - s_{jk})^2}.$$

Problem se tada može formulisati kao problem kvadratnog 0-1 celobrojnog programiranja, gde promenljiva x_i uzima vrednost 1 ako je element i izabran ili 0 u protivnom, za $i=1,2,\dots,n$.

Maksimizovati:

$$z = \sum_{i=2}^{n-1} \sum_{j=i+1}^n d_{ij} x_i x_j$$

takvo da:

$$\sum_{i=1}^n x_i = m$$
$$x_i \in \{0, 1\}, 1 \leq i \leq n.$$

U ovakvoj formulaciji, funkcija cilja predstavlja sumu rastojanja između svih parova izabranih elemenata, a uslov ograničava da mora biti tačno m izabranih elemenata.

Glover i Woolsey (1974) su pokazali kako se problem 0-1 polinomijalnog programiranja može prevesti u 0-1 linearno programiranje, menjanjem proizvoda određenih promenljivih

novim promenljivima koje zadovoljavaju dodatne uslove. Kuo et al. (1993) su primenili ovu transformaciju i zamenili proizvode $x_i x_j$ novim promenljivima y_{ij} .

Maksimizovati:
$$z = \sum_{i=1}^{n-1} \sum_{j=i+1}^n d_{ij} y_{ij}$$

takvo da:

$$\sum_{i=1}^n x_i = m,$$

$$x_i + x_j - y_{ij} \leq 1, \quad 1 \leq i < j \leq n,$$

$$-x_i + y_{ij} \leq 0, \quad 1 \leq i < j \leq n,$$

$$-x_j + y_{ij} \leq 0, \quad 1 \leq i < j \leq n,$$

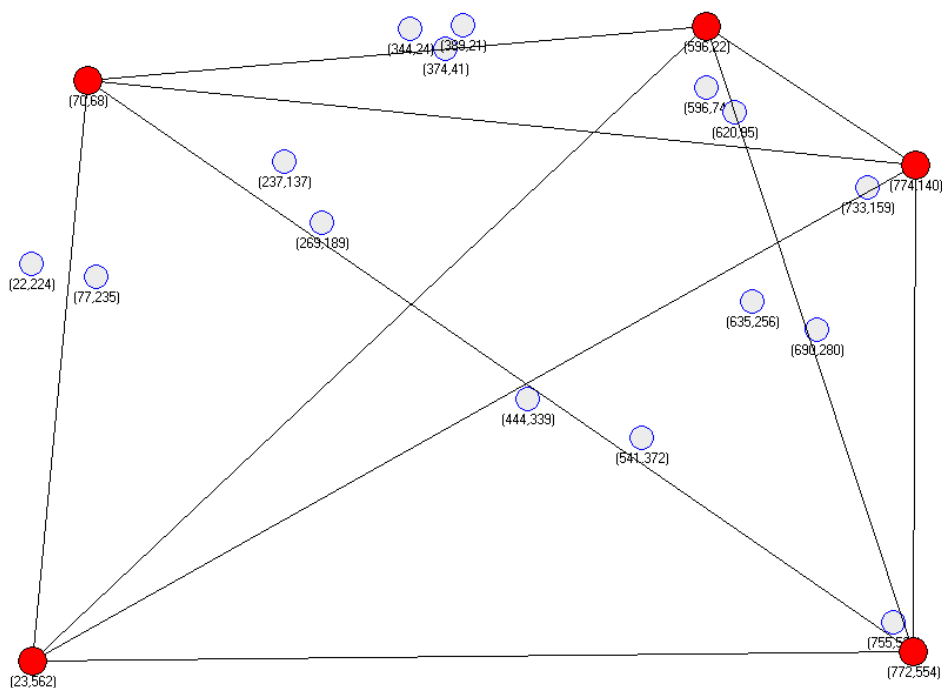
$$y_{ij} \geq 0, \quad 1 \leq i < j \leq n,$$

$$x_i \in \{0, 1\}, \quad 1 \leq i \leq n.$$

Ovako postavljena funkcija cilja je linearna i predstavlja zbir rastojanja parova odabranih tačaka, za koje je $y_{ij}=1$. Ograničenja obezbeđuju da x_i i y_{ij} promenljive budu binarne i da y_{ij} ima vrednost jedan akko x_i i x_j promenljive obe imaju vrednost 1.

