SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Matea Pejčinović, Fran Stanić

Primjena lokalnog pretraživanja u rješavanju problema izrade rasporeda zaposlenika

Zagreb, 2016

Ovaj rad izrađen je u Zavodu za telekomunikacije pod vodstvom doc.dr.sc. Lee Skorin-Kapov i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2015./2016.“.

Sadržaj

1. Uvod

Problem raspoređivanja osoblja prisutan je u mnogim ljudskim djelatnostima, počevši od određivanja smjena u bolnicama [2][3][14] ili vatrogasnim službama, rasporeda zaposlenika u tvornicama, zračnim lukama [5], studenata i profesora na učilištima i slično. Iako je danas poznat značajan broj različitih algoritama koji su primjenjivi u mnoge svrhe**, zbog kompleksnosti problema raspoređivanja često je gotovo nemoguće pronaći optimalno rješenje u vremenski razumnom razdoblju**. Ipak, moguće je doći do rješenja primjenom heurističkih algoritama koji ne garantiraju optimalnost, no generirana su u relativno razumnom vremenskom roku te se mogu poboljšavati jednom kad se dođe do početnog rješenja. Fokus ovog rada je upravo primjena nekih uobičajenih i osmišljavanje novih heuristika koje omogućuju izradu učinkovitog rasporeda postojećih instanci problema, ali i nekih budućih.

U okviru ovoga rada rješava se problem raspoređivanja osoblja (engl. „*Employee Rostering Problem*“) uz brojna ograničenja. Radi se o problemu koji je dosta izazovan jer se nastoji postići takva raspodjela smjena koja će zadovoljiti različite preferencije članova osoblja uz poštivanje velikog broja ograničenja postavljenih na svakog od njih. **Iako dosta zahtijevan zbog velikog broja kombinacija koji otežavaju pronalazak rješenja s minimalnim troškom, korištenjem algoritma temeljenog na heuristici lokalnog pretraživanja problem raspoređivanja osoblja u ovom je radu obrađen na inovativan način. Njime su postignuta poboljšanja vrijednosti do sad najboljih poznatih vrijednosti funkcije cilje instanci problema preuzetih sa stranica Sveučilišta u Nottighamu.**

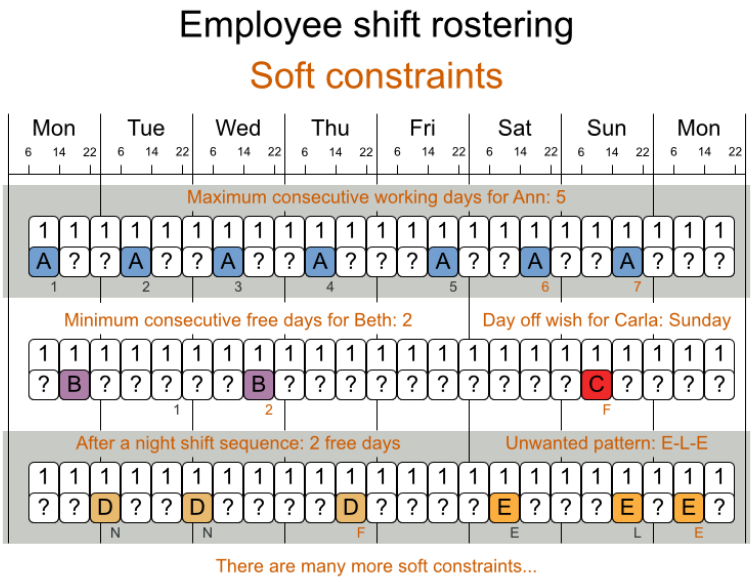
U drugom poglavlju opisana je motivacija i radovi koji se bave sličnom tematikom dok se ciljevi rada i postavljene hipoteze obrađuju u sljedećem poglavlju. Treće, pak, poglavlje sadrži opis komponenata heuristike predložene u okviru ovog rada čija je procjena točnosti i uspješnosti analizirana u četvrtom poglavlju. Posljednji dio rada usmjeren je na koncizno iznošenje zaključaka izvedenih iz rada s naglaskom na evaluaciju postignutih rezultata kompariranih s do sad najboljim poznatim vrijednostima.

2. Opis problema

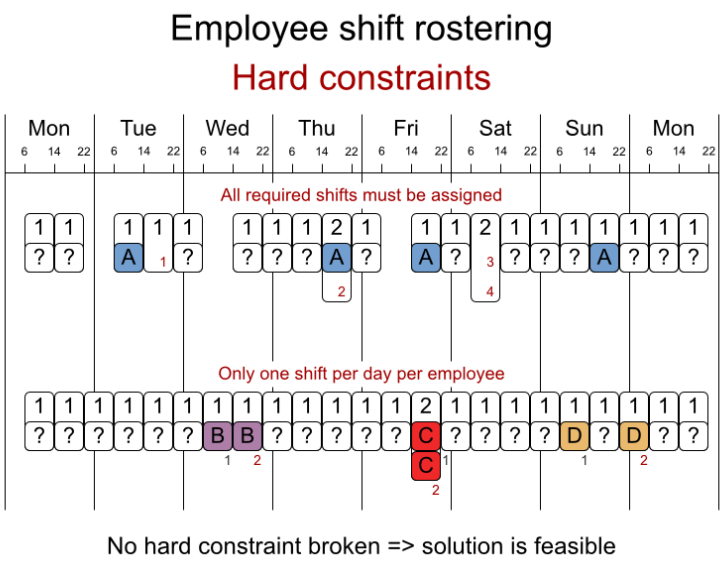
U ovom poglavlju dan je kratak uvod u samu problematiku kojom se bavi radi. Zbog važnosti ovog problema i njegove primjenjivosti u svakodnevnim ljudskim djelatnostima, prikazani su i pristupi koji su do sad korišteni u svrhu njegovog rješavanja.

2.1 Problem raspoređivanja osoblja

Problem raspoređivanja osoblja nastoji izgraditi raspored u pojedinoj organizaciji koji će zadovoljiti njezine zahtjeve za distribucijom resursa. Najprije se određuje skup osoblja i vremensko razdoblje na koje se treba primijeniti takav raspored uz broj i vrstu smjena. Ovaj je problem poznat kao izazovan zadatak u području optimizacijskih metoda te općenito operacijskog istraživanja. Takva klasifikacija proizlazi iz mnoštva faktora koji ga oblikuju, kao što su brojnost smjena koje se trebaju dodjeliti, način rada (puno ili fleksibilno radno vrijeme pojedinih zaposlenika), ograničenja u pogledu slobodnih dana i broja radnih vikenda itd. Ovakva jaka ograničenja (eng. *hard constraints*) definiraju zahtjeve koje svako rješenje mora zadovoljiti kako bi se smatralo valjanim (eng. *feasible*) dok procjena njegove uspješnosti proizlazi iz slabih ograničenja (eng. *soft constraints*). Ona se ne moraju zadovoljiti kako bi rješenje bilo moguće, no što ih se više ispuni, to je rješenje bliže optimalnom u pogledu preferencija svakog zaposlenika i organizacije te podjednake raspodjele smjena. Slike 1 i 2 ilustriraju kako ova ograničenja izgledaju na jednom stvarnom primjeru u definiranom vremenskom razdoblju za koje se izrađuje raspored.



