Univerzitet u Beogradu

Matematički fakultet

Mihailo Simić

Problem rasporeda časova u osnovnim i srednjim školama

Diplomski – master rad

B e o g r a d

2 0 1 2.

Mentor: Prof. Đorđe Dugošija, **Matematički fakultet u Beogradu**

Članovi komisije:

Prof. Milan Dražić, Matematički fakultet u Beogradu

Prof. Zorica Stanimirović, Matematički fakultet u Beogradu

Datum odbrane: _____

Contents

1. Istorijski pregled problema rasporeda časova

Problem rasporeda ili raspoređivanja je jedan od najstarijih i najviše ispitivanih problema. Može se pojaviti u različitim oblicima – raspored časova u osnovnim i srednjim školama, na fakultetu (raspored ispita je još jedan vid problema rasporeda časova), zatim raspored za zaposlene na poslu ukoliko se radi po smenama, itd.

U prošlosti se raspored pravio ručno, to jest uzimali su se svi potrebni podaci i nastojalo se da se svi zahtevi i ograničenja uklope u odgovarajuću celinu. U poslednjih 50 godina radi se na automatizovanju i rešavanju problema uz pomoć kompjuterskih programa. Na žalost, mnogi kompjuterski programi i dalje nisu usavršeni, tako da se u mnogim osnovnim i srednjim školama kod nas i dalje radi ručno, jer je to lakši i efikasniji način nego rad sa programom.

Prve tehnike u rešavanju ovog problema bile su zasnovane na simulaciji ljudskog pristupa u rešavanju problema. Sledeći korak je bio da se primene tehnike kao što su linearno i celobrojno programiranje, bojenje grafova, bipartitni problem sparivanja, itd. Prvi radovi vezani za pravljenje rasporeda časova napisali su Kuhn (1955) i Haynes (1959). Njihovi pristupi se razlikuju u tome što je Kuhn prihvatao matematički pristup problemu za razliku od Haynesa koji se koncentriše na praktičnije probleme raspoređivanja.

Interesovanje za ovaj problem posebno je bilo izraženo tokom šezdesetih godina prošlog veka zahvaljujući bržem razvoju kompjutera koji su mogli da izvode više operacija potrebnih za algoritam. U ovim radovima se prvi put koriste *look-ahead* tehnike. Na primer, Appleby, Blake i Newman[1] i Lewis (1961) koriste primitivne *look-ahead* tehnike uključujući heuristike za brojanje argumenata. Prvi pristup koji nije uključivao heuristike razvio je Gotlieb (1963). To je bio verovatno prvi pristup problemu na skupu delimičnih rešenja i posle toga su ga sledili Berghuis, Van der Heiden i Becker (1964) koji su primenjivali koncept virtuelnih klasa nastavnika kako bi uveli klasični bipartitni problem. Csima (1965) je proširio ovaj problem na stohastički problem koji koristi pseudo klase. Posle ovoga, a na osnovu njegovog rada, Csima i Gotlieb[2] bave se ovim problemom kao trodimenzionalnim problemom (assignment) raspoređivanja obraćajući pažnju na vezu između rasporeda časova i bojenja grafova. Nakon toga, a kao posledica Gotlieb-ovog rada (1963) , Becker (1964) i Baraclough (1965) simuliraju implementaciju sa *ručnim* izračunavanjima, koja je zasnovana na heuristikama. Krajem 60-ih bili su neki pokušaji da se ograniči generalni problem uzimajući u obzir primere nekih slučajeva, to jest na osnovu dosad poznatih radova i dokazanih činjenica da se olakša problem rasporeda časova. Lawrie (1969) razvija model koristeći celobrojno linearno programiranje.

Tokom sedamdesetih godina prošlog veka nekoliko autora se suočilo sa problemom koristeći heuristike. Junginger (1972) je redukovao problem rasporeda časova koristeći

[1] Vidi [13]
[2] Vidi [12]

trodimenzionalni transportni problem. Schmidt i Strohlein[3] objavljuju prvu bibliografiju o rasporedu časova sa preko 200 unosa. Predvideli su da će problem preći sa ručnog rešavanja na velikim kompjuterima u mikro kompjutere koje će svako imati. Tokom sedamdesetih i osamdesetih godina prošlog veka primenjivale su se Simulated Annealing(SA), Tabu Search(TS) i Genetic Algorithm(GA) metaheuristike, koje su zasnovane na veštačkoj inteligenciji.

I tokom osamdesetih godina se nastavljaju ispitivanja problema rasporeda časova tako da De Werra (1985) nabraja probleme i navodi različite metode kako je moguće rešiti ih. Junginger (1986) je opisivao pristupe zasnovane na direktnim heuristikama. Od drugih ispitvanja tokom osamdesetih godina prošlog veka možemo spomenuti i : Hilton (1980), Klein (1983), Vincke (1984), Feland, Roy i Gia-Loc(1986).

Iako je i u devedesetim bilo radova zasnovanih na prethodno nabrojanim tehnikama veštačke inteligencije (TS, SA, GA), pojavio se i novi pristup nazvan *Constraint Satisfaction Programming (CSP*) - gleda da se zadovolje sva ograničenja. Abramson (1991) je koristio SA kao optimizacionu tehniku. Dodao je komponentu - cost funkciju kako bi uključio sve kompleksnije zahteve koji su se pojavili u školama i pokazao je da ova funkcija dozvoljava nekim komponentama da budu važnije od drugih. Implementirao je ovu svoju zamisao u paralelni kompjuterski sistem i dokazao da se brzina algoritma popravlja zajedno sa rezultatima. Cooper i Kingston (1993) su napisali program koji rešava problem ne uzimajući u obzir veća ograničenja. Costa (1994) se bavio problemima ograničenja koja moraju biti uzeta u obzir. Tsang, Williams, Ford i Borrett[4] su raspravljali o važnosti ograničenja u rešavanju problema. Schaerf[5] je ispitivao različite tehnike u razvoju ovog problema, a bilo je i drugih naučnika koji su se bavili ovim problemom.

[3] Vidi [20]
[4] Vidi [16]
[5] Vidi [17]

2. Problem rasporeda časova

U svakoj školi (osnovnoj ili srednjoj) je potrebno napraviti raspored časova po kome se odvija nastava. Raspored časova je pravilo kojim se odredjuje koji nastavnik kojem razredu predaje koji predmet u koje vreme i u kojoj učionici. Iz njega se mogu posebno izvesti raspored rada nastavnika, raspored za učenike, raspored zauzetosti sala i dr. Raspored časova za nastavnike je tabela na kojoj se po vertikali nalaze imena nastavnika, a po horizontali dani i časovi u tom danu, tako da je u polju u rasporedu kad nastavnik ima čas, upisano odeljenje kome taj nastavnik predaje. Za učenike to izgleda drugačije, tako da oni dobiju tabelu sa predmetima i nastavnicima koje im taj predmet predaju, kao i učionicu u kojoj će se predmet održati ako se ne održava u istoj učionici u kojoj se određeno odeljenje nalazi. Pored ova dva, postoji i treći - raspored učionica na koji takođe treba obratiti pažnju i na kome se nalaze vremenski intervali kada je određena učionica slobodna, a kada je zauzeta. Pri pravljenju rasporeda pretpostavljamo da:

1. Imamo spisak svih nastavnika iz svih predmeta i spisak časova koje svako od njih predaje;
2. Imamo spisak odeljenja kojima oni predaju određen broj časova;
3. Imamo spisak učionica u kojima se obavlja nastava;

Dakle, uprava škole je odredila koliko nastavnika je potrebno za tu školsku godinu i koliko ih ima po određenom predmetu i oni su rasporedjeni po godinama i odeljenjima,a zadatak osobe koja je zadužena da napravi raspored časova je da sve to uklopi u jednu celinu .

2.1. Matematički model problema rasporeda časova:

Ako je unapred dat četvoro-elementni skup <N, R, U, V> gde je :

- N je skup nastavnika $N=\{n_1, n_2, \ldots, n_\alpha\}$,

- R je skup razreda $R=\{r_1, r_2, \ldots, r_\beta\}$,

- U je skup učionica u kome se održavaju časovi $U=\{u_1, u_2, \ldots, u_\gamma\}$,

- V je skup vremenskih perioda $V=\{v_1, v_2, \ldots, v_\delta\}$.

i ako sa x_{ijkl} označimo broj iz skupa {0,1} za koji važi:

$$x_{ijkl}=\begin{cases} 1, \text{ ako nastavnik i drži čas razredu j u učionici k u vremenu l,} \\ 0, \text{ inače} \end{cases}$$

dobijamo problem rasporeda časova preveden na matematički jezik.

2.1.1. Ograničenja

Sada uvodimo ograničenja koja moraju ili je poželjno da budu zadovoljena kako bi raspored časova bio što bolje napravljen. Prva vrsta ograničenja su teška (hard constraints) dok su druga lakša (soft constraints).
Prva ograničenja su :

1. Svi časovi određenog odeljenja moraju biti raspoređeni u različito vreme:
   $\forall$ j= 1,....., β $\forall$ l = 1,...., $\delta \sum_{i=1}^{\alpha} \sum_{k=0}^{\gamma} x_{ijkl} \leq 1$

2. Dva časa ne mogu biti u istoj učionici u isto vreme:

   $\forall$ k= 1,....., γ $\forall$ l = 1,...., $\delta \sum_{i=1}^{\alpha} \sum_{j=0}^{\beta} x_{ijkl} \leq 1$

3. Časovi istog nastavnika moraju biti u drugo vreme:

   $\forall$ i= 1,....., α $\forall$ l = 1,...., $\delta \sum_{j=1}^{\beta} \sum_{k=0}^{\gamma} x_{ijkl} \leq 1$

4. Blok nastava - ako su dva časa u bloku, moraju biti uzastopni u vremenskom rasporedu i moraju biti u istoj učionici. To jest, ako smo odredili da je neki čas određenog razreda koji drži nastavnik u učionici raspoređen u odgovarajući vremenski period, to jest x_{ijkl}= 1 tada je i x_{ijkl+1} = 1 ili x_{ijkl-1} = 1 u zavisnosti od toga kako nam odgovara i šta je slobodno.

5. Časovi za odeljenja ne smeju imati praznine i moraju počinjati od prvog časa u danu:

   Prvo proveravamo da li sva odeljenja imaju prvi čas u danu to jest da li je sledeći uslov ispunjen:

   $\forall r_1, r_2, \ldots, r_\beta \; \forall$ o = 1,.... , 5 (dani u nedelji) $x_{ijk(o\cdot\delta/5+1)} = 1$

   i onda ima prvi čas u danu, a ako nije onda gledamo da neki drugi čas stavimo na prvo mesto u danu;

   Ova formula važi za osnovnu školu, a za srednju je slična samo što umesto o·δ/5 +1 treba da stoji o·δ/14 +1 jer srednja škola ima 14 perioda u danu (7 prepodne i 7 popodne), a osnovna 5 (ne računajući 6-i čas koji je obično čas razrednog starešine koji se održava jednom nedeljno i može biti naknadno dodat);

   Sada razmatramo drugi uslov da časovi za sve razrede ne smeju da imaju praznine:

   $\forall r_1, r_2, \ldots, r_\beta \forall$ o = 1,.... , 5 (dani u nedelji)

$\sum_{q=1}^{\frac{\delta}{5}-1} xijk((o-1)*\frac{\delta}{5}+q)*xijk(o*\frac{\delta}{5}+q+1) \geq \delta/5\text{-}1;$

Za srednju školu važi slična formula to jest važi sledeće:

$\sum_{q=1}^{\frac{\delta}{5}-1} xijk((o-1)*\frac{\delta}{5}+q)*xijk(o*\frac{\delta}{5}+q+1) \geq \delta/10\text{-}2$ jer je ovde $\delta = 14$ a nama je važno da svako odeljenje ima 6 časova bilo da je prva ili druga smena i u nekom danu čak i 7 časova ali da budu uzastopni.

6. Svi časovi moraju biti raspoređeni:

   -ako imamo datu matricu $B_{\alpha x \beta}$ gde je b_{ij} broj časova koji nastavnik i drži razredu j onda na kraju raspoređivanja časova ova matrica mora biti zadovoljena, to jest broj časova koje nastavnik u toku nedelje drži nekom razredu mora se poklapati sa brojem iz ove matrice gde se promenljive koje označavaju nastavnika i razred ukrštaju.

U drugu grupu ograničenja spadaju :

1. raspoloživost nastavnika i neke njihove posebne želje, koje moraju biti uzete u obzir:

   Uvodimo matricu $A_{\alpha x \delta}$za koju važi da je:

   $a_{il} = \begin{cases} 1, \text{ ako nastavnik i može da predaje u vreme l} \\ 0 \text{ ako ne može} \end{cases}$

2. minimalan i maksimalan broj radnih časova nastavnika mora biti uzet u obzir - ne manje od 2 i ne više od 5 časova dnevno uz uslov da ima što manje rupa u rasporedu - ako može jedna ili nijedna:

   $\forall n_1, n_2, \ldots, n_\alpha \forall o = 1, \ldots, 5\ 2 \leq \sum_{q=1}^{\frac{\delta}{5}} xijk((o-1)*\frac{\delta}{5}+q) \leq 5$ i ta formula važi i za osnovnu i za srednju školu (za osnovnu školu δ je 5, a za srednju 14).

   Drugi deo uslova je nešto teže ispuniti jer ne znamo kada nastavnik ima prvi čas pa će nam to biti prvi uslov koji treba da odredimo, to jest:

   $\forall n_1, n_2, \ldots, n_\alpha \forall o = 1, \ldots, 5$ naći kada je prvi put u određenom danu za određenog nastavnika $x_{ijkl} = 1$, a zatim počevši od tog časa, vidimo koliko nastavnik ima ukupno časova u tom danu (mora da ima između 2 i 5), a onda postavljamo uslov da nema više od jedne pauze u danu:

   $\delta/5\text{-}3 \leq \sum_{q=1}^{\frac{\delta}{5}-1} xijk((o-1)*\frac{\delta}{5}+q)*xijk(o*\frac{\delta}{5}+q+1) \leq \delta/5\text{-}1$ ako je broj časova 5, a ako je broj časova i manji (2, 3, 4), onda ova suma može biti i 0 (recimo 2 časa i pauza između njih). Ova formula važi i za srednju i za osnovnu školu, a može se uraditi i na drugi dosta teži način

   Odredimo kada je prvi put u toku dana $x_{ijkl} = 1$ i odredimo kada je poslednji put 1 - idemo za svaki dan i za svakog nastavnika i kada je prvi put 1 stavimo brojač na 1 i svako put kad je x_{ijkl} 1 pamtimo za koje i, j, k, l je bilo 1 na kraju tog dana izvučemo

prvo i poslednje pojavljivanje 1, a onda gledamo koliko je između njih bilo 0 i pokušavamo da taj broj bude što manji - jednom ili nijednom.

3. Ako ima više časova iz istog predmeta , trebalo bi da bude u više dana – ako ima tri časa, to su ponedeljak, sreda i petak, dva časa – ponedeljak i sreda ili utorak i četvrtak najčešće, mada može biti i sreda i petak ili ponedeljak i petak po potrebi, samo da ne budu dva dana uzastopno, recimo utorak i sreda.
4. U odgovarajućim učionicama (kabinetima) da se drži nastava i zato ne postoji neki određeni uslov već se određeni čas raspoređuje u određenu učionicu za nju predviđenu;
5. Da nastavnik nema samo jedan čas u danu - to je sadržano u 2 - da ima između dva i pet časova dnevno;
6. Neke učionice služe i za druge časove – kada su kabineti slobodni mogu služiti i za druge časove ako je to neophodno.

Zadatak rasporeda časova je da zadovolji sva ograničenja iz prve grupe i da zadovolji što može veći broj ograničenja iz druge grupe.

