UNIVERZITET U BANJOJ LUCI

ELEKTROTEHNIČKI FAKULTET

PRIMJENA ALGORITAMA I TEHNIKA OPTIMIZACIJE PRI AUTOMATSKOM KREIRANJU RASPOREDA NASTAVE

MASTER RAD

Mentor:
prof. dr Zoran Đurić

Kandidat:
Eldin Okanović

Banja Luka, decembar 2018.

UNIVERZITET U BANJOJ LUCI

ELEKTROTEHNIČKI FAKULTET

PRIMJENA ALGORITAMA I TEHNIKA OPTIMIZACIJE PRI AUTOMATSKOM KREIRANJU RASPOREDA NASTAVE

MASTER RAD

Mentor:
prof. dr Zoran Đurić

Kandidat:
Eldin Okanović

Banja Luka, decembar 2018.

UNIVERSITY OF BANJA LUKA

FACULTY OF ELECTRICAL ENGINEERING

APPLICATION OF OPTIMIZATION ALGORITHMS AND TECHNIQUES FOR AUTOMATED TIMETABLING

MASTER THESIS

Mentor:
prof. dr Zoran Đurić

Candidate:
Eldin Okanović

Banja Luka, December 2018.

Informacije o mentoru i master radu

Mentor:	prof. dr Zoran Đurić Vanredni profesor Elektrotehnički fakultet Banja Luka Univerzitet u Banjoj Luci
Tema:	Primjena algoritama i tehnika optimizacije pri automatskom kreiranju rasporeda nastave
Ključne riječi:	Optimizacija, Algoritam, Vještačka inteligencija, Raspored nastave, Minimizacija
Naučna oblast:	Inženjerstvo i tehnologija
Naučno polje:	Informaciono inženjerstvo, operaciona istraživanja, računarske i informacione nauke
Sažetak:	Pregled algoritama, tehnika optimizacije i postojećih rješenja za automatsko kreiranje rasporeda nastave. Detaljna analiza specifičnih zahtjeva za automatsko kreiranje rasporeda, na osnovu koje će biti predloženo novo web servis bazirano rješenje.
Klasifikaciona oznaka:	T 120
Tip licence:	CC BY-NC
Komisija:	prof. dr Branko Blanuša, redovni profesor, predsjednik komisije prof. dr Zoran Đurić, vanreni profesor, mentor doc. dr Dražen Brđanin, docent, član

Information about mentor and master thesis

Mentor:	Zoran Đurić, PhD Associate profesor Faculty of Electrical Engineering University of Banja Luka
Title:	Application of optimization algorithms and techniques for automated timetabling
Keywords:	Optimization, algorithm, artificial inteligence, class schedule, minimization
Scientific area:	Engineering and technology
Scientific field:	Information engineering, Operational research, Computer and information science
Abstract:	This thesis contains a review of well-known algorithms, optimization techniques and software tools for automated timetabling. It will be presented a detailed analysis of the specific constraints for automated timetabling, based on which will be presented a new web service based solution.
Classification mark:	T 120
Licence type:	CC BY-NC
Thesis committee:	Full professor, prof. dr Branko Blanuša, president of the committee Assoc. profesor, prof. dr Zoran Đurić, mentor Assoc. profesor, doc. dr. Dražen Brđanin, member of the committee

SADRŽAJ

1. UVOD

Kreiranja rasporeda nastave predstavlja veoma težak optimizacijski problem, a njegova ručna izrada je vremenski veoma zahtjevna. Proces kreiranja rasporeda se sastoji od pridruživanja nastavnih jedinica, koje obuhvataju nastavnika (predavača) i studente (učenika ili polaznika), prostorijama (učionicama, kabinetima i laboratorija) i ograničenom broju vremenskih slotova. Vremenski slot je najmanji vremenski period u danu kojeg je moguće alocirati.

Pored rješavanja konflikata između nastavnih jedinica u procesu kreiranja rasporeda, autor rasporeda mora uzeti u obzir i kvalitet dobijenog rasporeda, što uveliko komplikuje sam proces ručnog kreiranja rasporeda. Uzimajući u obzir ove, ali i druge probleme pri ručnom kreiranju rasporeda nastave, kao i činjenicu o velikom broju istraživanja i objavljenih naučnih radova iz oblasti primjene naprednih algoritama i tehnika vještačke inteligencije (genetski algoritmi[1], hiperheuristika[2], tabu pretraga[3], itd.) u automatizaciji procesa kreiranja rasporeda nastave, glavni cilj istraživanja u ovom radu je odabir i prilagođavanje pomenutih algoritama specifičnim zahtjevima analiziranim u četvrtom poglavlju ovog rada. Problem automatskog kreiranja rasporeda, između ostalih, pripada i klasi kombinatornih problema, tj. pri kreiranju rasporeda moguće je dobiti veliki broj izvodljivih (*eng. feasable*) rješenja, što predstavlja prostor rješenja. U praksi prostor izvodljivih rješenja je velik, te je neophodno iskoristiti algoritme sa efikasnom heuristikom kako bi se u što kraćem vremenu pronašlo što bolje rješenje u tom prostoru.

U drugom poglavlju dat je pregled i analiza postojećih algoritama i tehnika optimizacije za automatsko kreiranje rasporeda. U netrivijalnim slučajevima, problem automatskog kreiranja rasporeda se smatra NP-problemom (*eng.* **N**on-deterministic **P**olynomial-time), što znači da je malo vjerovatno da će se pronaći efikasna metoda za njegovo rješavanje. Zbog toga treba naglasiti da nijedna postojeća tehnika ne garantuje pronalazak optimalnog rješenja. Takođe je

[1] Genetski algoritam (GA) je algoritam pretrage inspiriran biološkim procesima kao što su mutacija, *crossover* i reprodukcija za dobijanje visoko kvalitetnih rješenja optimizacijskog problema.
[2] Hiperheuristika je algoritam pretrage koji, umjesto da pretražuje prostor/domen rješenja problema, vrši selekciju heurističkog algoritma ili sekvencu heurističkih algoritama nižeg nivoa koji dalje pretražuju prostor/domen rješenja određenog problema.
[3] Tabu pretraga je algoritam pretrage koji koristi tzv. „tabu listu“ u koju smješta posjećena rješenja koja ne zadovoljavaju definisane kriterije pretrage, te ih zadržava u listi određeni vremenski period. Ovim se izbjegava ponavljanje pretrage istih rješenja i povećava nivo diversifikacije u procesu pretrage.

napravljena razlika između tehnika i pristupa rješavanju problema optimizacije, gdje se tehnikom optimizacije smatra algoritam ili grupa algoritama za rješavanje problema optimizacije (npr. genetski algoritmi), a pristup rješavanju problema optimizacije se smatra opštim okvirom za razvoj algoritma za optimizaciju (npr. programiranje bazirano na logici ograničenja).

U trećem poglavlju prikazani su detalji najčešće korištenih softverskih alata za automatsko kreiranje rasporeda nastave, te je izvršena analiza njihovih mogućnosti.

Pored osnovnog skupa ograničenja za automatsko kreiranje rasporeda, na kojem je bazirana većina postojećih alata za automatsko kreiranje rasporeda nastave, u četvrtom poglavlju dat je pregled specifičnih zahtjeva i ograničenja.

U petom poglavlju je dokumentovano implementirano rješenje.

Šesto poglavlje sadrži primjer upotrebe implementiranog rješenja pri automatskom kreiranju rasporeda nastave nad realnim ulaznim parametrima.

U sedmom poglavlju napravljena je komparativna analiza implementiranog rješenja sa postojećim rješenjima prema različitim skupovima definisanih ograničenja odnosno ulaznih parametara.

Na kraju rada dat je zaključak.

2. ALGORITMI I TEHNIKE OPTIMIZACIJE U KREIRANJU RASPOREDA – PREGLED I ANALIZA

Kreiranje rasporeda nastave sastoji se od raspoređivanja nastavnih jedinica, koje dalje povezuju nastavnika sa studentima / učenicima, po prostorijama / kabinetima / laboratorijama / učionicama u ograničenom broju vremenskim slotova, pri čemu je potrebno uzeti u obzir različite vrste ograničenja. Ova ograničenja su podjeljena u dvije kategorije:

- fiksna ograničenja (eng. *hard constraints*),
- varijabilna ograničenja (eng. *soft constraints*).

Da bi dobijeni raspored nastave bio izvodljiv, on mora zadovoljiti sva postavljena fiksna ograničenja. Npr. dva nastavnika ne mogu se rasporediti u isti kabinet u isto vrijeme. Međutim, kao što je već pomenuto, u praksi broj izvodljivih rješenja može da bude velik, pri čemu je potrebno odabrati najkvalitetnije rješenje. Kvalitet rješenja se ogleda u zadovoljavanju postavljenih varijabilnih ograničenja. Tako npr. jedno od varijabilnih ograničenja može biti da pauza između uzastopnih nastavnih jedinica treba da bude što je moguće manja. Relacija između varijabilnih ograničenja i načina ocjenjivanja njihovog narušavanja/zadovoljavanja definiše objektivnu funkciju, prema kojoj se procjenjuje kvalitet izvodljivog rješenja. Cilj optimizacije rasporeda nastave je odabir onog izvodljivog rješenja kojim se želi ili minimizirati (rezultat objektivne funkcije predstavlja nivo narušavanja varijabilnih ograničenja) ili maksimizirati (rezultat objektivne funkcije predstavlja nivo zadovoljavanja varijabilnih ograničenja) rezultat objektivne funkcije.

2.1. Genetski algoritmi

Glavni problem optimizacijskih algoritama je pojava lokalnog optimuma[4]. Da bi se izbjegao lokalni optimum, koriste se različite tehnike kako bi se povećao stepen diversifikacije (eng. *diversification*) u procesu pretrage. Genetski algoritmi (GA) predstavljaju naprednu

[4] Lokalni optimum u optimizacijskom problemu je optimalno rješenje unutar skupa susjednih rješenja

optimizacijsku tehniku baziranu na prirodnoj evoluciji, koji su često sposobni pronaći globalni optimum[5]. Opšti princip rada GA prikazan je algoritmom datim na slici 2.1.

```
1:      formulate initial population
2:      randomly initialize population
3:      repeat
4:              evaluate objective function
5:              apply genetic operators: crossover, mutation, reproduction
6:      until stopping criteria
```

Slika 2.1. Pseudo kôd konvencionalnog genetskog algoritma

Glavnu ulogu u povećavanju nivoa diversifikacije ima linija 5 u kojoj se vrše operacije *crossover*, mutacija i reprodukcija. Ove operacije imaju zadatak da na osnovu inicijalne populacije[6] kreiranoj u liniji 2 proizvedu/reproduciraju novu populaciju rješenja koja se evaluiraju objektivnom funkcijom u liniji 4.

Algoritam na slici 2.2 predstavlja opis genetskog algoritma sa usmjerenom pretragom, prilagođenom problemu automatskog kreiranja rasporeda nastave [1].

```
1:    input: A problem instance I
2:    set the generation counter g := 0
      {initialize a random population}
3:    for i:= 1 to population size do
4:       s_i ← create a random solution
5:       s_i ← solution s_i after applying LocalSearch()
6:    end for
7:    while the termination condition is not reached do
8:       if(g mod τ) == 0 then
9:              apply ConstructMEM() to construct the data structure MEM
10:      end if
11:      s ← child solution generated by applying GuidedSearchByMEM() or with probability
         γ
12:      s ← child solution after mutation with probability P_m
13:      s ← child solution after applying LocalSearch()
14:      replace worst member of the population by child solution s
15:      g := g + 1
16:   end while
17:   output: The best achieved solution s_best for the problem instance I
```

Slika 2.2. Pseudo kôd genetskog algoritma sa usmjerenom pretragom

[5] Globalni optimum u optimizacijskom problemu je optimalno rješenje unutar čitavog prostora/domena rješenja

[6] Populacija u GA predstavlja skup kandidatskih rješenja, gdje se svaki element populacije dalje sastoji od skupa svojstava ili hromozoma koji se mogu izmijeniti primjenom nekih od genetskih operatora.

Za razliku od opšteg GA, gdje je bilo moguće generisanje više potomaka primjenom različitih genetskih operatora tokom procesa kreiranja nove populacije, ovdje se kreira samo jedan potomak po iteraciji/generaciji. Svaki član populacije čini jedno od mogućih rješenja rasporeda, koja se u daljnjem procesu mogu ili modifikovati (unaprijediti) ili zamjeniti sa boljim članom. Kvalitet člana – rješenja se određuje evaluacijom svih nastavnih jedinica u rješenju, radi određivanja broja i težine prekršenih ograničenja. Algoritam na slici 2.3 predstavlja opis metode *LocalSearch*.

```
1:   input: Individual I from the population
2:   for i := 1 to n do
3:      if event e_i is infeasable then
4:             if there is untried move left then
5:                    calculate the moves: first N1, then N2 if N1 fails, and finally
                      N3 if N1 and N2 fail
6:                    apply the matching algorithm to the time slots affected by the
                      move and delta evaluate the result
7:                    if moves reduce hard constraints violation then
8:                           make the moves and go to line 3
9:                    end if
10:            end if
11:     end if
12:  end for
13:  if no any hard constraints remain then
14:     for i := 1 to n do
15:            if event e_i has soft constraint violation then
16:                   if there is untried move left then
17:                          calculate the moves: first N1, then N2 if N1 fails, and
                            finally N3 if N1 and N2 fail
18:                         apply the matching algorithm to the time slots affected by
                            the move and delta evaluate the result
19:                         if moves reduce soft constraints violation then
20:                                make the moves and go to line 14
21:                         end if
22:                   end if
23:            end if
24:     end for
25:  end if
26:  output: A possibly improved individual I
```

Slika 2.3. LocalSearch metoda

Na početku izvršavanja algoritma vrši se inicijalizacija populacije čime se svakoj nastavnoj jedinici dodjeli proizvoljni vremenski slot prema uniformnoj distribuciji, te se

primjeni algoritam za alokaciju kabineta svakoj od nastavnih jedinica. Zatim se nad svakim članom inicijalne populacije primjeni metoda *LocalSearch*. Ova metoda vrši pomjeranje nastavnih jedinica unutar rasporeda pokušavajući smanjiti narušavanja definisanih ograničenja, tj. pokušavajući unaprijediti rješenje.

Nakon izvršene inicijalizacije, konstruiše se *MEM* struktura podataka[7]. Ova struktura se dalje koristi kao smjernica pri generisanju potomka za narednu generaciju. *MEM* struktura podataka se rekonstruiše nakon izvršenih τ generacija. Algoritam na slici 2.4 opisuje proces kreiranja *MEM* strukture.

```
1:   input: The whole population P
2:   sort the population P according to the fitness of individuals
3:   Q ← select the best α individuals in P
4:   for each individual I_j in Q do
5:      for each event e_i in I_j do
6:             calculate the penalty value of event e_i from I_j
7:             if e_i is feasible (i.e., has zero penalty) then
8:                    add the pair of room and time slot (r_{e_i}, t_{e_i}) assigned to e_i into the
                      list l_{e_i}
9:             end if
10:     end for
11:  end for
12:  output: The data structure MEM
```

Slika 2.4. ConstructMEM metoda za kreiranje MEM strukture

Dalje, prije nego se započne generisanje novog potomka, generiše se proizvoljan broj p [0.0, 1.0]. Ukoliko je p manji od vjerovatnoće γ onda se koristi metoda *GuidedSearchByMEM()* za generisanje novog potomka, u suprotnom se koristi *crossover* operator. Algoritmom sa slike 2.5 je opisan način kreiranja novog potomka (rješenja) primjenom metode *GuidedSearchByMEM()*.

Ukoliko se potomak generiše sa *crossover* operatorom, onda se iz populacije, na bazi takmičarske selecije (eng. *tournament selection*), izdvajaju dva rješenja kao roditelji novog potomka. Zatim se razmjenjuju vremenski slotovi između njih, te se alociraju kabineti za nastavne jedinice iz svakog popunjenog vremenskog slota, čime se dobija novi potomak tj. novo rješenje.

[7] MEM struktura podataka je lista nastavnih jedinica gdje svaka nastavna jedinica sadrži opet listu parova (prostorija, vremenski slot).

```
1:   input: The MEM data structure
2:   Es := randomly select β * n events
3:   for each event ei in Es do
4:      randomly select a pair of room and time slot from the list l_ei
5:      assign the selected pair to event ei for the child
6:   end for
7:   for each remaining event ei not in Es do
8:      assign a random time slot and room to event ei
9:   end for
10:  output: A new child generated using the MEM data structure
```

Slika 2.5. GuidedSearchByMEM metoda za kreiranje novog potomka

Nakon generisanja novog potomka, nad njim se primjenjuje *LocalSearch* metoda kojom će se pokušati unaprijediti novonastali potomak. *LocalSearch* metoda ujedno predstavlja i operaciju mutacije u genetskom algoritmu. Operacija mutacije se sastoji od proizvoljnog odabira jedne od susjednih struktura N1, N2 ili N3. Susjedne strukture predstavljaju lokalne izmjene nad rasporedom nastave, te su opisane na sljedeći način:

- N1: susjed definisan kao operator kojim se jedna nastavna jedinica pomjera iz jednog vremenskog slota u drugi vremenski slot,
- N2: susjed definisan kao operator kojim se zamjenjuju vremenski slotovi dvaju nastavnih jedinica,
- N3: susjed definisan kao operator kojim se vrši permutacija vremenskih slotova za tri nastavne jedinice na jedan od dva moguća načina permutacije.

Novonastali potomak se zatim zamjenjuje sa najlošijim potomkom iz populacije, čime se završava ciklus kreiranja nove populacije/generacije [1].
Testiranje algoritma je izvršeno nad ulaznim parametrima podijeljenim u 3 grupe: Small, Medium, Large sa karakteristikama datim u tabeli 2.1.

Tabela 2.1. Grupe ulaznih parametara za analizu efikasnosti algoritma [1]

Grupa	**Small**	**Medium**	**Large**
Broj nastavnih jedinica	100	400	400
Broj kabineta	5	10	10
Broj ograničenja	5	5	10
Broj ograničenja po kabinetu	3	3	5
% korištenih ograničenja	70	80	90
Broj studenata	80	200	400
Broj nastavnih jedinica po studentu	20	20	20
Broj studenata po nastavnoj jedinici	20	50	100

U tabeli 2.2 prikazana je usporedba rezultata ovog algoritma sa drugim naprednim algoritmima iz literature prema identičnim ulaznim parametrima.

Tabela 2.2. Usporedba algoritama nad malom, srednjom i velikom grupom ulaznih parametara [1]

Alg. /	**GSGA**		**RIIA**	**GALS**	**GBHH**	**VNS**	**THHS**	**LS**	**EA**	**AA**	**FA**
Grupa	Najb.	%	Najb.	Najb.	Najb.	Najb.	Najb.	%	Najb.	%	Najb.
S1	**0**	0	**0**	2	6	**0**	1	8	**0**	1	10
S2	**0**	0	**0**	4	7	**0**	2	11	3	3	9
S3	**0**	0	**0**	2	3	**0**	**0**	8	**0**	1	7
S4	**0**	0	**0**	**0**	3	**0**	1	7	**0**	1	17
S5	**0**	0	**0**	4	4	**0**	**0**	5	**0**	**0**	7
M1	240	242.5	242	254	327	317	**146**	199	280	195	243
M2	**160**	164	161	258	419	313	173	202.5	188	184	325
M3	**242**	245	265	251	359	357	267	77.5% (∞)	249	248	249
M4	**158**	161	181	321	348	247	169	177.5	247	164.5	285
M5	**124**	126.5	151	276	171	292	303	100% (∞)	232	219.5	132
L	**801**	822	100% (∞)	1027	1068	100% (∞)	80% (∞)	100% (∞)	100% (∞)	851.5	1138

Algoritmi prikazani u tabeli 2.2 su sljedeći:

- GSGA (eng. *The Guided Search Genetic Algorithm*) - genetski algoritam sa usmjerenom pretragom [1],
- RIIA (eng. The *Randomized Iterative Improvement Algorithm*) – stohastički iterativni algoritam baziran na poboljšanju rješenja [2],

- GALS (eng. *The Genetic Algorithm with Local Search*) – genetski algoritam sa lokalnom pretragom [3],
- GBHH (eng. *The Graph-Based Hyper Heuristic*) – hiperheuristika bazirana na grafovima [4],
- VNS (eng. *The Variable Neighbourhood Search*) – varijabilna pretraga susjedstva [5],
- THHS (eng. *The Tabu-based Hyper-Heuristic Search*) – tabu pretraga bazirana na hiperheuristici [6],
- LS (eng. *The Local Search*) – lokalna pretraga [7],
- AA (eng. *The Ant Algorithm*) – algoritam baziran na ponašanju mravlje kolonije [7],

EA (eng. *The Evolutionary Algorithm*) – evolucijski algoritam [8],

- FA (eng. *The Fuzzy Algorithm*) – fuzzy algoritam [9].

S1, S2, S3, S4, S5, M1, M2, M3, M4, M5 i L su instance koje pripadaju Small, Medium i Large grupi ulaznih parametara respektivno. Znak ∞ u tabeli 2.2 predstavlja procentualni prosjek slučajeva u kojima algoritam nije davao izvodljivo rješenje. Kolone „Najb.“ i „%“ predstavljaju rezultat najboljeg slučaja i medijana, respektivno. Pri testiranju izvršeno je 50 iteracija algoritma nad istom instancom ulaznih parametara. Najbolji rezultati su prikazani podebljanim slovima.

2.2. Simulated annealing algoritam

Simulated annealing (SA) je algoritam lokalne pretrage (pretraga susjednih rješenja), baziran na vjerovatnoći, koji se koristi za dobijanje rješenja optimizacijskog problema. Koncept algoritma se zasniva na simuliranju procesa hlađenje grupe zagrijanih vibrirajućih atoma korištenog u metalnoj industriji. Na slici 2.6 prikazan je SA algoritam [10].

Proces započinje kreiranjem proizvoljnog inicijalnog rješenja. Glavna procedura se sastoji od petlje u čijim iteracijama se kreira proizvoljni susjed (rješenje nastalo lokalnim izmjenama tekućeg rješenja). Označimo sa Δ razliku rezultata objektivne funkcije tekućeg i susjednog rješenja, i pretpostavimo da se radi o problemu minimizacije. Ako je $\Delta < 0$ (susjed je bolje rješenje), tada se prihvaća susjedno rješenje i postaje tekuće rješenje. Ova tehnika je veoma slična Hill-Climbing algoritmu [11], s tim ako je $\Delta \geq 0$ (susjed je lošije rješenje), tada će susjedno rješenje biti prihvaćeno sa vjerovatnoćom $e^{-\Delta/T}$ (sigmondova funkcija) [12], gdje

je T parametar, i predstavlja temperaturu. Na početku ovaj parametar se inicijalizuje na relativno visoku vrijednost T_0. Nakon fiksnog broja iteracija, temperatura se smanjuje na osnovu rasporeda hlađenja. Jedan od najčešće korištenih rasporeda za hlađenje jeste geometrijski raspored, prema sljedećem izrazu: $T_{i+1} = T_i * \beta$, gdje β predstavlja faktor hlađenja čija vrijednost se bira iz intervala (0.0 – 1.0) [10].

```
1:    input: s0 %initial solution
2:    i ← 0
3:    s ← s0
4:    sbest ← s0
5:    while global stop condition not satisfied do
6:       T ← CoolingSchedule(i)
7:       i ← i + 1
8:       while local stop condition not satisfied do
9:              s* ← PickRandomNeighbor(s)
10:             Δ ← ObjectiveFunction(s*) - ObjectiveFunction(s)
11:             if Δ < 0 then
12:                    s ← s*
13:                    Δbest ← ObjectiveFunction(s) - ObjectiveFunction(sbest)
14:                    if Δbest < 0 then
15:                           sbest ← s
16:                    end if
17:             else
18:                    r ← <random uniform number between 0 and 1>
19:                    if r < e^(-Δ/T) then
20:                           s ← s*
21:                    end if
22:             end if
23:      end while
24:   end while
25:   output: The best achieved solution sbest for the given initial solution s0
```

Slika 2.6. SA algoritam

2.3. Tabu pretraga

Tabu pretraga je metaheuristička tehnika[8] pretrage koja takođe koristi lokalnu pretragu radi pronalaska optimalnog rješenja. Za razliku od opšte lokalne pretrage ili iterativne lokalne pretrage, tabu pretraga vodi računa o posjećenim rješenjima, tj. posjećena rješenja pohranjuje u tzv. tabu listu, čime se daljni proces pretraga može preciznije usmjeriti u neistraženi prostor rješenja. Na slici 2.7 prikazan je opšti algoritam tabu pretrage.

Prema [13], opšti algoritam *tabu* pretrage može se unaprijediti njegovom integracijom sa iterativnom lokalnom pretragom čime se povećava stepen diversifikacije. Dva esencijalna parametra koja najviše utječu na proces izvršavanja adaptivne *tabu* pretrage su dubina *tabu* pretrage θ, i jačina perturbacije (eng. *perturbation*) iterativne lokalne pretrage η. Veća

[8] Metaheuristika je opšti algoritamski okvir namijenjen za rješavanje različitih optimizacijskih problema, a kojeg je, uz male modifikacije, moguće prilagoditi za rješavanje specifičnog problema.

vrijednost θ osigurava intenzivniju pretragu, dok jača perturbacija omogućava izbjegavanje lokalnog minimuma. Na slici 2.8 opisan je algoritam adaptivne *tabu* pretraga, prema kojoj se ova dva parametra podešavaju u samom procesu pretraživanja [13].

```
1:    input: s0 % initial solution
2:    s ← s0
3:    sbest ← s0
4:    tabuList ← []
5:    while global stop condition not satisfied do
6:       candidateList ← findNeighboors(s)
7:       bestCandidate ← null
8:       for each candidate cs in candidateList do
9:              if (not (tabuList contains cs)) and ObjectiveFunction(cs) <
                ObjectiveFunction(bestCandidate) then
10:                    bestCandidate ← cs
11:             end if
12:      end for
13:      s ← bestCandidate
14:      if ObjectiveFunction(bestCandidate) < ObjectiveFunction(sbest) then
15:             sbest ← bestCandidate
16:      end if
17:      add bestCandidate to end of tabuList
18:      if size of tabuList > maximum tabu list size then
19:             remove first solution from tabuList
20:      end if
21:   end while
22:   output: The best achieved solution sbest for the given initial solution s0
```

Slika 2.7. Opšti algoritam tabu pretrage

```
1:    input: s0 % feasable initial solution
2:    ξ ← 0, θ ← θ0, η ← ηmin
3:    sbest ← TS(s0, θ)
4:    while global stop condition not satisfied do
5:       s' ← Perturb(sbest, η) % perturbacija sbest jačinom η
6:       s*' ← TS(s', θ) % tabu pretraga dubinom θ
7:       if objectiveFunction(s*') ≤ objectiveFunction(sbest) + 2 then
8:              while no better solution is obtained do
9:                     θ ← (1+μ)*θ % postepeno povećanje dubine TS-a
10:                    s*' ← TS(s*',θ)
11:             end while
12:      end if
13:      if objectiveFunction(s*') < objectiveFunction(s*) then
14:             sbest ← s*' % prihvati s*' kao najbolje pronađeno rješenje
15:             θ ← θ0, η ← ηmin
16:      else
17:             θ ← θ0, ξ ← ξ + 1
18:             η ← max{ηmin + λ*ξ,ηmax}
19:      end if
20:   end while
21:   output: The best achieved solution sbest for the given initial solution s0
```

Slika 2.8. Algoritam adaptivne tabu pretrage

Takođe, za razliku od opšte *tabu* pretrage, adaptivna *tabu* pretraga sprema atribute posjećenih rješenja u *tabu* listu, umjesto kompletnih rješenja. Na početku procesa pretrage, dubina *tabu* pretrage se postavlja na minimalnu vrijednost θ_0, npr. na 10. Kada procedura *tabu* pretrage TS ne može unaprijediti rješenje, primjenjuje se perturbacija sa minimalnom jačinom ($\eta = \eta_{min}$). U daljnjem procesu (linija 13), pri pronalasku boljeg rješenja, ovi parametri se

vraćaju na početne vrijednosti. U suprotnom slučaju, uvećava se vrijednost ξ, koja dalje utiče na eventualno povećanje jačine perturbacije η, čime se povećava i stepen diversifikacije, odnosno izbjegavanje lokalnog minimuma.

U tabeli 2.3 nalaze se vrijednosti bitnih parametara algoritma.

Tabela 2.3. Vrijednosti parametara algoritma [13]

Parametar	Opis	Vrijednost ili proračunski izraz
θ_0	Minimalna dubina pretrage TS-a	10
μ	Brzina povećanja vrijednosti θ	0.6
θ	Dubina pretrage TS-a	$\theta = (1+\mu)*\theta$
ξ	Broj faza TS-a bez poboljšanja tekućeg rješenja	$\xi = \xi + 1$
η_{min}	Minimalna jačina perturbacija	4
η_{max}	Maksimalna jačina perturbacije	15
η	Jačina perturbacije	$\eta = \max\{ \eta_{min} + \lambda*\xi, \eta_{max} \}$
λ	Koeficijent ažuriranja η	0.3

Testiranje algoritma adaptivne *tabu* pretrage je izvršeno nad instancama tj. grupama parametara sa ITC[9] iz 2007. godine, te su rezultati upoređeni sa rezultatima dobijenim sa opštom *tabu* pretragom (TS), iterativnom lokalnom pretragom (ILS) i rezultatima objavljenim na stranici takmičenja ($ITC2007_{min}$) [14]. Rezultati testiranja algoritma adaptivne *tabu* pretrage prikazani su u tabeli 2.4.