Ovaj opis problema je korišćen u implementaciji rešenja baziranog na VINS matheuristici i CPLEX-u.

Na slici 1 predstavljeno je jedno optimalno rešenje instance dvodimenzionalnog MDP-a za 20 slučajno raspoređenih tačaka od kojih se bira 5, pri čemu je metrika različitosti euklidsko rastojanje. Izabrane tačke su obojene crveno i spojene linijama i zbir njihovih rastojanja, tj. vrednost funkcije cilja iznosi 6171.48971279.



Slika 1: Primer optimalnog rešenja problema maksimalne različitosti u 2 dimenzije, sa euklidskim rastojanjem kao metrikom različitosti. Od 20 slučajno generisanih tačaka izabrano je 5 i dobijena je vrednost optimalnog rešenja koja iznosi 6171.48971279

2. Uvodne definicije

U ovom odeljku opisani su glavni pojmovi i algoritmi korišćeni u opisu VINS matheuristike. Oni obuhvataju probleme mešovito celobrojnog programiranja, Lokalno grananje, metode promenljivih okolina (Variable Neighborhood Search, VNS) i metodu Dekompozicije kroz promenljive okoline (Variable Neighborhood Decomposition Search, VNDS).

2.1 Matheuristike i MIP problemi

Iako postoje algoritmi za rešavanje problema 0-1 mešovito celobrojnog programiranja, kao što je problem maksimalne različitosti, zbog veličine problema koji se sreću u primenama, često je njihovo korišćenje nepraktično. Zbog toga su uvođeni heuristički metodi koji su prilagođeni datom problemu i manje generalni i obično ne daju dokaze optimalnosti rešenja, niti garantuju nalaženje globalnog optimalnog rešenja. Metaheuristike su generalni obrasci po kojima se mogu konstruisati takve heuristike.

Matheuristike su hibridi između metaheuristika i metoda zasnovanih na matematičkom programiranju. One su u generalnom slučaju egzaktno metode, u smislu da garantuju nalaženje optimalnog rešenja ako im se da dovoljno vremena. Međutim, ako je dato vreme kraće od potrebnog za nalaženje optimalnog rešenja i dokazivanje njegove optimalnosti, one se ponašaju kao metaheuristike koje bi trebalo brže da nalaze optimalnija rešenja fokusirajući pretragu na određene delove prostora mogućih rešenja. To navođenje pretrage se postiže manipulacijama nad matematičkim modelom problema, npr. dodavanjem linearnih ograničenja.

Problem nad kojem su primenjene matheuristike prikazane u ovom radu spada u kategoriju problema 0-1 mešovito celobrojnog programiranja (Mixed Integer Programming, MIP) koji se mogu definisati na sledeći način:

Neka je data funkcija cilja koju treba optimizovati (maksimizovati ili minimizovati):

$$\max(\min)\{v(\xi) = c^T \xi \mid \xi \in X\},$$

gde je:

- $X = \{\xi \in \mathbb{R}^N \mid A\xi \leq b; \xi_j \in [0, 1] \text{ za } j \in B; \xi_j \in \mathbb{Z}_0^+ \text{ za } j \in G; \xi_j \geq 0 \text{ za } j \in C\}$ skup dopustivih rešenja,
- $N = C \cup G \cup B$ skup indeksa svih promenljivih,
- C skup indeksa neprekidnih promenljivih,
- G skup indeksa celobrojnih promenljivih (koje nisu binarne), a
- B skup indeksa binarnih promenljivih,
- $C \cap G = \emptyset, C \cap B = \emptyset, G \cap B = \emptyset, B \neq \emptyset$.

Definiše se i potproblem kao redukcija početnog MIP problema dobijena kada se određenim promenljivima dodeli i fiksira vrednost. Formalno, ako imamo x^0 kao proizvoljan vektor indeksa binarnih varijabli, u kome je $x_j^0 \in \{0, 1\}$, $j \in B$ i $J \subseteq B$, potproblem

$P(x^0, J)$ se može definisati na sledeći način:

minimizovati/maksimizovati: $c^T x$

takvo da: $Ax \geq b$

$$\begin{aligned} x_j &= x_j^0, & \forall j \in J \\ x_j &\in \{0, 1\}, & \forall j \in B \setminus J \\ x_j &\geq 0, & \forall j \in C \end{aligned}$$

Osim definicije MIP problema, koristi se i linearna relaksacija MIP problema, a pod njom se podrazumeva problem linearnog programiranja ekvivalentan datom MIP problemu u kome su uklonjena ograničenja da promenljive iz skupova B i G moraju biti binarne ili celobrojne.