Slika 1. Primjer slabih ograničenja u odabranom vremenskom razdoblju (izvor: https://dzone.com/articles/automated-nurse-rostering)



Slika 2. Primjer jakih ograničenja u vremenskom razdoblju za koje se određuje raspored (izvor: https://dzone.com/articles/automated-nurse-rostering)

Prilikom zadavanja problema svakom se članu osoblja pripišu vlastite želje, npr. u pogledu smjena koje želi raditi i slobodnih dana, ali i ograničenja koja su često zakonske prirode ili proizlaze iz organizacijske politike. Primjeri zakonskih

ograničenja bili bi maksimalan broj smjena koje osoba smije odraditi bez slobodnih dana ili maksimalan broj radnih vikenda. Ograničenja koja određuje organizacija više pripadaju domeni slabih ograničenja jer se radi o isplativosti poslovanja. Određuje se koliko bi trebalo biti zaposlenika u pojedinoj smjeni pri čemu do troška dovodi svaki broj koji odstupa od optimalnog. Također, ako se dogodi da neki zaposlenik radi na gornjoj granici dozvoljenog broja radnih sati, a riječ je o članu osoblja čija je satnica visoka, organizacija će imati povećane troškove. Iz navedenog je jasno kolika je važnost ovog problema u industriji. **Pronalazak rješenja sa što manjim troškom dovest će do poboljšanja poslovanja ustanova i tvrtki, a također pridonijeti povećanju morala zaposlenika te boljeg radnog učinka.**

Prilikom rješavanja ovakvih problema koriste se mnogobrojni algoritmi koji se, zbog prirode problema, temelje na heurističkim pristupima. Neki od najpoznatijih su implementacije stohastičkih algoritama, genetskih algoritama, simuliranog kaljenja, tabu pretrage...

2.2 Slični radovi

Radovi koji obrađuju ovu tematiku obuhvaćaju mnoge stvarne probleme s kojima se svakodnevno susrećemo. Uglavnom se radi o problemu raspoređivanja medicinskih sestara u bolnicama kao što je istaknuto u [2], [4], [6] i [14] ili općenito medicinskog osoblja [3]. No, algoritmi koji se bave ovom tematikom obuhvaćaju i druge grane ljudske djelatnosti [5]. Pri tome je bitno istaknuti kako neovisno o temeljnoj motivaciji, znanstvenici su primjenjivali različite algoritme. Tako se za problem raspoređivanja medicinskih sestara uglavnom u radovima spominje genetski algoritam [2][8], no zanimljiv je i pristup u [6] u kojem se navodi korištenje pohlepnih algoritama,

mrežnog planiranja te tabu pretrage. U [2]su isprobane određene verzije tabu pretrage, inače dosta uobičajene u problemu raspoređivanja, te je izabrana jedna od njih kao ona koja daje najbolje rezultate.

Nerijetko je i matematički pristup onaj za koji se autori heuristike odlučuju pri modeliranju problema [10][15]. Unatoč njegovoj kompleksnosti koja proizlazi iz velikog broja ograničenja koja se moraju definirati, pokazalo se da su distribucije koje on generira prihvatljivog troška i ne odstupaju previše od donje granice (eng. *lower bound*). Na samom početku rada nužno je eliminirati, odnosno umanjiti poteškoće na koje se može naići pri računanju [15], no nakon toga algoritam radi dobro u okvirima optimizacijskog postupka.

Algoritam prikazan u [11] nastoji ostvariti eksplicitno učinje u domeni raspoređivanja sestara za što se koriste heuristička pravila. Lokalno pretraživanje koje se spominje u radu se, u biti, odnosi na kreiranje parova medicinskih sestara i pravila tako da se smanjuje trošak. Rezultati do kojih se došlo u ovom radu nadmašili su mnoge već postojeće pristupe te su bili polazišna točka za heuristiku smišljenu za potrebe ovog rada.

3. Ciljevi rada i hipoteze

Dinamičnost područja koje se bavi izradom rasporeda dovela je do toga da se neprestano pojavljuju novi i intuitivni pristupi. Jedan od njih je predmet ovog rada. Njegova inovativnost se ističe u primjeni kombinacije operatora mutiranja u jednom od najpoznatijih aproksimativnih pristupa – lokalnom pretraživanju. Prvo potpoglavlje se bavi ciljevima rada dok će drugo više istaknuti mnogobrojne hipoteze kojima smo se vodili u izradi heuristike.

3.1 Ciljevi rada

Cilj izrade rasporeda osoblja je generirati efikasnu raspodjelu sredstava na zadani skup zadataka uz zadovoljenje slabih i jakih ograničenja. U ovom se radu sredstva odnose na smjene koje je nužno distribuirati pojedinim zaposlenicima pri čemu je potrebno zadovoljiti ograničenja postavljena na zaposlenika, poput rotacije smjena, njihovog maksimalnog broja, maksimalnog ili minimalnog broja uzastopnih dana odmora i sl. Ograničenja su poprilično brojna, no tek njihovim zadovoljenjem može se reći da je dobiveno moguće rješenje. Isto se heuristikom dalje pokušava poboljšati primjenom različitih operatora koji nastoje zadovoljiti što više slabih ograničenja koja u konačnici i određuju ukupni trošak. Ta ograničenja su bitna jer predstavljaju pojedinačne preferencije članova osoblja što je važno za svaku organizaciju u smislu održavanja morala zaposlenika te kvalitetnog obavljanja zaduženja. Budući da se radi o radnim organizacijama koje se moraju voditi na učinkovit način, ključno je osigurati i da u pojedinoj smjeni nema više ili manje zaposlenika nego što je zahtijevano.

Iz svega predstavljenog, jasno je zašto je teško naći optimalno rješenje. Svaka organizacija definira svoj skup ograničenja, ali i pravnih regulacija, preferencija osoblja te ciljeva u izradi rasporeda.

Nakon što se uspije konstruirati algoritam, potrebno je na odgovarajući način validirati rješenje te procijeniti njegovu uspješnost. Primijenjene metode s dobivenim rezultatima uz odgovarajuću analizu detaljno su prikazani i obrazloženi u nastavku rada.

3.2 Hipoteze

Za početak rada potrebno je bilo odrediti nekoliko hipoteza. Prva se odnosila na metode optimizacije. Problem je uobičajeno definiran korištenjem ograničenja i ciljeva koji se odnose na potrebu optimiranja funkcije cilja. Ono se može odnositi na, primjerice, minimizaciju troška ili maksimizaciju raspodjele zaposlenika po njihovim zahtjevima. Stoga nije neuobičajeno da se ovako formuliranom problemu pristupa matematičkim modeliranjem[4][6]; ponajprije korištenjem cjelovitog linearnog programiranja [10] uz metodu relaksacije. Iako je kao ideja dobro zamišljeno, ova metoda ne funkcionira najbolje u slučaju većeg broja varijabli, a kako se radi o uglavnom jako velikom broju ograničenja, ali i sredstava (smjene, dani i zaposlenici), ista bi zahtijevala nezanemarivo vremensko razdoblje za dolazak do početnog rješenja, a i još poprilično vremena za optimiranje i dolazak do koliko-toliko zadovoljavajućeg rješenja [1]. Iz tog razloga je odlučeno ne koristiti egzaktne metode već pokušati s aproksimativnim. Već iz njihova opisa je bilo jasno da je moguće dobiti rješenja koja će se moći poboljšavanjem dovesti jako blizu optimalne vrijednosti

promatranih instanci problema. Ideja za korištenjem ovog pristupa potekla je proučavanjem [11].

