

UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET



Dušan Trtica

PRIMENA TEHNIKA ZADOVOLJENJA OGRANIČENJA I CELOBROJNOG PROGRAMIRANJA NA PROBLEM GENERISANJA RASPOREDA ČASOVA

master rad

Beograd, 2026.

Mentor:

dr Filip MARIĆ, redovni profesor
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

dr Aleksandar KARTELJ, vanredni profesor
Univerzitet u Beogradu, Matematički fakultet

dr Miljan KNEŽEVIĆ, docent
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: _____

Mojoj porodici

Naslov master rada: Primena tehnika zadovoljenja ograničenja i celobrojnog programiranja na problem generisanja rasporeda časova

Rezime: U ovom radu razmatramo problem automatskog generisanja rasporeda časova na univerzitetskom nivou, na podacima Matematičkog fakulteta Univerziteta u Beogradu. Rad je sproveden u tri koraka. Najpre su razvijene dve formulacije osnovnog problema, u kojima se traži raspored koji zadovoljava tvrda ograničenja: jedna u paradigmi programiranja ograničenja (CP-SAT) i jedna kao binarni celobrojni linearni model (0-1 ILP), poseban slučaj mešovitog celobrojnog programiranja, rešavan SCIP-om (oznaka MIP/SCIP); obe su implementirane pomoću biblioteke Google OR-Tools. Zatim je sprovedeno njihovo eksperimentalno poređenje na tri skale realnih podataka, po vremenu konstrukcije modela i rešavanja, veličini modela i memorijskom otisku. Obe formulacije daju validna rešenja na svim skalama, pri čemu CP-SAT do dopustivog rasporeda dolazi brže, te je iz praktičnih razloga izabran kao osnova za nastavak rada. U trećem koraku CP model je proširen: uvedena je arhitektura pravila sa tvrdim i mekim ograničenjima i funkcijom cilja, dodata su ograničenja kvaliteta rasporeda – spajanje časova u blokove, jedna lokacija po danu, izbegavanje procepa i kapacitet učionica – i raspoređivanje nastavnog osoblja. Prošireni model je ponovo evaluiran, uz praćenje vrednosti funkcije cilja kroz vreme i dodatno merenje u okruženju sa manjim brojem termina.

Ključne reči: raspored časova, programiranje ograničenja, celobrojno programiranje, CP-SAT, MIP, OR-Tools, optimizacija, Matematički fakultet

Sadržaj

1	Uvod	1
1.1	Cilj rada	2
1.2	Struktura rada	2
2	Teorijske osnove i formulacija problema	4
2.1	Problem raspoređivanja časova	4
2.1.1	Vrste ograničenja	5
2.1.2	NP-težina problema	5
2.2	Programiranje ograničenja	6
2.2.1	Problem zadovoljenja ograničenja (CSP)	6
2.2.2	Tehnike rešavanja	6
2.2.3	Globalna ograničenja	7
2.2.4	CP-SAT pristup	7
2.3	Celobrojno linearno programiranje	7
2.3.1	Definicija MIP problema	8
2.3.2	SCIP rešavač	8
2.4	Formalna specifikacija	9
2.4.1	Zajednička notacija	9
2.4.2	CP model: promenljive, domeni i ograničenja	10
2.4.3	MIP model: promenljive, domeni i ograničenja	12
2.4.4	Ekvivalentnost dve formulacije	13
3	Implementacija i korišćene tehnologije	14
3.1	Alati i biblioteke	14
3.1.1	Google OR-Tools	14
3.1.2	Bazel	15
3.1.3	Python biblioteke	15

3.2	Ulazni podaci i sesije	15
3.2.1	Model ulaznih podataka	15
3.2.2	Realni podaci i poreklo ulaza	16
3.2.3	Formiranje sesija	17
3.2.4	Obim modela i pretpostavke	18
3.3	Implementacija	19
3.3.1	Ulazni model u kodu	19
3.3.2	Klasa sesije	20
3.3.3	CP implementacija	22
3.3.4	MIP implementacija	23
3.4	Strukturno poređenje formulacija i rešavača	24
4	Nadogradnja CP rešavača dodatnim ograničenjima	26
4.1	Arhitektura pravila	27
4.1.1	Konfiguracija pravila	27
4.1.2	Primena pravila i funkcija cilja	28
4.2	CP-SAT tehnike modeliranja	29
4.2.1	Reifikacija i <code>OnlyEnforceIf</code>	29
4.2.2	Logičko ILI preko <code>AddMaxEquality</code>	30
4.2.3	<code>AddBoolAnd</code> i <code>AddBoolOr</code>	30
4.2.4	Indeksiranje niza preko <code>AddElement</code>	30
4.3	Pravilo: spajanje časova u blokove	31
4.3.1	Formiranje blokova	31
4.3.2	Tvrda varijanta	31
4.3.3	Meka varijanta	32
4.4	Pravilo: jedna lokacija po danu za grupu	32
4.4.1	Mehanizam	33
4.5	Pravilo: bez procepa u rasporedu	34
4.5.1	Matrica zauzetosti	34
4.5.2	Detekcija procepa	34
4.5.3	Tvrda i meka varijanta	35
4.6	Pravilo: kapacitet učionice kao meko ograničenje	35
5	Raspoređivanje nastavnog osoblja	38
5.1	Model osoblja i dodela nastave	38
5.2	Grupe studenata izvedene iz dodela nastave	39

5.3	Kohorte i zajedničke sesije	40
5.4	Pravila nastavnog osoblja	42
5.4.1	Nastavnik ne može držati dva časa istovremeno	42
5.4.2	Jedna lokacija po danu za nastavnika	43
5.4.3	Najveći broj radnih dana nastavnika	43
5.4.4	Nedeljni budžet procepa (broj sati pauze) nastavnika	44
5.5	Konfiguracija i provera	45
6	Evaluacija	47
6.1	Preliminarna evaluacija osnovnog modela	47
6.1.1	Metodologija	47
6.1.2	Rezultati	49
6.1.3	Analiza rezultata	49
6.2	Završna evaluacija proširenog modela	52
6.2.1	Metodologija	52
6.2.2	Veličina modela	54
6.2.3	Funkcija cilja u zavisnosti od vremena	55
6.2.4	Zaoštreno okruženje	56
6.2.5	Analiza rezultata	59
7	Zaključak i dalji rad	65
7.1	Zaključak	65
7.2	Dalji rad	66
	Literatura	68

Glava 1

Uvod

Generisanje rasporeda časova predstavlja jedan od najčešćih i najpraktičnijih kombinatornih problema sa kojim se suočavaju obrazovne institucije širom sveta. Svake godine, na početku akademskog semestra, univerziteti i fakulteti moraju da rasporede stotine predavanja, vežbi i laboratorijskih časova u ograničen broj učionica, uzimajući u obzir radne dane, raspoložive termine i različite vrste ograničenja.

Problem raspoređivanja časova (eng. *University Course Timetabling Problem* – UCTP) javlja se u više varijanti, a za opšte varijante poznato je da su NP-teške [4, 5]: algoritam polinomijalne složenosti za njihov opšti slučaj povlačio bi $P = NP$. Tradicionalno, rasporedi se prave ručno, što zahteva veliku količinu ljudskog rada, vremena i stručnosti. Čak i iskusni raspoređivači mogu da propuste kolizije ili da stvore neoptimalan raspored.

U ovom radu istražujemo dva pristupa za automatsko rešavanje UCTP-a:

1. **Programiranje ograničenja** (eng. *Constraint Programming* – CP) – deklarativna paradigma u kojoj se problem definiše preko skupa promenljivih, njihovih domena i ograničenja koja moraju biti zadovoljena. Koristimo Google OR-Tools CP-SAT rešavač [8, 12].
2. **Mešovito celobrojno programiranje** (eng. *Mixed Integer Programming* – MIP) – matematička optimizacija u kojoj se problem zapisuje linearnom funkcijom cilja i linearnim ograničenjima nad promenljivama koje mogu biti celobrojne ili neprekidne. Model koji razvijamo u ovom radu koristi isključivo binarne promenljive, pa je reč o binarnom celobrojnom linearnom modelu (0-1 ILP, odnosno BIP), posebnom slučaju šire klase MIP modela. Rešavamo ga SCIP rešavačem [1] integrisanim u OR-Tools.

1.1 Cilj rada

Rad ima tri cilja koja se nadovezuju jedan na drugi.

1. **Dve formulacije osnovnog problema.** Za isti skup tvrdih ograničenja razvijamo CP i MIP verziju, svaku u idiomu njene paradigme.
2. **Eksperimentalno poređenje.** Obe formulacije poredimo na tri skale realnih podataka (mali, srednji i veliki podskup) po vremenu konstrukcije modela, vremenu rešavanja, memorijskom otisku, veličini modela i validnosti dobijenog rasporeda.
3. **Proširenje i ponovna evaluacija.** Jednu od dve formulacije proširujemo mekim ograničenjima, funkcijom cilja i raspoređivanjem nastavnog osoblja, pa prošireni model ponovo evaluiramo.

Veličinu modela (broj promenljivih i ograničenja) posmatramo kao opisnu karakteristiku formulacije, a ne kao meru njenog kvaliteta: dve paradigme problem zapisuju na različite načine, pa sam broj promenljivih nije uporediv između njih. Kao merila koristimo ukupno vreme do dopustivog rešenja, ponašanje pri rastu instance i nezavisno proverenu validnost rasporeda.

Za nastavak rada izabran je CP model. Izbor je pragmatičan i vezan za konkretan problem, instance i implementacije opisane u ovom radu: CP-SAT je na svim ispitanim skalama brže dolazio do dopustivog rasporeda, a njegov zapis se pokazao pogodnim za postupno dodavanje novih pravila. MIP formulacija takođe daje validne rasporede, a drugačija formulacija ili drugi rešavač mogli bi dati drugačiji odnos rezultata.

1.2 Struktura rada

Rad je organizovan na sledeći način:

- U glavi 2 izlažemo teorijske osnove – definiciju problema raspoređivanja časova, paradigmu programiranja ograničenja i paradigmu celobrojnog linearnog programiranja – i na njih nadovezujemo formalnu specifikaciju problema u obe paradigme, sa promenljivama, domenima i ograničenjima (odeljak 2.4).

- U glavi 3 prikazujemo izvedbu: korišćene alate (odjeljak 3.1), model ulaznih podataka i formiranje sesija, implementaciju oba modela u OR-Tools-u i strukturno poređenje dve formulacije.
- U glavi 4 CP model nadograđujemo arhitekturom pravila, funkcijom cilja i novim ograničenjima kvaliteta rasporeda.
- U glavi 5 model proširujemo nastavnim osobljem: dodelom nastave, zajedničkim sesijama i pravilima koja štite raspored nastavnika.
- Glava 6 objedinjuje merenja. Odeljak 6.1 donosi preliminarnu evaluaciju osnovnog modela, sa poređenjem dve formulacije na tri skale problema, koja služi i kao osnov za izbor formulacije za dalji razvoj; odeljak 6.2 donosi završnu evaluaciju proširenog modela, sa naglaskom na vrednost funkcije cilja u zavisnosti od proteklog vremena izračunavanja.
- U glavi 7 sumiramo zaključke i navodimo pravce daljeg rada.

Glava 2

Teorijske osnove i formulacija problema

U ovoj glavi izlažemo teorijski okvir problema, a potom prelazimo na specifikaciju. Najpre definišemo problem raspoređivanja časova, zatim opisujemo paradigmu programiranja ograničenja, pa celobrojno linearno programiranje, i na kraju, u odeljku 2.4, zapisujemo konkretan model u obe paradigme – promenljive, njihove domene i ograničenja.

2.1 Problem raspoređivanja časova

Problem raspoređivanja univerzitetskih časova sastoji se od dodele časova predavanja, vežbi u raspoložive vremenske termine i učionice, tako da budu zadovoljena određena ograničenja.

Formalno, problem se može opisati sledećim elementima:

- **Skup sesija** $\mathcal{S} = \{s_1, s_2, \dots, s_n\}$ – časovi koje treba rasporediti. Svaka sesija pripada određenom predmetu i grupi studenata.
- **Skup vremenskih termina** $\mathcal{T} = \{t_1, t_2, \dots, t_m\}$ – raspoloživi termini u toku radne nedelje, određeni kombinacijom radnog dana i sata.
- **Skup učionica** $\mathcal{R} = \{r_1, r_2, \dots, r_k\}$ – prostorije u kojima se nastava može održati.
- **Skup ograničenja** $\mathcal{C}$ – pravila koja raspored mora da zadovolji.

Cilj je pronaći funkciju dodele $\sigma : \mathcal{S} \rightarrow \mathcal{T} \times \mathcal{R}$ takvu da su sva ograničenja iz $\mathcal{C}$ zadovoljena.

2.1.1 Vrste ograničenja

Ograničenja delimo u dve kategorije:

1. **Tvrda ograničenja** (eng. *hard constraints*) – ograničenja koja moraju biti bezuslovno zadovoljena. Rešenje koje krši neko tvrdo ograničenje nije dopustivo. Primeri:
 - Nikoje dve sesije ne mogu se održavati u istoj učionici u isto vreme.
 - Jedna grupa studenata ne može istovremeno pohađati dva različita predavanja.
 - Sesija koja zahteva računare mora biti raspoređena u učionicu opremljenu računarima.
2. **Meka ograničenja** (eng. *soft constraints*) – ograničenja čije zadovoljenje je poželjno, ali ne i obavezno. Njihovo kršenje ne čini rešenje nedopustivim, već samo lošijim. Primeri:
 - Ravnomerna raspodela časova tokom nedelje.
 - Minimizacija broja promena lokacija za jednu grupu u toku dana.
 - Izbegavanje procepa (eng. *gap*) – slobodnih sati između dva časa – u rasporedu za studente.

U prvom delu ovog rada bavimo se isključivo tvrdim ograničenjima, tj. pronalaženjem dopustivog rasporeda (eng. *feasibility problem*). Proširenje mekim ograničenjima i optimizacija kvaliteta rasporeda predviđeni su za drugi deo rada.

2.1.2 NP-težina problema

Raspoređivanje nastave nije jedan problem, već porodica varijanti koje se razlikuju po tome šta se raspoređuje i koja ograničenja važe, pa se i složenost mora vezati za tačno određenu varijantu. Even, Itai i Shamir [4] pokazali su da je jedna ograničena varijanta problema raspoređivanja nastave po razredima i nastavnicima NP-kompletna, svodenjem problema 3-SAT na nju (teorema 1, str. 692), i odatle

zaključuju da su i uobičajene, opštije varijante NP-kompletne. Garey i Johnson [5] uvrstili su taj rezultat u svoju katalogizaciju NP-kompletnih problema.

Za ovaj rad je dovoljna posledica tog rezultata: kada je neka varijanta NP-teška, algoritam polinomijalne složenosti za njen opšti slučaj povlačio bi $P = NP$. Pošto se za to ne očekuje da važi, u praksi se pribegava heuristikama ili egzaktnim metodama čija je najgora složenost eksponencijalna. Model koji razvijamo u ovom radu ne poklapa se ni sa jednom varijantom iz navedenih dokaza, ali deli njihovu kombinatornu strukturu, pa ga i posmatramo kao računski težak.

2.2 Programiranje ograničenja

Programiranje ograničenja (eng. *Constraint Programming* – CP) je deklarativna paradigma za rešavanje kombinatornih problema [16, 2]. Problem se formuliše kao *problem zadovoljenja ograničenja* (eng. *Constraint Satisfaction Problem* – CSP).

2.2.1 Problem zadovoljenja ograničenja (CSP)

CSP se formalno definiše trojkom $(\mathcal{X}, \mathcal{D}, \mathcal{C})$ gde je:

- $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ – skup promenljivih,
- $\mathcal{D} = \{D_1, D_2, \dots, D_n\}$ – skup domena, gde D_i sadrži moguće vrednosti promenljive x_i ,
- $\mathcal{C} = \{C_1, C_2, \dots, C_m\}$ – skup ograničenja, gde svako ograničenje C_j specificira dopustive kombinacije vrednosti nad podskupom promenljivih.

Rešenje CSP-a je dodela vrednosti svim promenljivama iz njihovih domena tako da su sva ograničenja zadovoljena.

2.2.2 Tehnike rešavanja

CP rešavači kombinuju sistematsku pretragu sa tehnikama propagacije ograničenja [16]:

1. **Vraćanje unazad** (eng. *backtracking*) – pretraga prostora rešenja dodeljujući vrednosti promenljivama jednu po jednu. Kada se nađe na ograničenje koje ne može biti zadovoljeno, vrši se vraćanje na poslednju tačku izbora.

2. **Propagacija ograničenja** (eng. *constraint propagation*) – smanjivanje domena promenljivih na osnovu ograničenja, pre i tokom pretrage. Ovo eliminiše nedopustive vrednosti rano i dramatično smanjuje prostor pretrage.
3. **Konzistentnost lukova** (eng. *arc consistency*) – tehnika propagacije koja obezbeđuje da za svaku vrednost u domenu jedne promenljive postoji kompatibilna vrednost u domenu svake povezane promenljive.

2.2.3 Globalna ograničenja

Bitna karakteristika CP-a je podrška za *globalna ograničenja* – specijalizovane strukture koje obuhvataju odnose između većeg broja promenljivih i koje imaju efikasne algoritme propagacije [14].

AllDifferent. Ograničenje `AllDifferent`($x_1, x_2, \dots, x_n$) zahteva da sve promenljive $x_1, \dots, x_n$ imaju međusobno različite vrednosti. Dok bi se isto moglo izraziti sa $\binom{n}{2}$ parnih nejednakosti, globalno ograničenje pruža značajno jaču propagaciju zahvaljujući algoritmima baziranim na sparivanju u bipartitnim grafovima [14].

AllowedAssignments. Ograničenje `AllowedAssignments` (poznato i kao *table constraint*) eksplicitno navodi dozvoljene vrednosti promenljivih pomoću tabele (skupa torki). Ovo je korisno za ograničenja tipa: da sesija može biti raspoređena samo u podskup učionica.