2.2 Lokalno grananje

Metodu lokalnog grananja su uveli Fischetti i Lodi (2003) i ona predstavlja jednu od prvih formulacija matheuristika. Za opis lokalnog grananja, potrebno je definisati rastojanje u 0-1 MIP prostoru rešenja.

Ako je S prostor rešenja za 0-1 MIP problem P , a $\bar{S}$ prostor rešenja za LP linearnu relaksaciju $LP(P)$ problema P , rastojanje između $x \in S$ i $y \in \bar{S}$ se definiše kao:

$$\delta(x, y) = \sum_{j \in B} |x_j - y_j|,$$

što se može linearizovati na sledeći način:

$$\delta(x, y) = \sum_{j \in B} x_j(1 - y_j) + y_j(1 - x_j).$$

Algoritam počinje nalaženjem prvog dopustivog rešenja x' za šta se koristi lokalni MIP rešavač. Zatim se postavlja ograničenje pretrage na ona rešenja x za koje važi $\delta(x, x') \leq k$, gde je k u početku zadato kao parametar metode. Tada se poziva egzaktni MIP rešavač (CPLEX) da reši takav potproblem sa zadatim vremenskim ograničenjem. U zavisnosti od rezultata koje vrati MIP rešavač, rastojanje k se povećava ili smanjuje, a proces se ponavlja do zadovoljenja kriterijuma zaustavljanja. Pseudokod za lokalno grananje je dat u listingu 1.

LocalBranching(*mip*, x' , k_init)

1. *proceed* = true
 2. **while**(*proceed*)
 3. Add cuts: $c^T x \geq c^T x'$, $\delta(x', x) \leq k$
 4. $x'' = \text{MIPsolve}(\textit{mip}, x')$
 5. **switch**(*status*)
 6. **case** "optSolFound": reverse last pseudo-cut into
 7. $\delta(x', x) \geq k+1$; $x' = x''$; $k = k_init$;
 8. **case** "feasibleSolFound": replace last pseudo-cut with
 9. $\delta(x', x) \geq 1$; $x' = x''$; $k = k_init$;
 10. **case** "provenInfeasible": reverse last pseudo-cut into
 11. $\delta(x', x) \geq k+1$; $k = k + \text{ceil}(k_init / 2)$;
 12. **case** "noFeasibleSolFound": delete last pseudo-cut
 13. $\delta(x', x) \leq k$; $k = k - \text{ceil}(k_init / 2)$;
 14. **end**
 15. **end**
 16. **return** x'
-

Listing 1: Lokalno grananje (Local Branching)

2.3 Metode promenljivih okolina

Metoda promenljivih okolina predstavlja metaheuristiku koja se zasniva na sistematskom menjanju okolina ka lokalnim optimumima, kao i van regiona koji ih sadrže. Strukture okolina se definišu u prostoru rešenja X u kome je definisano i rastojanje $d(x, x')$ između bilo koja dva rešenja $x \in X$ i $x' \in X$. Dve najpoznatije matheuristike bazirane na ovoj metodi su grananje kroz promenljive okoline (Hansen, Mladenović, Urošević, 2006) i dekompozicija kroz promenljive okoline (VNDS).

2.4 Dekompozicija kroz promenljive okoline za 0-1 MIP (VNDS)

VNDS je matheuristika bazirana na metodi promenljivih okolina koja je opisana u radu Lazić et al. (2010). Njena specifičnost u odnosu na VNS je u tome što se okoline definisane u prostoru promenljivih u funkciji cilja menjaju ne direktno u zavisnosti od rastojanja između rešenja, nego u zavisnosti od rastojanja svake pojedinačne promenljive između rešenja. Iskorišćen je koncept potproblema iz odeljka 2.1, zasnovan na fiksiranju i otpuštanju promenljivih, gde fiksiranje znači da se vrednost promenljive neće menjati pri optimizaciji, a otpuštanje ponovo dozvoljava menjanje vrednosti date promenljive.

