Izrada rasporeda ispita na PMF-U

Uroš Bojić

30 June 2023

Sadržaj

Uvod

Opis problema

Izrada rasporeda ispita na fakultetu je veoma složen zadatak koji zahtijeva pravilno raspoređivanje ispita, gdje pri tome imamo ograničene resurse. Cilj je stvoriti optimalan raspored koji zadovoljava različite zahtjeve i ograničenja kako bi se osiguralo učinkovito i ravnomjerno raspoređivanje ispita na obraznovnoj ustanovi.

Raspored ispita mora uzeti u obzir različite faktore kao što su dostupni termini i opterećenost studenta. Osim toga, treba uzeti u obzir i eventualne konflikte kao što su preklapanje termina. Ovaj problem je posebno izazovan jer je potrebno postići ravnotežu između različitih zahtjeva, dok pri tome treba uskladiti raspored da ne dolazi do preklapanja ili konflikta između različitih termina ispita.

Izrada rasporeda ispita na fakultetu je operacija koja može biti vremenski veoma zahtjevna i podložna greškama ako se obavlja ručno. Ručno raspoređivanje često zahtijeva mnogo vremena i truda, a moguće je i da se pojave greške kao rezultat greške osoblja fakulteta ili kao rezultat same složenosti zadatka. Stoga se koriste različiti algoritmi i metode raspoređivanja kako bi se olakšao proces i postigao optimalan raspored ispita.

Važno je napomenuti da je izrada rasporeda ispita na fakultetu NP-težak problem, što znači da nije moguće pronaći brzo i jednostavno optimalno rješenje. Različiti faktori i ograničenja doprinose složenosti problema, pa je potrebno primijeniti napredne algoritme i heuristike kako bi se pronašlo prihvatljivo rješenje u razumnom vremenskom okviru.

Problem izrade rasporeda ispita se može definisati na sljedeći način:
Imamo definisan skup disjunktnih termina T={ t_1, t_2, ..., t_m}. Za svaki termin poznat je datum kada se održava, vrijeme kada počinje, te kapacitet termina (koliko studenata termin da može prihvati). Razliku između termina definišemo kao razliku broja dana između dva termina. Ukoliko se termini održavaju na isti dan tada je razlika između njih jedan.

Definisan je skup predmeta P={p_1, p_2, ..., p_n}. Svaki predmet moguće je polagati samo u jednom terminu.

Definisana je relacija zajedno: P×P→{0,1}, na sljedeći način: predmeti pi i pj su u relaciji zajedno ako su oba biti smještena u isti termin.

Definisan je skup studenata S={ s_1, s_2, ..., s_k}.

Definisana je relacija sluša P×S→{0,1}, na sljedeći način: predmet p_i i student s_j su u relaciji sluša, ako student s_j sluša predmet p_i, i stoga treba polagati ispit iz tog predmeta. Definisana je relacija prihvatljivo s P×T→{0,1}, na sljedeći način: predmet p_i i termin t_j su u relaciji prihvatljivo ako je termin t_j prihvatljiv za održavanje ispita predmeta p_i.

Zadatak je izrada rasporeda ispita, gdje niti u jednom terminu ti ne postoje dva predmeta c_k i c_l dodijeljena tom terminu koji imaju zajedničkog studenta, te da ne postoji termin u kojem je broj smještenih studenata veći od kapaciteta termina.

Kvalitet potencijalnih rješenja se definiše na sljedeći način:

- jako je loše ako postoje studenti koji u istom danu imaju više ispita (ovo je moguće jer u danu postoji više termina kada se mogu polagati ispiti). Što je više ovakvih sukoba to je raspored gori.
- loše je ako postoje studenti koji imaju ispite u susjednim danima (npr. dva za redom) međutim ovo nije toliko loše, koliko je loše kao kad imaju ispite u istom danu).

Pregled radova drugih istraživača

U radu pod nazivom "*An integer programming approach to curriculum-based examination timetabling*" [1], objavljenom u novembru 2017. godine, primijenjena je tehnika Integer programming uz korištenje Python i R programskih jezika. Ovaj pristup ima prednost u većoj vjerovatnoći i efikasnosti u pronalaženju najboljih optimizacija u okviru rasporeda ispita. Međutim, nedostatak ovog pristupa je što ne rješava probleme rasipanja, odnosno ne uzima u obzir konflikte između ispita.

