SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Petar Knezović

SUSTAV ZA AUTOMATSKO RASPOREĐIVANJE SATNICA U SKOLAMA

PROJEKT

VIŠEAGENTNI SUSTAVI

Varaždin, 2023.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Petar Knezović

Matični broj: 0069088496 6

Studij: Informacijski i poslovni sustavi

SUSTAV ZA AUTOMATSKO RASPOREĐIVANJE SATNICA U SKOLAMA

PROJEKT

Mentor:

dr. sc. Bogdan Okreša Đurić

Varaždin, siječanj 2023.

Petar Knezović

Izjava o izvornosti

Izjavljujem da je ovaj projekt izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autorica potvrdila prihvaćanjem odredbi u sustavu FOI Radovi

Sažetak

U ovom seminarskom radu istražuje se problematika razvoja programskog rješenja za raspored nastave u srednjoj školi. Glavni cilj rada je prikazati kako se, koristeći tehnike umjetne inteligencije i koristeći objektno orijentirano programiranje u Pythonu, može razviti efikasan i prilagodljiv sustav za kreiranje i organizaciju školskog rasporeda.Rad se temelji na principima umjetne inteligencije obrađenih na nastavi, prvotno na izradu sustava automatskog planiranja te korištenja algoritma pretraživanja za rješavanje problema. Rad se također temelji na teorijskim i metodološkim principima objektno orijentiranog programiranja, gdje se kroz klase i objekte modelira raspored nastave. U radu se detaljno analizira struktura i funkcionalnost programskog koda koji omogućava raspoređivanje nastavnih sati, učitelja, učionica i grupa učenika. Poseban naglasak se stavlja na algoritam za raspoređivanje koji uzima u obzir različite ograničavajuće faktore poput dostupnosti učitelja i učionica te sprječava preklapanje termina. Zaključno, rad predstavlja kritički osvrt na razvijeni program, njegove prednosti i mogućnosti daljnjeg unapređenja.

Ključne riječi: automatsko planiranje. objektno orijentirano programiranje, Python, školski raspored, algoritam pretraživanja, DFS.

Sadržaj

1. Uvod

Tema razvoja programskog rješenja za automatsko raspoređivanje nastave u srednjim školama odabrana je zbog sve veće potrebe za digitalizacijom i optimizacijom obrazovnih procesa. U današnjem vremenu, tradicionalne metode kreiranja školskih rasporeda često su previše vremenski zahtjevne za nastavnike i podložne ljudskim greškama. Stoga je motivacija za ovu temu proizašla iz želje da se prikaže kako moderne metode umjetne inteligencije mogu unaprijediti i pojednostavniti ove procese, čineći ih jednostavnijima i pouzdanijima.

Osim toga, izbor ove teme podupire i težnja za razumijevanjem i primjenom metoda umjetne inteligencije poput kreiranja sustava za automatsko raspoređivanje i aplikacija algoritama traženja za rješavanje kompleksnih problema sa višestrukim ograničenjima. Također istražujemo kombiniranje metoda umjetne inteligencije sa objektno orijentiranim programiranjem. OOP nam daje jednostavan način za oblikovanje kompleksnih sustava poput ovog kojeg obrađujemo u ovom radu. Kroz ovu temu, istražuju se osnovni principi OOP-a, kao što su korištenje klasa i objekata koji nam omogućuju detaljno strukturiranje našeg programskog rješenja.

U konačnici, rad ima za cilj istaknuti važnost i praktičnost primjene informatičkih znanja u obrazovnom sektoru, s fokusom na unaprjeđenje obrazovnih procesa.

2. Metode i tehnike rada

U razradi ove teme, primjenjuju se različite metode i tehnike rada kako bi se postigao detaljan uvid u problematiku razvoja programskog rješenja za raspored nastave. Glavna metodologija koja se koristi su metode umjetne inteligencije poput sustava za automatsko planiranje koje nam omogućuje generiranje mogućih rješenja za kreiranje našeg rasporeda. Također koristimo DFS algoritam pretraživanja koji nam na jednostavan i efikasan način omogućuje pretragu i pronalazak odgovarajućeg rješenja.

