12. Dvonivovski lokacijski problem sa neograničenim kapacitetima

Matei Jon Stanču, 1137/2015

2016-8-12

Sadržaj

1 Apstrakt

Razmotren je dvonivovski lokacijski problem sa neograničenim kapacitetima. Ovaj problem podrazumeva sistem koji omogućava izbor fabrika i skladišta iz skupa potencijalnih fabrika i skladišta, gde je svakoj fabrici odnosno skladištu pridružena cena uspostavljanja na potencijalnoj lokaciji, i skup zahteva potrošača koji mora biti zadovoljen iz uspostavljenih fabrika kroz uspostavljena skladišta tako da je ukupna cena celog sistema minimalna. U ovom radu biće predstavljeni matematički modeli koji opisuju ovaj problem i takođe u nastavku biće predstavljene nekoliko metaheurističkih metoda za rešavanje razmatranog problema. Korišćene metaheuristike su Metoda roja čestica, Simulirano kaljenje i hibridizacija ove dve heuristike. Na kraju će biti upoređeni rezultati svih heuristika međusobno kao i sa optimalnim rešenjima dobijenih rešavanjem problema pomoću CPLEX-a.

Ključne reči: TSUFLP, kombinatorna optimizacija, Metoda roja čestica, Simulirano kaljenje, hibridizacija.

2 Opis problema

Problemi lokacije i alokacije spadaju u grupu problema kombinatorne optimizacije i od velikog su značaja u oblasti kao što su upravljanje lancima snabdevanja, telekomunikacione mreže. U ove probleme spadaju različiti problemi u kojima je potrebno uspostaviti lokacije nekih resursa onosno postrojenja da bi se zadovoljile određene potrebe tj. ispunili određeni ciljevi. Na uspostavljanje postrojenja mogu uticati različiti faktori kao što su udaljenost od čvorova koje snabdevaju, kapacitet postrojenja, cena uspostavljanja postrojenja ili cena transporta od jednog do drugog čvora. Obično je pri odlučivanju potrebno minimizovati troškove snabdevanja ili maksimizovati profit, ili se može koristiti neki drugi kriterijum optimizacije. Dakle, cilj je odrediti optimalnu raspodelu elemenata pri čemu je potrebno da budu ispunjena izvesna ograničenja datog sistema.

Konkretno, u ovom radu je razmatran Dvonivovski lokacijski problem sa neograničenim kapacitetima (*eng.* Two-Stage Uncapacitated Facility Location Problem, TSUFLP) predstavlja prirodno proširenje problema Jednostavan lokacijski problem (*eng.* Simple Plant Location Problem, SPLP). Razlika između ova dva problema je u tome sto je u TSUFLP uveden jos jedna nivo potencijalnih lokacija na kojima se uspostavljaju skladišta koja predstavljaju posrednike u prevozu robe od fabrika do potrošača. Izbor lokacija fabrika i skladišta predstavlja samo jedan deo odlučivanja koji se mora napraviti u problemu, drugi deo predstavlja izbor fabrika i skladišta koja će učestvovati u prevozu robe kako bi zahtevi svih potrošača bili ispunjeni. TSUFLP ne podržava informaciju o kapacitetima fabrika i skladišta, odnosno svaka fabrika proizvodi neograničenu količinu proizvoda a skladište može da čuva neograničenu količinu datog proizvoda. U daljem tekstu su dati matematički modeli koji opisuju ovaj problem, kao i instance koje su korišćene u analizi problema, i kako su one generisane.

3 Matematički model

Postoje mnogi načini i tehnike za rešavanje TSUFLP kao što su svođenje na problem pakovanja (SPTSUFLP) [1], "Clique facet" [1], "Lift odd hole facet" [1] i "Lift fan facet" [1]. U ovom tekstu akcenat je stavljen na osnovnu formulaciju (BTSUFLP) [1] i na implementaciju raznih heurističkih metoda koje ga rešavaju.

Ulazni podaci TSUFLP su:

- Skup $K = \{1, \ldots, q\}$ potencijalnih fabrika sa cenama uspostavljanja $g_k > 0, k \in K$;
- Skup $J = \{1, \ldots, m\}$ potencijalnih skladišta sa cenama uspostavljanja $f_j > 0, j \in J$;
- Skup $I = \{1, \ldots, n\}$ potrošača sa zahtevima $D_i > 0$ jedinica proizvoda, $i \in I$;
- Cene $c_{ij} > 0$ prevoza jedne jedinice proizvoda od skladišta j do potrošača i, $i \in I$, $j \in J$;
- Cene $d_{jk} > 0$ prevoza jedne jedinice proizvoda od fabrike k do skladišta j, $j \in J$, $k \in K$;

Cilj je da se zadovolje zahtevi svih potrošaca, pri čemu je potrebno minimizovati ukupnu cenu čitavog sistema. Ukupna cena se dobija sabiranjem cena svih uspostavljenih skladišta i fabrika i cena prevoza robe od fabrika do skladišta odnosno od skladišta do potrošača.

3.1 Matematička formulacija

Matematička formulacija koristi dva skupa promenljivih x_{ijk}, $i \in I$, $j \in J$, $k \in K$, koje opisuju prevoz dela zahteva potrošača i koji je snabdevan iz fabrike k preko skladišta j. Binarne promenljive $y_j^{'}$, $j \in J$, imaju vrednost 1 ako, i samo ako je skladište j uspostavljeno, u suprotnom imaju vrednost 0, dok binarne promenljive $z_k^{'}$, $k \in K$, imaju vrednos 1 ako, i samo ako je fabrika k uspostavljena, inače imaju vrednos 0. Osnovna formulacija problema (BTSUFLP) je data na sledeći način:

$$\min \quad \sum_{j=1}^{m} f_j y_j^{'} + \sum_{k=1}^{q} g_k z_j^{'} + \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{q} C_{ijk} x_{ijk} \tag{1}$$

$$s.t. \quad \sum_{j=1}^{m}\sum_{k=1}^{q} x_{ijk} = 1, \quad \forall i \tag{2}$$

$$x_{ijk} \leq y_j^{'}, \quad \forall i \ \forall j \ \forall k \tag{3}$$

$$x_{ijk} \leq z_k^{'}, \quad \forall i \ \forall j \ \forall k \tag{4}$$

$$y_j^{'} \in \{0, 1\}, \quad \forall j$$

$$z_k^{'} \in \{0, 1\}, \quad \forall k$$

$$0 \leq x_{ijk} \leq 1, \ \forall i \ \forall j \ \forall k \ .$$

C_{ijk} predstavlja cenu prevoza celog zahteva potrošača i iz fabrike k preko skladišta j, i računa se sledećom formulom:

$$C_{ijk} = D_i(c_{ij} + d_{jk}),\ i \in I,\ j \in J,\ k \in K.$$

Izraz (1) predstavlja funkciju cilja koja minimizuje ukupnu cenu uspostavljanja sistema odnosno zbir cena uspostavljanja fabrika cena uspostavljanja skladišta i cene prevoza robe iz fabrika preko skladišta do potrošača. Ograničenja (2) obezbeđuju da je svaki potrošač snabdevan kroz jednu ili više fabrika-skladište kombinacija. Ograničenja (3) obezbeđuje da su svi potrošači snabdevani samo preko uspostavljenih skladišta. Ograničenja (4) obezbeđuje da su svi potrošači snabdevani samo iz uspostavljenih fabrika dok ostala ograničenja su ograničenja nad vrednostima promenljivih. Zaključujemo da ova formulacija koristi $mnq + m + q$ promenljivih i $2mnq + n$ ograničenja ne računajući ograničenja koja opisuju granice promenljivih. Opisani problem je NP-težak jer predstavlja opštiji slučaj SPLP [1], ili preciznije Jednonivovskog problema sa neograničenim kapacitetima (*eng.* Uncapacitated Facility Location Problem, UFLP) [2]. Ova dva problema su ekvivalentna [7] i za njih je već dokazano da su NP-teški u [5] i [6].

4 Metaheuristike

Izračunavanje optimalnih rešenja je od suštinskog značaja za mnoge optimizacione probleme u svim oblastima. U praksi, obično smo zadovoljni sa "dobrim" rešenjima koje nam pružaju heuristički odnosno metaheuristički algoritmi. Metaheuristike predstavljaju familiju približnih optimizacionih tehnika čija je popularnost značajno porasla u poslednjih par decenija i predstavljaju jednu od najuspešnijih i obećavajućih tehnika. Metaheuristike pružaju "prihvatljiva" rešenja u razumnom vremenu pri rešavanju teških i kompleksnih problema u nauci i inženjerstvu. Ovo objašnjava značajni porast interesovanja u domen metaheuristika. Za razliku od egzaktnih optimizacionih algoritama, metaheuristike ne garantuju optimalnost pronađenih rešenja, i za razliku od aproksimativnih algoritama metaheuristike ne definišu koliko je odstupanje pronađenih rešenja od optimalnih. Metaheurističke metode pretrage mogu biti definisane kao generalizacija heuristika na višem nivou apstrakcije, i koriste se kao šablon za definisanje strategija koje se koriste za rešavanje različith specifičnih problema optimizacije. Više o metaheuristikama može se naći u knjizi [3].

U ovom radu biće opisano nekoliko metaheuristika, Metoda roja čestica, Simulirano kaljenje i hibridizacija prethodne dve metaheuristike, kao i njihovu adaptaciju konkretnom problemu TSUFLP kojeg proučavamo. Na kraju će biti predstavljene tabele sa rezultatima izvršavanja pomenutih metaheuristika i poređenja sa rezultatima dobijenih pomoću CPLEX paketa.