Metoda počinje nalaženjem rešenja linearne relaksacije problema. Zatim se nalazi prvo dopustivo rešenje originalnog MIP problema i računa se rastojanje vrednosti svake promenljive u dopustivom rešenju od odgovarajućih promenljivih u rešenju linearne relaksacije i pravi se lista promenljivih sortirana prema tom rastojanju. To rastojanje se može definisati kao apsolutna vrednost razlike vrednosti promenljivih.

Lista promenljivih se zatim deli na d jednakih intervala i promenljive u svim intervalima se fiksiraju, osim u poslednjem, gde se nalaze one najudaljenije od njihovih vrednosti u rešenju linearne relaksacije. Tada se izvršava lokalna optimizacija MIP rešavačem. Ako se pronađe bolje rešenje od trenutnog, trenutno rešenje postaje inkumbent i ponovo se računaju rastojanja promenljivih od promenljivih u rešenju linearne relaksacije i lista se sortira. Ako se ne nađe bolje rešenje, otpušta se sledeći interval promenljivih. Ova procedura se ponavlja do isteka vremena, nalaženja optimalnog rešenja.

Kada se iscrpe intervali sa promenljivima koje su daleko od vrednosti u rešenju linearne relaksacije, preostali interval promenljivih koje su bliske vrednostima u LP se deli na 2 intervala i oni se otpuštaju i istražuju na isti način kao prethodni intervali. Granica kojom se određuje bliskost se može posmatrati i kao parametar algoritma.

Pseudokod VNDS metode je prikazan u listingu 2, a listing 3 sadrži metodu spusta kroz promenljive okoline (variable neighborhood descent) za MIP, koja se koristi u VNDS.

VNDS($P, d, t_{max}, t_{sub}, t_{vnd}, t_{mip}, rhs_{max}$)

1. Find optimal solution $\bar{x}$ of LP(P)
2. **if** $\bar{x}$ is integer feasible
3. **return** $\bar{x}$
4. Find first feasible 0-1 MIP solution of P
5. $t_{Start} = \text{cpuTime}()$; $t = 0$;
6. **while** ($t < t_{max}$)
7. $\delta_j = |(x_j - \bar{x})|$, $j \in B$; index variables x_j , $j \in B$,
 so that $\delta_1 \leq \delta_2 \leq \dots \leq \delta_p$, $p = |B|$;
8. $n_d = |\{j \in B | \delta_j \neq 0\}|$; $k_{step} = \lfloor n_d / d \rfloor$; $k = p - k_{step}$;
9. **while** ($t < t_{max}$)

```

10.    $x' = \text{MIPsolve}(P(x, \{1, 2, \dots, k\}), t_{\text{sub}}, x)$ 
11.   if ( $c^T x' < c^T x$ )
12.        $x = \text{VND-MIP}(P, t_{\text{vnd}}, t_{\text{mip}}, rhs_{\text{max}}, x')$ 
13.       break;
14.   else
15.       if ( $k - k_{\text{step}} > p - n_d$ )
16.            $k_{\text{step}} = \max\{\lfloor k/2 \rfloor, 1\}$ 
17.            $k = k - k_{\text{step}}$ 
18.            $t_{\text{end}} = \text{cpuTime}(); t = t_{\text{end}} - t_{\text{start}};$ 
19.   end
20. end
21. return  $x$ 