Rad "*Algorithms for optimizing fleet staging of air ambulances*" [2], objavljen je u septembru 2020. godine i u njemu je korišten je algoritam tabu pretrage uz korištenje okvira poput Tabu Search Framework-a. Ovaj pristup kombinuje prednosti kolonija mrava i imunološke teorije kako bi osigurao efikasnost u optimizaciji rasporeda flote vazdušnih ambulanti. Međutim, implementacija ovog pristupa je veoma kompleksna.

"*Elite Immune Ant Colony Optimization-Based Task Allocation for Maximizing Task Execution Efficiency in Agricultural Wireless Sensor Networks*" [3] je rad iz 2020. godine, te se u njemu se primjenjuje algoritam tabu pretrage i to u Matlab programskom jeziku. Ovaj pristup postepeno konvergira ka globalnom optimalnom rješenju kako bi izbjegao lokalna rješenja. Međutim, loše se ponaša u kontekstu nesavršene optimizacije.

U radu "*Intelligent timetable scheduler: A comparison of genetic, graph coloring, heuristic and iterated local search algorithms*"[4], predstavljenom na ICAC (International Conference on Advancements in Computing) 2019. godine, proučavane su četiri metode za izradu rasporeda: genetski algoritam, algoritam bojenja grafova, heuristički algoritam i algoritam iterativne lokalne pretrage.

- Genetski algoritam je pokazao fleksibilnost i sposobnost za globalnu optimizaciju, koristeći paralelnu pretragu i iterativno poboljšanje. Međutim, ovaj pristup je vremenski zahtjevan i efikasnost mu zavisi od izbora parametara i načina reprezentacije problema. Također postoji mogućnost konvergencije ka suboptimalnim rješenjima.
- Algoritam bojenja grafova se istakao po jednostavnoj implementaciji i brzini izvršavanja. On je posebno koristan za identifikovanje konflikata u rasporedu. Međutim, ova metoda je manje prilagodljiva za složenije instance problema raspoređivanja.
- Heuristički algoritam je jednostavan za implementaciju i brz u izvršavanju. Ova metoda može biti prilagodljiva i fleksibilna u odnosu na specifične zahtjeve rasporeda. Međutim, treba uzeti u obzir da heuristički algoritmi obično ne garantuju optimalno rješenje u poređenju s drugim pristupima.
- Algoritam iterativne lokalne pretrage postepeno poboljšava trenutno rješenje putem lokalnih promjena. On je efikasan u pronalaženju blizu-optimalnih rješenja. Međutim, ovaj pristup može zapeti u lokalnom minimumu i ne pruža garanciju globalne optimizacije.

Opis rješenja

Prostor pretrage: Prostor pretrage u sklopu našeg problema obuhvata sve moguće rasporede časova, odnosno sve moguće kombinacije predmet - termin, uključujući i kombinacije gdje termin nije određen. Ove kombinacije se generišu tokom pretrage prostora kako bi se pronašao optimalan raspored koji zadovoljava sve ograničenja i kriterijume koje smo postavili.

Iterated Local Search

Opšti rad algoritma može se opisati kao:

1. Inicijalizacija: Algoritam započinje inicijalizacijom početnog rasporeda ispita. Početni raspored je generisan tako što prolazimo kroz svaki termin i svaki predmet i pokušavamo pronaći prihvatljivu kombinaciju. Takođe, se vrši početna procjena fitness-a rasporeda.

2. Iteriranje: Algoritam izvodi iteraciju određen broj puta (koji unapred definišemo). U svakoj iteraciji petlje, algoritam nastoji poboljšati trenutni raspored ispita korištenjem lokalne pretrage.

3. Generiranje kandidata: U svakoj iteraciji petlje, algoritam generiše nasumičnog kandidata za novi raspored ispita. Novi kandidat je neki modifikirani oblik trenutnog rasporeda, koji dobijamo dodavanjem nasumičnog ispita umjesto postojećeg.