2.2.4 CP-SAT pristup

Moderna generacija CP rešavača, kakav je Google CP-SAT [12], kombinuje tehnike programiranja ograničenja sa tehnologijama SAT rešavača. Ovaj pristup se zasniva na *lenjom generisanju klauza* (eng. *lazy clause generation*) [11], gde se ograničenja propagiraju kao u klasičnom CP, ali se konflikti pamte u obliku SAT klauza, što sprečava ponavljanje istih pogrešnih izbora tokom pretrage. Ovo kombinuje snagu globalne propagacije sa efikasnošću učenja konflikata poznatom iz SAT rešavača.

2.3 Celobrojno linearno programiranje

Mešovito celobrojno programiranje (eng. *Mixed Integer Programming* – MIP) je paradigma matematičke optimizacije u kojoj se problem formuliše pomoću linearne funkcije cilja i linearnih ograničenja nad promenljivama koje mogu biti celobrojne ili neprekidne [17, 10].

2.3.1 Definicija MIP problema

MIP problem u opštem obliku glasi:

$$\min \quad \mathbf{c}^T \mathbf{x} \quad (2.1)$$

$$\text{t.d.} \quad A\mathbf{x} \leq \mathbf{b} \quad (2.2)$$

$$x_i \in \mathbb{Z}, \quad i \in I \quad (2.3)$$

$$x_j \geq 0, \quad j \notin I \quad (2.4)$$

gde je $\mathbf{c}$ vektor koeficijenata funkcije cilja, A matrica koeficijenata ograničenja, $\mathbf{b}$ vektor desnih strana, a I skup indeksa celobrojnih promenljivih.

Unutar ove klase razlikujemo uže slučajeve. Ako su sve promenljive celobrojne, reč je o *celobrojnom linearnom programu* (eng. *Integer Linear Program* – ILP). Ako su uz to sve promenljive binarne, $x_i \in \{0, 1\}$, govori se o *binarnom celobrojnom linearnom programu* (0-1 ILP, odnosno BIP).

Formulacija razvijena u ovom radu (odeljak 2.4.3) pripada poslednjoj klasi: sve njene promenljive su binarne, pa je reč o 0-1 ILP modelu, posebnom slučaju opšteg MIP oblika (2.1)–(2.4). U tabelama i u tekstu zadržana je kraća oznaka MIP/SCIP, koja imenuje paradigmu i rešavač.

Osnovni model, izložen u odeljku 2.4, postavljen je kao problem dopustivosti: funkcija cilja iz izraza (2.1) se ne koristi, već se traži bilo koja dodela vrednosti koja zadovoljava sva linearna ograničenja. Prošireni CP model iz glava 4 i 5, nasuprot tome, uvodi meka ograničenja i funkciju cilja (4.1) koja se minimizuje.

2.3.2 SCIP rešavač

SCIP (Solving Constraint Integer Programs) [1] je akademski rešavač za celobrojno i mešovito celobrojno programiranje. Kao i drugi rešavači ove klase [17], oslanja se na opuštanje uslova celobrojnosti (LP relaksacija), koje daje donju granicu funkcije cilja, na grananje i odsecanje (eng. *branch and bound*) i na ravni odsecanja (eng. *cutting planes*), uz propagaciju ograničenja i heuristike. U ovom radu koristimo SCIP integrisan u biblioteku Google OR-Tools.

2.4 Formalna specifikacija

2.4.1 Zajednička notacija

Obe formulacije opisuju isti problem nad istim ulazom, pa oznake uvodimo na jednom mestu. Sesija je ovde jednosatni čas jedne kohorte iz jednog predmeta; kako se od ulaznih podataka dobija skup sesija opisuje odeljak 3.2.3, a ovde taj skup uzimamo kao dat.

- $\mathcal{S}$ – skup sesija, indeksiran sa $s \in \{0, 1, \dots, |\mathcal{S}| - 1\}$;
- $\mathcal{D}$ – skup radnih dana i $D = |\mathcal{D}|$ njihov broj; $\mathcal{H}$ – skup raspoloživih sati u toku jednog dana i $H = |\mathcal{H}|$ njihov broj; $\mathcal{R}$ – skup učionica i $R = |\mathcal{R}|$ njihov broj. Dani, sati i učionice indeksiraju se redom sa $d \in \{0, \dots, D - 1\}$, $h \in \{0, \dots, H - 1\}$ i $r \in \{0, \dots, R - 1\}$;
- $\mathcal{G}$ – skup grupa studenata i $G = |\mathcal{G}|$ njihov broj;
- $\mathcal{S}_g \subseteq \mathcal{S}$ – skup sesija koje pohađa grupa $g \in \mathcal{G}$. Sesija koju zajednički sluša više grupa pripada skupu svake od njih (odeljak 5.3);
- $\text{comp}_s \in \{0, 1\}$ – oznaka da li sesija s zahteva računarsku učionicu;
- $\mathcal{R}_{\text{comp}} = \{r \in \mathcal{R} : \text{hasComputers}(r)\}$ – skup učionica opremljenih računarima, gde je $\text{hasComputers}(r)$ atribut učionice iz ulaza;
- $\text{Elig}(s) \subseteq \mathcal{R}$ – skup učionica u kojima sesija s sme da se održi:

$$\text{Elig}(s) = \begin{cases} \mathcal{R}_{\text{comp}}, & \text{comp}_s = 1, \\ \mathcal{R}, & \text{comp}_s = 0. \end{cases}$$

Rešenje pridružuje svakoj sesiji s trojku $(d_s, h_s, r_s) \in \mathcal{D} \times \mathcal{H} \times \mathcal{R}$ – dan, sat i učionicu. Tvrda ograničenja problema, zajednička za obe formulacije, označavamo istim oznakama u oba modela:

- **HC1** – svaka sesija dobija tačno jedan termin i tačno jednu učionicu;
- **HC2** – u jednoj učionici u jednom terminu održava se najviše jedna sesija;
- **HC3** – nijedna grupa nema dve sesije u istom terminu;
- **HC4** – sesija koja zahteva računare održava se u učionici iz skupa $\mathcal{R}_{\text{comp}}$.

Zahtev za računarima jedini je tvrdi uslov na izbor učionice: kapacitet ne ulazi u $\text{Elig}(s)$, već se tek u glavi 4 uvodi kao meko ograničenje (odeljak 4.6). Osnovni model radi u režimu izvodljivosti – traži se bilo koje rešenje koje zadovoljava sva tvrda ograničenja, bez funkcije cilja.

2.4.2 CP model: promenljive, domen i ograničenja

Promenljive i domen. Za svaku sesiju $s \in \{0, 1, \dots, |\mathcal{S}|-1\}$ definišemo sledeće celobrojne promenljive:

$$\text{day}_s \in \{0, 1, \dots, D-1\} \quad (2.5)$$

$$\text{slot}_s \in \{0, 1, \dots, H-1\} \quad (2.6)$$

$$\text{room}_s \in \{0, 1, \dots, R-1\} \quad (2.7)$$

Dodatno, definišemo tri izvedene promenljive:

$$\text{flatTime}_s = \text{day}_s \cdot H + \text{slot}_s \quad (2.8)$$

$$\text{roomTime}_s = \text{room}_s \cdot (D \cdot H) + \text{flatTime}_s \quad (2.9)$$

$$\text{loc}_s = \text{loc}(\text{room}_s) \quad (2.10)$$

Promenljiva flatTime_s linearizuje par (dan, sat) u jedinstven indeks unutar nedelje, a roomTime_s trojku (učionica, dan, sat) u jedinstven indeks preko svih učionica i termina. Promenljiva loc_s nosi lokaciju dodeljene učionice, dobijenu indeksiranjem niza lokacija vrednošću room_s ; ona ne sužava skup dopustivih rešenja osnovnog modela, ali je koriste lokacijska pravila iz glava 4 i 5, pa je navodimo već ovde. Izvedene promenljive nisu automatski vezane za osnovne: one se uvode kao nove celobrojne promenljive, a jednakosti (2.8)–(2.10) dodaju se eksplicitno kao ograničenja. Ukupno model ima $6 \cdot |\mathcal{S}|$ celobrojnih promenljivih.

HC1: Svaka sesija dobija tačno jedan termin i učionicu. U CP formulaciji ovaj uslov ne zahteva nijedno ograničenje: svaka sesija ima po jednu promenljivu za dan, sat i učionicu, pa svaka potpuna dodela vrednosti daje tačno jedan termin i tačno jednu učionicu po sesiji.

HC2: Najviše jedna sesija u datoj učionici u datom terminu. Nikoje dve sesije ne mogu se održavati u istoj učionici u isto vreme:

$$\text{AllDifferent}(\text{roomTime}_0, \text{roomTime}_1, \dots, \text{roomTime}_{|\mathcal{S}|-1}) \quad (2.11)$$

Ovo globalno ograničenje obuhvata sve sesije i garantuje da su sve trojke (d_s, h_s, r_s) međusobno različite.

HC3: Bez kolizija unutar grupe. Nikoje dve sesije iste grupe ne smeju biti u isto vreme:

$$\text{AllDifferent}(\{\text{flatTime}_s : s \in \mathcal{S}_g\}), \quad \forall g \in \mathcal{G} \quad (2.12)$$

Oba ograničenja imaju isti oblik, samo nad različitim skupovima promenljivih: HC2 nad svim vrednostima `roomTime`, a HC3 nad vrednostima `flatTime` jedne grupe. Sesija ulazi u HC3 za *svaku* grupu kojoj pripada, što je bitno za zajedničke sesije koje istovremeno pohađa više grupa.

HC4: Ograničenje računarskih učionica. Ako sesija s zahteva računare, njen izbor učionice je ograničen na učionice sa računarima:

$$\text{comp}_s = 1 \implies \text{room}_s \in \mathcal{R}_{\text{comp}} \quad (2.13)$$

U implementaciji se ovo zapisuje tabelarnim ograničenjem, koje u OR-Tools-u nosi ime `AddAllowedAssignments` (odeljak 3.3.3). Pošto je `comps` atribut sesije, ograničenje se odnosi samo na one sesije čiji tip nastave traži računare: predavanja iz predmeta čije vežbe traže računare ostaju slobodna da idu u bilo koju učionicu.

Veličina modela. Broj celobrojnih promenljivih je $6|\mathcal{S}|$. Broj ograničenja koja model prijavljuje jeste

$$3|\mathcal{S}| + 1 + G + |\{s \in \mathcal{S} : \text{comp}_s = 1\}|, \quad (2.14)$$

gde $3|\mathcal{S}|$ dolazi od tri jednakosti koje vezuju izvedene promenljive za osnovne, jedinica od globalnog ograničenja HC2, G od po jednog globalnog ograničenja HC3 za svaku grupu, a poslednji sabirak od tabelarnog ograničenja HC4, koje se dodaje samo sesijama sa zahtevom za računarima. Na instancama iz odeljka 6.1 izraz (2.14) daje izmerene vrednosti: za MATF-S je $3 \cdot 204 + 1 + 8 + 28 = 649$, za MATF-M $3 \cdot 398 + 1 + 16 + 58 = 1269$ i za MATF-L $3 \cdot 722 + 1 + 30 + 122 = 2319$. Broj ograničenja je pri tome slaba mera obima posla: jedno globalno ograničenje HC2 obuhvata sve sesije instance.

2.4.3 MIP model: promenljive, domen i ograničenja

Promenljive i domen. Za svaku sesiju s i svaku dopustivu kombinaciju (dan, sat, učionica) definišemo binarnu promenljivu:

$$x_{s,d,h,r} \in \{0, 1\}, \quad \forall s \in \mathcal{S}, \forall d \in \mathcal{D}, \forall h \in \mathcal{H}, \forall r \in \text{Elig}(s) \quad (2.15)$$

Promenljiva $x_{s,d,h,r}$ ima vrednost 1 kada se sesija s održava u danu d , satu h i učionici r . Promenljive za učionice van skupa $\text{Elig}(s)$ se uopšte ne kreiraju, čime je ograničenje HC4 zadovoljeno implicitno, kroz domen modela, a ne kroz zasebno ograničenje. Broj binarnih promenljivih je

$$D \cdot H \cdot \sum_{s \in \mathcal{S}} |\text{Elig}(s)| \leq D \cdot H \cdot R \cdot |\mathcal{S}|, \quad (2.16)$$

dakle $O(|\mathcal{S}| \cdot D \cdot H \cdot R)$

Sve promenljive modela su binarne, a sva ograničenja linearna, pa je ovo binarni celobrojni linearni model (0-1 ILP), poseban slučaj opšteg MIP oblika iz odeljka 2.3. Rešava ga SCIP, odakle i oznaka MIP/SCIP koju koristimo u tabelama.

HC1: Svaka sesija raspoređena tačno jednom.

$$\sum_{d \in \mathcal{D}} \sum_{h \in \mathcal{H}} \sum_{r \in \text{Elig}(s)} x_{s,d,h,r} = 1, \quad \forall s \in \mathcal{S} \quad (2.17)$$

HC2: Najviše jedna sesija u datoj učionici u datom terminu.

$$\sum_{s : r \in \text{Elig}(s)} x_{s,d,h,r} \leq 1, \quad \forall d \in \mathcal{D}, \forall h \in \mathcal{H}, \forall r \in \mathcal{R} \quad (2.18)$$

HC3: Bez kolizija unutar grupe.

$$\sum_{s \in \mathcal{S}_g} \sum_{r \in \text{Elig}(s)} x_{s,d,h,r} \leq 1, \quad \forall g \in \mathcal{G}, \forall d \in \mathcal{D}, \forall h \in \mathcal{H} \quad (2.19)$$

Veličina modela. Broj linearnih ograničenja je najviše

$$\underbrace{|\mathcal{S}|}_{\text{HC1}} + \underbrace{DHR}_{\text{HC2}} + \underbrace{GDH}_{\text{HC3}}, \quad (2.20)$$

dakle $O(|\mathcal{S}| + DHR + GDH)$. Kraći zapis $O(DHR + GDH)$ opravdan je kada važi $|\mathcal{S}| \leq DHR$, što na instancama iz ovog rada jeste slučaj (722 naspram 1740 na najvećoj skali).

Kao i CP model iz prethodnog pododeljka, osnovni MIP model radi u režimu izvodljivosti, bez funkcije cilja. Meka ograničenja i funkcija cilja uvode se tek u proširenom CP modelu (glava 4).

2.4.4 Ekvivalentnost dve formulacije

Ista četiri tvrda uslova zapisana su u dva idioma. Uslov HC1 u MIP modelu traži jednakost (2.17), dok u CP modelu važi po konstrukciji, jer svaka sesija ima po jednu promenljivu za dan, sat i učionicu. Uslov HC2 je u CP modelu ograničenje **AllDifferent** nad `roomTime`, a u MIP modelu familija nejednakosti (2.18), po jedna za svaku trojku (d, h, r) . Uslov HC3 je u CP modelu **AllDifferent** nad `flatTime` po grupi, a u MIP modelu nejednakost (2.19) za svaku trojku (g, d, h) . Uslov HC4 oba modela izražavaju kroz isti skup $\text{Elig}(s)$ – CP eksplicitnim ograničenjem nad domenom promenljive `rooms`, a MIP izostavljanjem promenljivih za nedozvoljene učionice.

Naredna glava opisuje alate kojima su te specifikacije implementirane, način na koji se od ulaznih podataka dobijaju sesije, i sam kod dva modela.

Glava 3

Implementacija i korišćene tehnologije

U ovoj glavi prelazimo sa specifikacije na izvedbu. Najpre ukratko predstavljamo alate i biblioteke korišćene u implementaciji, zatim opisujemo ulazne podatke i način na koji se od njih formiraju sesije koje treba rasporediti, pa prikazujemo kod kojim se CP i MIP specifikacija iz glave 2 prevode u modele dva rešavača. Glavu zatvaramo strukturnim poređenjem dve formulacije.

3.1 Alati i biblioteke

Ovde dajemo kratak pregled alata i tehnologija korišćenih u implementaciji, u meri u kojoj su potrebni za razumevanje narednih odeljaka. Projekat je napisan u programskom jeziku Python [13] i koristi Bazel [7] kao sistem za gradnju i upravljanje zavisnostima. Izvorni kod, ulazni podaci i izvor ovog rada javno su dostupni na adresi <https://github.com/dusantrtica/matf-master>.

3.1.1 Google OR-Tools

Google OR-Tools [8] je biblioteka otvorenog koda za kombinatornu optimizaciju razvijena od strane Google-a. Nudi rešavače za više tipova problema, uključujući:

- **CP-SAT** – rešavač za programiranje ograničenja sa SAT tehnikama, opisan u odeljku 2.2. U Pythonu se koristi putem modula `ortools.sat.python.cp_model`. Ključne klase su `CpModel` (za definisanje promenljivih i ograničenja) i `CpSolver` (za pokretanje pretrage).

- **Linear solver wrapper (pywraplp)** – uniformni interfejs za više MIP rešavača, od kojih u ovom radu koristimo SCIP. Ključna klasa je `Solver`, sa metodama za kreiranje binarnih promenljivih (`BoolVar`), dodavanje linearnih ograničenja (`Add`) i pokretanje rešavanja (`Solve`).

OR-Tools je izabran zato što je otvorenog izvornog koda i što kroz jedinstven Python API pokriva obe paradigme, pa se dve formulacije mogu porediti bez razlika koje bi poticale od različitih biblioteka.

3.1.2 Bazel

Bazel [7] je sistem za gradnju i testiranje softvera. U ovom projektu koristi se zbog hermetičkih, reproducibilnih gradnji: iste verzije zavisnosti i isti izvorni kod daju isti rezultat na svakoj mašini.

3.1.3 Python biblioteke

Ulazni podaci se učitavaju i validiraju pomoću biblioteke Pydantic [3], koja Python anotacije tipova koristi kao specifikaciju ulaza. Svaka entitetska klasa (`Settings`, `Classroom`, `Course`, itd.) definisana je kao `pydantic.dataclass`, pa neispravan ulaz otkazuje već pri parsiranju, a ne u rešavaču; deklaracija `Field(alias=...)` pri tome preslikava `camelCase` imena iz JSON-a u `snake_case` imena u Python kodu. Konkretan model ulaznih podataka daje odeljak 3.2.1.