```

Listing 2: VNDS pseudo kod za MIP probleme.

```

VND-MIP( $P, t_{\text{vnd}}, t_{\text{mip}}, rhs_{\text{max}}, x'$ )
1.   $rhs = 1; t_{\text{start}} = \text{cpuTime}(); t = 0;$ 
2.  while ( $t < t_{\text{vnd}}$  and  $rhs \leq rhs_{\text{max}}$ )
3.       $TimeLimit = \min(t_{\text{mip}}, t_{\text{vnd}} - t)$ 
4.      add the pseudo-cut  $\delta(B, x', x) \leq rhs$ 
5.       $x'' = \text{MIPsolve}(P, TimeLimit, x')$ 
6.      switch ( $solutionStatus$ )
7.          case "optSolFound":
8.              reverse last pseudo-cut into  $\delta(B, x', x) \geq rhs + 1$ 
9.               $x' = x''; rhs = 1;$ 
10.         case "feasibleSolFound":
11.             replace last pseudo-cut with  $\delta(B, x', x) \geq 1$ 
12.              $x' = x''; rhs = 1;$ 
13.         case "provenInfeasible":
14.             reverse last pseudo-cut into  $\delta(B, x', x) \geq rhs + 1$ 
15.              $rhs = rhs + 1$ 
16.         case "noFeasibleSolFound":
17.             break;
18.     end
19.      $t_{\text{end}} = \text{cpuTime}(); t = t_{\text{end}} - t_{\text{start}}$ 
20. end
21. return  $x''$ 

```

Listing 3: VND pseudo kod za MIP probleme.

3. Variable Intensified Neighborhood Search

VINS matheuristika je generalizacija VNDS matheuristike, koja definiše nove obrasce za generisanje potproblema otpuštanjem i fiksiranjem promenljivih, kao i variranje vremenskog ograničenja za lokalnu optimizaciju u zavisnosti od veličine okoline koja se pretražuje. Sličan

pristup je dat u radu Mitrović-Minić i Punnen (2009), u kome je uvedena heuristika za Multi-resource generalized assignment problem (MGRAP) bazirana na Very-Large Scale Neighborhood Search i VNS, u kojoj se takođe generišu potproblemi pomoću binding setova definisanih unapred zadatim pravilima.

Kao i u dekompoziciji kroz promenljive okoline, VINS koristi meru rastojanja definisanu u prostoru promenljivih rešenja X , koja predstavlja razliku pojedinačnih promenljivih x_i i y_i za svaka dva rešenja x i y , $x \in X$, $y \in X$.

Optimizacija počinje tako što se reši linearna relaksacija problema da bi se dobila procena pozicije optimalnog rešenja u prostoru promenljivih. Zatim se nalazi početno dopustivo celobrojno rešenje. Računa se razlika vrednosti promenljivih rešenja linearne relaksacije i početnog rešenja, i prema tom rastojanju se promenljive sortiraju.

U VNDS-u postoji jedinstven obrazac za fiksiranje i otpuštanje promenljivih koji se ne menja tokom izvršavanja dok VINS definiše 10 različitih načina za fiksiranje i otpuštanje promenljivih, iz kojih nastaje 10 različitih obrazaca za okoline, a to su:

- N1: $\alpha\%$ najgorih promenljivih se otpušta (kao VNDS)
- N2: u 10 jednakih intervala se otpušta $(\alpha/10)\%$ najgorih promenljivih
- N3: počev od slučajne pozicije otpušta se $\alpha\%$ promenljivih
- N4: 10 puta počevši od slučajne pozicije se otpušta $(\alpha/10)\%$ promenljivih
- N5: $\alpha\%$ najboljih promenljivih se otpušta
- N6: u 10 jednakih intervala se otpušta po $(\alpha/10)\%$ najboljih promenljivih
- N7: $(\alpha/2)\%$ najgorih i $(\alpha/2)\%$ najboljih promenljivih se otpuštaju
- N8: u 10 jednakih intervala primeni se šema kao N8
- N9: $\alpha\%$ slučajno izabranih promenljivih se otpušta
- N10: u 10 intervala se otpušta po $(\alpha/10)\%$ slučajno odabranih promenljivih.

Ako bi se koristila samo N1 okolina, VINS bi se sveo na dekompoziciju kroz promenljive okoline.

Osim iteracija u otpuštanju promenljivih prema šemama okolina, vrši se i povećavanje α , koje predstavlja veličinu okoline, dok šema predstavlja njen oblik. Iteracija se izvršava ciklično po obrascima okolina (N1... N10) a zatim po veličinama okolina (α).

Kada se u nekoj okolini nađe bolje rešenje ono se uzima kao trenutno rešenje, a promenljive se ponovo sortiraju prema rastojanju od promenljivih rešenja linearne relaksacije. Dalje iteracije po okolinama se nastavljaju po redosledu, bez vraćanja na N1 ili najmanje α , tj. ne vrši se intenziviranje pretrage oko novog rešenja.