[9] ITC – International Timetabling Competition [14]

Tabela 2.4. Usporedba rezultata adaptivne tabu pretrage prema uslovima ITC-2007 takmičenja u automatskom kreiranju rasporeda nastave [13]

Inst.	**Adaptivna *tabu* pretraga**						**TS**	**ILS**	**ITC2007$_{min}$**
	of_{min}	of_{avg}	σ	Iter.	Pert.	Sec.	of_{min}	of_{min}	of_{min}
test1	**224**	229.5	1.8	15586	208	189	230	226	234
test2	**16**	17.1	1.0	35271	406	182	16	16	17
test3	**74**	82.9	4.1	20549	369	160	82	79	86
test4	**74**	89.4	6.1	37346	735	208	92	83	132
comp01	**5**	5.0	0.0	321	5	5	5	5	5
comp02	**34**	60.6	7.5	15647	545	370	55	48	78
comp03	**70**	86.6	6.3	8246	102	257	90	76	93
comp04	**38**	47.9	4.0	5684	68	124	45	41	45
comp05	**298**	328.5	11.7	35435	54	191	315	303	326
comp06	**47**	69.9	7.4	13457	245	116	58	54	62
comp07	**19**	28.2	5.6	15646	368	383	33	25	38
comp09	**43**	51.4	4.6	17404	190	380	49	47	50
comp10	**16**	38	10.8	16026	160	389	23	23	27
comp11	**0**	0.0	0.0	236	3	3	0	0	0
comp12	**320**	365.0	17.5	40760	590	382	330	324	358
comp13	**65**	76.2	6.1	16779	182	330	71	68	77
comp14	**52**	62.9	6.4	24427	270	368	55	53	59

2.4. Hiperheuristika

Hiperheuristika je pristup koji koristi viši nivo apstrakcije u odnosu na metaheuristiku, odnosno ovim pristupom se vrši odabir heurističke metode nižeg nivoa, koja će biti primjenjena u određenom trenutku prema njenim karakteristika i karakteristikama regiona prostora rješenja koji se trenutno pretražuje. Pored obabira heurističke metode nižeg nivoa, zadatak hiperheuristike je da evidentira vrijeme izvršavanja odabrane metode, te njen utjecaj na rezultat objektivne funkcije. Važno je napomenuti, da hiperheuristika poznaje samo rezultate objektivnih funkcija heurističkih metoda, ali ne i informaciju šta objektivne funkcije predstavljaju [15]. Kao što je već napomenuto, hiperheuristika se sastoji od heurističkih metoda nižeg nivoa i selektora heurističke metode na višem nivou. Razlike u primjeni hiperheuristike se najviše ogledaju u implementaciji pomenutog selektora.

U [15] dat je opis implementacije selektora primjenom genetskog algoritma (GA) sa adaptivnom dužinom hromozoma. Ova implementacija predstavljena je oznakom hyper-GA. Hromozom u hyper-GA predstavlja sekvencu hiperheurističkih metoda. Skup heurističkih metoda nižeg nivoa se sastoji od 14 metoda, podjeljenih u 4 grupe i to:

a) Grupa za dodavanje nastavne jedinice
    1. Add-first
    2. Add-random
    3. Add-best
    4. Add-first-improvement
    5. Add-best-improvement
b) Kombiniranje dodavanja sa zamjenom nastavne jedinice
    6. Add-swap-first
    7. Add-swap-randomly
    8. Add-swap-first-improvement
    9. Add-swap-best-improvement
c) Kombiniranje dodavanja sa uklanjanjem nastavne jedinice
    10. Add-remove-first
    11. Add-remove-random
    12. Add-remove-worst
d) Grupa za uklanjanje nastavne jedinice
    13. Remove-first
    14. Remove-random

Primjer hyper-GA hromozoma dat je na slici 2.9. Brojevi unutar gena hromozoma predstavljaju redne brojeve heurističkih metoda.

3	4	2	6	1	8	10	9	12	2	11	10	13	14

Slika 2.9. Primjer hyper-GA hromozoma

U sklopu GA implementiran je jedan crossover operator nazvan *best-best* operator, te dvije vrste mutacije *removing-worst* i *inserting-good*. Zadatak *best-best* operatora jeste da od dva proizvoljno odabrana hromozoma iz populacije (roditelja) odabere najbolju grupu gena (heurističke metode nižeg nivoa koje su dale najbolje rezultate objektivne funkcije) i izvrši njihovu zamjenu, čime se dobijaju nova dva potomka u populaciji. Mutacija[10] *removing-worst*

[10] Mutacija predstavlja jedan od genetskih operatora GA kojim se pokušava član populacije unaprijediti

će ukloniti najlošiju grupu gena iz hromozoma (heurističke metode nižeg nivoa koje umanjuju rezultate objektivne funkcije, ili najduža grupa heurističkih metoda koje nisu uvećavale rezultate objektivne funkcije). Druga vrsta mutacije, *inserting-good* dodaje najbolju grupu gena iz proizvoljno odabranog hromozoma na proizvoljnu tačku u hromozomu nad kojim se vrši mutacija. Međutim, pošto je određivanje loše/dobre kombinacije gena veoma proračunski zahtjevna operacija, neophodno je implementirati jednostavniji i efikasniji mehanizam dodavanja i uklanjanja gena iz hromozoma. Ovo je postignuto primjenom *tabu* metode unutar hyper-GA algoritma koja je predstavljena novom oznakom hyper-TGA. Uloga *tabu* metode u hyper-GA algoritmu se ogleda u „penalizaciji“ gena koji nemaju utjecaj na rezultat objektivne funkcije. Npr. ako prvi gen (*add-best)* sa slike 2.10, nema utjecaja na rezultat objektivne funkcije, neće mu se dozvoliti poziv heurističke metode nižeg nivoa narednih *n* generacija. Parametar *n* predstavlja trajanje mandata (eng. *tabu tenure*) gena. Takođe, sa slike je vidljivo da dva različita gena sa istom heurističkom metodom mogu imati različito trajanje mandata. To znači da poziv određene heurističke metode može poboljšati rezultat objektivne funkcije, dok poziv iste heurističke metode tokom narednih *n* generacija neće uticati na rezultat objektivne funkcije.

3^2	4^0	2^0	6^0	1^0	8^0	10^1	9^0	12^0	2^1	11^2	10^2	13^0	14^0

Slika 2.10. Primjer hyper-TGA hromozoma

U [16] predstavljena je varijanta hiperheurističke tehnike sa nešto drugačijom skupinom heurističkih metoda nižeg nivoa, i implementacijom selektora zasnovanom na nagradi i kazni.

```
1:    input: s0 % feasable initial solution
2:    generate 30 initial chromosomes;
3:    initialise the tabu tenure tj for each gene to 0 %j is position of gene in chromosome
4:    store chromosomes into a pool
5:    sbest ← s0
6:    while global stop condition not satisfied do
7:       for each chromosome in the pool do
8:              apply low-level heuristic method to s0 according to the order of the genes
              in chromosome, when the script of corresponding gene equals to 0
9:              record the solution Sk % k is position of chromosome in the pool
10:             record the change for each applied low-level heuristic makes to the
              objective function: Cj
11:             for each tj > 0 do
12:                    tj ← tj – 1
13:             end for
14:             if Cj = 0 then
15:                    tj ← n % n is length of tabu list
16:             end if
17:      end for
18:      for each solution Sk do
19:             if Sk > s0 then
20:                    s0 ← Sk
21:             end if
22:      end for
23:      if s0 > sbest then
24:             sbest ← s0
25:      end if
26:      apply crossover operator and mutation to each chromosomes in current pool
27:      add new chromosomes and 10 best chromosomes from current pool to a new pool
28:   end while
29:   output: The best achieved solution sbest for the given initial solution s0
```

Slika 2.11. Hyper-TGA algoritam

2.5. Programiranje bazirano na ograničenjima

Programiranje bazirano na ograničenjima (eng. *Constraint-based programming*) predstavlja tehniku za deklarativni opis i efikasno rješavanje problema većih razmjera, najčešće iz oblasti planiranja, raspoređivanja i optimizacije. Problem koji se rješava ovom tehnikom se naziva problem zadovoljavanja ograničenja (eng. *Constraint-Satisfaction Problem – CSP*). Svaki *CSP* problem se može predstaviti sa sljedeće tri komponente:

- konačnim skupom varijabli V ($v_1, v_2, v_3, \ldots, v_N$),
- konačnim skupom domena D ($d_{v1}, d_{v1}, d_{v1}, \ldots, d_{vN}$), gdje je d_{vi} skup dozvoljenih vrijednosti za varijablu v_i,
- konačanim skupom ograničenja O ($o_1, o_2, o_3, \ldots, o_M$), kojim se ograničavaju vrijednosti iz pripadajućih domena koje se ne mogu istovremeno dodijeliti varijablama.

Rješenje CSP problema predstavlja dodjelu vrijednosti svim varijablama Vi iz pripadajućeg domena Di na način da se zadovolje sva postavljena ograničenja iz skupa O.

U slučaju primjene ove tehnike pri rješavanju problema automatskog kreiranja rasporeda nastave, primjenjuje se sljedeća pojmovna analogija:

- varijable – nastavne jedinice koje treba rasporediti,
- domen – skup dozvoljenih termina koji se mogu dodijeliti nastavnoj jedinici,
- ograničenja – skup fiksnih i varijabilnih ograničenja pri dodjeli termina nastavnoj jedinici.

Psuedo kodom, sa slike 2.12, opisan je opšti CSP algoritam.

```
1:    input: A: variables, B: values, C: constraints
2:    function SCHEDULE(A, B, C){
3:       timetable = ∅; //timetable is initially empty
4:       scheduling = true;
5:       a = highestpriority({x ∈ A; ¬ scheduled(x)}); //variable selection
6:       while(scheduling)∧({x ∈ A; ¬ scheduled(x)} ≠ ∅){
7:              b = mostsuitable({y ∈ Ba; ¬ considered(y)∧C(timetable+<a,y>)}); //value
                selection
8:              if found(b){ //successful selection
9:                     add(<a,b>, timetable);
10:                    a = highestpriority({x ∈ A; ¬ scheduled(x)});
11:             }else{ //unsuccessful selection or deadend
12:                    t = mostsensible({x ∈ A;scheduled(x)}) //find backtracking point
13:                    if found(t){ // bactracking point found
14:                           errase(a...t, timetable);
15:                           a = t;
16:                    }else{
17:                           scheduling = false; //general failure
18:                    }
19:             }
20:      }
21:   }
22:   output:({x ∈ A; ¬ scheduled(x)} = ∅) ? timetable : failure
```

Slika 2.12. Opšti CSP algoritam

Glavni zadatak koji se obavlja na početku svake iteracije algoritma jeste odabir varijable, odnosno nastavne jedinice. Opšta strategija odabira nastavne jedinice se bazira na stepenu ograničenosti termina iz pripadajućeg domena, te se najčešće bira ona nastavna jedinica sa najvećim stepenom. Drugim riječima, bira se ona nastavna jedinica koju je u tom trenutku najteže rasporediti.

Prateći određeni redoslijed odabira nastavnih jedinica moguća je pojava *deadend* situacije, u kojoj su svi razmatrani termini rezultirali sa *failure* statusom. U ovoj situaciji potrebno je pronaći tačku u tom redoslijedu odabira, te poništiti dodjeljene termine do te tačke. Ovo predstavlja *backtrace* proces. Strategija odabira tačke povratka predstavlja Također bitan faktor koji utječe na performanse samog algoritma. Prema [17], odabirom nastavne jedinice *c* u prvom koraku kreira se pripadajuća *failure* tabela koja sadrži onoliko redova koliko ima termina u pripadajućoj domeni odabrane nastavne jedinice. Za svaki termin iz pripadajućeg domena, čije razmatranje rezultira sa *failure* statusom, u njegov red *failure* tabele upisuju se

druge nastavne jedinice koje su uzrokovale *failure* status. Ova tabela se dalje koristi pri odabiru tačke povratka, odnosno nastavne jedinice do koje će se vršiti poništavanje dodjeljenih termina. Prvo se u svakom *failure* redu ostavljaju nastavne jedinice koje su prve upisane (proces minimizacije). Zatim se od preostalih nastavnih jedinica u *failure* tabeli bira zadnja koja je upisana (proces maksimizacije). Pronađena nastavna jedinica u procesu maksimizacije predstavlja tačku povratka u *backtrace* procesu.

3. POSTOJEĆI SOFTVERSKI ALATI ZA AUTOMATSKO KREIRANJE RASPOREDA – PREGLED I ANALIZA

Postoje mnogobrojni softverski alati, komercijalni i besplatni, za automatsko kreiranje rasporeda. Neki od najčešće korištenih su: FET, OCTT, UniTime, ASC Timetables i dr.

3.1. FET

FET [18] (eng. *Free Timetabling Software)* je besplatni open source alat, razvijen korišćenjem C++ programskog jezika, koji omogućava automatsko kreiranje rasporeda nastave na osnovu definisanih parametara i ograničenja. Zadnja verzija 5.27.2 je predstavljena 12.05.2015. godine. U nastavku dat je pregled svih vrsta ograničenja podjeljenih u dvije grupe: vremenska i prostorna (resursna) ograničenja:

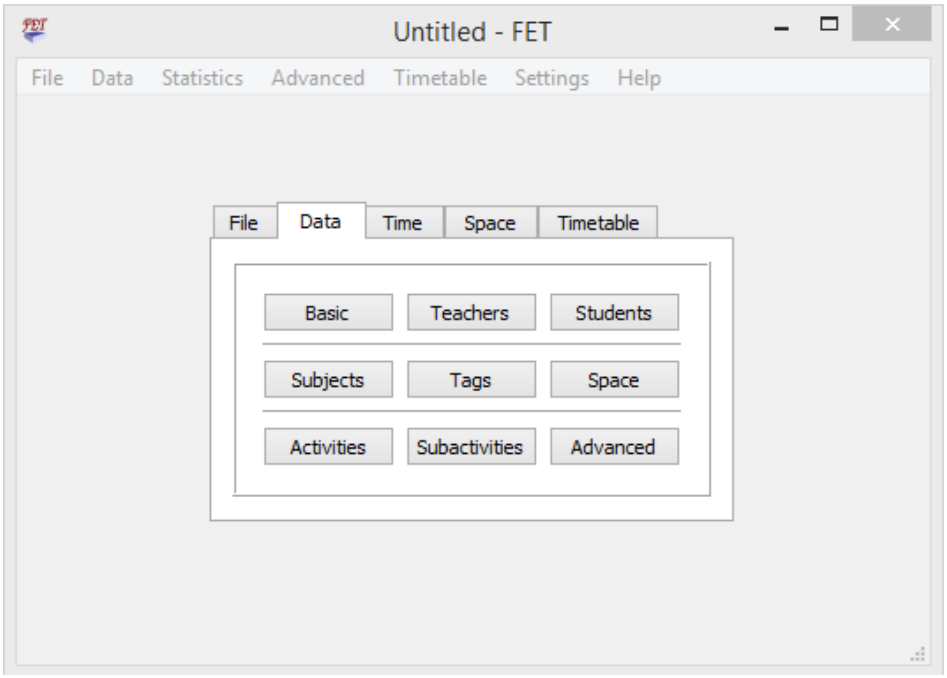
- vremenska ograničenje – zajednička:
  - ✓ osnovna obavezna vremenska ograničenja,
  - ✓ pauze(svi nastavnici + svi studenti nisu dostupni);
- vremenska ograničenja – nastavnici:
  - ✓ nastavnik nije dostupan,
  - ✓ maksimalni broj dana u sedmici po nastavniku,
  - ✓ maksimalna praznina (eng. *gap*) između predavanja po sedmici po nastavniku,
  - ✓ maksimalna praznina u sedmici za sve nastavnike,
  - ✓ maksimalna praznina po danu i nastavniku,
  - ✓ maksimalna praznina po danu za sve nastavnike,
  - ✓ maksimalan broj sati dnevno po nastavniku,
  - ✓ maksimalan broj sati dnevno za sve nastavnike,
  - ✓ minimalan broj sati dnevno za nastavnika,
  - ✓ minimalan broj sati dnevno za sve nastavnike,
  - ✓ maksimalan broj uzastupnih sati po nastavniku,
  - ✓ maksimalan broj uzastupnih sati za sve nastavnike,
  - ✓ nastavnik radi u definisanoj satnici maksimalan broj dana u sedmici,
  - ✓ svi nastavnici rade u definisanoj satnici maksimalna broj dana u sedmici;

- vremenska ograničenja – studenti:
  - ✓ grupa studenata nije dostupna,
  - ✓ maksimalna praznina između predavanja u sedmici po grupi studenata,
  - ✓ maksimalna praznina između predavanja u sedmici za sve studente,
  - ✓ nastava za grupu studenata počinje ranije,
  - ✓ nastave za sve studente počinje ranije,
  - ✓ maksimalan broj sati nastave dnevno za grupu studenata,
  - ✓ maksimalna broj sati nastave dnevno za sve studente,
  - ✓ minimalan broj sati nastave dnevno za grupu studenata,
  - ✓ minimalan broj sati nastave dnevno za sve studente,
  - ✓ maksimalan broj uzastupnih sati nastave dnevno za grupu studenata,
  - ✓ maksimalan broj uzastupnih sati nastave dnevno za sve studente;
- vremenska ograničenja – nastavne jedinice:
  - ✓ nastavna jedinica ima poželjno vrijeme početka izvođenja,
  - ✓ nastavna jedinica ima skup poželjnih vremenskih slotova,
  - ✓ skup nastavnih jedinica ima skup poželjnih vremenskih slotova,
  - ✓ skup određenih vrsta nastavnih jedinica ima skup poželjnih vremenskih slotova,
  - ✓ nastavna jedinica ima skup poželjnih vremena početka izvođenja,
  - ✓ skup određenih vrsta nastavnih jedinica ima skup poželjnih vremena početka izvođenja,
  - ✓ minimalan broj dana između skupa nastavnih jedinica,
  - ✓ nastavna jedinica se izvodi zadnja u danu,
  - ✓ skup nastavnih jedinica se izvodi zadnja u danu,
  - ✓ skup nastavnih jedinica imaju isto vrijeme početka izvođenja (dan i sat),
  - ✓ skup nastavnih jedinica imaju isto vrijeme početka izvođenja (bilo koji dan),
  - ✓ skup nastavnih jedinica imaju isto vrijeme početka izvođenja (bilo koji sat),
  - ✓ dvije nastavne jedinice su u slijedu,
  - ✓ dvije nastavne jedinice su uzastopne,
  - ✓ skup nastavnih jedinica se ne preklapa,
  - ✓ minimalna praznina (broj sati) između skupa nastavnih jedinica;
- prostorna ograničenja – zajednička:
  - ✓ osnovna obavezna prostorna ograničenja;

- prostorna ograničenja – kabineti:
  - ✓ kabinet nije dostupan;
- prostorna ograničenja – nastavnici:
  - ✓ nastavnik preferira korištenje određenog kabineta,
  - ✓ nastavnik preferira korištenje skupa kabineta,
  - ✓ maksimalan broj promjena zgrada po danu i nastavniku,
  - ✓ maksimalan broj promjena zgrada po danu za sve nastavnike,
  - ✓ maksimalan broj promjena zgrada po sedmici i nastavniku,
  - ✓ maksimalan broj promjena zgrada po sedmici za sve nastavnike,
  - ✓ minimalna praznina između promjene zgrade po nastavniku,
  - ✓ minimalna praznina između promjene zgrade za sve nastavnike;
- prostorna ograničenja – studenti:
  - ✓ skup studenata preferira korištenje određenog kabineta,
  - ✓ skup studenata preferira korištenje skupa kabineta,
  - ✓ maksimalan broj promjena zgrada po danu za skup studenata,
  - ✓ maksimalan broj promjena zgrada po danu za sve studente,
  - ✓ maksimalan broj promjena zgrada po sedmici za skup studenata,
  - ✓ maksimalan broj promjena zgrada po sedmici za sve studente;
- prostorna ograničenja – nastavna jedinica:
  - ✓ nastavna jedinica se izvodi u određenom kabinetu,
  - ✓ nastavna jedinica se može izvoditi u skupu kabineta;

Kao što se može zaključiti iz gore navedenog, FET alat može zadovoljiti potrebe većine obrazovnih ustanova u automatskom kreiranju rasporeda nastave. Bitno je napomenuti da ograničenja nisu podijeljena na fiksna i varijabilna, već istim se daje prioritet na osnovu parametra težine ograničenja (eng. *constraint weight*), čija je vrijednost u rasponu od 0 - 100%. Npr. ograničenje sa težinom 100% predstavlja fiksno ograničenje koje je neophodno zadovoljiti pri kreiranju rasporeda nastave. Jedna od bitnih funkcionalnosti ovog alata jeste mogućnost pregleda onih nastavnih jedinica, koje je program u toku kreiranja rasporeda ocijenio kao teške ili nemoguće za rasporediti, a na osnovu kojeg se onda mogu izvršiti dodatne korekcije nad definisanim ograničenjima. Druga značajna funkcionalnost jeste mogućnost fiksnog raspoređivanja određenih nastavnih jedinica prije pokretanja generatora rasporeda, koje se neće

uzimati u obzir u procesu automatskog kreiranja rasporeda. Na slici 3.1 prikazan je početni interfejs FET alata.



Slika 3.1. FET 5.27.2

3.1.1. Algoritam raspoređivanja

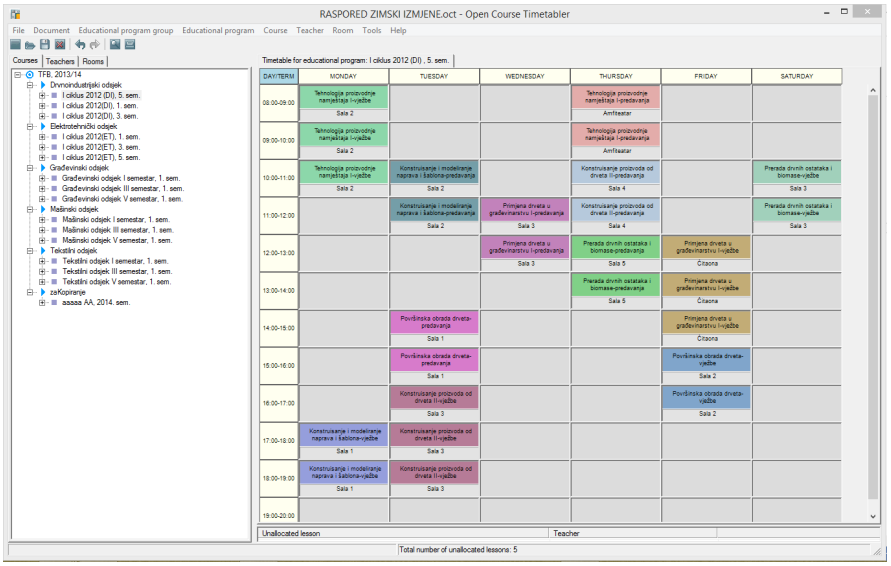
Na početku projekta FET alat je koristio genetski algoritam pri automatskom kreiranju rasporeda nastave. Međutim, implementacija algoritma nije davala zadovoljavajuće rezultate, te je mogla riješiti samo jednostavnije probleme raspoređivanja. Trenutni algoritam koji se koristi u zadnjoj verziji alata implementiran je 2007. godine, te je njegovo izvršavanje puno brže i u mogućnosti je da riješi mnogo kompleksnije probleme raspoređivanja. Algoritam je nazvan „rekurzivna zamjena“ (eng. *recursive swapping*) i opisan je sa algoritmom sa slike 3.2.

```
1:  input: set of activities and constraints
2:  Sort activities, most difficult first. Not critical step, but speeds up the algorithm
    maybe 10 times or more.
3:  Try to place each activity (A_i) in an allowed time slot, following the above order,
    one at a time. Search for an available slot (T_j) for A_i, in which this activity
    can be placed respecting the constraints. If more slots are available, choose a
    random one. If none is available, do recursive swapping:
4:  For each time slot T_j, consider what happens if you put A_i into T_j. There will
    be a list of other activities which don't agree with this move (for instance,
    activity A_k is on the same slot T_j and has the same teacher or same students as
    A_i). Keep a list of conflicting activities for each time slot T_j
5:  Choose a slot (T_j) with lowest number of conflicting activities. Say the list of
    activities in this slot contains 3 activities: A_p, A_q, A_r.
6:  Place A_i at T_j and make A_p, A_q, A_r unallocated.
7:  Recursively try to place A_p, A_q, A_r (if the level of recursion is not too large,
    say 14, and if the total number of recursive calls counted since step (3) on A_i
    began is not too large, say 2*n), as in step (3).
8:  If successfully placed A_p, A_q, A_r, return with success, otherwise try other time
    slots (go to step (5) and choose the next best time slot).
9:  If all (or a reasonable number of) time slots were tried unsuccessfully, return
    without success.
10: If we are at level 0, and we had no success in placing A_i, place it like in steps
    (5) and (6), but without recursion. We have now 3 - 1 = 2 more activities to place.
    Go to step (3) (some methods to avoid cycling are used here)
11: output: set of time slots containing assigned activities, and activities containing
    assigned rooms
```

Slika 3.2. FET Algoritam „rekurzivna zamjena“

3.2. OCTT

OCTT [19] (eng. *Open Course Timetabler*) je besplatni open source alat razvijen u C# programskom jeziku, te se može izvršavati na Windows operativnim sistemima sa .NET platformom. Mogućnosti alata su prilagođene zahtjevima i načinu izvođenja nastave na obrazovnim ustanovama iz regiona. Na slici 3.3 prikazan je interfejs OCTT alata.



Slika 3.3. OCTT 0.8.1

Skup ograničenja je podijeljen na fiksna i varijabilna ograničenja. Kao fiksna ograničenja moguće je definisati:

- dodjelu kabineta nastavnoj jedinici u kojim se ona može izvoditi,
- dodjelu kabineta nastavniku u kojim je nastavniku dozvoljeno izvođenje nastave,
- dodjelu dozvoljenih/zabranjenih vremenskih slotova nastavniku,
- dodjelu dozvoljenih/zabranjenih vremenskih slotova grupi studenata,
- dodjelu dozvoljenih/zabranjenih vremenskih slotova kabinetima,
- grupisanje nastavnih jedinica koje istovremeno izvodi jedan nastavnik u istom kabinetu.

Varijabilna ograničenja je moguće definisati na nivou čitavog rasporeda ili pojedinačno po nastavniku i grupi studenata. Neka od tih ograničenja su:

- maksimalan broj uzastupnih sati nastave za nastavnike,
- maksimalan broj uzastupnih sati nastavne za grupu studenata,
- maksimalan broj sati nastave dnevno za nastavnike,
- maksimalan broj sati nastave dnevno za grupu studenata,
- maksimalan broj dana u sedmici za nastavnike,
- maksimalan broj dana u sedmici za grupu studenata,
- maksimalan broj pauza između nastave za grupu studenata,
- željeno vrijeme početka nastave za grupu studenata.

Za svako postavljeno varijabilno ograničenje može se definisati i dodatni parametar težina ograničenja kojim se prioritetizuju postavljena varijabilna ograničenje.

3.2.1. Algoritam raspoređivanja

Algoritam automatskog raspoređivanja OCTT-a baziran je na ACO (eng. *Ant Colony Optimization*) algoritmu. ACO je probabilistički algoritam za pronalazak optimalne putanje kroz grafove, baziran na ponašanju mrava u koloniji pri traženju izvora hrane. Na početku izvršavanja algoritma mravi proizvoljno obilaze čvorove grafa, gdje se pronalaskom izvora hrane (odredišni čvor grafa) vraćaju u koloniju ostavljajući trag - feromon. Ukoliko se drugi

mravi nađu na putanji koja sadrži feromon, ti mravi se neće više kretati proizvoljno već će pratiti trag do izvora hrane. Takođe, poznato je da mravi biraju putanju sa najviše feromona na osnovu vjerovatnoće, čime se postiže viši stepen diversifikacije, tj. neki mravi će odabrati i alternativne putanje. Bitna karakteristika feromona je njegov rok trajanja, odnosno feromon tokom vremena isparava. Tako neke putanje koje mravi ne koriste ostaju bez feromona, i samim time postaju manje „interesantne“ za ostale članove kolonije. Ovaj princip razmjene informacija o okolini poznat je kao fenomen kolektivne inteligencije [20, 7].

Graf [21] u ovom algoritmu se sastoji od sljedećih vrsta čvorova: čvor nastavne jedinice, čvor kabineta, čvor vremenskog slota, te čvor spoja (eng. *docking node*) koji služi kao veza između prethodne tri vrste čvorova. Svaka veza između čvorova sadrži podatak o količini feromona, te heurističku vrijednost koja određuje kvalitet veze. Pri izvršavanju algoritma korišteno je *m* mrava. U svakoj iteraciji algoritma svaki mrav je kreirao izvodljivo rješenje, tj. raspored nastave koji zadovoljava sva postavljena fiksna ograničenja. Pri odabiru sastavne komponente rješenja, vjerovatnoća p_{ij}^k, kojom će mrav k koji se trenutno nalazi na čvoru i odabrati čvor j, se računa prema sljedećem izrazu:

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \aleph_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta}, \qquad j \in \aleph_i^k \tag{1}$$

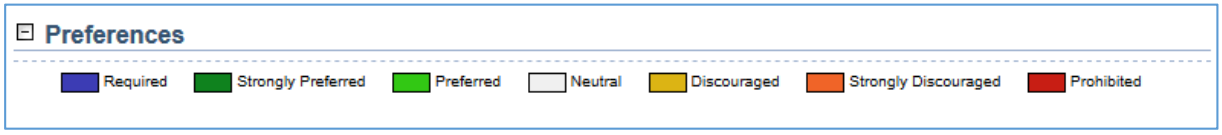
gdje su τ_{ij} količina feromona između čvora i i čvora j, η_{ij} heuristička vrijednost veze između čvorova i i j, α i β parametri koji određuju relativni utjecaj između količine feromona i heurističke vrijednosti, te $\aleph_i^k$ je prihvatljiva susjedna komponenta za mrava k kada se nalazi na čvoru i.

3.3. UniTime

UniTime [22], je web bazirani alat za kreiranje rasporeda nastave, ispita, te raznih vrsta događaja. Implementacija alata je open source i besplatna, razvijena u Java programskom jeziku, te je dostupna na službenoj web stranici alata. Zadnja verzija 4.2 je predstavljena 21.06.2017. godine. Kao i pomenuti FET alat, UniTime nudi veliku paletu ograničenja podijeljenu u tri glavne grupe: vremenska, prostorna i distribucijska. Za svako definisano

ograničenje moguće je dodatno definisati težinu tog ograničenja koja je podijeljena u sedam nivoa, prikazanih na slici 3.4. Neka od bitnijih ograničenja su:

- vremenska ograničenja po nivoima:
  - ✓ čitav raspored,
  - ✓ nastavnik,
  - ✓ vrsta nastavne jedinice (predavanja, vježbe, itd...),
  - ✓ nastavna jedinica,
  - ✓ prostorija;
- prostorna ograničenja po nivoima:
  - ✓ odsjek,
  - ✓ nastavnik,
  - ✓ vrsta nastavne jedinice (predavanja, vježbe, itd...),
  - ✓ nastavna jedinica;
- distribucijska ograničenja:
  - ✓ čitav raspored,
  - ✓ nastavnik,
  - ✓ vrsta nastavne jedinice (predavanja, vježbe, itd...),
  - ✓ nastavna jedinica;



Slika 3.4. Nivoi ograničenja Unitime alata

Ograničenja „Required“ ili „Prohibited“ predstavljaju fiksna ograničenja, i kao takva se moraju zadovoljiti pri generisanju rasporeda, te se ne mogu zaobići sa ograničenjima druge težine u nižem nivou.