4. Lokalna pretraga: Algoritam primjenjuje lokalnu pretragu nad generisanoim kandidatom kako bi pronašao bolji raspored ispita. Lokalnu pretragu vršimo tako što na početku postavljamo ovog kandidata (raspored) novi najbolji raspored. Vršimo unapred određen broj iteracija gdje tražimo susjede od kandidata. Susjedima kandidata smatramo sve rasporede koji se dobijaju zamjenom termina između dva različita kursa kod kandidata, pri čemu je to jedino što se razlikuje između kandidata i susjeda. Nakon generisanja susjeda biramo nasumičnog susjeda. Ako je on bolji od generisanog kandidate postavljamo njega kao novi najbolji raspored. Nakon izvršavanja unapred određenog broja iteracija vraćamo raspored koji je najbolji.

5. Procjena: Nakon lokalne pretrage, algoritam izračunava fitness novog rasporeda ispita. Ako je novi raspored bolji, ažuriraju se najbolji raspored i njegov fitness.

6. Povrat najboljeg rasporeda: Nakon što je glavna petlja završena, algoritam vraća najbolji raspored ispita koji je pronađen tokom izvođenja algoritma.

```
function find_schedule():
    best_schedule <- find_initial_schedule()
    best_fitness <- calculate_schedule_fitness(best_schedule)
    repeat number_of_iteration times:
        candidate_schedule <- generate_candidate(best_schedule)
        improved_schedule <- local_search(candidate_schedule)
        potential_best <- calculate_fitness(improved_schedule)
        if potential_best < best_fitness:
            best_fitness <- potential_best
            best_schedule <- improved_schedule
    return best_schedule
```

Pseudokod algoritma iterativne lokalne pretrage

```
function local_search(initial_schedule):
    best_schedule <- current_schedule
    best_fitness <- calculate_schedule_fitness(best_schedule)
    repeat number_of_iteration times:
        neighbors <- generate_neighbors(best_schedule)
        if there is no neighbors:
            stop
        neighbor <- randomly choose an element from neighbors
        neighbor_fitness <- calculate_schedule_fitness(neighbor)
        if neighbor_fitness < best_fitness:
            best_schedule <- neighbor
            best_fitness <- neighbor_fitness
    return best_schedule
```

Pseudokod algoritma lokalne pretrage

Računanje fitnesa rasporeda: Neka je *v* ,,velika kazna", a *m* ,,mala kazna". Tada fitnes računamo uz pomoć sljedeće formule:

$$f = \sum_{i \in A} (v - m) + \sum_{i \in B} (v) + \sum_{i \in C} (v) + \sum_{i \in D} (m) + \sum_{i \in E} (m * 2)$$

Gdje je:

A - broj termina u rasoredu koji nemaju određen termin

B - broj termina u rasporedu koji su dodjeljeni kursu kog sluša više studenata nego što može da stane u učionicu

C - broj termina u rasporedu za koje postoji konflikt (postoji još jedan ispit ili više ispita u rasoredu koji slušaju isti studenti kao i ovaj ispit, takvih da im se vremena preklapaju)

D - broj termina u rasporedu takvih da za studente koji polažu ovaj predmet u rasporedu postoji još termin na koji izlaze, gdje su dani ova dva termina uzastopni

E - broj termina u rasporedu takvih da za studente koji polažu ovaj predmet u rasporedu postoji još termin na koji izlaze, gdje su oba termina u istom danu

Napomena: ,,velika kazna", i ,,mala kazna" su unapred definisane vrijednosti. Za veliku kaznu je korištena vrijednost 100000, a za malu kaznu 100.

Genetski algoritam

Opšti rad genetskog algoritma može se opisati sljedećim koracima:

1. Inicijalizacija: Generisanje početne populacije kandidata rješenja. Svaki kandidat predstavlja raspored ispita. Početnu populaciju čine razne modifikacije prvog prihvatljivog rasporeda kojeg pronađemo.

2. Procjena fitness-a: Definišemo funkciju koja ocjenjuje kvalitet svakog kandidata. Funkcija uzima u obzir prilagođenost termina i ukupnu prilagođenost rasporeda. Procjena fitness-a kadidata se vrši na način opisan u prethodno navedenoj formuli.