2.1.2. Primer kako izgleda konačan raspored časova:

Dakle, raspored časova za profesore bi trebalo ovako da izgleda (uzećemo primer za jedan dan u nedelji i deset profesora zbog lakšeg prikaza):

Ponedeljak	1. čas	2. čas	3. čas	4. čas	5. čas	6. čas	7. čas	8. čas
A.A.	I_6	I_5	I_2	I_3				
B.B.				IV_5	III_3		II_4	
C.C.						III_1	IV_1	
D.D.	I_1	II_5	I_4	II_3				
E.E.					IV_4			
F.F.								
G.G.							IV_4	III_5
H.H.			III_2	III_5				
I.I.						II_3	II_4	
J.J.		I_4		I_5	IV_1			

Ovde je dat primer rasporeda za jednu srednju školu, a osmi čas se najčešće ne održava, iako u nekim slučajevima može da bude i održan ako se za tim ukaže potreba (za prepodnevnu smenu je to prvi čas druge smene). Naravno, u stvarnom rasporedu su dati svi dani i svi profesori, a ovo je samo primer kako bi raspored izgledao za jedan dan u nedelji. Ovde takođe nije uzeto u obzir da su učenici podeljeni po smenama (najčešće I i III razred idu zajedno, a II i IV) .

Sledeći raspored časova koji takođe pominjemo je raspored časova za učenike (ovde ćemo dati primer rasporeda časova jednog učenika, odnosno odeljenja kome taj učenik pripada (u pitanju je srednja škola):

	Ponedeljak	Utorak	Sreda	Četvrtak	Petak
1. čas	Hemija	Srpski	Francuski	Geografija	Srpski
2. čas	Matematika	Geografija	Hemija	Latinski	Engleski
3. čas	Fizičko	Likovno	Fizika	Srpski	Istorija
4. čas	Latinski	Biologija	Srpski	Matematika	Francuski
5. čas	Istorija	Matematika	Fizika	Biologija	Muzičko
6. čas	Engleski	Razredni čas		Fizika	Fizičko
7. čas					

Prepodne	1.čas	2.čas	3.čas	4.čas	5.čas	6.čas	7.čas	Popodne	1.čas	2.čas	3.čas	4.čas	5.čas	6.čas	7.čas
Ponedeljak	I_1	I_4	I_6	III_3								II_2	IV_2	II_5	
Utorak					III_2	III_6	III_4		IV_1	IV_5		II_6			
Sreda				I_4	III_3	III_5							II_2	IV_2	IV_6

Četvrtak				I_6	I_4		I_1				IV_1	II_5	II_1		
Petak			III_2	III_6	III_4	III_5			II_1	II_6	IV_5	IV_6			

Na kraju ćemo dati i raspored časova za jednu učionicu (kabinet, jer za učionice koje pripadaju odeljenjima raspored nije bitan) i vreme kada je ona zauzeta i ko u njoj ima časove, a kada je slobodna (uzećemo primer u jednom danu):

A raspored časova kad se sva tri spoje izgleda otprilike ovako:

ponedeljak	1.čas	2.čas	3. čas	4.čas	5.čas	6.čas	7.čas	8.čas
I_1								
I_2								
I_3								
I_4								
I_5								
I_6								

Ovo je samo deo rasporeda za prvu godinu i to samo jedan dan. Boje označavaju profesore i unapred su date kako bi se lakše snalazili sa njima.

3. Kako se pravi raspored časova u školama u Srbiji

Kao što smo do sada već i videli, problem rasporeda časova je jedan od onih problema kojim su se mnogi u svetu i kod nas bavili. Slobodno se može konstatovati da je značajan napredak napravljen u tom pravcu. Petar Hotomski je napisao nekolike knjige o ovom problemu, a napravio je i program koji doprinosi njegovom uspešno rešavanju. Konkretno, u našim školama (osnovnim i srednjim) nastavnici i đaci se susreću sa istim problemom - i jedni i drugi bi želeli što bolji raspored - đaci sa časovima, da ne bude suviše teških časova (matematika, srpski, biologija, istorija) u jednom danu, a nastavnici da što manje pauza imaju, da im časovi budu jedni za drugim, itd. Naravno, tu su i druga ograničenja - blok časovi, kad su nastavnici slobodni, neki časovi da budu na kraju dana, u zavisnosti od godine i smera zavisi i to koliko časova imaju tako da se za svaku godinu i odeljenje mora napraviti raspored i sve to da zadovoljava i nastavnike, a i neke druge uslove.

Kako naši nastavnici i škole to rešavaju? Neke od škola, naročito osnovnih, to i dalje rade ručno - jedan od nastavnika koji sastavlja raspored prikupi sve podatke koji su mu značajni, poznaje svakog od nastavnika, zna kada kome od njih odgovara da radi i i za kratko vreme napravi koliko - toliko pristojan raspored koji opet ne zadovoljava sve uslove u potpunosti, ali delimično da. Međutim, kako su srednje (a i neke osnovne) škole dosta velike, onda se najčešće pribegava drugom metodu - otkupe se programi sa interneta za pravljenje časova, ubace se podaci i program izbaci gotov raspored časova.

A kako se to radi ručno? Konkretno, u razgovoru sa jednim nastavnikom matematike saznao sam kako se on ovim problemom rasporeda časova bavio 20 godina u osnovnim školama gde nije bilo toliko đaka, razreda i nastavnika i gde je, znajući sa kojim kadrom raspolaže, relativno lako uspevao da uklopi sve kockice na svoje mesto. Na početku svake godine rasporede se profesori (najviše 100 u toj školi) ko će šta da predaje po odeljenjima - 40 odeljenja najviše u nekim školama, u našem konkretnom slučaju 24 - recimo manje od 30, i svaki nastavnik dobije određeni broj slobodnih mesta u rasporedu za razrede kojima predaje, a najviše 26 časova nedeljno, mada neki nastavnici rade pola radnog vremena, neki i manje, neki su zauzeti zbog drugih obaveza, itd. Po danu imamo 7-9 časova, najčešće i manje od 7, 5 dana u nedelji i ukupno oko 28 časova bez pauza < 30 časova. Imamo najviše 50 učionica i krećemo sa podelom: izvršena je podela časova, svaki profesor uzima tačno utvrđeno odeljenje i nijedan drugi ne uzima ta odeljenja.U početku se ne gleda da li se poklapaju časovi.

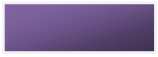
Pravljenje rasporeda možemo podeliti u 7 koraka :

a) svakom profesoru dodeliti onoliko kockica koliko ima časova. Profesorima dajemo kockice i uslove za njegovo radno vreme (koji dan, kada je slobodan, radno vreme, itd.);
b) raspoređivanje profesora u rešetku (ne gleda se da li se časovi istog profesora poklapaju);
c) ako se poklapaju, pomeraju se dok se ne napravi raspored u kome se profesori ne preklapaju;
d) optimizacija - ako ima 1, 2. i 7. čas, da li može da se nekako prebaci i kako?
e) specifični uslovi se unose na početku (zahtevi kad su profesori slobodni, itd.)
f) pedagoški uslovi - za predmete, da ne budu dan za danom, ako ima 3 časa - ponedeljak, sreda, petak, itd.
g) blok časovi, blok učionice, drugi dodatni uslovi.

Kao što vidimo, problema ima mnogo, ali ako se dobro poznaje materija, onda se konkretan raspored časova može napraviti relativno brzo premeštanjem kockica (pod ovim pojmom se podrazumevaju za svakog profesora drugačije označeni pravougaonici tako da premeštanjem zapravo menjamo raspored dvojici profesora) tako da skoro svi budu zadovoljni i veći broj zahteva bude ispunjen.

3.1. Primer kako bi se pravio raspored časova u školama u Srbiji:

Dakle, ovako bi izgledalo pravljenje rasporeda časova:

A.A.  x 20 B.B.  x 20 C.C.  x 16 D.D.  x 10,....

gde su A.A. , B.B., C.C. nastavnici koji redom imaju po 20, 20, 16, 10 časova i svaki čas je označen odgovarajućom bojom kako bi po njoj znali koji nastavnik drži čas u odgovarajuće vreme.

Njih smeštamo u raspored časova (uzećemo zbog prostora samo jedan dan):

Ponedeljak	1.čas	2.čas	3. čas	4.čas	5.čas	6.čas	7.čas	8.čas
I_1								
I_2								
I_3								
I_4								
I_5								
I_6								

Zatim bi se taj raspored proširio na ostale dane i ostala odeljenja sve dok ne dobijemo kompletan raspored časova u kome ne dolazi do preklapanja dve ili više boja odnosno dva ili više nastavnika i sve dok odgovarajući zahtevi – da učenici nemaju pauze u rasporedu, da nastavnici imaju što manje pauza, itd. - ne budu potpuno ili delimično ispunjeni

4. Heuristike

Heuristike predstavljaju konačan skup koraka kojim se dobijaju rešenja problema kombinatorne optimizacije (bez garancije njihove optimalnosti) za relativno kratko vreme. Osnovna prednost heurističkih metoda je njihova brzina, što omogućava dobijanje zadovoljavajućih rešenja za probleme velikih dimenzija kakvi se najčešće javljaju u realnim primenama. Nedostatak je što za dobijeno rešenje, ne samo da ne postoji garancija optimalnosti, već najčešće ni procena kvaliteta dobijenog rešenja. Međutim, za probleme velikih dimenzija osnovni cilj je da se neko rešenje dobije, bez obzira na njegov kvalitet. Kada rešenje već postoji, mogu se primeniti razne tehnike za njegovo poboljšanje, što je u suštini osnovna ideja prilikom nastajanja metaheuristika.

Heuristika je algoritam koji može da napravi prihvatljivo rešenje problema u mnogo praktičnih scenarija, ali za njih nema formalnog dokaza za njenu korektnost. Može da bude korektna, ali možda ne može da bude dokazano da će naći optimalno rešenje ili da će koristiti razumne izvore. Obično se koriste kada nema poznatog metoda koji može da nađe optimalno rešenje, pod datim ograničenjima. Dva važna zadatka nauke su da nađe algoritam sa dobrim vremenom rada i sa dobrim ili optimalnim rešenjem.

Heurističke metode mogu se podeliti u nekoliko grupa :

(i) *konstruktivne heuristike*;

(ii) *heuristike iterativnog poboljšavanja*;

(iii) *heuristike matematičkog programiranja*;

(iv) *dekompozicione heuristike*;

(v) *heuristike bazirane na podeli dopustivog skupa*;

(vi) *heuristike bazirane na restrikciji dopustivog skupa*;

(vii) *heuristike bazirane na relaksaciji*.

(i) *Konstruktivne metode*. Ova grupa metoda postepeno gradi (konstruiše) rešenje maksimalno koristeći specifična znanja svakog pojedinačnog zadatka.

(ii) *Iterativno poboljšavanje*. Savremeni naziv za ovu grupu je *metode lokalnog pretraživanja*. Polazi se od proizvoljnog početnog rešenja, koje se postepeno poboljšava pretraživanjem njemu *bliskih* rešenja. Proces se ponavlja sve dok u *blizini* tekućeg rešenja postoji bolje rešenje od tekućeg. Naravno, problem odredjivanja bliskih rešenja je ključan.

(iii) *Matematičko programiranje*. Problem se formuliše kao zadatak matematičkog programiranja, pa se onda rešava približnim (a ne egzaktnim) metodama.

(iv) *Dekompozicione heuristike*. Početni problem se razbije (dekomponuje) na više manjih potproblema, čijim se čak ni egzaktnim rešavanjem ne mora garantovati optimalnost rešenja za polazni problem.

(v) *Podela dopustivog skupa*. Dopustivi skup se podeli na više podskupova, pa se delimičnim pretraživanjem rešenja u svakom od njih nadje najbolje. Heurističko rešenje je ono kod kojeg je funkcija cilja najbolja.

(vi) *Restrikcija dopustivog skupa*. Restrikcijom se jednostavno eliminišu određeni podskupovi dopustivog skupa i tako smanjuje prostor pretraživanja. To omogućava da se novi zadatak lakše rešava.

(vii) *Relaksacija*. Suprotan pristup od restrikcije je relaksacija, kojom se dopustiv skup proširuje, ali tako da se omogućuje jednostavnije rešavanje novog problema.

Naravno, u rešavanju nekog konkretnog problema, moguće je kombinovati različite tipove heurističkih pristupa. Na primer, primenom konstruktivne heuristike pronađe se neko rešenje optimizacionog problema koje se zatim koristi kao polazno rešenje za iterativno poboljšavanje. Kako su se heuristike pokazale kao praktično jedini način rešavanja optimizacionih zadataka, istraživačka populacija u poslednje vreme sve više pažnje posvećuje njihovom razvoju i usavršavanju.

4.1. Metaheuristike

To je dovelo i do razvoja univerzalnih heuristika ili tzv. *metaheuristika*. Klasične heuristike su, uglavnom, bile namenjene rešavanju nekih konkretnih, pojedinačnih problema i koristile su poznate osobine datog problema pri njegovom rešavanju.

Metaheuristike, naprotiv, sastoje se od uopštenih skupova pravila koja se mogu primeniti za rešavanje raznovrsnih problema optimizacije. Metaheuristički pristupi u rešavanju optimizacionih problema zasnovani su na opštim algoritmima optimizacije koji podrazumevaju primenu iterativnih postupaka u cilju popravljanja nekog postojećeg rešenja. Metaheuristika je iterativni proces koji vodi niže heuristike kombinujući različite koncepte istraživanja i eksploatacije prostora pretrage koristeći naučene strategije da bi našli rešenje blizu optimalnog. + kažu da je metaheuristika iterativan master proces koji vodi i modifikuje ostale heuristike da bi došli do visokokvalitetnih rešenja. Može da upravlja sa kompletnim, nekompletnim jednim rešenjem ili kolekcijom rešenja u svakoj iteraciji. Niže heuristike mogu biti procedure na višem ili nižem nivou ili prosto lokalna pretraga ili samo konstruktivni metod.

Sa druge strane, razvile su se mnoge metaheurističke metode optimizacije, najčešćce po uzoru na neke poznate procese, prvenstveno u biologiji (genetski algoritmi (GA) i razne specijalne varijante evolutivnih algoritama (EA), neuralne mreže (NN), mravlje kolonije (AC), u fizici (simulirano kaljenje(SA)), ali i metode inspirisane lokalnim pretraživanjem sa raznim idejama da se izbegne zamka lokalnog minimuma (višestartno lokalno pretraživanje

(MLS), metoda promenljivih okolina (VNS), tabu pretraživanje (TS), Greedy Randomized Adaptive Search Procedure (GRASP) i druge.

Metaheurističke algoritme možemo podeliti po različitim osnovama, a najvažnije podele su :

- *Prirodom-inspirisani algoritmi* i *algoritmi koji nisu prirodom inspirisani*. Primeri prirodom-inspirisanih algoritama su *optimizacija mravljom kolonijom* i *genetski algoritmi*, a primeri onih koji nisu prirodom inspirisani su *tabu pretraživanje* i *lokalno pretraživanje*.
- *Algoritmi koji imaju dinamičku funkciju cilja* i *algoritmi koji imaju statičku funkciju cilja.* Većina algoritama ne menja svoju funkciju cilja (funkcija koja se optimizuje), ali postoje algoritmi poput *vođenog lokalnog pretraživanja* čija se funkcija cilja menja u svrhu bežanja iz lokalnog optimuma. Takvi algoritmi imaju dinamičku funkciju cilja.
- *Algoritmi koji koriste jednu strukturu okoline* i *algoritmi koji koriste skup struktura okoline.* Algoritmi uglavnom koriste jednu strukturu okoline, ali postoje algoritmi poput *metode promenljvih okolina*, koja koristi skup struktura okoline.
- *Algoritmi koji imaju mogućnost pamćenja prethodnih rešenja* i *oni koji to nemaju.* Primer*, tabu pretraživanje (TS)* ima mogućnost pamćenja prethodno odabranih rešenja, dok *algoritam simuliranog kaljenja (SA)* nema. On se koristi drugim metodama da bi odabrao dobro rešenje u sledećoj iteraciji.
- *Konstruktivni, poboljšavajući i hibridni algoritmi.* Konstruktivni algoritmi grade rešenje (primer, *pohlepni algoritam*). Poboljšavajući algoritmi biraju rešenje i usavršavaju ga kroz niz iteracija (primer, *algoritam simuliranog kaljenja*). Hibridni algoritmi su njihova kombinacija (primer, *GRASP*).
- *Algoritmi bazirani na populaciji rešenja* i *algoritmi putanje.* Algoritmi koji se baziraju na populaciji rešenja koriste skup rešenja u proceni trenutnog optimuma (primer, *Stohastic Diffusion Search*), dok algoritmi putanje koriste jedno trenutno rešenje u svakom koraku (primer *Lokalno pretraživanje*).

