SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 225

Optimizacija rasporeda smjena medicinskih sestara

Luka Matijević

Zagreb, srpanj 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 12. ožujka 2021.

ZAVRŠNI ZADATAK br. 225

Pristupnik:	**Luka Matijević (0036517220)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	doc. dr. sc. Marko Đurasević

Zadatak: **Optimizacija rasporeda smjena medicinskih sestara**

Opis zadatka:

Proučiti i opisati problem optimiranja rasporeda smjena medicinskih sestara u bolničkim sustavima. Navesti do sada upotrebljavane metode u literaturi za rješavanje prethodno navedenog problema. Predložiti i prilagoditi metaheurističku metodu za optimiranje problema izrade rasporeda smjena. Primijeniti predloženu metodu na skupu problema izrade rasporeda smjene rasporeda. Analizirati rezultate i ocijeniti učinkovitost predložene metode. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna objašnjenja i korištenu literaturu.

Rok za predaju rada: 11. lipnja 2021.

SADRŽAJ

1. Uvod

Problem izrade optimalnog rasporeda smjena radnika podskup je optimizacijskih problema. To je često prisutan problem u velikim organizacijama kao što su uslužni centri, obrazovne ustanove i drugi veliki javni entiteti. Svoju primjenu često pronalazi u bolničkim sustavima pri raspoređivanju smjena medicinskih sestara (*engl.* Nurse Scheduling Problem - NSP).

Ovaj rad detaljnije istražuje taj problem i heurističke metode rješavanja vezane uz njega. Opisat će se optimizacijski problem općenito te moguće heurističke metode za njegovo rješavanje. Konkretni problem bit će opisan formalno i ponudit će se njegovo implementacijsko rješenje. Ukratko, NSP se sastoji od određenih jakih ograničenja koja se moraju poštovati i slabih ograničenja čije poštovanje dovodi do boljeg rezultata. Dakle, svako validno rješenje morat će zadovoljiti bolničke zahtjeve i u isto vrijeme težiti maksimizaciji zadovoljstva medicinskih sestara ispunjavajući njihove želje. Budući da ovaj problem ima veliki broj različitih rješenja, njega neće biti moguće riješiti tradicionalnim pretraživanjem, već će se koristiti metaheurističke metode koje će usmjeravati algoritam prema rješenju i na taj način smanjiti vrijeme pretrage.

Na početku opišimo detaljnije što je to optimizacijski problem i prikladne metode kojima se on može riješiti.

2. Optimizacijski problem

Optimizacijski problem (*engl.* Optimization Problem - OP) matematički je problem čiji je cilj pronaći najbolje rješenje u skupu svih rješenja. OP sastoji se od različitih ograničenja koje svako rješenje mora poštovati. Dakle, cilj je pronaći optimalno rješenje koje zadovoljava navedene probleme. Na primjer, cilj može biti odredište do kojeg putnik želi doći pa u tom slučaju funkcija traži minimalan put do tog grada. Također, optimizacijski problem može se primijeniti pri kalkulaciji minimalnog troška pri obavljanju nekog posla. Svakodnevni primjer na sveučilištima je i izrada studentskih kalendara, kako za nastavu, tako i za ispite. U svim navedenim slučajevima, problem definira neka ograničenja (*engl. constraints*) koja je potrebno zadovoljiti. Prilikom pronalaska optimalnog rješenja, algoritam nastoji maksimizirati, odnosno minimizirati ciljnu funkciju, ovisno o kontekstu problema. Funkcija cilja je realna funkcija čijim izračunom nad pojedinim rješenjem saznajemo koliko dobro to rješenje zadovoljava zahtjeve zadane problemom.

Često se optimizacijski problem može prikazati sljedećom definicijom: Pronađi najveću ili najmanju (ovisno o kontekstu) vrijednost funkcije [7]:

$$f(x)$$

kada vrijedi

$$a \leq x \leq b$$

U navedenom zapisu konstante *a* i *b* su izravna posljedica ograničenja koja predstavlja problem. Naravno, problem ne traži maksimalnu ili minimalnu vrijednost u intervalu [a, b], već maksimalnu ili minimalnu vrijednost koju funkcija

$$f(x)$$

može poprimiti unutar tog intervala.
Pri rješavanju ovakvog problema bitno je definirati za što točno tražimo optimalnu vrijednost i definirati ograničenja. Prije nego dublje razradimo problem raspoređivanja

smjena medicinskih sestara, pokušajmo ga detaljnije analizirati kako bismo pronašli najbolji pristup njegovu rješavanju.

2.1. Optimizacija rasporeda smjena

Problem optimizacije rasporeda smjena medicinskih sestara problem je raspoređivanja čije rješenje mora zadovoljiti bolničke zahtjeve i pritom pokušati maksimalno udovoljiti željama medicinskih sestara. Dakle, bolnički zahtjevi su jaka ograničenja koja se moraju poštovati, dok su zahtjevi medicinskih sestara slaba ograničenja čije će poštivanje povećati vrijednost rješenja. Zaključujemo da je to takozvani 'Constraint Satisfaction Problem - CSP'. CSP probleme općenito karakterizira potraga optimalnog rješenja imajući na umu neka ograničenja.