Jedna od prednosti ovog alata u odnosu na prethodno pomenute alate jeste mogućnost kreiranje rasporeda ispita koji se održavaju tokom nastave, te definisati dodatna ograničenja između nastave i ispita.

Takođe, ovaj alat omogućava detaljniji pregled statističkih podataka dobijenih mjerenjem performansi algoritma raspoređivanja. To je jedan od razloga zašto je u sekciji 7.3

izvršena detaljnija analiza i poređenje rezultata dobijenih ovim alatom sa rezultatima dobijenim korišćenjem rješenja predloženog u ovom radu.

3.3.1. Algoritam raspoređivanja

Algoritam automatskog raspoređivanja Unitime alata baziran je na IFS (eng. *Iterative forward search*) algoritmu. Za razliku od klasičnih metoda lokalne pretrage, ovaj algoritam se izvršava nad izvodljivim rješenjima, koja ne moraju nužno biti kompletna. Proces izvršavanja algoritma nastave se sastoji od dvije faze i to:

- kreiranje inicijalnog izvodljivog rješenja,
- primjena IFS algoritma za minimiziciju narušavanja varijabilnih ograničenja

Pretraga se izvršava iterativno (slika 3.5), počevši sa inicijalnim rasporedom ω, koji može biti prazan. Tokom svakog koraka, bira se nastavna jedinica (Linija 4). U većini slučajeva bira se neraspoređena nastavna jedinica. Nakon toga bira se „najbolji“ termin za odabranu nastavnu jedinicu (Linija 5). Nakon odabira „najboljeg“ termina, vrši se pretraga raspoređenih nastavnih jedinica kojima ovaj termin narušava fiksna ograničenja (Linija 6). Ukoliko dodjela „najboljeg“ termina odabranoj nastavnoj jedinici narušava fiksna ograničenja već raspoređenim nastavnim jedinicama, te nastavne jedinice se uklanjaju sa rasporeda i „najbolji“ termin se dodjeljuje odabranoj nastavnoj jedinici (Linija 7). Ovim se osigurava da je u svakom trenutku tekući raspored ω izvodljiv. Dalje, algoritam koristi CBS (eng. *Conflict-based* statistics) tehniku učenja kako bi izbjegao beskonačno ponavljanje i poboljšao kvalitetu konačnog rješenja [23].

```
1:    input: ω % initial timetable ω
2:       σ = ω % initial timetable is set as the best solution
3:       while(canContinue(ω))
4:              ν = selectVariable(ω) % class selection
5:              d = selectValue(ν,ω) % period selection
6:              γ = hardConflicts(ω,ν/d) % list of assigned classes with hard constraint
                                           violation
7:              ω = ω\γ U {ν/d} % period assignment to the selected class
8:              if better(ω,σ) then
9:                     σ = ω % remember the better solution
10:             end if
11:      end while
12:   output: σ % return best found solution
```

Slika 3.5. IFS algoritam

Implementacija CBS tehnike sadrži strukturu podataka koja pamti narušavanja fiksnih ograničenja tokom pretrage, njihovu frekvenciju i dodjele termina nastavnim jedinicama koje su uzrokovale narušavanja. Tačnije, struktura je predstavljena kao niz polja:

$$CBS[A/a \rightarrow \neg B/b] = c_{ab} \quad (2)$$

Izraz (2) se može tumačiti na sljedeći način: Dodjela termina *a* nastavnoj jedinici *A* (predstavljeno sa *A/*a) je uzrokovao c_{ab} puta narušavanje fiksnih ograničenja tokom izvršavanja algoritma pri dodjeljenom terminu *b* nastavnoj jedinici *B* (predstavljeno kao *B/b*) [23].

4. ANALIZA SPECIFIČNIH ZAHTJEVA I OGRANIČENJA ZA AUTOMATSKO KREIRANJE RASPOREDA

Ovaj rad je motivisan problemom izrade rasporeda nastave na Tehničkom fakultetu Univerziteta u Bihaću. Za kreiranje rasporeda nastave zadužen je prodekan za nastavu, koji treba da, pri ručnom kreiranju rasporeda nastave, balansira razne zahtjeve nastavnika, odsjeka, nastavnih jedinica, pa čak i studenata. Dodatni problem u ovom, ali i drugim sličnim slučajevima, stvara kratak vremenski period između dobijanja svih ulaznih parametara i samog početka nastave, što za posljedicu obično ima česte izmjene rasporeda nastave na početku semestra. U ovom poglavlju biće predstavljeni specifični zahtjevi i ograničenja kao ulazni parametri u procesu automatskog kreiranja rasporeda nastave.

U ljetnom semestru 2016/2017. godine trebalo je rasporediti preko 230 nastavnih jedinica u 15 prostorija, sa oko 500 studenata unutar šest dvanaestosatnih dana (od ponedjeljka do subote, od 8h do 20h). Kad bi smo zanemarili pojedine zahtjeve i ograničenja pri kreiranju rasporeda, sam proces kreiranja rasporeda ne bi predstavljao problem. Međutim, uzimajući u obzir pomenute zahtjeve i ograničenja uveliko se smanjuje mogućnost zadovoljavanja istih, pogotovo ukoliko se raspored kreira ručno.

4.1. Opis osnovnog problema

Prije formulisanja problema, potrebno je dodijeliti oznake osnovnim pojmovima koji su potrebni pri izradi rasporeda. Neka oznake $c_1, \dots, c_m$ označavaju m studijskih grupa, oznake $t_1, \dots, t_n$ označavaju n nastavnika i neka brojevi $1, \dots, p$ predstavljaju p termina (vremenski i prostorni resurs). Zatim, neka $R_{m\times n}$ predstavlja matricu nastavnih jedinica gdje vrijednost r_{ij} označava broj nastavnih jedinica koje nastavnik t_j održava studijskoj grupi c_j u zadanom terminu (unutar p termina).

De Werra [24] je formulisao kreiranje rasporeda kao smještanje nastavnih jedinica (studenti i nastavnik) u prostorije i vremenske slotove (satnica), pri čemu se moraju ispuniti osnovni zakoni fizike, tj. nastavnik i student ne mogu biti istovremeno raspoređen u više od jedne prostorije, kao:

pronađi $X_{m\times n\times p}$ tako da

$$\forall(i,j) \sum_{k=1}^{p} x_{ijk} = r_{ij}, \tag{3}$$

$$\forall(i,k) \sum_{j=1}^{n} x_{ijk} \leq 1, \tag{4}$$

$$\forall(j,k) \sum_{i=1}^{m} x_{ijk} \leq 1, \tag{5}$$

$\forall(i,j,k)\ x_{ijk} = 0$ ili 1,

pri čemu je $x_{ijk} = 1$ ako nastavnik t_j predaje studijskoj grupi c_i u k-tom terminu, u suprotnom $x_{ijk} = 0$. Ograničenje prikazano izrazom (3) osigurava da nastavnik održi onoliko nastavnih jedinica koliko mu je i zadano, dok ograničenja (4) i (5) osiguravaju da niti jedna studijska grupa i nastavnik nemaju raspoređenu dvije ili više nastavnih jedinica u isto vrijeme.

Takođe je dokazano [25] da uvijek postoji rješenje za navedeni problem uz poštivanje uslova da niti jedna studijska grupa i niti jedan nastavnik nemaju zadano više od p nastavnih jedinica koje je potrebno rasporediti, tj. ako su ispunjena sljedeća dva uslova:

$$\forall(i) \sum_{j=1}^{n} r_{ij} \leq p,$$

$$\forall(j) \sum_{i=1}^{m} r_{ij} \leq p.$$

Iako je raspored dobijen rješavanjem problema opisan izrazima (3), (4) i (5) ispravan i izvodljiv, mala je vjerovatnoća da će isti zadovoljiti realne potrebe fakulteta. Svakako u obzir treba uzeti i satnice u kojima određena studijska grupa ili nastavnik nije u mogućnosti prisustvovati/održati nastavu. Ovo otežava rješavanje problem, ali dobijeno rješenje čini prihvatljivijim.

Neka su $C_{m\times p}$ i $T_{n\times p}$ dvije binarne matrice takve da vrijedi $c_{ik} = 1$ (odnosno $t_{jk} = 1$) ukoliko je studijska grupa c_i (odnosno nastavnik t_j) raspoloživ u zadanom terminu k, a u suprotnom $c_{ik} = 0$ (odnosno $t_{jk} = 0$). Sada se problem može formulisati na sljedeći način:

pronađi $X_{m\times n\times p}$ tako da

$$\forall(i,j)\ \sum_{k=1}^{p} x_{ijk} = r_{ij},$$

$$\forall(i,k)\ \sum_{j=1}^{n} x_{ijk} \leq c_{ik}, \tag{6}$$

$$\forall(j,k)\ \sum_{i=1}^{m} x_{ijk} \leq t_{jk}, \tag{7}$$

$\forall(i,j,k)\ x_{ijk} = 0$ ili 1,

Izrazi (6) i (7) zamjenjuju i čine strožim izraze (4) i (5), čime se osigurava da niti jednom nastavniku i niti jednoj studijskoj grupi ne može biti raspoređena nastavna jedinica u terminu kad nisu raspoloživi.

Na identičan način može se definisati i matrica raspoloživosti prostorije u određenom terminu $U_{o\times p}$, za koju vrijedi $u_{lk} = 1$ ukoliko je prostorija u_l raspoloživa u zadanom terminu k, a u suprotnom $u_{lk} = 0$. Zatim, svakoj prostoriji se pridružuje skup opreme kojom raspolaže F_l. Jednako tako se i svakoj nastavnoj jedinici r_{ij} može pridružiti skup neophodne opreme za njeno izvođenje E_{ij}. Ovom dopunom problem raspoređivanja se formuliše na sljedeći način:

pronađi $X_{m\times n\times p\times o}$ tako da

$$\forall(i,j,k)\ \sum_{l=1}^{o} x_{ijkl} \leq u_{lk}, \tag{8}$$

$$\forall(x_{ijkl}) \text{ tako da } x_{ijkl} = 1 \rightarrow E_{ij} \subseteq F_l \quad (9)$$

Izrazom (8) se osigurava da nastava u određenoj prostoriji ne može biti raspoređena u one termine kad navedena prostorija nije raspoloživa. Izraz (9) ograničava održavanje nastave u samo onim prostorijama koje zadovoljavaju definisane kriterijume opremljenosti prostorije.

4.2. Dodatna ograničenja

Glavni cilj u razvoju automatizovanog sistema za kreiranje rasporeda nastave jeste minimizacija vremenskih i prostornih konflikata između raspoređenih nastavnih jedinica. Ovo značajno utiče na sam proces generisanja rasporeda budući da postoji mnogo specifičnih zahtjeva pri odabiru adekvatne prostorije i satnice.

Kako bi se minimizirali potencijalni vremenski konflikti, dnevna satnica je podijeljena u 12 jednosatnih slotova (npr. vremenski slot od 8:00 – 9:00). Ovakva podjela vremenskih resursa, dalje omogućava jednostavnije definisanje vremenskih ograničenja po raznim nivoima, koja su opisana detaljnije u narednim sekcijama.

4.2.1. Vremenska ograničenja

Parametar raspoloživosti nastavnika ili studijske grupe u određenom terminu, predstavljen u sekciji 4.1., ne zadovoljava realne potrebe fakulteta, čime bi se dobijeno rješenje smatralo prihvatljivim. Neophodno je uvesti širu skalu između prihvatljivih i neprihvatljivih satnica, odnosno širu skalu vremenskih ograničenja. Skala ograničenja sastoji se od sedam različitih ograničenja (slika 4.1).

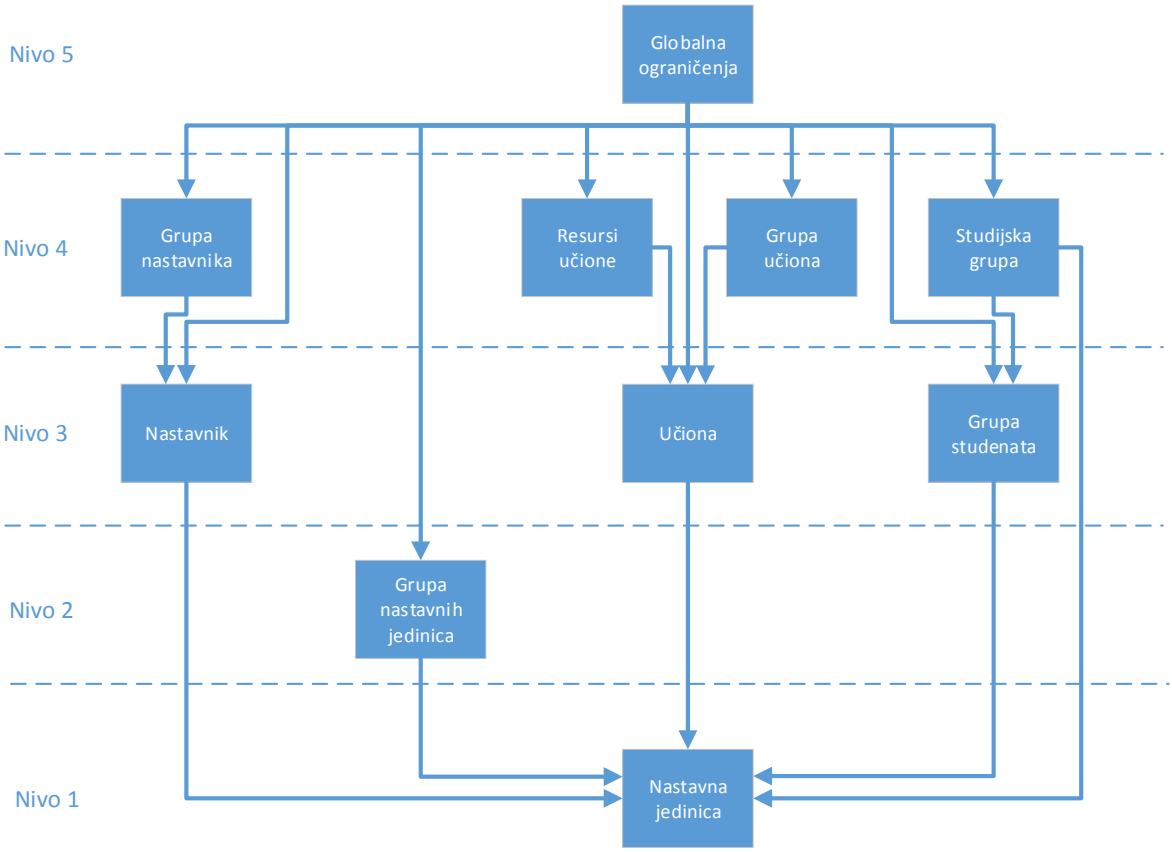
	Neophodan
2	**Veoma poželjan**
1	**Poželjan**
0	**Neutralan**
-1	**Nepoželjan**
-2	**Veoma nepoželjan**
	Zabranjen

Slika 4.1. Skala ograničenja

Ograničenja „Neophodan“ i „Zabranjen“ smatraju se fiksnim ograničenjima, tj. ograničenja koja se moraju ispoštovati kako bi se rješenje smatralo izvodljivim, dok se ostala međuograničenja smatraju varijabilnim ograničenjima. Važno je istaknuti, da ukoliko je termin označen ograničenjem „Neophodan“, svi ostali termini neće biti uzeti u obzir pri raspoređivanju.

Kako je prikazano na skali, svakom od varijabilnih ograničenja dodjeljena je numerička vrijednost, koja će se dalje koristiti pri evaluaciji, odnosno optimizaciji rješenja.

Da bi omogućilo što fleksibilnije definisanje vremenskih ograničenja koja opisuju realna očekivanja od dobijenog rješenja, omogućeno je da se vremenska ograničenja definišu nad različitim elementima raspoređivanja prikazanim na dijagramu sa slike 4.2. Zatim, da bi se ubrzao proces definisanja vremenskih ograničenja za pojedine elemente, omogućeno je njihovo grupisanje. Npr. nastavnici se mogu grupisati u grupe „Vanjski saradnici“ i „Stalno zaposleni“. Zatim za svaku grupu moguće je definisati vremenska ograničenja. U ovom slučaju, i ukoliko nastavniku nisu direktno definisana vremenska ograničenja, nastavnik preuzima vremenska ograničenja iz grupe kojoj pripada. Dalje, ako se nastavnik nalazi u više od jedne grupe, izvršiće se operacija *crossover* nad vremenskim ograničenjima svih grupa kojim on pripada, a rezultat operacije će predstavljati vremenska ograničenja nastavnika. Ukoliko su nastavniku direktno definisana vremenska ograničenja, onda ona redefinišu (eng. *override*) ograničenja iz višeg nivoa, odnosno ograničenja pripadajuće grupe/a, ili globalna ograničenja ako nastavnik ne pripada niti jednoj grupi.



Slika 4.2. Dijagram distribucije vremenskih ograničenja

Iz navedenog se može zaključiti da će se *crossover* operacija vršiti nad ograničenjima elemenata sa istog nivoa, dok ograničenja elemenata sa nižeg nivoa redefinišu ograničenja definisana nad elementima iz višeg nivoa. Slijed distribucije vremenskih ograničenja prikazan je smjernicama na dijagramu. Rezultati redefinisanja i operacije *crossover* dati su u tabelama 4.1 i 4.2 respektivno.

Tabela 4.1. Rezultat redefinisanja ograničenja

A	B	A override B
Neophodan	Neophodan	Neophodan
Neophodan	Veoma poželjan	Neophodan
Neophodan	Poželjan	Neophodan
Neophodan	Neutralan	Neophodan
Neophodan	Nepoželjan	Neophodan
Neophodan	Veoma nepoželjan	Neophodan
Neophodan	Zabranjen	Zabranjen
Veoma poželjan	Neophodan	Neophodan
Veoma poželjan	Veoma poželjan	Veoma poželjan
Veoma poželjan	Poželjan	Veoma poželjan
Veoma poželjan	Neutralan	Veoma poželjan
Veoma poželjan	Nepoželjan	Veoma poželjan
Veoma poželjan	Veoma nepoželjan	Veoma poželjan
Veoma poželjan	Zabranjen	Zabranjen
Poželjan	Neophodan	Neophodan
Poželjan	Veoma poželjan	Poželjan
Poželjan	Poželjan	Poželjan
Poželjan	Neutralan	Poželjan
Poželjan	Nepoželjan	Poželjan
Poželjan	Veoma nepoželjan	Poželjan
Poželjan	Zabranjen	Zabranjen
Neutralan	Neophodan	Neophodan
Neutralan	Veoma poželjan	Veoma poželjan
Neutralan	Poželjan	Poželjan
Neutralan	Neutralan	Neutralan
Neutralan	Nepoželjan	Nepoželjan
Neutralan	Veoma nepoželjan	Veoma nepoželjan
Neutralan	Zabranjen	Zabranjen
Nepoželjan	Neophodan	Neophodan
Nepoželjan	Veoma poželjan	Nepoželjan
Nepoželjan	Poželjan	Nepoželjan
Nepoželjan	Neutralan	Nepoželjan
Nepoželjan	Nepoželjan	Nepoželjan
Nepoželjan	Veoma nepoželjan	Nepoželjan
Nepoželjan	Zabranjen	Zabranjen
Veoma nepoželjan	Neophodan	Neophodan
Veoma nepoželjan	Veoma poželjan	Veoma nepoželjan
Veoma nepoželjan	Poželjan	Veoma nepoželjan
Veoma nepoželjan	Neutralan	Veoma nepoželjan
Veoma nepoželjan	Nepoželjan	Veoma nepoželjan
Veoma nepoželjan	Veoma nepoželjan	Veoma nepoželjan
Veoma nepoželjan	Zabranjen	Zabranjen
Zabranjen	Neophodan	Zabranjen
Zabranjen	Veoma poželjan	Zabranjen
Zabranjen	Poželjan	Zabranjen
Zabranjen	Neutralan	Zabranjen
Zabranjen	Nepoželjan	Zabranjen
Zabranjen	Veoma nepoželjan	Zabranjen
Zabranjen	Zabranjen	Zabranjen

Tabela 4.2. Rezultat crossover operacije nad ograničenjima

A	B	A crossover B
Neophodan	Neophodan	Neophodan
Neophodan	Veoma poželjan	Neophodan
Neophodan	Poželjan	Neophodan
Neophodan	Neutralan	Neophodan
Neophodan	Nepoželjan	Neophodan
Neophodan	Veoma nepoželjan	Neophodan
Neophodan	Zabranjen	Zabranjen
Veoma poželjan	Neophodan	Neophodan
Veoma poželjan	Veoma poželjan	Veoma poželjan
Veoma poželjan	Poželjan	Poželjan
Veoma poželjan	Neutralan	Veoma poželjan
Veoma poželjan	Nepoželjan	Neutralan
Veoma poželjan	Veoma nepoželjan	Neutralan
Veoma poželjan	Zabranjen	Zabranjen
Poželjan	Neophodan	Neophodan
Poželjan	Veoma poželjan	Poželjan
Poželjan	Poželjan	Poželjan
Poželjan	Neutralan	Poželjan
Poželjan	Nepoželjan	Neutralan
Poželjan	Veoma nepoželjan	Nepoželjan
Poželjan	Zabranjen	Zabranjen
Neutralan	Neophodan	Neophodan
Neutralan	Veoma poželjan	Veoma poželjan
Neutralan	Poželjan	Poželjan
Neutralan	Neutralan	Neutralan
Neutralan	Nepoželjan	Nepoželjan
Neutralan	Veoma nepoželjan	Veoma nepoželjan
Neutralan	Zabranjen	Zabranjen
Nepoželjan	Neophodan	Neophodan
Nepoželjan	Veoma poželjan	Neutralan
Nepoželjan	Poželjan	Neutralan
Nepoželjan	Neutralan	Nepoželjan
Nepoželjan	Nepoželjan	Nepoželjan
Nepoželjan	Veoma nepoželjan	Veoma nepoželjan
Nepoželjan	Zabranjen	Zabranjen
Veoma nepoželjan	Neophodan	Neophodan
Veoma nepoželjan	Veoma poželjan	Neutralan
Veoma nepoželjan	Poželjan	Nepoželjan
Veoma nepoželjan	Neutralan	Veoma nepoželjan
Veoma nepoželjan	Nepoželjan	Veoma nepoželjan
Veoma nepoželjan	Veoma nepoželjan	Veoma nepoželjan
Veoma nepoželjan	Zabranjen	Zabranjen
Zabranjen	Neophodan	Zabranjen
Zabranjen	Veoma poželjan	Zabranjen
Zabranjen	Poželjan	Zabranjen
Zabranjen	Neutralan	Zabranjen
Zabranjen	Nepoželjan	Zabranjen
Zabranjen	Veoma nepoželjan	Zabranjen
Zabranjen	Zabranjen	Zabranjen

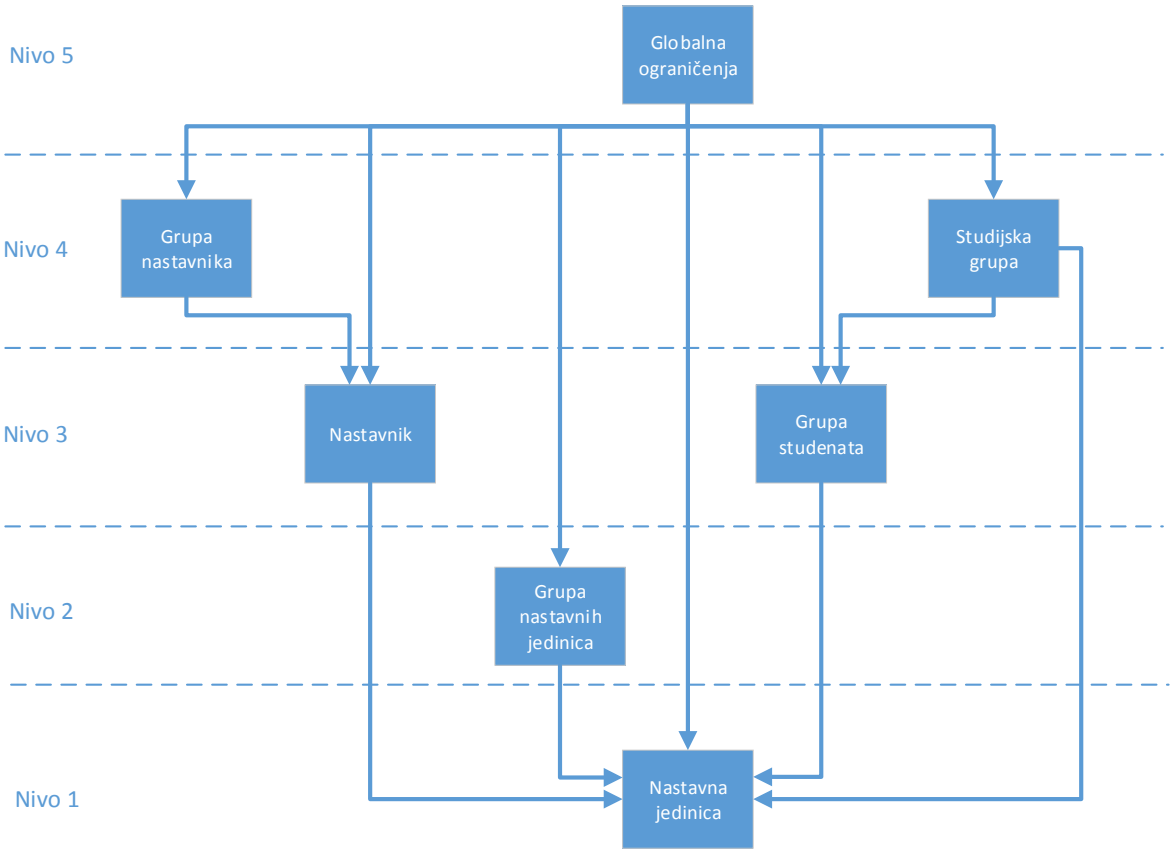
Iz rezultata obe operacije, može se zaključiti da su ograničenja „Neophodan“ i „Zabranjen“ nepromjenjiva bez obzira na vrstu operacije, odnosno ograničenje „Neophodan“ se može promijeniti samo ograničenjem „Zabranjen“. Ovim je osigurano da pomenuta ograničenja predstavljaju fiksna ograničenja, što je pomenuto na početku poglavlja.

4.2.2. Prostorna ograničenja i ekskluzivitet prostorije

Izvođenje nastave iz određenog predmeta može zahtijevati određeni skup prostorija ili laboratorija. Samim tim, pored definisanja vremenskih ograničenja, neophodno je omogućiti i definisanje ograničenja za korištenje pojedinih prostorija. Za definisanje ograničenja prostorija koristit će se ista skala kao i za definisanje vremenskih ograničenja (slika 4.1). Takođe, kao i kod definisanja vremenskih ograničenja, ograničenje „Neophodan“ i „Zabranjen“ smatraju se fiksnim ograničenjima. Isto tako, ako se prostorija označi sa ograničenjem „Neophodan“, ostale prostorije neće biti uzete u obzir pri raspoređivanju.

Dalje, ekskluzivitet prostorije omogućava da se ista izuzme pri raspoređivanju onih nastavnih jedinica koje nemaju definisana ograničenja nad prostorijama. Npr. ako laboratorije označimo kao ekskluzivne prostorije, onemogućit će se da predavanje iz predmeta „Matematika I“ bude raspoređeno u laboratorije ukoliko za isto nisu definisane ograničenja za korištenje laboratorija.

Na slici 4.3. prikazan je dijagram distribucije prostornih ograničenja kroz različite elemente raspoređivanja. ence



Slika 4.3. Dijagram distribucije prostornih ograničenja

4.2.3. Zajednička i dijeljena nastava

Neki studijski programi mogu sadržavati predmete čiji je plan i program identičan. U ovoj situaciji jedan nastavnik održava nastavu istovremeno za studente sa različitih studijskih programa. Ova vrsta ograničenja otežava proces raspoređivanja, jer se mora osigurati da studenti iz različitih studijskih grupa moraju biti raspoloživi u isto vrijeme, te obezbijediti prostorija koja kapacitetom zadovoljava ukupan broj studenata koji prisustvuju nastavi. Ovo fiksno ograničenje se definiše tako što se unutar nastavne jedinice definiše lista drugih nastavnih jedinica s kojim se nastava treba održavati u isto vrijeme. Bitno je napomenuti da se prije početka procesa kreiranja rasporeda vrši *crossover* operacija nad vremenskim i prostornim ograničenjima svih nastavnih jedinica koje se trebaju održavati istovremeno, a rezultat operacije predstavlja zajednička vremenska i prostorna ograničenja za sve navedene nastavne jedinice.

Takođe, nastava iz pojedinih predmeta se izvodi u laboratorijima koje često imaju manji kapacitet od broja studenata koji pohađa nastavu. Zbog toga je neophodno dodatno podijeliti

studente u grupe, te za svaku grupu definisati različitu nastavnu jedinicu koju treba rasporediti. Takođe, ako postoji potreba da se iz dva ili više predmeta iz iste studijske grupe nastava izvodi u grupama, moguće je iskoristiti iste grupe, što omogućava da se nastava za dvije ili više različitih grupa održava u isto vrijeme od strane različitih nastavnika i u različitim prostorijama. Kao dodatak pomenutoj vrsti ograničenja, uzeta je u obzir i sljedeća situacija: npr. vježbe iz predmeta „Matematika I“ se održavaju u dvije grupe A1 i A2, dok se vježbe iz predmeta „Uvod u programiranje“ održavaju u tri grupe G1, G2 i G3. Kako bi se izbjeglo raspoređivanje grupa sa zajedničkim studentima u isto vrijeme, a sa druge strane omogućilo istovremeno održavanje nastave za grupe bez zajedničkih studenata, bilo je potrebno označiti koje od ovih pet grupa sadrže zajedničke studente u obliku parova, npr. A1-G1, A1-G2, A2-G2 i A2-G3. Prema ovakvim postavkama, nastava za grupu A1 se može održavati u isto vrijeme kad i nastava za grupu G3, i isto tako nastava za grupe A2 i G1, jer ne sadrže iste studente.