3. Selekcija: Vrši odabir jedinki iz populacije za kreiranje nove populacije u sljedećoj generaciji. Selekcija uzima u obzir fitness jedinki, dajući veće vjerovatnoće odabira onima s boljim fitness-om. U ovom slučaju, selekcija primjenjuje elitizam, što znači da određeni postotak najboljih jedinki (u našem slučaju. 20%) se direktno prenosi u sljedeću generaciju, čime se osigurava zadržavanje najboljih rješenja.

4. Ukršatanje (Crossover): Vršimo ukrštanje nad odabranim jedinkama kako bi se generisala potomstva. Ukrštanje podrazumeva kombinovanje genetskog materijala dva roditelja kako bi se stvorile nove jedinke. Mi koristimo jednostavno ukrštanje u jednoj tački (simple one-point crossover), a vjerovatnoća da će se izvršiti ukrštanja se proverava pomoću unapred definisanog parametra CROSSOVER_RATE. U našem slučaju ovo znači da će nove jedinke imati određen broj ispita prvog roditelja i određen broj ispita drugog roditelja.

5. Mutacija: Primjenjujemo operaciju mutacije radi uvođenja malih nasumičnih promjena u potomcima. Mutacija pomaže istraživanju novih regiona prostora rješenja. U našem slučaju, vršimo mutaciju rasporeda ispita tako što nasumično odabiremo predmete iz rasporeda i njima dodjeljujemo nasumične termine, koji ne moraju da se nalaze u trenutnom rasporedu. Vjerovatnoća izvršenja mutacije se proverava na osnovu unapred definisanog parametra MUTATION_RATE.

6. Kriterijum zaustavljanja: Algoritam se zaustavlja nakon određenog broja generacija. Broj generacija je unapred definisan parametar.

7. Najbolje rješenje: Nakon završetka iteracija (ponavljanja koraka od 3 do 6 određeni broj puta) dobijamo najbolji raspored. Najbolji raspored je jedinka iz posljednje populacije koja ima najbolju fitness vrijednost.

```
function genetic_algorithm():
    population <- initialize_population()
    best_schedule <- null
    repeat number_of_generations times:
        selected_population <- selection(population)
        offspring_population <- crossover(selected_population)
        mutated_population <- mutation(offspring_population)
        population <- mutated_population
    best_schedule <-  get_best_from_population(best_schedule)
    return best_schedule

function selection(population):
    elite <- select_top_percentage(population)
    return elite

function crossover(population):
    new_population <- population
    while new_population_size < population_size:
        parent1, parent2 <- randomly_select_parents(population)
        if should_do_crossover():
            offspring <- perform_crossover(parent1, parent2)
            new_population.add(offspring)
    return new_population

function mutation(population):
    repeat for every individual in population:
        if should_do_mutation():
            perform_mutation(individual)
```

Pseudokod genetskog algoritma

ILS model:

min: $$\sum_{c_1,c_2,s_1,s_2} (sdp \cdot X[c_1,s_1] \cdot X[c_2,s_2] \cdot \text{sd}(c_1,c_2,s_1,s_2)) + \sum_{c_1,c_2,s_1,s_2} (cdp \cdot X[c_1,s_1] \cdot X[c_2,s_2] \cdot \text{cd}(c_1,c_2,s_1,s_2))$$

s.t. $$\sum_{s} X[c,s] = 1 \quad \forall c \in C \qquad (1)$$

$$X[c_1,s] + X[c_2,s] \leq 1 \quad \forall c_1,c_2 \in C \text{ gdje je share(c1,c2)} = 1, \text{ i } s \in S \quad (2)$$

$$X[c_1,s_1] + X[c_2,s_2] \leq 1 \quad \forall c_1,c_2 \in C \text{ gdje je share(c1,c2)} = 1 \text{ i overlap(s1,s2)} = 1 \quad (3)$$

$$\sum_{c} \text{num\_of\_students}(c) \cdot X[c,s] \leq \text{capacity}(s) \quad \forall s \in S \quad (4)$$

