Pridruživanje s prioritetnim listama u raspodijeljenim sustavima primijenjeno na nastavne procese.

Paunović, Vlatka

Doctoral thesis / Disertacija

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:168:486371

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-18**

Repository / Repozitorij:

FER Repository - University of Zagreb Faculty of Electrical Engineering and Computing repozitory

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

Sveučilište u Zagrebu
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Vlatka Paunović

PRIDRUŽIVANJE S PRIORITETNIM LISTAMA U RASPODIJELJENIM SUSTAVIMA PRIMIJENJENO NA NASTAVNE PROCESE

DOKTORSKI RAD

Zagreb, 2020.

Sveučilište u Zagrebu
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Vlatka Paunović

PRIDRUŽIVANJE S PRIORITETNIM LISTAMA U RASPODIJELJENIM SUSTAVIMA PRIMIJENJENO NA NASTAVNE PROCESE

DOKTORSKI RAD

Mentor:
Prof. dr. sc. Mario Žagar

Zagreb, 2020.

University of Zagreb
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Vlatka Paunović

PREFERENCE BASED ALLOCATION IN DISTRIBUTED SYSTEMS APPLIED ON EDUCATIONAL PROCESSES

DOCTORAL THESIS

Supervisor:
Professor Mario Žagar, PhD

Zagreb, 2020

Doktorska disertacija izrađena je na Sveučilištu u Zagrebu Fakultetu elektrotehnike i računarstva, na Zavodu za automatiku i računalno inženjerstvo.

Mentor:
Prof. dr. sc. Mario Žagar

Disertacija ima: 140 stranica

DOKTORSKA DISERTACIJA BR: _____

O MENTORU:

Mario Žagar rođen je u Kupjaku (Gorski kotar) 1952. godine. Diplomirao je, magistrirao i doktorirao u polju računarstva na Sveučilištu u Zagrebu Fakultetu elektrotehnike i računarstva (FER), 1975., 1978. odnosno 1985. godine.

Od siječnja 1977. godine radi na Zavodu za automatiku i računalno inženjerstvo FER-a. i od tada je uključen u različite znanstvene projekte, nastavne aktivnosti, poboljšanja nastavnih programa, laboratorija i opreme. Bio je gostujući istraživač na Sveučilištu u Rostocku (Njemačka, 1980. godine), nositelj je stipendija „British Council fellowship“ (UMIST - Manchester, 1983. godine) i „Fulbright fellowship“ (UCSB - Santa Barbara, 1983./1984. godine). Godine 2002. izabran je za redovitog profesora računarstva u trajnom zvanju. Sudjelovao je u brojnim domaćim i međunarodnim istraživačkim projektima. Bio je član i voditelj pet znanstvenih projekata koje financira Ministarstvo znanosti, obrazovanja i sporta Republike Hrvatske. Bio je također suvoditelj/voditelj projekata „Unity Through Knowledge (UKF)“, „TEMPUS“ (u suradnji sa Sveučilištem Paderborn, Njemačka) te više projekata sa Sveučilištem Mälardalen, Västerås, Švedska. M. Žagar autor je i koautor više od stotinu članaka i pet knjiga iz područja računalnih arhitektura, raspodijeljenog, sveprisutnog i prožimajućeg računarstva, Interneta stvari, otvorenog računarstva, e-učenja i drugih Internetu usmjerenih tehnologija.

Prof. Žagar je senior member IEEE-a i IEEE CS-a, počasni predsjednik udruge HrOpen (Hrvatske udruge za Otvorene sustave i Internet) te redoviti član Akademije tehničkih znanosti Hrvatske (HATZ). Sudjelovao je u više programskih odbora međunarodnih konferencija, član je dva urednička odbora i recenzent u više međunarodnih časopisa. M. Žagar primio je Brončanu plaketu Josip Lončar FER-a 1975. godine, nagrade INFORMATIKA93 i INFORMATIKA96 Hrvatske Informatičke Zajednice (HIZ), najviše priznanje Zlatnu plaketu Josip Lončar za unaprjeđenje nastave, znanstveno-istraživačkog rada i organizacije dodjeljuje FER 2002. godine, nagradu „Najbolji edukator“ Hrvatske Sekcije IEEE-a, 2006. godine, nagradu Hrvatske akademije znanosti i umjetnosti (HAZU) “Josip Juraj Strossmayer” za najuspješniji znanstveni rad u Republici Hrvatskoj u polju informacijskih znanosti, 2007. godine (za e-izdanje knjige „UNIX i kako ga iskoristiti“). Također je dobitnik „Priznanja za vođenje razvoja dinamičke aplikacije FERWeb“ koja je od 2001. godine do danas središnja stranica Weba FER-a, prvu nagradu za najbolji e-kolegij na Sveučilištu u Zagrebu za predmet Otvoreno računarstvo (akademska godina 2009./2010.) i još niz drugih nagrada i priznanja.

ABOUT THE SUPERVISOR:

Mario Žagar was born in Kupjak (Gorski kotar) in 1952. He received his Dipl.ing., M.Sc.CS and Ph.D.CS degrees, all from the University of Zagreb, Faculty of Electrical Engineering and Computing (FER) in 1975, 1978 and 1985 respectively.

From January 1977 he is working at the Department of Control and Computer Engineering at FER and since then was involved in different scientific projects, educational activities, improvements of the educational programs, laboratories and computer equipment. He was visiting researcher at the University of Rostock, Germany (1980), received British Council fellowship (UMIST – Manchester, 1983) and Fulbright fellowship (UCSB – Santa Barbara, 1983/84). In 2002 he was promoted to Tenure Professor of Computing. He participated in numerous domestic and international research projects. He was a member and project leader in five scientific projects financed by Ministry of Science, Education and Sports of the Republic of Croatia. He was also co-leader/leader in Unity Through Knowledge (UKF) project, TEMPUS project (in cooperation with the University of Paderborn, Germany), several projects in cooperation with the Mälardalen University, Västerås, Sweeden. M. Žagar is author and co-author of more than hundred articles and five books in the area of computer architectures, distributed, ubiquitous and pervasive computing, Internet of Things (IoT), Open computing, e-learning and other Internet related technologies.

Prof. Žagar is a senior member of IEEE/CS, honorable president of Croatian Society for Open Systems, regular member of Croatian Academy of Technical Sciences. He participated in several international conference program committees, he is a member of two journal editorial boards and he serves as a technical reviewer for various international journals. M. Žagar was awarded by FER with the Bronze plaque Josip Lončar (1975), awards INFORMATIKA93 (1993) and INFORMATIKA96 (1996) from the Croatian Informatics Society and the highest award, Golden plaque Josip Lončar for the improvement of education, scientific and research work and organization of the Faculty from FER (2002), “Best educator” award, IEEE/CS Croatia Section (2006), Croatian Academy of Sciences and Arts (HAZU) award “Josip Juraj Strossmayer” for the most successful scientific work in Croatian in the field of information science (2007), for e-edition of “Unix and how to utilize it” book. He also received the “Acknowledgment for leading the development of the dynamic FERWeb application” started in 2001 and currently the central FER Web site, first prize for the best e-course at the University of Zagreb (academic year 2009/2010), Open Computing course and several other awards and acknowledgments.

ZAHVALE

Prilikom izrade disertacije, nekoliko osoba u mojem životu su iznimno pomogle, posredno ili neposredno, te svakako zaslužuju i ovu kratku zahvalu unutar disertacije.

…najprije, veliko hvala, mojem mentoru, **Mariu**, jer je uvijek vjerovao u mene i nikad nije odustao u podršci, čak i kada je njemu postalo teško. Uz mene je tijekom mog cijelog puta do doktorata i uvijek je bio otvoren za sva pitanja s beskonačnom količinom vremena.

…iako na drugom mjestu, jednako važnom, veliko hvala **Siniši**, na svoj pomoći, savjetima, slušanjima mojih ideja, čitanjima, ispravljanjima, kao i traganjima za rješenjima.

…veliko hvala veseloj i dragoj **Ivani**, koja je svojim savjetima postigla da članak ugleda svjetlo dana i moj prvi članak bude objavljen u Q1 časopisu. Hvala i na neizmjernim pitanjima i podsjećanjima, te na svemu što je učinila kako bi iz mene izvukla ono najbolje i najviše.

…za čitanje s razumijevanjem, lektoriranje, engleskih ispravaka, traženja matematičkih pogrešaka i davanje savjeta neposrednih čitatelja, veliko hvala mojem bratu **Miši** i njegovoj ženi **Lyubomiri**.

…veliko hvala i mojoj **Mami** na svom uskraćenom vremenu za koje je imala razumijevanje, a ja joj ga nisam pružila zbog istraživanja i tipkanja, te svog strpljenja u mom procesu doktoriranja.

…veliko hvala **Sveboru**, **Damiru i Mirjani** na višegodišnjem poticanju,…

…i na samom kraju, veliko hvala, svoj bližoj i daljoj rodbini i prijateljima, koji su me pitali kako doktorat napreduje i koji su se putem iskreno veselili i vjerovali u mene.