Problem je potrebno riješiti u navedenom broju dana nad zadanim skupovima medicinskih sestara i smjena. Matematički je dokazano da je ovaj problem NP-težak. Ukratko, to znači da taj problem ima faktorijelnu složenost što predstavlja veliki vremenski problem. Samim time, za pronalazak optimalnog rješenja nije prikladno koristiti tradicionalne metode pretraživanja jer bi one u većini slučajeva (nad iole većim skupovima mogućih rješenja) trajale predugo.
Za rješavanje je bolje koristiti metaheurističke metode koje pomažu u potrazi optimalnog rješenja. Takve metode općenito nastoje pronaći 'zadovoljavajuće' rješenje u slučajevima kada su tradicionalne metode prespore ili kada one ne mogu pronaći točno rješenje, pogotovo u slučajevima kada nedostaju krucijalne informacije ili su one jednostavno šture.
One usmjeravaju algoritam prema optimalnom rješenju. Na taj način obavlja se takozvano parcijalno pretraživanje u kojem nije potrebno ispitati svaku moguću kombinaciju rješenja, već algoritam traži "u smjeru" dobrog rješenja. Iako metaheuristike neće uvijek biti u mogućnosti pronaći optimalno rješenje, njihov cilj je ponuditi dovoljno dobro rješenje. Takve algoritme stoga često koristimo nad velikim skupovima podataka s ciljem da problem svedemo na polinomijalnu složenost. Navedimo neke primjere metaheuristika:

- simulirano kaljenje
- tabu pretraživanje
- evolucijski algoritmi
- mravlji algoritmi

– algoritmi rojeva

Budući da problem rasporeda smjena ima veliki broj mogućih rješenja, za njega je najprikladniji evolucijski algoritam pa istražimo detaljnije što se krije iza tog pojma. Kao što i samo ime govori, to su algoritmi inspirirani evolucijom čija je odlika iterativno poboljšanje rješenja. Tijekom druge polovice 20. stoljeća, u vrijeme prvih detaljnijih istraživanja ovih pristupa, evolucijski algoritmi razvili su se u nekoliko smjerova [1]. To su evolucijsko programiranje, evolucijske strategije, genetski algoritmi te genetsko programiranje.
Optimizacijske probleme općenito prikladno je riješavati iterativno, generacijski. Na taj način moguće je pratiti razvoj najboljih rješenja koja svakom generacijom napreduju. Svaka generacija ima određenu količinu jedinki koje nastaju kao produkti križanja najboljih jedinki prijašnje generacije. Takav pristup odgovara (generacijskom) genetskom algoritmu. Stoga ćemo u nastavku detaljno opisati njegov način rada kako bismo ga mogli primijeniti pri optimizaciji smjena medicinskih sestara.

2.2. Genetski algoritam

Genetski algoritam (GA) metaheuristički je algoritam čije je obilježje "opstanak najboljih jedinki". Taj pristup inspiriran je Darwinovom teorijom razvoja vrsta te je on jedan od najpoznatijih evolucijskih algoritama. Stvorio ga je H. J. Holland 1975. godine [1]. Ne postoji standardizirani postupak rada genetskog algoritma budući da se njegova implementacija razlikuje od problema do problema te ovisi o kontekstu u kojem se on koristi. Usprkos tome, u nastavku ćemo ukratko opisati njegov okviran način rada koji se naknadno prilagođava pojedinom problemu.

Na početku algoritam generira nasumične jedinke. U stvarnome životu, biološka jedinka sadrži neka svoja obilježja koja ju karakteriziraju. Na isti način se i u implementaciji algoritma svakoj jedinci pridjeljuju kromosomi koji su specifični za tu jedinku. Nakon generiranja jedinki, procjenjuje se njihova vrijednost (*engl.* fitness value) ovisna o tome koliko jedinka zadovoljava zahtjeve. Nekoliko najboljih jedinki trenutačne populacije prenosi se u sljedeću generaciju bez promjene njihovih kromosoma. Taj postupak naziva se elitizam. Iz skupa kvalitetnijih jedinki, odnosno onih s većom dobrotom, biraju se roditelji na temelju kojih se križanjem generiraju njihova

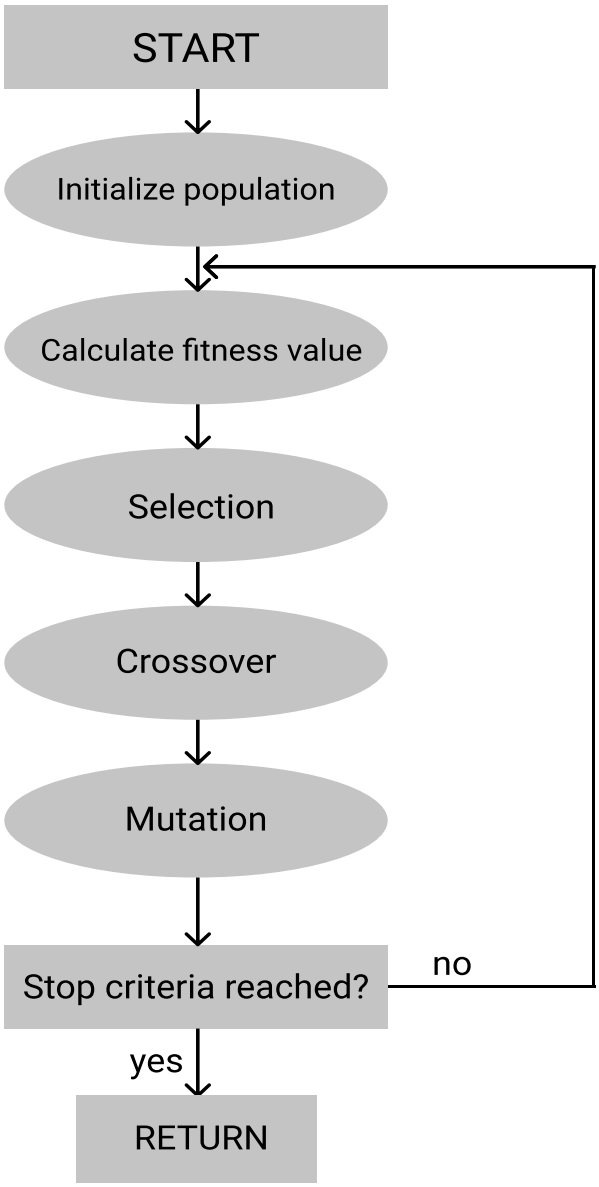
djeca. Dakle, dijete nasljeđuje genetske informacije od oba roditelja. Nakon toga se odvija mutacija djece. To je postupak tijekom kojeg se svojstva djece na određeni način mijenjaju. Mutacija je krucijalan korak genetskog algoritma jer osigurava da algoritam ne ovisi samo o prvim (nasumično) generiranim jedinkama.
Navedeni postupak ponavlja se sve dok se ne postigne optimalno rješenje ili se ne dosegne posljednja iteracija. Broj generacija i roditelja te vrsta križanja koja se koristi hiperparametri su koji su obično implementirani na način da se mogu modificirati. Njihove promjene drastično mijenjaju način rada genetskog algoritma. Određene kombinacije tih postavki više odgovaraju određenim problemima, dok neke kombinacije sporije dolaze do rješenja.
GA je uobičajeno najbolji pristup rješavanju problema koji zahtijevaju neku vrstu optimizacije ili pretrage. Često se koristi i pri rješavanju problema usmjeravanja vozila (*engl.* Vehicle Routing Problem). Okviran način rada genetskog algoritma prikazan je slikom 2.1. Iako je GA prikladan za pronalaženje dobrog rješenja nad velikim skupom, izrazito je važno biti dobro upoznat s prednostima i manama različitih implementacija selekcije, križanja i mutacije kako bismo pokušali izbjeći stagnaciju između generacija, odnosno, zapinjanje na najboljoj lokalnoj vrijednosti. Stoga ćemo u nastavku objasniti ulogu tih funkcija, a zatim i usporediti različite vrste ovih važnih dijelova genetskog algoritma.