Druga hipoteza se odnosila na kodiranje rješenja kojima barata algoritam pretraživanja. Naime, ova komponenta uvelike utječe na efikasnost i učinkovitost metaheuristike pa treba biti oprezan pri definiranju određenog načina prikazivanja rješenja među mnogo alternativnih. Smatrali smo da odabrani način prezentacije rješenja mora biti cjelovit, efikasan i omogućiti da između bilo koja dva rješenja postoji put pretraživanja čime bi bilo moguće doći u bilo koji optimum neovisno o početnom rješenju [1]. Uzevši prethodno navedene kriterije u obzir, smatrali smo da je najbolje pokušati s linearnim prikazom rješenja koji će koristiti varijable odluke iz skupa diskretnih vrijednosti.

Prilikom definiranja hipoteza, bilo je potrebno odrediti što ćemo koristiti kao funkciju evaluacije rješenja. Kao komponenta heuristike koja treba voditi pretragu prema dobrim rješenjima, izuzetno je bitno definirati prikladnu funkciju evaluacije koja će omogućiti rangiranje svih rješenja u prostoru pretraživanja. Bit ovog problema je naći optimalno rješenje. Za ovaj oblik problema raspodjele zaposlenika, bitno je maksimizirati raspodjelu osoblja po zahtjevima uz istodobnu minimizaciju broja nedovoljnog ili prekobrojnog osoblja po smjenama. Stoga se hipoteza sastojala u ideji da je bitno dobiti prihvatljivo rješenje sa što manjim troškom te smo se odlučili u postupku poboljšavanja postojećeg rješenja voditi time da prihvaćamo uvijek prvo sljedeće rješenje koje nađe heuristika, a koje je moguće (engl. *feasible*) i manjeg troška.

Nakon što su oblikovane prethodno navedene hipoteze, došao je red i na raspravu oko početnog rješenja. Ideja kojom smo se vodili je da nam početno rješenje ne mora

biti jako dobro u smislu malog troška, već da trebamo pokušati dobiti rješenje koje će zadovoljiti sva jaka ograničenja, a kasnije ćemo heuristikom nastojati postići ono koje će zadovoljiti i većinu slabih ograničenja. Pri tome je zaključeno kako, primjerice matematičko modeliranje ipak jest vremenski prezahtjevno za derivaciju početnog rješenja, a da bi najbolji pristup vjerojatno bio primjena nekog konstruiranog pohlepnog algoritma koji će u razumnom vremenu naći zadovoljavajuće početno rješenje [13].

4. Primijenjene metode

U ovom se poglavlju detaljno navode sve primjenjene komponente izgrađene heuriste. Prvo potpoglavlje se bavi generiranjem početnog rješenja pri čemu su navedena sva ograničenja koje isto mora zdovoljiti da bi bilo moguće. Potom se u nastavku prikazuju odabrane funkcije cilja i prikladnosti, ali i operatori mutacije.

4.1 Početno rješenje

Kao što je već i navedeno, određeno je koristiti pohlepni algoritam kako bi se generiralo početno rješenje. Ono mora zadovoljiti sva jaka ograničenja:

- Zaposlenik može imati samo jednu smjenu dnevno.
- Neke smjene ne smiju ići jedna iza druge (primjerice, ne smije se dogoditi da zaposlenik koji je odradio noćnu smjenu, sljedeći dan bude u jutarnjoj smjeni).
- Svaki zaposlenik ima određen maksimalan broj smjena određene vrste koje mu smiju biti dodijeljene.
- Ukupan broj radnih sati koje zaposlenik smije odraditi mora biti u intervalu između unaprijed definirane minimalne i maksimalne vrijednosti.
- Svakom zaposleniku određen je minimalan i maksimalan broj uzastopnih smjena koje smije odraditi bez dana odmora.
- Određen je i minimalan broj uzastopnih dana odmora zaposlenika kako bi mu ponovno mogla biti dodijeljena smjena.
- Za planirano razdoblje za koje se određuje raspored određen je maksimalan broj radnih vikenda za članove osoblja.

- Točno su specificirani slobodni dani na koje zaposleniku ne smije biti dodijeljena nijedna smjena.

Za sva ograničenja pretpostavlja se povoljna situacija izvan promatranog razdoblja – ako, na primjer, zaposlenik mora odraditi barem 3 dana zaredom, a na granici promatranog razdoblja ima samo 2 radna dana zaredom, uzima se da je ograničenje zadovoljeno.

Slaba ograničenja, koja se koriste da bi izračunali trošak rješenja, su:

- Zahtjevi za dodjelu smjena – zaposlenik može zahtijevati da mu se dodijeli određena smjena na određeni dan, uz određeni trošak ako se ne dodijeli.
- Zahtjevi protiv dodjele smjena – zaposlenik može zahtijevati da mu se ne dodijeli određena smjena na određeni dan, uz određeni trošak ako se dodijeli.
- Pokrivanje smjena – svaki dan ima određen broj potrebnih zaposlenika za svaku smjenu te određene troškove za svakog zaposleniku iznad ili ispod tog broja (troškovi za premalo i previše zaposlenika ne moraju biti isti).

Za generiranje početnog rješenja odlučili smo promatrati svakog zaposlenika zasebno, jer jaka ograničenja djeluju samo u okviru jednog zaposlenika, a nikada ne uključuju više zaposlenika. Kada uspijemo generirati sve zaposlenike takve da zadovoljavaju jaka ograničenja, jednostavno ih sve spojimo u početno rješenje.

Najprije smo razmatrali situaciju u kojoj se dogodi da u rasporedu nekog zaposlenika imamo dane koji moraju biti slobodni, ali da je broj takvih uzastopnih dana manji nego minimalan broj uzastopnih dana odmora tog zaposlenika. U tom smo slučaju razmatrali sve kombinacije u kojima smo dodavali slobodne dane s prije i nakon dok ne bismo dobili moguće rješenje.

Pojednostavljenje kojem smo pribjegli u nastavku odnosi se na postojanje samo jedne imaginarne smjene koju se može dodijeliti te slobodnog vremena. Razlog ovom pristupu leži u činjenici da smo htjeli postići zadovoljenje ograničenja vezanog uz najveći broj radnih vikenda, dana odmora te najmanjeg i najvećeg broja odrađenih smjena bez slobodnog dana. Kad na ovaj način nađemo moguće rješenje, smjene zamjenjujemo stvarnim smjenama tako da dobiveno rješenje poštuje činjenicu da određene smjene ne smiju slijediti jedna iza druge te da neki zaposlenik ne radi pojedine smjene. U slučaju da se dogodi da netko radi previše (jer smjene nisu jednake duljine trajanja), dodajemo slobodne dane dok ne dobijemo moguće rješenje.

