MAT-KOL (BANJA LUKA)
ISSN 0354-6969 (p), ISSN 1986-5228 (o)
VOL. XXI (2)(2015), 117-129
http://www.imvibl.org/dmbl/dmbl.htm

Rješavanje lokacijskog problema ograničenih kapaciteta primjenom algoritma promjenjivih okolina i algoritma roja čestica

Marko Djukanović[1]

Sažetak

Ovaj rad opisuje lokacijski problem ograničenih kapaciteta (CPLP) i njegove primjene u svakodnevnom životu. Za rešavanje problema CPLP koriste se heuristike promjenjivih okolina i rojeva čestica i daje se opis rješavanja ovog problema pomoću navedenih heuristika.

Ključne riječi: Variable neighbourhood search, Particle swarm optimization, lokacijski problem ograničenih kapaciteta, mip, CPLEX

Abstract

The paper describes Capacitated Plant Location Problem and its applications in everyday life. We use heuristics Variable Neighborhood Search and Particle Swarm Optimization for solving CPLP and we give description for solving this problem using these heuristics.

Categories and Subject Descriptors (according to AMS MSC (2010): 90C11 Mixed integer programming, 90C27 Combinatorial optimization, 68T20 Problem solving (heuristics, search strategies, etc.)

General terms: heuristic, mip, search strategy, optimization software, location analysis.

Key words and phrases: Variable Neighbourhood Search, Particle Swarm Optimization, Capacitated plant location problem, mip, CPLEX.

1 UVOD

U radu se opisuje način rješavanja lokacijskog problema ograničenih kapaciteta primjenom metaheurističkih metoda Variable Neighbourhood Search(VNS) i Particle Swarm

[1] Prirodno-matematički fakultet, Mladena Stojanovića 2, 78000 Banja Luka,
e-mail: marko.djukanovic@ymail.com

Optimization(PSO). Biće opisana i hibridizacija ove dvije metode. U uvodu se iznose osnovne činjenice o problemu CPLP, iznosi se istorijat problema, opis problema, matematički model kojim je predstavljen kao i njegova primjena. Daje se i konkretan primjer koji se vezuje za CPLP. U poglavljima koja slijede su izložne osnove VNS-a i PSO-a kao i način rješavanja CPLP-a pomoću ovih heuristika. Dati su pseudokodovi koji čine kostur algoritama pomoću kojih se testiraju instance problema. Testni rezultati programa VNS-a, PSO-a i hibrida ove dve metode koji rješavaju CPLP su dati u odgovarajućim tabelama. Na kraju se upoređuju rezultati prethodno opisanih algoritama, analiziraju se dobijeni rezultati i razmatraju mogući pravci daljih istraživanja. Rad sadrži i preođenje rezultata dobijenih prethodno opisanim metaheuristikama i rezultata dobijenih primenom rešavača CPLEX na CPLP.

Capacitated Plant Location Problem(CPLP) je problem kombinatorne optimizacije i jedan od osnovnih lokacijskih problema. Problem je često rješavan u literaturi koja pokriva oblast kombinatorne optimizacije. Počeci rješavanja i formulacije ovog problema mogu se naći u [10] . Detaljan opis problema je dat u [12] . Prve heuristike koje rješavaju CPLP mogu se naći u [9] i [14] gdje se problem rješava pomoću *heuristic lagrangean* algoritma. Kasnije je problem rješavan pomoću *Tabu Search*-a(TS) u [11] , *localSearch*-a u [13]. CPLP se sastoji u određivanju rasporeda potecijalnih lokacija, koje imaju svoje kapacitete, da bi se uslužo svaki korisnik sa svim njegovim zahtjevima.

Zadatak kod CPLP-a je minimizirati fiksne cijene otvorenih skladišta sa ograničenim kapacitetima i cijenama snabdjevanja. Jasno je da se lokacije koje ćemo birati mogu modelirati sa cjelobrojnim(binarnim)varijablama odluke. Logičan je zahtjev da se ne može potražiti ništa u skladištu koje nije otvoreno, a ako je otvoreno, ono ne može ponuditi više od svog kapaciteta. Svaki kupac treba da zadovolji potražnju idući od lokacije do lokacije. Svako otvaranje skladišta zahtjeva određenu cijenu i troškove, pa treba obratiti pažnju pri njihovom odabiru. Zahtjeva se da bude otvoreno p skladišta. Potreba za rješenjem problema CPLP je ekonomske prirode, konkretno u postavljanu skladišta sa proizvodima(time i prodavnica). Cilj je da se potrošači snabdiju proizvodima(u kojoj količini žele), ali da cijena otvaranja skladišta zajedno sa cijenom snabdjevanja tim proizvodom u skladištu bude minimalna. Treba da se izračuna u kom postotku se kupac može snabdjeti proizvodima u svakom skladištu. Te promjenjive su neprekidnog tipa(procentualne promjenjive), to jest njihova vrijednost je između 0 i 1. Binarne promjenjive odluke definišu otvorenje određenog skladišta. Napomenimo da je CPLP NP težak problem(vidjeti u [12]).