Metaheuristike su postale značajno, a najčešće i jedino, sredstvo u rešavanju realnih problema baziranih na optimizaciji. Osnovni zahtev je dobijanje rešenja bliskih optimalnom u razumnom vremenu. Osim toga, mogu se primeniti za ubrzavanje tačnih metoda tako što će se koristiti za dobijanje dobrog početnog rešenja. Skorašnje primene potvrđuju da su se pokazale veoma uspešnim i kao sastavni delovi sistema za otkrivanje znanja u okviru veštačke inteligencije. U svetlu ovih primena mogu se definisati osobine koje efikasne metaheuristike treba da poseduju kako bi obezbedile značaj i na praktičnom i na teorijskom planu:

1. *Jednostavnost* : treba da budu zasnovane na jednostavnim i lako razumljivim pravilima;
2. *preciznost* : koraci kojima se opisuje metaheuristička metoda treba da su formulisani preciznim, po mogućnosti matematičkim terminima;
3. *doslednost* : svi koraci metode treba da budu u skladu sa pravilima kojima je metaheuristika definisana.

5. SA

Algoritam simuliranog kaljenja (Simulated Annealing, SA) jedan je od stohastičkih optimizacionih algoritama (metaheuristika), a razlikuje se od algoritama lokalnog pretraživanja po tome što omogućuje beg iz lokalnog optimuma. Algoritam je nastao po analogiji sa metalurškim kaljenjem, čiji je cilj oplemenjivanje metala tako da oni postanu čvršći. Ako se želi postići čvrstoća metala, potrebno je kristalnu rešetku metala pomeriti tako da ima minimalnu potencijalnu energiju. U procesu metalurškog kaljenja metal se prvo zagreva do visoke temperature, a potom se, nakon kraćeg zadržavanja na toj temperaturi, polagano hladi do sobne temperature. Prebrzo hlađenje bi moglo uzrokovati pucanje metala. Posledica toga je da atomi metala nakon procesa kaljenja formiraju pravilnu kristalnu rešetku. Time je postignut i energetski minimum kristalne rešetke. U skladu s navedenim, SA će sadržati parametar temperature, a funkciju kojoj želimo odrediti globalni optimum možemo posmatrati kao energiju rešetke, ukoliko određujemo minimum, odnosno negativnu energiju rešetke, ukoliko određujemo maksimum.

Prevedeno na matematički model to izgleda ovako:

Algoritam počinje izborom nekog početnog rešenja, a početna temperatura ima relativno veliku vrednost. Postojeće rešenje se zamenjuje sa boljim, ali se može zameniti i sa lošijim uz određenu verovatnoću prihvatanja. Ta verovatnoća se određuje izborom slučajnog broja *a* iz intervala [0, 1] i uslova da je K funkcija za koju se traži globalni minimum, a *T* temperatura. Ukoliko je izraz istinit novo rešenje se prihvata. Ovo nam omogućava beg iz lokalnog minimuma. Isto tako, verovatnoća da će biti odabrano lošije rešenje je veća kada je veća temperatura. To znači da je u početku prostor pretrage rešenja dosta veliki i da se smanjuje padom temperature, a pri kraju procesa je usko lokalizovan. Ponašanje funkcije je u velikom određeno početnom vrednosti temperature i njenom brzinom padanja. Najčešće temperatura opada eksponencijalno s parametrom *a* iz intervala $0 < a < 1$ (a je blizak jedinici). Za premali *a* hlađenje je prebrzo (pucanje metala) i veća je verovatnoća upada u lokalni optimum. Za preveliki *a* temperatura presporo pada, a zbog velike verovatnoće da se u početku zamenjuju bolja rešenja s lošijim, algoritam ima svojstvo da nasumično pretražuje veliki prostor rešenja. To dodatno usporava rad, pa se nastoji da se *a* odabere da bude dovoljno veliko, ali da temperatura pri kraju procesa bude niska. Time se rešenje pred kraj procesa stabilizuje u lokalnom području. Zbog velike osetljivosti na koeficijent *a*, posebnu pažnju treba obratiti prilikom njegovog izbora . Kada se formira početno rešenje, algoritam generiše move m (to je bilo koja promena, pokret) u svakoj iteraciji i izvodi ga po sledećem pravilu: Move je prihvaćen ako poboljšava cost vrednost trenutnog stanja, inače se prihvata prema verovatnoći opadanja vremena. Kasnije, ako je test prihvatanja negativan, ništa se ne menja. Kontrolni parametar se naziva temperatura. Na početku se stavlja na neku visoku vrednost i vremenom opada prateći šemu hlađenja. Proces staje kada je T blizu 0. Sistem je smrznut i dostignut je lokalni minimum.

SA je produžetak prostog algoritma opadanja ali sa manje striktnim pravilom prihvatanja. Bolji potezi se uvek prihvataju, a lošiji se prihvataju sa određenom verovatnoćom. Metropolis

algoritam simulira promenu energije sistema kada se hladi dok ne dođe do mirnog stanja. Kirkpatrick et al. (1983) predlažu da se SA koristi da traži prihvatljiva rešenja u problemu optimizacije čiji zadatak je da konvergira ka optimalnom stanju. Važan je raspored hlađenja-početna, krajnja T, kada i kako je T snižena i izbor strukture i okruženja (Aarts and Korst, 1998). Pretraga počinje sa visokom T da bi loši potezi bili prihvaćeni i onda se T polako spušta.

5.1. Osnovni koraci u korišćenju SA :

Δ je razlika između novog i starog rešenja;

$\Delta<0$ - novo rešenje postaje pravo rešenje;

$\Delta\geq0$- novo rešenje se prihvata sa verovatnoćom $e^{-\Delta/T}$, T se zove temperatura

T je u početku najviše – To, a posle određenog broja iteracija temperatura opada sa koeficijentom a, $Tn=a\cdot Tn-1$, a pripada intervalu (0, 1) uključujući i krajnje tačke 0 i 1 .

Završava se kada je temperatura približna i nijedno rešenje ne povećava objektnu funkciju, zamrznut sistem. To je lokalni minimum.

Kontrolni faktori - a, broj iteracija i To.

Primenjeno na naš problem rasporeda časova - atome menjamo elementima, a energiju sa vrednošću rasporeda časova. Inicijalna alokacija - elementi su nasumično raspoređeni u odgovarajuće periode. Dajemo inicijalnu cost funkciju i temperaturu. Cost - kvalitet rasporeda, temperatura - kontrola verovatnoće povećanja cost funkcije. U svakoj iteraciji period nasumično biramo – from period i za element slučajno odabran iz tog perioda računamo cost. Početni period se bira nasumično, zovemo ga To-period i iz toga računamo cost.

Princip rasporeda časova primenjen na SA (Abramson i Dang, 1993) :

While (cost<>0) i raspored nije smrznut repeat

repeat određeni broj puta

izaberi Tn (odeljenje , učionica, period)

izaberi polje za period- Period

izračunaj cost - brisanje Tn iz perioda

izračunaj cost- ubacivanjem Tn u period

izračunaj razliku u cost-u

ako je razlika u cost-u ≤ 0 razlika je prihvatljiva

prihvati promenu i obnovi cost

izračunaj novu temperaturu.

Predloženi SA algoritam hiper heuristika je zasnovan na sledećim pretpostavkama i zapažanjima:

1. Svaka heuristika je povezana sa težinom koja označava njenu trenutnu važnost. Tokom pretrage ove vrednosti se dinamično menjaju.
2. Mehanizam promene težina je strategija nagrada-kazna.
3. Težina se povećava ako se poboljšava rešenje i obrnuto. Za one heuristike koje ne mogu da poboljšaju funkciju, razlikujemo heuristike koje generišu novo rešenje i one koje to ne rade. Zapažamo da tokom pretrage, iako neke heuristike ne mogu da generišu rešenje direktno, one su i dalje korisne u kreiranju nekih međusituacija, od kojih optimalno ili dobro rešenje može biti nađeno. Ovde dajemo najmanji pozitivan skor onim heuristikama koje mogu da promene položaj rešenja, ali ne i da poboljšaju traženo rešenje. U međuvremenu kažnjavamo one heuristike koje ne poboljšavaju rešenje niti generišu novo.

6. TS

Tabu pretraživanje (Tabu Search, TS) je metaheuristika koja je detaljno opisana u knjizi Glovera i Lagune[6]. Metodu su nezavisno razvili Glover i Hansen. TS metoda zasniva proces pretraživanja na okolini tekućeg minimalnog rešenja. Kada se u toku pretraživanja naiđe na lokalni minimum, potrebno je definisati kriterijum izlaska iz njega. Taj kriterijum je kod TS metode zasnovan na upotrebi memorija. Za razliku od ostalih navedenih metoda koje čuvaju samo trenutno najbolje rešenje i odgovarajuću (trenutno minimalnu) vrednost funkcije cilja, TS metoda pamti kretanje preko rešenja posećenih u nekoliko prethodnih iteracija. Te informacije se koriste za izbor narednog rešenja kao i za modifikaciju definicije okoline u smislu izbacivanja nekih "zabranjenih" rešenja. Obzirom na to, jasno je da okolina $N(x)$ rešenja x varira od iteracije do iteracije, te se stoga TS ubraja u procedure koje se nazivaju *tehnike dinamičkog pretraživanja okoline*.

Preciznije, opis izvršavanja osnovne verzije TS metode sastoji se u sledećem: ova metoda sistematski prati proces minimizacije sve dok se ne dostigne lokalni minimum. Zatim se taj lokalni minimum isključuje iz okoline za pretraživanje i traži minimalno rešenje u tako modifikovanoj okolini. Isključena rešenja se pamte u listi nazvanoj *tabu lista* (*TL*) koja predstavlja zabranjena rešenja u toku nekoliko narednih iteracija (definisanih dužinom *TL*). Po isteku zabrane, ova rešenja vraćaju se u okolinu, a neka druga postaju zabranjena.

Nekoliko strategija je predloženo u literaturi da bi rešili problem TS-a primenjujući ih u određenim fazama algoritma TS-a. Recimo promena kazni je mehanizam koji menja izgled cost funkcije u prilagodljiv oblik. Na taj način omogućava algoritmu lokalne pretrage da poseti rešenja sa različitom strukturom od prethodnih. Da bi ispitali veću okolinu trenutnog rešenja, neka ograničenja će biti relaksirana tako da ćemo ispitivati i područja gde ta ograničenja ne važe.

Drugi uslov na koji ćemo ovde takođe obratiti pažnju je veličina tabu liste. Dinamički tabu list pristup menja dužinu tabu liste na sledeći način: ako poslednji pokret menja odnosno poboljšava cost funkciju, onda je dužina liste kraća da bi ubrzalo pretragu, a ako sekvenca pokreta izaziva opadanje cost funkcije, lista se povećava da bi izašli iz traženog dela. Radeći sa strategijama pretrage, koristimo mešavinu HC i TS gde HC izvodi duple, a TS atomske pokrete. Početno rešenje se formira nasumično poštujuići potrebnu matricu ili može biti dobijena iz prethodnih izvršenja. Onda radi HC dok ne bude prestala sa poboljšanjima u nekoliko iteracija, a onda radi TS. HC se koristi iz 2 razloga :

- Pravi dobro početno rešenje za TS
- Kada TS ne pravi pomak, opet treba primeniti HC na dobijeno rešenje.

HC može da nađe poboljšanja koja TS nije uspeo, a i HC sa duplim pokretima menja solidna rešenja i prilazi sa druge strane problemu iako ne dovodi do poboljšanja. Dakle, prodrma rešenje pre nego što TS opet krene. Ideja je da TS opet krene sa druge strane jer je stao i nije mogao dalje. Glavnu ideju za problem kombinatorne optimizacije je dao Glover (1986). Glover and Laguna (1997), Voß (2000), Voß et al (1999), Osman and Laporte (1996) su se takođe bavili problemom TS-a.

[6] Vidi [9]

Definicija TS -a Glovera i Lagune (1997) glasi : TS je meta heuristika koja vodi lokalnu heurističku pretragu da ispita prostor rešenja ispod lokalnog optimuma.

Glover and Laguna (1997) daju dve važne strategije tabu pretrage : intensifikacija i diversifikacija. Prva označava promenu pravila izbora da bi pojačali pretragu i ispitali okolinu najboljih rešenja to jest ako je određena oblast davala dobra rešenja u prošlosti, davaće još bolja u budućnosti. Druga strategija ispituje neispitane oblasti i stvara rešenja koja se značajno razlikuju. Tri pristupa za definisanje veličine tabu liste: fiksirana, sličajno odabrana ili dinamički se menja da bi se prilagodila nekoj vrednosti.

Na osnovu do sada izloženog može se opisati pseudokod osnovne verzije TS metode:

Hertz (1991.) obezbeđuje dobar algoritam :

6.1. Algoritam TS-a

s:= inicijalno rešenje u X

nbiter:=0 – trenutna iteracija

bestiter:=0- iteracija kada je nađeno najbolje rešenje

bestsol:=s – najbolje rešenje

T:=0

inicijalizujemo željenu funkciju A

while (f(s)>f·) i (nbiter-bestiter<nbmax) do

nbiter:= nbiter+1

generiši skup V· rešenja s u n(s) koja ili nije tabu ili takva da je A(f(s))≥f(s)

izaberi rešenje s· minimizirajući f na V·

obnovi funkciju A i tabu listu T

if f (s·) <f (bestsol) then

bestsol := s·

bestiter := nbiter

s:= s·

Tabu restrikcija se koristi da izbegne ponavljanje pretrage i to ako se u više iteracija primeti ponavljanje. Kriterijumi zaustavljanja su različiti:

1. Staje posle određenog broja iteracija;
2. Ako broj iteracija posle poslednjeg poboljšanja prevazilazi određeni broj iteracija;

3. Dužina tabu liste;
4. Funkcija aspiracije;
5. Broj skupova okoline rešenja testiranih u svakoj iteraciji.

Dakle, kao što smo videli ovaj algoritam se može primeniti na naš problem rasporeda časova tako što generišemo proizvoljni raspored i onda prateći algoritam TS-a pokušavamo da ga popravimo i dobijemo što optimalnije rešenje.

Algoritam penjanja uzbrdo (**Hill Climbing**, **HC**) po svom načinu optimizacije pripada familiji algoritama lokalnog pretraživanja. Temeljen je na logici uspona: algoritam počinje od nekog rešenja i ukoliko postoji *sused* koji bolje optimizira funkciju, novo rešenje postaje taj *sused*. Veliki problem HC-a je mogućnost zapinjanja u lokalnom optimumu. Ovakva skraćenica algoritma je poznata pod nazivom **jednostavni HC** (**Simple HC**). Bolji od predhodnog HC-a je **stohastičko-restartovan HC** (**Random-Restart HC, RRHC**). RRHC u svakoj iteraciji slučajno bira početno rešenje i nad njime, provodi SHC. Kao svoje rešenje bira najbolje rešenje iz svih iteracija. Ovakvo rešenje je slično *višestartnom lokalnom pretraživanju*. HC vrši samo promene koji ne menjaju ili poboljšavaju cost funkciju. Selekcija pokreta se vrši ili slučajno ili istražujući okolinu i tražeći najbolju promenu i pokret (dalje poznato kao najdublje spuštanje). Pretraga staje kad ne može doći do poboljšanja ili posle određenog broja koraka. Glavni problem je što može da se zaglavi u strogom lokalnom minimum području okruženom najgorim rešenjima ili da kruži u skupu jednakih vrednosti. Da bi rešili ovaj problem uvedeni su SA i TS koji se oslanjaju na dve različite filozofije : verovatne odluke i zabrane zasnovane na pamćenju.