Uz to primjenjujemo metode i tehnike objektno orijentiranog programiranja (OOP) u programskom jeziku Python. Ova metoda omogućava strukturiran pristup razvoju programa, gdje se svaki element sustava za raspored nastave modelira kao objekt sa svojim atributima i metodama. Korištenje Pythona kao programskog jezika odabrano je zbog njegove široke primjene, čitljivosti koda i jednostavnosti upotrebe.

Istraživanje je usmjereno na razumijevanje tipičnih izazova u ovom procesu, kao što su konflikti u rasporedu i ograničenja resursa. Program mora uzeti u obzir da jedan profesor ne može predavati više od jednog predavanja u određenom terminu, da jedna učionica ne može biti korištena za više od jednog predavanja u specifičnom terminu, da profesori mogu predavati samo onaj predmet/e koji su im pripisani te da svaka grupa učenika mora imati sve predmete unutar jednog tjedna. Na temelju ove analize, razvijen je prilagođeni algoritam za raspoređivanje koji uzima u obzir ove faktore.

Također je implementirana je funkcionalnost za sortiranje rasporeda po danima i satima za svaku grupu zasebno. Ovo je postignuto kroz primjenu ugrađenih Python funkcija i metoda za rad s listama i stringovima, što omogućava efikasno sortiranje i prikazivanje podataka.

Za razvoj i testiranje koda koristi se Google Collab okruženje koje omogućuje pojedincima pisanje i izvršavanje Python koda preko browsera. Riječ je o okruženju koje je adekvatno za brojne projekte vezane za aplikaciju metoda umjetne inteligencije poput strojnog učenja kao i za naprednu analizu podataka.

Kroz primjenu ovih metoda i tehnika, rad pruža sveobuhvatan pristup razvoju i optimizaciji programskog rješenja za raspored nastave, demonstrirajući kako se teoretska znanja mogu primijeniti u stvaranju praktičnih i efikasnih softverskih rješenja.

3. Razrada teme

U ovom dijelu ćemo detaljno analizirati i objasniti programski kod kojeg koristimo za izradu ovog progrmaskog rješenja. Također ćemo ukratko opisati output samog programa i njegovu preglednost te ćemo se na kraju kritički osvrnuti na naš kod, analizirajući njegove snage i slabosti.

3.1. Analiza programskog koda

Struktura klasa je ključan koncept OOP-a koji se koristi unutar ovog programskog koda. Strukturiranje klasa nam omogućava kreiranje objekata i jednostavnu manipulaciju s istima.

```
1 class Room:
2     def __init__(self, name):
3         self.name = name
```

Isječak koda 1: Klasa Room

Klasa Room predstavlja učionicu. Svaka učionica ima svoje ime koje ju jedinstveno identificira.

```
1 class Teacher:
2     def __init__(self, name, subjects):
3         self.name = name
4         self.subjects = subjects
```

Isječak koda 2: Klasa Teacher

Klasa Teacher predstavlja nastavnika i sadrži informacije kao što su ime nastavnika i predmeti koje može predavati.

```
1 class Subject:
2     def __init__(self, name):
3         self.name = name
```

Isječak koda 3: Klasa Subject

Klasa Subject se odnosi na školske predmete. Svaki predmet ima svoje ime.

```
class ClassSession:
    def __init__(self, subject, teacher, room, time_slot, group):
        self.subject = subject
        self.teacher = teacher
        self.room = room
        self.time_slot = time_slot
        self.group = group
```

Isječak koda 4: Klasa ClassSession

Klasa ClassSession predstavlja pojedinačni nastavni sat i uključuje informacije o predmetu, nastavniku, učionici, vremenskom terminu i grupi učenika.

```
class Schedule:
    def __init__(self, teachers, rooms, subjects, time_slots, groups):
        self.teachers = teachers
        self.rooms = rooms
        self.subjects = subjects
        self.time_slots = time_slots
        self.groups = groups
        self.classes = []
```