SAŽETAK

Problem pridruživanja uparuje elemente iz dva skupa. Svaki od elemenata iz jednog skupa rangira na svojoj prioritetnoj listi elemente iz drugog. Uobičajen problem pridruživanja u nastavnim procesima je pridruživanje studenata grupi. Grupa pri tome može označavati projekt, mentora, poziciju, smjer i slično, a za sve je zajedničko da imaju ograničeni broj mjesta. Prioritetne liste pojedinih studenata su međusobno različite i mogu biti nepotpune. U ovoj disertaciji grupe koriste zajedničku prioritetnu listu – glavni popis u kojem su rangirani svi studenti prema bodovima koje su ostvarili. Nepotpune prioritetne liste koje su izradili studenti u kombinaciji s ograničenim brojem slobodnih mjesta u grupama rezultira nepridruženim studentima.

Cilj doktorske disertacije je istražiti problem pridruživanja s prioritetnim listama u nastavnim procesima, a temelji se na pretpostavci da je primjenom ekspertnog znanja o studentima i samom nastavnom procesu moguće razviti nove metode za kvalitetnije pridruživanje studenata.

Nova predložena *metoda odabira redoslijeda elemenata* koristi ekspertna znanja izrade glavnog popisa, a uvodi i novi neizrazit pristup koji pretpostavlja jednakost u rangu dva studenta čak i kada postoji mala razlika u njihovim bodovima. Rezultat nove metode je značajno smanjivanje nepridruženih studenata do 10% uz zadržavanje važnosti poretka studenata na glavnom popisu.

Nova predložena *metoda pridruživanja nepridruženih studenata* koristi znanje dobiveno iz odabira drugih studenata pomoću kojeg se kreira proširena prioritetna lista. Drugačija je za svakog studenta, a sadrži listu grupa koje student nije odabrao na svojoj prioritetnoj listi, poredanu kako bi to on najvjerojatnije sam napravio. Predložena metoda omogućava pridruživanje grupe bolje rangiranim studentima kada na izrađenoj prioritetnoj listi studenta više nema odabranih grupa. Rezultira pridruživanjem bolje rangiranih studenata u popularnije grupe i smanjuje broj studenata kojima su grupe odabrane slučajnim odabirom.

Obje metode predstavljaju središnji dio novog *modela raspodijeljenog rješavanja problema pridruživanja.* Model je definiran s četiri nova predložena algoritma, a temeljni je (i) *Algoritam strogog jednostrukog pridruživanja s glavnim popisom* (i). On je proširen s dodatna tri algoritma: (ii) *Algoritam višestrukog pridruživanja s glavnim popisom*, (iii) *Algoritam višestrukog pridruživanja s glavnim popisom i višeslojnom prioritetnom listom* te (iv) *Algoritam jednostrukog pridruživanja grupiranih elemenata*.

Analiza rezultata predloženog modela na primjeru podataka predmeta Seminar Fakulteta elektrotehnike i računarstva za više od 500 studenata i više od 200 projekata pokazala je poboljšanje kvalitete pridruživanja za 25% studenata u odnosu na Algoritam odgođenog prihvaćanja koji se smatra jednim od temeljnih algoritama za rješavanje problema pridruživanja. Kvaliteta pridruživanja obuhvaća manji broj nepridruženih studenata – 10%, te smanjenje broja nasumično pridruženih studenata grupama za više od 15%.

Model je definiran i u raspodijeljenim sustavima – arhitekturom agenata, a poboljšanje koje ostvaruje implementacija u vidljiva je kako u kvaliteti rješenja, tako i u brzini dolaska do njih. Izrađeni prototip modela primijenjen je na sedam primjera problema u nastavnik aktivnostima čime je potvrđen teorijski dio disertacije.

Ključne riječi: Problem pridruživanja, Raspodijeljeni sustavi, Genetski algoritmi, Nastavni procesi, Pridruživanje studenata projektima, Pridruživanje studenata mentoru, Odabir predmeta za upis, Preporučiteljski sustav

Preference based allocation in distributed systems applied on educational processes

Introduction

The allocation problem is the problem of matching two sets of members, where each member has a preferred member from the other set. This problem is also known as the *two-sided matching algorithm with a preference list*, and is a part of the matching theory. It has application in various real-life problems like school enrolment processes, allocating kidneys to transplant recipients, and even in everyday computer problems like resource allocation in a cloud computing system.

The theory for matching and allocation problem was introduced in 1962 when Gale and Shapley published the article "College admissions and the stability of marriage". An allocation is stable "where no individuals perceive any gains from further trade".

They proposed an algorithm, known as the Gale-Shapley algorithm, or deferred acceptance algorithm, which always produces a stable solution. The described algorithm is applicable for one variation of the allocation problem. During the 1990s, Roth and his colleagues created a version of the algorithm intended to solve the problem of allocating doctors to hospitals. For their work on allocation problems, Shapley and Roth won the Nobel Memorial Prize in Economic Sciences, 2012. This theory is still researched and developed in various allocation problems.

This dissertation describes the new model with a set of algorithms designed to solve allocation problems with preference lists in distributed systems. The model is applied to seven education related problems. In all of them, students are required to create a preference list of the groups in which they want to be placed. Depending on the application, the term "group" may refer to a project, mentor, position, study program, etc.

The new model applies to the allocation problem where: students create incomplete preference lists; the number of available places in groups is limited for each group; master list of ranked students is used; higher-ranked students are given a priority for allocation. The described setting produces a significant number of unallocated students and is different from the one, described by Gale-Shapley, as it requires a different approach.

This dissertation describes two methods. The first one is *the method for ranking students,* which uses fuzzy logic for creating master preference lists and has a direct effect on reducing the number of unallocated students. The second one, *the method for allocating*

unallocated students, uses expert knowledge about other students' preference lists and it directly influences the quality of allocation for higher ranked students.

These methods, in combination with other proposed algorithms in the dissertation, are used as a model for solving the problem of preference-based student allocation. This dissertation also provides a prototype of the proposed model for the software system inside a distributed environment.

Allocation problems in educational processes

This chapter provides examples of allocating students within educational processes. The presented examples are real problems, which were arising during the student allocations each year at the University of Zagreb (i) on the Seminar course more than 500 students need to be allocated to more than 200 projects in groups of two to three students; (ii) more than 580 undergraduate students and more than 550 graduate students need to get a mentor; (iii) student practice program uses allocation algorithm to assign more than 600 students to more than 300 available positions; (iv) allocating more than 100 students to study programs where students are ranked according to their subject and grades; (v) distributing students to groups with fixed time schedule; (vi) enrolment process where over 3,000 students are required to choose courses for the next semester using preference list; (vii) at the Seminar course where up to ten small groups, consisting of two or three students, need to be merged into larger group.

Related work

This chapter provides an overview of current scientific research in the field of allocation problems. Some authors present an algorithm that aims to achieve a satisfactory solution in linear time. Others consider the quality of solutions like the level of student and mentor satisfaction. These researches do not consider giving the advantage to better-ranked students in allocation. Their goal is to achieve better satisfaction while producing the lowest number of unallocated students.

They suggest several methods for allocation of the unallocated students: (i) *second-round* – repeating the allocation process for the unallocated students; (ii) *random selection* – allocating students by random; (iii) *trading* – allowing students to swap groups after initial allocation; (iv) *increasing* – increasing the number of places available in groups or the total number of groups.

The first three suggested methods try to allocate the unallocated students after the initial allocation when the most attractive groups are no longer available. The described methods allocate high-ranking students to less attractive groups.

The fourth one, the method of increasing, requires an increase in the number of groups, but that may not be viable since the university regulates these numbers.

After analysing the described approaches and algorithms, none of the methods suggested by other authors completely fulfilled the needs of the problem presented in this dissertation.

Theoretical background

The two-sided matching problem allocates members to two sides. Members from one side can have preference over members from the other side. Allocation uses the *master list* when all members on one side share the same preference list. The preference list is *complete* when it includes all elements from the opposite side. If all elements on the preference list have a unique rank, the list is *strictly ordered*.

The allocation is considered to be a simple problem that can be solved in linear time using the greedy algorithm for a strictly ordered master preference list. The algorithm allocates the student from the top of the master list to the first available group from the student's preference list. This continues until all students are allocated.

A similar algorithm is applicable even when the master preference list is not strictly ordered. In that case, the algorithm generates a different master preference list for every possible combination of student rank order. As the number of same ranked members rises, the number of possible allocations grows significantly. The number of allocations is the greatest when all students share the same rank – this generates N! solutions. When there is more than one *stable* solution, an evaluation mechanism chooses the best allocation.

Methods for allocating unallocated students

This chapter introduces two new methods for allocating unallocated students: (i) *the new method for ranking students*; (ii) *the new method for allocating unallocated students*. These methods aim to achieve a reduction of unallocated students and to increase the quality of the allocation for higher-ranked students.

The new method for ranking students uses expert knowledge to create the master preference list. This dissertation analyses several types of expert knowledge: similarity in

student's grades; top-ranking student in the group; group interest; average group score; the name of a student, mentor, or a group.

The most important one is the expert knowledge of similarity in two student's grades. The new method introduces the *fuzzy approach* for ranking students in the master preference list. It assumes that two students have the same grades even with a small difference in grades. Ignoring the slight difference allows the algorithm to change the order of these two students in the master preference list and to generate more master preference lists, thus creating more allocation solutions. The *distance* parameter - one of the input parameters of the algorithm – is the maximum allowed difference between the grades of two students. The maximum value (3.0) for the distance is the difference between the highest (5.0) and lowest (2.0) passing grade. When the distance is minimum (0.0), all students are ranked based only on their average study grade. When the distance is maximum, their study grade does not influence the ranking.