Lokalno pretraživanje (Local search, LS) podrazumeva da se za svako $x' \in N(x)$ u okolini nekog početnog rešenja x, izračunava vrednost $f(x')$ i ukoliko je $f(x') < fmin$, čuva se novo $xmin = x'$ i $fmin = f(x')$. Kada se "obiđu" sva rešenja u okolini $N(x)$, nastavlja se pretraga u okolini $N(xmin)$. Ispitivanje svih suseda nekog rešenja *xmin* naziva se pretraživanje okoline (neighborhood exploration, NE). NE predstavlja jednu iteraciju lokalnog pretraživanja. Proces lokalnog pretraživanja odvija se u iteracijama i zaustavlja se kada u okolini $N(xmin)$ ne postoji bolje rešenje lokalnog pretraživanja. Pseudokod za LS može se prikazati u sledećem obliku :

1. *Inicijalizacija*. Izabrati početno rešenje x (slučajno ili primenom neke konstruktivne heuristike).Postaviti $xmin = x$ i $fmin = f(x)$.

2. **repeat**

POPRAVKA = 0;

$x' \in N(xmin)$

if ($f(x') < fmin$) **then**

$xmin = x'$;

$fmin = f(x')$;

POPRAVKA = 1;

endif

while POPRAVKA == 0;

7. GA i EA

Neće se mnogo pogrešiti ako se kaže da su genetski algoritmi izum prirode. Tačnije, genetski algoritmi su nastali kao simulacija nekih procesa zapaženih u prirodnoj evoluciji. Biolozi su bili zaintrigirani mehanizmom evolucije još od kada je teorija evolucije prihvaćena kao opšti uzrok bioloških promena. Mnogi su zapanjeni saznanjem da se život na našoj planeti mogao razviti do sadašnjeg nivoa složenosti u relativno kratkom vremenu, sa obzirom da kao dokaz toga imamo brojne fosilne ostatke. Evolucija je neprekidan proces prilagođavanja živih bića na svoju okolinu tj. na uslove u kojima žive. U prirodi vlada nemilosrdna borba za opstanak u kojoj pobeđuju najbolji a loši umiru. Da bi neka vrsta tokom evolucije opstala, mora se prilagoditi uslovima i okolini u kojoj živi, jer se uslovi i okolina menjaju. Svaka sledeća generacija neke vrste mora pamtiti dobra svojstva prethodne generacije, pronalaziti i menjati ta svojstva tako da ostanu dobra u neprekidno novim uslovima. Mehanizam po kome funkcioniše evolucija nije u potpunosti razjašnjen, ali su neke od njegovih karakteristika poznate. Evolucija se odigrava na hromozomima, koji su organski uređaji za kodiranje strukture živih bića. Kako specifičnosti hromozomskog kodiranja i dekodiranja nisu poznate, sledeće karakteristike o teoriji evolucije široko su prihvaćene:

- Proces evolucije operiše na hromozomima, a ne na živim bićima koja su kodirana hromozomima;
- Proces prirodne selekcije prouzrokuje da se češće reprodukuju hromozomi koji kodiraju uspešne strukture nego drugi hromozomi;
- Reprodukcija je tačka na kojoj evolucija počinje svoje delovanje:
  - rekombinacija (ukrštanje) može stvoriti različite hromozome kod dece, kombinovanjem materijala iz hromozoma njihovih roditelja,
  - mutacija može rezultirati da hromozomi kod dece budu različiti od hromozoma njihovih bioloških roditelja,
- evolucija nema memoriju.

U ranim sedamdesetim godinama prošlog veka, gore navedene karakteristike prirodne evolucije, zainteresovale su naučnika Johna Hollanda (1975). On je verovao da će uspeti da napravi tehniku za rešavanje teških problema ako na odgovarajući način sjedini ove karakteristike u računarski algoritam. Tako je počeo istraživanje na algoritmu koji upravlja nizovima binarnih cifara (0,1), u kome nizovi predstavljaju hromozome. Koristeći jednostavno kodiranje i mehanizam reprodukcije, Hollandov algoritam je rešio neke ekstremno teške probleme. Kao i priroda, algoritam je bio običan upravljač nad hromozomima. Primenom nekih verzija ovog algoritma danas, postižu se bolja rešenja na širokom spektru problema koje ne možemo rešiti drugim tehnikama. Kada je Holland počeo da izučava ove algoritme, oni nisu imali ime. Obzirom na njihovo poreklo iz nauke o genetici nazvani su **genetski algoritmi (GA).** Posle velikog broja istraživanja sprovedenih u ovoj oblasti, genetski algoritmi su razvijeni.

Sada su *GA stohastička metoda globalnog pretraživanja koja imitira prirodnu biološku evoluciju.* Primenom principa preživljavanja, koji se sastoji u tome da se od najpogodnijih jedinki proizvode bolja rešenja, dolazimo do bitne osobine GA: genetski algoritmi operišu na populaciji potencijalnih rešenja. Aproksimacija novih rešenja u svakoj generaciji dobija se procesom *selekcije* jedinki prema njihovom *nivou prilagođenosti (fitness)* u domenu problema kao i stvaranjem novih jedinki korišćenjem operatora "pozajmljenih" iz genetike kao što su *ukrštanje* i *mutacija*. Ovi procesi rezultiraju na takav način da se jedinke dobijene

njima bolje uklapaju u okolinu nego one od kojih su stvorene, baš kao i u prirodnom prilagođavanju.

Ukratko, osnovne ideje metode su da se izabere inicijalna populacija i da se zatim kroz niz generacija evoluiranja te populacije vrši popravka trenutno najboljeg rešenja. Popravka, tj. transformacija polaznih rešenja i generisanje potomaka se vrši primenom tzv. *operatora*, od kojih su osnovni *selekcija*, *ukrštanje* i *mutacija*.

7.1. Osnovna struktura GA

Postoji više načina da se prethodno pomenute karakteristike prirodne evolucije povežu sa genetskim algoritmima. Za početak treba uvažiti dva mehanizma koja povezuju GA sa problemom koji se rešava. Jedan od njih je način kodiranja potencijalnih rešenja na problem hromozoma, tj. njihovo kodiranje u hromozome. Drugi je funkcija ocene (fitness), koja predstavlja meru valjanosti nekog hromozoma. Način kodiranja rešenja igra važnu ulogu u GA.

Tehnika kodiranja može varirati od problema do problema kao i od GA do GA. U ranim GA za kodiranje su korišćeni nizovi bitova, binarno kodiranje. Kasnije su naučnici razvijali i mnoge druge tehnike kodiranja. Najverovatnije je da ni jedna tehnika kodiranja ne radi najbolje za sve probleme, pa je potrebna velika veština u izboru dobre tehnike kada se problem proučava. Zato, kada se bira tehnika za kodiranje u nekom stvarnom problemu, treba obratiti pažnju na niz faktora, o kojima će biti reč kasnije.

Funkcija ocene (*fitness function*) je veza između GA i problema koji se rešava. Funkcija ocene uzima kao ulazni podatak hromozom, a kao rezultat vraća broj ili listu brojeva koji predstavljaju performanse tog hromozoma. Ova funkcija igra istu ulogu u GA kao što okolina to radi u prirodnoj evoluciji. Interakcija jedne jedinke sa okolinom daje meru njene prilagođenosti, a interakcija hromozoma sa funkcijom ocene određuje meru pogodnosti, tj. koliko je taj hromozom pogodan za dalju reprodukciju. Ako se uzme da su date početne komponente: problem, tehnika za kodiranje rešenja i funkcija koja daje informaciju o tome koliko je neko rešenje dobro, tada se može primeniti GA da sprovede simuliranu evoluciju na nekoj populaciji rešenja. Osnovna struktura GA izgleda ovako:

a) Inicijalizovati populaciju hromozoma,
b) Oceniti “kvalitet” svakog hromozoma u populaciji,
c) Stvoriti nove hromozome od hromozoma iz postojeće populacije,
d) Ukloniti neke članove populacije da bi se napravilo mesta za nove hromozome,
e) Ubaciti nove hromozome u populaciju,
f) Zaustaviti proceduru i prikazati najbolji hromozom ako je vreme isteklo, u suprotnom ići na korak b.

Treba napomenuti da kriterijum zaustavljanja kod GA može biti maksimalan broj generacija, broj generacija bez popravke trenutno najboljeg rešenja, prevelika sličnost jedinki ili maksimalno dozvoljeno vreme izvršavanja koje je i najrasprostranjenije jer omogućava poređenje sa drugim metodama. Na osnovu formulisane osnovne strukture, pseudokod GA ima sledeci oblik (u pseudo PASKAL-u):

7.1.1. Procedura GA

begin

t = 0;

Generiši inicijalnu populaciju P(t)=P(0);

“Oceni” P(t);

While (dok vreme ne istekne) **do**

begin

t = t +1;

Selektuj P(t) od P(t-1);

Reprodukuj parove u P(t)

begin

Ukrštanje;

Mutacija;

Reinsertovanje;

end

“Oceni” P(t);

end

end

U ovom pseudokodu populacija hromozoma u trenutku t je predstavljena pomoću vremenski zavisne promenljive P(t) sa inicijalnom populacijom P(0). Ako u ovoj proceduri sve prođe dobro, početna populacija će biti zamenjena boljim hromozomima. Najbolja jedinka u finalnoj populaciji može biti najbolje rešenje problema.

Genetski algoritmi se bitno razlikuju od mnogih tradicionalnih optimizacionih metoda i metoda pretraživanja. Najznačajnije razlike su :

- GA pretražuje populaciju čvorova u prostoru potencijalnih rešenja, a ne jedan čvor, što mu daje osobinu robustnog algoritma;
- Tokom procesa rešavanja, GA ne koristi dodatne informacije o prirodi problema;
- GA koriste probabilistička (verovatnosna) pravila, a ne deterministička
- GA ne koristi same parametre,već vrši kodiranje parametara.

Važno je primetiti da GA proizvodi mnogo potencijalnih rešenja za zadati problem, a izbor konačnog rešenja se ostavlja dizajneru. Izbor takođe zavisi od prirode i veličine problema, kao i od toga da je pored kvaliteta jako bitna i brzina dobijanja rešenja.

Genetski algoritmi koriste mehanizam selekcije za izbor jedinki koje će učestvovati u reprodukciji. Selekcijom se omogućava prenošenje boljeg genetskog materijala iz generacije u generaciju. Zajedničko svojstvo svih vrsta selekcija je veća verovatnoća izbora boljih jedinki za reprodukciju. Postupci selekcije se međusobno razlikuju po načinu izbora boljih jedinki. Prema načinu prenošenja genetskog materijala boljih jedinki u sledeću iteraciju postupci selekcije se dele na :

a) *generacijske selekcije* - selekcijom se direktno biraju bolje jedinke čiji će se genetski materijal preneti u sledeću iteraciju i
b) *eliminacijske selekcije* - biraju se loše jedinke za eliminaciju, a bolje jedinke prežive postupak selekcije.

Generacijskom selekcijom se biraju bolje jedinke koje će učestvovati u reprodukciji. Između jedinki iz populacije iz prethodnog koraka biraju se bolje jedinke i kopiraju u novu populaciju. Ta nova populacija, koja učestvuje u reprodukciji, naziva se *međupopulacijom*. Na taj način se priprema nova generacija jedinki za postupak reprodukcije. To je ujedno prvi nedostatak generacijskih selekcija, jer se u jednoj iteraciji odjednom nalaze dve populacije jedinki. Broj jedinki koje prežive selekciju je manji od veličine populacije. Najčešce se ta razlika u broju preživelih jedinki i veličine populacije popunjava duplikatima preživelih jedinki . Pojava duplikata u sledećoj iteraciji je drugi nedostatak generacijskih selekcija, jer duplikati nikako ne doprinose kvalitetu dobijenog rešenja, već samo usporavaju proces evolucije. Generacijska selekcija postavlja granice među generacijama. Svaka jedinka egzistira tačno samo jednu generaciju. Izuzetak su jedino najbolje jedinke koje žive duže od jedne generacije i to samo ako se primeni elitizam.

Kod eliminacijskih selekcija bolje jedinke preživljavaju mnoge iteracije pa stroge granice između generacija nema, kao što je to slučaj kod generacijskih selekcija. Eliminacijska selekcija briše M jedinki. Parametar M naziva se mortalitetom ili generacijskim jazom. Izbrisane jedinke se zamenjuju jedinkama koje se dobijaju reprodukcijom. Eliminacijskom selekcijom se otklanjaju oba nedostatka generacijske selekcije: nema dve populacije u istoj iteraciji i u postupku selekcije se ne stvaraju duplikati. Na prvi pogled je uvođenje novog parametra M nedostatak eliminacijske selekcije, jer se svakim dodatnim parametrom značajno produžava postupak podešavanja algoritma. Međutim, uvođenjem novog parametra, nestaje parametar verovatnoće ukrštanja, jer ukrštanje treba obaviti tačno onoliko puta koliko je potrebno da se dopuni populacija do početne veličine N. Što je veća verovatnoća ukrštanja, treba biti veći mortalitet i obrnuto. Na primer, ako se koristi uniformno ukrštanje, koje proizvodi jednu novu jedinku, tada tačno M puta treba ponoviti ukrštanje kako bi se dopunila populacija, odnosno zamenile eliminisane jedinke.

Selekcijom se osigurava prenošenje boljeg genetskog materijala s većom verovatnoćom u sledeću iteraciju. Zavisno od metode izbora boljih jedinki kod generacijskih selekcija, odnosno loših jedinki kod eliminacijskih selekcija, postupci selekcije se dele na **proporcionalne** i **rangirajuće**. Važno je napomenuti da je zajedničko obeležje svim selekcijama veća verovatnoća preživljavanja bolje jedinke od bilo koje druge lošije jedinke. Selekcije se razlikuju prema načinu određivanja vrednosti verovatnoće selekcije određene jedinke.

Proporcionalne selekcije biraju jedinke s verovatnoćom koja je proporcionalna prilagođenosti jedinke, odnosno verovatnoća selekcije određene jedinke zavisi od koeficijenta prilagođenosti jedinke i prosečne prilagođenosti populacije. Broj jedinki s određenim svojstvom u sledećoj iteraciji je proporcionalan koeficijentu prosečne prilagođenosti tih jedinki i prosečne prilagođenosti populacije.

Rangirajuće selekcije biraju jedinke s verovatnoćom koja zavisi od položaja jedinke u poretku jedinki sortiranih po prilagođenosti. Rangirajuće selekcije se dele na *sortirajuće* i *turnirske selekcije*. Sortirajuće selekcije su: linearno rangirajuća selekcija i selekcija najboljih i selekcija isecanjem. Turnirske selekcije se dele prema broju jedinki koje učestvuju u turniru.

U proučavanju operatora selekcije autori polaze od različitih prethodno definisanih veličina. Najčešće korišćene veličine koje karakterišu operator selekcije su: trajanje preuzimanja, koje su uveli naučnici Goldberg i Deb, zatim stopa reprodukcije, intenzitet selekcije, selekciona razlika, gubitak raznovrsnosti i selekcioni pritisak.

7.2. Veza sa rasporedom časova:

Kako bi rešili problem rasporeda časova, razvili smo metod optimizacije baziran na GA koji sadrži određene napredne tehnike. Prvi problem je bio kako kodirati rešenje problema rasporeda časova u formu hromozoma, prikladnu za genetske operatore. U literaturi se koriste dva pristupa: direktni i indirektni.

Direktni direktno kodira sve potrebne događaje - dane, periode, nastavnike, učionice za sve događaje. U ovim slučajevima GA mora da uradi za sve parametre problema i da isporuči kompletni raspored bez sukoba. Prateći ovaj princip rezultira pretragom u veoma širokoj oblasti gde su rešenja kao igla u plastu sena. Najčešće ovakva rešenja nisu tačna i moraju na neki način biti ispravljana i popravljana.