Isječak koda 5: Klasa Schedule

Klasa Schedule je centralna klasa koja upravlja kreiranjem rasporeda. Sadrži metode za dodavanje satova, provjeru valjanosti rasporeda te generiranje konačnog rasporeda.

```
def add_class(self, class_session):
        self.classes.append(class_session)
```

Isječak koda 6: Metoda add class

Metoda add class služi za dodavanje pojedinačnog nastavnog sata, predstavljenog kroz objekt ClassSession, u skup nastavnih sati koji čine školski raspored. Kada se pozove metoda add class, objekt ClassSession se dodaje na kraj liste self.classes. Ovaj proces omogućava postupno kreiranje rasporeda dodavanjem nastavnih sati jedan po jedan. Lista self.classes tako postaje kolekcija svih nastavnih sati koji su raspoređeni u školskom rasporedu.

```
def is_valid(self):
        for i in range(len(self.classes)):
            for j in range(i + 1, len(self.classes)):
                class1 = self.classes[i]
                class2 = self.classes[j]


                if class1.teacher == class2.teacher and class1.time_slot ==
                ↪  class2.time_slot:
                    return False
                if class1.room == class2.room and class1.time_slot ==
                ↪  class2.time_slot:
                    return False


                if class1.group == class2.group and class1.time_slot ==
                ↪  class2.time_slot:
                    return False

        return True
```

Isječak koda 7: Metoda is valid

Metoda is valid unutar klase Schedule igra ključnu ulogu u osiguravanju valjanosti školskog rasporeda. Ova metoda provjerava da li raspored zadovoljava niz ograničenja kako bi se osiguralo da nema sukoba u terminima za učitelje, učionice, i grupe učenika. Metoda prolazi kroz svaki par nastavnih sati u rasporedu koristeći dvije ugniježđene petlje. U svakom koraku, uspoređuju se dva nastavna sata (označena kao class1 i class2) da bi se utvrdilo postoji li konflikt. Ova metoda provjerava tri moguća konflikta: učitelja (ako dva razreda imaju istog učitelja u istom vremenskom terminu), učionice (ako dva predavanja koriste istu učionicu u tom specifičnom terminu) i razredne grupe (ako nastava pokriva dvi grupe u istom terminu). Ako se detektira bilo koji od ovih konflikata, metoda odmah vraća False, što ukazuje na to da raspored nije valjan. Inače vraća True ako nije došlo do ovih konflikata.

```
def schedule_classes_dfs(self, class_index=0):

        if class_index == len(self.subjects) * len(self.groups):
            return True


        subject_index = class_index // len(self.groups)
        group_index = class_index % len(self.groups)
        subject = self.subjects[subject_index]
        group = self.groups[group_index]

        for teacher, room, time_slot in product(self.teachers, self.rooms,
        ↪  self.time_slots):
            if subject in teacher.subjects:
                class_session = ClassSession(subject, teacher, room, time_slot,
                ↪  group)
                self.add_class(class_session)

                if self.is_valid():
                    if self.schedule_classes_dfs(class_index + 1):
                        return True

                self.classes.pop()

        return False
```