Primjene problema:

Primjena ovog problema može biti opisan sljedećim primjerom: Poslodavac otvara nekoliko novih skladišta u gradu. Postoji nekoliko lokacija na kojima on može da postavi skladišta. Neka su skuplja za otvaranje, a neka jeftinija(može zavisiti od lokacije skladišta u gradu). Pretpostavimo da poslodavac zna potražnju svakog kupca(teorijski nije moguće znati, ali kroz neku statisiku/anketu koja se prethodno uradi mogu se približno ocijeniti ove vrijednosti). Sada treba da se odredi gdje da se postave lokacije skladišta(zahtjev je

da ih bude ukupno p), da se zadovolje potražnje svakog kupca(eventualno kada obiđe sve lokacije) , a da cijena otvaranja skladišta zajedno sa cijenom plaćanja potražnje kupca po postrojenju bude minimalna.

2 MATEMATIČKA FORMULACIJA PROBLEMA I PRIMJENA

Definišemo problem CPLP matematičkim modelom na sljedeći način (prema [12]):

Neka je dat skup korisnika U i skup snabdjevača(skladišta) V. Uvedimo sljedeće oznake:
c_{ij}- označava cijenu snabdjevanja kupca j∈V snabdjevačem i ∈U
d_i- zahtjev kupca(potražnja)
p- broj željenih snabdjevača koja trebaju da budu pristupačna kupcima.
q_j- kapacitet snabdjevača
f_j- fiksna cijena snabdjevača(skladišta) na lokaciji j

Potrebno je minimizovati funkciju:

$$\sum_{i\in U}\sum_{j\in V} c_{ij}x_{ij} + \sum_{i\in U} f_i y_i,$$

uz sljedeća ograničenja:

$$\sum_{i\in U} x_{ij} = 1,\ za\ sve\ j \in V,$$

$$\sum_{j\in U} d_j x_{ij} \le q_i y_i,\ za\ sve\ i \in U,$$

$$y_i \in \{0,1\},$$

$$0 \le x_{ij} \le 1.$$

3 METODA PROMJENJIVIH OKOLINA ZA RJEŠAVANJE CPLP-A

Metoda promjenjivih okolina(Variable neighbourhood search) je predložena od strane Mladenovića i Hansena(1997)-u [5]. Bazira se na sistematičnoj promjeni okolina koja uključuje i lokalnu pretragu. Okoline zavise od oblasti domena problema koji se rješava. Označimo sa d metriku u prostoru rješenja X datog problema. Metrika je funkcija $d : X^2 \longrightarrow R$. Strukture okolina mogu biti definisane pomoću više metrika. K-ta okolina

rješenja $x \in X$ definiše se obično kao skup:

$$N_k(x) = \{y \in X | d(x,y) \leq k\} \text{ [1]}$$

Primijetimo da je $N_k(x) \subset N_{k+1}(x)$ za sve $x \in X$. Osnovna varijanta VNS-a za problem CPLP-a se sastoji u sljedećem:
Početna inicijalizacija problema je data slučajanim binarnim nizom dužine broja potecijalnih skladišta (u nastavku rada koristimo oznaku *facilities* za broj skladišta). Početni niz je niz nula. Uslov je da na p mijesta u nizu budu jednince.

Generišu se slučajni brojevi između 0 i *facilities* koji označavaju poziciju jednice u nizu. Korišten je i način generisanja početnih lokacija pomoću pohlepnog(greedy) algoritma. Ovaj algoritma može da bude korišten i pri nalasku dopustivih rješenja u domenu pretrage. Radi na principu nalaženja najjeftinijih skladišta za otvaranje(pomoću sortiranja). Uspostavlja se m najjeftinijih skladišta ($m < p$). Ostala skladišta (ima ih p)izgenerišemo na slučajan način. Uspostavljena lokacija(binarni niz sa p jedinica) će biti početno rješenje.

Za k-okolinu dopustivog rješenja x uzet je skup $N_k(x)$- skup svih binarnih nizova dužine *facilities* koji se razlikuju po koordinatama na najviše k mijesta od x ili simbolički zapisano:

$$N_k(x) = \{y \subseteq \{0,1\}^{facilities} | \sum |x_i - yi| \leq k\}$$

Varijabla k_{max} označava maksimalnu vrednost parametra k(maksimalna razlika tačke u bitima u odnosu na tačku x). Odabran je kriterijum zaustavljanja na k_{max} ili t_{max}-maksimalno CPU vrijeme izvršavanja programa. Po dostizanju parametara algoritam se završava i dotad najbolje rješenje se uzima za krajnje(optimalno). Glavni koraci algoritma su zapisani u pseudikodu ***VNS shema***.

VNS shema
Ponavljaj dok se ne zadovolji zaustavni kriterijum
1) Postavi $k \leftarrow 1$
2) Ponavljaj sve dok vrijedi k$\leq k_{max}$
a) Razmrdavanjanje. Generiše se x' $\in N_k(x)$ slučajnim načinom.
b)Pokretanje lokalne pretrage sa inicijalnim rješenjem x'.
Označimo lokalni min sa x".
c)Pomicanje u rješenju ili ne.
Ako je x" bolje od x, pomičemo pretagu ka x" (x"$\leftarrow$ x) i nastavljamo u $N_1(x)$ pri čemu je k=1, inače radimo pomak i proširujemo pretragu na k+1(k+1 $\leftarrow$ k).[5]

Kvalitet uspostavljene lokacije(fitnes) se računa pomoću Cplex-a koji rješava neprekidni linearni potproblem početnog problema. Način odabira tačke iz $N_k(x)$ u algoritmu je pojednostavljen tako da se bira $\frac{k}{2}$ slučajnih brojeva u intervalu 0..*facilities* koji na toj poziciji mijenjaju nulu u jedinicu i obrnuto.