2.3. Usporedba različitih metoda selekcije

Selekcija je jedan od temeljnih dijelova genetskog algoritma. O njenoj implementaciji ovisi koje jedinke ćemo izabrati kao roditelje iz kojih će križanjem nastati nova jedinka. Jedna od prvih solucija u povijesti genetskog algoritma bila je selekcija bazirana na dobroti jedinke (*engl.* Roulette Wheel Selection). Ukratko, ideja se bazira na tome da je šansa odabir određene jedinke proporcionalna njenoj dobroti (*engl.* fitness value). Ovakav pristup jednostavan je za implementaciju, ali može dovesti do stagnacije u slučajevima kada dobrote puno variraju. Prikaz vjerojatnosti odabira određene jedinke prikazan je sljedećom funkcjiom [1]:

$$probSel(i) = \frac{fit(i)}{\sum_{j=1}^{n} fit(j)}$$

Drugo rješenje bio bi odabir roditelja ovisno o rangu u populaciji koji se sortira na temelju dobrote. Dakle, nasumično se odabiru roditelji iz, recimo, prve trećine sortirane



Slika 2.1: Prikaz rada genetskog algoritma

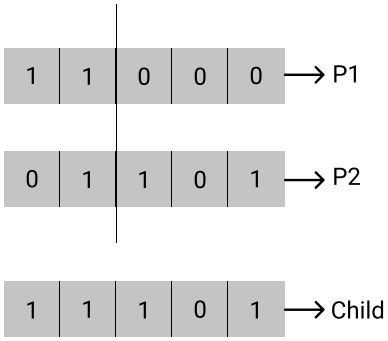
populacije.
U kontekstu ovoga rada, korištena je kombinacija ova dva pristupa. Dakle, implementirana je selekcija na temelju dobroti jedinke, ali u boljoj polovici populacije. Ovaj pristup pokazao je bržu konvergenciju, ali također u nekim slučajevima vodi do zapinjanja u lokalno najboljoj vrijednosti.

U ovom radu je također implementirana selekcija elitizmom u kojoj se ne odabiru roditelji za križanje, već se najbolje jedinke trenutačne populacije prenose bez promjene genetskih informacija u sljedeću generaciju. Elitizam je važan dio genetskog algoritma jer se na taj način ne zaboravljaju najbolje generirane jedinke. Bez elitizma, algoritam

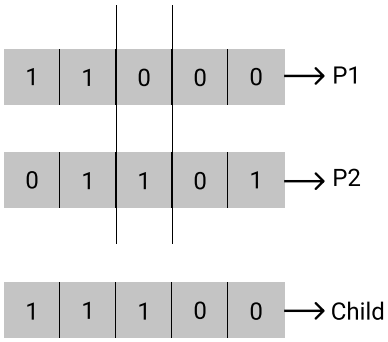
bi bio slabije usmjeren prema optimalnom rješenju.

2.4. Usporedba različitih metoda križanja

Križanje je proces generiranja nove jedinke na temelju karakteristika roditelja. Naravno, postoje različite implementacije križanja koje ćemo i opisati.
Jednostavna varijanta je takozvani 'one-point crossover' prikazan slikom 2.2. Na početku se nasumično bira cijeli broj koji će biti točka prekida. Lijevo od te točke dijete dobiva kromosome prvog roditelja, a desno od te točke dijete dobiva kromosome drugog roditelja. Druga varijanta ove metode bila bi 'two-point crossover' koji funkcionira na analogan način. Lijevo od prve točke dijete nasljeđuje kromosome prvog roditelja, između dvije točke nasljeđuje kromosome drugog roditelja i desno od druge točke dobiva kromosome prvog roditelja. Naravno, ovim metodama moguće je generirati dva dijeteta na način da se obrne redoslijed odabira roditelja.
Križanje s jednolikom vjerojatnošću implementirano je na način da dijete za svaki kromosom ima jednaku vjerojatnost da ga naslijedi od bilo kojeg roditelja. Moguće je podesiti da dijete naslijedi više informacija od pojedinog (boljeg) roditelja što može rezultirati kvalitetnijom jedinkom.
Križanje se pokazalo kao slaba točka algoritma za pronalazak optimalnog rasporeda jer su sva predložena križanja često izazivala stagnaciju između generacija.