Isječak koda 8: Metoda schedule classes dfs

Metoda schedule classes dfs u klasi Schedule je ključna za automatsko kreiranje školskog rasporeda. Ova metoda koristi tehniku pretrage u dubinu (Depth-First Search - DFS) za generiranje valjanog rasporeda koji zadovoljava sve postavljene uvjete. Metoda prolazi kroz sve kombinacije nastavnika, učionica i vremenskih slotova koristeći Pythonovu funkciju product. Za svaku kombinaciju, provjerava se da li nastavnik može predavati dati predmet. Ako može, stvara se novi ClassSession i dodaje se u raspored pomoću metode add class. Nakon dodavanja, **metoda is valid** se poziva da provjeri da li je novi raspored valjan. Ako jest, rekurzivno se poziva schedule classes dfs za sljedeći class index. Ako sljedeći poziv schedule classes dfs vrati True, što znači da je pronađen valjan raspored, metoda također vraća True. Ako kombinacija ne vodi do valjanog rasporeda, posljednji dodani ClassSession se uklanja iz rasporeda (koristeći pop), a pretraga se nastavlja.

```
def print_schedule(schedule):

    def parse_time_slot(time_slot):
        days = ["Ponedjeljak", "Utorak", "Srijeda", "Cetvrtak", "Petak"]
        day, time = time_slot.split(" ", 1)
        hour = int(time.split(":")[0])
        day_index = days.index(day)
        return day_index, hour


    for group in schedule.groups:
        print(f"Raspored za grupu {group}:\n")
        group_classes = [c for c in schedule.classes if c.group == group]
        sorted_classes = sorted(group_classes, key=lambda x:
        ↪  parse_time_slot(x.time_slot))


        for class_session in sorted_classes:
            print(f"Time Slot: {class_session.time_slot}, "
                  f"Subject: {class_session.subject.name}, "
                  f"Teacher: {class_session.teacher.name}, "
                  f"Room: {class_session.room.name}")
        print("\n" * 2)
```

Isječak koda 9: Metoda print schedule

Metoda print schedule u klasi Schedule je dizajnirana za ispisivanje konačnog rasporeda nastave na organiziran i pregledan način. Ova metoda prvo sortira nastavne sate po grupama učenika i vremenskim terminima, a zatim ih ispisuje. Metoda prvo prolazi kroz sve grupe učenika unutar rasporeda. Za svaku grupu, filtrira se lista schedule.classes tako da se izdvoje samo oni nastavni sati koji pripadaju trenutnoj grupi. Ovi filtrirani nastavni sati se zatim **sortiraju koristeći sorted funkciju i parse time slot** kao ključ za sortiranje. Nakon što su nastavni sati sortirani, metoda ih ispisuje, prikazujući vremenski termin, predmet, ime nastavnika i ime učionice za svaki nastavni sat.

```
matematika = Subject("Matematika")
biologija = Subject("Biologija")
hrvatski = Subject("Hrvatski")
glazbeni = Subject("Glazbeni")
kemija = Subject("Kemija")
latinski = Subject("Latinski")
fizika = Subject("Fizika")
njemacki = Subject("Njemacki")
psihologija = Subject("Psihologija")
sociologija = Subject("Sociologija")
logika = Subject("Logika")
```

Isječak koda 10: Inicijalizacija predmeta

Nakon toga provodimo **inicijalizaciju predmeta** . Svaki predmet u školi predstavljen

je instancom klase Subject. Primjeri predmeta uključuju "Matematika", "Biologija", "Hrvatski", i tako dalje. Svaki predmet se inicijalizira s njegovim nazivom. Na primjer, **matematika = Subject("Matematika")** stvara objekt Subject koji predstavlja predmet Matematika.

```
d1 = Room("D1")
d2 = Room("D2")
d3 = Room("D3")
d4 = Room("D4")
d5 = Room("D5")
```

Isječak koda 11: Inicijalizacija učionica

Zatim prelazimo na **inicijalizaciju učionica**. Slično predmetima, učionice su također predstavljene instancama klase Room. Svaka učionica ima jedinstveni identifikator, kao što su "D1", "D2", itd. Inicijalizacijom učionica, kao što je **d1 = Room("D1"),** stvaraju se objekti koji predstavljaju fizičke prostorije u kojima se održava nastava.

```
teacher1 = Teacher("Antonija Maric", [matematika, biologija])
teacher2 = Teacher("Lidija Matic", [hrvatski])
teacher3 = Teacher("Mario Matosic", [hrvatski, latinski])
teacher4 = Teacher("Lea Kalinic", [kemija])
teacher5 = Teacher("Tena Sakic", [fizika])
teacher6 = Teacher("Marko Radic", [njemacki])
teacher7 = Teacher("Toni Saric", [psihologija, logika])
teacher8 = Teacher("Mario Vlasic", [sociologija])
teacher9 = Teacher("Maja Krovinovic", [logika])
```