Tako ostaje isti broj jednica u nizu(lokaciji), a razlika u broju bita je maksimalno k_{max}(što je i uslov). Lokalna pretraga koja se vrši u VNS-u je ograničena uslovom ako

nema popravke rješenja u 100 pokušaja odabira tačaka, prestaje pretraživanje i najbolje do tada nađeno lokalno rješenje se poredi u VNS metodi sa trenutno najboljim rješenjem. Ako je ono bolje, VNS starta najužom pretragom od date tačke(k=1), ili se pretraga proširuje(tj. k postaje k+1- okolina od x se proširi).
Procedura LocalSearch je zasnovana na običnoj lokalnoj pretrazi. U okolini tačke x su tačke koje se razlikuju od nje za manje od k_max bita(pozicija). Ako se nađe bolje rješenje x_nova u odabranoj okolini *LocalSearch* pretražuje novu okolinu $N_{k\_max}(x\_nova)$.
Prekid pretrage je određen na osnovu sljedećih parametara:
TimeLim(50) u peudokodu označava ograničenje izvršenja algoritma na 50s, parametar *StepMin*(m)– ako u m koraka nije bilo napretka u algoritmu prekidamo izvršenje. Parametar m je odabran bez prethodnog ispitivanja. $m$je fiksna veličina i ne zavisi od veličine problema u ovom radu(daljnje istraživanje može teći u smjeru ispitivanja ovih parametara radi poboljšanja algoritma). $Stop$ kriterijum nastupa kada nema poboljšanja u 100/500 koraka algoritma malih/velikih instanci. Za male instance se uzimaju instance $inst$19-$instA$26, a za velike instance $instA$57-$instA$38 redosljedom kao u tabeli 5.1.

Algorithm 1 VNS

VNS algoritam CPLP

1: while ne zadovolji zaustavni kriterijum $stop$
2: postavi k←1
3: while k≤ k_{max} do
4: Generiši x'∈ $N_k(x)$ slučajno
5: min=$MIP\_solve(x')$
6: x"=LocalSearch(x',TimeLim(50),StepMin(100)) //vraca localni minimum iz pretrage
7: x"<min
8: min = min_loc
9: k=1
10: ELSE
11: k=k+1;
12: END

4 METODA ROJEVA ČESTICA ZA RJEŠAVANJE CPLP

PSO(Particle swarm optimization) je heuristika zasnovana na grupi i njenoj inteligenciji. Razvili su je Dr Eberhart i Dr Kennedy 1995 godine([6]). Ona je evolutivna heuristika zasnovana na korištenju populacija sačinjenoj od niza čestica. Preživljavanje čestice zavisi od socijalnog ponašavanja, a ne fitnesa same čestice(što je slučaj kod genetskih algoritama). Algoritam je prirodno zasnovan na imitiranju usklađenog leta ptica. U algoritmu se generiše početna generacija rješenja(roj) pomoću koje se traži optimalno rješenje update-ujući položaj čestica u prostoru rješenja. Čestica se kreće po prostoru rješenja koristeći ne samo svoje iskustvo, nego i iskustvo čitave grupe(tj. roja). Svaka čestica pamti svoj najbolji optimum(*pbest*) do tog trenutka, ali zna i optimum čitavog roja do

tada(*gbest*). Na osnovu ta dva parametra i parametara bitnih za varijaciju rješenja(socijalni, kognitivni, slučajni, inertivni isl.)se formira kretanje čestice u datom trenutku. U ovom radu su korišteni svi navedeni parametri. Svaka čestica posjeduje brzinu kretanja po prostoru(koja je ograničena odozgo i odozdo jer bi u slučaju nepostojanja granica moglo doći do prebrzog/presporog kretanja po domenu pretrage i time pretrage malog dijela prostora odnosno preskakanja velikog broja lokalnih minimuma čime smanjujemo šanse za pronalazak globalnog). U svakom narednom koraku vrše se kretnje cjelokupnog roja po prostoru rješenja.

Napomenimo da se fitnes za datu lokaciju računa pomoću Cplex-a(ovdje je dovoljno i pustiti simplex metod koji možemo kodirati- detaljnije o programu u [8]). Pri uspostavljanju lokacija ostaje da se riješi neprekidni linearni problem). Ako česica populacije u narednom koraku dospije u bolje rješenje vrši se update čestice(update-uje se *lbest* ili *gbest*). Promjena brzine čestice *i* vrši se po sljedećoj formuli:

$$v[i] = \omega v[i] + c1 * rand() * (pbest[i] - tr[i]) + c2 * rand() * (gbest[i] - tr[i]) \quad (1)$$

$$trenutni\_položaj[i] = trenutni\_položaj[i] + v[i] \quad (2)$$

gdje su:
c1 -koeficijent kognitivnog učenja(uticaj islustva čestice na kretanje)
c2- koeficijent socijalnog učenja(uticaj iskustva grupe na kretanje čestice)
ω- faktor inercije(Shi and Eberhart),ω se obično bira iz (0,1)
$v[i]$- brzina i-te čestice populacije u datom trenutku
tr[i]- položaj čestice u datom trenutku.