Pseudokod za VINS metodu je dat u listingu 4, dok je metoda za kontrolu redosleda okolina prikazana u listingu 5. Parametri koje VINS metoda uzima su sledeći:

- *mip* - struktura koja sadrži MIP problem koji treba rešiti;
- *alphas* - niz veličina okolina, gde su veličine izražene kao procenat promenljivih koje se otpuštaju;
- *time_limits* - niz vremenskih ograničenja za istraživanje okolina svake veličine.

Metoda NextNeighborhood održava niz okolina generisanih iz gorepomenutih obrazaca i veličina okolina, kao i brojač koji označava koja se okolina sledeća istražuje. Obe ove promenljive su globalne ili statičke, tj. njihovo stanje ostaje zapamćeno između poziva funkcije.

U svakoj od definisanih okolina se poziva lokalna optimizacija (CPLEX) sa vremenskim ograničenjem koje se povećava sa povećanjem veličine okoline (α).

```

VINS(mip, alphas, time_limits)
1. linp = LinearRelaxation(mip)
2. linx = MIPSolve(linp)
3. x = MIPSolve(mip, max_solutions = 1) //first feasible
4. bestx = x
5. while time_limits not exceeded
6.   SortVars(x, linx)
7.   while N = NextNeighborhood()
8.     if time limit exceeded
9.       break
10.  FixVars(mip, N, alpha)
11.  ReleaseVars(mip, N, alpha)
12.  x = MIPSolve(mip, TimeLimit(alpha))
13.  if Improved(x)
14.    bestx = x
15.    break
16.  if not Improved(x)
17.    break
18. return bestx

```

Listing 4: Variable Intensified Neighborhood Search

```

NextNeighborhood()
  neighPattern je niz nizova bitova koji signalizuju da li se promenljiva na poziciji datog
  bita fiksira (1) ili otpušta (0)
  currentNeigh je brojač kojim se indeksira niz neighPattern da bi se dobio odgovarajuć
  binarni obrazac za fiksiranje i otpuštanje promenljivih.
1. currentNeigh = currentNeigh + 1
2. if currentNeigh < nNeigh * nSizes
3.   return neighPattern[floor(currentNeigh / nNeigh) + currentNeigh mod nNeigh]
4. else
5.   return 0

```

Listing 5: metoda za zadavanje redosleda okolina u VINS matheuristici.

U nekim definicijama matheuristika, postoji zahtev da se one ponašaju deterministički u obilaženju rešenja, tj. da se nikada ne vraćaju u već posećena potencijalna rešenja. U trenutnoj definiciji VINS-a ova osobina nije obezbeđena, ali bi se mogla postići uvođenjem dodatnih ograničenja u definiciji problema. Ovde to nije urađeno jer i bez toga postoji gornja granica na broj puta koliko puta jedno rešenje može biti provereno (što je broj alfi, koji je obično vrlo mali), a dodatna ograničenja u problemu bi mogla produžiti vreme optimizacije.

4. Rezultati i analiza

Rezultati prikazani u ovom odeljku dobijeni su nad instancama iz MDPLIB seta problema maksimalne različitosti. Zbog ograničenja resursa, korišćene su samo instance iz GKD-c, MDG-a i MDG-b u kojima su dimenzije problema $n=500$ i $m=50$. Takođe su korišćene i instance SOM-a i SOM-b u kojima ima više varijacije u veličinama problema.

Parametri testova su podešeni na sledeći način:

- koristi se svih 10 navedenih okolina
- ukupno vreme izvršavanja po instanci je ograničeno na 2000 s
- veličine okolina su podeljene na 40%, 80% i 100%(otpuštanje svih promenljivih) a vremenska ograničenja za lokalne optimizacije tih veličina okolina su redom 200, 300, 500 i (preostalih) 1000 sekundi.

Ovakvi parametri su izabrani bez prethodnog testiranja i razmatranja optimalnijih podešavanja za problem maksimalne različitosti.

Test na svakoj instanci je ponovljen 10 puta i u Tabeli 1 su prikazane vrednosti za prosečno vreme do nalaženja najboljeg rešenja ($\text{avg } t_{\text{best}}$), srednje ukupno vreme izvršavanja ($\text{avg } t_{\text{tot}}$), vrednost funkcije cilja najboljeg nađenog rešenja (sol_{best}), prosečno procentualno odstupanje vrednosti nađenih rešenja od najboljeg (agap) kao i standardnu devijaciju vrednosti nađenih rešenja.