Isječak koda 12: Inicijalizacija nastavnika

Na kraju prelazimo na **inicijalizaciju nastavnika**. Nastavnici su predstavljeni kroz klasu Teacher. Svaki nastavnik se inicijalizira s imenom i listom predmeta koje može predavati. Na primjer, **teacher1 = Teacher("Antonija Maric", [matematika, biologija])** stvara objekt Teacher koji predstavlja nastavnicu Antoniju Marić s mogućnošću predavanja matematike i biologije.

```
groups = ["4a", "4b", "4c", "4d"]
```

Isječak koda 13: Inicijalizacija grupa

Zatim **definiramo grupe učenika** . Lista groups sadrži oznake za različite grupe učenika (npr., "4a", "4b", "4c", "4d"). Svaka grupa predstavlja određeni razred ili skupinu učenika kojima će biti dodijeljeni nastavni sati.

```
time_slots = ["Ponedjeljak 8:00 - 10:00 ", "Ponedjeljak 10:00 - 12:00", "Ponedjeljak
↪  12:00 - 14:00", "Utorak 8:00 - 9:00", "Utorak 9:30 - 11:00", "Utorak 11:15 -
↪  12:00", "Srijeda 8:00 - 10:00", "Srijeda 10:00 - 11:30", "Srijeda 12:00 -
↪  13:30", "Cetvrtak 8:00 - 10:00", "Cetvrtak 10:00 - 12:00", "Cetvrtak 12:00 -
↪  14:00", "Petak 8:00 - 9:00", "Petak 9:00 - 11:00", "Petak 11:00 - 13:00"]
```

Isječak koda 14: Inicijalizacija termina

Zatim definiramo vremenske slotove (**time slots**). **Lista time slots**sadrži vremenske termine za nastavne sate, strukturirane po danima u tjednu i vremenima. Svaki vremenski slot je definiran kao string koji sadrži dan u tjednu i vremenski raspon (npr., "Ponedjeljak 8:00 - 10:00").

```
schedule = Schedule(teachers, rooms, subjects, time_slots, groups)


if schedule.schedule_classes_dfs():
    print_schedule(schedule)
else:
    print("Raspored nije pronadjen.")
```

Isječak koda 15: Inicijalizacija i generiranje rasporeda

Onda prelazimo na **inicijalizaciju i generiranje rasporeda**. Objekt schedule klase Schedule inicijalizira se s listama nastavnika, učionica, predmeta, vremenskih slotova i grupa. Ovaj objekt predstavlja glavni entitet koji će upravljati i generirati konačni raspored.Poziva se metoda schedule classes dfs na objektu schedule za generiranje rasporeda. Ova metoda koristi tehniku pretrage u dubinu (DFS) za izradu valjanog rasporeda koji zadovoljava sve uvjete. Ako metoda schedule classes dfs vrati True, to znači da je uspješno pronađen valjan raspored, i tada se poziva print schedule za ispisivanje rasporeda. Ako metoda vrati False, ispisuje se poruka "Raspored nije pronađen", što ukazuje na to da nije moguće sastaviti raspored koji zadovoljava sve postavljene uvjete.

3.2. Rezultat programskog rješenja

U ovom poglavlju ćemo se kratko osvrnuti na output ovog programskog rješenja. Output programa uključuje detaljno strukturiran raspored nastave za različite grupe učenika. Ovaj raspored sadrži informacije o rasporedu sati, uključujući vremenske slotove, predmete, nastavnike, učionice i grupe učenika. Raspored je predstavljen u formatu koji je lako čitljiv i pregledan. Za svaku grupu učenika, raspored prikazuje listu nastavnih sati organiziranu po danima i vremenima. Svaki zapis u rasporedu sadrži sljedeće informacije: vremenski slot (dan i sat), predmet, ime nastavnika, i oznaku učionice. Raspored je organiziran tako da su satovi za svaku grupu učenika prikazani zasebno. Unutar svake grupe, satovi su sortirani po vremenskim slotovima, počevši od ponedjeljka i završavajući petkom. To osigurava da je raspored pregledan i prati logičan tijek tjedna. Programski izlaz je dizajniran da bude koristan za školske administratore i

nastavnike, pružajući jasan pregled rasporeda nastave.