Algorithm 2 PSO

Opis:Pseudokod PSO heuristike

1: For each particle do
2: Inicijalizuj česticu: uspostavi lokaciju x, fitnes=MIP_solve(x).
3: END;
4: Za sve čestice DO
5: Izračunaj brzinu prema formuli(1)
6: Update pozicije prema formuli (2)
7: Normalizacija lokacije
8: For each particle do
9: IF fitness vrijednost bolja nego najbolji fitnes (pBest) do sada
10: SET trenutni pBest je vrijednost nađenog fitnesa.
11: IF pBest čestice bolji nego gBest
12: SET trenutni gBest je pBest
13: While maximum iteracija ili nakon određenog vremena ili dostizanja prethodno definisanog broja u kojem nema napretka u alg.

PSO primjenjen na CPLP:

CPLP je problem mješovitog cjelobrnojnog programiranja. Upošten način rješavanja ovakvih problema pomoću PSO se može naći u [7]. Roj (swarm) se sastoji od $S = 100$ čestica (particles) koje *lete* u M-dimenizionom prostoru. M je varijabla *facilities*. Trenutna pozicija čestice je predstavljena M-dimenezionalnim vektorom(koji odgovara otvorenim skladištima). Svaka čestica odgovara potencijalnom rješenju problema u prostoru pretrage. Početna generacija je izgenerisana na slučajan način. Po uspostavljenju lokacija, ostaje nam riješiti niz potproblema početnog problema dobijen umetanjem uspostavljanih lokacija u početni problem. Potproblemi su linearnog tipa. To uradimo pomoću Cplex-a, ili puštanjem simplex metoda koju možemo sami generisati([8]). Ovim računamo *fitnes* date lokacije. Slična ideja je rađena u [2] na problemu *RCP*, kao i u uopstenom slučaju u [7]. Potom vršimo update čestica. Za parametre inercije uzeta je vrijednost $\omega = 0.8$, faktor socijalnog učenja je $f_q = 0.7$, dok je faktor kognitivnog učenja $f_p = 0.4$
Za kriterijum zaustavljanja je uzet broj $PSO_iteracija$ koja označava broj iteracija bez napretka(popravka trenutnog minimuma) u algoritmu. Update brzine čestice računamo po sljedećoj formuli(za koordinatu d):

$$v_d = \omega * v_d + f_p * \varphi_p(loc_i - x_d) + \varphi_q * f_p(g_d - x_d)$$

Brzina je ograničena varijablom V_{max} koja je po defaultu u programu podešena na vrijednost 10.
Po update-u položaja čestice lokacija ne mora biti kompatibilna. Za normalizaciju koordinata brzine čestice u datom položaju korištena je funkcija $Sigmoid(x)$. Funkcija je data sa:
$sig(v_{id}^{k+1}) = \frac{1}{1+exp(-v_{id}^{k+1})}$, iz koje dobijamo lokaciju:

$$x_{id}^{k+1} = \begin{cases} 0, & sig(v_{id}^{k+1}) \leq \rho \\ 1, & sig(v_{id}^{k+1}) \geq \rho \end{cases}$$

Za ρ se najčešće uzima $\frac{1}{2}$.

5 HIBRODIZACIJA METODE PROMENLJIVIH OKOLINA I METODE ROJEVA CESTICA ZA RESAVANJE CPLP

Hibridizacija metoda postoji odavno. Ideja hibridizacije je u spajanju dvije heuristke (njihovih dijelova) u jednu radi poboljšavanja rezultata istih. Često se spajaju dijelovi dvije heuristike koje su komplementarne po pitanju prednosti da bi se poboljšao algoritam sa lošom stranom. Korištenje i kombinovanje više različitih heuristika osigurava bolju raznovrsnost rješenja populacije, te lakše napuštanje lokalnih optimuma. Više o hibridizaciji pogledati u [15].

Način hibridizacije odabran u ovom radu je sljedeći: Na 10 procenata čestica se primjenjuje heuristika VNS-a. Rad algoritma zasnovan je na kretanju čestica pomoću PSO-a, a zatim odabira 10% čestica na koje se primjenjuje VNS radi poboljšanja lokalnog minimuma istih($lbest$). VNS-om se ove čestice pomjeraju po prostoru radi poboljšanja

fitnesa. Zbog ograničenja resursa vrijeme izvršavanja VNS je ograničeno na 100s. Čestice na koje se vrše popravke VNS-a se generišu slučajno. Nakon završetka rada VNS-a na odabranim česticama, popravlja se lokacija čestica nakon čega ponovo primjenjujemo korak PSO-a sa update-om položaja.