4.1 Metoda roja čestica

Metoda optimizacije roja čestica (*eng.* Particle Swarm Optimization, PSO) predstavlja stohastičku metaheuristiku zasnovanu na populaciji i inspirisana "inteligencijom rojeva" koja se

pojavljuje u prirodi [8]. Ova metaheuristika opisuje socijalno ponašanje organizama u grupi kao što je jato ptica ili jato riba u potrazi za mestom sa dovoljno hrane. Zaista, u tim jatima zajedničko ponašanje formira se lokalnim pokretima bez ikakve centralne kontrole. PSO je uspešno dizajnirana za neprekidne probleme međutim kasnije je prilagođena i za diskretne probleme tako da može da se koristi i za probleme kombinatorne optimizacije.

Osnovni model podrazumeva roj formiran od N čestica koje se kreću kroz D-dimenzioni prostor pretrage. Svaka čestica i je kandidat rešenje problema i predstavljena je vektorom x_i iz prostora pretrage. Čestice imaju svoje pozicije i brzinu kretanja što određuje pravac kretanja i korak čestice. PSO iskorišćava saradnju čestica, uspeh nekih čestica će uticati na ponašanje susednih čestica. Svaka čestica prilagođava svoj položaj x_i ka globalnom optimumu u zavisnosti od sledećih faktora: najbolja posećena pozicija posmatrane čestice ($pbest_i$) označena sa $p_i = (p_{i1}, p_{i2}, \cdots, p_{iD})$ i najbolja posećena pozicija na nivou celog roja ($gbest$)(ili $lbest$, najbolja pozicija nekog podskupa roja čestica) označena sa $p_g = (p_{g1}, p_{g2}, \cdots, p_{gD})$. Vektor $(p_g - x_i)$ predstavlja razliku između trenutne pozicije čestice i i najbolje pozicije njene okoline.

4.1.1 Okolina čestice

Okolina se definiše za svaku česticu jer ona određuje socijalni uticaj između čestica. Postoje mnogi načini za definisanje okoline. U praksi se najčešće koriste sledeće dve strategije:

- **gbest strategija:** Ova strategija podrazumeva da se kao najbolji posećeni položaj uzima na nivou cele populacije čestica.
- **lbest strategija:** Ova strategija podrazumeva topologiju koja je pridružena roju. Naime, okolina čestice predstavlja skup neposredno povezanih čestica kao što je topologija kompletnog grafa, prstena, zvezde ili "mali svet graf".

Zbog jednostavnosti implementacije, u ovom radu je korišćena prva metoda. U zavisnosti od okoline vođa predstavlja česticu koja usmerava pretragu ka boljim delovima prostora pretrage. Čestica se sastoji od:

- x-vektora koji beleži trenutni položaj čestice u prostoru pretrage.
- p-vektora koji sadrži najbolju posećenu lokaciju od strane posmatrane čestice.
- v-vektora koji sadrži gradijent odnosno pravac kretanja čestice ako njeno kretanje ne remeti spoljašnja okolina.
- dve vrednosti prilagođenosti, x-prilagođenost opisuje prilagođenost x-vektora i p - prilagođenost opisuje prilagođenost p-vectora.

Čestice mogu biti posmatrane kao *automati* gde svaka nova vrednost čestice zavisi samo od prethodne vrednosti posmatrane čestice i od vrednosti čestica iz okoline.

4.1.2 Iterativni korak

U svakoj iteraciji svaka čestica primenjuje sledeće operacije:

- **Ažuriranje brzine:** brzina koja određuje veličinu promene u čestici pri svakoj iteraciji definiše se na seledeći način

$$v_i(t) = v_i(t-1) + \rho_1 C_1 \times (p_i - x_i(t-1)) + \rho_2 C_2 \times (p_g - x_i(t-1))$$

  gde su ρ_1 i ρ_2 dva slučajna broja iz intervala $[0, 1]$, a konstante C_1 i C_2 predstavljaju faktore učenja. Faktori učenja predstavljaju težnju cestice ka svom uspehu ili ka uspehu okoline. Parametar C_1 predstavlja kognitivni faktor učenja koji predstavlja težnju čestice ka svom uspehu dok je parametar C_2 socijalni faktor učenja i predstavlja težnju čestice ka uspehu okoline. Brzina određuje pravac i rastojanje koje čestica treba da pređe. Ova formula oslikava fundamentalni aspekt ljudskog društva gde socijalno-psihološke tendencije pojedinaca utiču na uspeh drugih. Primenom prethodne formule čestica će kružiti oko tačke koja je definisana kao:

$$\frac{\rho_1 p_i + \rho_2 p_g}{\rho_1 + \rho_2}$$

  Elementi vektora v_i ograničeni su sa maksimalnom vrednošću $[-V_{max}, +V_{max}]$ kako ne bi pretraga bila previše slučajna. Ako element vektora v_i premaši V_{max} ondosno $-V_{max}$ onda se vrednost datog elementa postavlja na V_{max} odnosno $-V_{max}$.
  Obično se u fromuli za ažuriranje brzine čestice dodaje jos i parametar inercije, tako da je formula sledećeg oblika:

$$v_i(t) = \omega \times v_i(t-1) + \rho_1 C_1 \times (p_i - x_i(t-1)) + \rho_2 C_2 \times (p_g - x_i(t-1))$$

  Parametar inercije određuje koliki uticaj prethodna brzina ima na sledeću. Za velike vrednosti parametra inercije uticaj prethodne brzine na sledecu će biti veliki i suprotno. Dakle, parametar inercije na neki način predstavlja izbor između lokalnog i globalnog pretraživanja tj. veliki parametar inercije podstiče globalnu pretragu odnosno diversifikaciju u celom prostoru pretrage dok mali parametar inercije podstiče lokalnu pretragu odnosno pretragu u okolini trenutnog položaja čestice.

- **Ažuriranje pozicije:** Svaka čestica ažurira koordinate u prostoru odlučivanja

$$x_i(t) = x_i(t-1) + v_i(t)$$

  a zatim prelazi na sledeću poziciju. Obično se u praksi PSO koristi za neprekidne probleme, zbog toga je u ovom slučaju, za problem kombinatorne optimizacije, korišćena

modifikacija PSO koja ima dve osnovne razlike u odnosu na verziju koja se koristi za neprekidne modele, to su preslikavanje između položaja čestice i rešenja problema, i model brzine čestice.

Kao kodiranje položaja čestice, odnosno rešenja problema, korišćena je binarna reprezentacija. Dakle, svakoj čestici su pridružena dva binarna vektora jedan K-dimenzioni koji odgovara skupu potencijalnih lokacija za fabrike i jedan J-dimezioni koji odgovara skupu potencijalnih lokacija za sladišta. Ako je u nekom vektoru i-ti bit jednak 1 to znači da je na i-toj lokaciji uspostavljena fabrika ili skladište, u suprotnom ako je jednak 0, to znači da na nije uspostavljena fabrka odnosno skladište na toj lokaciji. Rešenja koja sadrže nula vektore nisu dopustiva i zbog toga je u računanju funkcije cilja postavljen uslov da ako je bar jedan od ta dva vektora nula vektor onda je vrednost funkcije cilja beskonačno.

Model brzine može biti realan odnosno neprekidan, stohastičan ili zasnovan na listi poteza. Za ovaj problem izabran je stohastički model brzine za binarno kodiranje, odnosno sigmoidna funkcija koja računa verovatnoću da bitovi imaju vrednost jedan na osnovu brzine čestince. U literaturi [3] predložena je Sigmoid funkcija

$$S(v_{id}) = \frac{1}{1 + e^{-v_{id}}}$$

koja transformiše vrednosti brzine v_i u vrednosti iz interval $[0, 1]$, zatim uzme se slučajan broj iz istog intervala i ako je generisani slučajan broj manji od $S(v_{id})$, onda promenljiva x_{id} će biti inicijalizovana na 1, a inače će imati vrednos 0. Brzina se obično povećava kad je $(p - x)$ pozitivno tj. kada je $p_{id} = 1$ ili $p_{gd} = 1$ a smanjuje se kada je $p_{id} = 0$ ili $p_{gd} = 0$. Dakle, kada v_{id} raste onda i verovatnoća da $x_{id} = 1$ takođe raste.

- **Ažuriranje najbolje pozicije:** Svaka čestica ažurira najbolju lokalnu poziciju u slučaju da je nova pozicija bolja od najbolje zabeležene pozicije:

$$If\ f(x_i) < pbest_i\ then\ p_i = x_i;$$

i zatim ako je ta pozicija bolja od globalne odnosno najbolja na nivou okoline, ažurira se i ta pozicija:

$$If\ f(x_i) < gbest_i\ then\ g_i = x_i;$$

Kako je razmatrani problem tipa minimizacije, onda se svako novo rešenje koje daje manju vrednost funkciji cilja od trenutnog najbljeg rešenja, prihvata kao novo najbolje rešenje. Dakle u svakoj iteraciji svaka čestica menja položaj u zavisnosti od svog iskustva i od iskustva čestica iz okoline.

4.1.3 Izbor parametara

Broj čestica n i broj iteracija potrebnih za izvršavanje algoritma su suprotni parametri, odnosno što je više čestica uključeno u pretragu to je manje iteracija algoritma potrebno. Povećavanje veličine populacije poboljšava rezultate ali zato je vremenski zahtevnije. Većina PSO implementacija koristi interval od $[20, 60]$ za veličinu populacije, pri čemu je u ovoj implementaciji korišćena veličina populacije od 40 čestica.