4.2 Heuristika lokalnog pretraživanja

Stavka koja je poprilično bitna jer ponajviše utječe na kvalitetu rješenja odnosi se na reprezentaciju rješenja. Budući da je implementacija u objektno orijentiranom jeziku, odlučili smo koristiti razred s dva polja; poljem zaposlenika i poljem dana. U prvo polje pohranjujemo informaciju o rješenju, tj. za svakog zaposlenika se pohranjuje informacija o dnevnim smjenama. Dakle, polje ima onoliko elemenata koliko ima zaposlenika za koje se određuje raspored, a svaki zaposlenik ima polje sa onoliko elemenata koliko je dana, popunjeno diskretnim vrijednostima smjena koje su pridružene tom zaposlenikua. Drugo se, pak, polje koristi kako bismo izračunali trošak po danima. Ocijenjeno je da je takav pristup nužan kako bismo mogli voditi računa o kaznama, tj. da bismo mogli dodati kaznu u slučaju prekomjernog broja zaposlenika ili njihovog nedostatka.

Kao što je već i uočeno, kao funkciju cilja koristila se minimizacija ukupnog troška rješenja jer je želja zadovoljiti što više preferencija pojedinih zaposlenika radi podizanja morala.

4.2.1 Funkcija prikladnosti

Funkcija prikladnosti je uvelike povezana s funkcijom cilja. Naime, neko rješenje je smatrano boljim u slučaju da mu je ukupni trošak manji od do tad najboljeg. To je moguće postići smanjenjem kazne koja proizlazi raspoređivanjem zaposlenika u smjene na dane kad je zaposlenik tražio da mu se ista ne dodijeli ili nedodjeljivanjem smjene zaposleniku u danu kad ju je tražio. Kazna se može smanjiti i ako se postigne smanjenje prekomjernog broja zaposlenika ili povećanjem njihova broja u slučaju malobrojnosti. Uobičajeno je za rješenja koje nisu moguća staviti vrijednost funkcije cilja na $+\infty$ kad je funkcija cilja minimizacija. No, u ovom se radu uopće ne razmatraju takva rješenja, tj. nemoguće rješenje neće proći provjeru zadovoljenja jakih ograničenja pa ih se ni ne pokušava mutirati.

4.2.2 Operator mutacije.

Temeljni dio pristupa kojeg smo razvili čine operatori mutacije koji unošenjem izmjena, kako u početno rješenje, tako i sva naknadna, pokušavaju izgraditi rješenje bliže optimalnom iznosu troška.

Vrijednosti koje koristimo kod određivanja složenosti mutatora:

D – broj dana,

Z – broj zaposlenika,

N – dubina,

s – broj smjena.

x – najveći maksimalni broj uzastopnih smjena bilo kojeg zaposlenika

R – prosjek radnih dana svih zaposlenika

Značenje simbola u matrici:

x – smjena koju mutator ne gleda

o – slobodno vrijeme koje mutator ne gleda

X – smjena koju mutator gleda

O – slobodno vrijeme koje mutator gleda

Pri osmišljavanju jednog od njih vodili smo se idejom pretraživanja u dubinu gdje smo parametar dubine promatrali kao broj promjena. Time smo odredili koje kombinacije gledamo. Primjerice, ako zamislimo matricu u kojoj su stupci dani, a retci zaposlenici, dubina 2 bi značila da promatramo sve skupove promjena veličine 2 (jedna promjena određena je zaposlenikom, danom i smjenom), te uzimamo onaj skup čija primjena ne narušava jaka ograničenja i daje najmanji trošak.

Složenost ovog mutatora je

$$O\left(\binom{D*Z}{N}*(s+1)^N\right)$$

Ovaj mutator gleda sve dane i zaposlenike:

$$\begin{pmatrix} X & O & X & X & X & O & O & X \\ X & X & X & X & X & O & O & O \\ X & O & X & X & X & X & O & X \\ O & O & O & X & O & X & X & X \end{pmatrix}$$

Pojednostavljeni pseudokod ovog mutatora za dvije promjene je:

```
primijeniSetMutator(Zaposlenik[] zaposlenici,Dan[] dani,Smjena[] smjene,int brojPromjena=2){
    int trenutnoRjesenje=evaluiraj();
    Zaposlenik[] najboljeRjesenje=NULL;
    za(Zaposlenik zaposlenik1:zaposlenici){
        za(Dan dan1:dani){
            Smjena staraSmjena1=zaposlenik1[dan1];
            za(Smjena smjena1:smjene){
                zaposlenik1[dan1]=smjena1;
                za(Zaposlenik zaposlenik2:zaposlenici){
                    za(Dan dan2:dani){
                        Smjena staraSmjena2=zaposlenik2[dan2];
                        za(Smjena smjena2:smjene){
                            zaposlenik2[dan2]=smjena2;
                            int rjesenje=evaluiraj();
                            ako(moguceRjesenje(zaposlenici)&&rjesenje<trenutnoRjesenje){
                                trenutnoRjesenje=rjesenje;
                                najboljeRjesenje=kopiraj(zaposlenici);
                            }
                        }
                        zaposlenik2[dan2]=staraSmjena;
                    }
                }
            }
            zaposlenik1[dan1]=staraSmjena1;
        }
    }
    vrati najboljeRjesenje;
}
```

Sljedeći operator mutacije kojeg smo primijenili stavlja fokus na ograničenja po danima. Odnosno, možemo zamisliti istu situaciju kao i kod prethodnog operatora, no

tako da promatramo samo promjene u jednom danu, i to činimo za sve dane.

Složenost ovog operatora je stoga:

$$O\left(D * \binom{Z}{N} * (s+1)^N \right)$$

Ovaj mutator gleda sve dane jedan po jedan i sve zaposlenike, za prvi dan:

$$O\left(Z * \frac{R}{x} * (s+1)^x \right)$$

Dakle, nastoji se za svaki dan postići takvu distribuciju smjena zaposlenicima da se trošak rješenja smanji, ali da rješenje ostane moguće.

Pseudokod ovog mutatora (prvi mutator se koristi zbog jednostavnosti) je:

```
primijeniVertikalniMutator(Zaposlenik[] zaposlenici,Dan[] dani,Smjena[] smjene,int brojPromjena){
    za(Dan dan:dani){
        Zaposlenici[] z=primijeniSetMutator(zaposlenici,Dan[]{dan},smjene,brojPromjena);
        ako(z!=NULL){
            zaposlenici=z;
        }
    }
    vrati zaposlenici;
}
```

Promatrajući dobivena rješenja, uočili smo da određeno poboljšanje može slijediti iz mutiranja sekvenci. Naime, sekvencom smatramo slijed dana u kojima neki zaposlenik radi bez odmora. Ovaj mutator za svaku sekvencu svakog zaposlenika nastoji pronaći onu kombinaciju smjena koja će rezultirati najmanjim troškom što u konačnici dovodi do složenosti:

$$O\left(Z * \frac{R}{x} * (s+1)^x \right)$$

Za jednu sekvencu zaposlenika u trećem danu:

$$\begin{pmatrix} x & o & x & x & x & o & o & x \\ x & x & x & x & x & o & o & o \\ x & o & X & X & X & X & o & x \\ o & o & o & x & o & x & x & x \end{pmatrix}$$

