Opis primijenjenog algoritma tj. heuristike

Heuristika koja je korištena pri rješavanju problema sastoji se od dva dijela. U prvom dijelu provođenja cijelog postupka korišten je **pohlepni algoritam** (eng. *greedy*) čiji je cilj konstruirati početno rješenje koje je bolje od rješavanja zahtjeva nasumičnim odabirom, a u drugom dijelu koristi se hibridna inačica **tabu pretraživanja.**

Pohlepni algoritam sortira zahtjeve studenata prema tome koliko je pojedini student postavio zahtjeva (potencijalnih grupa za promjenu) za pojedinu aktivnost. Studenti koji imaju više ponuđenih grupa za zamjenu za pojedinu aktivnost imaju prednost.

Nakon toga, sve grupe koje je student postavio kao potencijalne za zamjenu sortiraju se prema dva parametra. Prvi se računa kao *min_preffered – student_count* i prema njemu se daje prednost grupama kojima je preferirani minimum veći od broja studenata koji su trenutno u toj grupi. Drugi parametar za sortiranje grupa je broj ukupnih zahtjeva za ulazak u tu grupu. Grupe koji inicijalno imaju manji broj zahtjeva za ulazak imaju prednost.

Varijabilni element ovog dijela algoritma je broj grupa koji se uzima u obzir kod tog određenog zahtjeva. Naime, ne uzimaju se uvijek sve grupe već *random* broj grupa između 1 i ukupnog broja tih grupa (po uzoru na GRASP), da bi se ostvario barem minimalni element stohastičnosti.

Također, uzima se u obzir aspekt broja zamjena kod pojedinog studenta, pa se iz tog razloga student kojem se izvrši jedna zamjena stavlja u svojevrsnu tabu listu koja onemogućava zamjene za tog studenta sljedećih 10 iteracija. Broj iteracija za trajanje tabua je varijabilan i omogućuje varijabilnost.

Algoritam zapravo nije u doslovnom smislu pohlepan jer dopušta poboljšanja – izvršava se nekoliko puta (trenutno je količina ponavljanja određena vremenskim parametrom) te se dopušta zamjena za nekog studenta u neku drugu grupu na toj aktivnosti, iako je već ušao u neku od traženih grupa. To je zbog toga što se za vrijeme trajanja algoritma ispituje može li se nekim drugim odabirom za tog studenta i tu aktivnost postići bolje ukupno rješenje i omogućiti veći broj zamjena.

Drugi dio algoritma je varijacija **tabu pretraživanja.** Izlaz nakon izvršavanja pohlepnog algoritma predstavlja ulaz u tabu pretraživanje. Ukoliko se rješenje nakon jedne

iteracije tabua poboljša, tada se to rješenje uzima kao trenutno globalno najbolje i predstavlja ulaz u sljedeću iteraciju. Koncept koji je preuzet iz tabu pretraživanja je upravo nastavak pretraživanja ukoliko se i lokalno rješenje nakon trenutne iteracije ne poboljša (za razliku od npr. lokalnog pretraživanja). Također, preuzeto je kreiranje susjedstva pretraživanja u jednoj iteraciji, čija veličina ovisi o veličini instance problema – uzima se 20 posto ukupnog broja zahtjeva zaokruženo na cijeli broj koji se odabiru nasumično.

Samo tabu pretraživanje radi tako da se provode zahtjevi iz susjedstva koji su mogući uz ograničenja s ciljem poboljšanja trenutnog najboljeg rješenja.

U tom dijelu algoritma ne postoji tabu lista zbog toga što je susjedstvo nasumično odabrano i ograničene veličine pa se time ionako izbacuje mogućnost ponovne zamjene za određeni zahtjev. Ono što svakako predstavlja potencijalno poboljšanje u ovom dijelu algoritma jest odabir zahtjeva za susjedstvo. Jedna od mogućnosti je davanje prednosti zahtjevima kod kojih su studenti još neraspoređeni (*student_count – max_preffered* je veći od 0).

Prikaz rješenja

Rješenje nakon izvođenja cijelog algoritma ekvivalentno je obliku prikaza tijekom pojedinih dijelova algoritma. Rješenje se prikazuje u obliku **liste stanja studenta** koja inicijalno poprima oblik prema ulaznoj datoteci *studentsFile* (ima parametre *student_Id, activity_Id, swap_weight, group_id, new_group_id*). Tijekom izvođenja se mijenja varijabla *new_group_id* i one vrijednosti koje se na kraju izvođenja algoritma nalaze u toj varijabli smatraju se rješenjem.

Funkcija cilja/prikladnosti

Funkcija cilja je ujedno i funkcija prikladnosti. Ona se računa prema formuli danoj u opisu problema, tj. **Ukupna ocjena rješenja = BodoviA + BodoviB + BodoviC – BodoviD – BodoviE.** Pojedini bodovi računaju se prema uputama iz opisa zadatka.

Način dobivanja početnog rješenja

Način dobivanja početnog rješenja opisan je iznad u opisu algoritma. Dakle, inicijalno rješenje je dobiveno nakon jedne iteracije pohlepnog dijela algoritma.