Postoje mnogi izbori tipova i veličine okoline, pri čemu za sve tipove važi pravilo da velike okoline podstiču intenzifikaciju pretrage oko jedne tačke odnosno lokalnu pretragu oko najboljeg globalnog optimuma, a male okoline podstiču diversifikaciju pretrage odnosno pretragu kroz veći deo prostora pretrage. Prva varijanta ima veću brzinu konvergencije ali zato ima i veću verovatnoću da se javi preuranjena konvergencija jer je veliki broj čestica privučeno jednom najboljem globalnom rešenju, dok je u drugom slučaju suprotno i dobija se veći kvalitet rešenja. Dva ekstrema za okolinu veličine k su upotreba globalne populacije ($k = n(n-1)/2$, gde je n veličina cele populacije) i lokalna okolina strukture zasnovana na topologiji prstena za $k = 2$. Optimalan izbor veličine okolina zavisi od prirode problema, pri čemu je u ovom slučaju veličina okoline izabrana empirički na osnovu postignutih rezultata za različitie vrednosti ovog parametra.

Parametri ρ_1 i ρ_2 predstavljaju jačinu kretanja čestice ka lokalnom najboljem rešenju odnosno globalnom najboljem rešenju. Ova dva parametra su izabrana slučajno u nekom zadatom intervalu. Kada su ρ_1 i ρ_2 blizu 0, brzine definišu kretanje čestice koje je više glatko, dok sa druge strane, ako su ρ_1 i ρ_2 veliki onda čestice oštro osciliraju što podstiče više globalnu pretragu. Većina implementacija koriste vrednos 2.0 za oba ova parametra, dok je u ovoj implementaciji izbor parametara ρ_1 i ρ_1 slučajan u intervalu $[2.0, 2.5]$.

Gornja granica V_{max} bira se u zavisnosti od prirode i velicine problema, dok je u ovom slučaju granica izabrana empirički na osnovu vrednosti koje su davale najbolje rezultate. Tipična vrednost parametra inercije ω je inicijalizovana na 0.9 i zatim kako algoritam napreduje tako se parametar smanjuje do 0.4 ali je u ovoj implementaciji postavljen na 0.99 bez smanjivanja kako bi se smanjila verovatnoća preuranjene konvergencije.

Svi parametri se mogu dinamički menjati u zavisnosti od napretka algoritma kako bi se menjala pretraga između intenzifikacije oko jedne tačke i diversifikacije u celom prostoru pretrage. Takođe se i broj čestica može menjati, npr. ako neka čestica pokazuje loše rezultate bez poboljšanja u određenom broju uzastopnih iteracija, onda je nju poželjno izbaciti kako bi se ubrzao rad ostalih čestica koje napreduju i samim tim izvršavanje celog algoritma. Bez obzira na prednosti promenljivog broja populacije, u ovom slučaju izabrana je strategija bez menjanja broja čestica kako se ne bi dobila previše složena implementacija bez značajnih poboljšanja u performansama. Struktura predloženog PSO algoritma predstavljena je Algoritmom 1.

Algorithm 1. Particle Swarm Optimization

---

foreach *particle* $p_i, i = 1, ..., S$ **do**
    Randomly initialize value vector: $x_i \leftarrow Floor(U(0.0, 1.0) + 0.5)$;
    Set local best known position to initial position: $b_i \leftarrow xi$;
    Initialize particle velocity: $v_i \leftarrow U(-v_{max}, v_{max})$;
    if *particle* p_i *better than global best particle* g **then**
        $g \leftarrow p_i$;
    end
end
repeat
    foreach *particle* $p_i, i = 1, ..., S$ **do**
        foreach *dimension* $d = 1, ...M$, **do**
            $f_p \leftarrow U(0.0, 1.0)$;
            $f_g \leftarrow U(0.0, 1.0)$;

            $v_i \leftarrow \omega v_{i,d} + f_p(b_{i,d} - x_{i,d}) + f_g(g_{i,d} - x_{i,d})$;
            if $v_{i,d} > v_{max}$ **then**
                $v_{i,d} \leftarrow v_{max}$;
            else if $v_{i,d} < -v_{max}$ **then**
                $v_{i,d} \leftarrow -v_{max}$;
            end

            $u \leftarrow U(0.0, 1.0)$;
            if $u < Sigmoid(v_{i,d})$ **then**
                $x_{i,d} \leftarrow 1$;
            else
                $x_{i,d} \leftarrow 0$;
            end
        end
        if x_i *better than local best position* b_i **then**
            $b_i \leftarrow x_i$;
            if b_i *better than global best position* g **then**
                $g \leftarrow b_i$;
            end
        end
    end
until *maximum nonimproving iterations reached*

4.2 Simulirano kaljenje

Metaheuristiku Simulirano kaljenje (*eng.* Simulated Annealing, SA) osmislili su Armen G. Khachaturyan, Svetlana V. Semenovskaya, Boris K. Vainshtein 1979. godine [9] i imala je veliki uticaj u oblasti heurističkih pretraga zbog svoje jednostavnosti i efikasnosti u rešavanju problema kombinatorne optimizacije, a kasnije je bila proširena kako bi mogla da se koristi i za neprekidne optimizacione probleme. SA je zasnovana na principu statističke mehanike u kom proces kaljenja zahteva zagrevanje i zatim postepeno hlađenje materijala kako bi se postigla jaka kristalna struktura. Ako inicijalna temperatura nije dovoljno velika ili je hlađenje previše brzo onda se javljaju nepravilnosti (metastabilna stanja). U ovom slučaju, predmet koji se hladi neće da postigne termalnu ravnotežu na svakoj temperaturi. Jaki kristali se dobijaju sporim i pažljivim hlađenjem. SA algoritam simulira promene energije u sistemu izloženom procesu hlađenja dok ne konvergira ka stanju ravnoteže (stabilno zaleđeno stanje). Slično je i sa rešavanjem optimizacionog problema pomoću ovog procesa, funkcija cilja je analog energetskom stanju sistema, rešenje optimizacionog problema odgovara stanju sistema. Vrednosti promenljivih koje su predstavljene rešenjem odgovaraju položajima molekula u materijalu. Globalni optimum odgovara najbliže traženom kristalnom obliku dok lokalni optimum nagoveštava da je postignuto metastabilno stanje.

SA je stohastički algoritam koji omogućava da se pod određenim uslovima rešenje degradira. Cilj je pobeći iz lokalnog optimuma i time odložiti konvergenciju. SA je algoritam koji ne zahteva nikakvu prikupljenu informaciju tokom izvršavanja. Od izabranog početnog rešenja, SA se izvršava u nekoliko iteracija. U svakoj iteraciji bira se slučajan sused trenutnog rešenja. Zatim potezi koji su povoljniji odnosno rešenja koja dovode do povoljnije vrednosti funkcije cilja su uvek prihvaćena, inače sused je prihvaćen sa određenom verovatnoćom koja zavisi od trenutne temperature i od degradiranja ΔE funkcije cilja. ΔE predstavlja razliku između vrednosti funkcije cilja trenutnog rešenja i izabranog suseda od trenutnog rešenja. Kako algoritam napreduje tako se smanjuje verovatnoća prihvatanja poteza koji nisu povoljniji. Verovatnoća kojom se loši potezi prihvataju je data Bolcmanovom raspodelom:

$$P(\Delta E, T) = e^{-\frac{f(s^{'})-f(s)}{T}}$$

gde je s trenutno rešenje a s' sused trnutnog rešenja. SA koristi kontrolni parametar temperature kako bi odredio verovatnoću prihvatanja lošijih rešenja. Na određenim temperaturama vrši se određeni broj pokušaja. Kada se postigne uravnoteženo stanje temperatura se postepeno smanjuje u skladu sa šemom hlađenja tako da nekoliko lošijih rešenja bude prihvaćeno na kraju pretrage. Pseudokod heuristike SA je predstavljen algoritmom 2.

Algorithm 2. Simulated Annealing

Input: Cooling schedule;
Generation of the initial solution: $s \leftarrow s_0$;
Initializing starting temperature: $T \leftarrow T_{max}$;
Repeat
    Repeat at a fixed temperature
        Generate a random neighbor s';
        $\Delta E = f(s') - f(s)$;
        if $\Delta E < 0$ **then**
            Accept the neighbor solution: $s \leftarrow s'$;
        else
            Accept s' with a probability $e^{-\frac{\Delta E}{T}}$;
        end
    Until a number of iterations executed at each temperature
    Temperature update: $T \leftarrow g(T)$;
Until stopping criteria satisfied: $T < T_{min}$
Output: Best solution found;

Dva parametra kontrolišu proces pretrage, to su trenutna temperatura i broj izvršenih iteracija pri svakoj temperaturi. Pored osnovnih problema kao što su definicija okoline i generisanje inicijalnog rešenja, pri dizajniranju SA algoritma neophodno je izabrati dva bitna elemanta koji moraju biti prilagodjeni problemu koji se razmatra. To su:

- **Funkcija verovatnoće prihvatanja:** predstavlja glavni element algoritma SA koji omogućava lošijim rešenjima da budu prihvaćeni sa određenom verovatnoćom.
- **Šema hlađenja:** predstavlja temperaturu u svakom koraku algoritma SA i ima ulogu u efikasnosti i delotvornost algoritma.

U nastavku teksta dati su detaljniji opisi ova dva elementa SA algoritma.