Pored toga, `openpyxl` [6] koristi se za izvoz rasporeda u Excel fajl sa posebnim radnim listom za svaku grupu studenata, `PyFunctional` [15] za fluent filtriranje listi predmeta i učionica, a `pytest` [9] za testove koji pokrivaju parsiranje ulaza, formiranje grupa i sesija i proveru da rešavači daju dopustiva rešenja. Testovi se pokreću kroz Bazel.

3.2 Ulazni podaci i sesije

3.2.1 Model ulaznih podataka

Ulazni podaci su definisani u JSON formatu i modelovani pomoću Pydantic klasa u modulu `model.py`; njihov zapis u kodu dat je uz ostatak implementacije, u listingu 3.1.

Struktura ulaznih podataka je sledeća:

1. **Settings** – podešavanja rasporeda:
 - **workingDays** – lista radnih dana (Ponedjeljak, Utorak, ..., Petak),
 - **startHour**, **endHour** – početak i kraj radnog vremena (8–20),
 - **duration** – trajanje jednog časa predavanja i vežbi (1 sat).
2. **Location** – lokacije/zgrade fakulteta (Studentski trg, Svetog Nikole, Jagićeva).
3. **Classroom** – učionice sa atributima:
 - identifikator, naziv, lokacija,
 - da li poseduje računare (**hasComputers**),
 - kapacitet.
4. **Smer** – smerovi studija (Profesor matematike i računarstva, Teorijska matematika, Informatika, itd.). U izvornom kodu odgovara klasi **StudyTrack**, odnosno polju **trackId** u JSON ulazu.
5. **Course** – predmeti sa:
 - identifikatorom, nazivom, semestrom, smerom,
 - kvotom (**Quota**) – koliko časova predavanja i vežbi nedeljno,
 - zahtevom za računarima (**needsComputers**), zadatim *odvojeno za predavanja i za vežbe*.
6. **StudentsEnrolled** – broj upisanih studenata po smeru i semestru.

Sve navedene celine objedinjuje korenska klasa **SchedulingInput**, koja odgovara jednom ulaznom JSON fajlu.

3.2.2 Realni podaci i poreklo ulaza

Sve evaluacije u ovom radu koriste jedan skup podataka, koji predstavlja letnji semestar nastave na Matematičkom fakultetu (**input_full_2_semester.json**, uz **staff_2_semester.json**):

- smerovi,
- predmeti raspoređena po semestrima (2, 4, 6 i 8),

- učionice i lokacije,
- upisani studenti,
- nastavnici i dodele nastave,
- radna nedelja i radno vreme fakulteta.

Iz plana i programa studija preuzeti su smerovi, predmeti, semestri u kojima se predmeti slušaju i nedeljne kvote časova predavanja i vežbi. Sa sajta rasporeda su preuzete učionice. Njihov kapacitet je podatak koji nije bio na raspolaganju, pa su brojevi izmišljeni. Takođe, podatak o tome da li učionice poseduju računare je proizvoljno procenjen.

Broj upisanih studenata je takođe procenjen. Podrazumevana veličina grupe od 50 je takođe proizvoljno određena. Ravnomerna podela na grupe je pojednostavljena: stvarne grupe nisu iste veličine.

3.2.3 Formiranje sesija

Pre nego što se pokrene rešavač, ulazni podaci se transformišu u listu *sesija* – atomskih jedinica raspoređivanja. Ovaj proces se odvija u modulu `data.py` i sastoji se od sledećih koraka:

1. **Formiranje grupa studenata.** Grupa je kohorta koja zajedno sluša nastavu i formira se na jedan od tri načina, po prioritetu (funkcija `build_groups` u modulu `data.py`):
 - ako je zadat fajl sa osobljem u kome dodele nastave imenuju grupe (polje `groupIds`), imena grupa se uzimaju odatle, a njihove veličine iz broja upisanih – tako radi realni ulaz i o tome detaljno govori odeljak 5.2;
 - inače, ako ulazni JSON ima eksplicitnu `groups` sekciju, grupe se koriste onako kako su zadate;
 - inače se za svaki par (smer, semestar) upis od n studenata deli na $\lceil n/50 \rceil$ grupa, gde je 50 podrazumevana veličina grupe (konstanta `GROUP_SIZE`). Na primer, za Informatiku u 2. semestru sa 100 studenata formiraju se dve grupe.
2. **Dodeljivanje predmeta grupama.** Svaka grupa pohada predmete koji odgovaraju njenom smeru i semestru.

3. **Generisanje sesija.** Za svaki predmet i svaku grupu, kreira se onoliko sesija koliko to nalaže kvota: `quota.theory` sesija tipa “predavanje” i `quota.practice` sesija tipa “vežbe”.

Svaka sesija s je okarakterisana sledećim atributima:

- identifikator grupe (g_s),
- identifikator predmeta,
- da li zahteva računare ($comp_s$),
- tip časa (predavanje ili vežbe).

Za sesiju s , rešavač treba da odredi trojku (d_s, h_s, r_s) – dan, sat i učionicu u kojima će se čas održati.

3.2.4 Obim modela i pretpostavke

Pošto se veći deo ulaza priprema pre pokretanja rešavača, korisno je izričito reći šta rešavač odlučuje, a šta ne.

Rešavač odlučuje isključivo o trojci (d_s, h_s, r_s) za svaku sesiju. Van njegove nadležnosti ostaje sve ostalo: grupe studenata formiraju se pre rešavanja, na jedan od tri opisana načina, i rešavač ne raspoređuje pojedinačne studente u grupe niti menja njihov broj i veličinu; nastavnici se preuzimaju iz unapred zadatih dodela nastave (glava 5), pa rešavač ne bira ko drži koji čas, već samo vodi računa da tako dodeljen nastavnik nema dva časa u istom terminu; nedeljni broj časova po predmetu dolazi iz kvote, a sastav kohorti koje zajedno slušaju predavanje iz dodela nastave.

Osnovna jedinica raspoređivanja je *jednosatna sesija*. Predmet sa tri časa vežbi nedeljno daje tri nezavisne sesije, a pravilo za spajanje časova u blokove (odeljak 4.3) može zahtevati da one budu u uzastopnim terminima istog dana, čime se dobija efekat dvočasa ili tročasa. Ulazni model ima polje **duration**, ali ga implementacija ne koristi i model pretpostavlja vrednost 1: vremenska osa je diskretizovana na jednočasovne termine, a ograničenje jedinstvenosti učionice poredi samo termine, ne i intervale. Podrška za časove proizvoljnog trajanja zahtevala bi da promenljiva termina nosi i trajanje i da se preklapanje sprečava ograničenjem nad intervalima; to je naznačeno kao pravac daljeg rada.

Predavanja i vežbe istog predmeta su nezavisne jedinice raspoređivanja: kvota (theory i practice), nastavnik, sastav slušalaca i zahtev za računarima zadaju se odvojeno po tipu nastave, pa se termin i učionica za njih biraju nezavisno.

3.3 Implementacija

Obe specifikacije prevedene su u kod korišćenjem biblioteke OR-Tools: CP model u klasi `SimpleCPSolver` (modul `cp_solver.py`), a MIP model u klasi `SimpleMIPSolver` (modul `mip_solver.py`). U nastavku prikazujemo isečke koji pokazuju kako se ulazni podaci i elementi specifikacije iz prethodnog odeljka zapisuju u kodu.

3.3.1 Ulazni model u kodu

Entiteti opisani u odeljku 3.2.1 definisani su kao Pydantic klase, sa tipovima koji služe i kao specifikacija ulaza i kao provera koju biblioteka sprovodi pri učitavanju. Klase pojedinačnih entiteta su proste liste polja, pa ih ne navodimo u celini; sve ih ilustruje učionica, na kojoj se vidi i uobičajeni obrazac – poziv `Field` povezuje ime polja u Python kodu sa imenom ključa u JSON ulazu. Korenska klasa `SchedulingInput` objedinjuje entitete i odgovara jednom ulaznom fajlu:

Listing 3.1: Model ulaznih podataka (skraćeno)

```
1 @dataclass(config=_config)
2 class Classroom:
3     id: int
4     name: str
5     loc_id: int = Field(alias="locId")
6     has_computers: bool = Field(alias="hasComputers")
7     capacity: int = 0
8
9 @dataclass(config=_config)
10 class SchedulingInput:
11     settings: Settings
12     locations: List[Location]
13     classrooms: List[Classroom]
14     courses: List[Course]
15     tracks: List[StudyTrack] = Field(
16         validation_alias=AliasChoices("tracks", "departments"))
```

```
17     students_enrolled: List[StudentsEnrolled] = Field(  
18         alias="studentsEnrolled")  
19     rules: dict[str, RuleConfig] = Field(default_factory=dict)  
20     groups: List[GroupDef] = Field(default_factory=list)
```

3.3.2 Klasa sesije

Jedinica koja se raspoređuje je *sesija* – jednosatni čas jedne kohorte iz jednog predmeta, uveden u odeljku 3.2.3 – pa je klasa `Session` (modul `data.py`) središnja struktura podataka implementacije: ulazni model se pre rešavanja prevodi u listu sesija, a oba rešavača i sva pravila iz glava 4 i 5 rade isključivo nad tom listom.

Listing 3.2: Klasa `Session` – jedinica raspoređivanja

```
1 class Session:  
2     def __init__(  
3         self, id, group_ids, track_id, course_id,  
4             needs_computers,  
5         session_type: str, teacher_id: int | None = None,  
6         size: int = 0,  
7     ):  
8         self.id = id  
9         # sve grupe koje prisustvuju sesiji; vise grupa =  
10            zajednicko  
11         # predavanje (npr. oba toka slusaju isti cas)  
12         self.group_ids: list[str] = list(group_ids)  
13         self.course_id = course_id  
14         self.track_id = track_id  
15         self.needs_computers = needs_computers  
16         self.session_type = session_type  
17         # None znaci da nema dodeljenog nastavnika pa se staff  
18            pravila  
19         # ne primenjuju na ovu sesiju  
20         self.teacher_id = teacher_id  
21         # ukupan broj studenata (zbir po grupama), za kapacitet  
22            ucionice  
23         self.size = size
```

Za razliku od ulaznih klasa, `Session` nije Pydantic klasa jer se ne učitava iz fajla, već se izvodi iz ulaza funkcijom `generate_sessions`, koja za svaki predmet formira kohorte i za svaku od njih onoliko sesija koliko nalaže kvota.

Polja klase su upravo atributi sesije iz formalne specifikacije (odjeljak 2.4.1):

- `group_ids` – grupe koje pohađaju sesiju. Lista, a ne jedna grupa, jer zajedničko predavanje više grupa jeste jedna sesija; preko ovog polja se gradi skup $\mathcal{S}_g$ iz ograničenja HC3, u koji sesija ulazi za svaku svoju grupu;
- `needs_computers` – oznaka `comps`, koja određuje skup dopustivih učionica `Elig(s)` u ograničenju HC4. Vrednost se preuzima sa predmeta, ali za tip nastave te sesije (odjeljak 3.2.4);
- `session_type` – „theory” ili „practice”; razdvaja predavanja od vežbi istog predmeta, koja su nezavisne sesije;
- `course_id`, `track_id` – predmet i smer, koji se ne koriste u tvrdim ograničenjima, nego u izveštaju i u pravilu za spajanje časova u blokove (odjeljak 4.3);
- `id` – čitljiv identifikator oblika *grupa_smer_predmet_tip_redni broj*, koji služi izveštaju i dijagnostici;
- `teacher_id` i `size` – nastavnik i ukupan broj studenata. U osnovnom modelu ove glave ostaju neiskorišćeni (podrazumevano `None` i 0); pune ih dodele nastave i veličine kohorti, a koriste ih pravila kapaciteta učionice (odjeljak 4.6) i pravila nastavnog osoblja (glava 5).

Ono što klasa *ne* sadrži jednako je važno: u njoj nema ni dana, ni sata, ni učionice. To su nepoznate koje rešavač određuje, pa sesija ostaje nepromenjena tokom rešavanja, a rezultat se čita iz promenljivih modela.

Veza između sesije i promenljivih uspostavlja se položajem u listi: indeks *s* iz formalne specifikacije je redni broj sesije u `self.sessions`, i on je ključ svih rečnika promenljivih u oba rešavača.

Listing 3.3: Sesije kao indeksni skup modela

```

1 self.sessions = generate_sessions(
2     scheduling_input, GROUP_SIZE, staff_input)
3 
```

```
4 for s in range(len(self.sessions)):
5     self.day_var[s] = self.model.NewIntVar(0, D - 1, f"day_{s}")
6     self.slot_var[s] = self.model.NewIntVar(0, H - 1, f"slot_{s}")
7     self.room_var[s] = self.model.NewIntVar(0, R - 1, f"room_{s}")
```

Zbog toga broj sesija neposredno određuje veličinu modela: u CP formulaciji svaka sesija donosi šest celobrojnih promenljivih, a u MIP formulaciji po jednu binarnu promenljivu za svaku kombinaciju dana, sata i dopustive učionice.

3.3.3 CP implementacija

Izvedene promenljive flatTime_s i roomTime_s uvode se kao nove celobrojne promenljive, a jednakosti (2.8) i (2.9) dodaju se eksplicitno:

Listing 3.4: Linearizacija termina i para (učionica, termin)

```
1 self.flat_time_var[s] = self.model.NewIntVar(
2     0, total_slots - 1, f"flat_time_{s}")
3 self.model.Add(
4     self.flat_time_var[s]
5     == self.day_var[s] * H + self.slot_var[s])
6
7 self.room_time_var[s] = self.model.NewIntVar(
8     0, R * total_slots - 1, f"room_time_{s}")
9 self.model.Add(
10    self.room_time_var[s]
11    == self.room_var[s] * total_slots + self.flat_time_var[s])
```

Ograničenja HC2 i HC3 svode se na isti poziv nad različitim skupovima promenljivih:

Listing 3.5: Ograničenja HC2 i HC3 u CP-SAT modelu

```
1 # HC2: nikoje dve sesije ne dele istu (ucionicu, dan, sat)
2 self.model.AddAllDifferent(list(self.room_time_var.values()))
3
4 # HC3: nikoje dve sesije iste grupe nisu u istom terminu
5 groups = defaultdict(list)
6 for s, session in enumerate(self.sessions):
```

```

7     for group_id in session.group_ids:
8         groups[group_id].append(s)
9
10    for group_id, session_indices in groups.items():
11        self.model.AddAllDifferent(
12            [self.flat_time_var[s] for s in session_indices])

```

Promenljiva loc_s iz jednakosti (2.10) dobija se indeksiranjem niza lokacija vrednošću $room_s$, pozivom `AddElement`. Ona ne utiče na skup dopustivih rešenja osnovnog modela, ali je koriste pravila vezana za lokacije u glavama 4 i 5. Sa njom model po sesiji ima šest celobrojnih promenljivih, koliko ih i merenje u odeljku 6.1 pokazuje.

Skup $Elig(s)$ iz ograničenja HC4 gradi se po sesiji, a ograničenje se dodaje samo kada je taj skup uži od skupa svih učionica. U suštini to su učionice koje zadovoljavaju zahtev za računarima, ako je sesija takva da zahteva računare.

3.3.4 MIP implementacija

Binarne promenljive $x_{s,d,h,r}$ kreiraju se samo za učionice iz skupa $Elig(s)$, koji vraća pomoćna metoda `_eligible_room_indices`:

Listing 3.6: Kreiranje binarnih promenljivih

```

1    for s, session in enumerate(self.sessions):
2        self.x[s] = {}
3        eligible = self._eligible_room_indices(session)
4        for d in range(D):
5            for h in range(H):
6                for r in eligible:
7                    self.x[s][(d, h, r)] = self.solver.BoolVar(
8                        f"x_{s}_{d}_{h}_{r}")

```

Svako od tri ograničenja zapisuje se kao familija linearnih nejednakosti ili jednakosti, po jedna za svaku kombinaciju indeksa nad kojima se kvantifikuje; rečnik `groups` gradi se isto kao u CP modelu:

Listing 3.7: Tvrdna ograničenja u MIP modelu

```

1    # HC1: svaka sesija rasporedjena tacno jednom
2    for s in range(S):
3        self.solver.Add(sum(self.x[s].values()) == 1)

```

```

4
5 # HC2: najviše jedna sesija u datoj (dan, sat, učionica)
6 for d in range(D):
7     for h in range(H):
8         for r in range(R):
9             vars_at_dhr = [
10                 self.x[s][(d, h, r)]
11                 for s in range(S) if (d, h, r) in self.x[s]
12             ]
13             if len(vars_at_dhr) > 1:
14                 self.solver.Add(sum(vars_at_dhr) <= 1)
15
16 # HC3: jedna grupa nema dve sesije u istom terminu
17 for group_id, session_indices in groups.items():
18     for d in range(D):
19         for h in range(H):
20             group_vars = [
21                 self.x[s][(d, h, r)]
22                 for s in session_indices for r in range(R)
23                 if (d, h, r) in self.x[s]
24             ]
25             if len(group_vars) > 1:
26                 self.solver.Add(sum(group_vars) <= 1)

```

3.4 Strukturno poređenje formulacija i rešavača

Tabela 3.1 sumira strukturne razlike između dve formulacije i dva rešavača.

Tabela 3.1: Strukturno poređenje formulacija i rešavača

Aspekt	CP-SAT	MIP/SCIP
Tip promenljivih	Celobrojne	Binarne
Broj promenljivih	$6 \mathcal{S} $	$DH \sum_s \text{Elig}(s) $
Tip ograničenja	Globalna	Linearne jednakosti i nejednakosti
Broj ograničenja	$3 \mathcal{S} + 1 + G + \{s : \text{comp}_s\} $	najviše $ \mathcal{S} + DHR + GDH$

CP formulacija koristi mali broj celobrojnih promenljivih sa globalnim ograničenjima, dok MIP formulacija eksplicitno enumeriše sve moguće dodele kroz binarne

promenljive. Brojevi iz tabele nisu mera obima posla koji rešavač obavlja, nego opis zapisa: jedno globalno ograničenje HC2 u CP modelu obuhvata sve sesije instance, dok mu u MIP modelu odgovara *DHR* zasebnih nejednakosti. Razlika u zapisu ipak ima merljivu posledicu na memorijski otisak i na vreme konstrukcije modela.