Pseudokod ovog mutatora je:

```
primijeniMutatorSekvence(Zaposlenik[] zaposlenici,Dan[] dani,Smjena[] smjene,int brojPromjena){
    za(Zaposlenik zaposlenik:zaposlenici){
        Dan[] početciSekvenci=nadjiPocetkeSekvenci(zaposlenik);
        Dan[] krajeviSekvenci=nadjiKrajeveSekvenci(zaposlenik);
        za(int index=0;index<početciSekvenci.velicina;index++){
            Dan pocetak=početciSekvenci[index];
            Dan kraj=krajeviSekvenci[index];
            Dan[] mDani={pocetak,pocetak+1,pocetak+2,...,kraj};
            Zaposlenik novi=primijeniSetMutator(zaposlenik,mDani,smjene,mDani.velicina);
            ako(novi!=NULL){
                kopirajIzU(novi,zaposlenik);
            }
        }
    }
    vrati zaposlenici;
}
```

Ukoliko se uoči neko preklapanje među zaposlenicima u sekvencama (dva zaposlenika imaju sekvence koje dijele barem jedan dan), nastoje se naći kombinacije smjena u tim sekvencama koje će voditi željenom smanjenju funkcije cilja.

Složenost ovog mutatora iznosi:

$$O\left(\binom{Z}{2} * (s+1)^{2x} * K\right)$$

pri čemu pretpostavljamo da svaka sekvenca nekog zaposlenika ima neku u drugom zaposleniku s kojom ima zajedničkih dana.

Za jedno preklapanje sekvenci zaposlenika u trećem i četvrtom redu:

$$\begin{pmatrix} x & o & x & x & x & o & o & x \\ x & x & x & x & x & o & o & o \\ x & o & X & X & X & X & o & x \\ o & o & o & x & o & X & X & X \end{pmatrix}$$

Pseudokod ovog mutatora je:

```
primijeniMutatorPreklapanjaSekvenci(Zaposlenik[] zaposlenici,Dan[] dani,Smjena[] smjene,int brojPromjena){
    za(Zaposlenik zaposlenik1:zaposlenici){
        za(Zaposlenik zaposlenik2:zaposlenici){
            ako(zaposlenik1==zaposlenik2){
                nastavi;
            }
            Dan[] početciSekvenci1=nadjiPocetke(zaposlenik1);
            Dan[] krajeviSekvenci1=nadjiKrajeve(zaposlenik1);
            Dan[] početciSekvenci2=nadjiPocetke(zaposlenik2);
            Dan[] krajeviSekvenci2=nadjiKrajeve(zaposlenik2);
            za(int index1=0;index1<početciSekvenci1.velicina;index1++){
                za(int index2=0;index2<početciSekvenci2.velicina;index2++){
                    Dan pocetak1=početciSekvenci1[index1];
                    Dan kraj1=početciSekvenci1[index1];
```

```
                    Dan pocetak2=početciSekvenci2[index2];
                    Dan kraj2=početciSekvenci2[index2];
                    ako(!postojiPreklapanje(pocetak1,kraj1,pocetak2,kraj2)){
                        nastavi;
                    }
                    Zaposlenik[] novi =
nadjiNajboljuKombinaciju(zaposlenik1,zaposlenik2,pocetak1,kraj1,pocetak2,kraj2,smjene);
                    ako(novi!=NULL){
                        kopirajIzU(novi[0],zaposlenik1);
                        kopirajIzU(novi[1],zaposlenik2);
                    }
                }
            }
        }
    }
    vrati zaposlenici;
}
```

5. Rezultati

U ovom poglavlju su prikazani rezultati primijenjenog pristupa algoritma na različitim instancama problema alokacije smjena po danima članovima osoblja. U prvom potpoglavlju je opisano koje smo podatke smo rabilikoristili radi provjere uspješnosti ostvarenog heurističkog algoritma dok se drugo bavi načinima validacije rješenja. U posljednjem potpoglavlju prikazani su rezultati postignuti izrađenom heuristikom.

5.1 Skup instanci problema

Budući da se radi o specifičnom problemu za koji ne postoji jedinstveno rješenje, bilo je potrebno naći adekvatnu bazu podataka koju bismo mogli koristiti za ispitivanje točnosti rada i performansi algoritma te njegovih značajki. U tu svrhu **korišten je skup instanci (njih 24) preuzet sa stranica Sveučilišta u Nottinghamu, tj. njihove istraživačke grupe iz područja računarske znanosti** [16]**.** Jako su izazovne za rješiti budući da sadrže velik broj različitih ograničenja i ciljeva organizacija koje je teško modelirati.

Tablica 1 prikazuje ove instance poredane po veličini. Prikazani su broj zaposlenika, smjena i duljina vremenskog razdoblja u kojem se određuje raspored. Stupac koji sadrži vrijednosti složenosti pretraživanja grubom silom (eng. *brute force*) ilustrira stvarnu veličinu problema u pogledu broja kombinacija koje je nužno pretražiti kako bi se našlo optimalno rješenje.

Tablica 1. Prikaz korištenih instanci poredanih po veličini i složenosti pretraživanja

Instanca	Broj zaposlenika	Broj dana	Broj smjena	Složenost pretraživanja grubom silom
instanca 1	8	14	1	10^{33}
instanca 2	14	14	2	10^{93}
instanca 3	20	14	3	10^{168}
instanca 4	10	28	2	10^{133}
instanca 5	16	28	2	10^{213}
instanca 6	18	28	3	10^{303}
instanca 7	20	28	3	10^{337}
instanca 8	30	28	4	10^{587}
instanca 9	36	28	4	10^{704}
instanca 10	40	28	5	10^{871}
instanca 11	50	28	6	10^{1183}
instanca 12	60	28	10	10^{1749}
instanca 13	120	28	18	10^{4296}
instanca 14	32	42	4	10^{939}

instanca 15	45	42	6	10^{1597}
instanca 16	20	56	3	10^{674}
instanca 17	32	56	4	10^{1252}
instanca 18	22	84	3	10^{1112}
instanca 19	40	84	5	10^{2614}
instanca 20	50	182	6	10^{7960}
instanca 21	100	182	8	10^{17367}
instanca 22	50	364	10	10^{18953}
instanca 23	100	364	16	10^{44788}
instanca 24	150	364	32	10^{82910}