Za PSO je korišteno 100 čestica, od kojih je na slučajnih 10 vršena VNS popravka. VNS se sastojao u istim koracima kao i prethodno opisana VNS procedura, stim da je vrijeme izvrăvanja skraćeno zbog vremenskih i prostornih ograničenja. Inertni, socijalni i kognitivni faktor ostao je isti kao i u PSO heuristici.

Algorithm 3 Pseudokod HyPsoVns

```
1: Inicijalizacija populacije početnih čestica 1,2,...,s
2: For each particle u populaciji do
3: Inicijalizuj početne vrijednosti svake čestice(pBest)
4: END;
5: odabir slučajno generisanog broja p
6: za svaku slučajno izabranu česticu i=1,2... 10%*|populacija| do
7: puštanje VNS-a na datu česticu
8: IF poboljšan pBest pomoću VNS-a date čestice
9: UPDATE čestice parametara lBest i položaj
10: END
11: za sve čestice Do
12: Izračunaj brzinu prema formuli(1)
13: Update pozicije prema formuli (2)
```

6 REZULTATI TESTIRANJA, ANALIZA REZULTATA I POREĐENJA

Programi su testirani na Java Eclipse Kepler na mašini HP Pavillon g6(2.6GHZ, i5) sa $Windows7$ operativnim sistemom. Instance koje su korištene su generisane modifikacijom instanci $CFLP$ problema(mogu se naći u [16]). Sastojao se u sljedećem pohlepnom pristupu: Generiše se veličina instance, kao i broj željenih skladišta (varijable $customers$, $facility$ i p). Generiše se random broj $korak$ (varijabla korak označava poziciju elementa od kog čitamo elemente instance koju modifikujemo). Elemente upisujemo u programske varijable- nizove $demand$, $fixed\_cost$, $capacity$ imatricu C_ij redom. Ispitujemo dopustivost instance pomoću CPLEX-a. Ako model nema rješenje, ponavlja se postupak generisanja slučajanog broja koji se dodjeljuje varijabli korak. Napomenimo da smo ovim pristupom izgenerisali gotovo sve instance u razumnom vremenskom periodu. Instance $instA49$, $instA66$ i $instA27$ nisu generisane, već su preuzete sa prethodnog linka i kao takve korištene u testiranju.

Parametri VNS-a rezultati i analiza:

Parametri VNS-a su podešeni na sljedeći način: VNS algoritam se prekida ako u prethodnih 500 pretraga nema popravke trenutno najboljeg rješenja. Za parametar

k_max je uzet $\frac{facilities}{2}$. Lokalna pretraga u VNS-u se zaustavlja ako u 50 pretraga okoline nema popravke trenutno najboljeg lokalnog rješenja. Pojedine instance su testirane nekoliko puta da bi se našli pogodni parametri koji daju bolja rješenja za suženo vrijeme izvršavanja(računalo se da sveukupno izvršavanje programa ne prelazi sat vremena za velike instance).

Svaki od algoritama je puštan po 5 puta na instancama (broj nađen na osnovu raspoloživih resursa). Postoji nekoliko instanci VNS-a gdje nije moguće da se odredi dopustivo rješenje (možemo opravdati lošom početnom inicijalizacijom rješenja). Dopustivi skup(koji je jako malen) je gotovo nemoguće naći imajući ograničene vremenske i ostale zaustavne kriterijume u algoritmu.

Rješenje za instance $instA37$ i $instA60$ nije tačno(ne nalazi se dopustivo rješenje). Ovo pripisujemo lošoj inicijalizaciji početnog rješnja koje je daleko od bilo kojeg feasible rješnja, lošem izboru okolina kao i premalom vremenu izvršavanja.

Parametri PSO-a rezultati i analiza:

Broj čestica za PSO je stavljen na 50 za male instance a 100 za velike(veličine su dobijene iz prethodnog testiranja pojednih malih i velikih instanci poštujući ograničenja u vremenskim resursima da dužina algoritma na velikoj instanci ne prelazi okvire između 30 minuta i jednog sata).

Variacioni parametri algoritma- socijalni, kognitivni i faktor inercije su odabrani bez prethodnog testiranja i iznose:

φ_p=0.4(kognitivni faktor)

φ_p=0.70(socijalni faktor)

ω=0.8(faktor inercije)

Maksimalna brzina čestice je podešena na 10. U toku testiranja primjetilo se da je socijalni faktor dosta uticajniji od kognitivnog. Ovo je intuitivno jasno, jer sa ovim uticajem ne dopuštamo nagomilavanje čestica blizu nekog rješenja, već raspršujemo čestice na razne strane domena pretrage(želimo da pretražimo što više razilčitih potencijalnih lokalnih minimuma, čime povećavamo šansu za nalaženje globalnog). U tabelama vidimo da je PSO-u potrebno više vremena da nađe rješenje. Primijetimo iz tabela da VNS heuristika daje ista ili bolja rješenja na malim i srednjim instancama, dok na većim PSO ima bolje rezultate.

Populacioni algoritmi mogu da "$zađu$" u pretragu i tako pronađu feasible rješenja koja VNS teško može da nađe(najviše zbog osobine da koristi LS). Međutim bez obzira na to na instanci $instA50$ PSO ne pronalazi niti jedno feasible rješenje. Drugo objašnjenje za to je da se heuristici daje premalo vremena da pronađe dopustovo(feasible= rješenja u domenu pretrage.