4.2.4. Kontinuirano izvođenje nastave

Na predmetima gdje jedan nastavnik izvodi i predavanja i vježbe, a najčešće se radi o vanjskom saradniku, javlja se potreba da se nastava izvodi u istom danu i u kontinuitetu. Ova vrsta ograničenja je regulisana na način da se nastavnoj jedinici definiše lista drugih nastavnih jedinica koje će se kontinuirano izvoditi prije ili poslije tekuće nastavne jedinice. Dodatak ovom ograničenju jeste definisanje parametra *ista_uciona* kojim se sve nastavne jedinice, pored toga što se raspoređuju u uzastopne termine, raspoređuju u istu prostoriju.

4.2.5. Virtuelna prostorija

Za neke predmete nastava se izvodi putem *online* platforme (npr. Adobe Connect), za koju nije potrebno rezervisati fizičku prostoriju, niti specijalnu opremu. Za ovu potrebu u parametrima se kreira fiktivna prostorija i označi kao virtuelna. Označavanjem prostorije virtuelnom omogućit će istovremeno izvođenje nastave od strane različitih nastavnika i za različite studijske grupe, tj. neće postojati vremenski konflikti među nastavnim jedinicama koje se istovremeno izvode u istoj virtuelnoj prostoriji.

5. IMPLEMENTACIJA ALGORITMA

U ovom poglavlju opisana je implementacija predloženog algoritma za automatsko kreiranje rasporeda nastave, počevši od pseudo koda, pa do pregleda strukture ulaznih parametara. Implementacija predloženog algoritma je izvršena korišćenjem Java programskog jezika. Radi jednostavnijeg opisa algoritma uvedeni su sljedeći pojmovi:

- varijabla – instanca klase net.etfbl.rg.NastavnaJedinica,
- vrijednost – instanca klase *net.etfbl.rg.Termin,*
- domen varijable – lista dozvoljenih vrijednosti *List<net.etfbl.rg.Termin>*.

5.1. Pseudo kod algoritma

Predloženi algoritam je baziran na kombinaciji dvije tehnike, i to na tehnici programiranja baziranog na ograničenjima, opisanoj u sekciji 2.5 ovog rada, i redukciji domena pretrage (eng. *Domain reduction*). Na slici 5.1 prikazan je pseudo kod predloženog algoritma.

```
1:    for every X in unassignedVarList l
2:       generateDomain(X)
3:    end for
4:    while(not empty unassignedVarList l)
5:       V = selectUnassignedVariable(l) // V - selected variable
6:       D = selectValueFromDomain(V) // D - selected value
7:       if(D ≠ null)
8:              if(propagateAssignment(V,D) ≠ DEAD_END)
9:                     assignValue(V,D)
10:                    l.remove(V)
11:             else
12:                    if(isDomainEmpty(V))
13:                           backtrackFrom(V)
14:                    end if
15:             end if
16:      else
17:             if(isExtendableDomain(V))
18:                    extendDomain(V)
19:             else
20:                    markVariableAsUnassignable(V)
21:                    l.remove(V)
22:             end if
23:      end if
24:   end while
```

Slika 5.1. Pseudo kod predloženog algoritma

Izvršavanje algoritma započinje generisanjem vrijednosti domena (linija 2) za svaku neraspoređenu nastavnu jedinicu. Vrijednost domena predstavlja par termin – prostorija, a pri generisanju vodi se računa da ista ne narušava definisana fiksna ograničenja. Prema definisanim ograničenjima u ulaznim parametrima, svakom paru se računa i pridružuju stepen vremenskog

i prostornog ograničenja. Nakon toga, vrši se redukcija domena, pri kojoj se zadržavaju vrijednosti sa najvećim stepenom ograničenja, odnosno najpoželjniji termini i prostorije. Ovom početnom redukcijom domena vrijednosti osigurava se da algoritam pokuša rasporediti nastavne jedinice u najpoželjnije termine i prostorije, odnosno maksimalno zadovoljiti varijabilna ograničenja.

Odabir neraspoređene nastavne jedinice (linija 5) vrši se najčešće na temelju veličine njenog domena, tj. odabire se ona nastavna jedinica sa najmanjim domenom. Na ovaj način odabira varijable, smanjuje se mogućnost pojave *deadend*[11] situacije. Svakako, algoritam može u toku izvršavanja dati prioritet varijabli na drugi način, npr. ukoliko je prethodno raspoređena nastavna jedinica koja treba da se izvodi u kontinuitetu sa drugom nastavnom jedinicom, u tom slučaju druga nastavna jedinica u nastavku dobija prednost pri odabiru.

Odabir vrijednosti iz domena odabrane varijable (linija 6) vrši se na osnovu utjecaja njene dodjele pripadajućoj varijabli na redukciju domena ostalih neraspoređenih nastavnih jedinica (eng. *look-ahead*), tj. odabire se ona vrijednost koja će imati najmanji utjecaj. Nakon odabira vrijednosti, ista se uklanja iz domena i sprema u listu testiranih vrijednosti. Uspješnim odabirom vrijednosti (linija 7), vrši se propagacija dodjele iste odabranoj varijabli. Glavni cilj propagacija jeste redukcija, odnosno uklanjanje vrijednosti iz domena drugih neraspoređenih varijabli koje direktno narušavaju fiksna ograničenja, čime se postiže ARC-konzistentnost[12] istih. Na osnovu navedenog, može se zaključiti da u svakom trenutku parcijalno rješenje zadovoljava sva fiksna ograničenja. Takođe važno je napomenuti, da nakon izvršene redukcije domena obuhvaćenih (eng. *affected*) varijabli, iste se provjeravaju da li su prazne, te ukoliko jesu, rezultat propagacije predstavlja *dead-end* situacija, pri kojoj se odabire sljedeća vrijednost iz domena i ponavlja postupak. U slučaju da nakon redukcije, u domeni ostane samo jedna vrijednost, ponavlja se postupak propagacije dodjele preostale vrijednosti pripadajućoj varijabli [26].

Ako je domena prazna nakon *dead-end* situacije i sve vrijednosti su testirane, tada se vrši *backtrack* (linija 13), odnosno poništavaju se prethodne dodjele vrijednosti raspoređenim varijablama u obrnutom redoslijedu njihove dodjele, i to svim varijablama u nizu zaključno sa zadnjom varijablom čija dodjela je utjecala na redukciju domena tekuće varijable. Zatim se,

[11] *Deadend* situacija u postupku raspoređivanja predstavlja slučaj kada domen neraspoređene nastavne jedinice, pri njegovoj redukciji u postupku raspoređivanja, postane prazan

[12] Varijabla x_i je ARC-konzistentna sa drugom varijablom x_j, ukoliko za svaku vrijednost a iz domena varijable x_i postoji vrijednost b iz domena varijable x_j, tako da (a, b) zadovoljava ograničenja između varijabli x_i i x_j [23]

zbog poništenja dodjele, vraćaju vrijednosti u domen pripadajućih varijabli, koje su bile uklonjene tokom propagacije. Nakon izvršenog *backtrack*-a, postavlja se prioritet nad tekućom varijablom pri odabiru naredne neraspoređene varijable, te se sve testirane vrijednosti vraćaju u njenu domenu.

Ukoliko sve vrijednosti domena vode do *deadend* situacije, vrši se provjera i proširenje domena (linija 18) sa manje poželjnim vrijednostima. Ako proširenje nije moguće, tada se varijabla označava kao nerasporediva (linija 20) sa odgovarajućom porukom, te se ista uklanja iz liste neraspoređenih varijabli.

Zbog pojednostavljenja pseudo koda algoritma, izostavljen je mehanizam detekcije *deadlock*-a, gdje se dvije varijable „takmiče“ za dodjelu iste vrijednosti. U slučaju *deadlock*-a, jedna varijabla se označava kao nerasporediva sa odgovarajućom porukom i uklanja iz liste neraspoređenih varijabli, dok se drugoj dodjeljuje vrijednost.

Predloženi algoritam implementiran je u Java programskom jeziku, a kompletan izvorni kôd dostupan je u prilogu 1 ovog rada. U nastavku poglavlja opisani su najvažniji dijelovi algoritma, te njihova implementacija. Pored implementacije algoritma, kreiran je web interfejs za definisanje ulaznih parametara, koji je opisan u narednom poglavlju ovog rada.

5.1.1. Kreiranje domena

U ovoj inicijalnoj fazi izvršavanja algoritma, svakoj neraspoređenoj nastavnoj jedinici se kreira njena kompletna domena na osnovu definisanih vremenskih i prostornih ograničenja u ulaznim parametrima. Pri kreiranju domena, svakoj vrijednosti iz domena se računa stepen vremenskog i prostornog ograničenja, nakon čega se sve vrijednosti sortiraju prema izračunatom stepenu. Zatim se u domeni zadržavaju samo vrijednosti sa najvećim stepenom, dok ostale vrijednosti se smještaju u listu rezervi. Ovim se postižu dvije prednosti i to:

- skraćuje se vrijeme odabira odgovarajuće vrijednosti iz domena u nastavku procesa kreiranja rasporeda,
- osigurava se da na početku algoritam odabire samo najpoželjnije vrijednosti, tj. da se maksimalno zadovolje definisana varijabilna ograničenja.

Na slici 5.2 prikazan je dio izvornog koda klase *net.etfbl.rg.model.NastavnaJedinica*, odnosno metoda zadužena za generisanje kompletnog domena varijable, te razvrstavanje vrijednosti prema izračunatom stepenu vremenskih i prostornih ograničenja.

```
public void generisiDomenu() {
        trenutnaPreferencaTermina = Kalkulator.PREFERENCA_NEOPHODAN - 1;
        posjeceniTermini.clear();
        slobodniTermini.clear();
        mapaSlobodnihTerminaPoUcioni.clear();
        mapaSlobodnihTerminaPoDanu.clear();
        mapaSlobodnihTerminaPoPreferenci.clear();
        dozvoljeneUcione.forEach((u) -> {
            Map<VremenskiSlot, Integer> mapaAlokacija =
matricaSlotova.crossoverAlokaciju(u.getMatricaSlotova(), true);
            Map<VremenskiSlot, Integer> mapaPreferenci =
matricaSlotova.crossoverPreference(u.getMatricaSlotova(), true);
            List<Termin> sts = Generator.generisiSlobodneTermine(this, u,
matricaUciona.getMapaPreferenciUciona().get(u), mapaAlokacija, mapaPreferenci, null,
trajanje);
            sts.forEach((x) -> {
                List<Termin> yx =
mapaSlobodnihTerminaPoPreferenci.get(x.getPreferencaSlotova());
                if (yx == null) {
                    yx = new LinkedList<>();
                    mapaSlobodnihTerminaPoPreferenci.put(x.getPreferencaSlotova(), yx);
                }
                yx.add(x);
            });
        });
        if (!mapaSlobodnihTerminaPoPreferenci.isEmpty()) {
            prosiriListuSlobodnihTermina();
        }
    }
```

Slika 5.2. Metoda za generisanje domena nastavne jedinice

Prije početka izvršavanja algoritma, odnosno kreiranja domena, za svaku neraspoređenu nastavnu jedinicu su izdvojene dozvoljene prostorije prema definisanim prostornim ograničenjima u ulaznim parametrima.

5.1.2. Odabir varijable – nastavne jedinice

Redoslijed odabira neraspoređene nastavne jedinice, kojoj treba da se dodijeli termin i prostorija, ima značajan utjecaj na ukupno vrijeme potrebno za kreiranje rasporeda nastave. Kako je već spomenuto u opisu pseudo koda, odabir neraspoređene nastavne jedinice se vrši najčešće na osnovu veličine njenog domena. Na slici 5.3 prikazana je metoda iz klase *net.etfbl.rg.model.Raspored* koja vrši odabir neraspoređene nastavne jedinice/varijable.

```
private NastavnaJedinica odaberiNastavnuJedinicu(List<NastavnaJedinica>
dostupneNastavneJedinice) {
        if (!dostupneNastavneJedinice.isEmpty()) {
            Collections.sort(dostupneNastavneJedinice, prioritetRasporedjivanja);
            return dostupneNastavneJedinice.remove(0);
        } else {
            return null;
        }
    }
```

Slika 5.3. Metoda za odabir neraspoređene nastavne jedinice

U komparatoru *prioritetRaspodjivanja* je implementirana logika sortiranja varijabli, nakon čega se odabire prva varijabla iz sortirane liste varijabli.

5.1.3. Odabir vrijednosti iz domena

Nakon što je odabrana varijabla, u sljedećem koraku se vrši odabir vrijednosti iz njenog domena koja će joj biti dodjeljena. Kao i redoslijed odabira varijable, također i redoslijed odabira vrijednosti ima značajan utjecaj na vrijeme potrebno za kreiranje rasporeda nastave. Odabir vrijednosti iz domena se prvenstveno vrši na osnovu utjecaja njene dodjele varijabli na redukciju domena ostalih neraspoređenih varijabli, tj. bira se ona vrijednost sa najmanjom redukcijom domena ostalih varijabli. Pod redukcijom domena smatra se uklanjanje vrijednosti iz domena koje su u koliziji sa tekućom vrijednosti. Na slici 5.4 dat je isječak izvornog koda klase *net.etfbl.rg.model.Raspored*, odnosno metoda koja vrši odabir vrijednosti iz domena odabrane varijable.

```
private Termin odaberiTermin(NastavnaJedinica nj) {
        Termin x = null;
        int ind = 0;
        int pref = nj.getSlobodniTermini().get(ind).getPreferencaSlotova();
        if (!nj.getTerminiPreference().isEmpty()) {
            pref = nj.getTerminiPreference().get(0).getPreferencaSlotova();
            for (Termin xy : nj.getTerminiPreference()) {
                if (!nj.getPosjeceniTerminiPoPreferenci().contains(xy)) {
                    x = xy;
                    break;
                }
            }
            if (x == null) {
                nj.getTerminiPreference().clear();
                boolean pronadjenaSljedecaPreferenca = false;
                Collections.sort(nj.getSlobodniTermini(), preferencaTermina);
                for (int i = 0; i < nj.getSlobodniTermini().size(); i++) {
                    Termin xy = nj.getSlobodniTermini().get(i);
                    if (xy.getPreferencaSlotova() > pref) {
                        pref = xy.getPreferencaSlotova();
                        pronadjenaSljedecaPreferenca = true;
                        ind = i;
                        break;
                    }
                }
                if (!pronadjenaSljedecaPreferenca) {
                    ind = -1;
                }
            }
        }
        if (x == null && ind > -1) {
            for (int i = ind; i < nj.getSlobodniTermini().size(); i++) {
                Termin xy = nj.getSlobodniTermini().get(i);
                if (xy.getPreferencaSlotova() == pref) {
                    xy.getKolizicioneJedinicePoUcioni().clear();
                    xy.getKolizicioneJedinicePoSlotovima().clear();
                    xy.getKolizicioneJedinicePoTipovimaNastave().clear();
                    xy.getKolizicioneJedinicePoVezanimJedinkama().clear();
                    xy.setBrojKolizionihJedinica(brojKolizionihTermina(nj, xy));
                    nj.getTerminiPreference().add(xy);
                } else {
                    break;
                }
            }
            Collections.sort(nj.getTerminiPreference(), brojKolizija);
            x = nj.getTerminiPreference().get(0);
        }
        return x;
    }
```

Slika 5.4. Metoda za odabir termina iz domena neraspoređene nastavne jedinice

5.1.4. Propagacija i *backtrace* dodjele odabrane vrijednosti odabranoj varijabli

Nakon što je uspješno izvršen odabir varijable i vrijednosti iz njenog domena, vrši se propagacija dodjele vrijednosti varijabli kroz neraspoređene varijable. Na slici 5.5 prikazana je rekurzivna metoda iz klase *net.etfbl.rg.Propagator* za propagaciju dodjele odabrane vrijednosti odabranoj varijabli *c*.

```
private int propagirajDodjelu(NastavnaJedinica c) {
```

```
        rasporedjeneJedinice.add(c);
        c.setStanje(NastavnaJedinica.STANJE_RASPOREDJENO);
        if (c.getNastavnik() != null) {
c.getNastavnik().dodajSumVremPrefTermina(c.getNastavnik().sumVremPrefTermina(c.getTermin()));
        }
c.getStudijskaGrupa().dodajSumVremPrefTermina(c.getStudijskaGrupa().sumVremPrefTermina(c.getTe
rmin()));
        backtraceList.add(c);
        List<NastavnaJedinica> njZaPropagaciju = new LinkedList<>();
        for (NastavnaJedinica nj : c.getTermin().getKolJedPoUcioni()) {
            if (!nj.isGrupisana()) {
                nj.propagirajZauzeceTerminaPoUcioni(c.getTermin());
                if (nj.getSlobodniTermini().isEmpty()) {
                    if (!propagiranaNJ.getDeadEndJedinice().contains(nj)) {
                        propagiranaNJ.getDeadEndJedinice().add(nj);
                    }
                    return STANJE_DEAD_END;
                } else if (nj.getSlobodniTermini().size() == 1) {
                    njZaPropagaciju.add(nj);
                }
            }
        }
        for (NastavnaJedinica nj : c.getTermin().getKolJedPoSlotovima()) {
            if (!nj.isGrupisana()) {
                nj.propagirajZauzeceTerminaPoSlotovima(c.getTermin());
                if (nj.getSlobodniTermini().isEmpty()) {
                    if (!propagiranaNJ.getDeadEndJedinice().contains(nj)) {
                        propagiranaNJ.getDeadEndJedinice().add(nj);
                    }
                    return STANJE_DEAD_END;
                } else if (nj.getSlobodniTermini().size() == 1 &&
!njZaPropagaciju.contains(nj)) {
                    njZaPropagaciju.add(nj);
                }
            }
        }
        for (NastavnaJedinica nj : c.getTermin().getKolJedPoTipNastave()) {
            if (!nj.isGrupisana()) {
                nj.propagirajZauzeceTerminaPoTipovimaNastave(c);
                if (nj.getSlobodniTermini().isEmpty()) {
                    if (!propagiranaNJ.getDeadEndJedinice().contains(nj)) {
                        propagiranaNJ.getDeadEndJedinice().add(nj);
                    }
                    return STANJE_DEAD_END;
                } else if (nj.getSlobodniTermini().size() == 1 &&
!njZaPropagaciju.contains(nj)) {
                    njZaPropagaciju.add(nj);
                }
            }
        }
        for (NastavnaJedinica nj : c.getTermin().getKolJedPoVezJedinicama()) {
            if (!nj.isGrupisana()) {
                nj.propagirajZauzeceTerminaPoVezanimJedinicama(c);
                if (nj.getSlobodniTermini().isEmpty()) {
                    if (!propagiranaNJ.getDeadEndJedinice().contains(nj)) {
                        propagiranaNJ.getDeadEndJedinice().add(nj);
                    }
                    return STANJE_DEAD_END;
                } else if (nj.getSlobodniTermini().size() == 1 &&
!njZaPropagaciju.contains(nj)) {
                    njZaPropagaciju.add(nj);
                }
            }
        }
        if (njZaPropagaciju.isEmpty()) {
            return STANJE_SUCCESS;
        } else {
            for (NastavnaJedinica nj : njZaPropagaciju) {
                raspored.getNerasporedjeneNastavneJedinice().remove(nj);
                Termin minTermin = nj.getSlobodniTermini().get(0);
                minTermin.getKolJedPoSlotovima().clear();
                minTermin.getKolJedPoUcioni().clear();
                minTermin.getKolJedPoTipNastave().clear();
                minTermin.getKolJedPoVezJedinicama().clear();
```

```
                minTermin.setBrojKolizionihJedinica(raspored.brKolTermina(nj, minTermin));
                nj.setTermin(minTermin);
                if (propagirajDodjelu(nj) == STANJE_DEAD_END) {
                    return STANJE_DEAD_END;
                }
            }
        }
        return STANJE_SUCCESS;
    }
```

Slika 5.5. Metoda za propagaciju dodjele odabranog termina odabranoj neraspoređenoj nastavnoj jedinici

Kao što se može vidjeti u prikazanom isječku koda, rezultat propagacije može biti *STANJE_DEAD_END* ili *STANJE_SUCCESS*. U slučaju da nakon propagacije, odnosno redukcije domena obuhvaćene varijable, njen domen ostane prazan, tad je rezultat propagacije *STANJE_DEAD_END*. U slučaju da u reduciranom domenu ostane samo jedna vrijednost, proces propagacije se rekurzivno ponavlja za dodjelu te preostale vrijednosti pripadajućoj varijabli, čiji rezultat će predstavljati rezultat tekuće propagacije.

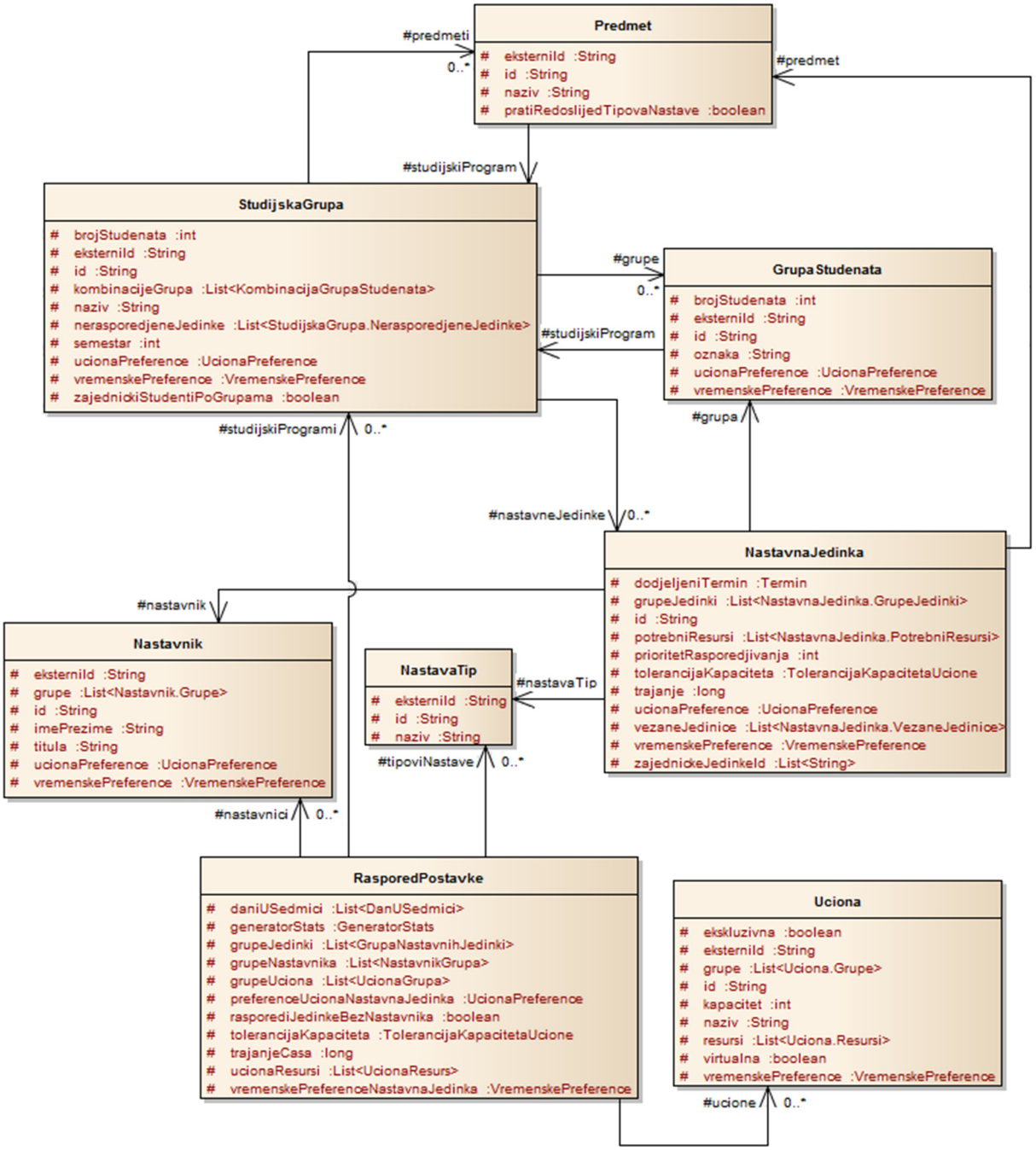
Kao što je prikazano u pseudo kodu sa slike 5.1, u slučaju *deadend* situacije vrši se *backtrace*, koji ima zadatak da poništi sve dodjele odabrane vrijednosti odabranoj varijabli, te da vrati redukcijom/propagacijom uklonjene vrijednosti u pripadajući domen. Na slici 5.6 prikazana je metoda iz klase *net.etfbl.rg.Propagator* zadužena za *backtrace* propagirane varijable *c*.

```
public void backtrace(NastavnaJedinica c) {
        c.getTermin().getKolJedPoUcioni().forEach((nj) -> {
            nj.backtraceZauzeceTerminaPoUcioni(c.getTermin());
        });
        c.getTermin().getKolJedPoSlotovima().forEach((nj) -> {
            nj.backtraceZauzeceTerminaPoSlotovima(c.getTermin());
        });
        c.getTermin().getKolJedPoTipNastave().forEach((nj) -> {
            nj.backtraceZauzeceTerminaPoTipovimaNastave(c.getTermin());
        });
        c.getTermin().getKolJedPoVezJedinicama().forEach((nj) -> {
            nj.backtraceZauzeceTerminaPoVezanimJedinkama(c.getTermin());
        });
        c.setStanje(NastavnaJedinica.STANJE_NERASPOREDJENO);
        c.getTermin().getKolJedPoSlotovima().clear();
        c.getTermin().getKolJedPoUcioni().clear();
        c.getTermin().getKolJedPoTipNastave().clear();
        c.getTermin().getKolJedPoVezJedinicama().clear();
        if (c.getNastavnik() != null) {
c.getNastavnik().ukloniSumVremPrefTermina(c.getNastavnik().sumVremPrefTermina(c.getTermin()));
        }
c.getStudijskaGrupa().ukloniSumVremPrefTermina(c.getStudijskaGrupa().sumVremPrefTermina(c.getT
ermin()));
        c.setTermin(null);
    }
```

Slika 5.6. Metoda za izvršavanje backtrace-a

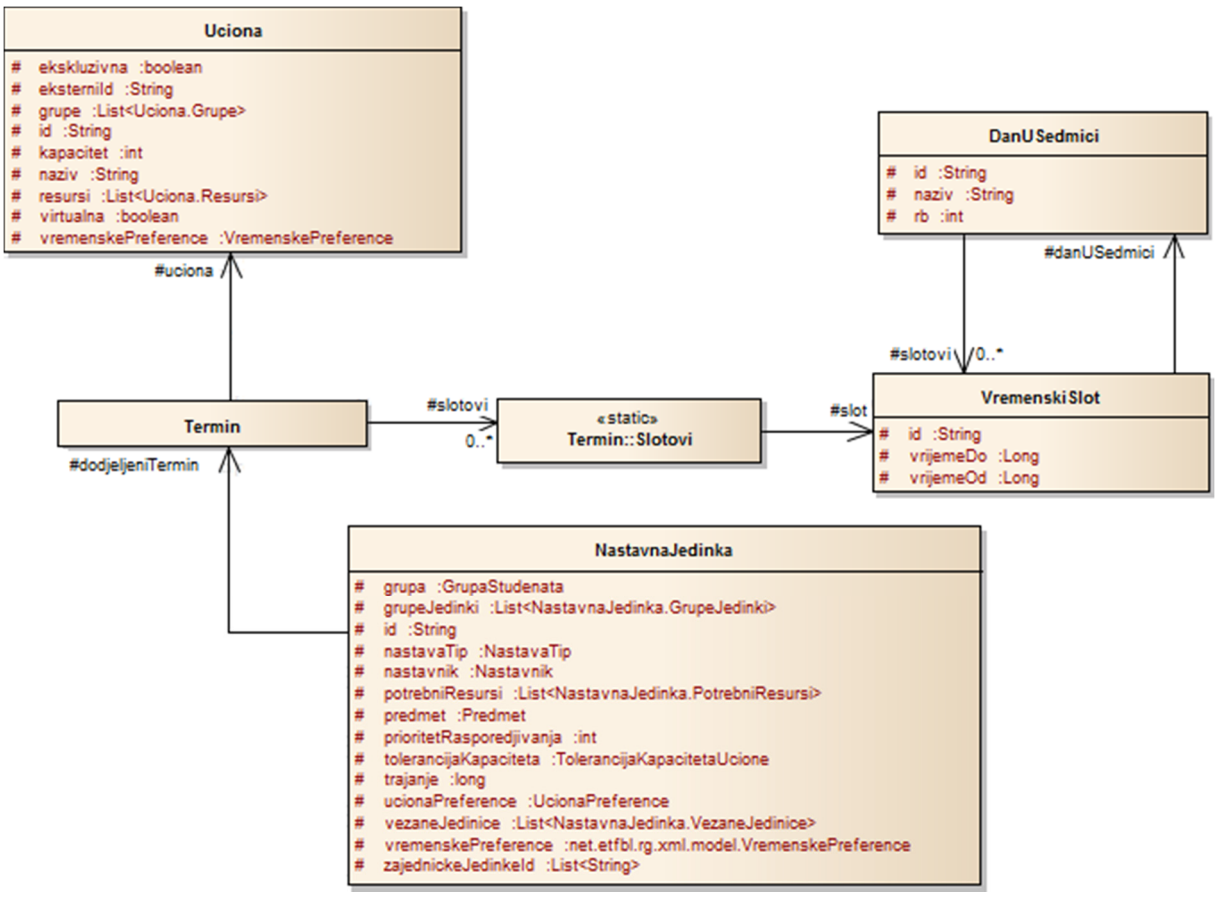
5.2. Struktura ulaznih parametara i rezultata kreiranog rasporeda

Ulazni parametri algoritmu se dostavljaju u XML formatu propisanim XML šemom datoj u prilogu 2 ovog rada. U sklopu implementacije algoritma izvršena je konverzija pomenute XML šeme u Java klase korištenjem Java Architecture XML Binding API (JAXB) *framework-a*. Na slici 5.7 prikazan je dijagram klasa sa najbitnijim strukturnim klasama. Da bi ulazni parametri bili validni, bitno je uočiti da svi strukturni i većina nestrukturnih elemenata sadrži atribut ID (eng. *identifier*), čija vrijednost mora biti definisana i jedinstvena na nivou tog elementa. Pored ovog atributa, većina elemenata sadrži i atribut *eksterniID*, čija vrijednost služi kao poveznica tog elementa sa elementom iz eksternog sistema, te omogućuje uvoz i izvoz elemenata u/iz eksternog sistema. Nakon generisanja rasporeda nastave, pri kojem se definiše vrijednost atributa *dodjeljeniTermin* u klasi *NastavnaJedinka*, primjenom JAXB *framework-a* vrši se serijalizacija *top-level* objekta klase *RasporedPostavke* u XML format. Ukoliko se algoritam koristi u obliku web servisa, tada se ovaj XML sadržaj vraća kao rezultat generisanja rasporeda.