The new method for allocating unallocated students uses knowledge hidden in other students' preference lists. This method creates a new list, called *the extended preference list*, which contains the ordered list of groups that the student did not select. Groups are ordered in a way the student themselves would most likely order. *The extended preference list* is calculated from other student's preference lists, based on the *preference list similarity (PS)* and *group order factor on their preference list (PF)*. Two students' preference lists are more similar to each other when they have the same groups ranked higher. The method ranks a group in the extended preference list higher when the group is: (i) ranked higher on a similar preference list; (ii) ranked higher on a shorter preference list.

Methods for ranking students and allocating unallocated students are integrated into the new proposed model represented as a (i) *mandatory single-allocation algorithm with the master preference list*. It is used as a base for these new algorithms: (ii) *multi-allocation algorithm with master preference list*; (iii) *multi-allocation algorithm with master preference list and multi-layered preference list; (iv) single-allocation algorithm with grouped elements*.

The model and algorithms combine new and improved existing algorithms for finding the best allocation: master preference list generation algorithm, extended preference list generation algorithm, student allocation algorithm, and allocation evaluation algorithm. Producing and evaluating all possible master preference lists may require a lot of power and time.

The model uses a genetic algorithm to find the best allocation, where the master preference list represents a single chromosome with a vector of average study grades. All solutions generated by this model are stable allocations and valid solutions.

The model for the preference-based allocation in distributed systems

This chapter describes the execution of the model in a distributed systems environment. It uses an unlimited number of agents that execute the model in parallel while producing the best solution. The server prepares data and parameters to execute the algorithm, while each agent runs a genetic algorithm that determines its own best solution. While generating the solution, all agents are communicating with the server in two-way communication. They are finding better solutions and sending them to the server simultaneously. They are also receiving the current network's best solution, which they may use to improve the current one. Agents can be completely distributed on an unlimited number of servers, and in the current implementation, they are communicating through the database. However, the model abstracts the communication layer and can be changed to another with no influence on the agent.

When all agents are done with calculations, the final solution is evaluated and presented to the user. The model allows users to run the calculation indefinitely while producing the current best solution for all agents. The number of agents can be increased and decreased during the calculation period, with no influence on the final result.

Prototype

This chapter describes the implementation of the preference-based allocation in the distributed systems. The prototype is written in C# and works on Microsoft Windows, macOS, and GNU/Linux platforms. There are three ways to execute the prototype: (i) standalone application, (ii) server application, and (iii) client application.

Input values are stored in structures: *_students* as the hash list of input data of students and their choices, *_mentors* as hash list of input data of groups, and *SingleSolutionSimple* as a one-dimensional field of solution structure.

A key in the *_students* structure is a JMBAG while the value is represented by the class *StudentsClass* with variables: JMBAG (string), Grade (double), StudentId (uint), preference and extended preference list.

A key in the *_mentors* structure is a group ID, while the value is represented by the *MentorClass* with variables*:* Code (string), FirstName (string), LastName (string), Unit (string), and MentorId (uint).

SingleSolutionSimple represents the chromosome for a genetic algorithm and consists of: (i) StudentOrder (StudentIdGrade[]) - a one-dimensional field consisting of ranked students with their grades and group assignments; (ii) SolutionGrade (float) - the overall grade of the solution.

Structure *StudentIdGrade* consist of elements: StudentId (uint), StudentGrade (int), MentorId (uint), AssigmentType (uint), AssigmentOrder (uint), RandomId (uint).

The prototype has been optimized for minimal memory consumption as it uses about 13MB of memory to calculate allocations for more than 500 students, in more than 200 groups, and with more than 3,000 student selections. The running time of the prototype for the specified number of students, groups, and selections is 200 iterations / second on a standard PC with i5 processor.

Application of methods and the model on educational processes

This chapter describes the implementation of the new methods and model in educational processes on seven real allocation problems at the University of Zagreb.

The *mandatory single-allocation algorithm with a master preference list* is used to allocate students to a project in the Seminar course at the Faculty of Electrical Engineering and Computing (FER). Each student creates their preference list by ranking projects on the said list. The best result is the one in which the least number of students remains unallocated, while preference in project allocation is given to higher-ranking students.

The same algorithm is used in student-mentor allocation. Students create their preference lists of mentors by ranking them. The mentors approve the students, after which, the strict-allocation algorithm is used for all students whose mentors did not reject or approve them. The extended preference list and random allocation are not used in this algorithm. The human supervisor allocates students who remain unallocated. The extended preference list can be used as a help for the human supervisor.

In the implementation of student allocation to the study programs, a master preference list is created by ordering students by their final score based on their study and other achievements.

The multi-allocation algorithm with the master preference list is used in allocating the students to groups with fixed time regarding their schedule. In this scenario, all preference lists are generated automatically. Students are ranked on the master preference list based on the number of groups in which they can participate.

Student preference lists are automatically generated as a list of groups in which student can enrol and they are sorted by the number of students that can participate. The group with the highest ranking is the one which allows for the shortest list of students.

Another application of the algorithm is in the internship program. Students make a preference list of available positions, and after this process, companies confirm applied students for their positions.

The multi-allocation algorithm with the master preference list and multi-layered preference list is used in allocating students to courses in the enrolment process on FER. Course enrolment may depend on the prerequisite course. If a student passed the prerequisite course, they could be enroled. Courses are grouped, and the student creates an incomplete preference list for each of them as well as a preference list of the groups. Based on the student current achievement and multi-layered preference list, the algorithm allocates students to courses.

The single-allocation algorithm with grouped elements is implemented in the Seminar course, where groups of students are grouped into larger groups.

Evaluation of the proposed model and methods

Evaluation of the proposed model and methods was done using the real data from the academic year 2018/2019. The experiments were repeated using the real data from academic years: 2015/2016, 2016/2017, and 2017/2018 in order to verify their integrity.

In the academic year 2018/2019, the Seminar course had 519 students enroled, and 244 projects were available. The student's preference list was obtained using the web application form. The student's grades used for the ranking and the list of available projects were retrieved from the student information system. The collected data was used as an input for the prototype.

The Method for ranking students uses expert knowledge for creating the master preference list of students. A distance parameter is used to determine how far two students' grades can be apart in order to be still considered equal. With the increase in distance from 0 to 0.1, the number of unallocated students decreased by 4.2%, from 36% to 31.8%. When the distance was increased to a maximum of 3, the number of unallocated students decreased by 14.4% (from 36% to 21.6%). A similar result was reproduced when the input data was from another academic year.

The Seminar course is required to give an advantage for group selection to students with better grades. When the distance is set to maximum, the grades are not used in ranking. This contradicts the Seminar course requirement to give an advantage for group selection to students

with better grades. Fulfilling this requirement is achieved by selecting the *distance* that does not influence the top 10% ranked students, but still decreases the number of unallocated students. The result of this analysis shows that when the *distance* is 0.5, it influences the selection of only 2% of the top students (0.2% of all students), which was acceptable since the total number of unallocated students was lowered by 10% for all students. When the distance is 1, there is one student (2% of the top students) that is allocated to a group higher on their preference list.

The *method for allocating unallocated students* uses knowledge data from the preference lists of other students and introduces the extended preference list. The extended preference list does not influence the number of unallocated students but changes the way they are allocated. By experimentally allocating students into groups, using an extended preference list, about 90% of the students are allocated, and only about 10% of the students are allocated by random selection. When this new method is not used, up to 35% of the students remain unallocated.

The proposed model is compared to the *Deferred Acceptance (DA) algorithm*, also known as the Gale–Shapley algorithm based on the: (i) *quality of allocations*; (ii) *group position on the student preference list*.

When compared in terms of *quality of allocations*, the proposed model was modified not to use the extended preference list and random allocation, as they influence the allocation of lower-ranking students, thus making results incomparable. When the distance is set to 0, the results of the proposed model are similar to the DA algorithm. More than 95% of the students are allocated to the same groups as with the DA algorithm. With a small increase in distance of 0.5, the number of unallocated students decreases by about 8%. The proposed model gives the best results when the distance parameter is set to 3.

When the comparison is based on the *group position on the student preference list*, the proposed method produces slightly better results than the DA algorithm. More students are allocated to the groups ranked higher on their preference list. By including an extended preference list and random group allocation in the proposed method, the analysis shows that the proposed method will assign more higher-ranking groups to higher-ranking students than the DA algorithm. The difference in allocation occurs because the DA algorithm allocates unallocated students randomly after completing the allocation algorithm. The proposed method allocates a student using its extended preference list and then randomly – this allocation is made before the next student is allocated.

The allocation algorithm of grouped elements uses the method for ranking students when creating the master preference list using expert data knowledge. The solution is the allocation in which each large group consists of an equal proportion of students with excellent, average, and below-average grades. Allocation using expert knowledge of group name, group area, and an average grade of students in a group, resulted in a lot of unbalanced groups. When expert knowledge was based around the student with the best average grades in a group, the allocation produced the most balanced groups.