U indirektnom pristupu kodirano rešenje to jest hromozomi obično predstavljaju listu događaja po nekom redu, koji se smeštaju u raspored po nekom ranije utvrđenom redosledu. Ovaj metod izrade rasporeda časova može koristiti bilo koju kombinaciju heuristika i lokalnih pretraga da bi napravio raspored, poštujući ograničenja. Ovde radimo indirektni pristup koji dekodira 4 polja za svaki događaj u hromozome:

a) Dan kada se održava događaj
b) Nastavnici dodeljeni za čas
c) Učionica
d) Prioritet časova koji će prvi u tom danu biti smešteni.

Sva polja su prvo kodirana kao celi brojevi i onda uneti kao binarni brojevi. Kada GA radi na ovaj način, prvo dekodira da bi dobila ova 4 polja za svaki čas u rasporedu. Onda pozove izgrađivač rasporeda časova zvan “timetabler” koji radi na sledeći način:

- Deli događaje u skupine, za svaki dan
- Za svaku skupinu, sortira časove po važnosti u opadajući poredak – male vrednosti znače visok prioritet i prve su na listi.
- Uzima prvi događaj iz skupine visokog prioriteta označava da je uzet i pokušava da ga rasporedi u dati dan.

- Počinjući od prvog perioda smešta događaje i proverava da li je neko ograničenje narušeno. Ako nije, smeštanje je završeno i ide se na sledeći događaj.
- Ako je neko ograničenje narušeno, pokušava da alocira čas u drugi period tako da sva ograničenja budu zadovoljena.
- Ako ne postoji period kada su ograničenja zadovoljena, čas se označava da krši "maksimum perioda po danu " ograničenje.

Algoritam nastavlja sa sledećim časom u listi. Kada završi sa svim časovima u jednom danu, ide na sledeću skupinu (dan) i tako dok ne završi. Timetabler ispunjava i teška ograničenja.

Kada timetabler napravi raspored, računa se fitness funkcija koja analizira rešenje i računa sveukupnu fitness vrednost kao sumu težina svih ograničenja i teških i lakih, kao i zadatka. Jasno je da su neka ograničenja rešena u samoj konstrukciji rasporeda časova. Ostala ograničenja su rešena koristeći kaznene funkcije koje su predstavljene kao suma uslova kazne, svaka označava udeo određenog ograničenja. Čak i neka lakša ograničenja moraju biti zadovoljena i uzeta u obzir prilikom pravljenja rasporeda.

Sledeća stvar na koju se mora obratiti pažnja je genetski operator koji mora biti uključen u GA, u svrhu da dostigne maksimalnu optimizaciju performansi. Prvo razmatramo standardne kao i kombinatorne operatore za opštu svrhu. Standardni GA koristi populaciju 50 rešenja, standardni 5 - tačka krosover operator, operator mutacije bitova sa verovatnoćom 0.001 po bitu, elitizam, zamenu celokupne generacije roditelja sa offspring (o.f.), fitness (f.f) skaliranje i generacijski limit na 5000.

Genetska popravka menja neispravan u ispravan raspored menjajući ga što je više moguće. Rešavamo nepravilnosti tako što filtriramo i sređujemo kaznene funkcije jer tako algoritam može da se kreće na širem području pretrage. Potreba za razlikovanjem o.f. i f.f. potiče od toga što se o.f. odnosi na problem minimizacije costa, dok GA sadrži strukturu koja rešava problem maksimizacije (f. f.).

Prenos sa o.f . na f.f. je urađeno mapirajući numeričke vrednosti bivših na one kasnijih, kao monotono rastuća funkcija. Sistem je zasnovan na linearnoj dinamičkoj fitness proceduri skaliranja. U svakoj generaciji izračunava se max i min o.f. vrednosti jedinki populacije. Definišu se intervali na o.f. osi a ona se linearno prenosi na f.f. osu limitirana sa dve konstante MINFIT i MAXFIT gde min o.f. odgovara MAXFIT i obrnuto. Ova procedura sadrži 2 zadatka: minimizuje o.f. dok maksimizuje f.f. i diskriminiše rešenja jedinki koje imaju malu promenu o.f. vrednosti, što je često slučaj u kasnoj fazi procesa.

O.f. našeg problema računa generalizovan cost koji predstavlja razliku između datog i idealnog rasporeda. o.f. za raspored R je

$$z(\mathbf{R}) = \alpha \cdot \#inf + \beta_1 \cdot s_\Delta + \beta_2 \cdot s_\Omega + \beta_3 \cdot s_\Pi$$

gde je

#inf je broj nepravilnosti u matrici R

s_Δ meri nivo nezadovoljenja didaktičkoh zahteva matrice R

s_Ω meri nivo nezadovoljenja organizacionih zahteva u matrici R

s_Π meri nivo neispunjenja nastavničkih zahteva u matrici R

α, β_1, β_2, i β_3 su vrednosti izabrane od korisnika- težine.

Tri metoda određivanja težina su izabrana:

1. Sve težine imaju vrednost 1;
2. Slaba ograničenja imaju vrednost 0, jaka imaju težine koje računaju broj jakih;
3. Automatski računa na osnovu pojavljivanja ograničenja u raporedu, tako da je skup rešenja napravljen i ono ograničenje sa najmanje pojavljivanja ima vrednost 1, a ostale imaju proporcionalno više vrednosti.

Birajući da je $\alpha >> \beta_1, \beta_2, \beta_3$,mi uvodimo hijerarhijsku strukturu u o.f. tako da ističemo važnost određenih grupa problema. Kod nas je sledeća struktura izabrana:

Nivo 1 uslovi zadovoljenja σ;

Nivo 2, uslovi menadžmenta (Δ, Ω, Π);

Nivo 3, uslovi jednog profesora.

Nivo 1- rešavamo nemoguće problem - dva nastavnika u isto vreme isti čas i nepokrivene časove za razred - vreme kada razred nema čas.

Nivo 2

1. Δ - - didaktički zahtevi
   - ne više od 4 časa za nastavnika($\Delta 1$);
   - da ne bude isti nastavnik uvek na poslednjem času ($\Delta 2$);
   - uniformna distribucija sati istog predmeta tokom nedelje ($\Delta 3$);
   - parovi časova blok nastave ($\Delta 4$);
2. Ω-organizacioni zahtevi
   - što više nastavnik - roditelj sastanaka u istom danu ($\Omega 1$);
   - ne manje od 2 časa dnevno za svakog nastavnika ($\Omega 2$);
   - što manje rupa za nastavnika ($\Omega 3$);
3. zahtevi nastavnikaΠ

Težina je izabrana, za ceo skup različitih nastavnika sa poštovanjem ostalih zahteva na nivou 2; štaviše, definiše, se rang profesora po starosti i iskustvu.

Na *nivou 3* razmatramo želje nastavnika za njihov raspored

Svaki nastavnik kaže svoje želje i one se normalizuju tako da svaki nastavnik uzme udeo u određivanju rasporeda i želja i ostalih nastavnika. Konačno, dodajemo dodatnu procedure za računanje f.f. Najčešće se raspored pravi tako što se prosto zamene dva časa. Samo one komponente koje utiču na rad se koriste.

FF je kompleksnost izračunavanja za f.f, a GR kompleksnost genetske popravke.

Reprodukcija je klasični operator reprodukcije koji ističe jedinke sa nadprosečnim vrednostima f.f. Svakoj jedinki h daje verovatnoću reprodukcije pr(h) jednaka količniku fitnesa h kroz fitnesa svih jedinki. Nove jedinke se formiraju koristeće ove verovatnoće i Monte Karlo tehniku.

Krosingover - ovaj operator ima zadatak da efikasno rekombinuje blokove gradnje tako da od dva roditelja stvori dva offspring-a (nove jedinke) sa boljim rezultatima f.f. Lokalna f.f. ili l.f.f. se odnosi na karakteristike svakog nastavnika. Ako imamo dve jedinke R_1 i R_2 red R_1 je sortiran po opadajućoj l.f.f i najbolji k_1 red je uzet kao blok za izgradnju.Onda preostalih m-k_1 gde je m broj nastavnika su uzeti iz R_2 da naprave prvog sina. Drugi sin je od neiskorišćenih redova iz R_1 i R_2. Vrednost k_1 je izračunatio na osnovu l.f.f. oba roditelja. Ovaj operator se pojavljuje u verovatnoći za svaki izabrani par mogućih roditelja a parameter pojavljivanja je sistemski parameter pc.

Algoritam:

par slučajno izabranih jedinki

for za svaki par jedinki **do**

sa verovatnoćom pc **dobegin** {pc je kontrolni parametar}

izračunaj *l.f.f.* redova 2 jedinke;

sortiraj po opadajućoj vrednosti *l.f.f.* redove 2 jedinke;

napravi 2 sina spajajući 2 jedinke

{prvi sin uzima najbolje osobine prvog I najgore drugog, a drugi ono što je ostalo};

end;

7.2.1. Evaluation Function- Funkcija Izračunavanja

Funkcija izračunavanja je zasnovana na kazni. Kazna za genotip g se izračunava gde je t broj perioda, c je broj tipova ograničenja , w_j je važnost svakog ograničenja - njena težina, a n_{ij} je faktor određen kaznenim metodom. Četiri različita metoda su razmatrana:

1. Raspored se kažnjava svaki put kad se ograničenje naruši pa je n_{ij} 0 ili 1;
2. Raspored se kažnjava svaki put kad se delimični tip ograničenja naruši;
3. Kao 1, samo što se kazna duplira ako se određeni tip ograničenja naruši;
4. Binarno - ako nema nezadovoljenih ograničenja, onda je 0, 1 inače.

Vrednost ove funkcije za populaciju g se izračunava tako što delimo najnižu vrednost kazne u populaciji sa kaznom za g. Kada stvorimo početnu populaciju EA počinje da radi. Stvaranje

nove populacije je zasnovano na genetskom ruletu, a 20% je isto kao i starije generacije, 10% sadrži najbolja rešenja, a 10% su udaljeni od ostatka populacije da bi sačuvali raznolikost populacije.

7.2.2. Faza TS-a

Faza Tabu Search - Preliminarni rezultati su pokazali interesantan fenomen koji se pojavljuje na pola rada. Ako je prosečna vrednost kaznene funkcije za određeno rešenje nađena, možemo primetiti pažljivo opadanje dok ne dođe do određenog nivoa populacije, gde vrednost oscilira neznatno za 1%. Ovaj fenomen pokazuje da iako se koriste direktni genetski operatori, i dalje se u nekom delu prostor pretrage ispituje nasumično. Zato može da se zaglavi u lokalnom minimumu i da se izvuče iz njega samo nasumičnom mutacijom. Da bi ubrzali postupak i izbegli oscilacije, uvodimo TS. Ako za 50 generacija prosečna kazna za jedinke odstupa manje od 20%, staje genetski rulet i počinje TS da radi. Tabu lista je dužine 10·k. Algoritam je sledeći:

1. Nađi mesto u rešenju koje krši najviše ograničenja
2. Generiši k rešenja sa prebačenim događajima za druge izvore i periode
3. Izaberi rešenje koje ima najniži rezultat kazne i nije na listi, dodaj ga na listu i idi na 1.
   Izabrano rešenje je sada trenutno. TS radi sa 50 iteracija, posle toga opet nastavlja EA.

EA je dobar optimizacioni model sa velikom fleksibilnošću i jednostavnošću u postavci. A lako i rešava probleme rasporeda časova.

8. Bin packing

Definicija: Bin Packing (Garey i Johnson, 1979)[7]. Problem je četvoro-elementni skup <U,s,B,K> gde: U je skup predmeta, s je vektor veličine |U| , s(u) je pozitivan broj - veličina predmeta u, B je pozitivan broj - kapacitet bina i K je pozitivan broj.

Zadatak Binpacking-a je proveriti da li postoji podela skupa U na disjunktne skupove $U_1,....,U_k$ tako da je suma veličina stvari u svakom binu $Ui \leq B$.

Problem binarnog pakovanja može biti definisan i na sledeći način: Dato je n objekata da budu smešteni u binove kapaciteta L svaki. Zadatak je odrediti minimalan broj binova potrebnih da se smesti n objekata. Ovaj problem je NP kompletan ako se formuliše kao problem odlučivanja, a kao problem optimizacije je NP - težak algoritam aproksimacije. Postoje mnoge varijacije ovog problema kao što su 2D pakovanje, linearno pakovanje, pakovanje po težini, po ceni… Pošto je NP - težak, najefikasniji algoritmi za rešavanje koriste heuristike koje ne moraju da daju optimalno rešenje iako su dobre u većini slučajeva. Na primer, algoritam da se prvo popunjava prvi koji odgovara je najbrži, ali ne daje optimalno rešenje najčešće. Zahteva O (nlogn) vremena gde je n broj elemenata koji treba da budu spakovani. Algoritam može da bude efikasniji ako prvo sortiramo listu po opadajućem redosledu – poznat kao prvo popunjavajući opadajući algoritam, iako ne mora da da optimalno rešenje i za duže liste može da poveća vreme rada algoritma.

Ako imamo veličinu bina V i listu sa binovima koji treba da se spakuju sa njegovim veličinama, naći broj B i B - particiju tako da nadjemo optimalno rešenje, to jest rešenje je optimalno ako ima minimalno B. Vrednost za optimalno rešenje je označeno sa OPT. Postoje 4 metode pakovanja binova i traženja optimalnog rešenja OPT:

1. First Fit (FF) Označimo binove sa 1,2,3... Objekti se pakuju redom. Pakuj objekat i u bin j gde je j najmanji indeks tako da j može da sadrži i. Prvo u odgovarajući bin gde može da stane. Ako nigde ne može, otvara novi bin. Ovaj algoritam postiže aproksimirajući faktor 2. To je zato što u svakom datom vremenu nije moguće da 2 bina budu polupuna jer ako je jedan bin polupun i ima V/2 slobodnog prostora onda algoritam neće otvoriti novi bin za bilo koji predmet veličine V/2 najviše. Tek kad se taj bin popuni ili naiđe neki bin veće od V/2 onda se otvara novi bin. Zato imamo B-1< 2 OPT, to jest $B \leq 2$ OPT.

2. Best Fit (BF) Isto kao prethodni samo što se pakuje tamo gde najbolje pristaje i najmanje mesta ostaje. *Best fit decreasing* i *first fit decreasing su* među najprostijim algoritmima za rešavanje problema binarnog pakovanja. Koriste 11/9 **OPT** + 1 binova gde je OPT broj binova datih za optimalno rešenje. Kod sortirajućeg algoritma sortirajući korak je relatvno skup, ali bez toga stižemo da 17/10 **OPT** + 2 binova. Skoro je dokazano da je 11/9 **OPT** + 6/9 donja granica za first fit decrease.

3. First Fit Decreasing (FFD) Novi raspored (po opadajućim vrednostima) i onda prva metoda. Varijanta FFD je MFFD, koristi 71/60 **OPT** + 1 binova. Iako je ovo najčešće

[7] Vidi [15]

dovoljno dobro, efikasni algoritmi aproksimacije mogu da reše problem bez nekog fiksnog procenta.

4. Best Fit Decreasing (BFD) Novi raspored (kao u FFD) pa best fit.

Kako radi algoritam binarnog pakovanja: Prvo raspoređujemo događaje sa najviše konflikata. Oni su teži za raspoređivanje. Postoji više metoda za određivanje događaja sa najviše konflikata:

1. Najveći broj konflikata prvo - Raspoređujemo prvo one događaje sa najvećim brojem konfliktnih događaja koji su već raspoređeni. I to je mnogo teže nego da smo prvo rasporedili one koji nemaju sukobe.
2. Najmanje zadovoljenje stepena prvo: Brelaz je predložio ovo [Brelaz,79][8]: Raspoređujemo prvo one elemente sa najmanjim brojem ispravnih perioda trenutno dostupnih. Ovo će dati važnost događajima koji imaju mali broj raspoloživih perioda možda i zato što je veliki broj njihovih konflikata već raspoređen, ili je broj raspoloživih perioda limitiran na početku. U svakom slučaju ovakve događaje je teško ili nemoguće rasporediti kasnije u radu.