$$\text{X[c,s]} = \begin{cases} 1, & \text{predmetu c je dodjeljen termin s} \\ 0, & \text{predmetu c nije dodjeljen termin s} \end{cases}$$

$$\text{sd(c1,c2,s1,s2} = \begin{cases} 1, & \text{postoje studenti koji slušaju predmete c1 i c2, sa terminima s1 i s2 koji su zakazani za isti dan} \\ 0, & \text{ne postoje studenti koji slušaju predmete c1 i c2, sa terminima s1 i s2 koji su zakazani za isti dan} \end{cases}$$

$$\text{cd(c1,c2,s1,s2} = \begin{cases} 1, & \text{postoje studenti koji slušaju predmete c1 i c2, sa terminima s1 i s2 koji se odžavaju dan za danom} \\ 0, & \text{ne postoje studenti koji slušaju predmete c1 i c2, sa terminima s1 i s2 koji se odžavaju dan za danom} \end{cases}$$

$$\text{share(c1,c2)} = \begin{cases} 1, & \text{predmete c1 i c2 sluša isti student} \\ 0, & \text{predmete c1 i c2 ne sluša isti student} \end{cases}$$

$$\text{overlap(e1,e2)} = \begin{cases} 1, & \text{termini e1 i e2 se preklapaju} \\ 0, & \text{termini e1 i e2 se ne preklapaju} \end{cases}$$

sdp - unapred definisana konstanta koja predstavlja kaznenu vrijednost rasporeda ispita koji u istom danu imaju više ispita za iste studente

cdp - unapred definisana konstanta koja predstavlja kaznenu vrijednost rasporeda ispita koji u uzastopnim danima imaju više ispita za iste studente

S predstavlja skup svih termina, dok *C* predstavlja skup svih predmeta. Pri izradi redoslijeda potrebno je minimizovati opterećenost rasporeda. U slučaju da se u rasporedu nalazi termin koji opterećava raspored, u skladu sa vrstom opterećenja takvom rasporedu se dodjeljuje kazna. Svaki od predmeta mora imati dodjeljen termin, te se ispit iz jednog predmeta ne može držati u više termina (1). Dva predmeta koje slušaju isti studenti ne mogu biti zakazani u istom terminu (2). Termini za dva predmeta koje slušaju isti studenti ne mogu da se preklapaju., odnosno ne može ispit za studenta da počne, dok drugi ispit traje (3). Na ispitu ne može da bude više studenata nego što može stane u učionicu (4).

Eksperimentalni rezultati

	ILS			GA			Solver	
Name	Best	Avg	t	Best	Avg.	t	Best	t
small_1	0	0	215	0	80	405	0	12
small_2	0	20	277	100	150	538	0	19
small_3	0	0	229	0	0	490	0	15
small_4	0	10	225	100	100	492	0	16
small_5	0	60	252	100	250	526	0	22

Rezultati za male instance

	ILS			GA		
Name	Best	Avg	t	Best	Avg.	t
medium_1	0	20	3.71	1100	34800	9.37
medium_2	200	350	3.75	10070	47850	9.29
medium_3	6500	8610	4.01	325600	467100	9.22
medium_4	6800	7920	3.65	18100	85260	9.32
medium_5	6300	6900	4.22	222500	289430	10.84

Rezultati za srednje instance

	ILS			GA		
Name	Best	Avg	t	Best	Avg.	t
big_1	37800	60850	32.89	1849000	3309710	42.29
big_2	46000	74380	33.09	2052500	3621250	44.27
big_3	62100	81970	33.17	1169800	2690540	57.78
big_4	81300	90730	32.76	483100	835760	43.11
big_5	86200	146540	34.51	887800	2769930	46.49

Rezultati za velike instance

Okruženje: Eksperimentalno okruženje u kojem je testiran program:

Operativni sistem: MacOS

CPU: Apple M1

Broj niti (threads): 8 fizičkih jezgara, 8 logičkih procesora

Programski jezik: Python

Okruženje: Visual Studio Code

Instance

Instance su generisane na nasumičan način, na početku smo definisali parametre koji su fiksirani.