Slika 2.2: 'One-point crossover'



Slika 2.3: 'Two-point' crossover

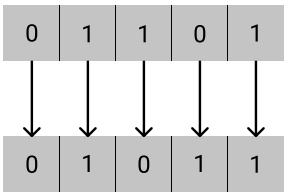
2.5. Usporedba različitih metoda mutacije

Mutacija je krucijalan dio genetskog algoritma jer nudi diverziju između generacija. Ovisno o vrsti korištenih varijabli kojima se predstavlja genetska informacija jedinke, moguće je koristiti različite tipove mutacija.

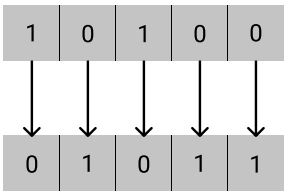
Često korištena mutacija za realne brojeve je Gaussov šum koja mutira svaki kromosom sa zadanom vjerojatnošću 'p' na način da mu se pribroji vrijednost uzorkovana iz normalne razdiobe sa zadanom standardom devijacijom 'K'.

Za binarne vrijednosti prikladna je 'Flip bit' mutacija koja okreće svaki kromosom jedinke iz nule u jedinicu i obrnuto. Ta mutacija dobra je u početnom pretraživanju kada ponuđena rješenja još nisu blizu optimalnom, ali u kasnijim generacijama previše okreće jedinke te se u kontekstu pronalaženja optimalnog rasporeda nije pokazala kao dobar odabir.

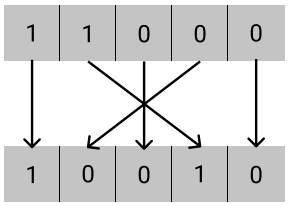
Posljednji tip mutacije koji je testiran u kontekstu ovoga rada je '2-Op' mutacija koja zamijenjuje dva kromosoma jedinke na način koji bi dao bolju jedinku [6]. Dakle, promjena se odvija samo ako mutirana jedinka pokazuje bolji rezultat od originalne. Ovaj pristup pokazao se najboljim od navedenih jer nudi usmjerenu mutaciju prema boljem rješenju. U kombinaciji s malo većom populacijom, ovaj tip mutacije je dobro funkcionirao.



Slika 2.4: Mutacija kromosoma s unaprijed zadanom vjerojatnošću



Slika 2.5: Mutacija obrtanjem svakog kromosoma (bita)



Slika 2.6: Mutacija zamjenom kromosoma

3. Implementacija rješenja

Programsko rješenje navedenog problema implementirano je u programskom jeziku 'Python'. Za potrebe određenih matematičkih operacija korištena je biblioteka 'numpy'. U nastavku ćemo opisati način na koji se manipulira podacima i komentirati implementaciju rješenja.

3.1. Formalan opis problema

Kako bismo preciznije objasnili jaka i slaba ograničenja, kao i cilj rješavanja problema, raspoređivanje medicinskih sestara prikazat ćemo na formalan način.
Koristit ćemo sljedeću notaciju:

- N –> skup medicinskih sestara
- h –> broj dana (*engl.* horizon)
- D –> skup dana = 0..h
- S –> skup smjena
- R_s –> skup smjena koje ne smiju biti zadane odmah nakon tipa smjene 's'
- N_n –> skup dana tijekom kojih sestra 'n' ne smije raditi
- l_s –> duljina smjene 's' u minutama
- m_{ns}^{max} –> maksimalan broj smjena tipa 's' koje se mogu dodijeliti sestri 'n'
- b_n^{max} –> maksimalan broj minuta koje sestri 'n' moraju biti dodijeljene
- b_n^{min} –> minimalan broj minuta koje sestri 'n' moraju biti dodijeljene
- c_n^{max} –> maksimalan broj uzastopnih smjena za sestru 'n'
- c_n^{min} –> minimalan broj uzastopnih smjena za sestru 'n'
- o_n^{min} –> minimalan broj uzastopnih slobodnih dana prije povratka na posao
- a_n^{max} –> maksimalan broj vikenda tijekom kojih sestra 'n' smije raditi

- q_{nds} –> cijena ako sestra 'n' na dan 'd' ne radi smjenu 's'
- p_{nds} –> cijena ako sestra 'n' na dan 'd' radi smjenu 's'
- u_{ds} –> optimalan broj zaduženih sestara na dan 'd' u smjeni 's'
- ${v_{ds}}^{min}$ –> težina kazne (cijene) ako je broj zaduženih sestara za smjenu 's' na dan 'd' manji od optimalnog broja
- ${v_{ds}}^{max}$ –> težina kazne (cijene) ako je broj zaduženih sestara za smjenu 's' na dan 'd' veći od optimalnog broja

U nastavku ćemo detaljnije promotriti važne varijable koje će pomoći u formalnijoj definiciji ograničenja:

- w_{nds} –> binarna varijabla, radi li sestra 'n' na dan 'd' u smjeni 's'
- y_{ds} –> ukupan zbroj za kazne 'ispod' optimalnog broja zaduženosti za dan 'd' i smjenu 's'
- z_{ds} –> ukupan zbroj za kazne 'iznad' optimalnog broja zaduženosti za dan 'd' i smjenu 's'

Sada možemo detaljnije pogledati jaka ograničenja koja se moraju poštovati. Za početak, svaka medicinska sestra smije raditi maksimalno jednu smjenu dnevno:

$$\sum_{s} w_{nds} \leq 1, \forall\ d\ \epsilon\ D, n\ \epsilon\ N$$

Također, potrebno je imati na umu da neke smjene ne smiju slijediti nakon određenih smjeni od dana prije, to je takozvani 'shift-rotation constraint'. Skup smjena koje ne smiju slijediti nakon smjene 's' nalaze se u R_s:

$$w_{nds} + w_{n(d+1)s'} \leq 1, \forall\ n\ \epsilon\ N,\ d\ \epsilon\{0, .., h-1\}, s'\ \epsilon\ R_s$$