Problem binarnog pakovanja možemo definisati i na ovaj način: Predmete veličine s_1,……., s_n treba spakovati u najmanje moguće binova veličine 1 ako su s_i veličine iz intervala (0,1) uključujući i 1. On je NP-težak problem i možemo svesti na (0,1) interval i da su binovi veličine 1 jer sve skaliramo.

Ako imamo predmete s_1,……,s_n da li oni mogu biti podeljeni u 2 skupa iste veličine?

Postoji algoritam polinomijalnog vremena koji rešava Bin Pack gde ima najviše K različitih veličina predmeta i najviše L predmeta može stati u jedan bin, K i L su konstante.

Najprostiji algoritam je sledeći odgovarajući Next Fit:

1. Smesti predmet u bin dok sledeći predmet može da stane
2. Zatvori bin i otvori novi
3. Najviše 1 je veličina bina
4. Odnos u Next Fit je 2
5. Posmatramo binove 1 i 2
6. Prvi predmet u binu 2 nije mogao u bin 1
7. Zato je ukupna veličina u binovima 1 i 2 veća od 1
8. Treba najmanje 1 bin da ih spakuje
9. Da li je odnos u Next Fit bolji od 2?
10. Sledeći unos: N parova koji sadrže predmet veličine 1
11. Next Fit pakuje predmete u N binova
12. OPT pakuje u N+1 binova
13. Za veliko N, odnos se bliži 2.
14. Nema algoritma sa približnim odnosom 2 osim ako je P=NP.

[8] Vidi [2]

8.1. Veza sa problemom rasporeda časova

Primena heuristika na problem rasporeda

Primenom bilo kojeg od ovih algoritama bin packing-a može se rešavati problem rasporeda časova: Ako su periodi u kome se održavaju časovi binovi, onda naše časove treba smestiti u te binove tako da najmanje tih binova odnosno rasporeda bude iskorišćeno. Svi periodi će imati veličinu bina 1, a časovi će biti rangirani po tome koliko su zastupljeni u rasporedu pa će im proporcionalno u odnosu na ostale časove biti dat kapacitet bina između 0 i 1. Časovi koji se biraju mogu biti izabrani po jednom od sledeća dva kriterijuma:

- Najviše konflikata prvo;
- Najmanje zadovoljenje stepena prvo.

Kad smo na taj način izabrali časove, onda ih uz pomoć jednog od gore pomenuta četiri algoritma (FF, BF**,**FFD**,**BFD) smeštamo u binove odnosno u časove gledajući da ne dođe do konflikata i da sva ograničenja budu zadovoljena. Možda je najbolje raditi sa BFD jer on prvo rangira predmete po opadajućoj veličini binova (to jest po broju časova koje zauzimaju u rasporedu) pa ih onda smešta u raspored.

9. Bojenje grafova i problem rasporeda

Pod pojmom grafa G označavamo uređeni par G=(V,A) gde je V skup čvorova, a A skup lukova u tom grafu. Ako oznacimo m := card(A) i n := card(V) tada je graf u potpunosti određen njegovom matricom incidencije. To je matrica tipa m×n čije su kolone indeksirane vrhovima, a redovi su indeksirani lukovima. Drugim rečima, svakoj koloni matrice incidencije odgovara vrh (i obratno) i svakom njenom redu odgovara luk grafa (i obratno). Ovde pretpostavljamo da smo skup vrhova V i skup lukova A potpuno uredili. To znaci da znamo koji je element prvi, koji drugi . . . Elemente matrice incidencije odredimo ovako: Ako je A luk, tada tom luku, sada indeksu reda, pridružujemo red matrice incidencije koji na mestu v sadrži −1 ako luk izlazi iz vrha v, a na mestu w sadrži 1 ako luk ulazi u vrh w. Na preostalim mestima u redu pišemo nulu.

Definicija: Problem bojenja grafova (Golumbic, 1980)[9] - Ovde je problem određivanja broja boja za dati graf G (to jest sa koliko minimalno boja mogu biti obojeni čvorovi datog grafa).

Definicija:(Graf K-boja) (Karp, 1972).[10] Element ovog problema je par <G, K> gde je G par (V, E) gde je V skup čvorova, a E skup veza među njima, a K je pozitivan broj. Zadatak ovog problema je odlučiti da li je G K-obojiv to jest da li postoji bojenje f: V→{1,....,K} tako da je f(u) ≠f(v) kad god je {u,v} ⊂ E. Ovaj problem deli skup predmeta (grafovi su predmeti) na podskupove nezavisnih predmeta: Ponekad nemamo nezavisnih među njima, ali imamo veličinu svakog predmeta. Ako nam je dat maksimalni broj podskupova i gornje ograničenje za veličinu svakog podskupa - imamo binarno pakovanje.

9.1. Veza sa rasporedom časova:

Problem određivanja minimalnog broja perioda potrebnih da bi se rasporedili svi časovi je problem bojenja grafova. Ako imamo časove i skup mogućih sukoba i ograničenja, možemo da napravimo raspored i nađemo minimalno bojenje. Čvorovi u G predstavljaju predmete, veze predstavljaju par predmeta u sukobu, a boja predstavlja period kada je predmet raspoređen. Welsh i Powell su prvi pokazali ekvivalentnost ova 2 problema. Raspored je praktična primena bojenja grafova. Oni su NP - teški problemi. Ručno pravljen program ne može da radi brzo i napravi dobro rešenje. Algoritam radi u 5 faza. To su : unos podataka, registracija, postavljanje ograničenja, generacija ili stvaranje rasporeda i rešenje rasporeda.

U prvoj fazi unosimo podatke. U fazi ograničenja unosimo koja su ograničenja data , najviše pazimo na profesore i njihove zahteve. U fazi registracije, ubacujemo učenike u odeljenja. Algoritam daje brzo rezultate ali pouzdanost opada sa porastom ograničenja. Postavljamo ograničenja u matricu. Označavamo sukobljene predmete . Farbamo rasporede za sukobljene predmete koristeći algoritam bojenja grafova. Postavljamo predmete u raspored na osnovu boja.

[9] Vidi [11]
[10] Vidi [14]

Označavanje ograničenja na matrici predmeta je urađeno u skladu sa ograničenjima profesora i učenika i tako su otkriveni sukobljeni predmeti. Kada smo prikupili sve potrebne podatke, pravimo konfliktni graf G koji sadrži date podatke. Čvorovi su predmeti, a veze povezuju parove sukobljenih predmeta.

Imamo n predmeta $\{c_1, c_2, \ldots, c_n\}$ i svaki c_i će biti predstavljen sa tačno jednim čvorom v_i u G pa G ima n čvorova i $V(G) = \{v_1, v_2, \ldots, v_n\}$. Veze predstavljaju predmete koje mi ne možemo ili nećemo da rasporedimo u isto vreme. Dodatne veze mogu biti dodate da pokažu željene uslove rasporeda. Ako je nastavnik za predmete c_i i c_j isti, mora postojati veza između v_i i v_j. Nastavnik k časova stvara skupinu reda k u grafu jer svi oni moraju biti u različito vreme. Ako časovi c_i i c_j traže istu učionicu, onda dodajemo vezu između v_i i v_j da ne bi bili u isto vreme. Ako r predmeta traži istu učionicu, onda stvaramo skupinu reda r da bi svaki od njih bio raspoređen u različito vreme. Nekad ima više istih tipova učionica pa ih raspređujemo u njih.

Kada napravimo graf, možemo pravilno da ofarbamo vrhove i da na osnovu toga napravimo raspored. Ako ima veza između njih, onda se vrhovi boje različitim bojama. Ako nema veza, farbaju se isto ili različito. Naš generalni pristup će pratiti sekvencijalni algoritam bojenja grafova. Ovaj algoritam se još zove i pohlepni algoritam . On ispituje svaki vrh u grafu jedan po jedan prateći neki određeni red i pokušava da ih oboji sa jednom od već korišćenih boja i da doda vrh već nekoj klasi. Ako ne može, dodaje novu klasu. Algoritam pokušava da oboji grafove koristeći što manje boja. Četiri algoritma (SIMPLESEARCH GREEDY (SSG), LARGEST-FIRSTSEARCH GREEDY (LFSG), SMALLEST-FIRSTSEARCH GREEDY (SFSG), and RANDOMSEARCH GREEDY (RSG)) su implementirana sa različitim početnim raspoređivanjem čvorova grafa:

1. Specifično predefinisano raspoređivanje - SO
2. Raspoređivasnje po opadajućen stepenu - DD
3. Po najmanjem stepenu – SD
4. Nasumično – Ro

Četiri algoritma prethodno navedena sa 4 početna pristupa čine 16 različitih varijanti. Ove obojene klase su skupovi predmeta koje mogu biti raspoređene u raspored bez sukoba. Čvorovi koji odgovaraju predmetima koji ne mogu biti dodeljeni istom periodu biće drugačije obojeni. Ovde predlažemo bojenje po grupama, a postoji više načina kako možemo da grupišemo čvorove u grupe. Na različite načine grupišemo čvorove. Njih prvo rasporedimo.

SMALLEST-FIRST-SEARCH GREEDY SFSG je najbolje koristiti u radu, jer on rangira klase boja po broju čvorova u njima i pretražuje ove rangirane liste po opadajućoj veličini. Zasnovan na ovoj heuristici pretrage, SFSG pokušava da balansira veličine klasa boja.

10. Predlog rešenja problema rasporeda časova:

Sad ćemo da rasporedimo profesore po odeljenjima i tako ćemo dobiti raspored za njih kao i za druga odeljenja. Kako nikada ne znamo sve podatke, uzećemo neke opšte i pokušati da rešimo ovaj problem kao problem linearnog progarmiranja, i to simpleks metodom (ako tako nešto može da se uradi).

Na početku godine uprava škole (u ovom slučaju gimnazije) određuje koji profesori predaju kom razredu i to izgleda otprilike ovako:

Srpski: 6 profesora ukupno 96 časova nedeljno, i to :

A.A. I1-4, I2-4, I6-4, III1-5 i III6-3 časa što je ukupno 20 časova;

B.B. I3-4, I4-4, I5-4, II4-3, IV1-5 časova i ukupno 20 časova;

V.V. II1-4, II2-4, II5-3, II6-3 i III5- 3 časa i ukupno 17 časova;

G.G III2-5, III3-5, IV2-5 časova i ukupno 15 časova;

D.D. II3-4, III4-3, IV3-5 časova i ukupno 12 časova;

Đ.Đ. IV4-4, IV5-4, IV6-4 časa i ukupno 12 časova;

Engleski: 4 profesora ukupno 66 časova nedeljno, i to:

E.E. II1-3, III1-5, III2-5, III3-5 časova i ukupno 18 časova;

Ž.Ž. I4-2, I5-2,I6-2, II4-2, II5-2,II6-2, IV1-4 časa i ukupno 16 časova;

Z.Z. I1-2, III4-2, III5-2,III6-2, IV2-4, IV3-4 časa i ukupno 16 časova;

I.I. I2-2, I3-2, II2-3, II3-3, IV4-2, IV5-2, IV6-2 časa i ukupno 16 časova;

Francuski: 3 profesora ukupno 24 časa nedeljno, i to:

J.J. I2-2, I3-2, I6-2, II2-2 časa i ukupno 8 časova;

K.K. II3-2, II6-2, III2-2, III3-2 časa i ukupno 8 časova;

L.L. III6-2, IV2-2, IV3-2, IV6-2 časa i ukupno 8 časova;

Nemački: 2 profesora ukupno 24 časa nedeljno, i to:

Lj.Lj. I1-2, I4-2, I5-2, II1-2, II4-2, II5-2 časa i ukupno 12 časova;

M.M. III1-2, III4-2, III5-2, IV1-2, IV4-2, IV5-2 časa i ukupno 12 časova;

Ruski: 1 profesor ukupno 16 časova, i to:

N.N. I3-2, I6-2, II3-2, II6-2, III3-2, III6-2, IV3-2, IV6-2 časa i ukupno 16 časova;

Za strane jezike pretpostavljamo da su nemački i francuski drugi jezici, ruski je prvi, a engleski je najčešće prvi osim kada je u paru sa ruskim i trebalo bi da prvi i drugi jezici idu u paru, naravno ako je to moguće izvesti;

Latinski: 2 profesora ukupno 18 časova, i to:

Nj.Nj. I1-2, I2-2, I3-2, I4-2, I5-2 časa i ukupno 10 časova;

O.O. I6-2, II1-2, II2-2, II3-2 časa i ukupno 8 časova;

Ustav: 1 profesor ukupno 6 časova, i to:

P.P. IV1-1, IV2-1, IV3-1, IV4-1, IV5-1, IV6-1 čas i ukupno 6 časova;

Sociologija: 2 profesora ukupno 15 časova, i to:

R.R. IV1-3, IV2-3, IV4-2 časa i ukupno 8 časova;

S.S. IV3-3, IV5-2, IV6-2 časa i ukupno 7 časova;

Psihologija: 1 profesor ukupno 12 časova, i to:

T.T. II1-2, II2-2, II3-2, II4-2, II5-2, II6-2 časa i ukupno 12 časova;

Filozofija: 2 profesora ukupno 27 časova, i to:

Ć.Ć. III1-2, III2-2, III3-2, III4-2, III5-2, III6-2, IV1-3 časa i ukupno 15 časova;

U.U. IV2-3, IV3-3, IV4-2, IV5-2, IV6-2 časa i ukupno 12 časova;

Istorija: 3 profesora ukupno 48 časova, i to:

F.F. I1-2, I2-2, I3-2, I4-2, III1-3, III2-3, III4-2 časa i ukupno 16 časova;

H.H. I5-2, I6-2, II1-2, II2-2, II3-2, III3-3, IV1-3 časa i ukupno 16 časova;

C.C. II4-2, II5-2, II6-2, III5-2, III6-2, IV2-3, IV3-3 časa i ukupno 16 časova;

Geografija: 2 profesora ukupno 36 časova, i to:

Č.Č. I1-2, I2-2, I3-2, I4-2, I5-2, I6-2, II1-2, II2-2, II3-2 časa i ukupno 18 časova;

Dž.Dž. II4-2, II5-2, II6-2, III1-2, III2-2, III3-2, III4-2, III5-2, III6-2 časa i ukupno 18;

Biologija: 3 profesora ukupno 48 časova, i to:

Š.Š. I4-2, I5-2, III4-3, III5-3, III6-3, IV4-3 časa i ukupno 16 časova;

A.B. I1-2, I2-2, I3-2, I6-2, III1-2, IV5-3,IV6-3 časa i ukupno 16 časova;

A.V. II1-2, II2-2, II3-2, II4-2, II5-2, II6-2, III2-2, III3-2 časa i ukupno 16 časova;

Matematika: 5 profesora ukupno 87 časova, i to:

A.G. I4-4, I5-4, II4-5, II-5, IV1-2 časa i ukupno 20 časova;

A.D. I1-4, I6-4, II6-5, III4-5, IV2-2 časa i ukupno 20 časova;

A.Đ. II1-3, II2-3, III5-5, III6-5, IV3- 2 časa i ukupno 18 časova;

A.E. I2-4, I3-4, II3-3, III1-2, III2-2, III3-2 časa i ukupno 17 časova;

A.Ž. IV4-4, IV5-4, IV6-4 časa i ukupno 12 časova;

Fizika sa astronomijom u 4-oj godini: 4 profesora ukupno 63 časa, i to :

A.Z. I6-2, III6-3, IV4-5, IV5-5, IV6- 5 časova i ukupno 20 časova;

A.I. I4-2, I5-2, II4-3, II5-3, II6-3, III4-3, III5- 3 časa i ukupno 19 časova;

A.J. I1-2, I2-2, I3-2, II1-2, II2-2, II3-2 časa i ukupno 12 časova;

A.K. III1-2, III2-2, III3-2, IV1-2,IV2-2,IV3-2 časa i ukupno 12 časova;