Naredne dve glave prate dalji razvoj jedne od njih, CP modela, koji se proširuje pravilima kvaliteta i raspoređivanjem nastavnog osoblja.

Glava 4

Nadogradnja CP rešavača dodatnim ograničenjima

Za dalji razvoj sistema izabran je CP-SAT model. Izbor je pragmatičan i vezan za dve konkretne formulacije, dva rešavača i ispitane instance: CP-SAT je na njima brže dolazio do dopustivog rasporeda, a njegov zapis se pokazao pogodnim za postupno dodavanje pravila. Merenja na kojima izbor počiva data su u odeljku 6.1, pošto su obe evaluacije objedinjene u glavi 6. U ovoj glavi opisujemo nadogradnju modela dodatnim ograničenjima koja raspored približavaju realnim potrebama Matematičkog fakulteta. Za razliku od prvog dela rada, gde smo tražili bilo koje dopustivo rešenje, ovde uvodimo i funkciju cilja koja optimizuje kvalitet rasporeda kroz meka ograničenja.

Implementirana su četiri nova ograničenja:

- **spajanje časova u blokove** – časovi istog predmeta, grupe i tipa grupišu se u uzastopne blokove (na primer dvočasi i tročasi);
- **jedna lokacija po danu za grupu** – grupa studenata ne menja zgradu u toku jednog radnog dana;
- **bez procepa u rasporedu** – izbegavanje slobodnih sati između dva zauzeta sata u istom danu;
- **kapacitet učionice** – prekoračenje kapaciteta se kažnjava brojem nedostajućih mesta, ali nije zabranjeno.

4.1 Arhitektura pravila

Svako ograničenje je modelovano kao zaseban dodatak (eng. *plugin*) koji se priključuje rešavaču. Ovakva arhitektura omogućava da se pravila nezavisno uključuju, isključuju i konfigurišu, bez izmena u jezgru rešavača.

Osnovu čini apstraktna klasa `SchedulingRule` (modul `src/rules/base.py`):

Listing 4.1: Apstraktna klasa pravila

```
1 class SchedulingRule(ABC):
2     def __init__(self, enabled=True, penalty=0, **kwargs):
3         self.enabled = enabled
4         self.penalty = penalty
5
6     @property
7     def is_hard(self) -> bool:
8         return self.penalty == 0
9
10    @abstractmethod
11    def apply(self, solver) -> list[cp_model.IntVar]:
12        ...
```

Svako pravilo ima dva parametra: `enabled` (da li je pravilo aktivno) i `penalty` (cena kršenja). Vrednost `penalty` istovremeno određuje i *tip* ograničenja:

- `penalty == 0` – **tvrd**o (eng. *hard*) ograničenje. Mora biti zadovoljeno; u suptnom rešavač prijavljuje da je problem nedopustiv (INFEASIBLE). Metoda `apply` dodaje ograničenja direktno u model i vraća praznu listu.
- `penalty > 0` – **meko** (eng. *soft*) ograničenje. Rešavač pokušava da ga zadovolji, ali ga može prekršiti uz datu cenu. Ovo predstavlja težinski faktor prekršaja.

4.1.1 Konfiguracija pravila

Pravila se konfigurišu u ulaznom JSON fajlu pomoću sekcije `rules`, gde svaki ključ odgovara nazivu pravila:

Listing 4.2: Primer konfiguracije pravila u JSON-u

```
1 "rules": {
```

```
2     "joinSameClasses": { "enabled": true, "penalty": 0 },
3     "noGapsInSchedule": { "enabled": true, "penalty": 5 }
4 }
```

Ova konfiguracija se učitava u Pydantic model `RuleConfig` (modul `model.py`). U gornjem primeru, pravilo `joinSameClasses` je tvrdo (penalty 0), dok je `noGapsInSchedule` meko sa cenom 5 po prekršaju.

Pored polja `enabled` i `penalty`, `RuleConfig` ima i opciono polje `params` – rečnik koji je podrazumevano prazan. Njegov sadržaj se prosleđuje konstruktoru pravila kao imenovani argumenti, pa pravilo tim putem dobija svoje parametre. Taj mehanizam koriste pravila nastavnog osoblja (`maxDays` i `maxGapHours`, odeljak 5.5); postupak je za sva pravila isti, bez obzira na to da li konfiguracija dolazi iz osnovnog ili iz staff ulaznog fajla.

4.1.2 Primena pravila i funkcija cilja

Rešavač održava registar pravila `RULE_REGISTRY` koji preslikava naziv pravila u odgovarajuću klasu. Metoda `apply_rules` instancira sva uključena pravila, prikuplja promenljive prekršaja i formira funkciju cilja:

Listing 4.3: Primena pravila i formiranje funkcije cilja

```
1 def apply_rules(self):
2     self.penalty_vars = []
3     for rule_name, config in self.scheduling_input.rules.items():
4         :
5         if not config.enabled:
6             continue
7         rule_class = RULE_REGISTRY.get(rule_name)
8         rule = rule_class(
9             enabled=config.enabled, penalty=config.penalty, **
10             config.params
11         )
12         violations = rule.apply(self)
13         for v in violations:
14             self.penalty_vars.append((config.penalty, v))
15
16 if self.penalty_vars:
17     self.model.Minimize(
```

```

16         sum(weight * var for weight, var in self.
           penalty_vars)
17     )

```

Funkcija cilja je dakle težinska suma svih prekršaja mekih ograničenja:

$$\min \sum_r \text{penalty}_r \cdot \sum_{v \in V_r} v \quad (4.1)$$

gde je V_r skup promenljivih prekršaja koje vraća pravilo r , a penalty_r njegova cena. Tvrdna ograničenja ne učestvuju u funkciji cilja jer vraćaju praznu listu prekršaja.

Težine su heuristički izabrane, kao izraz pretpostavljenog redosleda važnosti, a ne merene na ishodima. One određuju kako rešavač trguje između pravila: pošto jedan procep u rasporedu grupe nosi 2 poena, a jedan dan preko limita nastavnika 3, rešavaču se isplati da nastavniku doda radni dan tek ako time ukloni bar dva procepa. Isto važi i za kapacitet: deset nedostajućih mesta nosi kaznu 10, pa preteže nad dodatnim radnim danom nastavnika, koji nosi 3. Drugačiji izbor težina daje drugačiji optimum, pa su i rezultati u odeljku 6.2 vezani za ovaj konkretan izbor.

4.2 CP-SAT tehnike modeliranja

Nova pravila koriste nekoliko naprednih tehnika modeliranja koje pruža CP-SAT [12]. U ovom odeljku ih ukratko objašnjavamo, jer se koriste u sve tri implementacije.

4.2.1 Reifikacija i `OnlyEnforceIf`

Reifikacija povezuje istinitost nekog ograničenja sa binarnom *indikatorskom* promenljivom (literalom). U CP-SAT, metoda `OnlyEnforceIf(b)` čini da ograničenje važi *samo ako* je literal b tačan:

Listing 4.4: Uslovno (reifikovano) ograničenje

```

1 # lit == 1 => flat_time[s] == t
2 model.Add(flat_time[s] == t).OnlyEnforceIf(lit)
3 # lit == 0 => flat_time[s] != t
4 model.Add(flat_time[s] != t).OnlyEnforceIf(lit.Not())

```

Navodenjem oba smera (za b i za $\neg b$, tj. `lit.Not()`) dobija se puna ekvivalencija $b \Leftrightarrow (\text{flat_time}_s = t)$, čime literal b tačno odražava istinitost ograničenja.

4.2.2 Logičko ILI preko AddMaxEquality

Za binarne promenljive (vrednosti 0 ili 1), maksimum je jednak logičkom ILI. CP-SAT to izražava ograničenjem `AddMaxEquality`:

Listing 4.5: Logičko ILI nad binarnim promenljivama

```
1 # b == 1 ako je bar jedan od literala u "lits" jednak 1
2 model.AddMaxEquality(b, lits)    # b = max(lits) = OR(lits)
```

Ovo koristimo da utvrdimo da li je grupa zauzeta u nekom terminu (bar jedna sesija je u tom terminu).

4.2.3 AddBoolAnd i AddBoolOr

Ograničenja `AddBoolAnd` (konjunkcija) i `AddBoolOr` (disjunkcija) nameću da svi, odnosno bar jedan literal budu tačni. U kombinaciji sa `OnlyEnforceIf` koriste se za potpunu reifikaciju složenih logičkih izraza:

Listing 4.6: Reifikacija ekvivalencije logičkog izraza

```
1 # gap == 1 <=> (has_before AND has_after AND NOT busy)
2 model.AddBoolAnd([has_before, has_after, busy.Not()]).
   OnlyEnforceIf(gap)
3 model.AddBoolOr([has_before.Not(), has_after.Not(), busy]).
   OnlyEnforceIf(gap.Not())
```

Prvi red obezbeđuje smer $gap \Rightarrow (\dots)$, a drugi (primenom De Morganovih zakona) smer $\neg gap \Rightarrow \neg(\dots)$.

4.2.4 Indeksiranje niza preko AddElement

Ograničenje `AddElement(index, array, target)` nameće $target = array[index]$, gde je `index` promenljiva. Ovo omogućava “pretragu” kroz niz na osnovu vrednosti druge promenljive:

Listing 4.7: Indeksiranje niza promenljivom

```
1 # loc[s] == locations_of_classrooms[room_var[s]]
2 model.AddElement(room_var[s], locations_of_classrooms, loc[s])
```

4.3 Pravilo: spajanje časova u blokove

Pravilo `JoinSameClassesRule` (modul `rule_join_same_classes.py`) grupiše časove istog predmeta, grupe i tipa (predavanje/vežbe) u uzastopne blokove. Cilj je izbeći rascepkane rasporede gde su, na primer, četiri časa istog predmeta razbacana kroz nedelju.

4.3.1 Formiranje blokova

Sesije se najpre grupišu u *familije* po ključu (predmet, grupa, tip). Za svaku familiju, broj sesija se deli na blokove funkcijom `split_into_blocks`:

$$\text{split}(n) = \begin{cases} [n] & \text{ako } n \leq 3 \\ [2, 2] & \text{ako } n = 4 \end{cases} \quad (4.2)$$

Tako se tročasovni blok drži zajedno, dok se četiri časa dele na dva dvočasa. Unutar svakog bloka, prva sesija se bira kao *sidro* (eng. *anchor*), a ostale se vezuju za nju.

Funkcija je definisana samo za $n \leq 4$, što je dovoljno za razmatrane ulaze: u sva tri korišćena fajla najveća kvota jeste četiri časa nedeljno, i po tipu nastave i po predmetu. Veće kvote model, dakle, ne podržava; da bi ih podržao, definiciju (4.2) bi trebalo dopuniti pravilom podele za $n \geq 5$.

4.3.2 Tvrda varijanta

U tvrdoj varijanti, svaka sesija s_i u bloku (sa rednim brojem i u odnosu na sidro a) mora biti istog dana, u istoj učionici i u uzastopnom satu:

$$\text{day}_{s_i} = \text{day}_a \quad (4.3)$$

$$\text{slot}_{s_i} = \text{slot}_a + i \quad (4.4)$$

$$\text{room}_{s_i} = \text{room}_a \quad (4.5)$$

$$\text{slot}_a + |B| - 1 \leq H - 1 \quad (4.6)$$

gde je $|B|$ veličina bloka, a poslednji uslov obezbeđuje da blok stane unutar radnog vremena. U kodu:

Listing 4.8: Tvrdo spajanje bloka oko sidra

```

1 for i, s in enumerate(block[1:], start=1):
2     solver.model.Add(solver.day_var[s] == solver.day_var[anchor
3         ])
4     solver.model.Add(solver.slot_var[s] == solver.slot_var[
5         anchor] + i)
6     solver.model.Add(solver.room_var[s] == solver.room_var[
7         anchor])
8 solver.model.Add(solver.slot_var[anchor] + len(block) - 1 <= H -
9     1)

```

4.3.3 Meka varijanta

U mekoj varijanti, ista ograničenja se uvode uslovno, preko literala b (“blok je spojen”). Ako blok ne može biti spojen, b je netačno i uvodi se promenljiva prekršaja:

Listing 4.9: Meko spajanje sa promenljivom prekršaja

```

1 b = solver.model.NewBoolVar(f"joined_{block[0]}_{block[-1]}")
2 for i, s in enumerate(block[1:], start=1):
3     solver.model.Add(solver.day_var[s] == solver.day_var[anchor
4         ]).OnlyEnforceIf(b)
5     solver.model.Add(solver.slot_var[s] == solver.slot_var[
6         anchor] + i).OnlyEnforceIf(b)
7     solver.model.Add(solver.room_var[s] == solver.room_var[
8         anchor]).OnlyEnforceIf(b)
9
10 violation = solver.model.NewBoolVar(f"violation_{block[0]}_{
11     block[-1]}")
12 solver.model.Add(violation == 1).OnlyEnforceIf(b.Not())
13 solver.model.Add(violation == 0).OnlyEnforceIf(b)

```

Promenljiva `violation` ulazi u funkciju cilja (4.1), pa nespojen blok nosi kaznu koju rešavač minimizuje zajedno sa ostalim komponentama cilja.

4.4 Pravilo: jedna lokacija po danu za grupu

Pravilo `SingleLocationInDayForGroupRule` (modul `rule_single_location_in_day_for_group.py`) sprečava da grupa studenata menja zgradu (lokaciju) tokom

jednog radnog dana. Matematički fakultet izvodi nastavu na više lokacija (Studentski trg, zgrada Sv. Nikole, Jagićeva), pa bi promena zgrade između dva uzastopna časa bila nepraktična. Ovo je tvrdo ograničenje.

4.4.1 Mehanizam

Ključ implementacije je dvostruka primena ograničenja `AddElement`. Najpre, za svaku sesiju s uvodimo promenljivu loc_s koja predstavlja lokaciju dodeljene učionice:

$$loc_s = locOfClassroom[room_s] \quad (4.7)$$

Pošto je $room_s$ promenljiva, koristimo `AddElement` da indeksiramo niz lokacija učionica:

Listing 4.10: Kanalisanje učionice u lokaciju

```

1 loc[s] = solver.model.NewIntVar(0, max_loc, f"loc_{s}")
2 solver.model.AddElement(
3     solver.room_var[s], self.locations_of_classrooms, loc[s]
4 )

```

Zatim, za svaku grupu g i dan d uvodimo promenljivu $groupDayLoc_{g,d}$ – lokaciju te grupe tog dana. Za svaku sesiju s grupe g namećemo da njena lokacija bude jednaka lokaciji grupe na dan kada se sesija održava:

$$loc_s = groupDayLoc_g[day_s] \quad (4.8)$$

Ponovo koristimo `AddElement`, ovog puta indeksirajući niz dnevnih lokacija promenljivom day_s :

Listing 4.11: Vezivanje lokacije sesije za lokaciju grupe po danu

```

1 for g, indices in group_session_indices.items():
2     day_locs = [group_day_loc[g, d] for d in range(D)]
3     for s in indices:
4         # loc[s] == group_day_loc[g][day_var[s]]
5         solver.model.AddElement(solver.day_var[s], day_locs, loc
                                [s])

```

Pošto sve sesije grupe g u danu d moraju imati istu vrednost loc_s (jednaku $groupDayLoc_{g,d}$), rešavač je prinuđen da ih sve smesti u učionice na istoj lokaciji.

4.5 Pravilo: bez procepa u rasporedu

Pravilo NoGapsInScheduleRule (modul `rule_no_gaps_in_schedule.py`) cilja na kvalitet rasporeda iz ugla studenata: izbegava *procepe* – slobodne sate koji se nalaze između dva zauzeta sata u istom danu za istu grupu. Na primer, ako grupa ima čas u 8h i u 11h, a 9h i 10h su slobodni, to su dva procepa.

4.5.1 Matrica zauzetosti

Najpre se za svaku grupu gradi matrica zauzetosti $\text{busy}_{d,h}$ koja je tačna ako grupa ima neki čas u danu d , satu h . Za svaku sesiju i svaki termin $t = d \cdot H + h$ uvodimo reifikovani literal koji je tačan kada je sesija u tom terminu, a zatim ih agregiramo logičkim ILI preko `AddMaxEquality`:

Listing 4.12: Gradnja matrice zauzetosti

```

1 for s in sess:
2     lit = solver.model.NewBoolVar(f"at_{s}_{t}")
3     solver.model.Add(solver.flat_time_var[s] == t).OnlyEnforceIf
        (lit)
4     solver.model.Add(solver.flat_time_var[s] != t).OnlyEnforceIf
        (lit.Not())
5     lits.append(lit)
6 b = solver.model.NewBoolVar(f"busy_{g}_{d}_{h}")
7 solver.model.AddMaxEquality(b, lits)    # OR preko svih sesija
        grupe

```

4.5.2 Detekcija procepa

Sat h je procep ako je slobodan ($\neg \text{busy}_{d,h}$), a postoji zauzet sat i pre i posle njega u istom danu. Pomoćni literali `hasBefore` i `hasAfter` se računaju kao logičko ILI nad prefiksom, odnosno sufiksom reda zauzetosti. Sama promenljiva `gap` se reifikuje kao konjunkcija:

$$\text{gap}_{d,h} \Leftrightarrow \text{hasBefore}_{d,h} \wedge \text{hasAfter}_{d,h} \wedge \neg \text{busy}_{d,h} \quad (4.9)$$

Listing 4.13: Detekcija unutrašnjeg procepa

```

1 has_before = solver.model.NewBoolVar(f"hb_{g}_{d}_{h}")

```

```

2 solver.model.AddMaxEquality(has_before, busy_day[:h])
3 has_after = solver.model.NewBoolVar(f"ha_{g}_{d}_{h}")
4 solver.model.AddMaxEquality(has_after, busy_day[h + 1:])
5
6 gap = solver.model.NewBoolVar(f"gap_{g}_{d}_{h}")
7 solver.model.AddBoolAnd(
8     [has_before, has_after, busy_day[h].Not()]
9 ).OnlyEnforceIf(gap)
10 solver.model.AddBoolOr(
11     [has_before.Not(), has_after.Not(), busy_day[h]]
12 ).OnlyEnforceIf(gap.Not())

```

Rubni sati (prvi i poslednji u danu) ne mogu biti unutrašnji procepi, pa se preskaču.