Vrsta instance	Broj ispita	Broj predmeta	Broj grupa studenenata	Broj različitih dana
Male	20	10	5	7
Srednje	150	150	50	14
Velike	300	300	150	31

Fiksirani parametri pri generisanju instanci

Preostali parametri se biraju na nasumičan način iz određenih opsega. Opsezi su određeni tako da za različite instance stvaraju različite brojeve "konflikata". Pod konfliktima smatramo konflikte kapaciteta između broja studenata na predmetu i broja dostupnih mjesta u učionici, te otežavanje raspoređivanja u vidu dodjeljivanja većeg broja studenata nekom predmetu, čim se otežava pronalaženje termina za predmet, gdje taj termin ne treba da se poklapa sa terminima drugih ispita koji studenti slušaju. Srednje instance imaju veći broj konflikata nego male instance, dok velike instance imaju veći broj konflikata nego i male i srednje instance. Za instance iste veličine, instanca sa indeksom i bi trebala da ima veći broj potencijalni konflikata nego instanca sa indeksom j za $i < j$, gdje $i, j \in [1,5]$. (što zavisi od vrijednosti odabranih parametra)

Naziv instance	Kapacitet učionice		Broj grupa studenata		Broj predmeta koji student sluša	
	od	do	od	do	od	do
small_1	50	100	10	20	1	5
small_2	60	100	10	20	1	5
small_3	65	100	10	20	1	5
small_4	70	100	15	20	2	5
small_5	80	100	18	20	4	5
medium_1	50	100	10	20	1	5
medium_2	50	150	10	20	3	8
medium_3	60	180	15	25	5	10
medium_4	80	180	15	25	7	12
medium_5	100	200	20	25	7	12
big_1	0	300	10	20	10	15
big_2	0	380	10	20	13	18
big_3	0	380	15	25	15	20
big_4	50	450	15	25	20	25
big_5	200	750	15	25	25	30

Parametri koji se koriste pri nasumičnom generisanju instanci

Napomene: Svi termini počinju između 08:00 i 16:00, dok svaki ispit traje između 1h i 4h 45m.

U tabeli *Rezultati za male instance* prosječno vrijeme je izraženo u milisekundama. Kod genetskog algoritma, za male instance smo koristili smo veličinu populacije 100, broj generacija 50 i stopu mutacije 20%. Algoritme smo puštali po 10 puta.

U tabelama *Rezultati za srednje instance i Rezultati za velike instance prosječno* vrijeme je izraženo u minutama. Kod genetskog algoritma, i za srednje i za velike instance smo koristili smo veličinu populacije 500, broj generacija 100 i stopu mutacije 60%. Algoritme smo puštali 5 puta.

Svi ostali parametri, pri oba algoritma, su isti bez obzira na veličine instance.

Vizuelizacija izgleda rješenja instance

```
Exam Schedule:
Course 1 - 2023-07-10, 14:00, 1h 0m
Course 2 - 2023-07-08, 11:00, 4h 30m
Course 3 - 2023-07-14, 16:00, 4h 45m
Course 4 - 2023-07-07, 11:00, 4h 30m
Course 5 - 2023-07-08, 11:00, 4h 30m
Course 6 - 2023-07-14, 16:00, 4h 45m
Course 7 - 2023-07-07, 11:00, 4h 30m
Course 8 - 2023-07-12, 15:00, 1h 15m
Course 9 - 2023-07-14, 16:00, 4h 45m
Course 10 - 2023-07-12, 15:00, 1h 15m
```

Raspored ispita za instancu small_3

```
Exam Schedule:
Group 1
Course 9 - 2023-07-14, 16:00, 4h 45m
Course 4 - 2023-07-07, 11:00, 4h 30m
Course 1 - 2023-07-10, 14:00, 1h 0m
Group 2
Course 8 - 2023-07-12, 15:00, 1h 15m
Group 3
Course 8 - 2023-07-12, 15:00, 1h 15m
Course 4 - 2023-07-07, 11:00, 4h 30m
Course 1 - 2023-07-10, 14:00, 1h 0m
Course 3 - 2023-07-14, 16:00, 4h 45m
Group 4
Course 6 - 2023-07-14, 16:00, 4h 45m
Course 10 - 2023-07-12, 15:00, 1h 15m
Course 7 - 2023-07-07, 11:00, 4h 30m
Course 1 - 2023-07-10, 14:00, 1h 0m
Group 5
Course 8 - 2023-07-12, 15:00, 1h 15m
Course 7 - 2023-07-07, 11:00, 4h 30m
```