Parametri HyPsoVns-a rezultati i analiza:

Hibridizovana metoda PSO-a i VNS-a ima iste parametre kao i PSO opisan prethodno. Algoritam VNS-a je pušten na 10% čestica. Biraju se slučajno odabrane čestice ukupno njih 10 %. Na njih se pušta VNS u nastojanju da data čestica popravi svoj položaj. VNS u hibridizaciji je ograničen sa sljedećim parametrima:

Ako nema popravke položaja čestice u 100 iteracija, prekida se VNS. Vrijeme izvršavanja VNS metode je ograničeno na 100s. Ako nema popravke u lokalnoj pretrazi u

Tabela 6.1: Rezultati izvršavanja VNS-a matheuristike nad Inst setom problema

Naziv instance	Kupci/ Skladišta/ p	$Best_{sol}$	$t_{tot}(ms)$	agap	σ
inst19	5/5/3	3.923	104.0	0.0	0.0
inst23	8/6/4	5.31576	207.2	0.0	0.0
inst27	9/8/6	4.319673	445.4	0.0	0.0
instA1	12/8/5	5.96497	436.4	0.0	0.0
instA3	12/10/7	5.789826	626.2	0.0	0.0
instA6	20/15/9	12.1699746	1460.8	0.146583	0.0723950
instA11	30/15/10	26.9901730	707.0	1.0830977	0.84335
instA15	32/10/7	26.0905657	115.4	0.0	0.0
instA27	1000/1000/9	203484.82362	321223.0	0.0	0.0
instA26	40/15/8	23.42505209	284.0	1.74735103	1.2835138
instA57	1000/330/120	574801.96135	249607.4	0.92297717	0.8988149
instA64	1060/310/200	594053.60720	296272.6	0.639030	0.5527240
instA47	915/360/225	323250.5438410	237907.0	0.3328775	0.408918
instA46	905/340/220	319454.06339	194095.8	1.150295	0.77349
instA50	1100/200/100	$1.79769313E^{308}$	80777.0	0.0	0.0
instA54	1000/300/200	571292.5705316	207022.8	0.944513	0.6277896
instA65	1000/350/160	454258.5251853	187053.8	2.7361	1.561810
instA66	1100/350/190	631716.080506	668743.4	0.5852716	0.881514
instA63	1000/300/160	559394.4975579	185316.2	1.24417	1.15308
instA62	1050/290/190	559394.497557	616553.8	0.49307	0.5965364
instA37	1000/200/90	$1.79769313E^{308}$	186751.4	0.0	0.0
instA39	1100/300/100	456480.8997701	130186.4	1.2432	0.9081890
instA40	1015/250/150	446113.7428982	2271109.4	1.2800	1.3183
instA44	850/200/140	353022.69281460	129325.2	1.45947	1.1683344
instA42	890/250/100	458227.1644990	95778.6	2.544842	1.48337875
instA61	1050/290/149	486131.7597156	72315.0	1.110032	0.88310720
instA55	1040/300/200	405581.686838998	131287.4	0.77935626	0.68922315
instA51	1000/250/140	487199.97427479	114426.8	0.9048867	0.6580317
instA53	1000/250/200	417276.1134391	125039.4	0.435083	0.2905691
instA58	1000/295/125	1115754.3943641	176386.4	0.915176	0.7524300
InstA41	915/360/225	477923.175130	79075.4	3.57945	2.019796
instA38	2000/100/55	4115754.3943641	176386.4	0.9151765	0.7524300

vremenu od 15s , ona se prekida.

Ipak, zbog hibridizacije vrijeme izvršavanja algoritma se povećava. Zbog toga su prethodne restrikcije na VNS-u neophodne da se izvrše.

Rezultate VNS-a, PSO-a i hibridnog metoda(HyPsoVns)nalazimo u tabelama 5.1, 5.2 i 5.3 respektivno. Sa sivom bojom je označen algoritam sa najboljim rezultatom nad datom instancom. Na većim instancama se vide poboljšanja HyPsoVns-a u odnosu na PSO i VNS. Srednje odstupanje od najboljeg rješenja koje dobije hibridizovani metod je manje od istog koje dobije PSO i VNS na većini ispitanih instanci. Međutim vrijeme izvršavanja, kao što vidimo u tabelama, je drastično veće. Zbog toga ne treba očekivati drastične popravke rješenja PSO-a hibridizacijom sa VNS-om(zbog gore napravljene restrikcije VNS-a u hibridizaciji) za isti vremenski period izvršenja algoritama.

Poređenja heuristika sa optimalnim rješenjem CPLEX-a

Dobijeni rezultati VNS-a , PSO-a i HyPsoVns-a u većini velikih instanci ne dostižu optimum pomoću ovako opisanih algoritama, fiksiranih parametara algoritama, vremenskih ograničenja isl. Manje instance($inst19$- $instA26$)su jednake optimalnim rješenjima koje dobija CPLEX u sve tri tabele. Instance $instA49$, $instA66$ i $instA27$ CPLEX ne može da nađe. Vidimo da su sve tri heuristike našle rješenja na ovim instancama.