Conclusion

The allocation problem with the preference list has multiple applications in the educational process, such as the enrolment process, assigning students to mentors and projects, creating groups of students. Several authors are suggesting their approach to solving these allocations. None of them has considered the approach when priority in allocation should be given to students based on additional criteria such as average grades while considering the expert knowledge of grade distance.

This dissertation solves the problem of giving priority to students with higher average grades by using a master preference list of all students in which students are ordered by their average study grade. This list contains all students and is considered complete, but it is not strictly ordered since some of them share the same average grades, therefore, the rank. The other side of the two-sided allocation problem is the preference lists of students. Their lists are incomplete since students do not rank all groups on their lists. A combination of not strictly ordered master preference list of ranked students and the incomplete students' lists leads to the problem of unallocated students.

A new *method for ranking students* was introduced to create a better allocation while still retaining the importance of the master list. The method uses a new fuzzy approach to create a master preference list and, as a result, reduces the number of unallocated students while maintaining the importance of students' rank.

Even with this new method, some students remained unallocated and were allocated to the group at random. In order to solve this problem, a new *method for allocating unallocated students* is introduced – the extended preference list. This is a list of groups that are selected and ordered in a way the student would most likely select and order themselves but have not. The validity of this list was checked by shortening an existing student's preference list, which was then used for the extended preference list generation. Both the generated extended

preference list and the original preference list were compared, producing significant similarities. This method allows better-ranked students to allocate to more popular groups, unlike other algorithms suggested before, where these better-ranking students were allocated to the least popular groups.

These methods are implemented as algorithms, and they represent the central part of the model for the preference-based allocation in distributed systems. The main algorithm, *the strict-allocation algorithm with master preference list and mandatory allocation*, uses both of these two proposed methods. The evaluation of the algorithm, based on the students' allocation on the Seminar course, shows the improvement in quality. It has increased by up to 25% using the abovementioned methods. The model was also introduced and developed as a prototype in a distributed system environment. When the agent architecture is used, it produces much better results compared to the solution calculated by just one computer. A prototype of solving the problem of allocation in the teaching processes was created. It implements the described model with methods in a distributed system. This prototype confirmed the improvement that is achieved in students' allocation groups.

SADRŽAJ

1. Uvod ... 1

2. Pregled odabranih problema pridruživanja u nastavnim procesima ... 5

2.1. Pridruživanje studenta projektu ... 5

2.2. Pridruživanje studenta mentoru ... 5

2.3. Pridruživanje studenta na studentske prakse ... 6

2.4. Odabir smjera ... 6

2.5. Pridruživanje studenta grupi s određenim terminima održavanja ... 6

2.6. Odabir predmeta za upis višeslojnom prioritetnom listom ... 7

2.7. Grupiranje grupa studenata ... 7

3. Pregled prethodnih istraživanja ... 8

4. Teorijska pozadina ... 12

4.1. Problem pridruživanja ... 12

4.2. Problem pridruživanja uporabom glavnog popisa ... 13

4.3. Problem pridruživanja s ograničenjima u broju slobodnih mjesta ... 14

5. Metode rješavanja problema pridruživanja ... 16

5.1. Metoda odabira redoslijeda elemenata ... 16

5.1.1. Ekspertno znanje o sličnosti ocjena studenta ... 17

5.1.2. Ekspertno znanje o ocjeni najuspješnijeg studenta u grupi ... 17

5.1.3. Ekspertno znanje o području grupe ... 18

5.1.4. Ekspertno znanje o prosječnoj ocjeni grupe ... 18

5.1.5. Ekspertno znanje o nazivu studenata, projekta ili voditelja grupe ... 18

5.2. Metoda pridruživanja nepridruženih elemenata ... 19

5.2.1. Sličnost prioritetne liste ... 20

5.2.2. Važnost grupe ... 22

5.2.3. Proširena prioritetna lista ... 23

5.3. Primjena metoda u algoritmima pridruživanja s prioritetnim listama ... 24

5.3.1. Algoritam strogog jednostrukog pridruživanja s glavnim popisom ... 24

5.3.2. Algoritam višestrukog pridruživanja s glavnim popisom ... 46

5.3.3. Algoritam višestrukog pridruživanja s glavnim popisom i višeslojnom prioritetnom listom 52
5.3.4. Algoritam jednostrukog pridruživanja grupiranih elemenata 57

6. Model raspodijeljenog rješavanja problema pridruživanja 60

7. Prototip programskog sustava 66

7.1. Načini rada prototipa programskog sustava 66

7.2. Struktura podataka 67

7.2.1. Struktura _students 67
7.2.2. Struktura _*mentors* 68
7.2.3. Polje rješenja 69

7.3. Optimizacija brzine izvođenja prototipne izvedbe algoritma 70

7.3.1. Prilagodba organizacije podataka u memoriji 70

7.4. Programska izvedba prototipa 70

7.4.1. Izvedba glavnog algoritma 70
7.4.2. Izvedba izrade proširene prioritetne liste 73
7.4.3. Izvedba izrade glavnog popisa studenata 74
7.4.4. Izvedba algoritma novog rješenja 76

7.5. Izvršavanje prototipa programskog sustava 78

7.5.1. Primjer ulaznih datoteka 78
7.5.2. Proširena prioritetna lista 80
7.5.3. Rezultat izvođenja – pridruživanje studenata grupama 81

7.6. Prikaz zaslona prilikom izvršavanja prototipa 81

8. Primjena metoda i modela u nastavnim procesima 84

8.1. Pridruživanje studenta projektu 84

8.2. Pridruživanje studenta mentoru 86

8.3. Odabir smjera 88

8.4. Pridruživanje studenata u grupe s određenim terminima održavanja 89

8.5. Pridruživanje studenata na studentske prakse 92

8.6. Odabir predmeta za upis s višeslojnom prioritetnom listom 95

8.7. Grupiranje grupa studenata 101

9. Ocjena predloženih metoda i modela 102

9.1. Metoda odabira redoslijeda elemenata 102

9.2. Metoda pridruživanja nepridruženih elemenata 108

9.3. Primjena metoda u algoritmima 111

9.4. Usporedba predloženih algoritama s Algoritmom odgođenog prihvaćanja 114

9.5. Algoritam jednostrukog pridruživanja grupiranih elemenata 117

9.5.1. Pridruživanje temeljem ekspertnog znanja o nazivu projekta 118

9.5.2. Pridruživanje temeljem ekspertnog znanja o području grupe 118

9.5.3. Pridruživanje temeljem ekspertnog znanja o prosječnoj ocjeni grupe 119

9.5.4. Pridruživanje temeljem ekspertnog znanja o ocjeni najuspješnijeg studenta u grupi 120

9.6. Model raspodijeljenog rješavanja problema pridruživanja 121

10. Zaključak 124

Literatura 127

Popis slika 130

Popis tablica 133

Popis algoritama 134

Popis programskog kôda 135

Pojmovnik 136

Životopis 137

Biography 140

1. UVOD

Problem pridruživanja (eng. *Allocation problem*) rješava uparivanje elemenata iz dva skupa, gdje svaki od elemenata bira one iz drugih skupina. Ovaj problem svrstava se u teoriju uparivanja (eng. *Matching theory*) a poznat je i kao dvostrani algoritam uparivanja s prioritetnim listama (eng. *Two-sided matching algorithm with a preference list*). Primjenjuje se za rješavanje raznih problema među kojima su poznatiji: odabir obrazovne ustanove [1]–[4], dodjela bubrega primatelju transplantacije [5]–[7], pridruživanje studenata medicine u bolnice [8], [9], ali je prisutan i u računarskom problemu alokacije resursa u oblaku [10]–[12].

Teorija uparivanja i problema pridruživanja uvedena je 1962., kada su Gale i Shapley objavili članak „College admissions and the stability of marriage“ [13]. U članku opisuju problem općenitog stabilnog pridruživanja (eng. *Stable allocation problem on a general level*) uporabom primjera odabira i dodjele bračnog partnera. U opisanom pridruživanju svaka žena i muškarac izrađuje prioritetnu listu svih osoba suprotnog spola. Pridruživanje se temelji na prioritetnim listama, a naziva se *stabilnim* kada više nitko ne primjećuje dobit od daljnje trgovine (eng. *where no individuals perceive any gains from further trade*) [14].

U svojem članku oni predlažu i algoritam koji uvijek rezultira stabilnim rješenjem. Algoritam je poznat i pod nazivom Gale-Shapleyev algoritam ili algoritam odgođenog prihvaćanja (eng. *deferred acceptance algorithm*). Predloženi algoritam primjenjiv je u rješavanju problema pridruživanja u idealnim uvjetima u kojima su svi parametri idealni. Na primjer, oba skupa sadrže jednak broj elemenata i svaki od elemenata na svojoj prioritetnoj listi ima striktno poredane sve elemente iz drugog skupa. Za rješavanje problema pridruživanja, u kojem je samo jedan od parametra različit od zahtijevanih, izvorni predloženi algoritam često se ne može upotrijebiti.