Svaka medicinska sestra ima određenu konstantu za svaku pojedinu smjenu koja označava maksimalan broj zaduženja koje ona može odraditi. Zbog toga definiramo sljedeće ograničenje:

$$\sum_{d} w_{nds} \leq m_{ns}^{max}$$

Osim toga, medicinske sestre imaju ograničenja o maksimalnom i minimalnom broju ukupnih odrađenih minuta:

$$b_n^{min} \leq \sum_{d,s} w_{nds} \times l_s \leq b_n^{max},\ \forall n\ \epsilon\ N$$

Sestre također moraju biti raspoređene na način da se poštuje ograničenje o maksimalnim i minimalnim uzastopnim danima prije godišnjeg odmora:

$$\sum_s (w_{nds} + w_{n(d+1)s} + ... + w_{n(d+max\_cons\_shifts-1)s}) \leq c_n^{max},$$
$$\forall d = 0, ..., horizon - max\_cons\_shifts + 1,\ n\ \epsilon\ N$$
$$\sum_s (w_{nds} + w_{n(d+1)s} + ... + w_{n(d+min\_cons\_shifts-1)s}) \geq c_n^{min},$$
$$\forall d = 0, ..., horizon - min\_cons\_shifts + 1,\ n\ \epsilon\ N$$

Prije vraćanja sestre na posao, sustav joj mora omogućiti minimalan uzastopan broj dana odmora:

$$\sum_s (w_{nds} + w_{n(d+1)s} + ... + w_{n(d+min\_cons\_days\_off-1)s}) = 0,$$
$$\forall\ d = 0, ..., horizon - min\_cons\_days\_off + 1$$

Potrebno je paziti da sestre nisu opterećene određeni broj uzastopnih vikenda. Stoga postoji ograničenje 'max_num_of_weekends'. Budući da po dogovoru indeks 0 označava ponedjeljak, prvu subotu označavamo s indeksom '5', a prvu nedjelju s indeksom '6'. U sljedećem zapisu, oznaka '$\cup$' označava operaciju unije:

$$\sum_s (w_{n(7\times d+5)s} \cup w_{n(7\times d+6)s}) + \sum_s (w_{n(7\times d+5)s} \cup w_{n(7\times d+5)s}) + ... + \sum_s (w_{n(7\times d+5)s} \cup w_{n(7\times d+5)s})$$
$$\leq max\_num\_of\_weekends,\ \forall\ d = 0, ..., (horizon/7) - 1$$

Podaci o tome koja sestra i kada želi godišnji odmor spremljeni su u skup N_n. To je jako ograničenje koje je također potrebno poštovati:

$$\sum_{s,n} w_{nds} = 0,\ \forall\ d\ \epsilon\ N_n$$

Na kraju, možemo definirati funkciju cilja:

$$min(\sum_{n,d,s} q_{nds}(1 - w_{nds}) + \sum_{n,d,s} p_{nds}w_{nds} + \sum_{d,s} y_{ds}v_{ds}^{min} + \sum_{d,s} z_{ds}v_{ds}^{max})$$

Ova definicija pokazuje da je naš cilj osigurati optimalan broj zaduženosti u svakoj smjeni svakoga dana, tj. minimizirati nedovoljno osoblja, kao i previše osoblja u smjeni. Minimalna vrijednost se postiže ako je osiguran optimum zaduženja za svaku pojedinu smjenu. Uz to, niti jedna sestra nije dežurna na smjeni u kojoj ne želi raditi i sustav ju postavi dežurnom na smjeni u kojoj želi raditi. Naravno, ovaj scenarij neće uvijek biti moguć, stoga postoje cijene za svako nepoštovanje ograničenja. Za svaki put kada uvjet nije poštovan, nadodaje se zadana cijena na ukupnu težinu rješenja.

3.2. Format podataka

Različite instance nad kojima ćemo tražiti optimalno rješenje nalaze se na stranici schedulingbenchmarks.org. Učitavanjem podataka saznajemo sljedeće detalje skupa nad kojem tražimo rješenje:

- broj dana nad kojim se razmatra problem
- skup medicinskih sestara
- skup smjena
- jaka ograničenja vezana uz pojedine medicinske sestre
    - maksimalan broj određenih vrsta smjena
    - maksimalan i minimalan broj minuta rada
    - maksimalan i minimalan broj uzastopnih smjena
    - minimalan broj uzastopnih slobodnih dana
    - maksimalan broj radnih vikenda
    - tražene slobodne dane
- slaba ograničenja
    - smjene nad kojima sestre (ne) žele raditi
    - bolničke zahtjeve za pojedine dane i smjene - optimalna kvota po smjeni

Imajući na umu učitane podatke, generirane su strukture podataka napravljene identično kao u odlomku '2.2 Formalan opis problema'. One omogućuju brzo pretraživanje te laku provjeru ograničenja.