Slika 5.7. Dijagram klasa ulaznih parametara (strukturni elementi)

Na slici 5.8 prikazan je dodatak dijagramu klasa koji se odnosi na strukturu dodijeljenog termina nastavnoj jedinici.



Slika 5.8. Dijagram klasa kreiranog rasporeda

Pored podataka o dodjeljenim terminima raspoređenih nastavnih jedinica, u objekt klase *RasporedPostavke* se upisuju i dodatne informacije, kao npr. lista nastavnih jedinica koje nisu uspješno raspoređene sa odgovarajućom porukom odnosno razlogom, ukupno vrijeme izvršavanje algoritma, broj *dead-end* situacija, brzina raspoređivanja itd.

6. PRIMJER UPOTREBE IMPLEMENTIRANOG RJEŠENJA

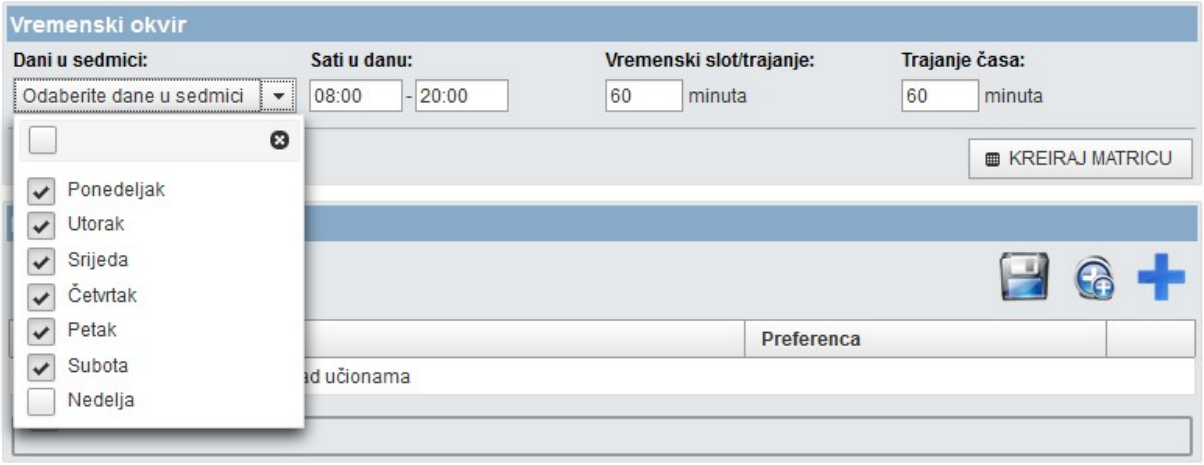
Kao primjer upotrebe implementiranog rješenja poslužiće modul informacionog sistema Univerziteta u Bihaću napisan u Java programskom jeziku. Ovaj modul zapravo predstavlja podmodul postojećeg modula zaduženog za upravljanje i ručno kreiranje rasporeda nastave. Pristup ovom modul je prvenstveno omogućen prodekanima za nastavu i rukovodiocima studijskih programa, da bi mogli na veoma brz i efikasan način kreirati ulazne parametre, definisati ograničenja, generisati raspored nastave, i na kraju dobijeni raspored uvesti u bazu podataka informacionog sistema, a koji će dalje isti distribuirati ka krajnjim korisnicima.

6.1. Kreiranje ulaznih parametara

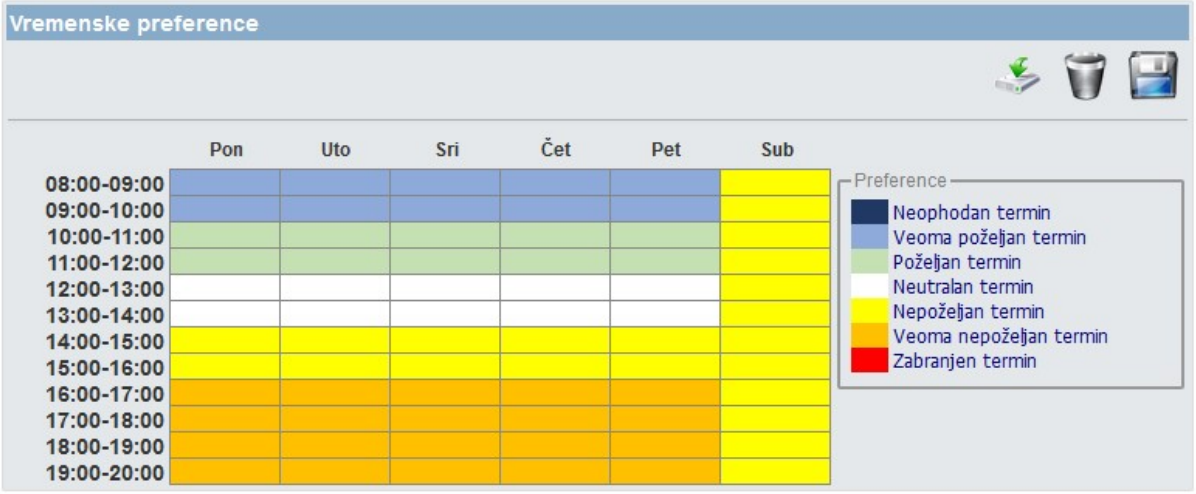
Kreiranje ulaznih parametara za kreiranje rasporeda nastave predstavlja vremenski najzahtjevniju fazu u čitavom procesu kreiranju rasporeda. Samim tim, svaki oblik automatizacija kreiranja ovih parametara ubrzava proces. Integracijom novog modula za kreiranje ulaznih parametara u informacioni sistem, a koji već sadrži većinu podataka za kreiranja ulaznih parametara, omogućen je jednostavaniji i brži način definisanje istih. Kao primjer generisanja rasporeda, opisan je kompletan proces kreiranja rasporeda za ljetni semestar 2017/2018. godine za I ciklus studija na Tehničkom fakultetu Univerziteta u Bihaću.

Prvi korak u kreiranju ulaznih parametara jeste definisanje vremenskog okvira unutar kojeg će se vršiti raspoređivanje nastavnih jedinica, što podrazumjeva odabir dana u sedmici, trajanje vremenskog slota, trajanje časa, te dnevnu satnicu. Web forma za definisanje vremenskog okvira data je na slici 6.1.

Nakon definisanja vremenskog okvira, sistem generiše matricu vremenskih slotova, što omogućuje definisanje kako globalnih tako i ostalih vremenskih ograničenja nad elementima prikazanim na slici 4.2. Na slici 6.2 je prikazana web forma za definisanje globalnih vremenskih ograničenja za ovaj primjer ulaznih parametara.



Slika 6.1. Web forma za definisanje vremenskog okvira



Slika 6.2. Web forma za definisanje vremenskih ograničenja

U tabeli 6.1 dat je detaljan pregled ulaznih parametara korištenih u ovom primjeru upotrebe implementiranog rješenja.

Tabela 6.1. Pregled ulaznih parametara primjera upotrebe implementiranog rješenja

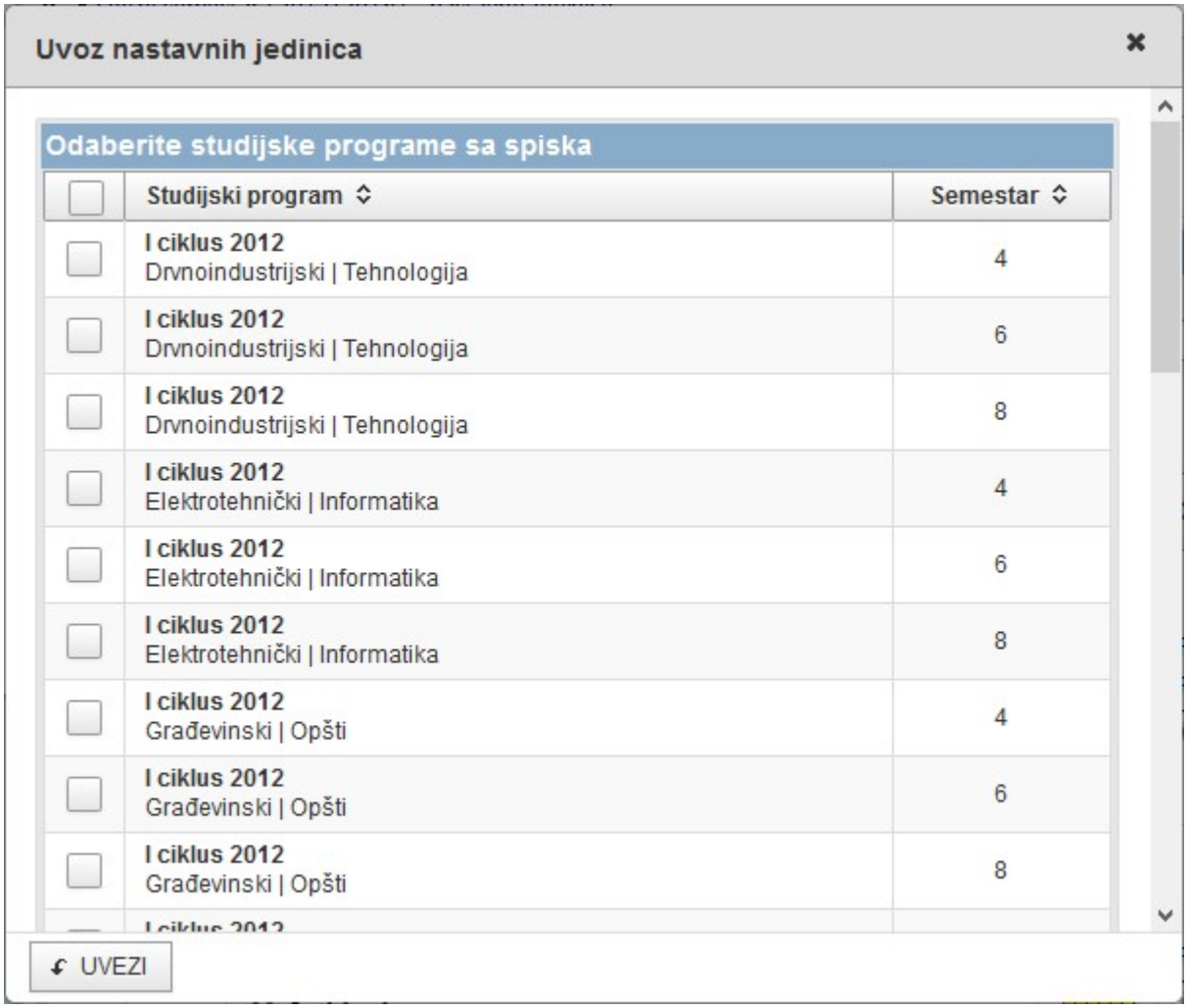
	Ljetni semestar 2017/2018	
Ukupan broj nastavnih jedinica	252	
Broj zajedničkih jedinica	**Br. grupa**	**Br. jedinica**
	15	55
Broj kontinuiranih nastavnih jedinica	**Br. grupa**	**Br. jedinica**
	5	10
Broj nastavnih jedinica za raspoređivanje	212	
Broj nastavnika	75	
Broj studijskih grupa	20	
Broj prostorija	15	
Prosječan broj dostupnih prostorija po nastavnoj jedinici	4.40	
Prosječan broj nastavnih jedinica po nastavniku	2.86	

6.1.1. Strukturni elementi ulaznih parametara

Za svaki od strukturnih elemenata ulaznih parametara sa slike 5.1, implementirana je web forma za kreiranje istog i definisanje pratećih postavki elementa.

Nastavna jedinica

Nastavne jedinice u ulaznim parametrima je takođe moguće ručno kreirati, stim da ćemo u ovom primjeru koristiti mogućnosti uvoza podataka iz drugog modula informacionog sistema, odnosno akademskog kalendara. U akademskom kalendaru su, pored vremenskih okvira trajanja nastave zimskog i ljetnog semestra, te ispitnih termina, evidentirane i studijske grupe na kojima se odvija nastava u toku akademske godine. Samim tim, nastavne jedinice će se automatski kreirati odabirom studijskih grupa iz akademskog kalendara, što je prikazano na slici 6.3.



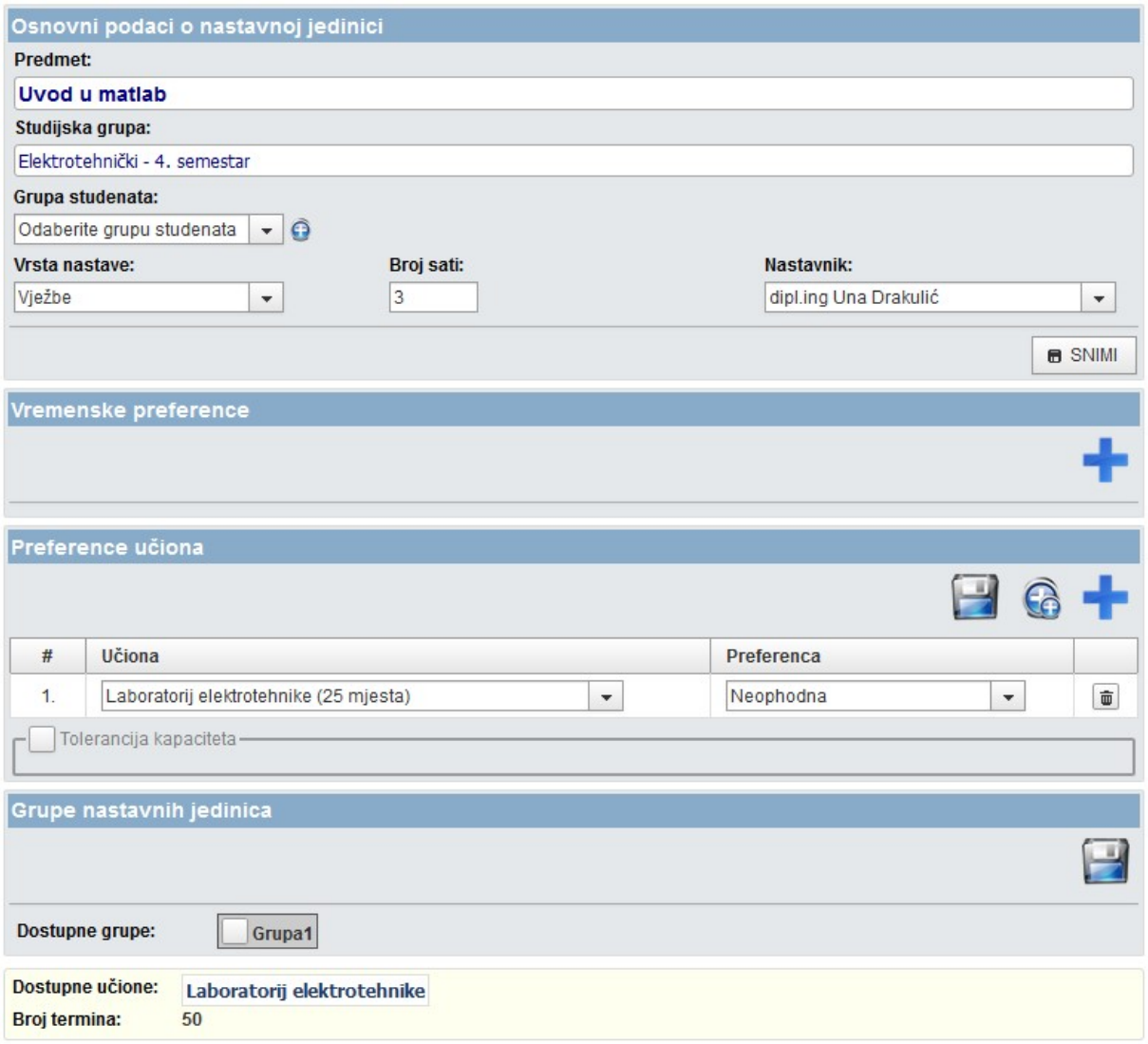
Slika 6.3. Web forma za odabir studijskih grupa za uvoz nastavnih jedinica

Kao što je vidljivo sa slike 6.3, studijska grupa u ovom slučaju predstavlja poveznicu između studijskog programa i semestra. Takođe nastavni predmeti u informacionom sistemu vezani su za pripadajući studijski program i semestar. Generisanje nastavnih jedinica u ulaznim parametrima se vrši na osnovu evidentirane nastavne opterećenosti predmeta u sistemu (sedmični broj sati predavanja i vježbi) i odabranih studijskih grupa. Na slici 6.4 prikazana je web forma za kreiranje nove ili uređivanje postavki postojeće nastavne jedinice.

Studijska grupa

Studijska grupa je strukturni element koji se ne može kreirati ručno u ovom modulu, već se on automatski kreira pri uvozu nastavnih jedinica iz akademskog kalendara ili ručnim kreiranjem nastavne jedinice. Međutim, nakon njegovog kreiranja, preko implementirane web forme, prikazane na slici 6.5, moguće je urediti postavke postojeće studijske grupe. U ovom

primjeru, uvozom nastavnih jedinica, kreirano je 20 studijskih grupa, tj. 5 odsjeka, po 4 semestra (2., 4., 6. i 8.).



Slika 6.4. Web forma za kreiranje nove i uređivanje postavki postojeće nastavne jedinice

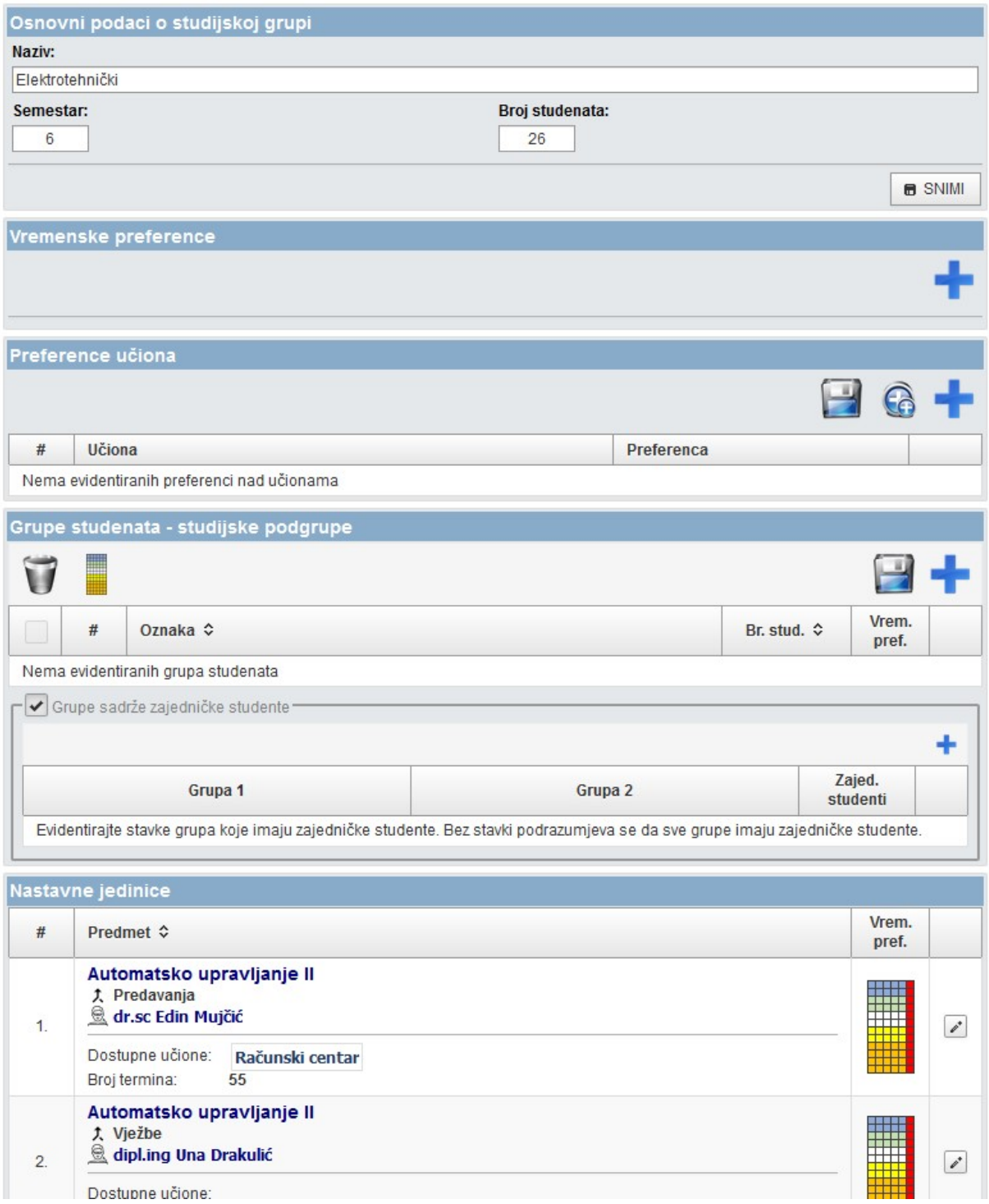
Nastavnik

Prilikom uvoza nastavnih jedinica iz akademskog kalendara, takođe se uvoze i pripadajući nastavnici kojima su, prema evidentiranoj pokrivenosti nastave u drugom modulu informacionog sistema, dodijeljena vrsta/e nastave (predavanja, vježbe, itd.) iz uvezenih predmeta. Samim tim, nema potrebe za dodatnim evidentiranjem nastavnika u ulaznim parametrima, niti dodijeljivanje nastavnih jedinica istim. Na slici 6.6 prikazana je web forma za uređivanje postavki nastavnika u ulaznim parametrima.

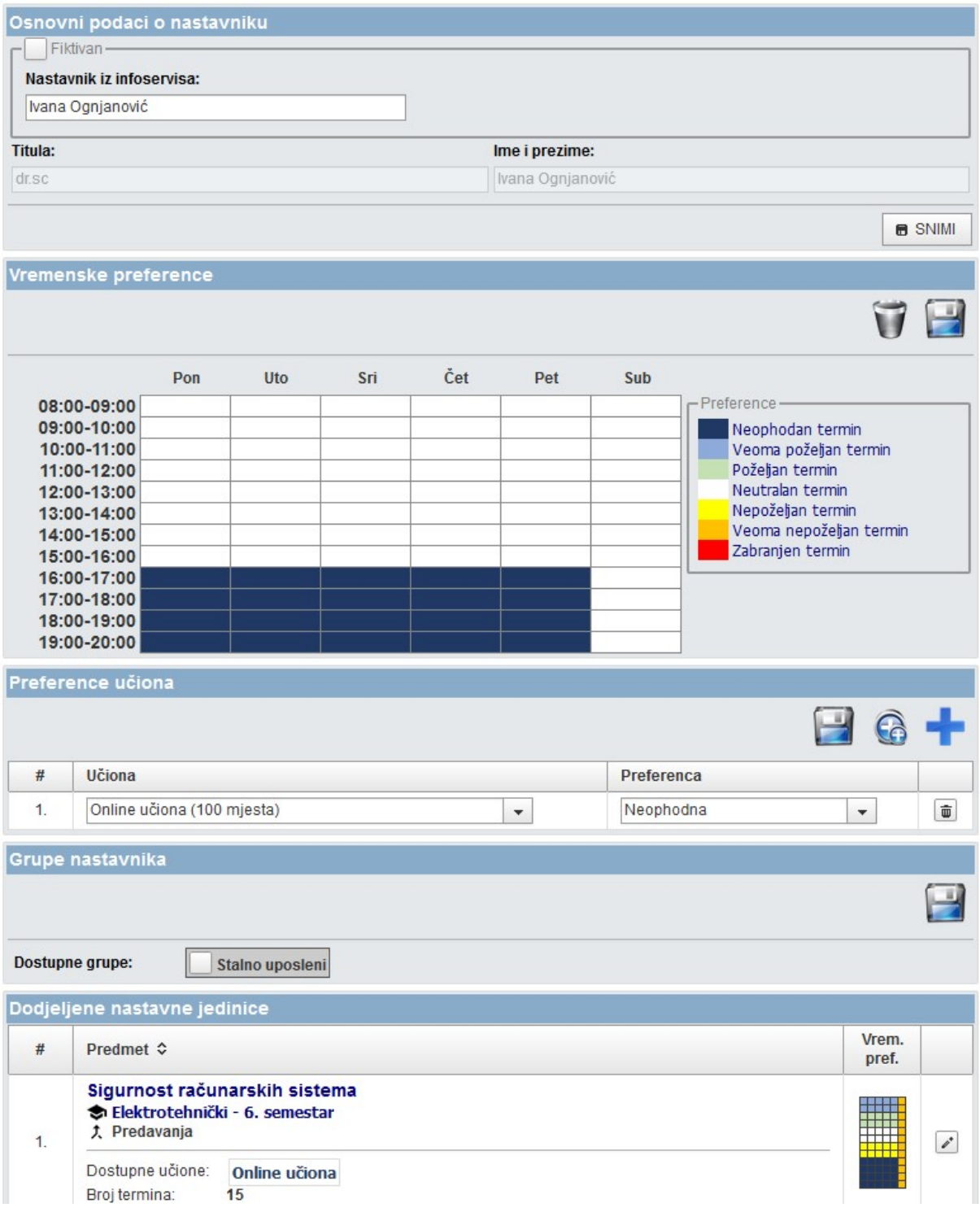
Takođe je moguće kreirati i fiktivnog nastavnika, tj. nastavnika koji nije uvezen iz informacionog sistema, npr. ako u trenutku kreiranja rasporeda nastave nije definisana pokrivenost nastave za određenu nastavnu jedinicu.

Grupa nastavnika

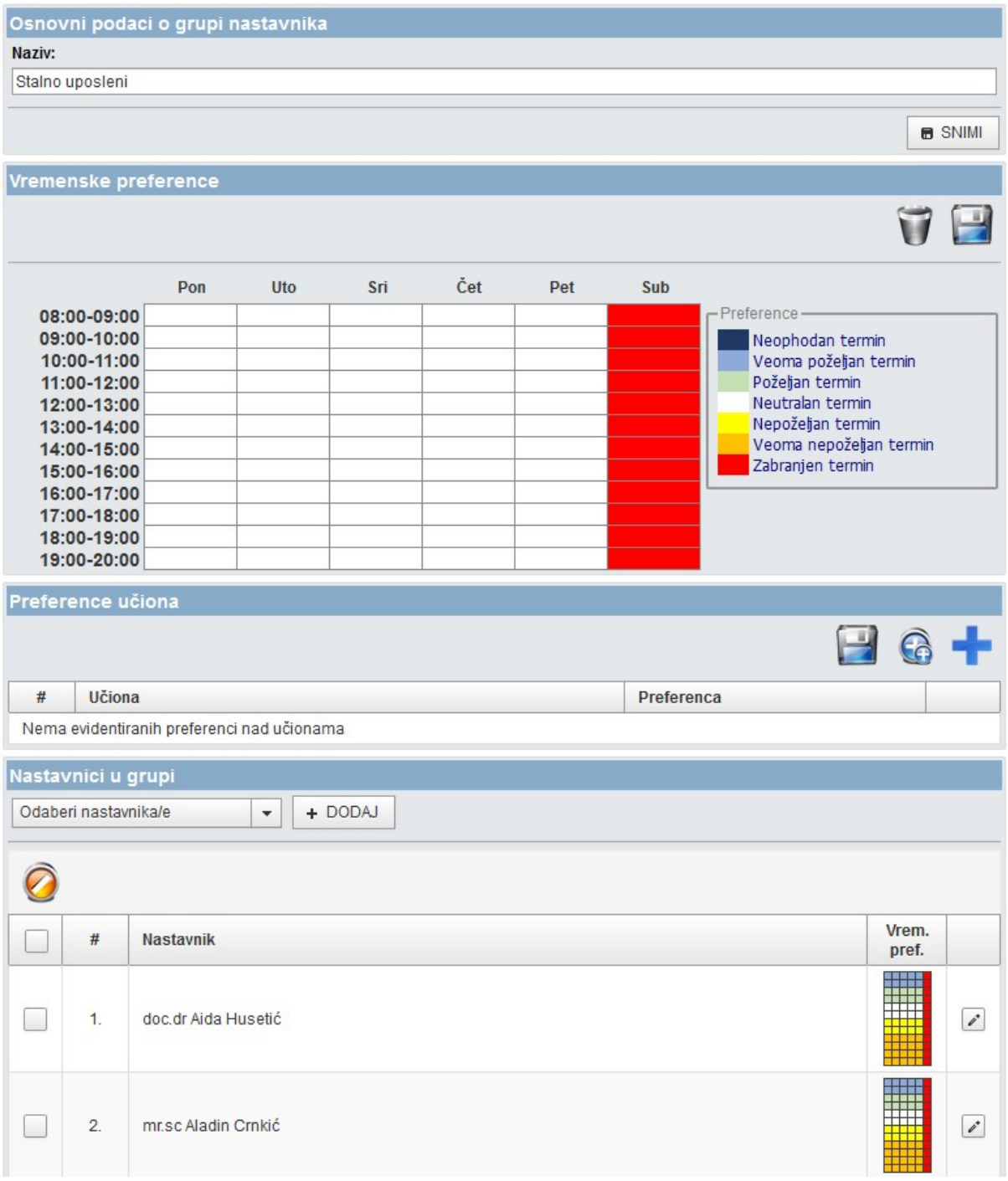
Grupisanje nastavnika je prvenstveno namijenjeno lakšem definisanju zajedničkih postavki za određenu grupu nastavnika, te samim tim se ne moraju pojedinačno definisati iste postavke za svakog nastavnika. Na ovaj način dovoljno je definisati postavke nad grupom, a koje se, prema distribuciji ograničenja sa slika 4.2 i 4.3, prenose na sve nastavnike iz grupe. Na slici 6.7 prikazana je web forma za uređivanje postavki grupe nastavnika.