Za potrebe pridruživanja liječnika u bolnice i autori su razvili drugačiju inačicu algoritma. U tom razvoju, 1990-ih im se pridružuje Roth s kolegama. Za cjelokupni doprinos u području pridruživanja, 2012. godine, Shapley i Roth [14] dobivaju Nobelovu memorijalnu nagradu za ekonomske znanosti. Nagrada im je dodijeljena za "teoriju stabilnih pridruživanja i praksu tržišnog dizajna" (eng. „for the theory of stable allocations and the practice of market design”). Danas se, za rješavanje problema pridruživanja, još uvijek istražuju algoritmi i metode rješavanja koji će dati bolje rezultate.

Ovaj rad opisuje skup novih algoritama namijenjenih rješavanju problema pridruživanja s prioritetnim listama u raspodijeljenim sustavima. Algoritmi su primijenjeni na sedam

problema vezanih uz nastavne procese, a u kojima se studenti pridružuju temeljem prioritetnih lista. Navedeni nastavni procesi su: pridruživanje studenta projektu, pridruživanje studenta mentoru, pridruživanje studenta na studentske prakse, odabir smjera, pridruživanje studenta u grupu s određenim terminom održavanja, odabir predmeta za upis višeslojnom prioritetnom listom te grupiranje grupa studenata.

Kod svih problema pridruživanja studenti izrađuju prioritetnu listu grupa u koje žele biti pridruženi. Ovisno o nastavnom procesu na kojem se student pridružuje, pojam grupe može označavati projekt, mentora, poziciju, smjer i slično. Studenti na svojoj prioritetnoj listi ne moraju poredati sve grupe. Takve, djelomične, prioritetne liste smatraju se nepotpunima. Sve grupe imaju gornje ograničenje u broju studenata koji im se mogu pridružiti – zbog čega sve studente nije moguće pridružiti njihovom prvom odabiru. Prednost prilikom pridruživanja studenata određuje se glavnim popisom studenata (eng. *master list of ranked students*). Na popisu se studenti rangiraju prema bodovima koji su iskazani za svakog studenta. Prednost odabira ima student s većim brojem bodova. Ako dva studenta imaju jednake bodove tada su ravnopravni u odabiru.

Posljedica nepotpune prioritetne liste studenata i ograničenog broja mjesta je nemogućnost pridruživanja svih studenata isključivo putem njihove prioritetne liste, odnosno dolazi do pojave *nepridruženih studenata*. Problem u kojem se koristi glavni popis rangiranih studenata različit je od onog koji opisuju Gale i Shapley u svojem radu te zahtjeva drugačiji pristup.

U ovoj disertaciji opisane su dvije metode koje primjenom ekspertnog znanja o studentima, nastavnicima i samom nastavnom procesu smanjuju broj nepridruženih studenata ili daju bolje rezultate u pridruživanju studenata. Prva metoda odabira redoslijeda studenata koristi pristup neizrazite logike temeljem ekspertnih znanja o studentima, te izravno utječe na smanjenje broja nepridruženih studenata. Druga metoda pridruživanja nepridruženih studenata, temeljem zaključaka izvedenih iz konteksta odabira, izravno utječe na kvalitetnije pridruživanje boljih studenata.

Obje metode implementirane su vlastitim algoritmima, a u kombinaciji s drugim predloženim algoritmima u ovoj disertaciji koriste se kao cjelovito rješenje za rješavanje problema pridruživanja studenata s prioritetnim listama u nastavnim procesima. Predloženi algoritmi implementirani su u obliku prototipa programskog sustava u raspodijeljenom okruženju.

Rad je strukturiran na sljedeći način. U prvom, uvodnom poglavlju „*1. Uvod*“, opisani su područje i ciljevi istraživanja te sadržaj rada.

Drugo poglavlje „*2. Pregled odabranih problema pridruživanja u nastavnim procesima*“ daje pregled sedam problema pridruživanja nastavnih procesa na Sveučilištu u Zagrebu, a na koje se može primijeniti navedeno istraživanje.

U trećem poglavlju „*3. Pregled prethodnih istraživanja*“ dan je pregled postojećih istraživanja problema pridruživanja s prioritetnim listama u nastavnim procesima. U njima se posebno ističe način rješavanja problema nepridruženih studenata s obzirom na to da se ovaj rad najviše bavi tom tematikom.

Četvrto poglavlje „*4. Teorijska pozadina*“ obuhvaća uvod u problem i teoriju vezanu uz rješavanje problema pridruživanja.

Peto poglavlje „*5. Metode rješavanja problema pridruživanja u nastavnim procesima*“ predlaže metodu odabira redoslijeda elemenata i metodu pridruživanja nepridruženih elemenata. *Metoda odabira redoslijeda elemenata* koristi ekspertno znanje o ocjenama studenata kako bi se kreiralo veći broj glavnih popisa studenata a time omogućilo veći prostor pretrage boljeg rješenja nepridruženih studenata. *Metoda pridruživanja nepridruženih elemenata* u nastavnim procesima temeljem zaključaka izvedenih iz konteksta odabira koristi se za izradu proširene prioritetne liste studenata. Time se želi postići kvalitetnije pridruživanje za nepridružene studente koji su bolje rangirani. Metode su ugrađene unutar četiri nova predložena algoritma koji rješavaju problem pridruživanja u nastavnim procesima, a temeljeni algoritam je *Algoritam strogog jednostrukog pridruživanja s glavnim popisom*. On problem nepridruženih studenata rješava kombinacijom pet algoritama: glavni genetski algoritam, algoritam izrade glavnog popisa, algoritam izrade proširene prioritetne liste, algoritam jednostrukog pridruživanja i algoritam ocjene jednostrukog pridruživanja. Algoritam izrade glavnog popisa koristi novu metodu odabira redoslijeda elemenata pri rješavanju problema pridruživanja s prioritetnim listama uporabom ekspertnog znanja. Algoritam izrade proširene prioritetne liste koristi novu metodu pridruživanja nepridruženih elemenata temeljem zaključaka izvedenih iz konteksta odabira. Drugi „*Algoritam višestrukog pridruživanja s glavnim popisom*“ i treći „*Algoritam višestrukog pridruživanja s glavnim popisom i višeslojnom prioritetnom listom*“ algoritmi temeljeni su na prvom, ali su prilagođeni rješavanju problema s drugačije zadanim postavkama problema pridruživanja. Četvrti algoritam „*Algoritam jednostrukog pridruživanja grupiranih elemenata*“ određuje potrebna ekspertna znanja metode odabira redoslijeda elemenata.

U šestom poglavlju „*6. Model raspodijeljenog rješavanja problema pridruživanja*“ određuje se model raspodijeljenog rješavanja problema pridruživanja s prioritetnim listama u nastavnim procesima.

Prototip programskog sustava prikazan je u sedmom poglavlju „*7. Prototip programskog sustava*“. Dan je pregled izvedbe programskog kôda za rješavanje problema pridruživanja, na primjeru *Algoritma strogog jednostrukog pridruživanja s glavnim popisom*. U navedenom se prototipu koriste nove metode i model raspodijeljenog rješavanja problema pridruživanja.

U osmom poglavlju „*8. Primjena metoda i modela u nastavnim procesima*“ prikazana je implementacija na odabranim primjerima pridruživanja u nastavnim procesima.

Ocjena novih metoda i modela dana je u devetom poglavlju „*9. Ocjena predloženih metoda i modela*“. Analizira se utjecaj metode odabira redoslijeda elemenata na broj nepridruženih studenata te utjecaj metode pridruživanja nepridruženih elemenata na kvalitetu pridruživanja kod bolje rangiranih studenata. Rezultat izvođenja jednostrukog pridruživanja uspoređen je s algoritmom odgođenog prihvaćanja. Za model raspodijeljenog rješavanja problema napravljeno je usporedba najboljeg rješenja koje je dobiveno korištenjem unutar i izvan raspodijeljenog okruženja.

Zadnje, deseto, poglavlje „*10. Zaključak*“ daje ukupni pregled doprinosa disertacije.

2. PREGLED ODABRANIH PROBLEMA PRIDRUŽIVANJA U NASTAVNIM PROCESIMA

U ovom poglavlju dani su primjeri pridruživanja u nastavnim procesima. Za sve navedene primjere napravljena je izvedba procesa pridruživanja na odabranim problemima nastavnih procesa Sveučilišta u Zagrebu.

2.1. Pridruživanje studenta projektu

Na pojedinim predmetima dio aktivnosti izvodi se u projektnim grupama. Za potrebe nastavnih procesa studente je potrebno pridružiti u odgovarajuće grupe. Problem pridruživanja studenata u grupe prepoznat je i u drugim radovima [15]–[20]. Jedan student može biti član samo jedne projektne grupe, a broj slobodnih mjesta u svakoj od projektnih grupa je ograničen. Studenti se u grupe pridružuju ravnomjerno. Svoj odabir studenti iskazuju izradom prioritetne liste projekata na kojima žele sudjelovati. Studenti nemaju ograničenje na broj projekata koje mogu izabrati. Za pridruživanje studenata upotrebljava se prioritetna lista studenata na kojoj su navedeni svi studenti s njihovim bodovima. Studenti s većim brojem bodova ostvaruju prednost pri pridruživanju. Studenti koji imaju jednak broj bodova imaju jednaki prioritet prilikom pridruživanja.