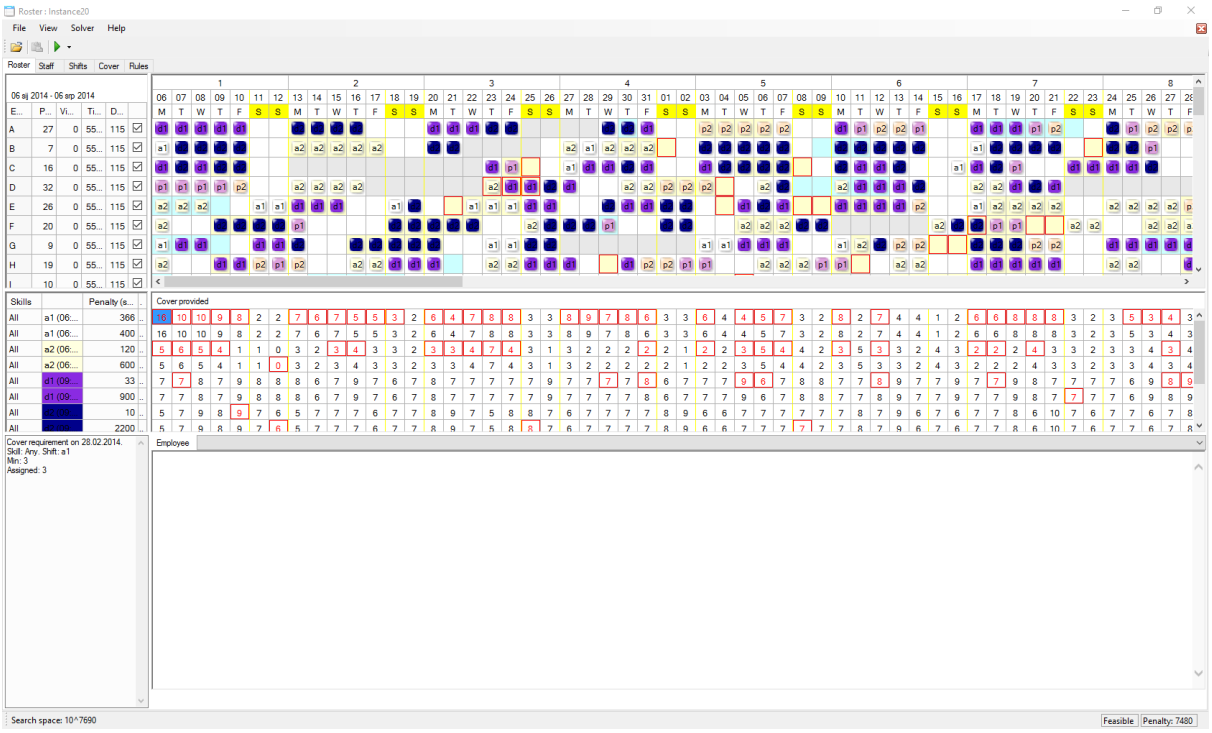
Instance su dostupne u jednostavnom tekstualnom obliku ili u XML formatu što ih čini jednostavnim za parsiranje i predstavljanje u računalnom programu, no svejedno su oblikovane tako da ih je velik izazov riješiti. Međusobno se poprilično razlikuju po veličini problema. Naime, ona se odnosi na broj članova osoblja, duljinu vremenskog perioda te količinu ograničenja što otežava nalazak početnog rješenja, ali i bilo kakvo mutiranje i poboljšavanje.

Na početku su manje instance kojima je već pronađeno optimalno rješenje dok većim instancama najbolja rješenja zastaju na vrijednostima koja su u nekim slučajevima poprilično daleko od optimalnih.

5.2 Validacija rješenja

Validacija rješenja je rađena dvojako. Naime, radi se o problematici za koju nije moguće reći da je dobiveno rješenje ako nisu zadovoljena sva jaka ograničenja te je sama činjenica da je algoritam postigao moguće rješenje, znak da je ono i valjano. Ipak, kako bi se imala određena sigurnost u nenarušavanje ograničenja, izradili smo validator u C# programskom jeziku koji učitava rješenja te ispituje postoji li neka situacija u kojoj bi se moglo reći da rješenje nije valjano, tj. moguće u pogledu optimizacijskih problema.

Drugi način koji je korišten za ispitivanje valjanosti rješenja je *Roster Booster*. Radi se o grafičkom korisničkom sučelju koje se može koristiti za verifikaciju postignutih rješenja. Ova aplikacija se uobičajeno koristi u svrhu ispitivanja postignutih rezultata u rješavanju problema raspodjele smjena, a raspoloživa je za Windows platforme. Podržava rad s instancama velikog raspona ograničenja, zahtjeva i ciljeva koje treba distribucija zadovoljiti.



Slika 3. Grafičko sučelje Roster Booster programa

Ipak, u pogledu rezultata važnih u znanstvene svrhe, važnije je bilo provjeriti koliko je dobro dobiveno rješenje, nego utvrđivanje njegove validnosti. Upravo je kvaliteta postignutih rezultata u sklopu ovog rada predmet sljedećeg poglavlja.

5.3 Evaluacija postignutih rezultata

Rješenja koja smo dobili pomoću algoritma predloženog u ovome radu uvelike poboljšavaju do sad poznato i najbolje rješenje za skoro sve veće instance. Na manje se nismo toliko fokusirali jer na onima koje smo provjerili, naše je rješenje ili bilo optimalno ili bilo blizu optimalnog.

Kao primjer za evaluciju kvalitete rješenja možemo navesti rezultate postignute na instanci 20. Naša distribucija smjena po danima za sve zadane članove osoblja poboljšala je najbolje poznato rješenje za nešto više od 1 000 jedinica troška. Pri tome treba imati na umu da smo prekinuli rad algoritma nakon prosječno deset sati

za svaku instancu što znači da je moguće da naša implementacija rješenja ovog problema raspoređivanja osoblja nađe i još bolje rješenje te time bliže optimalnom.

U tablici 2 su predstavljene vrijednosti do sad najboljih poznatih rješenja pojedinih instanci analiziranog problema prikazanih u prethodnoj tablici te iznosi funkcije cilja do kojih smo uspjeli doći primjenom lokalnog pretraživanja. Stupac s prikazom do sad najboljih rješenja odnosi se vrijednosti koje su postignute primjenom različitih algoritama čije su donje granice (eng. *lower bound*) predočene također u tablici 2. Ove dvije vrijednosti preuzete su iz tehničkog izvješća [17] o računalnim rezultatima na novim instancama za izradu rasporeda autora Tima Curtoisa i Rong Qua sa Sveučilišta u Nottighamu.

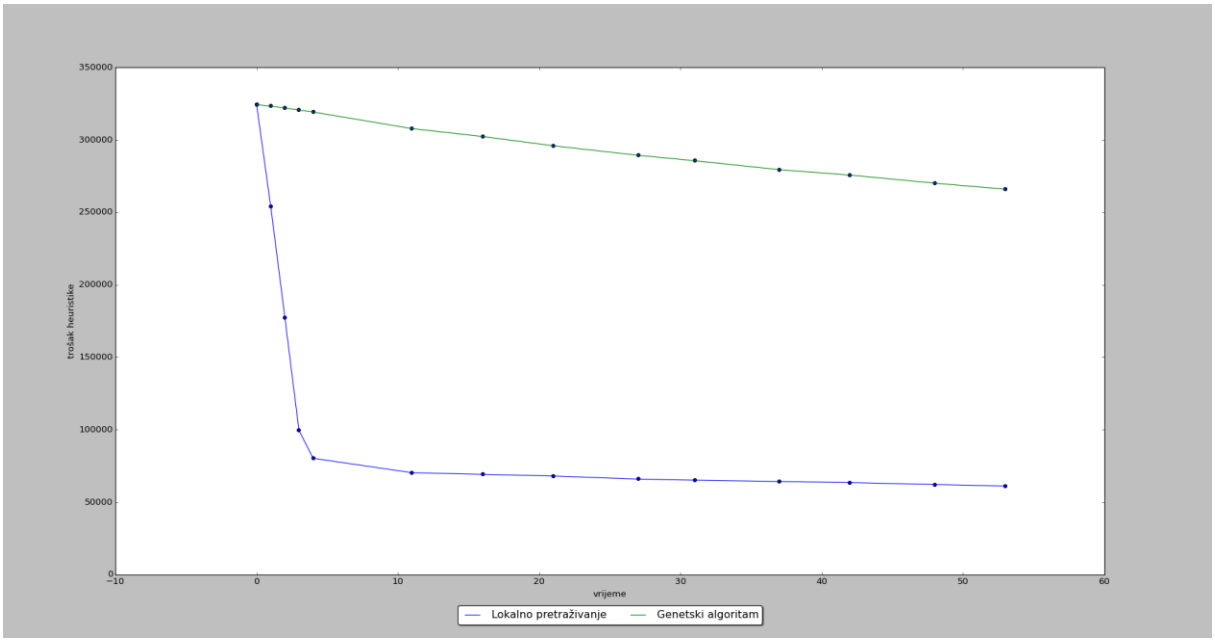
Tablica 2. Analiza do sad najboljih poznatih rješenja i vrijednosti dobivenih lokalnim pretraživanjem za pojedine instance