4.2.1 Prihvatanje rešenja

SA algoritam moze da izbegne lokalni optimum pomoću funkcije verovatnoće prihvatanja lošijeg rešenja. Verovatnoća prihvatanja lošijeg rešenja je proporcionalna temperaturi T i inverzno proporcionalna promeni funkcije cilja ΔE. Zakon termodinamike tvrdi da je pri temperaturi T verovatnoća povećanja energije ΔE data izrazom $P(\Delta E, T) = exp(-\Delta E/kt)$, gde je k konstanta poznata kao Bolcmanova konstanta. Dakle, verovatnoća prihvatanja lošijeg rešenja data je sledećim izrazom:

$$P(\Delta E, T) = exp\left(\frac{-\Delta E}{kT}\right) > R$$

gde je ΔE razlika funkcije cilja za trenutno rešenje i suseda trenutnog rešenja, T je trenutna temperatura, a R je slučajan broj određen uniformnom raspodelom između 0 i 1.
Pri visokim temperaturama verovatnoća prihvatanja lošijeg rešenja je velika, ako je $T = \infty$, sva rešenja su prihvaćena, što odgovara slučajnom lutanju kroz prostor pretrage. Sa druge strane, pri niskim temperaturama, verovatnoća prihvatanja lošijeg rešenja je mala, ako je $T = 0$, tada se ne prihvata nijedno lošije rešenje i pretraga je ekvivalentna lokalnoj pretrazi. Na osnovu Bolcmanove raspodele verovatnoća prihvatanja velikog pogoršanja u vrednosti funkcije cilja opada eksponencijalno ka 0.

4.2.2 Šema hlađenja

Šema hlađenja definiše temperaturu T_i za svaki korak algoritma i ima veliki uticaj na uspeh optimizacije algoritmom SA. Ekspeimentalno je utvrđeno da performanse SA su veoma osetljive na izbor šeme hlađenja. Šema hlađenja ima 3 bitna elementa, prvi je inicijalna temperatura, drugi je stanje ravnoteže i treći je hlađjenje. U nastavku je opisan svaki od ova tri elementa.

- **Inicijalna temperatura.** Ako je početna temperatura jako visoka onda će pretraga manje više ličiti na lokalnu pretragu. Sa druge strane ako je inicijalna temperatura jako niska onda će pretraga manje vise ličiti na lokalnu pretragu sa prvim poboljšanjem. Dakle potrebno je izabrati sredinu između ta dva ekstrema. Pocetna temperatura ne sme biti mnogo visoka, kako se ne bi javljala slučajna pretraga na kratko vreme, ali dovoljno visoka da mogu biti prihvaćeni skoro svi susedi od trenutnog rešenja. Razlikujemo tri strategije koje rešavaju ovaj problem, *Prihvati sve*, *Devijacija prihvatanja* i *Odnos prihvatanja*. Za ovaj problem izabrana je prva strategija zbog jednostavnosti implementacije i ona podrazumeva da se početna temperatura postavi dovoljno visoko da funkcija prihvatanja može da prihvati svakoh suseda u početnoj fazi algoritma. Mana ove strategije jo to što je vremenski zahtevna.

- **Stanje ravnoteže.** Da bi postigli stanje ravnoteže na svakoj temperaturi, potrebno je izvršiti određeni broj proveravanja novih rešenja. Teorija nam ukazuje da broj iteracija na svakoj temperaturi može da bude eksponencijalan u odnosu na veličinu problema, što je veoma složeno da bi se primenilo u praksi. Broj iteracija mora biti u skladu sa veličinom instance problema i proporcionalan broju suseda trenutnog rešenja $|N(s)|$. Broj ovakvih provera može biti određen na dva načina, *Statički* ili *Prilagodljivo*. Za ovaj problem izabran je prvi način opet zbog jednostavnosti implementacije, ali po ceni vremenske složenosti. U statičkoj strategiji, broj provera je određen pre nego što počne pretraga. Dakle, deo y svih suseda $N(s)$ je pretraženo. Znači da broj generisanih suseda rešenja s je $y|N(s)|$, što je veće y to je vremenski skuplje izračunavanje ali su bolji rezultati.

- **Hlađenje.** U SA algoritmu temperatura se postepeno smanjuje tako da je

$$T_i > 0, \forall i, \ \lim i \to \infty = 0$$

Uvek postoje kompromisi između kvaliteta rešenja i brzine izvršavanja. Ako se temperatura sporije smanjuje onda će kvalitet rešenja biti bolji ali je veće vreme izvršavanja i suprotno. Temperatura T može da se ažurira na nekoliko načina, *Linearno*, *Geometrijsko*, *Logaritamsko*, *Veoma sporo smanjivanje*, *Nemonotono* i *Prilagodljivo*. Izabrana je *Geometrijska* šema koja podrazumeva ažuriranje temperature pomoću formule

$$T = \alpha T$$

gde $\alpha \in [0, 1]$. Ovo je najpopularnija funkcija hlađenja, pri čemu se u praksi pokazalo da je najbolje da α bude imeđu 0.5 i 0.99.

- **Kriterijum zaustavljanja.** Što se tiče kriterijuma zaustavljanja, teorija predlaže da je poslednja temperatura 0. U praksi kriterijum može biti kada verovatnoća prihvatanja dostigne zanemarljivu vrednost, određeni broj izvršenih iteracija bez promena u najboljem pronađenom rešenju ili ako je postignut određen predefinisan broj puteva kada je procenat suseda pri svakoj temperaturi prihvaćeno tj. podrazumeva se brojač koji se povećava svaki put kada iteracija završava sa manjim od određenog procenta prihvaćenih rešenja i resetuje se ako se pronađe novo najbolje rešenje, a ako dostigne određeni broj onda se SA algoritam zaustavja. Postignuta je finalna temperatura T_F je najpopularniji kriterijum zaustavljanja zbog čega je ova metoda bila izabrana u ovoj implementaciji. Ova temperatura mora biti mala (npr. $T_min = 0.01$).

4.3 Hibridizacija

Često se dešava da jedna metaheuristika nije dovoljna za nalaženje dobrih rešenja nekih teških problema pogotovo kada su u pitanju jako velike dimenzije, tada se primenjuje tehnika koja se zove hibridizacija. Hibridizacija podrazumeva kombinovanje dve ili više metaheuristika u cilju poboljšanja rezultata pretrage rešenja, i obično se to vrši na takav način da se upotrebljene metaheuristike nadopunjuju, odnosno jedna drugoj nadoknađuju nedostatke. Na primer, ako je prva upotrebljena metaheuristika podstiče diversifikaciju onda kao drugu je dobro izabrati metaheuristiku koja podstiče intenzifikaciju pretrage.

U ovom slučaju hibridizacija je vršena tako što je algoritam optimizacije Metoda roja čestica korišćen kao osnova hibridnog algoritma, dok se algoritam Simulirano kaljenje koristi kao dodatni element koji se poziva u svakoj iteraciji izvršavanja osnove. Takođe je funkcija ažuriranja brzine čestica korišćena kao element SA koja se poziva svaki put kada SA pronađe rešenje koje je bolje od trenutnog globalnog rešenja. Parametri algoritma PSO su inicijalizovani tako da podstiču što veću diversifikaciju, korišćena je populacija od 20 čestica kako ne bi

hibridni algoritam bio previše vremenski zahtevan jer je SA komponenta, koja se poziva u svakoj iteraciji, prilično zahtevna. Zbog toga za PSO je korišćena jedna okolina od svih 20 čestica sa jednim globalnim rešenjem na nivou celog roja. U svakoj iteraciji PSO, nakon ažuriranja brzina i položaja čestica i zatim računanja vrednosti funkcije cilja, poziva se SA komponenta za svako najbolje rešenje tekuće iteracije svake okoline, konkretno u ovoj implementaciji za jedno globalno rešenje na nivou celog roja i oko tog rešenja SA vrši intenzifikaciju pretrage. Parametri algoritma SA inicijalizovani su tako da ne spuste previše performanse celog hibridnog algoritma a da pri tom očuvaju kvalitet rešenja što više moguće. Temperatura kriterijuma zastavljanja je postavljena na veću vrednost nego inače a funkcija geometrijskog hladjenja je modifikovana tako da ubrza proces dobijanja metastabilnog stanja. Opis hibridne heuristike je predstavljen algoritmom 3.

Algorithm 3. Hybrid (PSO and SA)

---

foreach *particle* $p_i, i = 1, ..., S$ **do**
    Randomly initialize value vector: $x_i \leftarrow Floor(U(0.0, 1.0) + 0.5)$;
    Set local best known position to initial position: $b_i \leftarrow xi$;
    Initialize particle velocity: $v_i \leftarrow U(-v_{max}, v_{max})$;
    if *particle* p_i *better than global best particle* g **then**
        $g \leftarrow p_i$;
    end
end
repeat
    foreach *particle* $p_i, i = 1, ..., S$ **do**
        Update velocity v_i;
        Update position x_i;

        if x_i *better than local best position* b_i **then** $b_i \leftarrow x_i$;
        if b_i *better than global best position* g **then** $g \leftarrow b_i$;
    end
    foreach *neighborhood* $h_i, i = 1, ..., N$ **do**
        Apply simulated annealing on global solution s_i;
    end
end
until *maximum nonimproving iterations reached*

---

5 Eksperimentalni rezultati i analiza

Sva izračunavanja su vršena na računaru sa procesorom iz familije Intel i3 procesora sa 4 jezgra, RAM memorijom od 4 GiB na 32-bitnom operativnom sistemu Linux distribucije Manjaro. U nastavku su date tabele sa rezultatima izvršavanja heuristika za instance malih, srednjih i velikih dimenzija, kao i tabele sa rezultatima izvršavanja CPLEX programa.