Hemija: 3 profesora ukupno 42 časa, i to:

A.L. I4-2, I5-2, II4-3, II5-3, II6-3, III4- 3 časa i ukupno 16 časova;

A.Lj. I6-2, III5-3, III6-3, IV4-2, IV5-2, IV6-2 časa i ukupno 14 časova;

A.M. I1-2, I2-2, I3-2, II1-2, II2-2, II3-2 časa i ukupno 12 časova;

Informatika: 3 profesora ukupno 36 časova, i to:

A.N. I4-2, I5-2, I6-2, II4-2, II5-2, II6-2 časa i ukupno 12 časova;

A.Nj. I1-2, I2-2, I3-2, II1-2, II2-2, II3-2 časa i ukupno 12 časova;

A.O. III1-1, III2-1, III3-1, III4-1, III5-1, III6-1, IV1-1, IV2-1, IV3-1, IV4-1, IV5-1,IV6-1 čas i ukupno 12 časova;

Muzičko: 1 profesor ukupno 18 časova, i to:

A.P. I1-1, I2-1, I3-1, I4-1, I5-1, I6-1, II1-1, II2-1, II3-1, II4-1, II5-1, II6-1, III1-1, III2-1, III3-1, IV1-1, IV2-1, IV3-1 čas i ukupno 18 časova;

Likovno : 1 profesor ukupno 18 časova, i to:

A.R. I1-1, I2-1, I3-1, I4-1, I5-1, I6-1, II1-1, II2-1, II3-1, II4-1, II5-1, II6-1, III1-1, III2-1, III3-1, IV1-1, IV2-1, IV3-1 čas i ukupno 18 časova;

Fizičko: 3 profesora ukupno 48 časova, i to :

A.S. I1-2, I2-2, I3-2, I4-2, I5-2, I6-2, II1-2, II2-2 časa i ukupno 16 časova;

A.T. II3-2, II4-2, II5-2, II6-2, III1-2, III2-2, III3-2, III4-2 časa i ukupno 16 časova;

A.Ć. III5-2, III6-2, IV1-2, IV2-2, IV3-2, IV4-2, IV5-2, IV6-2 časa i ukupno 16 časova;

Veronauka i Građansko vaspitanje : jedan profesor po svakom predmetu; to su izborni predmeti; održavaju se na kraju natave; učenici sve 4 godine imaju ili jedan ili drugi predmet i svi moraju da prisustvuju ili jednom ili drugom; 1 čas nedeljno.

Dakle, kad nam je sve ovo poznato možemo da napravimo matricu u koju ćemo smestiti profesore i brojeve časova koje drže određenim odeljenjima:

	I 1	I 2	I 3	I 4	I 5	I 6	II 1	II 2	II 3	II 4	II 5	II 6	III 1	III 2	III 3	III 4	III 5	III 6	IV 1	IV 2	IV 3	IV 4	IV 5	IV 6
A.A.	4	4	0	0	0	4	0	0	0	0	0	0	5	0	0	0	0	3	0	0	0	0	0	0
B.B.	0	0	4	4	4	0	0	0	0	3	0	0	0	0	0	0	0	0	5	0	0	0	0	0
V.V.	0	0	0	0	0	0	4	4	0	0	3	3	0	0	0	0	3	0	0	0	0	0	0	0
G.G	0	0	0	0	0	0	0	0	0	0	0	0	0	5	5	0	0	0	0	5	0	0	0	0
D.D	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	3	0	0	0	0	5	0	0	0
Đ.Đ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4
E.E.	0	0	0	0	0	0	3	0	0	0	0	0	5	5	5	0	0	0	0	0	0	0	0	0
Ž.Ž.	0	0	0	2	2	2	0	0	0	2	2	2	0	0	0	0	0	0	4	0	0	0	0	0
Z.Z.	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	0	4	4	0	0	0
I.I.	0	2	2	0	0	0	0	3	3	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2
J.J.	0	2	2	0	0	2	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K.K.	0	0	0	0	0	0	0	0	2	0	0	2	0	2	2	0	0	0	0	0	0	0	0	0
L.L.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	2	2	0	0	2
Lj.Lj	2	0	0	2	2	0	2	0	0	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0
MM	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	2	2	0	2	0	0	2	2	0
N.N	0	0	2	0	0	2	0	0	2	0	0	2	0	0	2	0	0	2	0	0	2	0	0	2
NjNj	2	2	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
O.O	0	0	0	0	0	2	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P.P.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
R.R.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3	0	2	0	0
S.S.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	2	2
T.T.	0	0	0	0	0	0	2	2	2	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0
Ć.Ć.	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	3	0	0	0	0	0
U.U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3	2	2	2
F.F.	2	2	2	2	0	0	0	0	0	0	0	0	3	3	0	2	0	0	0	0	0	0	0	0
H.H	0	0	0	0	2	2	2	2	2	0	0	0	0	0	3	0	0	0	3	0	0	0	0	0
C.C.	0	0	0	0	0	0	0	0	0	2	2	2	0	0	0	0	2	2	0	3	3	0	0	0
Č.Č.	2	2	2	2	2	2	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DžDž	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	0	0	0	0	0	0
Š.Š.	0	0	0	2	2	0	0	0	0	0	0	0	0	0	0	3	3	3	0	0	0	3	0	0

A.B.	2	2	2	0	0	2	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	3	3
A.V.	0	0	0	0	0	0	2	2	2	2	2	2	0	2	2	0	0	0	0	0	0	0	0	0
A.G.	0	0	0	4	4	0	0	0	0	5	5	0	0	0	0	0	0	0	2	0	0	0	0	0
A.D.	4	0	0	0	0	4	0	0	0	0	0	5	0	0	0	5	0	0	0	2	0	0	0	0
A.Đ.	0	0	0	0	0	0	3	3	0	0	0	0	0	0	0	0	5	5	0	0	2	0	0	0
A.E.	0	4	4	0	0	0	0	0	3	0	0	0	2	2	2	0	0	0	0	0	0	0	0	0
A.Ž.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4
A.Z.	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	5	5	5
A.I.	0	0	0	2	2	0	0	0	0	3	3	3	0	0	0	3	3	0	0	0	0	0	0	0
A.J.	2	2	2	0	0	0	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A.K.	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	0	0	0	2	2	2	0	0	0
A.L.	0	0	0	2	2	0	0	0	0	3	3	3	0	0	0	3	0	0	0	0	0	0	0	0
A.Lj	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	3	3	0	0	0	2	2	2
A.M	2	2	2	0	0	0	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A.N.	0	0	0	2	2	2	0	0	0	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0
A.Nj	2	2	2	0	0	0	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A.O	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
A.P.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	0	0	0
A.R.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	0	0	0
A.S.	2	2	2	2	2	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A.T.	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	0	0	0	0	0	0	0	0
A.Ć.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2
A.U.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
A.F.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Sada ćemo pokušati da napravimo raspored časova za nastavnike u kojima ćemo njih označavati brojevima od 1 do 54 umesto slovima A.A. do A.C. radi lakših oznaka,a stavićemo da su I i III godina prepodne,a II i IV popodne.Krenućemo od prvog nastavnika i raspoređivaćemo časove redom do poslednjeg. Dakle:

	Ponedeljak							Utorak							Sreda							Četvrtak							Petak						
	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7
1			A	Đ	Đ	L	L			B	B	Đ	L	O				A	B							O	L	L		O	Đ	A	A	B	
2	I		V	G	G		D			P	P					I		V	V	G	D					P	P		I	P		V	G	D	D
3	J	K	E	E			Nj		Ž	Ž					J	K	E	Ž		Nj									E	K	J	Ž			Nj
4	R			M	M	Lj	Lj								R	R			Lj	Lj	M								R	R			Lj	M	M
5								Z	S	S			N									Z	Z	S			N		Z	S	S				N
6									T	T	Č	U											Ć	Ć	U	T				U	U	Ć	T		
7	E			M	Lj	L	L				M	L	Lj	Lj	E					L	Lj								E			M	M	Lj	L
8	I	K				D	G	J	P	P			Đ		K	I			D	G		J	P	P			Đ								
9	R	S				N	A	R	S				Nj	O	S	R				N	A	S	R				O	Nj							
10	Ž	Z	T	U		B	Z								Z	Ž	Ć			V	B								Ž	Z	T	Ć	U		
1	Ž				Đ	V	B								Ž				B	Đ	V														

1																																	
2	Z	K				Lj	M								K	Z				Lj	M												
3								R	U	S				O								U	S	R			O						
4	I	J				A	G	E					D			I	J			G	A	E					D						
5	P	Ć					Nj	T					L	N	P	Ć				Nj			T				N	L					
6	Z	K				V	Đ	S	U				M	O	K	Z				Đ	V	U	S				O	M					
7			A	B	V	D	G										G	D	A	B	V												
8								E	Z	Ž				Đ								Z	Ž	E				Đ					
9									P	R	T	U	Ć	S																			
10											P	P	R	T											T	R	R	P					
11											S	S	Ć	U												S	Ć	U					
12				I	Z	Ž	E										E	Z	K	J					Z	I	J	K					
13		P				LJ	M				LJ	O	NJ	L											L	NJ	O	LJ	P			M	LJ
14				T	R	S	Ć									U	R	S	T										S	R	Ć	U	
15			A	LJ	L	G			B	V						G	A	LJ	N	L								B	LJ	N	L	V	
16	P	E			M	D	Đ								P	Ž	Z		D	M	Đ						P	E	Ž	Z			M
17				J	R	S	K	I	R	S			NJ	O	J	K						I	R	S			O	NJ					
18	Ž	Z	E			D	Đ	A	G	V	B				E	Z	Ž			Đ	D				V	G	A	B					
19								I	K	J			L	M			O	LJ	N	NJ		J	I				M	L			LJ	NJ	N	O	
20	T			G	N	NJ	O									T	G	D	O	NJ	N							T		D	N	O	NJ	
21				L	V	B	Đ	Ć	T	U				A											B	L	Đ	V	U	U	Ć				A
22		Ž	I	K	J			E	Z				LJ	M			J	K	I	Ž		E	Z			M	LJ							
23	P	I	I			G	G	J	J			G	D	D	P	I	I	J			D								G	I	J	J		D
24	K	R			Đ	A	A								K	K			A	N	N		R		Đ	Đ	A	N			K	K	Đ	N	N
25								E	Ž	Ž	S	O					NJ	NJ	O	O		Ž	E	E	S	NJ					NJ	NJ	O	O	
26		Z	Z		LJ	L	M		B	B	V	V				Z			LJ	L	M									B	B	V	V	
27				T	T	Ć	Č										Ć	Ć	U	U									U	U	T	T		
28		T	Ć	Ć	U	U		T	T			O	O	Đ		U	T	T	Ć	Ć		U	U			Đ	O	O						
29			I	NJ	NJ	G	D			I	I	J	K	K								K	K	J			G	D	I			NJ	N	N
30	Z	E	Ž		V	B	A								E	Ž	Z		V	B	A												
31								R	S				L	LJ								S	R	P			M		P			L	LJ	M

42	I				N	G	D			J	K	I			J	I				G	N		I	J	K		N	D							
43	U	T	Ć			NJ	O								T				O	NJ	Đ							Ć	U			O	NJ	Đ
44	E	Z				V	A								Ž	E				B	V								Ž	Z				B	A
45									G	D	Đ						I	J	K			K				G	D	Đ		J	I				
46	Ž	Z			V	A		E			A	B	V								E	Ž	Z			B								
47											L	Lj	M	N								P	R				Nj	O		S	T	Ć	U		
48															P	R	S		L	LJ	M	E	Ž	Z	I	J	K		A	B	V	G	D	Đ	
49	A	B	V	G	D	Đ		E	Ž	Z	I	J	K		R	S	P		Lj	M	L														
50				Đ	D	A	B						V	G	E	Ž				A	B						V	G	E	Ž			D	Đ	
51	Z	I	J	K							L	Lj	M	N									M	N	Lj	L			Z	J	K	I			
52											S	T	Ć	U	P	R				O	Nj				S	T	Ć	U	P	R			O	Nj
53	V	E	R	O	N	A	U	K	A																									
54	G	R	A	Đ	A	N	S	K	O	V	A	S	P	I	T	A	N	J	E															

Naravno, ovde smo zbog skraćenog zapisa numerisali profesore brojevima od 1-54 tako da je broj 1 A.A. a broj 54 je A.F. dok smo razredima dodelili brojevne vrednosti tako da je I1 A,I2 B,I3 V...., a IV6 je U da bi nam raspored bio kompletan. Kad smo ovo uradili, dobili smo kompletan raspored za profesore, odeljenja i učionice, čije smo delimične rasporede prikazali kako izgledaju u postavljanju problema rasporeda časova. Sad kad imamo napravljen početni raspored časova možemo da iskoristimo neke od gore pomenutih metoda i heuristika za rešavanje problema:

Konkretno, na ovaj primer bi mogli da primenimo baš ovaj poslednji metod - ručno pravljenje časova, traženje grešaka i vraćanje nazad dok ne napravimo zadovoljavajući raspored. Pošto je ovo škola srednje veličine, neće biti toliko problema i raspored će se vrlo brzo napraviti. Ali, šta ako je škola veća - sa 10 odeljenja po godini i oko 100 profesora? Kako to sve uklopiti?

10.1.Primena metode EA/GA na problem rasporeda časova:

Na kraju ćemo pokušati da primenimo jednu od heuristika – EA/GA na dobijanje rešenja rasporeda časova kako je to predstavljeno u poglavlju posvećenom ovoj heuristici:
Još jednom se vraćamo na našu školu i krećemo iz početka:
Razredi i broj časova su raspoređeni po profesorima i pravljenje rasporeda može da krene;
Krećemo od prvog profesora na listi - to je A.A. koji ima 20 časova nedeljno. Kodiramo njegove časove (a taj postupak možemo i da izbegnemo) i rasporedimo ih po danima i

odeljenjima: ponedeljak, sreda, četvrtak i petak po 5 časova svaki dan tako da ima i jedan dan slobodan, a i da mu časovi budu pravilno raspoređeni, zatim ide B.B.... i tako dobijamo sledeću tabelu :

	Ponedeljak	Utorak	Sreda	Četvrtak	Petak
A.A.	5	5	0	5	5
B.B.	5	0	5	4	6
V.V.	5	2	5	0	5
G.G	5	0	5	0	5
D.D	0	4	0	4	4
Đ.Đ	0	4	0	4	4
E.E.	5	4	4	0	5
Ž.Ž.	4	4	4	4	0
Z.Z.	4	4	4	4	0
I.I.	6	0	5	0	5
J.J.	4	0	4	0	0
K.K.	4	0	4	0	0
L.L.	0	4	0	4	0
Lj.Lj	4	2	4	2	0
MM	3	3	3	3	0
N.N	4	4	4	4	0
NjNj	5	0	5	0	0
O.O	0	4	0	4	0
P.P.	0	6	0	0	0
R.R.	0	4	0	4	0
S.S.	0	4	0	4	0
T.T.	4	0	4	4	0
Ć.Ć.	4	4	0	4	3
U.U	4	0	4	0	4
F.F.	4	2	5	0	5
H.H	5	0	6	0	5
C.C.	4	5	2	5	0
Č.Č.	5	4	5	4	0
DžDž	0	5	4	5	4
Š.Š.	5	0	6	0	5
A.B.	4	4	0	4	4
A.V.	4	4	4	4	0
A.G.	5	5	5	0	5
A.D.	5	0	5	5	5
A.Đ.	0	5	4	5	4
A.E.	5	4	4	0	4
A.Ž.	4	0	4	0	4
A.Z.	5	5	5	5	0
A.I.	5	5	0	5	4

A.J.	6	0	6	0	0
A.K.	0	4	0	4	4
A.L.	4	3	4	5	0
A.Lj	5	0	4	0	5
A.M	4	0	4	0	4
A.N.	0	3	3	4	2
A.Nj	4	4	0	4	0
A.O	0	4	0	4	4
A.P.	0	0	6	6	6
A.R.	6	6	6	0	0
A.S.	4	2	4	2	4
A.T.	4	4	0	4	4
A.Ć.	0	4	4	4	4
A.U.	Veronauka				
A.F.	Građansko				