4.5.3 Tvrda i meka varijanta

U tvrdoj varijanti svaki proces se zabranjuje ograničenjem $\text{gap}_{d,h} = 0$. U mekoj varijanti, promenljive $\text{gap}_{d,h}$ se vraćaju kao prekršaji i ulaze u funkciju cilja, pa rešavač minimizuje ukupan broj procepa:

Listing 4.14: Tvrda naspram meke varijante

```

1 if self.is_hard:
2     for gap in gap_vars:
3         solver.model.Add(gap == 0)    # zabrana svih procepa
4     return []
5 return gap_vars                        # meko: minimizuj broj
    procepa

```

4.6 Pravilo: kapacitet učionice kao meko ograničenje

Svaka učionica ima kapacitet, a svaka sesija poznat broj slušalaca size_s (zbir grupa koje je pohađaju, odeljak 5.3). Prirodno je zahtevati da svi slušaoci stanu u učionicu, $\text{size}_s \leq \text{capacity}_{r_s}$, i model je u ranijoj verziji to nametao kao tvrd uslov – učionice nedovoljnog kapaciteta prosto nisu ulazile u listu dozvoljenih učionica sesije (odeljak 2.4.2).

Takav tvrd uslov se, međutim, pokazao i previše krut i previše osetljiv na kvalitet ulaznih podataka:

- brojevi upisanih studenata su procene, a stvarna prisutnost je po pravilu manja od upisa; nerealno je da jedan student preko kapaciteta učini ceo raspored nedopustivim,
- u kombinaciji sa zahtevom za računarima kapacitet drastično sužava izbor učionica (računarskih učionica je malo i one su male), pa rešavač lako prijavi INFEASIBLE i ne vrati *nikakav* raspored – a raspored sa nekoliko prepunjenih učionica u praksi je daleko korisniji od nijednog,
- svako sužavanje dopustivog skupa gubi informaciju: umesto poruke „nema rešenja“ korisnije je videti *gde* i *koliko* raspored odstupa od željenog.

Zbog toga kapacitet prelazi iz tvrdih ograničenja u pravilo `rule_room_capacity.py` (`roomCapacity`). U mekoj varijanti pravilo ne sužava dopustiv skup – sesija sme u bilo koju učionicu koja zadovoljava tvrda ograničenja – ali plaća cenu jednaku broju nedostajućih mesta u toj učionici:

$$\text{cost}_s = \max(0, \text{size}_s - \text{capacity}_{r_s}) \quad (4.10)$$

Za razliku od procepa, prekršaj ovde nije „jedan događaj“ nego *količina* – pravilo je prvo koje kao prekršaj vraća celobrojne, a ne binarne promenljive. Funkcija cilja (4.1) je težinska suma vraćenih promenljivih, pa se takva cena uklapa bez izmena u jezgru rešavača. Cena je 1 po nedostajućem mestu, ali se sabira po mestima, pa dovoljno veliko prekoračenje može prevagnuti nad ostalim komponentama cilja.

U implementaciji se za svaku sesiju gradi tabela prekoračenja po učionicama (konstante, jer su i veličina sesije i kapaciteti poznati unapred), a promenljiva prekoračenja se vezuje za izabranu učionicu preko `AddElement` (odjeljak 4.2). Sesije koje staju u svaku učionicu ne dobijaju nikakvu promenljivu:

Listing 4.15: Prekoračenje kapaciteta po sesiji

```

1 overflows = [max(0, session.size - cap) for cap in capacities]
2 if not any(overflows):
3     continue                                # sesija staje u svaku
4     ucionicu
```

```
5 if self.is_hard:                                # tvrdo: samo ucionice u koje
    staje
6     fitting = [i for i, over in enumerate(overflows) if over ==
    0]
7     solver.model.AddAllowedAssignments(
8         [solver.room_var[s]], [[i] for i in fitting])
9     continue
10
11 overflow = solver.model.NewIntVar(0, max(overflows), f"
    room_overflow_{s}")
12 solver.model.AddElement(solver.room_var[s], overflows, overflow)
13 overflow_vars.append(overflow)
```

Glava 5

Raspoređivanje nastavnog osoblja

Do sada je model raspoređivao samo grupe studenata u učionice i termine. U ovoj glavi model proširujemo nastavnim osobljem: svakoj sesiji se dodeljuje nastavnik, uvodi se pojam *kohorte* – skupa grupa koje zajedno slušaju isti čas – i tek potom, nad tako proširenim modelom, uvode se ograničenja vezana za nastavnike.

Nastavno osoblje se zadaje u *odvojenom* ulaznom fajlu, tako da osnovni ulaz (predmeti, učionice, broj upisanih) ostaje nepromenjen. Taj fajl sadrži podatke o nastavnicima, koje predmete drže i kojim grupama. Recimo, profesor Predrag Jančić drži čas iz predmeta “Programske paradigme” za grupe 1i1a, 1i1b i 1i1v. Fajl je unapred zadat, i dostavljen je kandidatu na uvid prilikom pisanja ovog rada.

5.1 Model osoblja i dodela nastave

Podaci o osoblju modelovani su klasom `StaffInput` (modul `model.py`) koja sadrži tri celine: listu nastavnika (`teachers`), listu dodela nastave (`assignments`) i konfiguraciju pravila (`rules`). Nastavnik (`Teacher`) ima identifikator, ime i opciono lično ograničenje najvećeg broja radnih dana:

Listing 5.1: Modeli nastavnika i dodele nastave

```
1 @dataclass(config=_config)
2 class Teacher:
3     id: int
4     name: str
5     max_working_days: Optional[int] = Field(default=None, alias=
6         "maxWorkingDays")
```

```
7 @dataclass(config=_config)
8 class TeachingAssignment:
9     teacher_id: int = Field(alias="teacherId")
10    course_id: int = Field(alias="courseId")
11    session_type: str = Field(alias="sessionType")    # "theory"
        | "practice"
12    group_ids: Optional[List[str]] = Field(default=None, alias="
        groupIds")
```

Dodela (`TeachingAssignment`) povezuje par (predmet, tip nastave) sa nastavnikom, i opciono navodi konkretne grupe. Prilikom generisanja sesija, svaka sesija dobija polje `teacher_id`. Sesije bez dodeljenog nastavnika (`teacher_id` jednako `None`) ne učestvuju u pravilima osoblja.

5.2 Grupe studenata izvedene iz dodela nastave

Dodela nastave imenuje grupe kojima nastavnik drži čas, na primer „groupIds“: [„1i1a“, „1i1b“, „1i1v“]. Time je u fajlu sa osobljem već zapisano *koje* grupe postoje, pa bi njihovo ponovno navođenje u osnovnom ulazu bilo redundantno.

Postupak je implementiran u funkciji `build_groups` (modul `data.py`) i odvija se u dva koraka.

Imena i pripadnost. Prolazi se kroz sve dodele koje navode `groupIds`; za svaki naziv grupe smer i semestar se *ne* zadaju posebno, nego se čitaju sa predmeta na koji se dodela odnosi (`courseId`). Grupa 1i1a tako pripada smeru i semestru predmeta koji sluša. Ako se isti naziv grupe pojavi na predmetima iz dva različita para (smer, semestar), to je nekonzistentan ulaz i prijavljuje se kao greška:

Listing 5.2: Imena grupa iz dodela nastave

```
1 for a in staff_input.assignments:
2     if not a.group_ids:
3         continue
4     course = courses_by_id.get(a.course_id)
5     key = (course.track_id, course.semester)
6     for gid in a.group_ids:
7         existing = meta.get(gid)
8         if existing is not None and existing != key:
```

```
9         raise ValueError(f"Group '{gid}' is used for both {  
10             existing} and {key}")  
    meta[gid] = key
```

Veličine. Broj studenata se i dalje uzima iz sekcije `studentsEnrolled` osnovnog ulaza, koja je zadana po paru (smer, semestar). Upis se ravnomerno deli na sve grupe tog para, a ostatak se raspoređuje na prve grupe po abecednom redu naziva, tako da se veličine razlikuju najviše za jednog studenta:

Listing 5.3: Deljenje upisanih studenata na imenovane grupe

```
1 base, rem = divmod(enrollment.count, len(group_ids))  
2 for i, gid in enumerate(sorted(group_ids)):  
3     count = base + (1 if i < rem else 0)  
4     groups.append(Group(gid, enrollment.track_id, count,  
        enrollment.semester))
```

Na primer, 100 upisanih studenata prvog semestra Informatike i šest grupa navedenih u dodelama (1i1a, 1i1b, 1i1v, 1i2a, 1i2b, 1i2v) daju četiri grupe od 17 i dve od 16 studenata.

Fajl sa osobljem određuje *koje* grupe postoje i *ko* im drži nastavu, a osnovni ulaz *koliko* ih je i *šta* slušaju. Ako neki par (smer, semestar) nema ni jednu imenovanu grupu u dodelama, za njega se koristi rezervni postupak deljenja po veličini grupe iz odeljka 3.2.3, pa delimično popunjen fajl sa osobljem ne obara raspoređivanje ostatka fakulteta.

5.3 Kohorte i zajedničke sesije

Bez dodatne pažnje, ako dve grupe slušaju isto predavanje, model bi napravio dve odvojene sesije i time primorao nastavnika da isti čas drži dva puta. Da bismo to izbegli, uvodimo pojam *kohorte* (Cohort) – skupa grupa koje zajedno pohađaju sesije jednog kursa i tipa, sa (opcionim) zajedničkim nastavnikom. Veličina kohorte je zbir broja studenata svih njenih grupa, što je bitno za kapacitet učionice.

Kohorte se formiraju funkcijom `build_cohorts` (modul `data.py`) na osnovu dodela nastave, po sledećim pravilima:

- **Zajednička (joint) dodela** – ako dodela navodi više grupa u `groupIds`, te grupe čine *jednu* kohortu i dobijaju *jednu zajedničku sesiju*. Tako nastavnik

drži čas *jednom*, a prisustvuju mu sve grupe (npr. oba toka informatike slušaju zajedno predavanje iz predmeta “Programske paradigme”).

- **Dodela za jednu grupu** – `groupIds` sa jednom grupom važi samo za tu grupu.
- **Generička dodela** – ako `groupIds` nije naveden, dodela važi za svaku *preostalu* grupu kursa pojedinačno (svaka grupa dobija svoju sesiju sa tim nastavnikom).
- Grupe bez ijedne dodele postaju pojedinačne kohorte bez nastavnika.

Listing 5.4: Formiranje kohorti iz dodela nastave

```
1 for a in assignments:
2     if a.course_id != course.id or a.session_type !=
        session_type:
3         continue
4     if a.group_ids is None:
5         generic_teacher = a.teacher_id          # primenjuje se
        na sve preostale grupe
6         continue
7     cohorts.append(Cohort([groups_by_id[g] for g in a.group_ids
        ], a.teacher_id))
8     for g in a.group_ids:
9         remaining.pop(g, None)                  # ove grupe su
        pokrивene
10
11 for g in remaining.values():                    # grupe bez
        eksplicitne dodele
12     cohorts.append(Cohort([g], generic_teacher))
```

Postupak pretpostavlja ispravan ulaz i ne proverava ga: ako se ista grupa pojavi u dve eksplicitne dodele za isti predmet i tip nastave, dobiće dve kohorte i time dvostruku kvotu časova, a ako postoji više generičkih dodela, primenjuje se poslednja pročitana. Takve slučajeve treba ukloniti pre pokretanja.

Za svaku kohortu i svaki traženi čas iz kvote kreira se tačno jedna sesija, čija je lista grupa `group_ids` jednaka grupama kohorte. Zajednička sesija tako ulazi u ograničenje o koliziji *svake* grupe koja je pohada (tvrdo ograničenje HC2 prolazi

kroz sve `session.group_ids`). Veličina kohorte ne ograničava izbor učionice tvrdo, ali kroz meko pravilo iz odeljka 4.6 usmerava sesiju ka dovoljno velikoj sali.

Primer. Dve grupe `ga` (30 studenata) i `gb` (25 studenata) slušaju predavanja zajedno, a vežbe odvojeno. Za predmet sa 2 predavanja i 2 vežbe nedeljno dobija se: 2 zajedničke teorijske sesije (kohorta `ga+gb`, veličine 55) i po 2 vežbe za svaku grupu – ukupno 6 sesija umesto 8. Zajedničkom predavanju je za 55 slušalaca potrebna sala kapaciteta najmanje 55, dok vežbama od 30, odnosno 25 studenata odgovara i znatno manja učionica.

5.4 Pravila nastavnog osoblja

Sada kada su uvedeni nastavnici, dodele nastave, grupe i kohorte, model proširujemo ograničenjima nad rasporedom nastavnika. Osim prvog, osnovnog ograničenja koje je ugrađeno u jezgro rešavača, pravila osoblja koriste istu *plugin* arhitekturu i CP-SAT tehnike (reifikacija, `AddMaxEquality`, `AddBoolAnd/AddBoolOr`, `AddElement`) uvedene u glavi 4. Konfiguriraju se u `rules` sekciji `staff` fajla i mogu biti tvrda (`penalty 0`) ili meka (`penalty > 0`).

5.4.1 Nastavnik ne može držati dva časa istovremeno

Prvo i osnovno ograničenje osoblja je tvrdo i dodaje se u jezgru rešavača: nijedan nastavnik ne može držati dve sesije u istom terminu. Analogno ograničenju za grupe (HC3, listing 3.5), koristimo `AllDifferent` nad apsolutnim vremenima svih sesija jednog nastavnika:

$$\text{AllDifferent}(\{\text{flatTime}_s : s \in \mathcal{S}_t\}), \quad \forall t \quad (5.1)$$

gde je $\mathcal{S}_t$ skup sesija nastavnika t . U kodu (modul `cp_solver.py`):

Listing 5.5: Nastavnik nije dvostruko zauzet

```
1 for teacher_id, session_indices in self.teacher_sessions.items():
2     if len(session_indices) > 1:
3         self.model.AddAllDifferent(
4             [self.flat_time_var[s] for s in session_indices]
5         )
```

5.4.2 Jedna lokacija po danu za nastavnika

Pravilo `rule_staff_single_location_in_day.py` (`staffSingleLocationInDay`) sprečava da nastavnik menja zgradu tokom jednog dana – fizički bi bilo neizvodljivo držati dva uzastopna časa na različitim lokacijama. Mehanizam je isti kao za grupe: za svakog nastavnika i dan uvodi se promenljiva lokacije, a sve njegove sesije tog dana vezuju se za nju pomoću `AddElement`:

Listing 5.6: Nastavnik na jednoj lokaciji dnevno

```

1 day_locs = [model.NewIntVar(0, max_loc, f"teacher_day_loc_{
    teacher_id}_{d}")
2             for d in range(D)]
3 for s in session_indices:
4     # loc_var[s] == day_locs[day_var[s]]
5     solver.model.AddElement(solver.day_var[s], day_locs, solver.
        loc_var[s])

```

Ovo je tvrdo ograničenje i vraća praznu listu prekršaja.

5.4.3 Najveći broj radnih dana nastavnika

Pravilo `rule_staff_max_working_days.py` (`staffMaxWorkingDays`) ograničava koliko različitih dana u nedelji nastavnik dolazi na fakultet. Za svaki dan d uvodi se indikator $\text{dayUsed}_{t,d}$ (logičko ILI reifikovanih jednakosti $\text{day}_s = d$ preko sesija nastavnika), a broj radnih dana je njihov zbir:

$$\text{worked}_t = \sum_{d=0}^{D-1} \text{dayUsed}_{t,d} \quad (5.2)$$

Tvrda varijanta nameće $\text{worked}_t \leq \text{limit}_t$, gde je limit_t lični limit nastavnika (`maxWorkingDays`) ili podrazumevana vrednost pravila (`maxDays`). Meka varijanta uvodi celobrojnu promenljivu viška i minimizuje ga:

$$\min \sum_t \text{penalty} \cdot \max(0, \text{worked}_t - \text{limit}_t) \quad (5.3)$$

Listing 5.7: Ograničenje broja radnih dana

```

1 worked_days = sum(day_used) # day_used[d] = OR
    preko sesija
2 if self.is_hard:

```

```

3     solver.model.Add(worked_days <= limit)
4 else:
5     excess = solver.model.NewIntVar(0, D, f"staff_days_excess_{
        teacher_id}")
6     solver.model.AddMaxEquality(excess, [worked_days - limit,
        0])
7     violations.append(excess)                # ulazi u funkciju
        cilja

```

5.4.4 Nedeljni budžet procepa (broj sati pauze) nastavnika

Pravilo `rule_staff_no_gap_greater_than.py` (`staffMaxGapHoursPerWeek`) ograničava ukupan broj procepa (pauza) nastavnika *tokom cele nedelje*. Procep se detektuje isto kao u pravilu Pravilo: bez procepa u rasporedu: matrica zauzetosti se gradi reifikovanim jednakostima i `AddMaxEquality`, a procep je konjunkcija “ima čas pre”, “ima čas posle” i “slobodan sat”. Za razliku od pravila za grupe, ovde se procepi *sabiraju kroz sve dane*:

$$\text{weekGaps}_t = \sum_{d=0}^{D-1} \sum_h \text{gap}_{t,d,h} \quad (5.4)$$

Tvrda varijanta nameće $\text{weekGaps}_t \leq B$ (budžet $B = \text{maxGapHours}$, podrazumevano 1). Meka varijanta minimizuje prekoračenje budžeta:

$$\min \sum_t \text{penalty} \cdot \max(0, \text{weekGaps}_t - B) \quad (5.5)$$

Listing 5.8: Nedeljni budžet procepa

```

1 total_gaps = sum(week_gaps)
2 if self.is_hard:
3     solver.model.Add(total_gaps <= self.max_gap_hours)
4 else:
5     excess = solver.model.NewIntVar(0, D * H, f"
        staff_gap_excess_{teacher_id}")
6     solver.model.AddMaxEquality(excess, [total_gaps - self.
        max_gap_hours, 0])
7     violations.append(excess)

```

U oba pravila prekoračenje se vezuje *jednakošću* $e = \max(0, w - L)$, a ne nejednakostima $e \geq 0$ i $e \geq w - L$. Same nejednakosti ne bi odredile vrednost promenljive: one dopuštaju i e veće od stvarnog prekoračenja, a na tačnu vrednost je u optimumu svodi tek pozitivna težina u funkciji cilja, pod uslovom da se e nigde drugde ne koristi. Takav zapis je za samu optimizaciju ispravan, ali vrednost pomoćne promenljive tada ne bi smela da se čita kao stvarno prekoračenje u međurešenjima koja rešavač prijavljuje tokom pretrage. Pošto te vrednosti očitavamo po komponentama (odeljak 6.2.4), jednakost namećemo izričito.