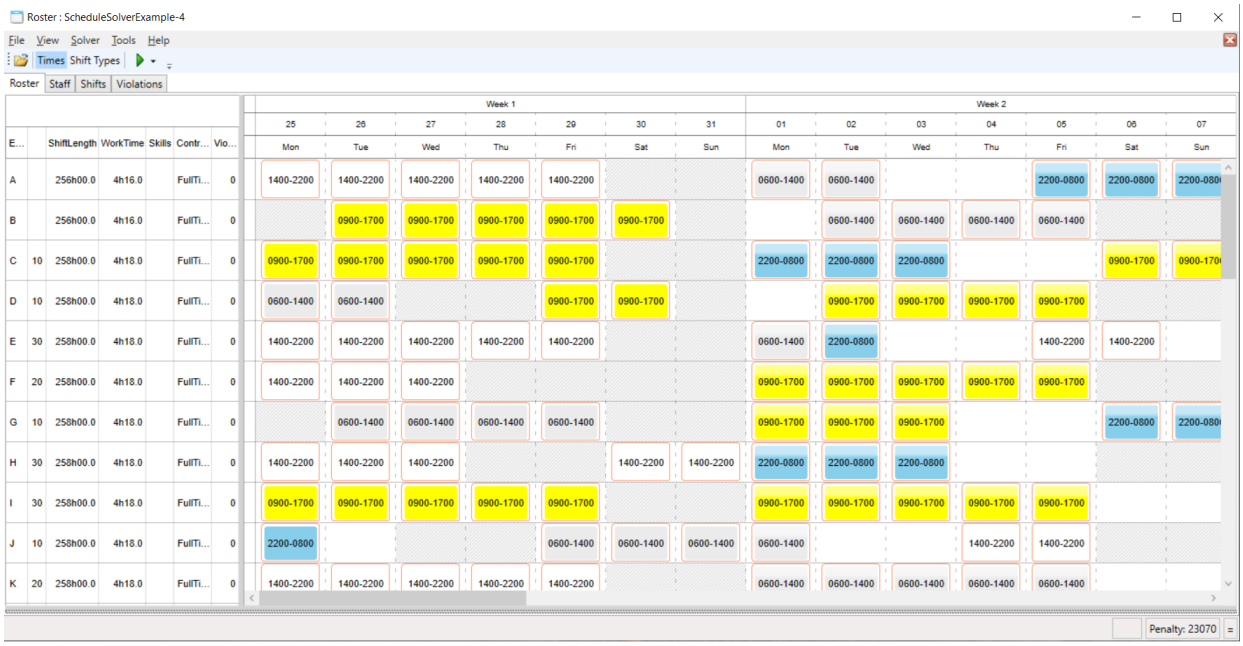
Kako bi sustav mogao manipulirati selekcijom, križanjem i mutacijom, pojedina jedinka je implementirana kao klasa koja, između ostalog, sadrži varijablu koja označava zaduženost pojedine sestre za pojedinu smjenu u određenom danu. Ta varijabla je implementirana kao rječnik čiji ključ je dvojka 'indeks_dana , indeks_sestre', a vrijednost je 'indeks_smjene'. Ako sestra nije zadužena za taj dan, tada je vrijednost 'None'. Na ovaj način, samom strukturom podataka, sustav se pridržava ograničenja od maksimalno jedne smjene dnevno za pojedinu sestru. Ova varijable koristi se za analizu dobrote jedinke. Točnije, pri analizi slabih ograničenja, ako se ustanovi da sestra radi smjenu koju ne preferira ili da ne radi smjenu koju preferira, ukupnom rješenju se dodaje određena cijena. Isti princip koristi se i pri provjeri optimalnih zaduženosti koje zahtjeva bolnica. Na taj način dobiva se ukupna cijena rješenja na temelju koje sustav procjenjuje dobrotu jedinke.

Osim toga, svaka jedinka sadrži 'boolean' varijablu koja označava je li to konkretno rješenje zadovoljavajuće ili ne. Ako algoritam generira rješenje koje je zadovoljavajuće, odnosno ne krši stroga ograničenja, sustav u izlaznu datoteku ispisuje rješenje u formatu prikladnom za prikaz u programu 'RosterViewer'.

4. Rezultati

Za prikaz i evaluaciju rješenja korišten je alat RosterViewer. Primjer prikaza validnog rješenja za jednu instancu nalazi se na slici 4.1. Opišimo ukratko sadržaj te tablice. Svaki redak u tablici označava jednu medicinsku sestru. Svaki stupac odvojen iscrtkanom linijom označava jedan radni dan, dok puna linija odvaja tjedne. Unutar svake ćelije nalazi se smjena za koju je ta sestra zadužena taj dan. Ako sestra taj dan ne radi, ćelija je prazna. Svaki tip smjene obojan je drugačijom bojom kako bi raspored bio pregledniji. Svaka smjena u ovom primjeru traje osam sati. Sivo obojana ćelija označava ranojutarnju smjenu s početkom u šest sati, a žuta ćelija prikazuje jutarnju smjenu koja započinje u devet sati. Bijele ćelije prikazuju dane kada sestre moraju raditi popodnevnu smjenu koja traje od 14:00 do 22:00 sata. Noćna smjena označena je plavom bojom. Primijetimo da ne postoji slijed od dva dana u kojem je sestra zadužena za noćnu pa jednu od jutarnjih smjena čime se poštuje strogo ograničenje o rotaciji smjena. Unutar programa moguće je provjeriti zadovoljava li rješenje stroga ograničenja te koliko je rješenje dobro. U donjem desnom kutu na slici ne prikazuje se posebna obavijest što govori da je ovo rješenje izvedivo. Uz svako rješenje veže se određena cijena koja opisuje koliko je ono dobro. Na primjeru ovog rješenja vidljivo je da je cijena 23,070 što je suma svih kazni zbog nepoštovanja slabih ograničenja.

Nad ponuđenom implementacijom napravljeni su testovi s ciljem usporedbe različitih metoda križanja i mutacije. Analizirajmo dobivene rezultate napravljene nad populacijama veličine deset jedinki. Broj elitista zadan je na dva, a algoritam je generirao sto tisuća generacija za svaki test. Napomenimo da je vjerojatnost 'p' prilikom korištenja uniformne mutacije jednaka 0.3 što označava da je otprilike 30% kromosoma podložno promjeni.
Korištene metode križanja su uniformno križanje i križanje s dvije točke prekida. Testirane metode mutacije jedinki su mutacija zamjenom dvaju kromosoma i uniformna mutacija. Napravljena su četiri testa koja uspoređuju performanse algoritma koristeći kombinacije navedenih metoda. Testovi su napravljeni nad sljedećim instancama i ana-



Slika 4.1: Primjer prikaza validnog rješenja unutar programa RosterViewer

lazirani u spomenutom alatu. Prva instanca sastoji se od 14 dana, osam medicinskih sestara i jednim tipom smjene, 'D' (09:00-17:00 sati). Druga instanca također ima 14 dana, ali ima i 14 medicinskih sestara i dvije različite smjene; 'L'(06:00-14:00 sati) i 'E' (14:00-22:00 sata).