Tabela 6.2: Rezultati izvršavanja PSO matheuristike nad Inst setom problema

Naziv inst.	Kupci/ Skladišta/ p	$Best_{sol}$	$t_{tot}(ms)$	gen	agap	σ
inst19	5/5/3	3.923	1050.2	11.0	0.0	0.0
inst23	8/6/4	5.3157669	1289.4	11.0	0.0	0.0
inst27	9/8/6	4.3196732	1585.2	11.0	0.0	0.0
instA1	12/8/5	5.9649709	1856.8	11.0	0.0	0.0
instA3	12/10/7	5.78982675	3201.8	13.8	0.0	0.0
instA6	20/15/9	12.169974	5128.2	16.6	0.56667	0.79
instA11	30/15/10	26.990173	4232.2	15.8	0.5410271	0.4467
instA15	32/10/7	26.0905657	2546.4	9.6	0.0	0.0
instA27	1000/1000/9	203484.82362	1241469	6.4	0.0	0.0
instA26	40/15/8	24.995176428	1469.6	11.8	1.382558	1.089501229
instA57	1000/330/120	581999.673721	459884.4	4.8	0.48763171	0.5256990
instA64	1060/310/200	599040.710380	507171.0	3.4	0.48092773	0.32792728
instA47	915/360/225	324651.4321	635274.4	6.0	0.373079783	0.30808
instA46	905/340/220	322036.68447	589672.2	4.6	0.880743966	0.524454
instA50	1100/200/100	$1.79769313E^{308}$	493566.4	5.4	0.0	0.0
instA54	1000/300/200	574422.0858	480218.0	4.4	0.650743281	0.74206
instA65	1000/350/160	457157.89550	701788.6	4.6	1.56488195	1.067880
instA66	1100/350/190	639079.65916	6295831.2	3.8	0.39259458	0.3432585
instA63	1000/300/160	569911.10262	484359.0	5.0	0.94455	0.57019
instA62	1050/290/190	599427.60980	456195.4	5.4	0.54306	0.44086
instA37	1000/200/90	560829.757897	180290.8	4.8	4.1636225	2.90044751
instA39	1100/300/100	455004.28867036	387953.0	5.2	0.907125	0.9064
instA40	1015/250/150	452373.0204093	443566.2	3.6	0.532466	0.46370
instA44	850/200/140	353022.69281	129325.2	4.4	1.4594764	1.16833
instA42	890/250/100	454509.48206	95778.6	4.2	0.31110131	0.215597
instA61	1050/290/149	479764.37947	469667.2	3.6	1.313432	0.7156
instA55	1040/300/200	406312.90061	497157.4	5.0	0.3396087	0.3716
instA51	1000/250/140	482369.49722	390295.6	3.8	1.0912151	1.16247
instA53	1000/250/200	414654.577054	437092.8	5.4	0.528685210	0.3517
instA58	1000/295/125	1104781.22855	453124.2	5.4	1.10848	0.6335
InstA41	915/360/225	454413.82542	267049.8	5.2	0.463407	0.49987
instA38	2000/100/55	4665315.42136	183272.2	6.0	1.322233	1.44330

Tabela 6.3: Rezultati izvršavanja HyPsoVns matheuristike nad Inst setom problema

Naziv inst	Kupci/ Skladišta/ p	$Bets_{sol}$	$t_{tot}(ms)$	$agap$	σ
inst19	5/5/3	3.923	1050.2	0.0	0.0
inst23	8/6/4	5.31576696123	1289.4	0.0	0.0
inst27	9/8/6	4.3196732680	1585.2	0.0	0.0
instA1	12/8/5	5.9649709869	1856.8	0.0	0.0
instA3	12/10/7	5.7898267534	3201.8	0.0	0.0
instA6	20/15/9	12.169974683	5128.2	0.5666795	0.79136597
instA11	30/15/10	26.99017302	4232.2	0.54102715	0.44673829
instA15	32/10/7	26.09056578	2546.4	0.0	0.0
instA27	1000/1000/9	203484.823622	1241469	0.0	0.0
instA26	40/15/8	23.4250520911	1876.8	0.5837725	0.333839
instA57	1000/330/120	582354.626369	459884.4	0.4899262	0.516146254
instA64	1060/310/200	597039.8162338	507171.0	0.3331451	0.215323190
instA47	915/360/225	324384.75993	360632.6	0.8312842	0.5749250
instA46	905/340/220	319454.063397	322448.02	0.367351	0.33739637
instA50	1100/200/100	676902.221975	241703.89	14.560792	3.2332210
instA54	1000/300/200	574325.56585	222031.0	0.77199359	0.388402
instA65	1000/350/160	448539.318102	221399.8	0.181961	0.1511192253
instA66	1100/350/190	641468.95352	365805.0	0.5045522	0.465891662
instA63	1000/300/160	569928.6528730	248319.4	0.94944	0.61585067
instA62	1050/290/190	600537.7789936	456195.4	0.508442	0.34274265
instA37	1000/200/90	616873.54789198	180290.8	4.1636225	2.90044751
instA39	1100/300/100	452465.99203846214	232201.4	0.39640917	0.37132398
instA40	1015/250/150	448837.87624718394	177206.2	0.9674432	0.641259
instA44	850/200/140	353022.69281460927	129325.2	1.45947	1.16833445
instA42	890/250/100	458227.16449908906	95778.6	2.544842	1.48337875
instA61	1050/290/149	438013.6078881711	146334.6	1.872752	1.7405624
instA55	1040/300/200	402154.8861330524	227206.6	0.5854621	0.5339262
instA51	1000/250/140	482263.7042221583	167219.0	0.8700081	1.0297596
instA53	1000/250/200	414961.092366793	232112.8	0.403086	0.3887506
instA58	1000/295/125	1096266.3504584618	352223.6	1.44234373	0.9572807
INSTA41	915/360/225	454754.7767792202	163872.2	2.0223241	2.22699234
instA38	2000/100/55	4665315.4213625228	183272.2	1.322233	1.443300