Rezultati za instance MDG-b 4,5,6,7,8 i 9 su netačni jer za njih nije pronađeno rešenje u nekoliko iteracija testova (2 ili 3). Ovo bi se moglo pripisati lošoj inicijalizaciji (izboru rešenja linearne relaksacije koje se nalazi daleko od bilo kog feasible rešenja početnog MIP problema) i veoma nepovoljnom slučajnom izboru okolina. Te instance su osenčene u tabeli.

instanca	n	m	avg t_{best} (s)	avg t_{tot} (s)	sol _{best}	agap (%)	σ (%)
GKD-c_10	500	50	566.8802	2004.0472	16345.03212	0	0
GKD-c_1	500	50	301.9495	2003.9376	16632.13844	0	0
GKD-c_2	500	50	324.1874	2003.8319	16466.1541	0.3952499995	0.3227202732
GKD-c_3	500	50	351.6639	1462.4301	16291.11558	0	0
GKD-c_4	500	50	267.214	1190.4412	16392.28983	0	0
GKD-c_5	500	50	380.0597	1402.8694	16633.69477	0	0
GKD-c_6	500	50	615.6064	1371.4874	16370.79536	0.5362045403	0.2924354079
GKD-c_7	500	50	348.6925	1476.3979	16404.70156	0.02999954	0.0099998467
GKD-c_8	500	50	281.0562	1469.0065	16514.38144	0	0
GKD-c_9	500	50	656.3047	1499.6279	16301.38305	0.2065997216	0.2733018087
MDG-a_10	500	50	156.0471	1332.7966	6476.19	0.1870544255	0.0623514752
MDG-a_1	500	50	256.9451	1358.4213	6398.84	0.6317551306	0.5200542157
MDG-a_2	500	50	173.3187	1356.7156	6432.83	0.1596964322	0.4790892966
MDG-a_3	500	50	1814.3899	1905.2132	6439.53	0.1383641353	0.4150924058
MDG-a_4	500	50	194.8343	1313.1923	6403.74	0.1685889808	0.2103227832
MDG-a_5	500	50	380.2829	1503.4237	6406.61	0.9732448206	0.3244149402
MDG-a_6	500	50	498.2157	1247.7577	6420.61	1.3109969302	0.5165807108
MDG-a_7	500	50	484.6	1410.1719	6423.29	1.0058241182	0.6357653982
MDG-a_8	500	50	255.9718	1333.3124	6506.86	1.3187774134	0.9466481724
MDG-a_9	500	50	754.4082	1252.2288	6343.99	0.2168036204	0.1003994819
MDG-b_10	500	50	156.4406	1848.5544	641177.9	0.7174143089	0.2601578912
MDG-b_1	500	50	475.4955	1332.1329	641777.21	1.1540853874	0.5638329571
MDG-b_2	500	50	370.9203	1606.4172	643308.46	1.1322827932	0.9464109641
MDG-b_3	500	50	678.2967	1667.5859	634644.54	0.5339080046	1.1785183221
MDG-b_4	500	50	442.7157	1271.1812	634293.08	2.9185E+057	5.8370E+057
MDG-b_5	500	50	402.7673	1178.4582	633194.82	2.9236E+057	5.8472E+057
MDG-b_6	500	50	166.2001	1131.3563	637200.47	2.9052E+057	5.8104E+057
MDG-b_7	500	50	360.6906	1187.2578	645794.37	4.2998E+057	6.5681E+057
MDG-b_8	500	50	192.2595	899.281	640957.6	4.3323E+057	6.6176E+057
MDG-b_9	500	50	306.5102	1264.5592	635805.2	4.3674E+057	6.6712E+057
SOM-a_10	25	7	2.6125	6.6763	158	0	0
SOM-a_1	25	2	1.3048	3.4343	9	0	0
SOM-a_2	25	2	0.6698	1.7623	9	9.99911E-011	3.33304E-011
SOM-a_3	25	2	0.2982	1.6759	9	0	0
SOM-a_4	25	2	0.3312	1.6346	9	0	0
SOM-a_5	25	2	0.6448	1.6606	9	0	0
SOM-a_6	25	7	2.4465	6.8838	153	0	0
SOM-a_7	25	7	1.472	6.5071	150	0	0
SOM-a_8	25	7	6.2708	7.4634	144	0	0
SOM-a_9	25	7	2.199	8.2042	147	0	0
SOM-b_10	300	60	809.6313	2001.2546	8239	0.0327709674	0.0903482141
SOM-b_1	100	10	1385.5911	2000.2265	333	3.5135135135	2.0764196893
SOM-b_2	100	20	2000.1547	2000.2628	1178	1.2139219015	0.9936440647
SOM-b_3	100	30	2000.1378	2000.239	2457	1.9983719984	1.3245284077
SOM-b_4	100	40	2000.0875	2000.2017	4137	0.3891708968	0.4407616225
SOM-b_5	200	20	313.3893	2000.816	1111	2.6372637264	1.4872200048
SOM-b_6	200	40	137.2598	2000.7022	4031	2.1706772513	1.4466656509