5.1 Instance

Za TSUFLP problem generisano je 60 instanci od kojih su 40 dobijene modifikacijom instanci TSCFLP problema a ostalih 20 generisano je pomoćnim programom "Generator". Prvih 40 generisano je tako što su TSCFLP instancama dodate cene za uspostavljanje fabrike. Cene su generisane pomoćnim programom "Random" koji kao argument komandne linije ima broj koji označava broj cena koje treba generisati. Cene se biraju slučajno uniformnom raspodelom iz intervla

$$[3 \cdot fc\_min,\ 3 \cdot fc\_max]$$

gde su fc_min i fc_max minimalna odnosno maksimalna cena uspostavljanja skladišta.

U ostalim instancama svi podaci su generisani slučajno, takođe uniformnom raspodelom, po uzoru na instance iz TSCFLP problema. Granice intervala iz kojih su podaci birani su postavljene na minimalne odnosno maksimalne vrednosti koje mogu biti pronađene u TSCFLP instancama za odgovarajuće podatke. Instance su podeljene u tri grupe: instance malih, srednjih i velikih dimenzija. Instance su obeležene sa tri cifre koje označavaju redni broj instance, dimenziju i verziju. Npr. ako imamo instancu sa kodom xyz tada je:

$$x - redni\ broj\ instance$$
$$y - dimenzija\ instance$$
$$z - verzija\ instance$$

Verzija instance označava razliku između posmatrane instance i instance sa verzijom 1. U nastavku su date moguće verzije i po čemu se one razlikuju od verzije 1:

$$2 - razlika\ u\ zahtevima\ potrosaca$$
$$3 - razlika\ u\ cenama\ uspostavljanja\ skladista$$
$$4 - razlika\ u\ cenama\ uspostavljanja\ fabrika$$
$$5 - razlika\ u\ cenama\ prevoza\ od\ skladista\ do\ potrosaca$$
$$6 - razlika\ u\ cenama\ prevoza\ od\ fabrike\ do\ skladista$$

Instance sa dimenzijama 1, 2 i 3 su male, zatim 4, 5, 6 su srednje a 7 i 8 su velike. U nastavku su date oznake dimenzija i njihove veličine u formatu (Portošači x Skladišta x Fabrike):

$$
\begin{array}{ccc}
0-(10x8x4) & 1-(25x10x5) & 2-(50x10x5) \\
3-(100x25x5) & 4-(100x50x5) & 5-(100x50x10) \\
6-(500x25x5) & 7-(500x50x5) & 8-(350x50x10)
\end{array}
$$

5.1.1 Primer instance

Svaka instanca sadrži dimenziju, niz zahteva potrošača, niz cena uspostavljanja fabrika, niz cena uspostavljanja skladišta, matricu cena prevoza proizvoda iz fabrika u skladišta i matricu cena snabdevanja potrošača iz skladišta. U nastavku je dat primer instance sa kodom 000:

10 8 4

29 32 12 7 14 22 18 11 12 5
899 342 679 507 480 1424 1359 559
1778 2774 1269 1900

422.6782	58.3029	93.5460	37.8450	313.8858	306.3226	158.7621	438.9453
419.1386	211.9909	248.2358	56.9618	156.4065	333.6976	443.8775	388.4653
249.8859	58.0584	176.9848	111.9939	319.8113	248.9988	175.8405	410.8566
287.3972	58.0000	309.9400	110.8258	436.5163	55.2100	1.7948	115.7135
433.9991	443.0866	351.9711	353.5808	396.2963	129.0723	351.0427	212.0081
93.0799	116.9830	193.9157	123.9930	373.8840	377.3706	41.4116	285.9760
97.8257	263.6473	152.9783	129.4507	26.1590	386.0011	433.9684	130.8762
300.0460	89.2431	211.9919	39.0608	40.9852	233.6205	377.9802	227.4712
106.2953	115.0072	133.3695	132.0945	63.9422	53.5437	379.6545	421.2564
208.8072	320.3719	346.0630	390.3560	91.4966	25.5755	127.0358	101.5479

2.9907	2.9921	7.1872	0.0365
6.1215	1.9455	7.2370	1.9926
7.0162	1.4994	3.0955	7.9707
7.9835	4.9976	7.3383	2.0196
6.0409	0.0081	5.1647	4.9805
1.7665	6.1584	5.8417	0.9733
5.8783	8.1789	0.0003	6.3871
3.0550	3.9904	1.0007	9.8126

5.2 Eksperimentalni rezultati

Sve tri heuristike su testeirane na svim instancama, izvršavanje svake heuristike je pokrenuto 30 puta za svaku instancu i rezultati su predstavljeni u obliku tabela za svaku grupu instanci, malih, srednjih i velikeih dimenzija. Tabele su kreirane tako da olakšaju poređenje rezultata između heuristika, tj. tebele sadrže informacije za sve tri heuristike. Prva tabela sadrži najbolja pronađena rešenaj za svaku heuristiku u svim istancama iste grupe, pri čemu prva kolona predstavlja kod instance i sledećih 9 kolona predstavljaju informacije vezane za najbolje pronađeno rešenje za svaku heuristiku u datoj instanci, odnosno broj fabrika i skladišta i vrednost

funkcije cilja za najbolje pronađeno rešenje. Druga tabela sadrži informacije vezane za izvršavanje heuristika, prva kolona sadrži kod instance i zatim sledećih 12 kolona sadrže informacije o najboljem vremenu izvršavanja, prosečnom vremenu izvršavanja, prosečnom procentualnom odstupanju, i standardnoj deviaciji za svaku heuristiku u datoj instanci.

5.2.1 Rezultati na instancama malih dimenzija

Za male instance vreme izvršavanja se kretalo od 0.164564 do 25.3182 sekundi za PSO sa prosekom od 4.29236, od 0.003251 do 6.56861 sekundi sa prosekom od 0.994674 sekundi za SA i od 0.025796 do 1.30182 sekundi za hibridni algoritam i prosekom od 0.192367 sekundi, pri čemu najbolja dobijena rešenja svih algoritama na svim instancama se poklapaju sa optimalnim rešenjima dobijenim pomoću CPLEX programa. Prosečno procentualno odstupanje se kretalo između 0.0 do 0.791525 za PSO, od 0.0657276 do 1.69587 za SA i od 0.0 do 0.0113216 za hibridni algoritam, dok se standardna devijacija kretala od 0.0 do 0.966332 za PSO, od 0.211621 do 1.54262 i od 0.000 do 0.061 za hibridni algoritam. U tabelama 1, 2 i 3 nalaze se rezultati.

Table 1: CPLEX results on small dimension instances

	Optimal Solution			CPLEX			
Instance	Plants	Depots	MIP	Nodes	Iterations	Time	Solution (Status)
111	1	5	33868.8876	48	1605	0.2030	OPTIMAL (101)
211	1	5	31695.9792	53	1767	0.2500	OPTIMAL (101)
311	1	4	32145.7672	19	903	0.2030	OPTIMAL (101)
411	1	5	40245.7301	67	1563	0.3100	OPTIMAL (101)
511	1	5	20348.9096	57	1563	0.2500	OPTIMAL (101)
611	1	7	36578.2906	88	1537	0.2350	OPTIMAL (101)
711	1	7	26209.1615	154	2119	0.2650	OPTIMAL (101)
121	1	11	43114.0233	87	3500	1.8100	OPTIMAL (101)
221	1	10	45200.8489	164	5552	2.0750	OPTIMAL (101)
321	1	9	47205.5729	1369	39251	13.5090	OPTIMAL (101)
421	1	10	43683.8233	149	4049	1.6380	OPTIMAL (101)
521	1	9	35956.0059	53	3573	1.6220	OPTIMAL (101)
621	1	11	32974.2337	332	8424	2.5120	OPTIMAL (102)
721	1	15	32798.7098	1963	44633	11.1220	OPTIMAL (101)
131	2	11	70868.6819	6788	231524	150.2980	OPTIMAL (102)
231	1	15	78850.0219	266	8759	6.2090	OPTIMAL (101)
331	1	13	81341.4474	1201	51140	46.5550	OPTIMAL (101)
431	2	16	72971.0852	1188	46864	39.1090	OPTIMAL (101)
531	1	14	69176.8127	1547	81960	61.4170	OPTIMAL (101)
631	2	18	60583.8662	1921	95731	82.2440	OPTIMAL (102)
731	2	19	52348.6335	1049	32882	33.4870	OPTIMAL (101)

Table 2: Heuristic results on small size instances - best solution found

	Particle Swarm			Simulated Annealing			Hybrid		
Inst	Plants	Depots	Function	Plants	Depots	Function	Plants	Depots	Function
111	1	5	33868.9	1	5	33868.9	1	5	33868.9
211	1	5	31696.0	1	5	31696.0	1	5	31696.0
311	1	4	32145.8	1	4	32145.8	1	4	32145.8
411	1	5	40351.0	1	5	40351.0	1	5	40351.0
511	1	5	20348.9	1	5	20348.9	1	5	20348.9
611	1	7	36578.3	1	7	36578.3	1	7	36578.3
711	1	7	26209.2	1	7	26209.2	1	7	26209.2
121	1	11	43114.0	1	11	43114.0	1	11	43114.0
221	1	10	45200.8	1	10	45200.8	1	10	45200.8
321	1	9	47205.6	1	9	47205.6	1	9	47205.6
421	1	10	43683.8	1	10	43683.8	1	10	43683.8
521	1	9	35956.0	1	9	35956.0	1	9	35956.0
621	1	11	32974.2	1	11	32974.2	1	11	32974.2
721	1	15	32798.7	1	15	32798.7	1	15	32798.7
131	2	11	70868.7	2	11	70868.7	2	11	70868.7
231	1	15	78850.0	1	15	78850.0	1	15	78850.0
331	2	14	73176.0	2	14	73176.0	2	14	73176.0
431	2	16	72971.1	2	16	72971.1	2	16	72971.1
531	1	14	69176.8	1	14	69176.8	1	14	69176.8
631	2	18	60583.9	2	18	60583.9	2	18	60583.9
731	2	19	52348.6	2	19	52348.6	2	19	52348.6