Svi prekršaji mekih pravila osoblja ulaze u istu globalnu funkciju cilja iz glave 4: $\min \sum_r \text{penalty}_r \cdot \sum_{v \in V_r} v$.

5.5 Konfiguracija i provera

Osoblje se zadaje u zasebnom JSON fajlu (na primer `staff_2_semester.json`). Sekcija `rules` bira koja pravila su aktivna i sa kojom cenom, `teachers` navodi nastavnike, a `assignments` povezuje predmete sa nastavnicima i istovremeno definiše grupe studenata (odeljak 5.2):

Listing 5.9: Isečak staff ulaznog fajla

```
1 {
2   "rules": {
3     "staffMaxWorkingDays": {"enabled": true, "penalty": 3, "
4       params": {"maxDays": 4}},
5     "staffMaxGapHoursPerWeek": {"enabled": true, "penalty": 0, "
6       params": {"maxGapHours": 1}},
7     "staffSingleLocationInDay": {"enabled": true, "penalty": 0}
8   },
9   "teachers": [
10     {"id": 4, "name": "milan.mitreski", "maxWorkingDays": 2}
11   ],
12   "assignments": [
13     {"teacherId": 43, "courseId": 54, "sessionType": "theory", "
14       groupIds": ["21a", "21b"]}
```

Rešavač se pokreće sa dodatnim argumentom `--staff:`

```
bazel run //src/algo:run_cp_solver -- \  
  --input input_full_2_semester.json \  
  --staff staff_2_semester.json --time-limit 600
```

Ispravnost pravila osoblja proverava se testovima u `test_staff_rules.py`: da nastavnik nije dvostruko zauzet, da se poštuje najveći broj radnih dana (uključujući lične limite), da je nedeljni budžet procepa ispunjen, da nastavnik ne menja lokaciju u toku dana, i da se zajednička sesija dve grupe rešava kao jedan čas bez kolizija u rasporedu obe grupe. Izvođenje grupa iz dodela pokriveno je testovima u `test_data.py`.

Glava 6

Evaluacija

Evaluacija je sprovedena u dva koraka, koji odgovaraju dvema fazama razvoja modela. Preliminarna evaluacija (odeljak 6.1) poredi CP i MIP formulaciju osnovnog modela, u režimu traženja bilo kog dopustivog rasporeda, i služi izboru formulacije koja se dalje razvija. Završna evaluacija (odeljak 6.2) meri prošireni CP model, sa mekim ograničenjima, funkcijom cilja i nastavnim osobljem, uz praćenje vrednosti funkcije cilja kroz vreme.

Obe evaluacije koriste isti ulazni fajl i parne semestre studija, pa se odnose na istu nastavu; razlikuju se po tome koliko je pravila u modelu i kako se grupe studenata formiraju.

6.1 Preliminarna evaluacija osnovnog modela

6.1.1 Metodologija

6.1.1.1 Podskupovi realnih podataka

Eksperimenti su sprovedeni na tri podskupa realnih podataka letnjeg semestra (`input_full_2_semester.json`), dobijena filtriranjem po godinama studija:

1. **MATF-S** (mali) – 1. godina (semestar 2): 30 predmeta, 8 grupa i 204 sesije.
2. **MATF-M** (srednji) – 1. i 2. godina (semestri 2 i 4): 63 predmeta, 16 grupa i 398 sesija.
3. **MATF-L** (veliki) – sve četiri godine (semestri 2, 4, 6 i 8): 134 predmeta, 30 grupa i 722 sesije. Ovo je pun obim nastave jednog semestra.

Skalira se, dakle, samo potražnja: sve tri instance zadržavaju svih 29 učionica na tri lokacije, pa se razlikuju jedino po količini nastave koju treba smestiti.

Ovde se grupe studenata dobijaju deljenjem broja upisanih na grupe od najviše 50 studenata, jer osnovni model ne poznaje nastavno osoblje, dok se u odeljku 6.2 grupe izvede iz imena navedenih u dodelama nastave (odeljak 5.2), koja opisuju stvarnu podelu na fakultetu. Na najvećoj skali to je razlika između 30 i 44 grupe, pa ista instanca tamo ima 835 umesto 722 sesije. Sam postupak raspoređivanja se ne menja – menja se samo koliko časova ulazi u model – ali apsolutne vrednosti iz ovog odeljka i iz odeljka 6.2 zato nisu neposredno uporedive.

6.1.1.2 Metrike

Za svaki par (rešavač, skala) merimo sledeće metrike:

- **Broj sesija** – ukupan broj časova koje treba rasporediti.
- **Broj promenljivih** – veličina modela u terminima odlučivačkih promenljivih.
- **Broj ograničenja** – ukupan broj ograničenja u modelu.
- **Vreme konstrukcije** – vreme potrebno za kreiranje modela (promenljivih i ograničenja), mereno pomoću `time.perf_counter()`.
- **Vreme rešavanja** – vreme potrebno za pronalaženje dopustivog rešenja.
- **Ukupno vreme** – zbir vremena konstrukcije i rešavanja.
- **Memorija modela** – memorijski otisak modela, meren pomoću `tracemalloc` (razlika memorijskih snimaka pre i posle konstrukcije modela).
- **Status** – da li je rešavač našao dopustivo rešenje ili je isteklo vreme.
- **Validnost rešenja** – nezavisna provera da li dobijeno rešenje zadovoljava sva tvrda ograničenja (bez kolizija učionica, bez kolizija grupa, poštovanje računarskih zahteva).

Napomena vezana za merenje memorije. `tracemalloc` prati isključivo alokacije na strani Pythona, dakle objekte kojima se model opisuje pre nego što se preda rešavaču, a ne interne strukture rešavača, koje su u oba slučaja u prevedenom kodu biblioteke. Zbog toga prikazujemo tu meru kao pokazatelj veličine zapisa modela, a ne kao ukupnu potrošnju memorije pri rešavanju. Vršna potrošnja celog procesa,

koju daje `resource.getusage()`, ne razdvaja pokretanja: ona je maksimum kroz ceo život procesa, pa u jednom pokretanju obuhvata i sve prethodno izmerene skale i oba rešavača. Zato je ne navodimo po instanci.

6.1.1.3 Validacija rešenja

Rezultati svakog rešavača se nezavisno verifikuju funkcijom `validate_solution` koja proverava:

1. da nijedna dva časa ne dele istu trojku (dan, sat, učionica),
2. da nijedna dva časa iste grupe nisu u istom terminu (dan, sat),
3. da časovi koji zahtevaju računare budu u učionicama sa računarima (provera se vrši nad zahtevom sesije, koji zavisi od tipa nastave),
4. da broj dodeljenih termina odgovara broju sesija.

6.1.2 Rezultati

U tabelama 6.1, 6.2 i 6.3 prikazani su rezultati za svaku skalu.

Tabela 6.1: Rezultati za MATF-S (1. godina)

Metrika	CP-SAT	MIP/SCIP
Sesije	204	
Promenljive	1 224	319 680
Ograničenja	649	2 424
Vreme konstrukcije	0,016 s	5,121 s
Vreme rešavanja	0,026 s	3,206 s
Ukupno vreme	0,042 s	8,327 s
Memorija modela	0,22 MB	71,54 MB
Status	FEASIBLE	FEASIBLE
Rešenje validno	da	da

6.1.3 Analiza rezultata

Zaključci koji slede odnose se na dve konkretne formulacije iz odeljka 2.4, na rešavače CP-SAT i SCIP u podrazumevanim podešavanjima, i na tri opisane instance. O paradigmama uopšte ne tvrdimo ništa: drugačija formulacija, drugačije podešen rešavač ili instanca druge strukture mogli bi dati drugačiji odnos.

Tabela 6.2: Rezultati za MATF-M (1–2. godina)

Metrika	CP-SAT	MIP/SCIP
Sesije		398
Promenljive	2 388	619 440
Ograničenja	1 269	3 098
Vreme konstrukcije	0,030 s	9,879 s
Vreme rešavanja	0,084 s	8,170 s
Ukupno vreme	0,114 s	18,049 s
Memorija modela	0,39 MB	138,56 MB
Status	FEASIBLE	FEASIBLE
Rešenje validno	da	da

Tabela 6.3: Rezultati za MATF-L (sve četiri godine)

Metrika	CP-SAT	MIP/SCIP
Sesije		722
Promenljive	4 332	1 102 560
Ograničenja	2 319	4 262
Vreme konstrukcije	0,055 s	18,600 s
Vreme rešavanja	1,598 s	17,804 s
Ukupno vreme	1,652 s	36,404 s
Memorija modela	0,70 MB	246,57 MB
Status	FEASIBLE	FEASIBLE
Rešenje validno	da	da

6.1.3.1 Veličina zapisa modela

CP-SAT formulacija ima $6|\mathcal{S}|$ celobrojnih promenljivih, dok MIP/SCIP formulacija ima $DH \sum_s |\text{Elig}(s)|$ binarnih. Sa $D = 5$ dana, $H = 12$ sati i $R = 29$ učionica svaka sesija koja ne traži računare u MIP-u daje $5 \cdot 12 \cdot 29 = 1740$ binarnih promenljivih, naspram šest celobrojnih u CP-u, pa je razlika u broju promenljivih dva do tri reda veličine. Tako izražena razlika nije mera kvaliteta formulacije – dve paradigme problem zapisuju na različite načine, pa brojevi promenljivih nisu neposredno uporedivi – ali se vidi u dvema izmerenim veličinama: u memoriji koju zauzima zapis modela na strani Pythona i u vremenu konstrukcije modela, koje je kod MIP-a od 5,1 do 18,6 sekundi, naspram 0,016 do 0,055 sekundi kod CP-a.

6.1.3.2 Vreme do dopustivog rešenja

Na sve tri instance oba rešavača nalaze validan raspored. CP-SAT do njega stiže brže: ukupno vreme je 0,042 s naspram 8,3 s na MATF-S, 0,114 s naspram 18,0 s na MATF-M i 1,65 s naspram 36,4 s na MATF-L. Deo te razlike otpada na konstrukciju modela, a deo na samu pretragu; kod najveće instance MIP troši 18,6 s na konstrukciju i 17,8 s na rešavanje, a CP 0,055 s i 1,60 s.

Objašnjenje koje se nameće jeste da CP-SAT radi sa globalnim ograničenjima, čiji specijalizovani algoritmi propagacije rano eliminišu nedopustive vrednosti iz domena, uz učenje konflikata preuzeto iz SAT rešavača, dok SCIP nad istim problemom rešava niz LP relaksacija u okviru grananja i odsecanja. Treba, međutim, imati u vidu i da je zadatak postavljen kao problem dopustivosti, bez funkcije cilja, pa mehanizam ograničavanja pomoću LP relaksacije, koji je snaga MIP pristupa, ovde nema šta da ograničava (odjeljak 6.1.3.4).

6.1.3.3 Ponašanje pri rastu instance

Odnos ukupnih vremena nije stalan i ne raste sa veličinom instance: MIP/SCIP je sporiji 198 puta na MATF-S, 158 puta na MATF-M i 22 puta na MATF-L. Relativna prednost CP-a, dakle, opada kako instanca raste, dok apsolutna razlika raste – sa 8,3 sekunde na MATF-S na 34,8 sekundi na MATF-L. Pošto sve tri instance koriste isti skup od 29 učionica, razlika između skala potiče isključivo od porasta broja sesija.

6.1.3.4 Ograničenje pristupa: MIP model bez funkcije cilja

Oba modela ovog odeljka postavljena su kao problemi dopustivosti, pa MIP model radi bez funkcije cilja. Time ostaje neiskorišćen mehanizam po kome je MIP pristup poznat – donja granica iz LP relaksacije, koja u optimizaciji vodi pretragu i omogućava odsecanje grana. Poređenje zato treba čitati kao poređenje dva načina da se *nađe dopustiv raspored*, a ne kao poređenje optimizacionih sposobnosti dva rešavača.

Meka ograničenja uvedena u glavi 4 mogla bi se zapisati i linearno i uvrstiti u MIP funkciju cilja; to bi omogućilo poređenje dva rešavača u optimizacionom režimu, sa donjim granicama i procepom optimalnosti sa obe strane. U ovom radu to nije urađeno i navedeno je kao pravac daljeg rada (glava 7).

Rezultati ovog odeljka odnose se na osnovni model i služe izboru formulacije za dalji razvoj. Pošto se model u glavama 4 i 5 znatno proširuje, u odeljku 6.2 dajemo završnu evaluaciju konačne verzije rešavača na istim trima skalama.

6.2 Završna evaluacija proširenog modela

Za razliku od prve evaluacije, ovde nas ne zanima samo da li rešenje postoji, nego i *kvalitet* rešenja i brzina kojom se on popravljja, pa pratimo vrednost funkcije cilja u zavisnosti od proteklog vremena izračunavanja.

Merenja su sprovedena u dva okruženja. Prvo je realno okruženje, sa podacima onakvim kakvi jesu, i ono odgovara na pitanje da li je prošireni model upotrebljiv u praksi. Drugo je zaoštreno okruženje (odeljak 6.2.4), u kome je broj raspoloživih termina smanjen tako da meka ograničenja dođu u stvarni sukob, i ono pokazuje kako se rešavač ponaša na granici rešivosti.

6.2.1 Metodologija

6.2.1.1 Instance po godinama studija

Prošireni model radi nad realnim podacima letnjeg semestra (`input_full_2_semester.json`) i dodelama nastave iz `staff_2_semester.json`, iz kojih se izvode grupe studenata (odeljak 5.2). Svaka grupa nasleđuje semestar predmeta koji sluša (2, 4, 6 ili 8), što odgovara godini studija:

1. **FINAL-S** – 1. godina (semestar 2),
2. **FINAL-M** – 1. i 2. godina (semestri 2 i 4),
3. **FINAL-L** – sve četiri godine (semestri 2, 4, 6 i 8).

Skalira se samo potražnja: sve tri instance zadržavaju svih 29 učionica i sve tri lokacije, pa se razlikuju jedino po količini nastave koju treba smestiti. Filtriraju se samo predmeti i upisi; grupe se povlače same, jer se izvode iz dodela za predmete koji su ostali, a dodele za izbačene predmete se prosto ne aktiviraju – ulaz sa osobljem tako ostaje nepromenjen.

Ovako definisane instance poklapaju se sa podskupovima iz odeljka 6.1: isti je ulazni fajl, iste godine studija i isti skup učionica, tako da se rezultati pre i posle proširenja modela odnose na istu nastavu. U konfiguraciji bez osoblja skup sesija je

doslovno isti (204, 398 i 722 sesije po skalama). Kada se uključi osoblje, grupe se izvode iz dodela nastave umesto deljenjem upisa na grupe od najviše 50 studenata, pa se broj sesija menja – raste broj imenovanih grupa, a predavanja koja više grupa sluša zajedno spajaju se u jednu sesiju.

6.2.1.2 Konfiguracije i funkcija cilja

Svaka skala se rešava u dve konfiguracije:

- **bez osoblja** – pravila iz glave 4: spajanje časova u blokove kao tvrdo ograničenje, izbegavanje procepa kao meko ograničenje sa cenom 2 i kapacitet učionica kao meko ograničenje sa cenom 1;
- **sa osobljem** – dodatno i ceo model iz glave 5: kohorte i zajedničke sesije, tvrdo ograničenje da nastavnik nije dvostruko zauzet, tvrda pravila za jednu lokaciju po danu i nedeljni budžet procepa nastavnika, i meko pravilo najvećeg broja radnih dana sa cenom 3.

Funkcija cilja je, u skladu sa izrazom (4.1) iz odeljka 4.1, težinska suma prekršaja mekih pravila:

$$\min \underbrace{2 \cdot \sum_{g,d,h} \text{gap}_{g,d,h}}_{\text{procepi grupa}} + 3 \cdot \underbrace{\sum_t \text{excess}_t}_{\text{dani preko limita nastavnika}} + 1 \cdot \underbrace{\sum_s \text{overflow}_s}_{\text{nedostajuća mesta}},$$

pri čemu u konfiguraciji bez osoblja drugi sabirak ne postoji. Treći sabirak je meko ograničenje kapaciteta iz odeljka 4.6 i u obe konfiguracije nosi najmanju težinu po jedinici prekršaja, mada se sabira po mestima, pa veliko prekoračenje može prevagnuti. Vrednost 0 znači da nijedno meko ograničenje nije prekršeno, dakle raspored bez ijednog procepa, bez ijednog nastavnika koji radi više dana nego što mu je dozvoljeno i bez ijednog nedostajućeg mesta u učionici. Težine 2, 3 i 1 su, kao što je rečeno u odeljku 4.1, heuristički izabrane, pa su i rezultati ovog odeljka vezani za taj izbor.

6.2.1.3 Uslovi merenja

Merenja su izvršena na računaru sa procesorom Apple M4 (10 jezgara) i 24 GB radne memorije, pod operativnim sistemom macOS 26.5, sa bibliotekom OR-Tools 9.15. Rešavač koristi 8 niti pretrage i vremenski limit od 600 sekundi po pokretanju.