Slika 6.5. Web forma za uređivanje postavki postojeće studijske grupe



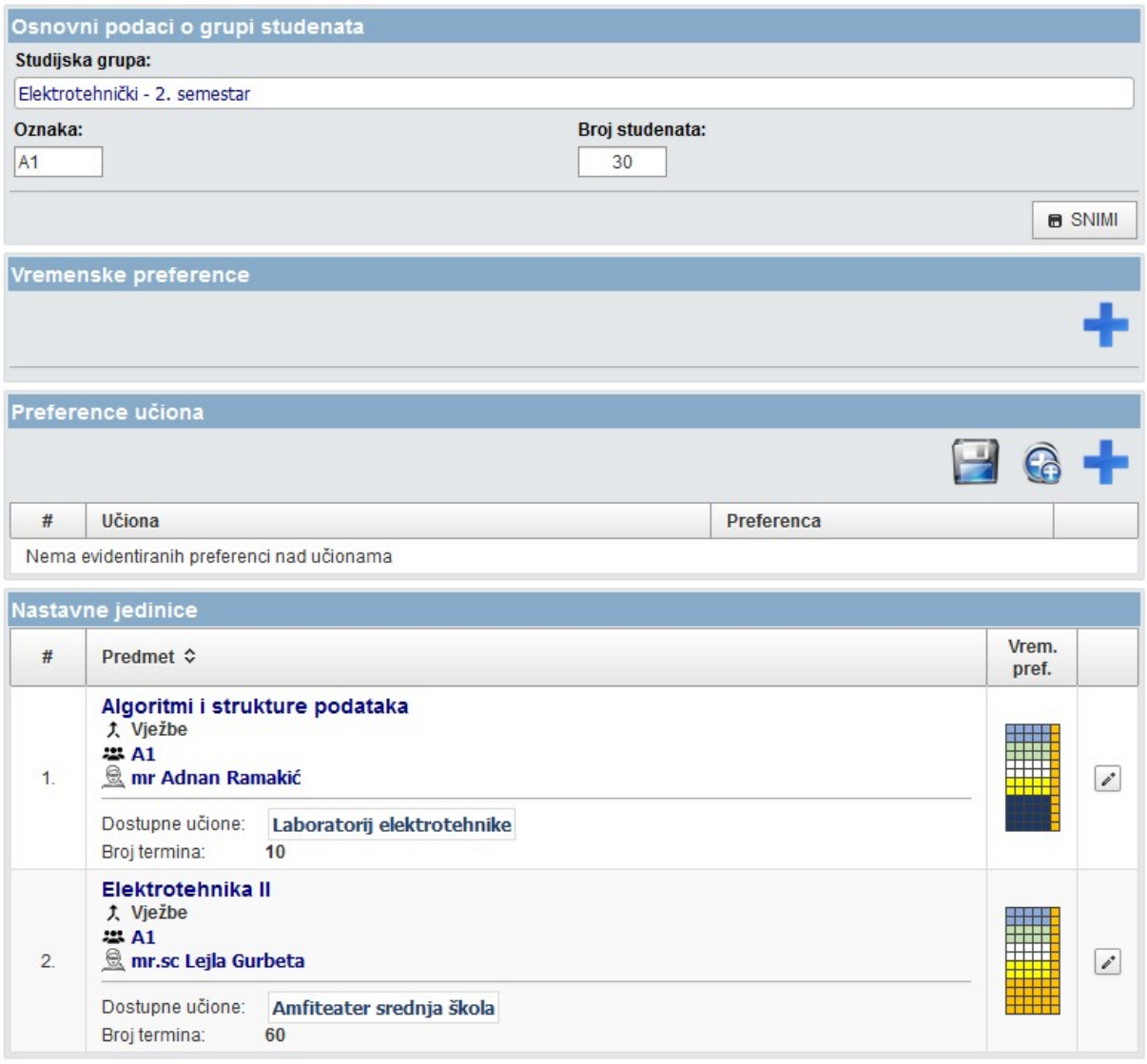
Slika 6.6. Web forma za uređivanje postavki nastavnika



Slika 6.7. Web forma za uređivanje grupe nastavnika

Grupa studenata

Grupa studenata predstavlja studijsku podgrupu. Nad istom se mogu definisati i vremenska i prostorna ograničenja kao i nad studijskom grupom. Na slici 6.8 prikazana je web forma za uređivanje pomenutih postavki nad grupom studenata.



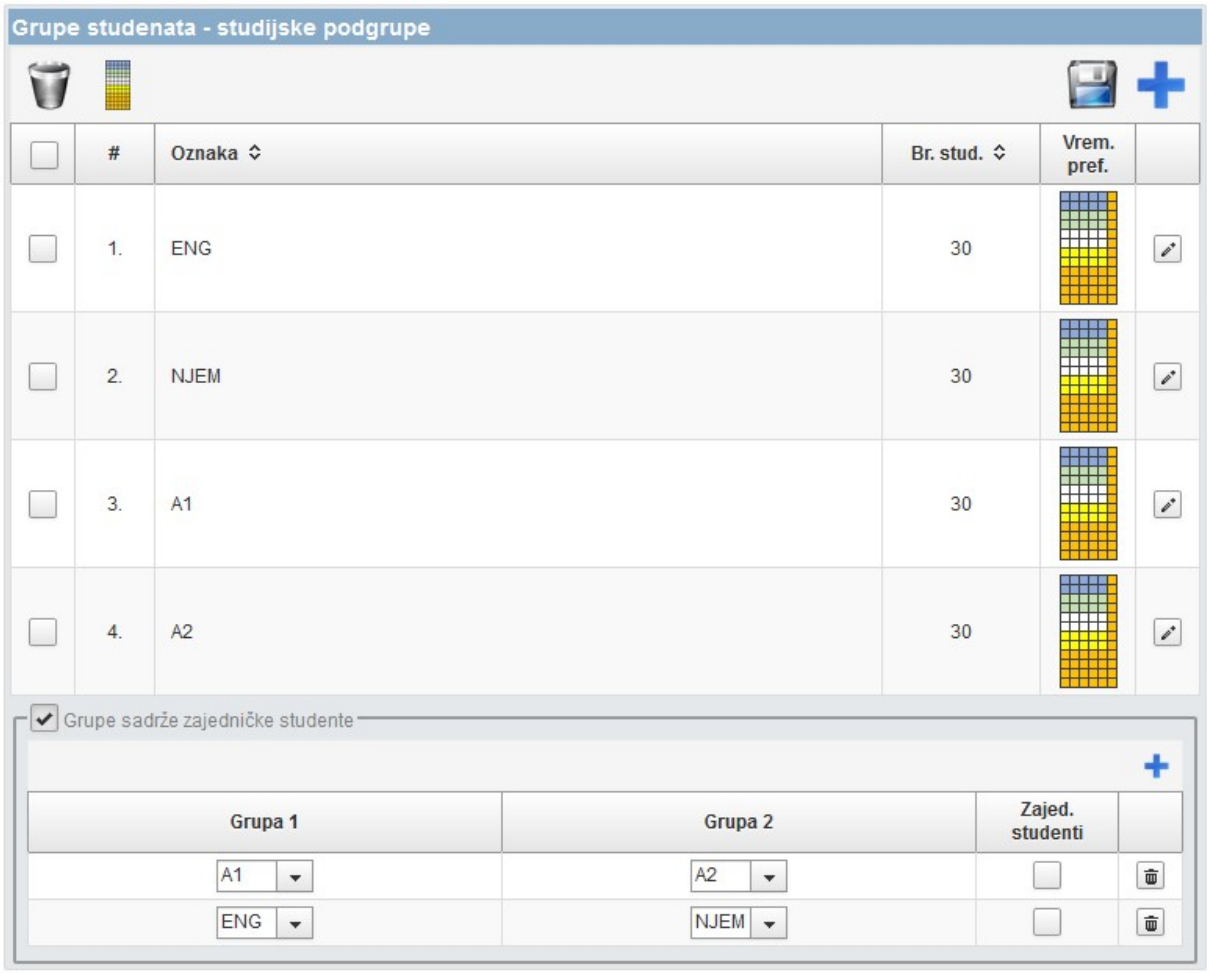
Slika 6.8. Web forma za uređivanje grupe studenata

Pored postavki sa slike 6.8, u web formi sa slike 6.5, moguće je definisati i kombinacije grupa studenata koje sadrže, odnosno ne sadrže zajedničke studente. Sa ovim je omogućeno efikasnije raspoređivanje nastavnih jedinica na način da grupe koje ne sadrže iste studente i imaju različite nastavnike, mogu se rasporediti u isto vrijeme. Na slici 6.9 prikazan je segment web forme sa slike 6.5 zadužen za definisanje pomenutih kombinacija grupa.

Prostorija

Prostorija predstavlja resurs koji se pri raspoređivanju dodjeljuje nastavnoj jedinici. Pri kreiranju ulaznih parametara, automatski se vrši uvoz evidentiranih prostorija iz informacionog sistema. Takođe je moguće kreirati i fiktivnu prostoriju, odnosno prostoriju koja nije

evidentirana u informacionom sistemu. Ovdje treba napomenuti da, pri uvozu generisanog rasporeda nastave nazad u informacioni sistem, nastavne jedinice koje su raspoređene u fiktivne prostorije neće biti uvezene. Na slici 6.10 je prikazana web forma za uređivanje postavki prostorije. Postavkom „Ekskluzivna učiona“ se uključuje/isključuje ekskluzivitet prostorije pomenutom u sekciji 4.2.2. Postavkom „Virtuelna učiona“ prostorija se može označiti virtuelnom, a namjena ove postavke je obrazložena u sekciji 4.2.5.



Slika 6.9. Segment web forme studijske grupe za kreiranje i kombiniranje grupa studenata

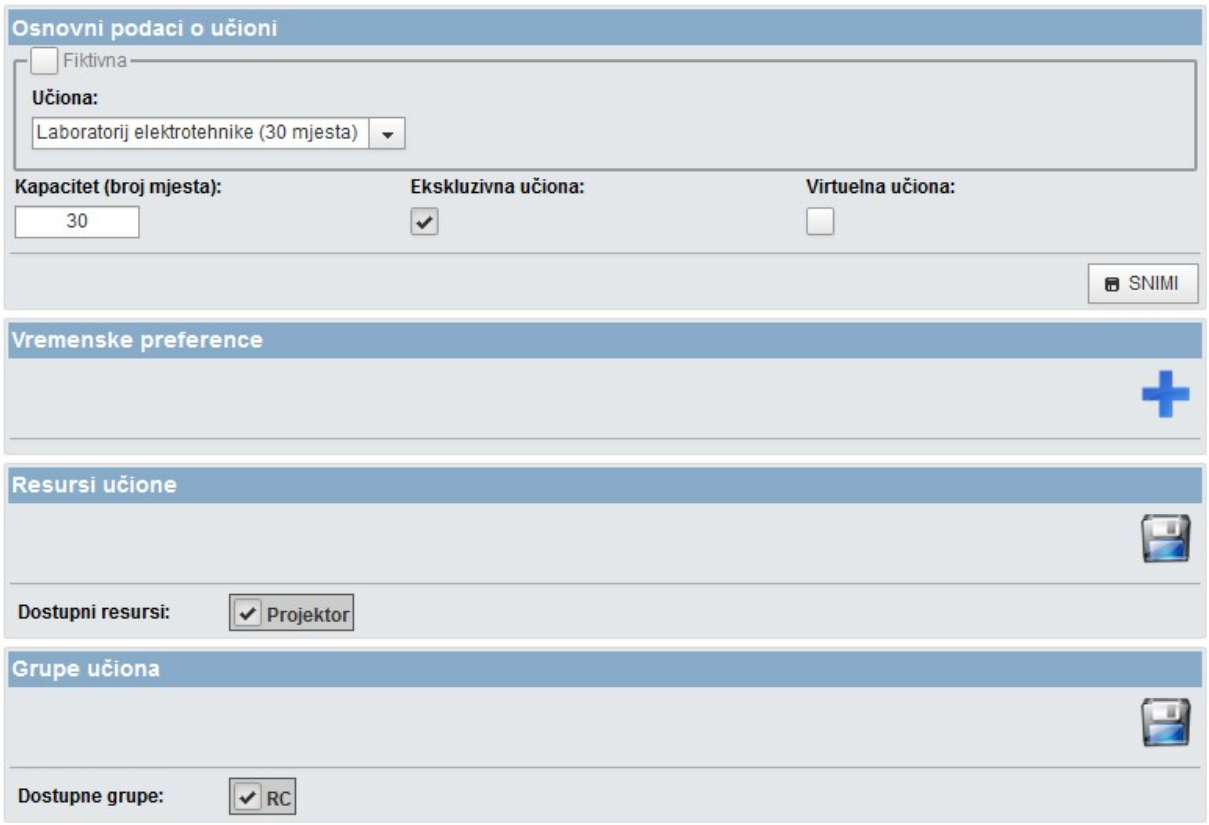
Prostorni resursi

U postavkama prostorije moguće je evidentirati i popis resursa koje prostorija sadrži, npr. projektor, video link, itd. Na slici 6.11 prikazana je web forma za uređivanje postavki prostornog resursa. Za svaki od resursa je moguće definisati i vremenska ograničenja za korištenje istih. Isto tako, u postavkama nastavne jedinice može se, umjesto prostornih ograničenja, evidentirati lista neophodnih prostornih resursa, prema kojima će se filtrirati

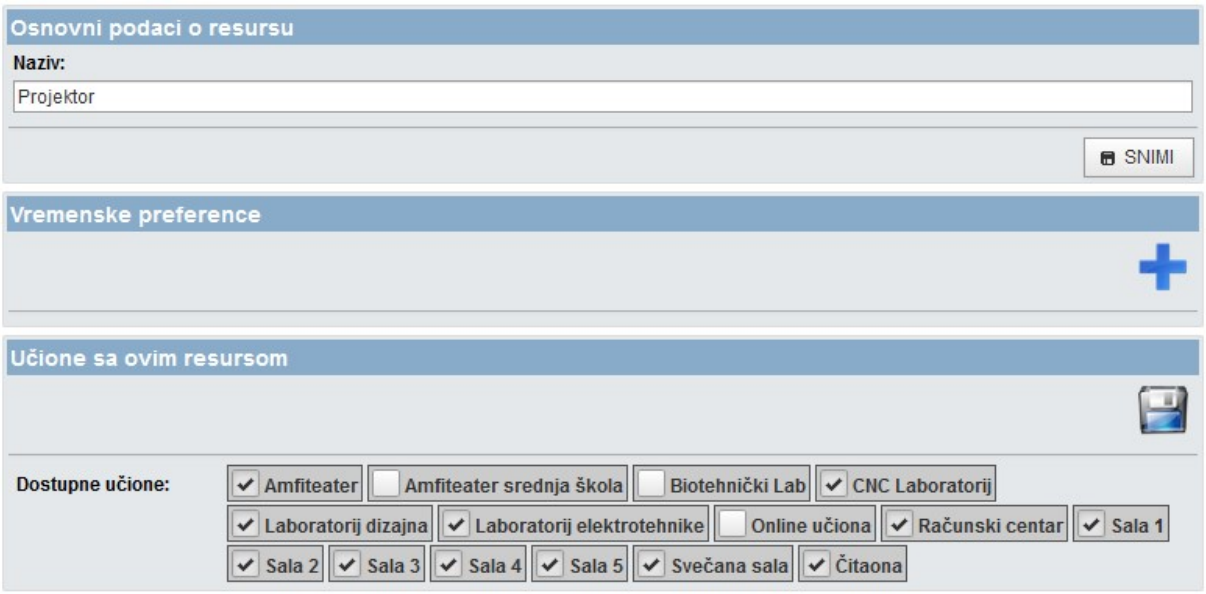
prostorije u kojim se nastavna jedinica može rasporediti, odnosno odabrati one prostorije koje sadrže definisane prostorne resurse.

Grupa prostorija

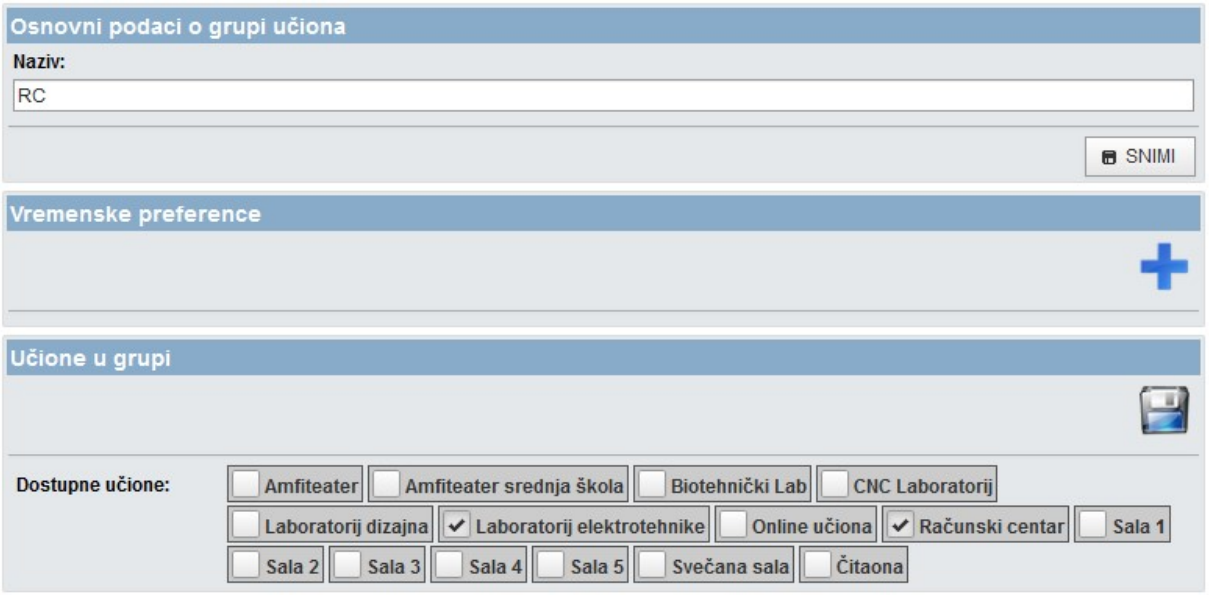
Grupisanje prostorija omogućuje brže definisanje vremenskih ograničenja nad više prostorija. Ukoliko dvije ili više prostorija trebaju da imaju ista vremenska ograničenja, dovoljno ih je grupisati i onda nad grupom definisati vremenska ograničenja koje se preslikavaju na sve prostorije u grupi prema dijagramu distribucije vremenskih ograničenja prikazanog na slici 4.2. Takođe, pri definisanju prostornih ograničenja nad npr. nastavnom jedinicom, moguće je, umjesto pojedinačnog definisanja ograničenja za svaku prostoriju, definisati ograničenje nad grupom učionica. Na slici 6.12 je prikazana web forma za uređivanje postavki grupe prostorija.



Slika 6.10. Web forma za uređivanje postavki prostorije



Slika 6.11. Web forma za uređivanje postavki prostornog resursa



Slika 6.12. Web forma za uređivanje postavki grupe prostorija

Tolerancija kapaciteta učione

Iako se prostorna ograničenja mogu definisati nad mnogim elementima raspoređivanja, prikazanim na slici 4.3, kapacitet prostorije ima ključnu ulogu pri filtriranju prostorija u kojim se može izvoditi nastava na osnovu broja studenata koji je pohađaju. Samim tim, da bi se ovo ograničenje moglo fleksibilnije regulisati, omogućeno je definisanje tolerancije kapaciteta prostorije nad čitavim rasporedom ili nad nastavnom jedinicom. Tolerancija se može definisati

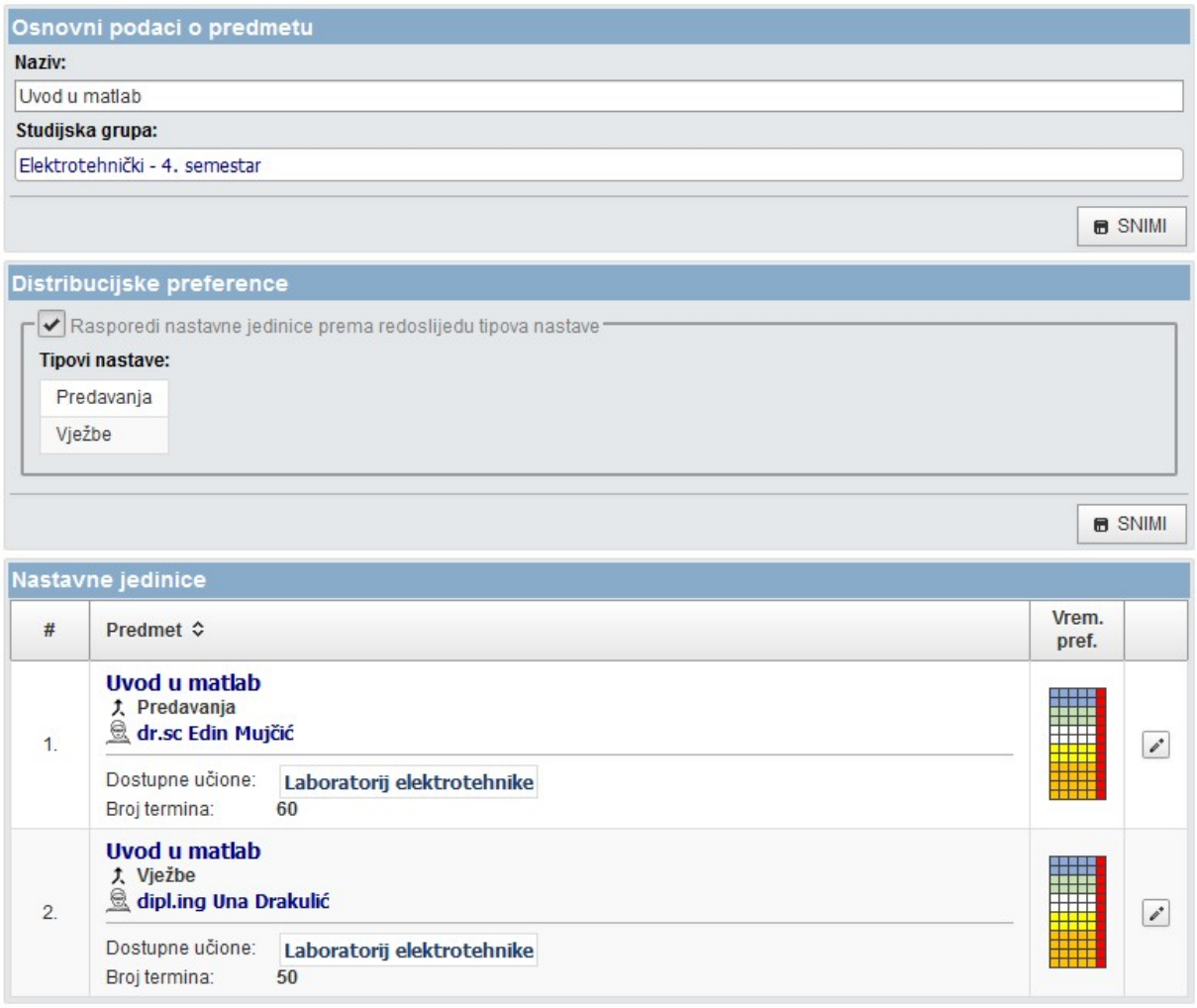
procentualno ili brojčano. Npr. ako je kapacitet učione 100 mjesta, a definisana tolerancija je 10%, tada će se u ovu učionu moći rasporediti nastavne jedinice koje pohađa maksimalno 110 studenata. Isti slučaj bi bio kada bi se definisala tolerancija od 10 mjesta, odnosno brojčano.

Predmet

Za predmete u ulaznim parametrima važi isto pravilo kao i za studijske grupe, tj. isti se kreiraju pri kreiranju nastavnih jedinica. Na slici 6.13 prikazana je web forma za uređivanje postavki predmeta.

Distribucijska ograničenja

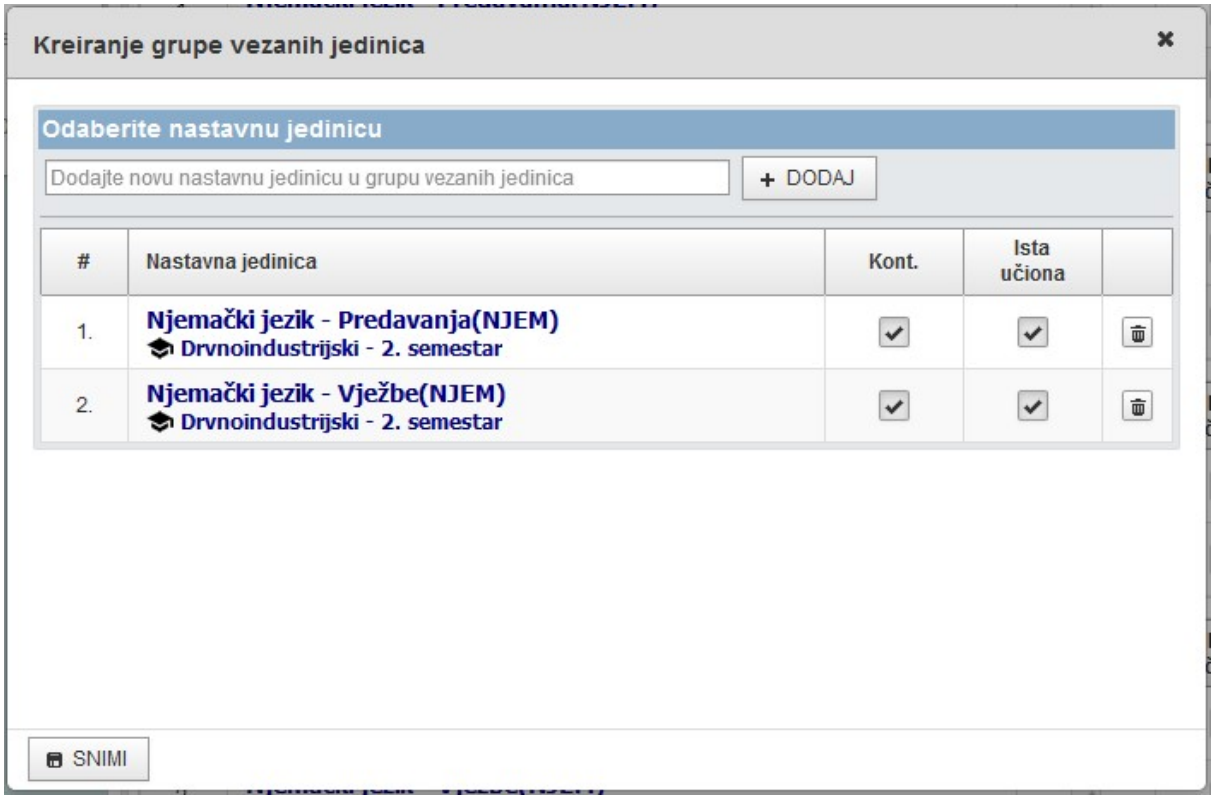
Sa distribucijskim ograničenjima je moguće definisati kontinuitet u odabranim terminima pri raspoređivanju dvije ili više nastavnih jedinica opisano u sekciji 4.2.4, te zajedničko izvođenje nastave za dvije ili više nastavnih jedinica odnosno raspoređivanje istih u isti termin pri kreiranju rasporeda opisano u sekciji 4.2.3. U ovom primjeru definisano je kontinuirano izvođenje nastave predavanja i vježbi iz predmeta „Njemački jezik“. Na slici 6.14 prikazana je web forma za kreiranje grupe vezanih/kontinuiranih nastavnih jedinica. Pri definisanju grupe vezanih nastavnih jedinica može se i dodatno definisati da li će se nastavne jedinice rasporediti u kontinuiranim terminima (kolona *Kont.*), te da li će se rasporediti u istu učionu (kolona *Ista učiona*). Ukoliko je isključena opcija kontinuiteta, onda će se vezane nastavne jedinice rasporediti u vremenskom periodu prema redoslijedu kako su prikazane u grupi.



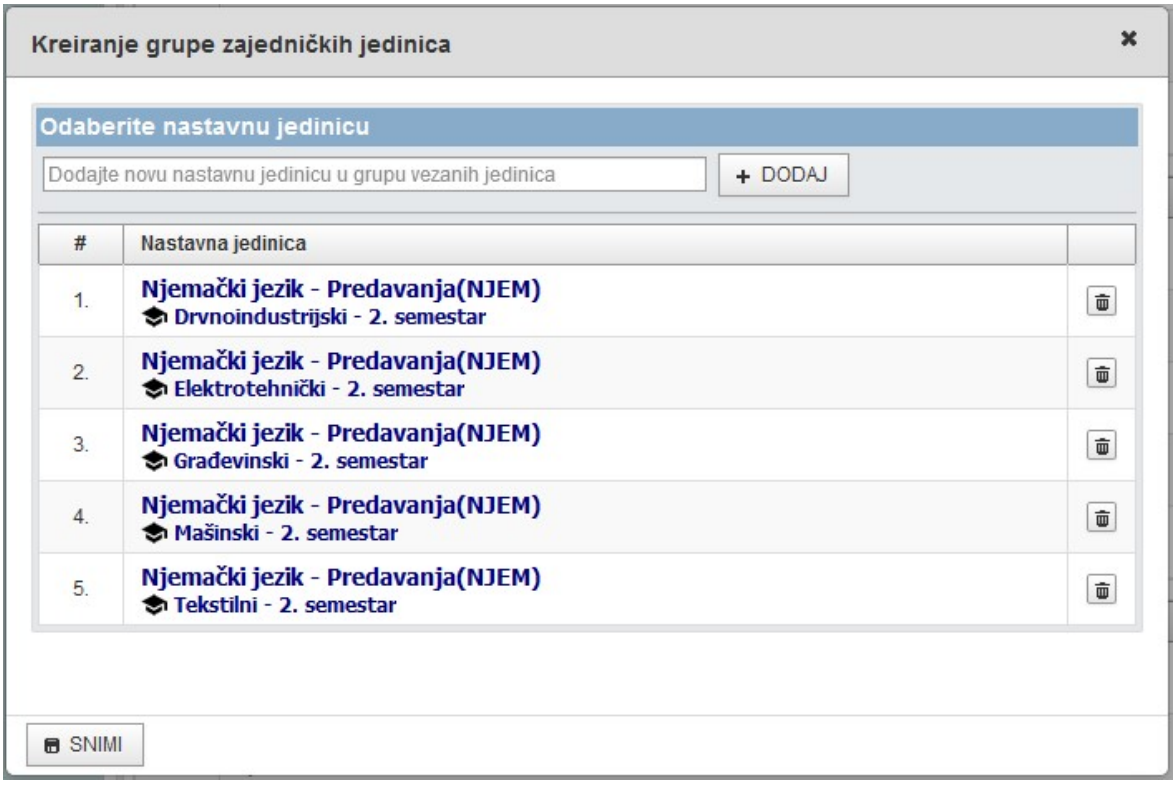
Slika 6.13. Web forma za uređivanje postavki predmeta

U sklopu web forme za uređivanje postavki predmeta sa slike 6.13, moguće je eksplicitno definisati vremenski redoslijed raspoređivanja nastavnih jedinica ovisno o vrsti nastave, npr. ukoliko je potrebno da se prvo održavaju predavanja, a iza njih vježbe iz istog predmeta.