Table 3: Heuristic results on small size instances - performance rsults

	Particle Swarm				Simulated Annealing				Hybrid			
Inst	T_{min}	T_{avg}	$agap$	σ	T_{min}	T_{avg}	$agap$	σ	T_{min}	T_{avg}	$agap$	σ
111	0.425	1.292	0.054	0.290	0.003	0.224	0.099	0.295	0.027	0.056	0.000	0.000
211	0.324	1.505	0.113	0.242	0.019	0.200	0.080	0.212	0.027	0.131	0.000	0.000
311	0.502	1.179	0.000	0.000	0.007	0.169	0.105	0.294	0.027	0.031	0.000	0.000
411	0.384	1.611	0.215	0.361	0.009	0.149	0.163	0.302	0.026	0.092	0.000	0.000
511	0.538	1.435	0.000	0.000	0.008	0.147	0.152	0.508	0.027	0.038	0.000	0.000
611	0.356	1.171	0.267	0.372	0.004	0.117	0.174	0.433	0.026	0.070	0.000	0.000
711	0.165	1.293	0.383	0.511	0.008	0.175	0.066	0.225	0.026	0.070	0.000	0.000
121	2.166	7.106	0.694	0.703	0.064	1.024	1.215	0.784	0.042	0.316	0.011	0.059
221	2.963	5.945	0.324	0.369	0.109	1.102	1.149	0.854	0.043	0.352	0.000	0.000
321	2.992	7.962	0.335	0.426	0.013	0.925	0.678	0.468	0.042	0.318	0.007	0.020
421	2.430	5.321	0.263	0.416	0.026	0.764	0.980	0.664	0.045	0.266	0.000	0.000
521	2.436	6.870	0.787	0.932	0.013	0.906	1.344	0.861	0.079	0.522	0.008	0.044
621	2.242	4.732	0.488	0.966	0.034	1.075	1.696	1.543	0.041	0.107	0.000	0.000
721	1.968	7.264	0.792	0.677	0.043	1.061	0.663	0.402	0.042	0.396	0.011	0.061
131	2.408	6.026	0.165	0.218	0.031	1.580	0.380	0.266	0.049	0.297	0.003	0.013
231	2.518	4.773	0.245	0.384	0.023	2.047	0.621	0.542	0.047	0.193	0.000	0.000
331	2.259	5.502	0.276	0.402	0.022	1.674	0.386	0.461	0.047	0.068	0.000	0.000
431	1.661	4.937	0.471	0.497	0.027	1.783	0.599	0.480	0.047	0.236	0.000	0.000
531	2.171	5.883	0.221	0.292	0.022	1.907	0.795	0.576	0.049	0.253	0.000	0.000
631	1.728	4.229	0.385	0.442	0.048	2.116	0.605	0.518	0.047	0.156	0.000	0.000
731	1.341	4.100	0.372	0.449	0.201	1.742	0.549	0.515	0.046	0.070	0.000	0.000

5.2.2 Rezultati na instancama srednjih dimenzija

Za instance srednjih dimenzija vreme izvršavanja za PSO se kretalo od 0.212051 do 61.8813 sekundi sa prosečnim vremenom izvršavanja od 14.0256 sekundi, za SA se kretalo između 0.098693 i 44.8919 sekundi sa prosečnim vremenom izvršavnja od 7.97063 sekundi, dok je za hibridni algoritam vreme izvršavanja između 0.071168 i 6.04593 sekundi sa prosečnim vremenom izvršavanja od 1.32939 sekundi. Prilikom izvršavanja PSO dobijeno je najmanje 12 optimalnih od 23, dok je izvršavanjem SA dobijeno 6 optimalnih od 23, a izvršavanjem hibridnog algoritma dobijena su najmanje 21 optimalno od 23 rešenja za instance srednjih dimenzija. Prosečno procentualno odstupanje se kretalo od 0.0 do 2.457 za PSO, od 0.13471 do 4.67323 za SA i od 0.0 do 0.317467 za hibridni algoritam a stndardna devijacija se nalazila između 0.0 i 1.51836 za PSO, 0.199206 i 2.05523 za SA, a za hibridni algoritam između 0.0 i 0.0392806. U tabeli 4 su detaljnije predstavljeni rezultati izvršavanja CPLEX programa, dok su tabelama 5 i 6 predstavljeni rezultati izvršavanja heuristika.

Table 4: CPLEX results on medium size instances

	Optimal Solution			CPLEX			
Instance	Plants	Depots	MIP	Nodes	Iterations	Time	Solution (Status)
141	2	19	66417.2535	809	56463	171.1320	OPTIMAL (102)
241	2	17	75345.8964	775	97876	304.4650	OPTIMAL (102)
341	1	16	62551.4082	564	44074	163.1130	OPTIMAL (101)
441	1	16	66654.2208	578	54491	190.1950	OPTIMAL (101)
541	2	17	73777.6805	1196	196552	521.8860	OPTIMAL (102)
641	2	22	40139.7081	532	30358	102.1330	OPTIMAL (101)
741	2	22	40680.3015	685	54940	171.2010	OPTIMAL (101)
841	2	23	51340.0375	1584	161292	347.3030	OPTIMAL (102)
151	1	18	70646.8217	605	100077	688.8000	OPTIMAL (102)
251	1	19	67942.9798	1570	451875	2536.8140	OPTIMAL (102)
351	1	18	74565.8724	7324	3097512	14400.5100	FEASIBLE (107)
451	2	14	59991.3303	7653	2940108	13640.1260	OPTIMAL (102)
551	1	15	56237.6217	848	234409	1612.8860	OPTIMAL (101)
651	2	21	47280.6651	9762	3181173	14400.2150	FEASIBLE (107)
751	2	23	42825.4930	10019	2282587	9119.2770	OPTIMAL (102)
851	2	19	43277.4175	4628	1612269	8305.8910	OPTIMAL (102)
161	3	23	305122.9296	334	16177	48.7500	OPTIMAL (102)
261	3	23	315121.2810	1066	68295	212.0350	OPTIMAL (101)
361	3	21	283054.5407	275	13523	27.6080	OPTIMAL (102)
461	4	22	343056.6359	471	21806	39.8850	OPTIMAL (101)
561	1	21	281458.3180	144	10339	34.0700	OPTIMAL (101)
661	3	25	196285.9103	168	8225	20.4920	OPTIMAL (101)
761	4	25	218507.3498	258	11185	24.4450	OPTIMAL (102)

Table 5: Heuristic results on medium size instances - best solution found

	Particle Swarm			Simulated Annealing			Hybrid		
Inst	Plants	Depots	Function	Plants	Depots	Function	Plants	Depots	Function
141	2	19	66552.6	2	20	66874.6	2	19	66417.3
241	2	17	75345.9	3	18	76215.9	2	17	75345.9
341	1	16	62551.4	1	15	62799.7	1	16	62551.4
441	1	17	66911.4	1	16	66970.7	1	16	66654.2
541	2	17	73777.7	2	17	74447.5	2	17	73777.7
641	2	21	40201.8	1	21	40494.9	2	22	40139.7
741	2	22	40680.3	2	21	40940.7	2	22	40680.3
841	2	24	51529.7	2	23	51465.6	2	23	51340.0
151	1	19	70747.5	1	19	70842.2	1	18	70646.8
251	1	19	68192.9	1	19	69911.5	1	19	67943.0
351	1	18	74565.9	1	19	75271.7	1	18	74565.9
451	2	14	60008.4	3	13	60743.7	2	14	59991.3
551	1	15	56237.6	1	16	58029.5	1	15	56237.6
651	2	21	47280.7	2	22	48631.7	2	21	47280.7
751	2	22	42999.4	2	23	43948.9	2	23	42825.5
851	2	20	43403.8	2	19	44851.2	2	19	43277.4
161	3	23	305123	3	23	305123	3	23	305123
261	3	23	315121	3	23	315121	3	23	315121
361	3	21	283055	3	22	283209	3	21	283055
461	4	22	343057	4	22	343057	4	22	343057
561	1	21	281458	1	21	281458	1	21	281458
661	3	25	196286	3	25	196286	3	25	196286
761	4	25	218507	4	25	218507	4	25	218507