Na primjeru predmeta „Seminar“ koji se izvodi na Fakultetu elektrotehnike i računarstva, više od 600 studenata pridružuje se na više od 200 projekata. Svaka projektna grupa ima približno jednak broj studenata, odnosno dva do tri studenta. Studenti najčešće na svoju prioritetnu listu stavljaju između 5 i 10 projekata. Prioritetna lista studenata na ovom predmetu određena je prosječnom ocjenom studija.

2.2. Pridruživanje studenta mentoru

Odabir mentora studenti obavljaju izradom prioritetne liste. Za potrebe nastavnih procesa potrebno je studentu pridružiti mentora koji će ga voditi kroz završetak studija. Studentu se pridružuje jedan mentor. Mentori imaju gornje ograničenje u broju studenata koje mogu prihvatiti. Ne postoji najmanji broj studenata koje mentor mora imati. Problem pridruživanja student-mentor poznat je problem pridruživanja [21]–[24].

Na Fakultetu elektrotehnike i računarstva, svake godine potrebno je odrediti mentora za više od 580 studenata na preddiplomskom studiju te više od 550 studenata na diplomskom studiju. Na svakoj od ovih razina studija dostupno je više od 150 mentora.

2.3. Pridruživanje studenta na studentske prakse

Studenti biraju željene pozicije za obavljanje studentske prakse prioritetnom listom. Svaka pozicija ima ograničen broj studenata koji na njoj mogu sudjelovati. Studenta se pridružuje samo na jednu od njih. Ako se studenta ne može pridružiti niti na jednu poziciju s njegove prioritetne liste, pridruživanje neće biti obavljeno.

Na Fakultetu elektrotehnike i računarstva, više od 2.500 studenata preddiplomskih i diplomskih studija, mogu se prijaviti za studentsku praksu koju obavljaju u tvrtkama. U studentskim praksama sudjeluje više od 600 studenata koji se prijavljuju na više od 300 pozicija.

2.4. Odabir smjera

Studenti odabiru smjer koji žele upisati putem potpune prioritetne liste. Za potrebe nastavnih procesa studente je potrebno pridružiti na različite smjerove. Svaki smjer ima zadano gornje ograničenje studenata koji se mogu upisati. Ukupan broj slobodnih mjesta na svim smjerovima jednak je broju studenata.

Na vojnim studijskim programima „Vojno inženjerstvo“ i „Vojno vođenje i upravljanje“ Sveučilišta u Zagrebu, studenti prilikom upisa mogu iskazati želje o njihovom prioritetu upisa pojedinog smjera. Ovisno o godini i dosadašnjem usmjerenju, studenti mogu rangirati između dva i pet usmjerenja. Smjerovi se studentima pridružuju temeljem ranga dobivenog zbrojem bodova koji su rezultat njihovih akademskih i vojnih uspjeha.

2.5. Pridruživanje studenta grupi s određenim terminima održavanja

Problem pridruživanja studenata grupi najčešće je povezan sa izradom rasporeda sati kojim se bave mnogi autori [25]–[27]. Ovaj problem se bavi pridruživanjem studenata u grupe, kada već imaju definiran raspored i treba ih pridružiti u dodatnu grupu.

Studenti se u sklopu nastavnih aktivnosti na predmetu pridružuju u grupe koje imaju određeno vrijeme održavanja. Za potrebe nastavnih procesa potrebno je pridružiti studente u grupe tako da student može prisustvovati svim takvim terminima. Svaka grupa ima zadan jedan ili više termina održavanja i gornje ograničenje u broju studenata koje može prihvatiti. Studenti se nalaze u više dodatnih grupa koje imaju zadane svoje termine održavanja. Pridruživanjem studenta u novu grupu ne smije doći do vremenskog preklapanja s dosadašnjim grupama čiji je član.

Na Fakultetu elektrotehnike i računarstva predmeti se sastoje od niza aktivnosti. Studenti su pridruženi na osnovne aktivnosti na predmetu poput predavanja, auditornih i laboratorijskih vježbi. Uz osnovne aktivnosti nastavnici na svakom predmetu mogu kreirati i dodatne aktivnosti poput predaja projekata ili dodatnih vježbi. Na njima je studente potrebno pridružiti u grupe. Svaka aktivnost ima više grupa te se za svaku od njih određuje termin održavanja. Studenata na predmetu može biti i više od tisuću, a broj grupa u koje ih nastavnici pridružuju može biti veći od pedeset.

2.6. Odabir predmeta za upis višeslojnom prioritetnom listom

Predmete za upis studenti biraju putem višeslojne prioritetne liste koja im omogućava određivanje niza grupa predmeta, poredak i broj željenih i zamjenskih predmeta u svakoj od njih, kao i zadavanje ukupnog broja predmeta za upis. Za potrebe nastavnih procesa potrebno je odrediti kojem studentu će se upisati koji predmet. Svaki od predmeta ima zadano ograničenje upisa najmanjeg i najvećeg broja studenata na predmet. Ako predmet nije odabralo dovoljno studenata, tada se na njemu ne upisuje niti jedan student. Shannon i dr. opisuju upis predmeta putem prioritetne liste [28], ali njihova lista nije višeslojna, što ne odgovara našem slučaju u kojem se studentu želi omogućiti veća fleksibilnost u odabiru .

Na Fakultetu elektrotehnike i računarstva, svakog semestra više od 3.000 studenata upisuje se na više od 200 predmeta. Većina studenata napravi odabir između deset i dvadeset predmeta od čega se upisuje u prosjeku šest predmeta.

2.7. Grupiranje grupa studenata

Dio predmeta je organiziran tako da studenti, u grupama prezentiraju svoj rad drugim studentima. Za potrebe nastavnih procesa potrebno je odrediti nove prezentacijske grupe studenata koje su sačinjene od manjih projektnih grupa studenata. Svaka nova prezentacijska grupa studenata mora se sastojati od studenata koji predstavljaju reprezentativni uzorak svih studenata na predmetu. Odnosno, u podjednakim omjerima trebaju biti zastupljeni studenti s boljim i lošijim ocjenama. Pri tome, studenti iz projektne grupe su unaprijed određeni, nije ih moguće mijenjati, te svi zajedno moraju prezentirati svoj rad iz projektne grupe. Zbog toga svi studenti iz jedne projektne grupe moraju biti u istoj prezentacijskoj grupi [29].

Na predmetu „Seminar“ na Fakultetu elektrotehnike i računarstva ukupno je upisano više od 600 studenata. Studenti su pridruženi u projektne grupe s dva do tri studenta, a svoj rad trebaju prezentirati u skupini od tridesetak studenata.

3. PREGLED PRETHODNIH ISTRAŽIVANJA

Problem pridruživanja pojavljuje se u različitim primjerima u obrazovnom sustavu. Neki od poznatih su: odabir obrazovne ustanove [1]–[4], određivanje mentora studentima [21] kao i pridruživanje studenata grupama ili projektima [30]–[32] – problem koji se najviše analizira i u ovom radu.

Svi ovi problemi različite su varijante problema pridruživanja čiju teoriju uvode Gale i Shapley 1962. godine [13]. Teo i Ho [33], 1998. godine, opisuju jedan od prvih računalnih procesa za pridruživanje studenata. Svoj proces implementiraju na završnoj godini studija na preddiplomskom studiju elektrotehnike na Nanyang Technological University, Singapur. U njihovom slučaju studente grupiraju u parove koji djeluju kao jedna nedjeljiva cjelina. U cilju pridruživanja, svaki od parova rangira točno deset grupa. Ukupni broj dostupnih grupa značajno je veći od broja studentskih parova. Algoritam koji predlažu pokušava pronaći što više pridruživanja čime bi preostalo što manje nepridruženih studenata. Kao rezultat njihovog algoritma oko 12% studenata (40 od 330) ostaje nepridruženo te za njih ponavljaju postupak u novom, drugom krugu. U procesu se ne upotrebljavaju ekspertna znanja niti prethodna postignuća ili ocjene studenata. Cijeli postupak odvija se na papirnatim obrascima koji se potom skeniraju i obrađuju automatizirano.

Njihov pristup nije primjenjiv za probleme pridruživanja u ovoj disertaciji, jer je jedna od važnih postavki da studenti s boljom prosječnom ocjenom studija imaju prednost pri pridruživanju. Pristup problemu nepridruženih studenata u njihovom radu je da se cijeli postupak ponovi samo za nepridružene studente, u drugom krugu. Glavni nedostatak ove metode je što su preostale samo manje popularne grupe. Time je moguće da studenti koji imaju izvrsne ocjene, a nisu uspjeli biti pridruženi grupi sa svoje prioritetne liste zbog čega ostaju nepridruženi, u drugom krugu biraju između manje popularnih grupa. Ovaj rad pokušava riješiti problem pridruživanja u kojem se boljim studentima žele pridružiti popularnije grupe. Zbog toga ova metoda rješavanja nepridruženih studenata nije razmatrana u ovoj disertaciji.