3.3. Kritički osvrt na programsko rješenje

U ovom poglavlju pruža se kritički osvrt na razvijeno programsko rješenje za generiranje školskog rasporeda. Analiziraju se njegove snage i slabosti, te se razmatraju potencijalne mogućnosti za unapređenje. Neke od snaga programa su to što program znatno smanjuje vrijeme i napor potreban za ručno kreiranje rasporeda, nudeći brzo i automatsko rješenje, mogućnost jednostavnog prilagođavanja programa ovisno o potrebama škole te preglednost samog ispisa koja olakšava distribuciju rasporeda nastavnicima i učenicima.

Neke od potencijalnih slabosti ovog programskog rješenja su ograničenje algoritma. Iako DFS algoritam omogućava efikasno pronalaženje rješenja, može biti manje efikasan u slučajevima izuzetno kompleksnih rasporeda s mnogo ograničenja. Također postoji mogućnost da program u trenutnoj verziji neće biti prikladan za vrlo velike škole sa jako velikim brojem učenika.

Smatram da je glavna mogućnost za unaprijeđenje programa dodavanje grafičkog korisničkog sučelja koje bi pomoglo u u lakšem upravljanju programom i pružilo bolju korisničku interakciju.

4. Zaključak

projekt razvoja programskog rješenja za generiranje školskog rasporeda demonstrira kako se principi umjetne inteligencije i objektno orijentiranog programiranja mogu uspješno primijeniti za rješavanje složenih problema u obrazovnom sektoru. Razvijeno rješenje predstavlja napredak u efikasnosti i točnosti kreiranja školskih rasporeda, nudeći automatizirani sustav koji minimizira ljudske greške i štedi vrijeme.

Kroz implementaciju algoritma pretrage u dubinu (DFS), program uspješno rješava problem raspoređivanja satova, učitelja i učionica, uzimajući u obzir različita ograničenja i zahtjeve. Dodatna funkcionalnost sortiranja rasporeda po grupama i vremenskim slotovima dalje povećava praktičnost i korisnost ovog rješenja, čineći ga prilagodljivim i lako upotrebljivim alatom za školske administratore.

Ipak, projekt također identificira određena područja za unapređenje, uključujući potrebu za optimizacijom algoritma, poboljšanjem korisničkog sučelja i pružanjem veće prilagodljivosti specifičnim zahtjevima korisnika.

Ukupno gledano, ovaj projekt na rudimentaran način predstavlja važnost i primjenu informatičkih vještina u rješavanju stvarnih problema, potičući inovativnost u edukativnom sektoru.

5. Literatura

Bibliografija

[1] A. Aggarwal and R. J. Anderson, "A random NC algorithm for depth first search," *Combinatorica*, vol. 8, no. 1, pp. 1–12, 1988, doi: 10.1007/BF02122548.

[2] P. D. Causmaecker, P. Demeester, and B. G. A. Vanden, "Decomposed metaheuristic approach for a real-world university timetabling problem," *European Journal of Operational Research*, vol. 195, pp. 307–318, 2009.

[3] M. Yazdani, B. Naderi, and E. Zeinali, "Algorithms for University Course Scheduling Problems," accessed on 03.01.2023, [Online]. Available: `https://hrcak.srce.hr/file/274394`.

[4] "Depth First Search or DFS for a Graph," GeeksforGeeks, [Online]. Available: `https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/`, 03.01.2023.

[5] B. Okreša Đurić, "Laboratorijske vježbe UUI @ SUZG FOI; dio 6." [Online]. Available: `https://colab.research.google.com/drive/1-S9UwVNuuTZ4zAkKuC5AxoNapIRnAHOT`, 03.01.2023.

[6] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Pearson Education Limited, 2022.