Table 6: Heuristic results on medium size instances - performance results

	Particle Swarm				Simulated Annealing				Hybrid			
Inst	T_{min}	T_{avg}	$agap$	σ	T_{min}	T_{avg}	$agap$	σ	T_{min}	T_{avg}	$agap$	σ
141	11.453	21.630	1.060	0.732	0.410	4.732	1.666	0.774	0.074	1.729	0.259	0.243
241	7.933	24.156	1.299	0.873	0.431	4.592	0.934	0.634	0.136	1.827	0.238	0.294
341	7.349	15.376	1.326	0.772	0.652	4.578	2.495	1.162	0.454	1.436	0.093	0.184
441	6.589	18.210	1.352	0.970	0.502	5.221	1.601	0.813	0.077	1.259	0.258	0.246
541	9.828	19.628	1.089	0.690	0.371	4.353	1.078	0.598	0.077	1.960	0.207	0.221
641	9.041	14.277	1.632	1.416	0.336	4.771	1.238	0.951	0.071	0.625	0.015	0.046
741	8.029	14.543	2.363	1.518	0.222	5.827	2.613	0.913	0.072	0.631	0.211	0.405
841	8.713	16.137	1.333	0.798	0.099	5.432	1.570	0.689	0.074	1.242	0.017	0.039
151	10.486	18.082	1.282	0.831	0.143	12.256	4.199	1.726	0.164	1.694	0.147	0.221
251	8.571	21.508	1.371	0.773	0.124	13.339	3.873	1.597	0.505	2.638	0.317	0.375
351	10.147	21.882	1.461	0.844	0.759	9.823	2.611	0.989	0.612	2.993	0.165	0.196
451	10.162	21.597	1.369	0.836	0.356	10.060	2.983	1.241	0.105	2.396	0.263	0.246
551	9.802	18.909	1.170	1.070	0.122	10.040	4.673	2.055	0.171	2.028	0.064	0.164
651	13.964	21.772	1.552	0.964	0.651	13.268	2.513	1.447	0.223	1.725	0.216	0.309
751	10.500	17.147	2.090	1.129	0.336	14.237	2.311	0.693	0.088	1.907	0.157	0.164
851	10.596	20.901	2.457	1.400	0.175	10.243	3.228	1.981	0.734	1.978	0.034	0.101
161	0.883	2.702	0.056	0.070	0.190	8.014	0.293	0.322	0.104	0.348	0.000	0.000
261	0.910	3.012	0.072	0.094	0.369	8.955	0.135	0.199	0.095	0.313	0.000	0.000
361	1.538	3.626	0.131	0.185	0.539	7.731	0.627	0.419	0.098	0.532	0.000	0.000
461	1.199	2.544	0.123	0.113	0.809	7.669	0.166	0.165	0.097	0.282	0.000	0.000
561	1.440	2.759	0.066	0.091	0.937	6.690	0.615	0.458	0.095	0.276	0.000	0.000
661	0.785	1.244	0.000	0.000	0.215	6.170	1.067	0.916	0.098	0.386	0.000	0.000
761	0.212	0.946	0.000	0.000	0.287	5.324	0.829	0.664	0.099	0.371	0.000	0.000

5.2.3 Rezultati na instancama velikih dimenzija

Za instance velikih dimenzija izvršavanje PSO je trajalo od 3.98384 do 54.637 sa prosekom 13.2277, za SA je trajalo od 0.648464 do 179.051 sa prosekom od 38.4367 sekundi, dok je za hibridini algoritam izvršavanje trajalo od 0.13967 do 7.45233 sa prosečnim vremenom izvršavanja od 1.65099 sekundi. Broj pronađenih optimalnih rešenja za PSO je dva, a prilikom izvršavanja SA pronađeno je samo jedno optimalno rešenje, dok se hibridni algoritam pokazao kao najbolji sa najmanje deset pronađenih optimalnih rešenja. Prosečno procentualno odstupanje se kretalo između 0.468161 do 1.28835 za PSO, od 0.420561 do 1.91988 za SA i od 0.0 do 0.0793323 za hibridni algoritam, dok se standardna devijacija kretala od 0.30365 do 0.965061 za PSO, od 0.298161 do 1.01097 i od 0.000 do 0.116593 za hibridni algoritam. Rezultati u instancama velikih dimenzija za CPLEX i heuristike detaljnije su prikazani u tabelama 7, 8 i 9.

Table 7: CPLEX results on large size instances

	Optimal Solution			CPLEX			
Instance	Plants	Depots	MIP	Nodes	Iterations	Time	Solution (Status)
171	3	36	257690.6474	533	40697	536.2600	OPTIMAL (102)
172	2	35	204898.2545	601	54057	746.4720	OPTIMAL (101)
271	1	35	238805.3677	534	37793	360.6870	OPTIMAL (102)
273	1	36	233124.2766	542	34914	379.3610	OPTIMAL (101)
371	3	32	270892.8639	768	79926	862.1540	OPTIMAL (102)
374	2	32	270700.1578	549	52292	520.2600	OPTIMAL (101)
471	3	34	243379.2342	520	33196	318.8610	OPTIMAL (101)
475	3	41	168194.0738	550	35817	342.1560	OPTIMAL (101)
571	3	34	225296.7037	560	44975	541.6290	OPTIMAL (101)
576	4	34	227976.8915	604	40236	468.2190	OPTIMAL (101)
181	10	48	124986.2415	10	9917	383.9940	FEASIBLE (109)
182	6	32	127637.2096	9	9072	394.3380	FEASIBLE (109)
281	4	47	122888.1030	12	11915	350.2670	FEASIBLE (109)
283	5	30	126028.4702	11	10328	520.7910	FEASIBLE (109)
381	5	48	108738.7921	11	10215	322.4520	FEASIBLE (109)
384	10	50	123120.0155	7	7903	292.7970	FEASIBLE (109)
481	5	46	110197.3864	10	10644	377.0830	FEASIBLE (109)
485	7	39	119124.8280	12	10352	376.1470	FEASIBLE (109)
581	4	31	129876.5793	10	8765	420.2649	FEASIBLE (109)
586	6	49	119750.9921	11	12514	496.2360	FEASIBLE (109)

Table 8: Heuristic results on large size instances - best solution found

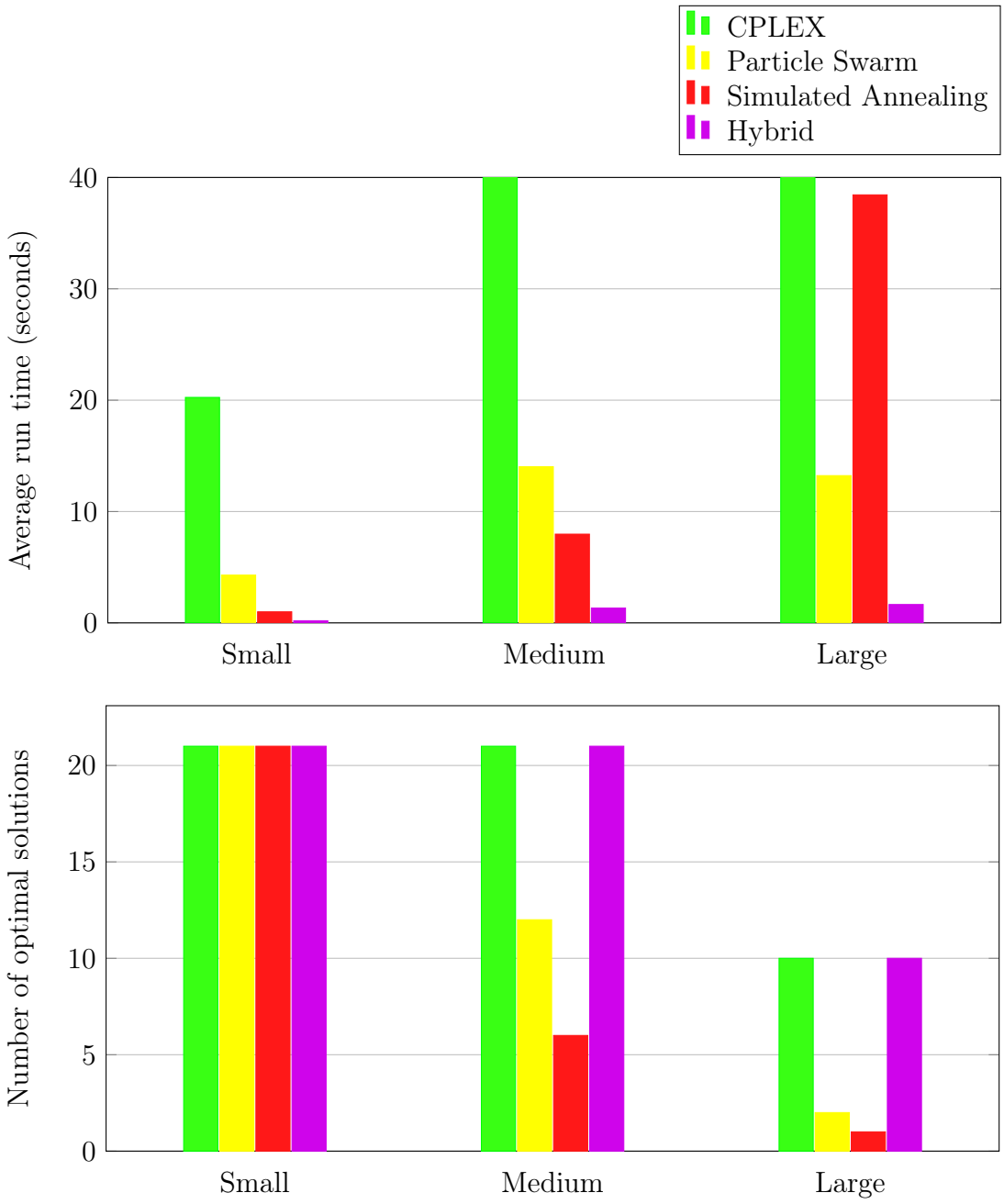
	Particle Swarm			Simulated Annealing			Hybrid		
Inst	Plants	Depots	Function	Plants	Depots	Function	Plants	Depots	Function
171	3	37	257719	3	36	258513	3	36	257691
172	2	35	204898	2	35	205299	2	35	204898
271	1	36	239013	1	35	238827	1	35	238805
273	1	38	233602	1	36	233124	1	36	233124
371	3	32	270935	3	33	271407	3	32	270893
374	2	33	271019	2	32	270891	2	32	270700
471	3	34	243379	3	35	243972	3	34	243379
475	3	42	175652	3	42	175640	3	40	175362
571	3	35	225461	2	34	225630	3	34	225296
576	4	35	228654	4	34	227977	4	34	227976
181	4	39	110776	4	39	111519	4	39	110553
182	4	41	120140	4	38	120674	4	38	119570
281	4	38	116876	4	38	117233	4	38	116876
283	4	39	112546	4	37	112827	4	38	112491
381	4	39	104306	4	39	104653	4	38	104300
384	4	39	105712	4	37	106352	4	38	105500
481	3	41	107287	4	39	107491	3	38	106865
485	4	40	110490	4	39	111171	4	39	110452
581	4	32	113656	3	39	114938	3	42	113656
586	3	40	110353	3	38	111948	3	38	110117