Raspored ispita za instancu small_3 po grupama

Zaključak

Izrada rasporeda ispita na fakultetu je veoma složen zadatak koji zahtijeva pravilno raspoređivanje ispita uz ograničene resurse. U ovom radu smo predstavili tri pristupa, odnosno metode za rješavanje problema raspoređivanja ispita. Analizirani su radovi drugih istraživača koji su primijenili tehnike kao što su Integer programming, algoritam tabu pretrage, genetski algoritam, algoritam bojenja grafova i heuristički algoritam.

Rezultati za male instance su vrlo dobri za ILS algoritam, s optimalnim rješenjima pronađenim za svaku instancu. ILS je efikasan za ove instance s prihvatljivim vremenom izvršavanja. GA nije postigao jednako dobre rezultate, s nekim optimalnim rješenjima samo u određenim slučajevima i lošijim prosječnim vrijednostima u usporedbi s ILS-om. GA je manje efikasan od ILS-a za male instance.
Što se tiče srednjih i velikih instanci, može se zaključiti da ILS ima nešto bolje performanse u pogledu pronađenih rješenja i vremena izvršavanja. Vidimo da oba pristupa imaju nedostatke, ILS ima visoke prosječne rezultate, a GA ima i visoke prosječne rezultate i veoma velike varijacije u rezultatima.

Dalje unapređenje rada može uključivati istraživanje i primjenu drugih heurističkih algoritama kao što je algoritam tabu pretrage ili simulirano kaljenje. Dalje unapređenje rada isto tako može uključivati i prilagođavanje parametara iterativne lokalne pretrage i genetskog algoritma kako bi se poboljšala njihova efikasnost i konzistentnost za srednje i velike instance..Također, u ova dva pristupa moguće je dodati dodatne faktore i ograničenja u procjenu fitness-a, kao što su preferencije studenata, vremenski intervali između ispita ili maksimizacija iskorišćenost prostora.

Literatura

- Michel Gendreau and Jean-Yves Potvin (2018). Handbook of Metaheuristics
- Suman B., Maniezzo, V., Ricciardelli, S. (2005). A Guided Tour of Combinatorial Optimization. Springer.
- B. Sigl, M. Golub, V. Mornar (2003). Solving timetable scheduling problem using genetic algorithms
- Hayat A., Tahani A., Hosam A. and Abdullah B. (2020). A Review of Optimization Algorithms for University Timetable Scheduling
- Ender Özcan and Ersan Ersoy (2005). Final Exam Scheduler – FES
- Marko Čupić (2009). Prirodom inspirirani optimizacijski algoritmi
- Masri Ayob, Ariff Md. Ab. Malik, Salwani Abdullah and Abdul Razak Hamdan (2007). Solving a Practical Examination Timetabling Problem: A Case Study
- Vittorio Maniezzo, Roberto Battiti, and Alberto Colorni (1996). Iterated Local Search
- Dario Landa-Silva and Edmund K. Burke (1999). A Genetic Algorithm Approach to Examination Timetabling Problems"
- Pierre Hansen, Nenad Mladenović (2008): Iterated Local Search: Framework and Application

Reference:

[1] Alejandro Cataldo, Juan-Carlos Ferrer, Jaime Miranda, Pablo A. Rey & Antoine Sauré. (2017). An integer programming approach to curriculum-based examination timetabling

[2] Joseph Tassone a, Geoffrey Pond b, Salimur Choudhury (2020). Algorithms for optimizing fleet staging of air ambulances

[3] Zhenxing Zhang (2020). *Elite Immune Ant Colony Optimization-Based Task Allocation for Maximizing Task Execution Efficiency in Agricultural Wireless Sensor Networks*

[4] Tiny Wijerathna Ekanayake, Pavani Subasinghe, Shawn Ragel, Anjalie Gamage (2019). Intelligent timetable scheduler: A comparison of genetic, graph coloring, heuristic and iterated local search algorithms