Tabela 1: Rezultati izvršavanja VINS matheuristike nad MDPLIB setom problema maksimalne različitosti.

Postignute vrednosti funkcije cilja nisu bolje od referentnih rezultata postignutih na istim instancama problema, što se može delimično objasniti manjim dozvoljenim vremenom izvršavanja (koje je u originalnim testovima prijavljeno da je bilo 2 sata).

Drugo potencijalno objašnjenje bi bilo da izabrane okoline (N1 ... N10) nisu optimalne za ovaj problem i da na neki način usporavaju optimizaciju. Ta neoptimalnost se može primetiti i u prosečnom vremenu za nalaženje najboljeg rešenja koje u nekim slučajevima pređe granicu vremena u kome se prelazi na veličinu okoline 100%, u kom slučaju se algoritam svede na običnu primenu MIP rešavača na problem. U testovima prikazanim u tabeli 1, to nije bila česta pojava, tj. većina optimalnih rešenja je pronađena u prvih 1000 sekundi koliko traju optimizacije u okolinama koje zadaje matheuristika.

5. Zaključak

U ovom radu predstavljena je Variable Intensified Neighborhood Search matheuristika, koja predstavlja generalizaciju Variable Neighborhood Decomposition Search matheuristike. Generalizacija se ogleda u tome što se uvodi mogućnost menjanja i kontrole "oblika" okolina na kojima se pretraga fokusira, kao i proporcionalno povećanje vremenskog ograničenja za lokalnu pretragu rešenja u zavisnosti od veličine okoline.

Postignuti rezultati nisu bolji od referentnih, ali zbog ograničenog broja testova, još je nepoznat najoptimalniji set okolina za rešavanje problema maksimalne različitosti koji su razmatrani u ovom radu. Dalje istraživanje bi se moglo fokusirati na nalaženje boljih konfiguracija okolina ili njihovog redosleda, uz eventualnu automatizaciju tog procesa.

6. Reference

1. Fischetti, M., & Lodi, A. (2003). Local branching. *Mathematical Programming*, 98(1-3), 23-47.
2. Glover, F., & Woolsey, E. (1974). Technical note—Converting the 0-1 polynomial programming problem to a 0-1 linear program. *Operations research*, 22(1), 180-182.
3. Hansen, P., Mladenović, N., & Urošević, D. (2006). Variable neighborhood search and local branching. *Computers & Operations Research*, 33(10), 3034-3045.
4. Kuo, C. C., Glover, F., & Dhir, K. S. (1993). Analyzing and Modeling the Maximum Diversity Problem by Zero-One Programming*. *Decision Sciences*, 24(6), 1171-1185.
5. Lazić, J., Hanafi, S., Mladenović, N., & Urošević, D. (2010). Variable neighbourhood decomposition search for 0–1 mixed integer programs. *Computers & Operations Research*, 37(6), 1055-1067.
6. Mitrović-Minić, S., & Punnen, A. P. (2009). Local search intensified: Very large-scale variable neighborhood search for the multi-resource generalized assignment problem. *Discrete Optimization*, 6(4), 370-377.
7. MDPLIB, <http://www.opticom.es/mdp/>