Table 9: Heuristic results on large size instances - performance results

	Particle Swarm				Simulated Annealing				Hybrid			
Inst	T_{min}	T_{avg}	$agap$	σ	T_{min}	T_{avg}	$agap$	σ	T_{min}	T_{avg}	$agap$	σ
171	4.708	10.526	0.704	0.426	0.816	27.818	0.518	0.479	0.323	2.402	0.045	0.062
172	6.470	14.063	0.628	0.319	2.887	41.454	0.447	0.345	0.943	2.145	0.036	0.051
271	4.797	10.337	0.641	0.395	7.618	28.910	0.734	0.383	0.140	1.363	0.006	0.028
273	5.424	8.543	0.468	0.362	1.716	26.350	0.802	0.357	0.227	2.004	0.031	0.054
371	8.201	16.494	0.571	0.319	1.359	24.864	0.421	0.223	0.149	2.231	0.026	0.063
374	7.862	14.225	0.493	0.304	2.188	32.364	0.763	0.326	0.357	2.175	0.027	0.047
471	6.217	13.739	0.610	0.391	0.856	26.848	0.916	0.529	0.141	1.726	0.065	0.069
475	3.984	9.084	0.615	0.454	1.092	36.870	1.137	0.586	0.288	0.879	0.000	0.000
571	6.387	14.420	0.927	0.455	1.188	21.146	0.680	0.298	0.160	2.661	0.079	0.113
576	6.069	13.318	0.936	0.513	0.648	19.814	1.006	0.438	0.522	2.205	0.074	0.117
181	6.107	16.107	1.031	0.650	5.058	60.576	1.189	0.597	0.151	1.931	0.011	0.030
182	5.322	13.316	1.181	0.720	0.858	33.428	1.120	0.602	0.155	1.016	0.000	0.000
281	7.301	15.465	1.052	0.699	0.729	49.284	1.526	0.671	0.147	1.157	0.001	0.002
283	7.157	14.308	1.073	0.694	0.787	52.809	1.917	0.684	0.230	1.278	0.004	0.010
381	8.225	12.797	1.272	0.505	4.679	57.745	1.845	0.618	0.230	1.045	0.000	0.000
384	7.774	15.793	0.663	0.503	2.820	50.550	1.324	0.683	0.145	1.851	0.010	0.030
481	5.784	12.031	1.279	0.622	1.314	35.757	1.678	0.716	0.154	1.322	0.005	0.029
485	7.295	13.326	1.045	0.555	4.280	51.642	1.341	0.679	0.147	1.341	0.015	0.037
581	7.160	14.080	0.984	0.605	0.744	51.476	0.920	0.513	0.243	1.226	0.003	0.014
586	8.096	12.308	1.288	0.965	1.879	39.031	1.920	1.011	0.152	1.062	0.000	0.000

5.3 Analiza rezultata

U ovom delu su upoređivane preformanse sve tri heuristike kao i performanse CPLEX programa pomoću bar-dijagrama za instance malih, srednjih i velikih dimenzija. Prvi dijagram opisuje prosečno vreme izvršavanja, dok drugi dijagram prikazuje broj pronađenih optimalnih rešenja za svaku grupu instanci. Prosečno vreme izvršavanja CPLEX programa je 2816.66226 za srednje i 450.52145 sekundi za velike instance, pa je zbog toga gornja granica y-ose postavljena na 40, jer bi u suprotnom, postavljanje granice na jednu od tih vrednosti, značajno narušilo preglednost dijagrama.



U drugom dijagramu broj pronađenih optimalnih rešenja hibridne heuristike i CPLEX programa su postavljeni na istu vrednost, čak i ako se hibridna heuristika bolje pokazala u instancama najveće dimenzije, u kojima optimalnost nije dokazana pomoću CPLEX programa. Međutim, velika je verovatnoća da su izvršavanjem hibridne heuristike pronađena sva optimalna rešenja što je zaključeno na osnovu činjenice da je za svaku instancu, od 30 izvršavanja bar 20 puta izračunata ista njamanja vrednost funkcije cilja. To se može zaključiti i iz gornje tabele, gde je prosčno procentualno odstupanje blizu nule.

Izvorni kod finalne verzije implementacije heuristika zajedno sa instancama koje su korišćene u ovom radu mogu se naći na adresi: **www.github.com/MrHofmann/TSUFLP**

6 Zaključak

Predmet proučavanja u ovom radu bio je TSUFLP problem za koji su poznata mnoga rešenja, međutim zbog jednostavnoti implementacije izabran je osnovni matematički model za rešavanje ovog problema. Cilj ovog rada bio je da se implementiraju tri heuristička algoritma koja će tražiti optimalna rešenja, odnosno rešenja što bliže optimalnim rešenjima jer ovi algoritmi nisu egzaktni i ne mogu dokazati optimalnost pronađenih rešenja. Međutim rešenja ovog problema u većini instanca je pronađeno u okviru nekog drugog rada pomoću CPLEX programa, pa su sa tim rešenjima upoređeni rezultati heuristika kako bi se ocenio njihov uspeh. Tri heuristike koje su korišćene su PSO, SA i njihova hibridizacija. Prilikom analize rezultata primećeno je da SA algoritam ima značajnu prednost u odnosu na ostala dva algoritma u instancama najmanjih dimenzija, to je i bilo očekivano jer male instance nemaju dovoljnu veliku širinu u smislu dimenzija pa je SA mogao da savlada tu veličinu bez značajnih poteškoća, odnosno bez pojavljivanja preuranjenih konvergencija. Međutim, kako su dimenzije instance rasle tako je moć algoritma SA naglo opala, jer algoritam nije mogao da iznese širine tih dimenzija, za čije rešavanje je neophodna diversifikacija pretrage, što je u principu glavni nedostatak SA algoritma. U instancama srednjih dimenzija dobro se pokazao PSO algoritam, jer je PSO prirodno rešenje za probleme gde je neophodna diversifikacija. PSO je davao dobre rezultate sve do najvećih instanci gde je dostigao svoj limit zbog neostatka dubine u pretraživanju, što je njegov glavni nedostatak. U instancama najvećih dimenzija obe heuristike, PSO i SA, bile su nemoćne, odnosno svaka od njih posebno je davala loše rezultate, međutim to nije bio slučaj pri izvršavanju njihove hibridizacije koja je dominirala u instancama svih dimenzija. Velika je verovatnoća da je hibridna heuristika pronašla optimalna rešenja i u slučajevima gde CPLEX program nije mogao da dokaže optimalnost. Kada se grade hibridni heuristički algoritmi, uvek se za izgradnju hibridizacije koriste algoritmi koji jadan drugome nadopunjuju nedostatke. Upravo je to u ovom radu i postignuto, upotrebljene su dve tehnike, jedna koja podstiče intezifikaciju a druga diversifikaciju, pri čemu su one sastavljene tako da funkcionišu kao jedno i kako bi sinergično savladale prepreke koje nisu mogle pojedinačnim izvršavanjem.

Reference

[1] Mercedes Landete, Alfredo Marín. *New facets for the two-stage uncapacitated facility location polytope*. Springer Science, Business Media, LLC 2008.

[2] Stefan Mišković, Zorica Stanimirović. *A Memetic Algorithm for Solving Two Variants of the Two-Stage Uncapacitated Facility Location Problem*. Faculty of Mathematics, University of Belgrade, 2012.

[3] El-Ghazali Talbi. *Metaheuristics, From Design to Implementation*. University of Lille – CNRS – INRIA, 2009.

[4] Dragan D. Đurđević. *Poređenje egzaktnih i heurističkih metoda za rešavanja nekih optimizacionih problema*. Matematički Fakultet, Univerzitet u Beogradu, 2014.

[5] Gerard Cornuejols, George L. Nemhauser, Laurence A. Wolsey *The Uncapacited Facility Location Problem*. Carnegie-Mellon University, August 1983.

[6] Jakob Krarup, Peter Mark Pruzan *The Simple Plant Location Problem: Survey and Synthesis*. Universirty of Copenhagen, February 1982.

[7] Vedat Verder *Uncapacitated and Capacitated Facility Location Problems*.

[8] Dr. James Kennedy, Dr. Russell Eberhart *Particle Swarm Optimization*. Purdue School of Engineering and Technology, Indianapolis 1995.

[9] Armen G. Khachaturyan, Svetlana V. Semenovskaya, Boris K. Vainshtein *Statistical-Thermodynamic Approach to Determination of Structure Amplitude Phases*. Sov.Phys. Crystallography. 1979.