Na slici 4.2 nalaze se rezultati dobiveni korištenjem mutacije zamjenom dvaju kromosoma. S obzirom na to da je najbolje rješenje za prvu instancu 607, algoritam za nju postiže dobre rezultate. Korištenjem uniformnog križanja i križanja s dvije točke prijeloma, najbolji rezultati su 816, odnosno, 919. Algoritam nudi rješenje s prosječnom cijenom 926 za uniformno križanje i 1,076 za križanje s dvije točke prijeloma. Za drugu instancu boljim se pokazalo križanje s dvije točke prijeloma koje u prosjeku nudi raspored s cijenom 1,705, dok druga kombinacija daje rezultate s prosječnom cijenom 1,814.

Na slici 4.3 nalaze se rezultati dobiveni metodom uniformne mutacije u kombinaciji s dvije vrste križanja. Uspoređujući ove podatke s prethodnima, očito je da su ovi rezultati manje obećavajući. U ovom slučaju prosječna cijena rasporeda za prvu instancu je 1,376, odnosno 1,516. Zanimljivi su rezultati za drugu instancu gdje algoritam

crossTwoPoint + mutationSwap				crossUniform + mutationSwap		
Pokretanje	Instance1	Instance2		Pokretanje	Instance1	Instance2
1	1011	1135		1	923	1655
2	924	1937		2	914	2548
3	1224	2244		3	921	1545
4	1117	2032		4	1018	1939
5	919	1832		5	918	1937
6	1109	2132		6	1015	1447
7	1015	1936		7	1012	1946
8	1113	1344		8	907	1263
9	1308	1034		9	819	2337
10	1020	1428		10	816	1529
Average	1076	1705.4		Average	926.3	1814.6
Minimum	919	1034		Minimum	816	1263
StdDev	117.7718133	410.5219117		StdDev	69.23301236	384.3977627

Slika 4.2: Rezultati dobiveni korištenjem mutacije zamjenom kromosoma

u prosjeku nudi duplo skuplje rješenje od metoda prikazanih tablicom 4.2.

crossTwoPoint + mutationUniform				crossUniform + mutationUniform		
Pokretanje	Instance1	Instance2		Pokretanje	Instance1	Instance2
1	1015	3770		1	1552	3784
2	1321	3455		2	1312	3577
3	1226	2769		3	1422	3864
4	1329	3679		4	1325	3072
5	1425	3073		5	1527	2966
6	1531	3065		6	1728	3264
7	1533	3080		7	1628	3380
8	1525	3873		8	1619	3772
9	1323	2975		9	1427	3881
10	1532	3073		10	1626	3067
Average	1376	3281.2		Average	1516.6	3462.7
Minimum	1015	2769		Minimum	1312	2966
StdDev	160.7718881	361.6116149		StdDev	132.8399036	339.1516033

Slika 4.3: Rezultati dobiveni korištenjem uniformne mutacije

Glavni problem ponuđene implementacije je loše usmjerenje prema optimalnom rješenju. Algoritam za niti jednu instancu ne nudi rješenje koje zaobilazi sva stroga ograničenja. Točnije, algoritam će uvijek ponuditi rješenje koje zadaje maksimalno jednu smjenu dnevno, ne prelazi zadani broj radnih vikenda te ne zadaje smjene kada sestre imaju godišnji odmor. Poštovanje ostalih ograničenja nije konzistentno. Algoritam je često nakon određenog broja generacija stagnirao te prestao nuditi bolja rješenja. Navedeni problem prikazan je usporedbom tablica na slici 4.4. Iako je sustav u lijevom slučaju imao duplo više generacija za pronaći optimalno rješenje, on je stagnirao te u

prosjeku nudio slična rješenja kao i kada je generirao samo pedeset tisuća generacija.

crossUniform + mutationSwap, 100 000 generations		
Pokretanje	Instance1	Instance2
1	923	1655
2	914	2548
3	921	1545
4	1018	1939
5	918	1937
6	1015	1447
7	1012	1946
8	907	1263
9	819	2337
10	816	1529
Average	926.3	1814.6
Minimum	816	1263
StdDev	69.23301236	384.3977627

crossUniform + mutationSwap, 50 000 generations		
Pokretanje	Instance1	Instance2
1	814	1535
2	1119	2753
3	822	1348
4	1118	1754
5	1116	1756
6	1020	1734
7	1218	1421
8	1010	1747
9	1013	1541
10	916	2955
Average	1016.6	1854.4
Minimum	814	1348
StdDev	126.6153229	520.4909605

Slika 4.4: Usporedba ponuđenih rješenja za različit broj generacija

Ipak, kroz razna testiranja bilo je očito da pri rješavanju ovog problema najbolje rezultate pokazuje mutacija zamjenom dvaju kromosoma kao i uniformno križanje. Ta kombinacija bila je najpovoljnija jer je manje 'zapinjala' te nudila brži dolazak do boljih rješenja. Uniformna mutacija pokazala se kao lošija alternativa.

Dakle, zaključujemo da jednostavni genetski algoritam nije dovoljno dobar za rješavanje ovog problema. Korištene operacije križanja nisu dovoljno kompleksne te bi bilo bolje koristiti naprednije metode križanja i mutacije. Konkretno, mogle bi se koristiti sljedeće metode: Order Crossover (OX), Partially Mapped Crossover (PMX), Edge Recombination Crossover (ERX), Cycle Crossover (CX) i druge metode križanja [8]. Opišimo neke od njih.