Na slici 6.15 prikazana je web forma za uređivanje grupe nastavnih jedinica za zajedničko izvođenje nastave.



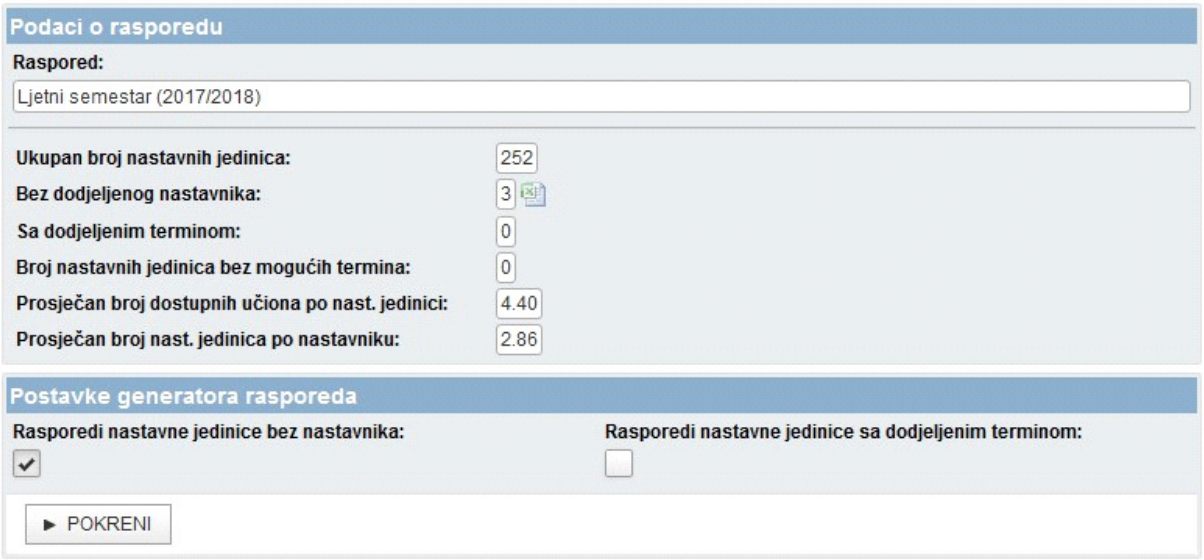
Slika 6.14. Web forma za uređivanje grupe vezanih/kontinuiranih nastavnih jedinica



Slika 6.15. Web forma za uređivanje grupe zajedničkih nastavnih jedinica

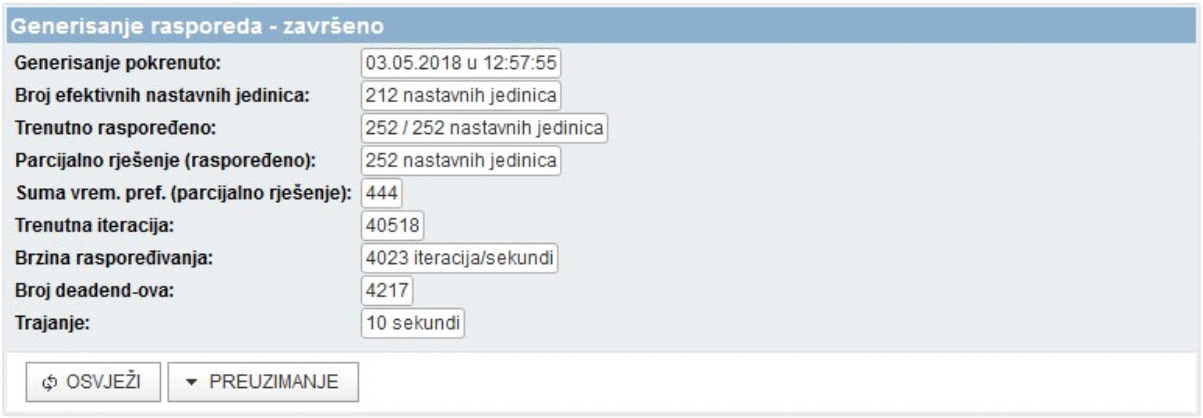
6.2. Generisanje rasporeda nastave

Nakon izvršenog unosa ulaznih parametara, korisnik otvara stranicu (slika 6.16) na kojoj se pokreće generator rasporeda nastave. Prije samog pokretanja generatora, na stranici se prikazuju i statistički podaci o ulaznim parametrima, kao npr. ukupan broj nastavnih jedinica, broj nastavnih jedinica bez dodjeljenog nastavnika, broj nastavnih jedinica koje nije moguće rasporediti zbog definisanih ograničenja, itd. Ovi statistički podaci mogu uveliko pomoći pri uklanjanju eventualnih propusta pri definisanju ulaznih parametara. U postavkama generatora moguće je dodatno definisati stavke kao npr. da li se pri generisanju rasporeda nastave trebaju rasporediti i nastavne jedinice bez dodjeljenog nastavnika, te da li se raporeduju nastavne jedinice kojima je već definisan termin, tj. koje su već unaprijed raspoređene. Ovom drugom stavkom omogućeno je parcijalno raspoređivanje nastavnih jedinica.



Slika 6.16. Web forma za generisanje rasporeda nastave

Pokretanje generatora započinje klikom na dugme „POKRENI“. S obzirom da je algoritam iterativnog karaktera, samo izvršavanje algoritma je moguće pauzirati, te u tom slučaju preuzeti parcijalno rješenje. Na slici 6.17 prikazani su statistički podaci generisanog rasporeda za ovaj primjer ulaznih parametara.



Slika 6.17. Statistički podaci generisanog rasporeda

6.3. Rezultati

Predstavljeni algoritam u ovom radu testiran je sa realnim ulaznim parametrima iz ljetnog semestra 2017/2018. školske godine na Tehničkom fakultet Univerziteta u Bihaću. XML dokument sa ulaznim parametrima se nalazi u prilogu br. 3.

Algoritmu je bilo potrebno u prosjeku 10 sekundi da kreira raspored nastave. U nastavku su date osobine dobijenog rasporeda nastave:

- sve 252 nastavne jedinice su raspoređene (sva fiksna ograničenja su zadovoljena)
- ne postoji slučaj da nastavnik ima više od 8 sati dnevno, izuzev jednog slučaja koji je, prema zahtjevu nastavnika, definisan u ulaznim parametrima
- ne postoji slučaj da nastavnik ima više od 6 sati u bloku nastave bez pauze, izuzev tri slučaja koji su, prema zahtjevu nastavnika, definisana u ulaznim parametrima
- 13% nastavnika ima negativnu sumu vremenskih ograničenja
- ukupna suma vremenskih ograničenja je 439 na nivou svih nastavnih jedinica
- ne postoji slučaj da studijska grupa ima više od 10 sati nastave dnevno
- 5 studijskih grupa ima blok nastave duži od 6 sati bez pauze

Na slikama 6.18, 6.19 i 6.20 prikazani su generisani rasporedi za prostoriju, studijsku grupu i nastavnika respektivno.

	Ponedeljak	Utorak	Srijeda	Četvrtak	Petak	Subota
8:00 - 9:00	Sistemi u... Vježbe LE	Računarske... Vježbe LE	Automatsko... Vježbe LE	Računarsko... Vježbe LE	Osnove... Vježbe LE	Inteligentni... Vježbe LE
9:00 - 10:00	Sistemi u... Vježbe LE	Računarske... Vježbe LE	Automatsko... Vježbe LE	Računarsko... Vježbe LE	Osnove... Vježbe LE	Inteligentni... Vježbe LE
10:00 - 11:00	Sistemi u... Predavanja LE	Elektronika II Vježbe LE	Automatsko... Vježbe LE	Projektovanje... Vježbe LE	Računarske... Predavanja LE	Inteligentni... Vježbe LE
11:00 - 12:00	Sistemi u... Predavanja LE	Elektronika II Vježbe LE	Inteligentni... Predavanja LE	Projektovanje... Vježbe LE	Računarske... Predavanja LE	Računarsko... Predavanja LE
12:00 - 13:00	Arhitektura... Vježbe LE	Elektronika II Vježbe LE	Inteligentni... Predavanja LE	Projektovanje... Vježbe LE	Obrada... Vježbe LE	Računarsko... Predavanja LE
13:00 - 14:00	Arhitektura... Vježbe LE	Uvod u MATLAB Vježbe LE	Uvod u MATLAB Predavanja LE		Obrada... Vježbe LE	
14:00 - 15:00		Uvod u MATLAB Vježbe LE			Obrada... Vježbe LE	
15:00 - 16:00		Uvod u MATLAB Vježbe LE				
16:00 - 17:00						
17:00 - 18:00		ALGORITMI I... Vježbe(A2) LE	ALGORITMI I... Vježbe(A1) LE			
18:00 - 19:00	Sigurnost... Vježbe LE	ALGORITMI I... Vježbe(A2) LE	ALGORITMI I... Vježbe(A1) LE			
19:00 - 20:00	Sigurnost... Vježbe LE	ALGORITMI I... Vježbe(A2) LE	ALGORITMI I... Vježbe(A1) LE			

Slika 6.18. Generisani raspored za prostoriju „Laboratorij elektrotehnike“

	Ponedeljak	Utorak	Srijeda	Četvrtak	Petak	Subota
8:00 - 9:00	Osnove... Predavanja S4	Računarske... Vježbe LE	Automatsko... Vježbe LE	Automatsko... Predavanja RC	Osnove... Vježbe LE	Inteligentni... Vježbe LE
9:00 - 10:00	Osnove... Predavanja S4	Računarske... Vježbe LE	Automatsko... Vježbe LE	Automatsko... Predavanja RC	Osnove... Vježbe LE	Inteligentni... Vježbe LE
10:00 - 11:00	Operaciona... Predavanja S1		Automatsko... Vježbe LE	Operaciona... Vježbe Č	Računarske... Predavanja LE	Inteligentni... Vježbe LE
11:00 - 12:00	Operaciona... Predavanja S1		Inteligentni... Predavanja LE	Operaciona... Vježbe Č	Računarske... Predavanja LE	
12:00 - 13:00			Inteligentni... Predavanja LE			
13:00 - 14:00						
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00		Sigurnost... Predavanja OU				
17:00 - 18:00		Sigurnost... Predavanja OU				
18:00 - 19:00	Sigurnost... Vježbe LE					
19:00 - 20:00	Sigurnost... Vježbe LE					

Slika 6.19. Generisani raspored za studijsku grupu „Elektrotehnički odsjek – 6. semestar“

	Ponedeljak	Utorak	Srijeda	Četvrtak	Petak	Subota
8:00 - 9:00				Automatsko... Predavanja RC		
9:00 - 10:00				Automatsko... Predavanja RC		
10:00 - 11:00	Sistemi u... Predavanja LE			Elektronika II Predavanja RC	Računarske... Predavanja LE	
11:00 - 12:00	Sistemi u... Predavanja LE		Inteligentni... Predavanja LE	Elektronika II Predavanja RC	Računarske... Predavanja LE	
12:00 - 13:00			Inteligentni... Predavanja LE			
13:00 - 14:00			Uvod u MATLAB Predavanja LE			
14:00 - 15:00						
15:00 - 16:00						
16:00 - 17:00						
17:00 - 18:00						
18:00 - 19:00						
19:00 - 20:00						

Slika 6.20. Generisani raspored za nastavnika „dr. sc. Edin Mujčić“

7. ANALIZA MOGUĆNOSTI I PERFORMANSI IMPLEMENTIRANOG RJEŠENJA

7.1. Analiza mogućnosti

Većina postojećih softverskih alata, koji rješavaju problem automatskog kreiranja rasporeda nastave, su ili prejednostavna da bi se mogla definisati sva neophodna ograničenja, ili preopširna i neintuitivna da bi se brzo i efikasno mogla koristiti. Skala ograničenja (slika 4.1), koja je omogućila preciznije definisanje prostornih, vremenskih i distribucijskih ograničenja nad različitim elementima raspoređivanja, je jedan od bitnijih faktora koji povećava mogućnosti implementiranog rješenja. Zbog činjenice da ova skala sadrži i varijabilna ograničenja koja služe pri evaluaciji kvaliteta rješenja, samim tim ista imaju i veliki utjecaj na performanse izvršavanja algoritma. U predstavljenom algoritmu je moguće definisati sljedeća ograničenja:

- vremenska ograničenje – zajednička:
  - ✓ preferiranje definisanih vremenskih perioda (od „Veoma poželjnih“ do „Veoma nepoželjnih“),
  - ✓ pauze (svi nastavnici, sve prostorije i sve studijske grupe nisu dostupne);
- vremenska ograničenja – nastavnici:
  - ✓ nastavnik nije dostupan u definisanom periodu/ima,
  - ✓ grupa nastavnika nije dostupna u definisanom periodu/ima,
  - ✓ dodjeljene nastavne jedinice se moraju rasporediti u definisanom periodu/ima,
  - ✓ nastavnik preferira određeni period/e (od „Veoma poželjnog“ do „Veoma nepoželjnog“),
  - ✓ grupa nastavnika preferira određeni period/e (od „Veoma poželjnog“ do „Veoma nepoželjnog“);
- vremenska ograničenja – prostorije:
  - ✓ prostorija nije dostupna u definisanom periodu/ima,
  - ✓ grupa prostorija nije dostupna u definisanom periodu/ima,
  - ✓ preferiranje korištenja prostorije u određenom periodu/ima (od „Veoma poželjnog“ do „Veoma nepoželjnog“),

- ✓ preferiranje korištenja grupe prostorija u definisanom periodu/ima (od „Veoma poželjnog“ do „Veoma nepoželjnog“),
- ✓ prostorni resurs nije dostupan u definisanom periodu/ima,
- ✓ preferiranje korištenja prostornih resursa u određenom periodu/ima (od „Veoma poželjnog“ do „Veoma nepoželjnog“);

- vremenska ograničenja – studijske grupe / studenti:
  - ✓ studijska grupa nije dostupna u određenom periodu/ima,
  - ✓ studijska podgrupa nije dostupna u određenom periodu/ima,
  - ✓ nastavne jedinice studijske grupe se moraju rasporediti u određenom periodu/ima,
  - ✓ nastavne jedinice studijske podgrupe se moraju rasporediti u određenom periodu/ima;
- vremenska ograničenja – nastavne jedinice:
  - ✓ nastavna jedinica se mora/ne smije rasporediti u definisanom periodu/ima,
  - ✓ grupa nastavnih jedinica se mora/ne smije rasporediti u definisanom periodu/ima,
  - ✓ nastavna jedinica preferira određeni period/e (od „Veoma poželjnog“ do „Veoma nepoželjnog“),
  - ✓ grupa nastavnih jedinica preferira određeni period/e (od „Veoma poželjnog“ do „Veoma nepoželjnog“),
  - ✓ nastavna jedinica se mora rasporediti u istom terminu zajedno sa drugom ili više nastavnih jedinica,
  - ✓ nastavna jedinica se mora rasporediti prije/poslije druge nastavne jedinice,
  - ✓ nastavna jedinica se mora rasporediti u kontinuitetu sa drugom nastavnom jedinicom,
  - ✓ predavanja/vježbe se moraju rasporediti prije/poslije vježbi/predavanja iz istog predmeta;
- prostorna ograničenja – zajednička:
  - ✓ prostorija nije dostupna za korištenje,
  - ✓ preferiranje korištenja određene učione/a, od „Veoma poželjno“ do „Veoma nepoželjno“;
- prostorna ograničenja – nastavnici:

- ✓ nastavniku je neophodna/zabranjena učiona/e za izvođenje nastave,
- ✓ grupi nastavnika je neophodna/zabranjena učiona/e za izvođenje nastave,
- ✓ nastavnik prefereira izvođenje nastave u definisanoj učioni/ama (od „Veoma poželjne“ do „Veoma nepoželjne“),
- ✓ grupa nastavnika preferira izvođenje nastave u definisanoj učioni/ama (od „Veoma poželjne“ do „Veoma nepoželjne“);

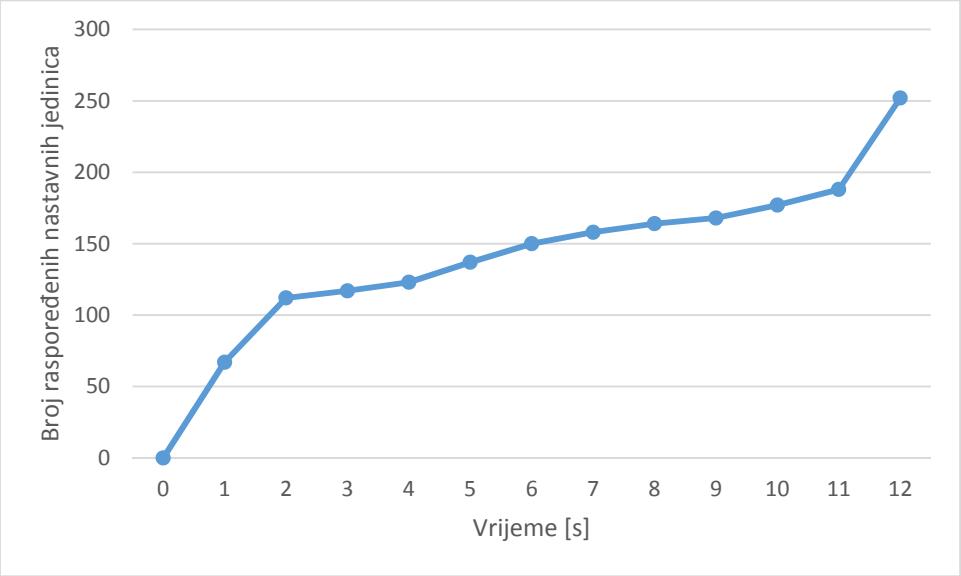
- prostorna ograničenja – studijske grupe / studenti:
  - ✓ studijskoj grupi je neophodna/zabranjena učiona/e za izvođenje nastave,
  - ✓ studijskoj podgrupi je neophodna/zabranjena učiona/e za izvođenje nastave,
  - ✓ studijska grupa preferira korištenje određene učione/a (od „Veoma poželjne“ do „Veoma nepoželjne“),
  - ✓ studijska podgrupa preferira korištenje određene učione/a (od „Veoma poželjne“ do „Veoma nepoželjne“);
- prostorna ograničenja – nastavne jedinice:
  - ✓ nastavna jedinica se mora/ne smije rasporediti u definisanoj učioni/ama,
  - ✓ grupa nastavnih jedinica se mora/ne smije rasporediti u definisanoj učioni/ama,
  - ✓ nastavnoj jedinici su neophodni određeni resursi za njeno izvođenje;

7.2. Analiza performansi

Performanse algoritma direktno ovise od obimnosti ulaznih parametara, odnosno broja nastavnih jedinica koje se raspoređuju i njihove međusobne povezanosti. U najboljem slučaju, kada nema definisanih ovisnosti između nastavnih jedinica (različit nastavnik, prostorija i studijska grupa), kompleksnost algoritma bi bila O(N), gdje je N broj nastavnih jedinica. Međutim, upravo definisana ograničenja i ovisnosti među nastavnim jedinicama povećavaju kompleksnost algoritma na O(N^C), gdje konstanta C predstavlja utjecaj pomenutih ograničenja na ukupnu kompleksnost algoritma [27].

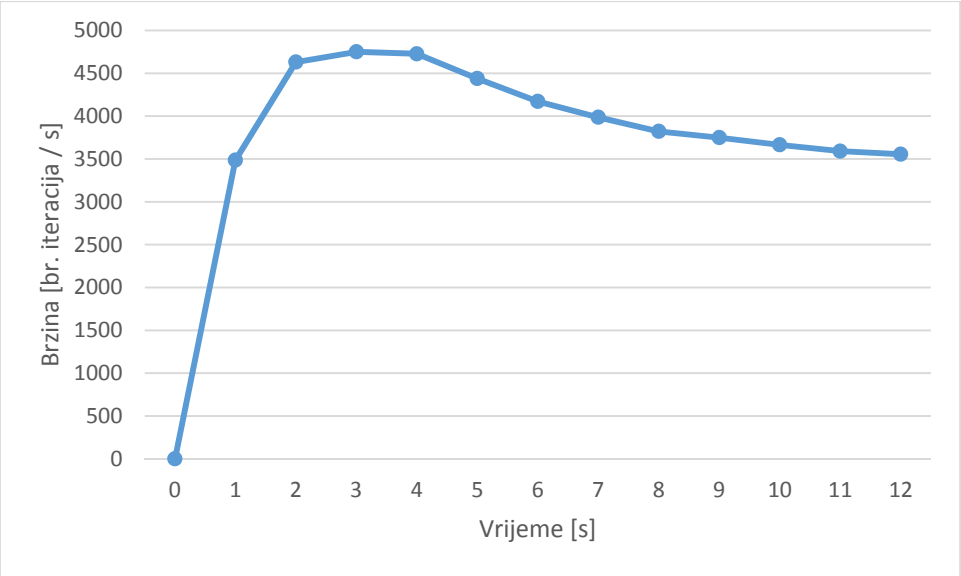
U slučaju predstavljenog algoritma, kada je $C > 1$, smatra se da u toku izvršavanja algoritma javlja *deadend* situacija pri kojoj se mora izvršiti *backtrace* stanja algoritma na jedno od prethodnih stanja i ponoviti postupak sa različitim redoslijedom odabira nastavnih jedinica. Iz navedenog može se zaključiti da *backtrace* postupak predstavlja narušavanje performansi algoritma. Brzina izvršavanja algoritma može se predstaviti sa brojem iteracija po sekundi,

odnosno brojem uspješno raspoređenih nastavnih jedinica u sekundi. U nastavku je dat komparativni prikaz brzine izvršavanja algoritma, broja raspoređenih jedinica i broja *deadend* situacija za primjer izvršavanja algoritma nad ulaznim podacima iz poglavlja 6 ovog rada.



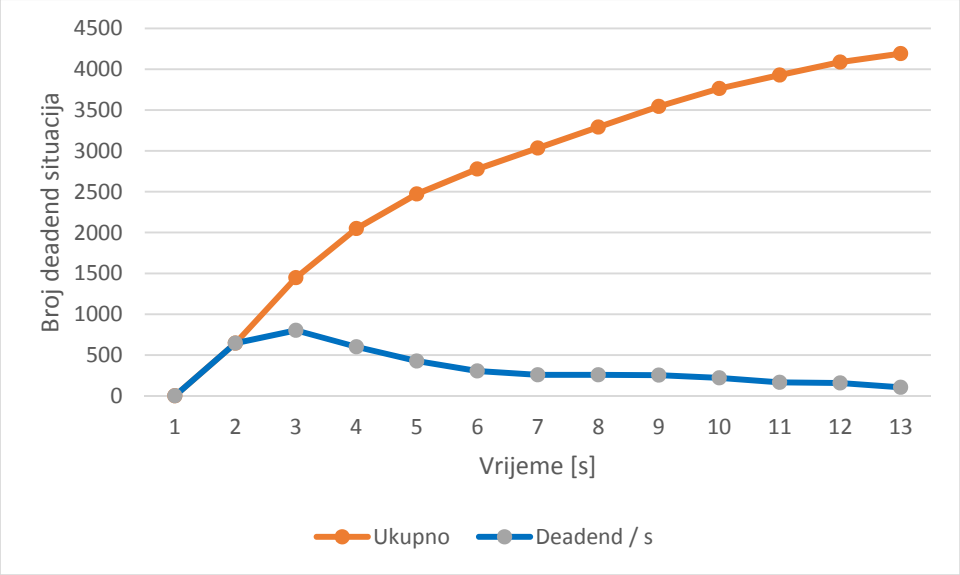
Slika 7.1. Broj raspoređenih nastavnih jedinica tokom izvršavanja algoritma

Kako se povećava broj raspoređenih nastavnih jedinica (slika 7.1), tako i *backtrace* postupak traje duže pri pojavi *deadend* situacija, jer mora da vrati algoritam unazad veći broj stanja, a što u konačnici smanjuje brzinu izvršavanja algoritma (slika 7.2).



Slika 7.2. Brzina izvršavanja algoritma

Iako na prikazanom dijagramu sa slike 7.3 možemo vidjeti blagi pad broja *deadend* situacija po sekundi, ipak duže trajanje *backtrace* postupka nakon svake *deadend* situacije potvrđuje već pomenutu činjenicu da se time i smanjuje brzina izvršavanja algoritma.



Slika 7.3. Deadend situacije tokom izvršavanja algoritma

7.3. Usporedba dobijenih rezultata implementiranog rješenja sa postojećim rješenjima

U ovom poglavlju prikazat ćemo analizu rezultata raspoređivanja dobijenih korištenjem Unitime alata iz sekcije 3.3. i algoritma predstavljenim u ovom radu. Unitime alat je odabran zbog sljedećih karakteristika:

- omogućava definisanje svih vrsta ograničenja kao i u predstavljenom algoritmu u ovom radu,
- open source,
- aktuelan alat razvijan unazad više od 10 godina,
- primjenjuje se na nivou čitavog Purdue Univerziteta (Indiana, USA) za raspoređivanje blizu 2500 nastavnih jedinica po semestru,
- od 2015. godine primjenjuje se na MIT Univerzitetu (Massachusetts, USA)[28].

Da bi rezultati bili mjerodavniji, koristit ćemo parametre za kreiranje rasporeda nastave za dva semestra, i to zimskog i ljetnog semestra 2016/2017. godine sa Tehničkog fakultetu Univerziteta u Bihaću [29]. Detalji parametara su prikazani u tabeli 7.1.

Eksperiment je izvršen na računaru sa 3.6 GHz i7 Intel procesorom, 16GB RAM memorije i Windows 10 operativnim sistemom. Unitime alat koristi stohastički pristup pri generisanju rasporeda nastave, tj. pri svakom pojedinačnom testu dobijaju se različiti rezultati, dok predstavljeni algoritam koristi deterministički pristup, tj. svaki pojedinačni test daje iste rezultate. Zbog navedenog, izvršit će se 10 pojedinačnih testova sa Unitime alatom.

Tabela 7.1. Pregled parametara za kreiranje rasporeda nastave

	Zimski semestar		Ljetni semestar	
Broj nastavnih jedinica	253		256	
Broj zajedničkih jedinica	**Br. grupa**	**Br. jedinica**	**Br. grupa**	**Br. jedinica**
	10	35	18	59
Broj nastavnih jedinica za raspoređivanje[13]	228		215	
Broj kontinuiranih nastavnih jedinica	10		20	
Broj nastavnika	72		72	
Prosječan broj dostupnih prostorija po nastavnoj jedinici	4.41		4.35	
Prosječan broj nastavnih jedinica po nastavniku	3.51		3.56	

Kako bismo mogli izvršiti evaluaciju kreiranih rasporeda nastave, koristit ćemo samo globalna vremenske ograničenja prikazanim u tabeli 7.2.

Prema skali ograničenja (slika 4.1) i definisanim globalnim vremenskim ograničenjima (tabela 7.2) moguće je valorizirati kreirani raspored sumiranjem pripadajućih numeričkih vrijednosti vremenskih slotova za dodjeljene termine nastavnim jedinicama. Npr. ako je nastavna jedinica raspoređena u terminu Ponedeljak od 08h-11h, suma numeričkih vrijednosti prema dodjeljenom terminu iznosi 5 (2 + 2 + 1), ili u terminu Petak od 15h-17h bi iznosila -3 (-1 - 2).

Tabela 7.2. Pregled globalnih vremenskih ograničenja pri kreiranju rasporeda nastave

	Ponedeljak	**Utorak**	**Srijeda**	**Četvrtak**	**Petak**	**Subota**
08:00-09:00						
09:00-10:00						

[13] Broj nastavnih jedinica za raspoređivanje predstavlja stvarni broj nastavnih jedinica koje algoritam raspoređuje, tj. preostali broj nastavnih jednica nakon što se izvrši grupisanje zajedničkih nastavnih jedinica, gdje se dalje jedna grupa zajedničkih jedinica predstavlja kao jedna nastavna jedinica.

10:00-11:00						
11:00-12:00						
12:00-13:00						
13:00-14:00						
14:00-15:00						
15:00-16:00						
16:00-17:00						
17:00-18:00						
18:00-19:00						
19:00-20:00						

U tabelama 7.3 i 7.4 prikazani su rezultati za svaki pojedinačni test. Svaki test sa Unitime alatom trajao je 30 minuta.