Odredili smo skupine dana i brojeve časova koje će profesor u njima držati. Što se tiče prioriteta, važno je pomenuti da najveći prioritet imaju časovi srpskog i matematike i njih pokušavamo da prve rasporedimo u raspored i to tako da bude u sredini rasporeda kada đaci imaju najveću pažnju. Posle njih po važnosti dolaze fizika, hemija, biologija, istorija, geografija, a na samom kraju po važnosti su fizičko, muzičko i likovno kao i izborni predmeti- veronauka i građansko. Kad smo to uradili, pravimo početni raspored časova:

	Ponedeljak							Utorak							Sreda							Četvrtak							Petak						
	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7
1			A	Đ	Đ	L	L			B	B	Đ	L	O										A	B	O	L	L		O	Đ	A	A	B	
2	I	P			V	D	D	P	P		G	G	V			I	P		V	G	D								I		P	V	G	D	
3	J	K	E	E			Nj		Ž	Ž					J	K	E	Ž		Nj									E	K	J	Ž			Nj
4	R			M	M	Lj	Lj								R	R			Lj	Lj	M								R	R			Lj	M	M
5								Z	S	S			N									Z	Z	S			N		Z	S	S				N
6									T	T	Č	U											Ć	Ć	U	T				U	U	Ć	T		
7	E			M	Lj	L	L				M	L	Lj	Lj	E					L	Lj								E			M	M	Lj	L
8	I	K				D	G	J	P	P			Đ		K	I			D	G		J	P	P			Đ								
9	R	S				N	A	R	S				Nj	O	S	R				N	A	S	R				O	Nj							
10	Ž	Z	T	U		B	Z								Z	Ž	Ć			V	B								Ž	Z	T	Ć	U		
11	Ž				Đ	V	B								Ž				B	Đ	V														
12	Z	K				Lj	M								K	Z				Lj	M														
13								R	U	S				O								U	S	R			O								
14	I	J				A	G	E					D			I	J			G	A	E					D								
15	P	Ć					Nj	T					L	N	P	Ć				Nj			T				N	L							
16	Z	K				V	Đ	S	U				M	O	K	Z				Đ	V	U	S				O	M							
17			A	B	V	D	G										G	D	A	B	V														
18								E	Z	Ž				Đ								Z	Ž	E				Đ							
19									P	R	T	U	Ć	S																					
20											P	P	R	T											T	R	R	P							
21											S	S	Ć	U												S	Ć	U							
22				I	Z	Ž	E										E	Z	K	J					Z	I	J	K							
23		P				LJ	M				LJ	O	NJ	L											L	NJ	O	LJ	P					M	LJ
24				T	R	S	Ć									U	R	S	T												S	R	Ć	U	
25			A	LJ	L	G			B	V						G	A	LJ	N	L										B	LJ	N	L	V	
26	P	E			M	D	Đ								P	Ž	Z		D	M	Đ								P	E	Ž	Z			M
27				J	R	S	K	I	R	S			NJ	O	J	K						I	R	S			O	NJ							
28	Ž	Z	E			D	Đ	A	G	V	B				E	Z	Ž			Đ	D				V	G	A	B							
29								I	K	J			L	M			O	LJ	N	NJ		J	I				M	L			LJ	NJ	N	O	
30	T			G	N	NJ	O									T	G	D	O	NJ	N								T		D	N	O	NJ	
31				L	V	B	Đ	Ć	T	U				A											B	L	Đ	V	U	U	Ć				A
32		Ž	I	K	J			E	Z				LJ	M			J	K	I	Ž		E	Z			M	LJ								
33	P	I	I			G	G	J	J			G	D	D	P	I	I	J			D									G	I	J	J		D

34	K	R			Đ	A	A								K	K			A	N	N		R		Đ	Đ	A	N			K	K	Đ	N	N
35								E	Ž	Ž	S	O					NJ	NJ	O	O		Ž	E	E	S	NJ					NJ	NJ	O	O	
36		Z	Z		LJ	L	M		B	B	V	V				Z			LJ	L	M									B	B	V	V	
37				T	T	Ć	Č										Ć	Ć	U	U										U	U	T	T		
38		T	Ć	Ć	U	U		T	T			O	O	Đ		U	T	T	Ć	Ć		U	U			Đ	O	O							
39			I	NJ	NJ	G	D			I	I	J	K	K								K	K	J			G	D	I				NJ	N	N
40	Z	E	Ž		V	B	A								E	Ž	Z		V	B	A														
41								R	S				L	LJ								S	R	P			M		P				L	LJ	M
42	I				N	G	D			J	K	I			J	I				G	N		I	J	K		N	D							
43	U	T	Ć			NJ	O								T				O	NJ	Đ							Ć	U			O	NJ	Đ
44	E	Z				V	A								Ž	E				B	V							Ž	Z				B	A
45									G	D	Đ						I	J	K			K			G	D	Đ		J	I					
46	Ž	Z			V	A		E			A	B	V								E	Ž	Z			B									
47											L	Lj	M	N								P	R				Nj	O		S	T	Ć	U		
48															P	R	S		L	LJ	M	E	Ž	Z	I	J	K		A	B	V	G	D	Đ	
49	A	B	V	G	D	Đ		E	Ž	Z	I	J	K		R	S	P		Lj	M	L														
50				Đ	D	A	B						V	G	E	Ž				A	B						V	G	E	Ž			D	Đ	
51	Z	I	J	K							L	Lj	M	N									M	N	Lj	L			Z	J	K	I			
52											S	T	Ć	U	P	R				O	Nj				S	T	Ć	U	P	R				O	Nj
53	V	E	R	O	N	A	U	K	A																										
54	G	R	A	Đ	A	N	S	K	O	V	A	S	P	I	T	A	N	J	E																

Posle ovog koraka dolazi korak kada ispravljamo greške koliko možemo, tako da novi raspored dobija sledeći oblik:

	Ponedeljak							Utorak							Sreda							Četvrtak							Petak						
	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7
1			A	Đ	Đ	L	L			B	B	Đ	L	O										A	B	O	L	L		O	Đ	A	A	B	
2	I	P			V	D	D	P	P		G	G	V			I	P		V	G	D								I		P	V	G	D	
3	J	K	E	E			Nj		Ž	Ž					J	K	E	Ž		Nj									E	K	J	Ž			Nj
4	R			M	M	Lj	Lj								R	R			Lj	Lj	M								R	R			Lj	M	M
5								Z	S	S			N									Z	Z	S			N		Z	S	S				N
6									T	T	Č	U											Ć	Ć	U	T				U	U	Ć	T		
7	E			M	Lj	L	L				M	L	Lj	Lj	E					L	Lj								E			M	M	Lj	L
8		K	I			D	G	J	P	P			Đ		K	I			D	G		J	P	P			Đ								
9	R	S				N	A	R	S				Nj	O	S	R				N	A	S	R				O	Nj							
10	Ž	Z	T	U		B	Z								Z	Ž	Ć			V	B								Ž	Z	T	Ć	U		
11		Ž			Đ	V	B								Ž				B	Đ	V														
12	K	Z				Lj	M								K	Z				Lj	M														
13								R	U	S				O								U	S	R			O								
14	J	I				A	G	E					D			I	J			G	A	E					D								
15	P	Ć					Nj	T					L	N	P	Ć				Nj			T				N	L							
16	Z	K				V	Đ	S	U				M	O	K	Z				Đ	V	U	S				O	M							
17			A	B	V	D	G										G	D	A	B	V														
18								E	Z	Ž				Đ								Z	Ž	E				Đ							
19									P	R	T	U	Ć	S																					
20											P	P	R	T											T	R	R	P							
21											S	S	Ć	U												S	Ć	U							
22				I	Z	Ž	E										E	Z	K	J					Z	I	J	K							
23		P				LJ	M				LJ	O	NJ	L											L	NJ	O	LJ	P				M	LJ	
24				T	R	S	Ć									U	R	S	T												S	R	Ć	U	
25			A	LJ	L	G			B	V						G	A	LJ	N	L										B	LJ	N	L	V	
26		E	P		M	D	Đ								P	Ž	Z		D	M	Đ								P	E	Ž	Z			M
27				J	R	S	K	I	R	S			NJ	O	J	K						I	R	S			O	NJ							
28			E	Ž	Z	D	Đ	A	G	V	B				E	Z	Ž			Đ	D				V	G	A	B							
29								I	K	J			L	M			O	LJ	N	NJ		J	I				M	L			LJ	NJ	N	O	
30	T			G	N	NJ	O									T	G	D	O	NJ	N								T		D	N	O	NJ	
31				L	V	B	Đ	Ć	T	U				A											B	L	Đ	V	U	U	Ć				A
32		Ž	I	K	J			E	Z				LJ	M			J	K	I	Ž		E	Z			M	LJ								

33		I	I	P		G	G	J	J			G	D	D	P	I	I	J			D								G	I	J	J		D
34		R	K		Đ	A	A								K	K			A	N	N		R		Đ	Đ	A	N			K	K	Đ	N	N
35								E	Ž	Ž	S	O					NJ	NJ	O	O		Ž	E	E	S	NJ					NJ	NJ	O	O	
36		Z	Z		LJ	L	M		B	B	V	V				Z			LJ	L	M									B	B	V	V	
37				T	T	Ć	Č										Ć	Ć	U	U										U	U	T	T		
38		T	Ć	Ć	U	U		T	T			O	O	Đ		U	T	T	Ć	Ć		U	U			Đ	O	O							
39			I	NJ	NJ	G	D			I	I	J	K	K								K	K	J			G	D	I				NJ	N	N
40		E	Ž	Z	V	B	A								E	Ž	Z		V	B	A													
41								R	S				L	LJ								S	R	P			M		P				L	LJ	M
42		I			N	G	D			J	K	I			J	I				G	N		I	J	K		N	D						
43	U	T	Ć			NJ	O								T				O	NJ	Đ							Ć	U			O	NJ	Đ
44	E	Z				V	A								Ž	E				B	V							Ž	Z				B	A
45									G	D	Đ						I	J	K			K			G	D	Đ		J	I				
46	Ž	Z			V	A		E			A	B	V								E	Ž	Z			B								
47											L	Lj	M	N								P	R				Nj	O		S	T	Ć	U		
48															P	R	S		L	LJ	M	E	Ž	Z	I	J	K		A	B	V	G	D	Đ	
49	A	B	V	G	D	Đ		E	Ž	Z	I	J	K		R	S	P		Lj	M	L													
50				Đ	D	A	B						V	G	E	Ž				A	B						V	G	E	Ž			D	Đ	
51		I	J	K	Z						L	Lj	M	N									M	N	Lj	L			Z	J	K	I			
52											S	T	Ć	U	P	R				O	Nj				S	T	Ć	U	P	R				O	Nj
53	V	E	R	O	N	A	U	K	A																									
54	G	R	A	Đ	A	N	S	K	O	V	A	S	P	I	T	A	N	J	E															

U ovom koraku popravljen je samo raspored za prvi čas u ponedeljak za obe smene, ali je time verovatno narušeno još nekoliko ograničenja tako da je ovo dug i naporan posao dolaska do rešenja. Zato ćemo da vidimo šta timetabler radi sledeće - sva mesta gde je došlo do kršenja označavamo i brojimo koliko ih je bilo: u našem primeru ima po 4-5 grešaka po svakom času, odnosno preko 100 grešaka u celom rasporedu što je previše (ovo je samo grubo napravljen raspored časova, inače oni koji prave rasporede imaju znatno manje grešaka i brzo ih ispravljaju).

Zatim računamo generalizovanu cost funkciju (funkciju ocene korektnosti našeg rasporeda časova) našeg zadatka po formuli:

$z(\mathbf{R}) = \alpha \cdot \#inf + \beta_1 \cdot s_\Delta + \beta_2 \cdot s_\Omega + \beta_3 \cdot s_\Pi$

gde je #infu ovom slučaju recimo 125;

$\alpha, \beta_1, \beta_2, \beta_3$ su težine koje sami zadajemo i neka u našem slučaju imaju vrednost 1;

s_Π možemo da izbacimo jer ovde smatramo da svi nastavnici mogu uvek da rade;

s_Ω je trenutno zadovoljeno skoro u potpunosti osim što neki profesori imaju po jednu rupu, što se dozvoljava, ali posle razmeštanja verovatno više neće biti tako;

s_Δ su samo delimično zadovoljeni, u većini slučajeva nisu i na njih treba obratiti najviše pažnje;

Kad smo na ovaj način postavili problem i cost funkciju, glavna stvar nam je kako je minimizovati. Zato vršimo prepravke na rasporedu i pokušavamo da ga što bolje sredimo i nađemo minimalnu vrednost funkcije. Iako ova heuristuka može da reši naš problem, i dalje ostaje problem prevelike složenosti algoritma i stalnog računanja i minimizovonja cost funkcije. Kada smo minimizovali cost funkciju, i dalje ostaje pitanje koliko je raspored kvalitetan i da li je dati raspored najbolji, odnosno da li je bolji od bilo kog drugog koji se mogao napraviti? Na ovo pitanje je vrlo teško dati konkretan odgovor jer će uvek neko od nastavnika ili odeljenja biti sa nešto lošijim rasporedom u odnosu na ostale, ali je važno da dobijeni raspored zadovolji sve zahteve iz prve grupe ograničenja koje smo naveli u postavci problema i što više zahteva iz druge grupe ograničenja. Naravno, to nije uvek moguće ispuniti pa se raspored časova u školama menja i po nekoliko puta u toku školske godine dok se ne dobije zadovoljavajući raspored, a razvijaju se i programi i metode (heuristike) kako bi se ovaj težak matematički problem učinio lakšim za rešavanje i doveo do približnog rešenja ako već ne može do optimalnog.

Bibliography

[1] Bill P. Buckles, Fred Petry. *Genetic algorithms.* n.d.

[2] Brelaz, D. *"New Methods to Color the Vertices of a Graph."* . 1979.

[3] Cook, Stephen. *The complexity of theorem-proving procedures.* 1971.

[4] Csima, Joseph. *Investigations on a time-table problem* . 1965.

[5] Elie Sanchez, Takanori Shibata,Lotfi Asker Zadeh. *Genetic algorithms and fuzzy logic systems: soft computing perspectives.* n.d.

[6] Emile Aarts, Jan Karel Lenstra. *Local Search in Combinatorial Optimization.* n.d.

[7] Even, S., A. Itai, and A. Shamir. *On the complexity of time table and multi-commodity flow problems.* 1976.

[8] Feiring, Bruce. *Linear programming.* n.d.

[9] Fred Glover, Manuel Laguna. *Tabu search.* n.d.

[10] Goldberg. *Genetic algorithms in search, optimization, and machine learning.* 1989.

[11] Golumbic, Martin Charles. *Algorithmic Graph Theory and Perfect Graphs.* 1980.

[12] Gotlieb, J. Csima and C.C. *Tests on a Computer Method for Constructing School.* n.d.

[13] J.S. Appleby, D.V. Blaske,E.A. Newman. *Techiques for producing School timetables on a Computer and their Application to other Scheduling Problems.* 1960.

[14] Karp, Richard M. *Complexity of Computer Computations.* 1972.

[15] M. R. Garey, D. S. Johnson. *A Guide to the Theory of NP-Completeness Computers and Intractability.* n.d.

[16] Mills, P., Tsang, E.P.K., Williams, R., Ford, J. & Borrett, J. *An Easy abstract Constraint optimisation Programming Language.* December, 1999.

[17] Schaerf, A. *A Survey of Automated Timetabling.* 1999 .

[18] Sharma, Ram Chandra. *Modern methods of curriculum organization.* 1999.

[19] Srinivas, M., and L.M. Patnaik. *Genetic search: analysis using fitness moments* . n.d.

[20] Strohlein, Gunther Schmidt and Thomas. *Relation algebras | Concept of points and representability.* n.d.