Kasnije, 2003. godine, Anwar i Bahaj [34] pokušavaju riješiti pridruživanje studenata projektima tako što osiguravaju više mjesta na projektima nego što ima studenata. Svaki projekt ima određenu gornju granicu broja studenata kojima može biti pridružen, ali nema određenu minimalnu granicu. U njihovom slučaju prihvatljivo je da neke projekte ne odabere niti jedan student i da stoga imaju projekte bez studenata. Cilj njihovog opisanog algoritma je pridružiti sve studente na projekte, pri čemu se koristi najmanji potreban broj grupa. Da bi to riješili, predstavili su algoritam s cjelobrojnim programiranjem (eng. *Integer programming*). Pomoću

opisanog pristupa uspješno su pridružili sve studente u samo jednom krugu. U svom zaključku, predlažu tri strategije za rješavanje problema nepridruženih studenata koji se mogu pojaviti zbog različitih uvjeta prilikom rješavanja pridruživanja. Prva povećanje ukupnog broja grupa. Drugo povećanje broja dostupnih mjesta u svakom projektu. Treće kreiranje potpune prioritetne liste tako da se svi neizabrani projekti stave na kraj studentove prioritetne liste.

Iz istih razloga kao i kod Tea i Hoa, niti ovo rješenje ne odgovara za rješavanje problema navedenih u ovoj disertaciji. Metode koje predlažu za rješavanje nepridruženih studenata, također nije moguće provesti ni zbog drugih zadanih ograničenja. U primjeru pridruživanja studenata projektu, njihova prva predložena metoda nije provediva, jer broj grupa ovisi o broju dostupnih nastavnika. Druga metoda u kojoj se predlaže povećanje broja slobodnih mjesta u slučaju pridruživanja studenata projektu također nije primjenjiva, jer se studenti moraju ravnomjerno pridružiti na projekte, zbog čega svi projekti imaju jednak broj slobodnih mjesta, a taj broj nije moguće povećati. Treća metoda ne osigurava da bolji, a nepridruženi, studenti budu pridruženi kvalitetnijim projektima, što također ne odgovara rješavanju problema kojim se bavi ovaj rad. Također, navedene metode nisu primjenjive niti u drugim primjerima pridruživanja u nastavnim procesima opisanim ovom disertacijom.

Calvo-Serrano i dr. [35], 2017. godine, predstavljaju sličan problem kao Anwar i Bahaj [34]. Oni predlažu rješenje temeljeno na zadovoljstvu studenata i nastavnika. Zadovoljstvo nastavnika je veće ako je broj studenata koje vode manji. Također, zadovoljstvo studenta je veće ako je pridružen nastavniku na višem mjestu na njegovoj prioritetnoj listi. Ovo rješenje predstavlja model mješovitog cjelovitog linearnog programiranja (eng. Mixed-Integer Linear Programming), gdje se od studenata traže dvije liste prioriteta: jednu kojom rangiraju nastavnike, te drugu kojom rangiraju projektne kategorije. U svojem radu opisuju da je lakše naći rješenje koje ispunjava jednu od prioritetnih lista od onog koje ima samo jednu prioritetnu listu. U svojem radu ne uzimaju u obzir rang studenata zbog čega algoritam nije primjenjiv u slučaju opisanom ovim radom.

Tijekom 2007. godine Abraham i dr. [30] predlažu dva algoritma kojima je cilj povećati zadovoljstva studenata i nastavnika. Prvi je orijentiran na studente, a drugi na nastavnike. Oni, kao i Teo i Ho [33], predlažu drugi krug kao rješenje za nepridružene studente. Predloženi algoritam dodatno je poboljšan u radovima koji 2008. opisuju Manlove i O'Malley [36], a dalje ga unaprjeđuju 2012. godine Iwama i dr. [32].

Svi navedeni algoritmi upotrebljavaju drugi krug odabira, koji, kao što je navedeno, ne osigurava da bolji, a nepridruženi, studenti budu pridruženi na popularnije projekte.

Abdulkadiroğlu i dr. [37], 2008. godine, predlažu sustav trgovine kao mehanizam rješavanja problema nepridruženih studenata. Svim studentima koji su ostali nepridruženi projekti se pridružuju slučajnim odabirom. Nakon pridruživanja, omogućava im se razmjena koja im omogućava međusobnu zamjenu grupa. Prednost ovog pristupa je što se studentima kojima je nasumično pridružen projekt pruža prilika za promjenu.

Uporaba razmjene slična je metodi drugog kruga. Iako u razmjeni mogu sudjelovati svi studenti, u stvarnosti će sudjelovati samo oni kojima je projekt nasumično pridružen. Svi ostali već su pridruženi projektu pozicioniranom najviše moguće na njihovoj rang listi i nemaju razlog za razmjenu. Zbog toga, navedeni način također nije odgovarajući za rješavanje problema pridruživanja u nastavnim aktivnostima.

Dosad opisani algoritmi promatraju ograničen gornji broj studenata na projektima. Godine 2016. Goto i dr. [38] predstavljaju algoritam koji rješava varijaciju problema pridruživanja sa zadanom donjom i gornjom granicom slobodnih mjesta. U većini drugih opisanih istraživanja zadana je samo gornja granica. U njihovom slučaju analiziran je slučaj u kojem se rangiraju svi članovi iz drugog skupa, odnosno upotrebljavaju se potpune prioritetne liste.

Određivanjem donje i gornje granice broja studenata koji mogu biti pridruženi na svakom projektu slično je problemu pridruživanja studenata projektima. Ipak njihov algoritam nije primjenjiv u ovom slučaju s obzirom na to da se u svim primjerima pridruživanja koja se rješavaju ovom disertacijom studenti imaju nepotpune prioritetne liste.

U cilju poboljšanja pridruživanja, Coper i Manlove [39], u 2018. godini, predstavljaju 3/2-algoritam aproksimacije, što predstavlja ne-trivijalni nastavak algoritma Hospitals/Residents with Ties (HRT), autora Királyja [40]. Cilj 3/2-algoritma aproksimacije je uvesti novi algoritam pridruživanja koji u linearnom vremenu pronalazi stabilno rješenje. Osnovna namjena ovog algoritma je smanjenje vremena potrebnog za dolazak do rješenja, dok pronalazak najboljeg rješenja nije bitan.

Kao i kod ostalih algoritama, niti ovaj algoritam nije prihvatljiv jer se ovom disertacijom rješava problem nepridruženih studenata tako što se boljim studentima pridružuju kvalitetniji projekti, što nije slučaj navedenog istraživanja.

Za rješavanja problema pridruživanja studenata projektima autori upotrebljavaju niz tehnika: genetski algoritam [12], [27], [28], cjelovito programiranje [20], 3/2 algoritam aproksimacije [25], linearno programiranje miješanih cijelih brojeva [21].

Analizom opisanih pristupa i algoritama, utvrđeno je da nijedna od navedenih metoda ne rješava problem nepridruženih studenata, niti daje prijedlog kvalitetnijeg rješenja nepridruženih studenata. U ovom radu želi se povećati kvaliteta pridruživanja za bolje studente koji bi ostali nepridruženi ili u drugom krugu na izboru imali samo manje popularne grupe. Također želi se smanjiti broj nepridruženih studenata tako da bolji studenti imaju prednosti pri pridruživanju.

Također u ovom radu se razmatra da nove metode budu implementirane unutar modela za raspodijeljeno rješavanje problema pridruživanja.

4. TEORIJSKA POZADINA

Algoritam uparivanja s prioritetnim listama na obje strane pripada skupini problema pridruživanja, kod kojih svaki od elemenata jedne skupine izrađuje svoju prioritetnu listu elemenata iz druge skupine. Prilikom uparivanja elementi iz jedne skupine ne mogu se upariti s elementom unutar iste skupine. Svi elementi mogu rangirati sve ili dio elemenata iz suprotne skupine kreirajući pritom potpunu ili nepotpunu prioritetnu listu.

Na primjeru problema pridruživanja u nastavnim procesima, određene su dvije skupine:

(i) Elementi jedne strane – studenti, koji rangiraju grupe kreirajući svaki svoju različitu prioritetnu listu grupa;

(ii) Elementi druge strane – grupe, kod kojih svaka može imati svoju različitu prioritetnu listu studenata ili isti glavni popis rangiranih studenata.

U primjerima problema analiziranih u ovoj disertaciji, sve grupe imaju istu prioritetnu listu temeljenu na glavnom popisu rangiranih studenata. Na glavnom popisu studenti su najčešće rangirani prema prosječnoj ocjeni studija za predmete koje su do sada položili. Veći prioritet prilikom pridruživanja grupe imaju studenti koji su više rangirani, odnosno koji imaju veću prosječnu ocjenu studija. Studenti s istim ocjenama, jednako su rangirani i time imaju jednaki prioritet prilikom pridruživanja.

4.1. Problem pridruživanja

Teorija pridruživanja [13] opisuje problem stabilnog pridruživanja na primjeru braka kao pojednostavljenog problema pridruživanja. Problem se opisuje kao zadatak pronalaska idealnog bračnog partnera u slučaju jednakog broja muškaraca i žena, pri čemu je svaka osoba rangirala sve potencijalne partnere prema svojim željama.