7 ZAKLJUČAK

U ovom radu predstavljene su metaheuristike VNS, PSO, kao i njihova hibridizacija. Algoritmi su testirani na instancama *Inst.* Za manje instance rješenja se poklapaju sa CPLEX-om(od inst19 do instA26 u tabelama). Heuristike dobijaju rješenja na instancama $instA49$, $instA55$ $instA66$ na kojima CPLEX ne može da nađe rješenje zbog svoga ograničenja u memoriji. Parametri koji se biraju u datim heuristikama(naročito vremenski) sprečavaju nalaženje boljeg rješenja. Ovo istraživanje može da se nastaviti daljom analizom parametara koji utiču na rad algoritama, kao i hibridizacije sa drugim heuristikama sa ciljem popravljanja rješenja dobijenih u ovom radu.

LITERATURA

[1] Jasmina Lazić: *New variants of variable neighbourhood search for* $0-1$ *mixed integer programming and clustering*, Ph. D. Thesis, Brunel University, London 2010.

[2] Sung-Soo Kim, Gon Kim,Ji-Hvan Byeon, Javid Taheri: *Particle Swarm Optimization for Location Mobility Management*, International Journal of Innovative Computing, Informatrion and Control, 8(12)(2012), 8387-8398.

[3] Olivier de Weck, *Multidisciplinary System Design Optimization*, Lecture Notes, MIT; Available at http://ocw.mit.edu/courses/engineering-systems-division/esd-77-multidisciplinary-system-design-optimization-spring-2010/lecture-notes/

[4] Fred Glover, Gary A. Kochenberger: *Handbook of metaheuristics*, Springer Science and Business Media, 2003.

[5] Pierre Hansen, Nenad Mladenović: *Variable neighborhood Search*, Computers and Operations Research, 24(11)(1997), 1097-1100.

[6] Yuhui Shi: *Particle Swarm Optimization*, IEEE Neural Networks Society February 2004, pp. 8-13, Available on http://www.computelligence.org/download/pso.pdf

[7] Satoshi Kitayama, Keiichiro Yasuda: *A method for mixed integer programming problems by particle swarm optimization*, Electrical Engineering in Japan, 157 (2)(2006), 40-49. Translated from Denki Gakkai Ronbunshi, Vol. 125-C, No. 5, May 2005, pp. 813Ű820

[8] Kevin Wayne and Robert Sedgewick, Algorithms 4th Edition, http://algs4.cs.princeton.edu/65reductions/Simplex.java.html

[9] J. Barcelo, Å. Hallefjord, E. Fernandez, K. Jörnsten *Lagrangean relaxation and constraint generation procedures for capacitated plant location problems with single sourcing*, Operations-Research-Spektrum, 12(2)(1990), 79-88.

[10] Bitran, Gabriel R.; Chandru, Vijaya; Sempolinski, Dorothy E.; Shapiro, Jeremy F. Inverse optimization: An application to the capacitated plant location problem, Management Science, 27(10)(1981), 1120-1141.

[11] Osman, Ibrahim H.; Christofides, Nicos *Capacitated clustering problems by hybrid simulated annealing and tabu search*, Int. Trans. Oper. Res. 1(3)(1994), 317-336.

[12] Sridharan, R. *The capacitated plant location problem*. European Journal of Operational Research, 87(2)(1995), 203-213.

[13] Bruns, Arno *A local search heuristic for the two-stage capacitated facility location problem*. In: Fleischmann, Bernhard (ed.) et al., *Advances in distribution logistic*s. Springer. Lect. Notes Econ. Math. Syst. 460(1998), (pp. 143-164).

[14] Agar, M.C.; Salhi, S. *Lagrangean heuristics applied to a variety of large capacitated plant location problems*. The Journal of the Operational Research Society, 49(10)(1998), 1072-1084

[15] Heikki Maaranen, *On heuristic hybrid methods and structured point sets in global continuous optimization*, Jyväskylä studies in computing, 43(2004), 24 pp.

[16] http://www-c.eco.unibs.it/ guastaro/InstancesCFLP.html)

Primljeno u redakciju časopisa 17.8.2014; Revidirana verzija 22.11.2014;
Dostupno na internetu 24.11.2014.