Tabela 7.3. Pregled rezultata pri kreiranju rasporeda nastave zimskog semestra

Zimski semestar							
	Vrijeme potrebno za kreiranje inicijalnog rješenja (sec)	Vrijeme potrebno za pronalazak najboljeg rješenja (sec)	Broj nastavnika sa negativnim ogračenjima	Suma negativnih ograničenja nastavnika	Broj studijskih grupa sa negativnim ograničenjima	Suma negativnih ograničenja studijskih grupa	**Suma ograničenja svih nastavnih jedinica**
Predstavljeni algoritam							
Test	**352.0**	**352.0**	**14**	**-31**	**1**	**-3**	**270**
Unitime alat							
1. test	10.8	**1743.0**	13	-61	1	-1	**237**
2. test	11.4	1686.6	**19**	**-66**	1	-9	**227**
3. test	9.6	1242.6	15	-49	**0**	**0**	**234**
4. test	10.8	1383.6	**12**	-51	0	0	**239**
5. test	**7.8**	1363.2	12	**-39**	1	-3	**199**
6. test	12.0	1158.0	18	-44	1	-8	**214**
7. test	9.0	1597.2	14	-52	**2**	**-12**	**213**
8. test	**14.4**	**478.2**	14	-52	1	-3	**245**
9. test	9.0	1050.6	15	-42	0	0	**254**
10. test	11.4	1581.6	14	-44	0	0	**239**
Prosječna vrijednost	**10.6**	**1328.5**	**14.6**	**-50.0**	**0.7**	**-3.6**	**230.1**

Tabela 7.4. Pregled rezultata pri kreiranju rasporeda nastave ljetnog semestra

Ljetni semestar	Vrijeme potrebno za kreiranje inicijalnog rješenja (sec)	Vrijeme potrebno za pronalazak najboljeg rješenja (sec)	Broj nastavnika sa negativnim ogračenjima	Suma negativnih ograničenja nastavnika	Broj studijskih grupa sa negativnim ograničenjima	Suma negativnih ograničenja studijskih grupa	**Suma ograničenja svih nastavnih jedinica**
Predstavljeni algoritam							
Test	**63.8**	**63.8**	**18**	**-36**	**0**	**0**	**334**
Unitime alat							
1. test	10.8	**1509.6**	**17**	-47	1	-1	**209**
2. test	8.4	780.6	16	-38	**0**	**0**	**275**
3. test	6.7	**298.8**	13	-51	1	-2	**224**
4. test	8.4	1333.2	14	-42	**2**	**-8**	**211**
5. test	8.4	483.0	11	**-36**	1	-4	**244**
6. test	8.4	852.0	15	-51	1	-3	**209**
7. test	**6.6**	1404.6	12	**-52**	0	0	**248**
8. test	**15.0**	1110.0	**7**	-41	1	-4	**247**
9. test	10.2	1252.2	14	-46	0	0	**210**
10. test	8.4	1293.6	13	-45	1	-2	**256**
Prosječna vrijednost	**9.1**	**1031.8**	**13.2**	**-44.9**	**0.8**	**-2.4**	**233.3**

U nastavku je obrazloženo značenje vrijednosti pojedinih kolona prikazanih u tabelama 7.3 i 7.4:

- „**Vrijeme potrebno za kreiranje inicijalnog rješenja (sec)**“ predstavlja vrijeme u sekundama potrebno da se kreira kompletan i izvodljiv raspored nastave. Kao što se može uočiti u prikazanim podacima, vrijeme za kreiranje inicijalnog rješenja kod Unitime alata je puno kraće iz razloga što isti pri kreiranju inicijalnog rješenja ne uzima u obzir zadovoljavanje varijabilnih ograničenja
- **„Vrijeme potrebno za pronalazak najboljeg rješenja (sec)“** predstavlja vrijeme u sekundama potrebno da se pronađe rješenje koje najbolje zadovoljava varijabilna ograničenja. Kod predstavljenog algoritma, vrijeme za kreiranje inicijalnog rješenja i vrijeme za pronalazak najboljeg rješenja je isto, jer algoritam pri kreiranju inicijalnog

rješenja pokušava da što bolje zadovolji varijabilna ograničenja, čime inicijalno rješenja predstavlja i najbolje rješenje.

- **„Broj nastavnika sa negativnim ograničenjima“** predstavlja broj nastavnika koji ima negativnu sumu ograničenja u najboljem rješenju. Kako bi se izbjeglo da u najboljem rješenju imamo situaciju favoriziranja nastavnika, neophodno je izvršiti balansiranje zadovoljavanja varijabilnih ograničenja po nastavniku. Ciljna vrijednost treba da bude što manja.
- **„Suma negativnih ograničenja nastavnika“** predstavlja sumu negativnih ograničenja nastavnika iz prethodne kolone.
- **„Broj studijskih grupa sa negativnim ograničenjima“** predstavlja broj studijskih grupa koje imaju negativnu sumu ograničenja u najboljem rješenju. Isti princip balansiranja je izvršen i za studijske grupe kao i za nastavnike.
- **„Suma negativnih ograničenja studijskih grupa“** predstavlja sumu negativnih ograničenja studijskih grupa iz prethodne kolone.
- **„Suma ograničenja svih nastavnih jedinica“** predstavlja ukupnu sumu ograničenja svih nastavnih jedinica sa dodjeljenim terminima. Ciljna vrijednost treba da bude što veća, tj. veća vrijednost predstavlja veći stepen zadovoljavanja varijabilnih ograničenja, a samim tim i kvalitetnije rješenje.

Usporedbom prosječne vrijednosti zadnje kolone između predstavljenog algoritma i Unitime alata može se zaključiti da je moguće dobiti kvalitetnije rješenje korištenjem determinističkih tehnika nasuprot stohastičkim tehnikama korištenim u Unitime alatu.

8. ZAKLJUČAK

8.1. Kritički osvrt na rad

Osnovna težnja ovog rada je da se na osnovu iskustva stečenog kako razvojem softverskih rješenja opšte namjene, tako i informacionog sistema za potrebe visokoobrazovne ustanove, implementira rješenje za automatizaciju procesa kreiranja i distribucije rasporeda nastave. Razvoj predloženog algoritma je vođen činjenicom da kvalitet i vrijeme potrebno za ručnu izradu rasporeda nastave direktno zavise od organizacionih sposobnosti njegovog autora. Pored toga, automatizacijom ovog procesa povećava se ne samo kvalitet, već i transparentnost kreiranog rasporeda. Iz rezultata testiranja predloženog rješenja može se zaključiti da isto zadovoljava realne potrebe Univerziteta u Bihaću. Međutim, s obzirom na razlike u obrazovnim sistemima u svakoj državi, pa čak između pojedinih ustanova u istoj državi, može se zaključiti da se i formulacije problema automatskog raspoređivanja nastave i modeli rješavanja istih razlikuju, čime se ograničava primjena predloženog rješenja na određenu ustanovu ili sistem.

Jedna od prednosti implementiranog rješenja opisanog u ovom radu, jeste mogućnost njegove integracije sa postojećim informacionim sistemom, što dodatno olakšava i ubrzava proces definisanja ulaznih parametara za kreiranje rasporeda nastave, te samim tim i njegovu svakodnevnu upotrebu. Dalje, iterativni karakter predloženog algoritma daje mogućnost da se proces njegovog izvršavanja može zaustaviti/pauzirati u bilo kojem trenutku i preuzeti parcijalno rješenje. Isto tako, algoritam je moguće pokrenuti nad parcijalnim rješenjem, što ostavlja mogućnost za eventualne dopune i izmjene već postojećeg rasporeda nastave.

Iako kreiranje korisničkog interfejsa za definisanje ulaznih parametara nije bio cilj istraživanja u ovom radu, predloženi algoritam je dovoljno otvoren i dokumentovan, da bilo koji softverski inženjer može razviti sopstveni interfejs te primijeniti implementirani algoritam u svom sistemu.

8.2. Teorijski i praktični doprinosi rada

Sa teorijskog aspekta, u radu je predstavljen novi algoritam za automatsko raspoređivanje nastave, sa detaljnom analizom njegovih mogućnosti i performansi. Na osnovu rezultata dobijenih analizom i testiranjem implementiranog algoritma, baziranog na determinističkim tehnikama, može se zaključiti da je moguće u kraćem vremenskom periodu

dobiti kvalitetnije rješenje u odnosu na alate koji koriste stohastičke tehnike. Takođe, data je specifikacija ulaznih parametara i pregled dodatnih ograničenja koja se mogu definisati u ulaznim parametrima, što bi trebalo pomoći prilikom daljnjeg razvoja i nadogradnje predstavljenog algoritma.

Praktični doprinosi ovog rada su već jednim djelom predstavljeni u poglavlju 6, u kojem je opisan primjer uspješne primjene implementiranog rješenja. Naime, svaki vid automatizacije procesa keiranja rasporeda nastave uveliko olakšava i ubrzava pomenuti proces, koji predstavlja težak i vremenski zahtjevan posao. Uzimajući u obzir činjenicu da je za ručno kreiranje rasporeda nastave iz poglavlja 6 bilo potrebno u prosjeku dva dana, a da se isti može kroz novoimplementirani modul informacionog sistema odraditi u nekoliko sati i pri tom dobiti kvalitetniji raspored nastave, predstavlja značajan pomak u efikasnosti obavljanja ovog posla.

8.3. Mogućnosti nadogradnje

Uzimajući u obzir činjenicu da implementirano rješenje predstavlja Java biblioteku, odnosno *open-source* rješenje, tada se može reći da su mogućnosti nadogradnje neograničene. Pošto predstavljeno rješenje sadrži dosta zajedničkih osobina sa drugim sličnim alatima, jedan od pravaca daljnje nadogradnje može ići u smjeru dodavanje novih vrsta ograničenja koja nisu trenutno podržana, a da se pri tom ne mijenja formulacija problema, odnosno algoritma. Pošto predloženi algoritam je prilagođen za kreiranje sedmičnog rasporeda, koji bi se u suštini ponavljao svake sedmice tokom semestra, drugi od pravaca nadogradnje bi išao u smjeru prilagođavanja algoritma, odnosno izmjena formulacije problema, a koja bi omogućila kreiranje semestralnog rasporeda nastave. Suštinska izmjena u formulaciji problema bi se odnosila na mogućnost definisanje vremenskih i distribucijskih ograničenja za čitav semestar.

Implementacijom dodatnog web servisa, kao međusloja između drugih web aplikacija/servisa i implementiranog rješenja, svakako bi se povećao stepen dostupnosti istog.

8.4. Pravci daljeg istraživanja

Dalja istraživanja u oblasti tehnika raspoređivanja svakako mogu imati više pravaca. Jedan od pravaca bi svakako bio primjena i adaptacija predstavljenog algoritma u drugim sferama kao što su raspoređivanja ispitnih termina, dežurstava u raznim ustanovama, alokacija raznih vrsta resursa, itd.

Iz rezultata primjene implementiranog rješenja, može se također zaključiti da ono više predstavlja rješenje problema zadovoljavanja postavljenih ograničenja, nego problema optimizacije. Ovim se ostavlja prostor za dalja istraživanja u smislu ukrštanja primjenjenih determinističkih tehnika sa drugim tehnikama optimizacije, odnosno stohastičkim tehnikama.

LITERATURA

[1] S. N. Jat, S. Yang, *A Guided Search Genetic Algorithm for the University Course Timetabling Problem*, Multidisciplinary International Conference on Scheduling : Theory and Applications, 2009.

[2] S. Abdullah, E. K. Burke, B. McCollum, Using a randomised iterative improvement algorithm with composite neighbourhood structures, International Conference on Metaheuristic, 2007.

[3] S. Abdullah, H. Turabieh, *Generating university course timetable using genetic algorithm and local search*, Internation Conference on Hybrind Information Technology, 2008.

[4] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic, R. Qu, *A graph-based hyperheuristic for timetabling problems*, European Journal of Operation Research, 2006.

[5] S. Abdullah, E. K. Burke, B. McCollum, *A investigation of variable neighbourhood search for university course timetabling,* Multidisciplinary Conference on Scheduling: Theory and Applications, 2005.

[6] E. K. Burke, G. Kendall, E. Soubeiga, *A tabu-search hyper-heuristic for timetabling and rostering,* Journal of Heuristic, 2003.

[7] K. Socha, J. Knowles, M. Sampels, A max-min ant system for the university course timetabling problem, Workshop on Ant Algorithms, 2002.

[8] O. Rossi-Doria, M. Sampels, M. Birattari, M. Chiarandini, M. Dorigo, L. M. Gambardella, J. Knowles, M. Manfrin, M. Mastrolilli, B. Paechter, L. Paquete, T. Stützle, *A Comparison of the Performance of Different Metaheuristics on the Timetabling Problem*, International Conference on Practice and Theory of Automated Timetabling, 2002.

[9] H. Asmuni, E. K. Burke, J. M. Garibaldi, *Fuzzy multiple heuristic ordering for course timetabling*, Workshop on Computer Intelligence, 2005.

[10] M. Nandhini, S. Kanmani, *A Survey of Simulated Annealing Methodology for University Course Timetabling*, International Journal of Recent Trends in Engineering, 2009.

[11] Y. Bykov, S. Petrovic, A Step Counting Hill Climbing Algorithm applied to University Examiniation Timetabling, Journal of Scheduling, 2016.

[12] J. Han, C. Morag, The influence of the sigmoid function parameters on the speed of backpropagation learning, From Natural to Artificial Neural Computation, 1995.

[13] Z. Lü, J. Hao, *Adaptive Tabu Search for Course Timetabling*, European Journal of Operational Research, 2010.

[14] International Timetabling Competition, www.cs.qub.ac.uk/itc2007/index.htm, datum posljednje posjete 14.06.2018. godine

[15] L. Han, G. Kendall, *An investigation of a tabu assisted hyper-heuristic genetic algorithm*, The Congress on Evolutionary Computation, 2003.

[16] J. S. Soria-Alcaraz, G. Ochoa, J. Swan, M. Caprio, H. Puga, E. K. Burke, *Effective learning hyper-heuristics for the course timetabling problem*, European Journal of Operational Research, 2014.

[17] A. Abbas, E. P. K. Tsang, *Constraint-Based Timetabling – a Case Study*, ACS/IEEE International Conference on Computer Systems and Applications, 2001.

[18] *FET – Free Timetabling Software*, https://lalescu.ro/liviu/fet/, datum posljednje posjete 09.05.2018. godine

[19] *Open Course Timetabler, http://opensourcejavaphp.net/csharp/openctt/,* datum posljednje posjete 02.06.2015. godine

[20] M. Dorigo, T. Stützle, *Ant Colony Optimization,* A Bradford book, 2004.

[21] V. D. Matijaš, G. Molnar, M. Čupić, D. Jakobović, B. Dalbelo Bašić, *University Course Timetabling Using ACO: A Case Study on Laboratory Exercises*, International Conference on Knowledge-based and intelligent information and engineering systems, 2010.

[22] *Unitime | University timetabling*, http://www.unitime.org/, datum posljednje posjete 09.05.2018. godine

[23] T. Müller, *University Course Timetabling: Solver Evolution*, International Conference on Practice and Theory of Automated Timetabling, 2016.

[24] D. de Werra, *An Introduction to Timetabling,* European Journal of Operational Research, 1985.

[25] S. Even, A. Itai, A. Shamir, On the complexity of timetabling and multi-commodity flow problems, SIAM Journal of Computation, 1976.

[26] C. Lecoutre, Constraint Networks: Targeting Simplicity for Techniques and Algorithms, John Wiley & Sons, 2013.

[27] C. Gutiérrez-Soto, *Time Complexity - P and NP Classes*, https://users.dcc.uchile.cl/~clgutier/Time_Complexity_P_vs_NP.pdf , datum posljednje posjete 09.05.2018. godine

[28] *Improving MIT's scheduling system,* http://news.mit.edu/2015/unitime-improving-mit-scheduling-system-0122, datum posljednje posjete 09.05.2018. godine

[29] E. Okanović, Z. Đurić, *Primjena deterministčkog pristupa u rješavanju problema automatskog kreiranja rasporeda nastave*, Međunarodni skup za informacijsku i komunikacijsku tehnologiju, elektroniku i mikroelektroniku (MIPRO), maj 2018.

PRILOZI

Ovaj master rad uključuje DVD-ROM na kojem su sadržani elektronska verzija rada, izvorni kod implementiranog algoritma, XML šema i dijagram klasa ulaznih parametara algoritma kao i XML dokument sa ulaznim parametrima testiranog primjera iz 6. poglavlja. U sljedećoj tabeli prikazan je sadržaj DVD-ROM-a.

Tabela 10.1. Pregled sadržaja DVD-ROM-a

Lokacija	Prilog
\zavrsniRad.docx	Elektronska verzija rada
\prilog1\RasporedGenerator\	Maven projekt sa izvornim kôdom implementiranog algoritma
\prilog2\rasporedModelSchema.xsd	XML šema i klasni dijagram ulaznih parametara
\prilog2\rasporedModelBindings.xjb	JAXB Binding datoteka – dodatak XML šemi
\prilog3\ulazniParametri.xml	XML dokument sa ulaznim parametrima testnog primjera upotrebe implementiranog algoritma

Izjava o identičnosti štampane i elektronske verzije master rada

Ime i prezime autora:	Eldin Okanović
Naslov rada:	Primjena algoritama i tehnika optimizacije pri automatskom kreiranju rasporeda nastave
Mentor:	prof. dr Zoran Đurić

Izjavljujem da je štampana verzija mog master rada identična elektronskoj verziji koju sam predao za digitalni repozitorijum Univerziteta u Banjoj Luci.

U Banjoj Luci, 28.11.2018.

Potpis kandidata,

Okanović Eldin

IZJAVA O AUTORSTVU

Izjavljujem da je
master rad

Naslov rada: **„Primjena algoritama i tehnika optimizacije pri automatskom kreiranju rasporeda nastave“**

Naslov rada na engleskom jeziku: **„Application of optimization algorithms and techniques for automated timetabling“**

- rezultat sopstvenog istraživačkog rada,
- da master rad, u cjelini ili u dijelovima, nije bio predložen za dobijanje bilo koje diplome prema studijskim programima drugih visokoškolskih ustanova,
- da su rezultati korektno navedeni i
- da nisam kršio autorska prava i koristio intelektualnu svojinu drugih lica.

U Banjoj Luci, 28.11.2018.

Potpis kandidata

Okanović Eldin

Izjava kojom se ovlašćuje Elektrotehnički fakultet

Univerziteta u Banjoj Luci da master rad učini javno dostupnim

Ovlašćujem Elektrotehnički fakultet Univerziteta u Banjoj Luci da moj master rad pod naslovom:

„Primjena algoritama i tehnika optimizacije pri automatskom kreiranju rasporeda nastave“

koji je moje autorsko djelo, učini javno dostupnim.

Master rad sa svojim prilozima predao sam u elektronskom formatu, pogodnom za trajno arhiviranje.

Moj master rad, pohranjen u digitalni repozitorijum Univerziteta u Banjoj Luci, mogu da koriste svi koji poštuju odredbe sadržane u odabranom tipu licence Kreativne zajednice *(Creative Commons),* za koju sam se odlučio.

1. Autorstvo
2. Autorstvo – nekomercijalno
3. Autorstvo - nekomercijalno - bez pregrade
4. Autorstvo - nekomercijalno - dijeliti pod istim uslovima
5. Autorstvo - bez prerade
6. Autorstvo - dijeliti pod istim uslovima

U Banjoj Luci, 28.11.2018.

Potpis kandidata

Okanović Eldin

UNIVERZITET U BANJOJ LUCI

PODACI O AUTORU ODBRANJENOG MASTER RADA

Ime i prezime autora master rada:

Eldin Okanović

Datum, mjesto i država rođenja autora:

31.03.1985, Polje, Bosna i Hercegovina

Naziv završenog fakulteta autora i godina diplomiranja:

Tehnički fakultet Univerziteta u Bihaću, 2008

Datum odbrane završnog rada autora:

08.02.2008.

Naslov završnog rada autora:

Izgradnja informacionog sistema fakulteta

Akademsko zvanje koje je autor stekao odbranom završnog rada:

Diplomirani inženjer elektrotehnike, smjer informatika

Naziv fakulteta na kojem je master rad odbranjen:

Elektrotehnički fakultet Univerziteta u Banjoj Luci

Naslov master rada i datum odbrane:

Primjena algoritama i tehnika optimizacije pri automatskom kreiranju rasporeda nastave, 18.12.2018

Naučna oblast master rada prema CERIF šifrarniku:

T 120

Imena mentora i članova komisije za odbranu master rada:

- **prof. dr Branko Blanuša, predsjednik**
- **prof. dr Zoran Đurić, mentor**
- **doc. dr Dražen Brđanin, član**

U Banjoj Luci, 28.11.2018. Dekan:

_____

prof. dr Branko Blanuša
Elektrotehnički fakultet Banja Luka

prof. dr Zoran Đurić
Elektrotehnički fakultet Banja Luka

doc. dr Dražen Brđanin
Elektrotehnički fakultet Banja Luka

Banja Luka, 28.09.2018. godine

Nastavno-naučnom vijeću
Elektrotehničkog fakulteta Univerziteta u Banjoj Luci

Odlukom Nastavno-naučnog vijeća Elektrotehničkog fakulteta Univerziteta u Banjoj Luci, broj 20/3.298 - 311/15 od 27.04.2015. godine, imenovani smo u Komisiju za pisanje izvještaja o urađenom završnom radu II ciklusa studija, kandidata Eldina Okanovića, pod nazivom „Primjena algoritama i tehnika optimizacije pri automatskom kreiranju rasporeda nastave". Nakon pregleda priloženog rada podnosimo sljedeći

IZVJEŠTAJ

1. Biografski podaci kandidata

Eldin Okanović rođen je 31.03.1985. godine u Cazinu. Upisao je Tehnički fakultet u Bihaću akademske 2003/2004. godine, odsjek Elektrotehnika, smjer Informatika. Diplomirao je sa prosjekom ocjena 8,50. Diplomski rad na temu „Izgradnja informacionog sistema fakulteta" uspješno je odbranio 08.02.2008. godine.

Od marta 2008. godine do septembra 2018. godine radio je na Tehničkom fakultetu Bihać, kao asistent na predmetima Programski jezici i Informacione tehnologije. Od 01.09.2018. godine zaposlen je u OSB Engineering & IT, Minhen, Njemačka.

Najznačajniji projekti na kojima je učestvovao:

- Razvoj i administracija kompletnog web baziranog informacionog sistema Tehničkog fakulteta Bihać, koji je u funkciji od 2008. godine – danas. Link: http://info.tfb.ba
- Razvoj CMS sistema za potrebe upravljanje sadržajem na zvaničnoj stranici Tehničkog fakulteta. Link: http://www.tfb.ba

- Implementacija informacionog sistema EDUFIS na fakultete i visoke škole Univerziteta u Bihaću, Link: http://info.vzs.unbi.ba

U svom dosadašnjem radu, kandidat je publikovao rezultate svojih istraživanja, uključujući tu i rad iz oblasti teme ovog završnog rada II ciklusa:

- E. Okanović, Z. Đurić, " Primjena determinističkog pristupa u rješavanju problema automatskog kreiranja rasporeda nastave", MIPRO 2018, 2018;
- E. Okanović, A. Džanić, „Model PKI infrastrukture za visokoobrazovnu ustanovu", RIM 2013, Budva, Crna Gora;
- A. Džanić, E. Okanović, „Model održavanja ERP sistema", RIM 2013, Budva, Crna Gora;
- E. Okanović, A. Džanić, A. Hodžić, „Primjena informacionog sistema za kontinuirano praćenje rada studenata", Kvalitet 2009, Neum, BiH, 2009.

2. Osnovni podaci o radu

Eldin Okanović, dipl. inženjer elektrotehnike, podnio je na pregled i ocjenu završni rad II ciklusa studija pod nazivom „Primjena algoritama i tehnika optimizacije pri automatskom kreiranju rasporeda nastave". Rad sadrži osam poglavlja: **Uvod, Algoritmi i tehnike optimizacije u kreiranju rasporeda – pregled i analiza, Postojeći softverski alati za automatsko kreiranje rasporeda – pregled i analiza, Analiza specifičnih zahtjeva i ograničenja za automatsko kreiranje rasporeda, Implementacija algoritma, Primjer upotrebe implementiranog rješenja, Analiza mogućnosti i performansi implementiranog rješenja** i **Zaključak**. Na kraju rada dat je pregled korištene literature (29 referenci) i Prilog. Rad ima 67 stranica, sadrži 51 sliku i 12 tabela.

U uvodnom dijelu istaknuta je kompleksnost problema automatskog kreiranja rasporeda. Proces kreiranja rasporeda se sastoji od pridruživanja nastavnih jedinica nastavnicima (predavači), studentima (učenici ili polaznici) i prostorijama (učionice, kabineti i laboratorije) u ograničenom broju vremenskih slotova. Vremenski slot je najmanji vremenski period u danu kojeg je moguće alocirati. Pored rješavanja konflikata između nastavnih jedinica u procesu kreiranja rasporeda, u obzir se mora uzeti i kvalitet dobijenog rasporeda, što značajno komplikuje sam proces kreiranja rasporeda. Problem automatskog kreiranja rasporeda, između ostalih, pripada i klasi kombinatornih problema, tj. pri kreiranju rasporeda moguće je dobiti veliki broj izvodljivih (eng. *feasable*) rješenja, što predstavlja prostor rješenja. U praksi je prostor izvodljivih rješenja veliki, te je neophodno iskoristiti algoritme sa efikasnom heuristikom kako bi se u što kraćem vremenu pronašlo što bolje rješenje u tom prostoru.

U drugom poglavlju (Algoritmi i tehnike optimizacije u kreiranju rasporeda – pregled i analiza) dati su pregled i analiza postojećih algoritama i tehnika optimizacije za automatsko kreiranje rasporeda. U netrivijalnim slučajevima, problem automatskog kreiranja rasporeda se smatra NP-problemom (eng. *Non-deterministic Polynomial-time*), što znači da je malo vjerovatno da će se pronaći efikasna metoda za njegovo rješavanje. Iz tog razloga niti jedna postojeća tehnika ne garantuje pronalazak optimalnog rješenja. U ovom poglavlju napravljena je i razlika između tehnika i pristupa rješavanju problema optimizacije, gdje se tehnikom optimizacije smatra algoritam ili grupa algoritama za rješavanje problema optimizacije (npr. genetski algoritmi), a pristup rješavanju problema optimizacije se smatra opštim okvirom za razvoj algoritma za optimizaciju (npr. programiranje bazirano na logici ograničenja).

U trećem poglavlju (Postojeći softverski alati za automatsko kreiranje rasporeda – pregled i analiza) prikazani su detalji najčešće korištenih softverskih alata za automatsko kreiranje rasporeda nastave, te je izvršena analiza njihovih mogućnosti. Detaljno su analizirani FET (eng. *Free Timetabling Software)*, OCTT (eng. *Open Course Timetabler*) i UniTime.

U četvrtom poglavlju (Analiza specifičnih zahtjeva i ograničenja za automatsko kreiranje rasporeda), pored osnovnog skupa ograničenja za automatsko kreiranje rasporeda, na kojem je bazirana većina postojećih alata za automatsko kreiranje rasporeda nastave, dat je pregled određenih specifičnih zahtjeva i ograničenja za automatsko kreiranje rasporeda. Detaljnije su opisana vremenska ograničenja, prostorna ograničenja i ekskluzivitet prostorije, zajednička i dijeljena nastava, kontinuirano izvođenje nastave, kao i virtuelna prostorija, tj. izvođenje nastave putem *online* platforme.

U petom poglavlju (Implementacija algoritma) detaljno je opisana implementacija predloženog algoritma za automatsko kreiranje rasporeda nastave. Predloženi algoritam baziran je na kombinaciji dvije tehnike, i to na tehnici programiranja baziranog na ograničenjima i redukciji domena pretrage (eng. *domain reduction*). Implementacija predloženog algoritma je izvršena korišćenjem Java programskog jezika.

U šestom poglavlju (Primjer upotrebe implementiranog rješenja) dat je primjer upotrebe implementiranog algoritma pri automatskom kreiranju rasporeda nastave nad realnim ulaznim parametrima. Kao primjer generisanja rasporeda, opisan je kompletan proces kreiranja rasporeda za ljetni semestar 2017/2018. godine za I ciklus studija na Tehničkom fakultetu Univerziteta u Bihaću. Algoritam predložen u ovom radu dio je modula informacionog sistema Univerziteta u Bihaću koji je zadužen za kreiranje rasporeda nastave. Pored procesa kreiranja ulaznih parametara, detaljno je opisan i proces generisanja rasporeda.

U sedmom poglavlju (Analiza mogućnosti i performansi implementiranog rješenja) napravljena je komparativna analiza implementiranog rješenja sa postojećim rješenjima prema različitim skupovima definisanih ograničenja odnosno ulaznih parametara. Detaljnim poređenjem predstavljenog algoritma i Unitime alata (jednog od najpopularnijih alata za automatsko kreiranje rasporeda danas) može se zaključiti da je moguće dobiti kvalitetnije rješenje korištenjem determinističkih tehnika nasuprot stohastičkim tehnikama korištenim u Unitime alatu.

U posljednjem poglavlju (Zaključak) su sumirani rezultati, pri čemu je kandidat dao kritički osvrt na rad, te ukazao na mogućnosti nadogradnje i moguće pravce daljeg istraživanja.

3. Analiza rada i najvažniji doprinosi

Ovaj rad sadrži dobar pregled oblasti čijim se problemima bavi, što pokazuje zrelost kandidata i njegovu sposobnost da savlada i sistematizuje znanje iz jedne istraživačke oblasti.

Kandidat je u radu dao originalan prijedlog algoritma za automatsko kreiranje rasporeda nastave, a koji je baziran na kombinaciji dvije tehnike: tehnici programiranja baziranog na ograničenjima i redukciji domena pretrage. Implementacija predloženog algoritma za automatsko kreiranja rasporeda nastave izvršena je korišćenjem Java programskog jezika, nakon čega je izvršena i eksperimentalna verifikacija implementiranog rješenja.

Teorijski doprinosi ovog rada ogledaju se u pregledu i analizi postojećih algoritama i tehnika optimizacije u kreiranju rasporeda nastave, pregledu i analizi postojećih softverskih alata za automatsko kreiranje rasporeda, te analizi određenih specifičnih zahtjeva i ograničenja za automatsko kreiranje rasporeda.

4. Zaključak i prijedlog

Na osnovu pregleda i analize završnog rada II ciklusa studija kandidata Eldina Okanovića, cijeneći postignute doprinose koji su verifikovani i publikovanjem četiri naučna rada, od kojih su tri publikovana na regionalnim naučno-stručnim konferencijama i simpozijumima, Komisija konstatuje da su u potpunosti ostvareni postavljeni ciljevi istraživanja, pa sa zadovoljstvom predlaže Nastavno-naučnom vijeću Elektrotehničkog fakulteta Univerziteta u Banjoj Luci da se rad "Primjena algoritama i tehnika optimizacije pri automatskom kreiranju rasporeda nastave", autora Eldina Okanovića, diplomiranog inženjera elektrotehnike, prihvati kao završni rad II ciklusa studija i da se zakaže usmena odbrana.

Banja Luka, 28.09. 2018. godine

Komisija:

1. _____

prof. dr Branko Blanuša, predsjednik

2. _____

prof. dr. Zoran Đurić, mentor

3. _____

doc. dr. Dražen Brđanin, član