Order Crossover kompleksnija je izvedba Two-point Crossovera pri kojoj se sadržaj između dvije točke kopira od prvog roditelja, dok se ostatak kopira na sljedeći način; počevši od desne točke kopiraju se kromosomi od drugog roditelja na način da se već prekopirani kromosomi ne uzimaju. Taj postupak odvija se ciklički dok se ne napuni cijeli gen dijeteta.
Partially Mapped Crossover slične je izvedbe. U ovom slučaju također postoje dvije nasumično odabrane točke te se sadržaj između te dvije točke kopira od prvog roditelja. Nakon toga se kopiraju kromosomi drugoga roditelja na način da se maksimalno

pokušava zadržati njihova početna pozicija.

Poznata solucija je i Cycle crossover čija ideja je da dijete naslijedi kromosom od jednog roditelja, ali njegovu poziciju od drugog roditelja [8]. Postupak započinje odabirom prvog kromosoma prvog rodtelja i kopiranjem dijetetu. Nakon toga bira se prvi kromosom drugog roditelja i kopira se dijetetu na mjesto gdje se taj kromosom nalazi kod prvog roditelja. Zatim se bira kromosom roditelja dva na poziciji označenoj prethodno izabranim kromosomom i on se opet kopira dijetetu na mjesto gdje se taj kromosom nalazi kod suprotnog roditelja. Postupak se nastavlja dok se u potpunosti ne dopune genetske informacije dijeteta.

5. Zaključak

Ovaj rad istražio je problem raspoređivanja medicinskih sestara. Na početku rada opisan je pojam optimizacijskog problema te različite metode kojima se rješava taj problem. Ustanovljeno je da je OP često prisutan u velikim organizacijama te da je stoga bitno formalno ga opisati i ponuditi prikladno rješenje. Problem je stoga predstavljen na formalan način tako da su opisana sva njegova jaka i slaba ograničenja. Isto tako, definirana je ciljna funkcija čiji minimum svako rješenje nastoji zadovoljiti. Istražene su metaheurističke metode koje su prikladne za njegovo rješavanje. Izabran je genetski algoritam kao prikladan način rješavanja navedenog problema. Testirane su i uspoređivane razne varijante selekcije, križanja i mutacije.
Pokazalo se da je odabir tih funkcija krucijalan za razvoj rješenja. Uočen je problem sa stagnacijom pri korištenju svih metoda križanja što se očituje u tome da rješenje često zapinje u lokalnom minimumu. S time se može zaključiti da je optimizaciju rasporeda medicinskih sestara teško riješiti jednostavnim varijantama genetskog algoritma. Za bolje rješenje navedenog problema, prikladnije bi bile navedene naprednije metode križanja kao što su 'Partially Mapped Crossover', 'Edge Recombination Crossover' ili 'Cycle Crossover'.
Što se tiče operatora mutacije, kao najbolja solucija pokazali su se operator međusobne zamjene pozicija kromosoma. FlipBit mutacija dobra je u početnim generacijama kada rješenja puno variraju, ali u kasnijim generacijama ne usmjerava algoritam prema optimalnom rješenju. Uniformna mutacija davala je znatno slabija rješenja od mutacije zamjenom pozicija kromosoma.

Zaključujemo da je veliki izazov pronaći optimalan algoritam za rješavanje promatranog problema. Različite kombinacije operacija selekcije, križanja i mutacije, kao i postavke hiperparametra uvelike utječu na uspjeh pronalaska optimalnog rješenja. Usprkos tome, predloženi algoritam, uz manje preinake vezane uz ograničenja, moguće je koristiti u različitim okruženjima raspoređivanja.

LITERATURA

[1] Čupić M, *Prirodom inspirirani optimizacijski algoritmi*, Zagreb, 2009.

[2] Adoly A., Gheith M., Nashat Fors M. *A new formulation and solution for the nurse scheduling problem: A case study in Egypt* (2018.)

[3] Aickelin U., Dowsland K.A. *An indirect Genetic Algorithm for a nurse scheduling problem*

[4] Augustine L., Faer M., Kavountzis A., Patel R. *A Brief Study of the Nurse Scheduling Problem (NSP)* (2009.)

[5] Curtois T., Qu R. *Computational results on new staff scheduling benchmark instances*, (2014.)

[6] Hassan N., Hmeidi I., Ammari H. *Solving the Constraint Satisfaction Problem Using Genetic Algorithms*

[7] *Optimization Problems*; Poveznica: www.sfu.ca; pristupljeno: 09. lipnja 2021.

[8] Puljić K., Manger R., *Comparison of eight evolutionary crossover operator for the vehicle routing problem*, (2013.)

[9] Instance nad kojima je implementirano rješenje: Poveznica:schedulingbenchmarks.org

Optimizacija rasporeda smjena medicinskih sestara

Sažetak

Ovaj rad bavi se problemom raspoređivanja medicinskih sestara. Na početku je definiran optimizacijski problem i njegov cilj. Rad istražuje metaheurističke metode koje bi mogle ponuditi dobro rješenje za ovaj problem. Kroz testiranja nad instancama moglo se zaključiti da je odabir kombinacije metoda križanja i mutacije krucijalan za brži pronalazak i usmjeravanje prema boljem rješenju. Uočen je problem stagnacije koji izazivaju jednostavne metode križanja i selekcije. Ponuđeno je bolje rješenje križanja i mutacije jedinki.

Ključne riječi: optimizacija, problem raspoređivanja medicinskih sestara, genetski algoritam, metaheuristika

Optimisation of nurse shift schedules

Abstract

This paper is based on the nurse scheduling problem. At the beginning, the optimization problem and its goal are defined. Paper then explores metaheuristic methods that could offer a good enough solution for this problem. Through testing on given instances it was concluded that the combination of crossover and mutation methods is crucial for faster route towards an optimal solution. Stagnation problem caused by simple selection and crossing methods was noticed. A more suitable solution was proposed for crossover and mutation of individuals.

Keywords: optimization, nurse scheduling problem, genetic algorithm, metaheuristics