INSTANCE	***DONJA GRANICA***	***NAJBOLJA POZNATA RJEŠENJA***	***HEURISTIKA LOKALNOG PRETRAŽIVANJA***
Instanca 18	4351	4 760	6 142
Instanca 20	4743	7 480	6 452
Instanca 22	24064	46 849	46 511
Instanca 23	2765	44 189	21 820
Instanca 24	?	71 765	61 164

U tablici 2 su prikazane pojedine instance koje smo izabrali zbog njihove veličine. Kao što je i vidljivo, rezultati u gotovo svim slučajevima (osim kod instance 18) poboljšavaju dosad najbolje poznato rješenje. Zbog brojnosti instanci te količine ograničenja i resursa, algoritam je zaustavljan nakon prosječno deset sati po instanci kako bismo provjerili njegov rad na svim instancama i uvjerili se u ispravnost izgrađene heuristike.

Usporedba s genetskim algoritmom.

Radi usporedbe, razvili smo i algoritam koji se temelji na korištenju genetskog algoritma po uzoru na [3] i [8]. Ova heuristika je isprva jako brzo spuštala trošak rješenja, no npr. za instancu 20 bi zapela u lokalnom optimumu s troškom oko 70 000 te je bilo očito kako genetski algoritam ne funkcionira najbolje u ovom tipu problema, barem ne sam, te ga je potrebno kombinirati s drugim pristupima [12]. Graf 1 prikazuje usporedbu rada algoritma temeljenog na genetskom algoritmu i pristupa koji je fokus ovog rada – lokalnog pretraživanja s operatorima mutiranja. Vidljivo je kako u istom vremenskom razdoblju, genetski algoritam prikazan zelenom bojom puno sporije dovodi do poboljšanja funkcije cilja.



Slika 4. Usporedba rada genetskog algoritma i implementirane heuristike u vremenu od 53 sekunde

Pseudokod genetskog algoritma je:

```
genetskiAlgoritam(Rješenje[] rješenja, Smjena[] smjene,int brojIteracija,int postotakNajgorih){
        Rješenje globalnoNajbolje=najboljeRješenje(rješenja);
        int iteracija=0;
        dok(iteracija<brojIteracija){
                iteracija++;
                za(Rješenje rješenje:rješenja){
                        Zaposlenik zaposlenik=nasumičniZaposlenik(rješenje);
                        Dan dan=nasumičniDan(rješenje);
                        Smjena staraSmjena=zaposlenik[dan];
                        Smjena smjena=nasumičnaSmjena(smjene);
                        zaposlenik[dan]=smjena;
                        ako(!moguceRjesenje(rješenje)){
                                zaposlenik[dan]=staraSmjena;
                        }
                }
                Rješenje najbolje=najboljeRješenje(rješenja);
                ako(evaluiraj(najbolje)<evaluiraj(globalnoNajbolje)){
                        globalnoNajbolje=najbolje;
```

```
                }
                sortirajRješenjaPoTrošku();
                izbaciPostotakNajgorihRješenja(postotakNajgorih);
                nadomjestiIzbačenaRješenjaSaGlobalnim();
        }
        vrati globalnoNajbolje;
}
```

6. Rasprava

Uzevši u obzir rezultate rada prikazane i obrazložene u prethodnom poglavlju, nužno je iznova načiniti procjenu uspješnosti izgrađene heuristike temeljene na aproksimativnoj metodi lokalnog pretraživanja. Ona treba uključivati validaciju točnosti i značaja rezultata koje je ista generirala na primijenjenom skupu podataka, tj. instanci ovog problema korištenjem osmišljenih operatora mutacije. Iz takve procjene se može izvući konačan zaključak o kvaliteti implementirane heuristike, ali i važnosti dobivenih rezultata za napredak istraživanja u ovom području.

Različiti faktori utječu na određivanje uspješnosti algoritma. Naime, jedan od izuzetno bitnih jest kvaliteta podataka koji se koriste za ispitivanje rada algoritma. Ispitni podaci su bile instance različite veličine i broja ograničenja, zaposlenika, dana i smjena što je jamčilo da ovaj faktor nikako ne može negativno utjecati na određivanje uspješnosti algoritma.

Ipak, najvažniji faktor su sami rezultati. Naime, postoji više algoritama koji mogu naći rješenje u smislu poštivanja jakih ograničenja, no njihova se uspješnost u području primjene optimizacijskih metoda najviše očituje u vrijednosti funkcije cilje, odnosno u tome koliko je rješenje blizu optimalnog. Usporedbe s do sad najboljim rješenjima pokazuju popriličnu uspješnost osmišljene heuristike, a činjenica da se rezultati bolji od do sad poznatih dobivaju na više instanci, ne samo na jednoj, jasno upućuju na zaključak o značajnoj kvaliteti dobivenih rezultata ovog rada.

Još bolje rezultate bismo mogli dobiti u slučaju da se odlučimo na paralelizaciju algoritma čime bismo neke operacije, poglavito mutaciju i dijelove vezane uz generiranje početnog rješenja mogli dodatno ubrzati. Tako bismo vjerojatno dobili i

bolje vrijednosti u puno manjem vremenskom razdoblju za skoro sve, ako ne i sve instance.

7. Zaključak

Područje izrade optimalnog rasporeda resursa jako je dinamično te se neprestano razvijaju novi algoritmi koji nastoje na što bolji način prikazati alokaciju resursa pri tome pokrivajući velk broj različitih instanci. Heuristički pristup predočen u ovom radu temelji se na primjeni aproksimativne metode lokalnog pretraživanja temeljene na različitim operatorima mutacije.

U radu se koristi dosta jednostavan pristup generiranju početnog rješenja kao polazišta za implementiranu heuristiku. Nije bilo potrebe razvijati kompleksne metode u ovu svrhu ili koristiti neke pristupe temeljene na matematičkom modeliranju [15] budući da je cilj dobiti inicijalno rješenje koje će ionako heuristika, ako je dobra, uspjeti izmijeniti i u konačnici dovesti do optimalne vrijednosti.