Vrednost funkcije cilja kroz vreme prati se povratnim pozivom (*callback*) koji CP-SAT poziva pri svakom pronađenom poboljšanju. Za svako poboljšanje beleže se proteklo vreme pretrage, vrednost cilja i najbolja donja granica, iz čega se dobija stepenasta funkcija: vrednost u trenutku t je cilj poslednjeg rešenja nađenog do tog trenutka. Rešenja se, kao i u prvoj evaluaciji, nezavisno proveravaju funkcijom `validate_solution`, koja je za potrebe proširenog modela dopunjena i proverom da nastavnik nema dva časa u istom terminu.

6.2.2 Veličina modela

Tabela 6.4 prikazuje veličinu modela po skalama i konfiguracijama.

Tabela 6.4: Veličina proširenog CP modela po skalama

Metrika	FINAL-S		FINAL-M		FINAL-L	
	bez	sa	bez	sa	bez	sa
Sesije	204	216	398	437	722	835
Nastavnici	0	39	0	81	0	112
Promenljive	15 348	41 036	30 026	87 697	54 652	144 159
Ograničenja	27 857	72 082	54 424	154 777	98 974	256 165
Vreme konstrukcije	0,08 s	0,24 s	0,17 s	0,48 s	0,31 s	0,83 s

Kolone označene sa „bez“ odnose se na konfiguraciju bez osoblja, a kolone „sa“ na konfiguraciju sa raspoređivanjem nastavnog osoblja.

Tabela 6.5 sumira tok rešavanja. Vremena u njoj imaju četiri različita značenja i u nastavku ih dosledno razlikujemo:

- *prvo dopustivo rešenje* – trenutak kada je rešavač prvi put prijavio raspored koji zadovoljava sva tvrda ograničenja;
- *vreme do najboljeg* – trenutak poslednjeg poboljšanja funkcije cilja;
- *dokaz optimalnosti* – kraj pretrage, kada je donja granica dostigla vrednost najboljeg rešenja (kod statusa OPTIMAL to je isti trenutak kad i kraj pokretanja);
- *ukupno vreme* – konstrukcija modela plus rešavanje, dakle ono što korisnik meri štopericom.

Red „broj nađenih rešenja“ broji sva rešenja koja je rešavač prijavio, uključujući i prvo dopustivo, pa je broj strogih poboljšanja za jedan manji.

Tabela 6.5: Tok rešavanja proširenog modela

Metrika	FINAL-S		FINAL-M		FINAL-L	
	bez	sa	bez	sa	bez	sa
Prvo dopustivo rešenje	1,62 s	2,67 s	4,90 s	8,76 s	23,17 s	27,46 s
Cilj prvog rešenja	1 355	696	1 009	1 444	2 440	3 095
Broj nađenih rešenja	22	19	16	32	43	73
Vreme do najboljeg	2,38 s	4,17 s	7,26 s	15,20 s	40,43 s	76,66 s
Dokaz optimalnosti	2,40 s	4,20 s	7,28 s	15,26 s	40,48 s	76,78 s
Ukupno vreme	2,49 s	4,44 s	7,45 s	15,75 s	40,80 s	77,61 s
Najbolji cilj	0	0	0	0	0	0
Donja granica	0	0	0	0	0	0
Status	OPTIMAL					
Rešenje validno	da					

6.2.3 Funkcija cilja u zavisnosti od vremena

Pošto se sva poboljšanja dešavaju u uskom intervalu odmah po nalaženju prvog rešenja, gruba mreža preseka ne pokazuje dinamiku pretrage. Zato u tabelama 6.6, 6.7 i 6.8 dajemo celu putanju poboljšanja za konfiguraciju sa osobljem, a za najveću skalu je ista putanja prikazana i krivom na slici 6.1. Konfiguracija bez osoblja prolazi kroz istu vrstu putanje (22, 16 i 43 prijavljena rešenja po skalama), samo je kraća: na dve manje skale ceo tok stane u prvih nekoliko sekundi, dok na najvećoj traje oko 40 sekundi.

Tabela 6.6: Putanja poboljšanja, FINAL-S sa osobljem

Vreme	Cilj	Vreme	Cilj	Vreme	Cilj
2,67 s	696	3,21 s	188	4,11 s	8
2,71 s	604	3,33 s	109	4,13 s	6
2,74 s	595	3,41 s	69	4,14 s	4
2,79 s	585	3,49 s	29	4,16 s	2
2,88 s	398	3,61 s	27	4,17 s	0
2,94 s	364	3,94 s	11		
3,04 s	358	4,09 s	10		

Napomena: prikazani rezultati dobijeni su komandom:

```
bazel run //src/algo:benchmark -- --mode final --max-time 600 \
--final-json report_final.json
```

Tabela 6.7: Putanja poboljšanja, FINAL-M sa osobljem

Vreme	Cilj	Vreme	Cilj	Vreme	Cilj
8,76 s	1 444	12,76 s	72	14,48 s	16
8,88 s	1 420	12,94 s	60	14,66 s	14
9,84 s	251	13,05 s	56	14,72 s	11
9,98 s	223	13,05 s	48	14,78 s	9
10,14 s	219	13,84 s	32	14,84 s	8
10,54 s	217	14,11 s	30	14,90 s	6
10,81 s	173	14,21 s	29	14,96 s	5
10,82 s	171	14,27 s	27	15,01 s	3
11,07 s	155	14,32 s	25	15,08 s	2
11,99 s	85	14,40 s	24	15,20 s	0
12,54 s	79	14,45 s	22		

6.2.4 Zaoštreno okruženje

Rezultati iz prethodnog odeljka pokazuju da rešavač na svim skalama dostigne cilj 0, i to za nekoliko desetina sekundi, najviše nešto više od minuta. Razlog je što realna instanca ima dovoljno slobodnog prostora: 29 učionica puta 12 jednosatnih termina dnevno puta 5 dana daje 1740 parova (učionica, termin) za najviše 835 časova, dakle popunjenost od 48%. U tako labavom rasporedu meka ograničenja retko dolaze u stvarni sukob, pa kriva funkcije cilja pouzdano padne na nulu i o granicama modela govori malo.

Da bismo videli kako se model ponaša kada resursi postanu oskudni, uvodimo *zaoštreno okruženje* – istu instancu sa dve izmene:

- nastava traje od 8 do 15 časova umesto do 20, dakle sedam jednosatnih termina dnevno umesto dvanaest, čime broj parova (učionica, termin) pada sa 1740 na 1015, a popunjenost najveće instance raste na 82%;
- podrazumevani najveći broj radnih dana nastavnika smanjen je sa 4 na 2 (lični limiti pojedinih nastavnika, koji su stroži, ostaju nepromenjeni).

Dužina dana je birana tako da problem ostane jedva rešiv. Skraćenje za još jedan čas (8–14, dakle šest termina dnevno, 870 parova i 96% popunjenosti) dovodi najveću instancu sa osobljem do statusa UNKNOWN: rešavač za 600 sekundi ne nađe *nijedno* dopustivo rešenje, ali ni ne dokaže da rešenje ne postoji.

Procenti popunjenosti su, treba naglasiti, gruba mera raspoloživosti: u imeniocu su svi parovi (učionica, termin), a mnogi od njih nisu upotrebljivi za datu sesiju,

Tabela 6.8: Putanja poboljšanja, FINAL-L sa osobljem

Vreme	Cilj	Vreme	Cilj	Vreme	Cilj
27,46 s	3 095	50,56 s	300	68,57 s	39
27,66 s	3 053	51,77 s	292	68,93 s	38
27,96 s	2 999	53,67 s	270	69,15 s	35
28,19 s	2 947	55,12 s	254	69,86 s	33
28,63 s	2 824	55,81 s	215	70,11 s	32
29,26 s	2 592	57,23 s	214	70,39 s	29
29,53 s	2 444	57,80 s	136	70,70 s	27
29,90 s	1 848	58,35 s	114	70,94 s	26
31,36 s	1 838	61,42 s	113	71,33 s	24
31,55 s	1 557	61,66 s	111	71,94 s	23
31,96 s	1 386	61,89 s	110	72,44 s	21
32,21 s	1 380	62,13 s	108	72,87 s	20
34,17 s	1 072	62,42 s	105	73,49 s	17
36,68 s	1 026	62,54 s	86	74,09 s	16
37,21 s	936	63,41 s	84	74,35 s	14
38,35 s	880	63,68 s	82	74,61 s	13
38,77 s	833	63,97 s	80	74,83 s	11
42,26 s	761	64,24 s	79	75,13 s	9
42,40 s	612	64,49 s	77	75,34 s	6
42,67 s	606	64,67 s	51	75,74 s	5
42,99 s	600	64,95 s	49	76,07 s	3
43,55 s	555	65,30 s	47	76,35 s	2
45,83 s	545	66,01 s	45	76,66 s	0
47,16 s	375	66,58 s	44		
49,95 s	367	67,20 s	42		

jer učionica nema računare, nije na odgovarajućoj lokaciji, ili je termin već zauzet grupom ili nastavnikom te sesije. Iz procenta popunjenosti ne sledi ni rešivost ni to da cilj može pasti na nulu; on služi samo da se dva okruženja uporede po količini prostora.

Tabela 6.9 prikazuje tok rešavanja u tako zaoštrenom okruženju.

Za razliku od realnog okruženja, ovde najbolji cilj *nije* nula: u konfiguraciji sa osobljem ostaje 3 na najmanjoj skali, a 6 na srednjoj i najvećoj. Pošto se donja granica poklapa sa nađenim rešenjem, reč je o dokazano najmanjoj mogućoj vrednosti cilja pri datim težinama.

Tabela 6.10 pokazuje od čega se ta vrednost sastoji. Komponente se očitavaju iz najboljeg nađenog rešenja, tako što se za svako meko pravilo saberu vrednosti

Tabela 6.9: Tok rešavanja u zaoštrenom okruženju (8–15 časova)

Metrika	FINAL-S		FINAL-M		FINAL-L	
	bez	sa	bez	sa	bez	sa
Promenljive	9 448	24 511	18 476	52 322	33 602	86 359
Ograničenja	16 657	42 782	32 524	91 827	59 124	152 265
Prvo dopustivo rešenje	1,14 s	1,67 s	3,80 s	6,63 s	37,29 s	50,61 s
Cilj prvog rešenja	744	1 050	1 837	1 676	2 545	2 821
Broj nadenih rešenja	16	13	26	31	19	57
Vreme do najboljeg	1,50 s	2,60 s	6,13 s	12,36 s	58,11 s	194,32 s
Dokaz optimalnosti	1,51 s	2,66 s	6,14 s	12,44 s	58,17 s	194,42 s
Ukupno vreme	1,57 s	2,82 s	6,27 s	12,77 s	58,39 s	194,94 s
Najbolji cilj	0	3	0	6	0	6
Donja granica	0	3	0	6	0	6
Status	OPTIMAL					
Rešenje validno	da					

njegovih promenljivih prekršaja.

Tabela 6.10: Komponente funkcije cilja u zaoštrenom okruženju

Komponenta (težina)	FINAL-S		FINAL-M		FINAL-L	
	bez	sa	bez	sa	bez	sa
Procepi u rasporedu grupa (2)	0	0	0	0	0	0
Nedostajuća mesta u učionicama (1)	0	0	0	0	0	0
Radni dani preko limita (3)	–	1	–	2	–	2
Ukupan cilj	0	3	0	6	0	6

Ceo preostali cilj potiče, dakle, od jednog jedinog pravila: u zbijenoj nedelji jedan nastavnik na najmanjoj skali, odnosno po dva nastavnika na druge dve, moraju da rade jedan dan više nego što im limit dozvoljava. Procepa u rasporedu grupa nema, niti ijedna sesija završava u premaloj učionici. U realnom okruženju su sve tri komponente nula, pa tu tabelu ne prikazujemo posebno.

Tabela 6.11 daje vrednost funkcije cilja u zadatim trenucima.

Zaoštreno okruženje se dobija dodavanjem opcije `--profiletight`:

```
bazel run //src/algo:benchmark -- --mode final --profile tight \
  --max-time 600 --final-json report_final_tight.json
```

Tabela 6.11: Funkcija cilja u vremenskim presecima, zaoštreno okruženje

Vreme	FINAL-S		FINAL-M		FINAL-L	
	bez	sa	bez	sa	bez	sa
10 s	0	3	0	90	–	–
20 s	0	3	0	6	–	–
30 s	0	3	0	6	–	–
40 s	0	3	0	6	1 817	–
50 s	0	3	0	6	165	–
60 s	0	3	0	6	0	1 431
90 s	0	3	0	6	0	103
120 s	0	3	0	6	0	41
300 s	0	3	0	6	0	6
600 s	0	3	0	6	0	6

6.2.5 Analiza rezultata

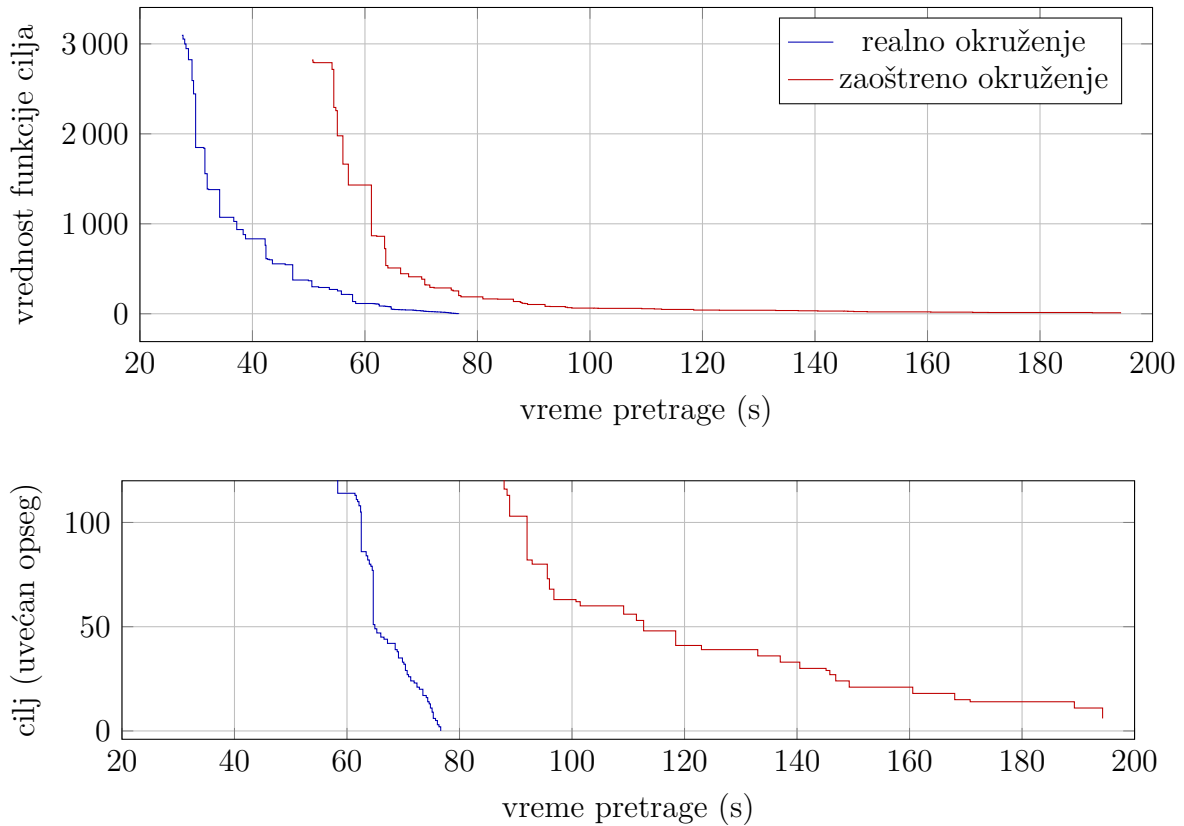
6.2.5.1 Cena proširenja modela

Uvođenje nastavnog osoblja model uvećava oko dva i po puta: broj promenljivih raste 2,6 do 2,9 puta, a broj ograničenja 2,6 do 2,8 puta na svim skalama. Deo tog rasta dolazi od pomoćnih promenljivih kojima se izražavaju pravila osoblja, a deo od toga što instanca sa osobljem opisuje *više* nastave: na najvećoj skali 835 naspram 722 časa.

Zbog te razlike dve konfiguracije ne treba čitati kao merenje „uticaja pravila osoblja“, nego kao poređenje dve različite konfiguracije sistema. One se razlikuju istovremeno po skupu pravila, po broju sesija i po podeli na grupe, pa se iz razlike u vremenima ne može izdvojiti doprinos samih pravila. Eksperiment koji bi to izdvojio – isti skup sesija, sa pravilima osoblja i bez njih – naveden je kao mogući dalji rad u glavi 7.

Razlog za veći broj časova je to što se grupe izvode iz dodela nastave (odjeljak 5.2). Bez fajla sa osobljem upis se deli na grupe od najviše 50 studenata, pa najveća skala ima 30 grupa; dodele nastave, međutim, opisuju stvarnu podelu na 44 grupe (na primer, prva godina Informatike ima šest grupa vežbi umesto dve). Više grupa znači i više časova vežbi.

Pojam kohorte iz odeljka 5.3 radi u suprotnom smeru: dodele oblika „`groupIds`“: [„1i1a“, „1i1b“, „1i1v“] spajaju više grupa u jedan zajednički čas, pa nastavnik isto predavanje ne drži više puta. Na najvećoj skali 139 od 835 časova su takve zajedničke sesije. Zbog toga broj časova raste 16% iako broj grupa raste 47% –



Slika 6.1: Vrednost funkcije cilja u zavisnosti od vremena pretrage, FINAL-L sa osobljem. Svaka tačka je jedno prijavljeno poboljšanje. Donji panel je uvećan na poslednjih stotinak poena cilja, gde se vidi da se kriva u zaoštrenom okruženju zaustavlja na 6, dok u realnom pada na nulu.

spajanje predavanja pojede veći deo razlike.