Iako se na prvi pogled može činiti dosta jednostavnim u usporedbi s idejama iznesenim u radovima [6], [7], [8] i [9], vrijednost metode opisane u ovom radu leži u primijeni operatora mutacije koji vode pretraživanje prostora rješenja istražujući različite kombinacije i birajući onu s najmanjim troškom. Pri tome se uopće ne razmatraju rasporedi koji ne zadovoljavaju ograničenja, a prihvaćaju se samo rješenja istog ili manjeg troška od trenutno razmatranog.

Algoritam lokalnog pretraživanja je primijenjen na različitim instancama čije veličine variraju čime je njegova uspješnost još od većeg značaja. **Analizom korištene baze primjera za ispitivanje valjanosti algoritma, uočeno je zamjetno poboljšanje do sad poznatih vrijednosti troška instanci**. Time je postignut cilj ovog rada, a koji se sastojao od toga da se nađe heuristika koja će na više instanci uspjeti naći ili optimalno rješenje ili poboljšati do sad najbolje poznato.

8. Zahvale

Željeli bismo se zahvaliti doc. dr. sc. Lei Skorin-Kapov na mentorstvu i podršci prilikom izrade ovog rada.

9. Literatura

[1] Talbi, E. G. (2009). *Metaheuristics: from design to implementation* (Vol. 74). John Wiley & Sons.

[2] Bester, M. J., Nieuwoudt, I., & Van Vuuren, J. H. (2007). Finding good nurse duty schedules: a case study. *Journal of Scheduling*, 10(6), 387-405.

[3] Puente, J., Gómez, A., Fernández, I., & Priore, P. (2009). Medical doctor rostering problem in a hospital emergency department by means of genetic algorithms. *Computers & Industrial Engineering*, 56(4), 1232-1242.

[4] Bard, J. F., & Purnomo, H. W. (2004). Real-time scheduling for nurses in response to demand fluctuations and personnel shortages. In *Proceedings of the 5th international conference on the practice and theory of automated timetabling* (pp. 67-87).

[5] Brusco, M. J., Jacobs, L. W., Bongiorno, R. J., Lyons, D. V., & Tang, B. (1995). Improving personnel scheduling at airline stations. *Operations Research, 43*(5), 741-751.

[6] Dowsland, K. A., & Thompson, J. M. (2000). Solving a nurse scheduling problem with knapsacks, networks and tabu search. *Journal of the Operational Research Society*, 51(7), 825-833.

[7] Chiarandini, M., Schaerf, A., & Tiozzo, F. (2000). Solving employee timetabling problems with flexible workload using tabu search. In *Proceedings of the 3 rd Int. Conf. on the Practice and Theory of Automated Timetabling (E. Burke & W. Erben, Eds.) pp* (pp. 298-302).

[8] Kawanaka, H., Yamamoto, K., Yoshikawa, T., Shinogi, T., & Tsuruoka, S. (2001). Genetic algorithm with the constraints for nurse scheduling problem. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on* (Vol. 2, pp. 1123-1130). IEEE.

[9] Reeves, C. R. (1993). *Modern heuristic techniques for combinatorial problems.* John Wiley & Sons, Inc..

[10] Jaumard, B., Semet, F., & Vovor, T. (1998). A generalized linear programming model for nurse scheduling. *European journal of operational research*, 107(1), 1-18.

[11] Aickelin, U., Burke, E. K., & Li, J. (2007). An estimation of distribution algorithm with intelligent local search for rule-based nurse rostering. *Journal of the Operational Research Society*, 1574-1585.

[12] Bailey, R. N., Garner, K. M., & Hobbs, M. F. (1997). Using simulated annealing and genetic algorithms to solve staff-scheduling problems. *Asia-Pacific journal of operational research, 14*(2), 27.

[13] Bellanti, F., Carello, G., Della Croce, F., & Tadei, R. (2004). A greedy-based neighborhood search approach to a nurse rostering problem. *European Journal of Operational Research*, 153(1), 28-40.

[14] Wong, G. Y. C., & Chun, H. W. (2003). Nurse rostering using constraint programming and meta-level reasoning. In *Developments in Applied Artificial Intelligence* (pp. 712-721). Springer Berlin Heidelberg.

[15] Beaumont, N. (1997). Scheduling staff using mixed integer programming. *European journal of operational research*, 98(3), 473-484.

[16] *Employee Shift Scheduling Benchmark Data Sets*, dostupno na adresi: *http://www.cs.nott.ac.uk/~psztc/NRP/*, datum pristupa: 27.4.2016.

[17] Curtois, T., & Qu, R. Computational results on new staff scheduling benchmark instances.

10. Sažetak

Autori: Matea Pejčinović, Fran Stanić

Primjena lokalnog pretraživanja u rješavanju problema izrade rasporeda zaposlenika

Problem raspoređivanja osoblja prilično je zastupljen u različitim granama industrije i ljudskih djelatnosti. Do sad nije poznat algoritam koji ga na pouzdan način u razumnom vremenu rješava već se primjenjuju mnogobrojni heuristički pristupi koji u prihvatljivom vremenu generiraju optimalna rješenja sa stajališta distribucije resursa. Njihova prihvatljivost varira u vrijednosti pripadajućeg troška pa je iznimno bitno naći pristup koji bi istodobno zadovoljio različita ograničenja postavljena na problem dovodeći pritom do malog troška alokacije resursa. Ovaj rad predstavlja inovativan pristup rješavanju raspodjele zaposlenika po smjenama pri čemu je bitno maksimizirati raspodjelu osoblja po zahtjevima te minimizirati broj nedovoljnog ili prekobrojnog osoblja po smjenama. Uočeno je kako pristup lokalnog pretraživanja u kombinaciji sa zamišljenim operatorima mutacije uvjerljivo dominira nad ostalim isprobanim heuristikama. Rezultati dobiveni ovim aproksimativnim pristupom su uspjeli poboljšati do sad najbolje poznato rješenje za veći broj poznatih instanci problema te su se poprilično približili njihovoj donjoj granici.

Ključne riječi: problem raspoređivanja, heuristički pristupi, alokacija resursa, lokalno pretraživanje, aproksimativni pristup, operatori mutacija

11. Summary

Authors: Matea Pejčinović, Fran Stanić

A Local Search Heuristic Approach in Solving the Employee Rostering Problem

The employee rostering problem is a well studied topic in the field of optimisation methods. One of its forms was the starting point for the work done in this report which aimed to propose a heuristic algorithm that found improved solutions as compared to those previously published. This paper presents the results we have accomplished by experimenting with several algorithms. Some of them showed to be less adequate for the scheduling problem we were dealing with while local search with a number of mutators have lead to significant improvements of known results. Considering that the way of generating an initial solution plays an important role in the speed and overall performance of the chosen algorithm, efforts were invested to derive innovative methods for finding feasible initial solutions with acceptable cost. The algorithm was tested on multiple well know problem instances and obtained results provide improved solutions as compared to the best known currently available solutions.

Keywords: employee rostering problem, heuristic optimisation, distribution of resources, local search, approximate methods, mutation operators