6.2.5.2 Skalabilnost

Vreme do prvog dopustivog rešenja raste sa veličinom instance (1,62 s, 4,90 s i 23,17 s bez osoblja, odnosno 2,67 s, 8,76 s i 27,46 s sa osobljem), što je očekivano jer raste broj sesija koje treba smestiti u isti broj termina i učionica. Dokaz optimalnosti u konfiguraciji sa osobljem pada u 4,20 s, 15,26 s i 76,78 s, a ukupno vreme sa konstrukcijom modela iznosi 4,44 s, 15,75 s i 77,61 s. Rast je nadlinearan u odnosu na broj sesija, i to pre svega u fazi optimizacije: na najvećoj skali od 77,6 sekundi ukupnog vremena na traženje prvog dopustivog rešenja odlazi 27,5 sekundi, a preostalih 49 na spuštanje cilja do nule i dokaz da bolje nema.

Svi rezultati dobijeni su sa statusom OPTIMAL, dakle donja granica se poklopila sa nadenim rešenjem. Vremenski limit od 600 sekundi ni na jednoj skali nije ni pri-

Tabela 6.12: Putanja poboljšanja, FINAL-L sa osobljem u zaoštrenom okruženju

Vreme	Cilj	Vreme	Cilj	Vreme	Cilj
50,61 s	2 821	71,53 s	293	100,71 s	62
50,76 s	2 795	72,27 s	287	101,46 s	60
50,95 s	2 791	75,34 s	263	109,18 s	56
54,15 s	2 716	75,67 s	254	111,45 s	53
54,46 s	2 293	76,66 s	202	112,73 s	48
54,82 s	2 258	77,06 s	188	118,43 s	41
55,09 s	1 978	80,96 s	165	123,01 s	39
56,08 s	1 663	83,59 s	162	133,02 s	36
57,05 s	1 431	86,39 s	136	137,02 s	33
61,15 s	867	87,60 s	128	140,49 s	30
62,09 s	861	87,78 s	125	145,16 s	29
63,49 s	724	87,93 s	116	145,83 s	27
63,70 s	536	88,46 s	113	146,88 s	24
64,07 s	509	88,91 s	103	149,28 s	21
66,34 s	445	92,01 s	82	160,57 s	18
67,76 s	411	92,92 s	80	168,03 s	15
70,09 s	384	95,61 s	73	170,78 s	14
70,63 s	322	95,99 s	68	189,30 s	11
70,84 s	320	96,78 s	63	194,32 s	6

bližno iskorišćen: najsporije pokretanje završeno je za oko 78 sekundi, dakle unutar 13% raspoloživog vremena. Prošireni model, dakle, ne samo da ostaje rešiv posle svih dodatih ograničenja, nego i dalje ima rezervu za buduća proširenja navedena u glavi 7.

6.2.5.3 Uticaj oskudice resursa

Poređenje dva okruženja pokazuje da težina ovog problema mnogo više zavisi od količine raspoloživog prostora nego od broja pravila. Skraćenje radnog dana sa dvanaest na sedam časova ne menja nijedno ograničenje modela, a ukupno vreme rešavanja najveće instance sa osobljem raste sa 77,6 na 194,9 sekundi, dok cilj prestaje da bude dostižan u nuli.

Najviše se pri tome produži traženje *prvog* dopustivog rešenja: na najvećoj skali sa osobljem sa 27,5 na 50,6 sekundi, a bez osoblja sa 23,2 na 37,3 sekunde. To je i suština razlike između dva okruženja – u realnom je težak deo posla optimizacija, a u zaoštrenom i sama dopustivost. Na manjim skalama je efekat blaži (FINAL-S sa osobljem: 2,67 s naspram 1,67 s), jer prva godina i u kratkom danu ima dovoljno

prostora; na toj skali je zaoštreno okruženje čak i brže, pošto ima manje termina kroz koje pretraga ide.

Vredi primetiti i da je cilj prvog rešenja u zaoštrenom okruženju *manji* nego u realnom (2821 naspram 3095 na najvećoj skali), iako je okruženje strože. U kratkom danu jednostavno ima manje prilika za procepe. Razlika je u tome što se ta manja kazna ne može svesti na nulu: u realnom okruženju cilj je veći ali potpuno uklonjiv, a u zaoštrenom je manji ali sadrži nesvodiv ostatak od 3 poena na najmanjoj i 6 na srednjoj i najvećoj skali.

Konačno, prostor u kome je problem rešiv uži je nego što se iz ovih brojeva vidi. Dan od 8 do 15 časova daje 1015 parova (učionica, termin) za 835 časova, dakle 82% popunjenosti, i rešavač tu još dokazuje optimalnost. Jedan čas kraći dan (870 parova, 96% popunjenosti) menja sliku: rešavač za 600 sekundi ne nađe nijedno dopustivo rešenje i vraća status UNKNOWN. Naglasimo šta taj status znači, a šta ne: u datom vremenskom limitu nije nađeno dopustivo rešenje niti je dokazana nedopustivost. Iz njega, dakle, ne sledi da rešenje ne postoji, niti da je pređena neka granica rešivosti – samo da ga ovaj rešavač u ovoj formulaciji i u ovom limitu nije našao. Da rešenje ne postoji tvrdio bi status INFEASIBLE, koji ovde nije dobijen.

6.2.5.4 Primer rasporeda

Brojevi iz prethodnih tabela ne pokazuju kako dobijeni raspored zaista izgleda, pa u tabeli 6.13 dajemo nedelju jedne grupe, izvučenu iz rešenja najveće skale sa osobljem u zaoštrenom okruženju (dakle iz rešenja sa ciljem 6). Reč je o grupi 1o1, jednoj od grupa prve godine sa 26 časova nedeljno, što je najviše na toj skali. Uzastopni časovi istog predmeta u istoj učionici sažeti su u jedan red.

Raspored ilustruje kako pravila iz glava 4 i 5 izgledaju u ishodu. Časovi istog predmeta spojeni su u dvočasove i tročasove, u skladu sa tvrdim pravilom `joinSameClasses`; nijedan dan nema procep, jer su svi časovi jednog dana uza-stopni; nijedna sesija nije u premaloj učionici, pošto je grupa od 30 studenata svuda smeštena u salu kapaciteta bar 30. Preostala kazna 6 u ovom pokretanju ne potiče od grupa, nego od dva nastavnika koji rade jedan dan više nego što im limit od dva radna dana dozvoljava (tabela 6.10).

6.2.5.5 Praktična upotrebljivost

Dobijeni raspored je dokazano optimalan za zadatu funkciju cilja, ali optimalnost nije isto što i prihvatljivost u praksi. Nekoliko ograda treba navesti eksplicitno:

Tabela 6.13: Nedeljni raspored grupe 1o1 (30 studenata)

Dan	Termin	Učionica (kap.)	Nastava	Časova
Ponedjeljak	8–10	704 (30)	Analitička geometrija, pred.	2
	10–12	N251 (36)	Analiza 1, vežbe	2
	12–15	N102 (36)	Linearna algebra, pred.	3
Utorak	8–10	N203 (36)	Programiranje 2, pred.	2
	10–12	JAG1 (35)	Analiza 1, pred.	2
	12–14	843 (30)	Linearna algebra, vežbe	2
Sreda	12–15	843 (30)	Strani jezik, pred.	3
Četvrtak	8–11	JAG2 (35)	Programiranje 2, vežbe	3
	11–13	JAG2 (35)	Analiza 1, pred.	2
Petak	8–10	830 (40)	Analiza 1, vežbe	2
	10–13	704 (30)	Analitička geometrija, vežbe	3

- **Kapacitet je mek.** Rešenje sa nula nedostajućih mesta dobili smo na svim merenim instancama, ali to nije garancija modela: pošto je kazna po nedostajućem mestu 1, rešavač sme da smesti grupu u premalu učionicu kad god time ukloni dovoljno procepa ili radnih dana preko limita. Raspored sa ciljem različitim od nule zato uvek treba pročitati po komponentama, a ne samo po ukupnoj vrednosti.
- **Grupa sme da menja zgradu u toku dana.** Pravilo o jednoj lokaciji po danu (odeljak 4.4) u završnoj konfiguraciji je isključeno za grupe, a uključeno samo za nastavnike, jer bi sa uključenim pravilom za grupe instanca postala znatno teža. U tabeli 6.13 to se i vidi: grupa u utorak prelazi iz zgrade u Ulici Svetog Nikole u Jagićevu, pa na Studentski trg.
- **Raspoloživost nastavnika nije modelovana.** Model ne poznaje termine u kojima nastavnik ne može da drži nastavu.
- **Ulazni podaci su procene.** Brojevi upisanih studenata i podela na grupe preuzeti su iz dostupnih izvora i ne moraju odgovarati stvarnom stanju na početku semestra.

Uz te ograde, izlaz rešavača je upotrebljiv kao polazna verzija rasporeda koju koordinator dalje doraduje, a ne kao konačan raspored koji se objavljuje bez provere.

6.2.5.6 Validnost

Na svih dvanaest pokretanja, u oba okruženja, nezavisna provera potvrđuje da nema dva časa u istoj učionici u istom terminu, da nijedna grupa nema dva časa istovremeno, da su časovi koji zahtevaju računare u odgovarajućim učionicama i da nijedan nastavnik nije dvostruko zauzet.

Obim te provere treba čitati doslovno: ona pokriva ograničenja ugrađena u model, ali ne i preostala tvrda pravila završne konfiguracije – spajanje časova u blokove, jednu lokaciju nastavnika po danu i nedeljni budžet procepa. Ta pravila su proverena testovima pojedinačnih pravila na malim primerima (odeljak 5.5), što nije isto što i provera na svakom dobijenom rasporedu. Njihovo poštovanje na izmerenim instancama zato počiva na ispravnosti samog modela, a ne na nezavisnoj potvrdi.

Glava 7

Zaključak i dalji rad

7.1 Zaključak

U ovom radu smo razmatrali problem automatskog generisanja rasporeda časova na primeru Matematičkog fakulteta Univerziteta u Beogradu. Implementirana su dva pristupa: programiranje ograničenja (CP-SAT) i binarni celobrojni linearni model (0-1 ILP), poseban slučaj mešovito celobrojnog programiranja, rešavan SCIP-om (oznaka MIP/SCIP). Oba su izvedena korišćenjem biblioteke Google OR-Tools.

Preliminarna evaluacija na tri skale realnih podataka (odeljak 6.1) pokazala je sledeće:

1. **Obe formulacije daju validne rasporede.** Nezavisna provera potvrđuje da rešenja oba rešavača zadovoljavaju sva tvrda ograničenja osnovnog modela, na sve tri instance.
2. **Na ovim instancama je CP-SAT bio brži.** Ukupno vreme MIP/SCIP formulacije veće je 198 puta na najmanjoj, 158 puta na srednjoj i 22 puta na najvećoj instanci. Relativna prednost, dakle, opada sa veličinom instance, dok apsolutna razlika raste (sa oko 8 na oko 35 sekundi).
3. **Dve formulacije se razlikuju po obimu zapisa.** CP model koristi šest celobrojnih promenljivih po sesiji, a 0-1 ILP model po jednu binarnu promenljivu za svaku kombinaciju sesije, dana, sata i dozvoljene učionice, što na najvećoj instanci daje okvirno 10^5 promenljivih. Iz tog odnosa se ne može zaključivati o težini rešavanja, jer jedno globalno ograničenje u CP modelu obuhvata mnogo promenljivih.

4. **Poređenje je uže nego što izgleda.** Oba modela rešavaju problem dopustivosti, pa MIP model radi bez funkcije cilja (odjeljak 6.1.3.4), a zaključci važe za ove dve konkretne formulacije, ova dva rešavača u podrazumevanim podešavanjima i ove tri instance.

Na osnovu tih rezultata, CP-SAT je izabran kao osnova za dalju nadogradnju sistema za raspoređivanje časova. Nadogradnja je sprovedena u dva koraka. U glavi 4 uvedena je arhitektura pravila (eng. *plugin*) sa podrškom za tvrda i meka ograničenja i funkcija cilja koja minimizuje težinsku sumu prekršaja, zajedno sa četiri nova ograničenja kvaliteta – spajanje časova u blokove, jedna lokacija po danu za grupu, izbegavanje procepa u rasporedu i kapacitet učionice kao meko ograničenje. U glavi 5 model je proširen nastavnim osobljem: uveden je pojam kohorte i zajedničkih sesija (kojima se izbegava da nastavnik drži isti čas više puta), tvrdo ograničenje da nastavnik nije dvostruko zauzet, i tri pravila osoblja – jedna lokacija po danu, najveći broj radnih dana i nedeljni budžet procepa.

Završna evaluacija iz odeljka 6.2 pokazala je da prošireni model ostaje praktično upotrebljiv. Uvođenje nastavnog osoblja uvećava model oko dva i po puta, a na najvećoj instanci (sve četiri godine studija, 835 časova) rešavač nalazi optimalan raspored – bez ijednog procepa u rasporedu grupa, bez ijednog nedostajućeg mesta u učionici i bez ijednog nastavnika preko dozvoljenog broja radnih dana – za 77,6 sekundi ukupno. Od toga na prvo dopustivo rešenje odlazi 27,5 sekundi, a preostalih 49 na spuštanje cilja do nule i dokaz optimalnosti.

7.2 Dalji rad

Kroz glave 4 i 5 implementirani su meka ograničenja sa funkcijom cilja, lokacijska ograničenja za grupe i nastavnike, grupisanje časova u blokove, raspoređivanje nastavnog osoblja i kapacitet učionica kao meko ograničenje. Preostali pravci daljeg rada su:

1. **Časovi različitog trajanja.** Model je ograničen na jednosatne sesije (odjeljak 3.2.4). Podrška za čas koji traje dva ili tri sata kao jednu jedinicu zahtevala bi da promenljiva termina nosi i trajanje, a da se zauzeće učionice i grupe izrazi ograničenjem nad intervalima, čime bi otpala potreba za pravilom o spajanju časova u blokove.

2. **Raspoloživost i preferencije nastavnika.** Uključivanje termina u kojima nastavnik ne može ili ne želi da drži nastavu, kao i mekih preferencija (na primer, prepodnevni termini).
3. **Dodatna meka ograničenja kvaliteta.** Na primer, ravnomerna raspodela časova kroz nedelju i izbegavanje kasnih termina, uz fino podešavanje težina u funkciji cilja i analizu njihovog međusobnog odnosa. Težine korišćene u ovom radu izabrane su heuristički, pa bi i njihovo opravdanje merenjem bilo deo tog posla.
4. **MIP model sa funkcijom cilja.** Poređenje iz odeljka 6.1 sprovedeno je nad problemom dopustivosti, pa MIP model radi bez funkcije cilja (odeljak 6.1.3.4). Zapisivanje mekih ograničenja iz glave 4 kao linearnih izraza omogućilo bi poređenje dva rešavača u optimizacionom režimu, sa donjim granicama i procepom optimalnosti sa obe strane.
5. **Izdvajanje uticaja pravila osoblja.** Konfiguracije „sa osobljem“ i „bez osoblja“ razlikuju se i po skupu pravila i po broju sesija, pa se iz njihovog poređenja ne vidi koliko koštaju sama pravila osoblja. Merenje nad istim skupom sesija, sa uključenim i isključenim pravilima osoblja, odgovorilo bi na to pitanje.
6. **Veb interfejs.** Izgradnja veb aplikacije koja omogućava unos podataka, pokretanje rešavača i vizuelni prikaz rasporeda.

Literatura

- [1] Tobias Achterberg. “SCIP: Solving Constraint Integer Programs”. English. In: *Mathematical Programming Computation* 1.1 (2009), pp. 1–41.
- [2] Krzysztof R. Apt. *Principles of Constraint Programming*. English. Cambridge University Press, 2003.
- [3] Samuel Colvin. *Pydantic – Data validation using Python type annotations*. English. on-line at: <https://docs.pydantic.dev/>. 2024.
- [4] S. Even, A. Itai, and A. Shamir. “On the Complexity of Timetable and Multicommodity Flow Problems”. English. In: *SIAM Journal on Computing* 5.4 (1976), pp. 691–703.
- [5] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. English. W. H. Freeman, 1979.
- [6] Eric Gazoni and Charlie Clark. *openpyxl – A Python library to read/write Excel files*. English. on-line at: <https://openpyxl.readthedocs.io/>. 2024.
- [7] Google LLC. *Bazel – a fast, scalable, multi-language build system*. English. on-line at: <https://bazel.build/>. 2024.
- [8] Google LLC. *OR-Tools – Google Optimization Tools*. English. on-line at: <https://developers.google.com/optimization>. 2024.
- [9] Holger Krekel and pytest-dev team. *pytest – The pytest framework*. English. on-line at: <https://docs.pytest.org/>. 2024.
- [10] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. English. John Wiley & Sons, 1988.
- [11] Olga Ohrimenko, Peter J. Stuckey, and Michael Codish. “Propagation via Lazy Clause Generation”. English. In: *Constraints* 14.3 (2009), pp. 357–391.

- [12] Laurent Perron and Frédéric Didier. “CP-SAT Solver”. English. In: on-line at: https://developers.google.com/optimization/cp/cp_solver. 2023.
- [13] Python Software Foundation. *Python Programming Language*. English. on-line at: <https://www.python.org/>. 2024.
- [14] Jean-Charles Régin. “A Filtering Algorithm for Constraints of Difference in CSPs”. English. In: *Proceedings of AAAI* (1994), pp. 362–367.
- [15] Pedro Rodriguez. *PyFunctional – Functional programming in Python*. English. on-line at: <https://github.com/EntilZha/PyFunctional>. 2024.
- [16] Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming*. English. Elsevier, 2006.
- [17] Laurence A. Wolsey. *Integer Programming*. English. John Wiley & Sons, 1998.

Biografija autora

Dušan Trtica rođen je 1983. godine u Beogradu. Završio je Devetu beogradsku gimnaziju 2002. godine. Matematički fakultet Univerziteta u Beogradu, smer informatika i računarstvo, završio je 2009. godine. Oblasti interesovanja uključuju razvoj softvera, funkcionalno programiranje, kao i algoritme veštačke inteligencije.