U svom radu, Roth i dr. [41] predstavljaju tri razlike između ovog pojednostavljenog problema pridruživanja i općeg slučaja pridruživanja. Ove razlike imaju ključnu ulogu u rješavanju problema pridruživanja s obzirom na to da promjena bilo kojeg od navedenih parametara zahtjeva drugačiji pristup rješavanju problema. Prema njima, ovo su ključne razlike u parametrima problema pridruživanja bračnih parova i problemima pridruživanja studenata u nastavnim procesima:

1. U problemu pridruživanja bračnih parova – jedna žena može biti udana za samo jednog muškarca i obratno.

   U problemima pridruživanja u nastavnim procesima – student je pridružen samo jednoj grupi, ali u jednoj grupi može biti više studenata;

2. U problemu pridruživanja bračnih parova – problem je određen jednakim brojem muškaraca i žena.
   U problemima pridruživanja u nastavnim procesima – ukupni broj mjesta u grupama jednak je ili veći od broja studenata;
3. U problemu pridruživanja bračnih parova – rangirani supružnici strogo su poredani.
   U problemima pridruživanja u nastavnim procesima – prioritetne liste ne moraju biti strogo poredane već studenti mogu više grupa staviti na isto mjesto, kao što više studenata može biti jednako rangirano na prioritetnoj listi.

Zbog navedenih razlika rezultat pridruživanja nudi više stabilnih rješenja, čime dovodi u pitanje pristup odabiru najboljeg rješenja. Svaki od problema može imati određene drugačije ciljeve za odabir najboljeg rješenja. Ciljevi mogu biti minimiziranje grupa, traženje što većeg broja pridruženih studenata, itd.

Također, zbog navedenih razlika pojavljuju se donje i gornje ograničenje u broju studenata koji se mogu pridružiti u grupu. Ujedno se time dodaju novi parametri koje rješenje mora zadovoljiti. Goto i dr. [38] u svojem radu prikazuju kako ova ograničenja utječu na pronalaženje stabilnog rješenja.

Iako Roth i dr. u svojem radu opisuju većinu razlika, ostaje još jedna koja je značajna za problem pridruživanja studenata u nastavnim procesima:

4. U problemu pridruživanja bračnih parova – rangiraju se svi supružnici.
   U problemu pridruživanja u nastavnim procesima – studenti ne rangiraju sve grupe. Ovakav popis, u kojem nisu rangirane sve grupe naziva se nepotpunom prioritetnom listom (eng. *incomplete preference list*).

Uporaba nepotpune prioritetne liste znatno povećava broj mogućih stabilnih pridruživanja. Kad postoji više od jednog stabilnog rješenja pridruživanja, ocjena rješenja određuje ono koje se smatra najboljim.

4.2. Problem pridruživanja uporabom glavnog popisa

Problem pridruživanja uporabom glavnog popisa proučavali su Perarch i dr. [42] na slučaju pridruživanja studenata spavaonicama, a detaljno ga, sa svim inačicama, opisuju Irving i dr. [43] u svojem radu. U problemu pridruživanja u nastavnim procesima, koristi se glavni popis prioriteta u kojem se svi studenti rangiraju prema prosječnim ocjenama. Ako svi studenti

imaju različite prosječne ocjene, glavni popis je strogo poredan, a pridruživanje se smatra jednostavnim problemom koje može se riješiti u linearnom vremenu uporabom pohlepnog algoritma (eng. *greedy algorithm*). Način rješavanja strogo poredane prioritetne liste uz nepotpune liste i strogo poredan glavni popis u svojem radu opisuju Irving i dr. [43].

U pohlepnom algoritmu, najbolje rangiranom studentu na glavnom popisu pridružuje se prva odabrana grupa koja ima slobodnih mjesta. Postupak se ponavlja iterativno za sve studente kojima u prethodnim koracima nije pridružena grupa. Ako na studentovoj listi ne postoji grupa koja ima slobodnih mjesta, student ostaje nepridružen.

Pohlepni algoritam može se primijeniti i kada glavni popis nije strogo poredan, odnosno kada su dva studenta jednako rangirana jer imaju jednake prosječne ocjene. Tada se izrade obje inačice glavnog popisa rangiranih studenata; prva u kojoj je prvi student više rangiran, i druga u kojoj je prvi student lošije rangiran. Za svaki od popisa može se primijeniti pohlepni algoritam i ovisno o ocjeni rješenja izabrati bolje od njih. Ovisno o broju studenata koji su jednako rangirani broj mogućih rješenja može biti značajan, a kada bi svi studenti imali jednake ocjene tada bi ukupni broj mogućih inačica bio jednak N!.

Pristup rješavanju problema pridruživanja u kojem se koristi glavni popis rangiranih studenata opisan je u radovima [43][44], a svoj pristup u rješavanju problema pridruživanja u kojima prioritetne liste s jedne ili obje strane nisu striktno poredane, te imaju više rješenja, analiziralo je više autora [30], [31], [33]–[35], [37], [39], [40], [43].

4.3. Problem pridruživanja s ograničenjima u broju slobodnih mjesta

Zbog ograničenog broja slobodnih mjesta u grupama, a s velikim interesom za pojedine grupe, svi studenti ne mogu se pridružiti svojem prvom odabiru, a nekad niti jednom sa svoje prioritetne liste. Kada je gornja granica broja studenata koje je moguće pridružiti u grupu preniska, tada je moguće da čak niti visoko rangirani studenti ne budu pridruženi grupama koje su na početku njihove prioritetne liste. Uz njih, značajno raste i broj studenata koji su niže rangirani na glavnom popisu, a koji, zbog kombinacije malog broja mjesta u grupama i kratkog popisa grupa koje su odabrali ostaju nepridruženi.

Niz do sada spomenutih autora je opisivalo problem pridruživanja kada je određeno samo gornje ograničenje u broju slobodnih mjesta. Goto *i dr.* [38] u svojem radu opisuju problem pridruživanja kada je zadano i donje i gornje ograničenje. Minimalno ograničenje može biti:

(i) Obavezno – ako grupa nakon pridruživanja mora imati najmanje onoliko studenata koliko je zadano donjom granicom. U tom slučaju potrebno je studente iz drugih grupa, gdje ih je više od donje granice, pridružiti u grupe koje imaju manje od potrebnog broja studenata. Time će nakon pridruživanja sve grupe imati zadovoljen minimalni broj studenata. Također, da bi ovo bilo zadovoljeno, ukupni minimalni broj slobodnih mjesta mora biti manji ili jednak od ukupnog broja studenata;

(ii) Neobavezno – ako za grupu nema dovoljno zainteresiranih studenata, grupa može imati nula studenata. U ovom slučaju studenti iz grupe koja ima manje od potrebnog broja studenata se pridružuju u druge grupe. Nakon pridruživanja takva grupa neće imati pridružene studente.

Pristup koji će se koristiti ovisi o samom zahtjevu problema koji se rješava. Za oba pristupa potrebno je odabrati i drugačije algoritme koji ih rješavaju. Broj stabilnih rješenja smanjuje se uvođenjem donje granice, ali se povećava i broj nepridruženih studenata.

Za rješavanje nepridruženih studenata neki od autora predlažu uporabu drugog kruga odabira [30], [32], [36], ali i mehanizam razmjene [37].

Navedeni algoritmi nisu primjenjivi u ovoj disertaciji s obzirom na to da je cilj rješavanja problema nepridruženih studenata u ovoj disertaciji osigurati prednost u odabiru i pridruživanju grupa bolje rangiranim studentima.

5. METODE RJEŠAVANJA PROBLEMA PRIDRUŽIVANJA

U ovoj disertaciji predlažu se dvije nove metode za rješavanje problema nepridruženih studenata:

- *Metoda odabira redoslijeda elemenata* pri rješavanju problema pridruživanja s prioritetnim listama u nastavnim procesima uporabom ekspertnog znanja;
- *Metoda za pridruživanje nepridruženih elemenata* u nastavnim procesima temeljem zaključaka izvedenih iz konteksta odabira.

Korištenjem ovih metoda želi se smanjiti broj nepridruženih studenata, odnosno povećati kvaliteta pridruživanja za bolje rangirane studente.

5.1. Metoda odabira redoslijeda elemenata

Metoda odabira redoslijeda elemenata pri rješavanju problema pridruživanja s prioritetnim listama u nastavnim procesima uporabom ekspertnog znanja koristi se pri izradi glavnog popisa rangiranih studenata.

Glavni popis je strogo poredana lista prema bodovima, najčešće prosječnim ocjenama studenata. Ako dva studenta imaju jednake bodove, tada se mogu generirati dva različita glavna popisa rangiranih studenata u kojima su varirane pozicije studenata na listi. Različiti glavni popisi rangiranih studenata uzrokuju različita pridruživanja, a time i različitim brojem nepridruženih studenata.

Kada je popis poredan slučajnim redoslijedom, tada se za njegovo poredavanje ne koristi nikakvo znanje. Za određivanje redoslijeda može se uporabiti i specifično znanje koje se odnosi na određeno usko područje. Takvo znane se naziva ekspertno znanje. Primjer ekspertnih znanja u problemima pridruživanja studenata ili grupiranih studenata može biti:

- Ocjena studenta koja se može koristiti za određivanje:
    - sličnosti ocjena studenta;
    - najuspješnijeg studenta u grupi;
    - prosječnoj ocjeni grupe.
- Područje:
    - grupe;
    - mentora.
- Nazivu:
    - studenta;

- projekta;
- voditelja grupe.

Ekspertna znanja u problemu pridruživanja koriste se kada se želi postići da su pridruživanja kvalitetnija. Ona trebaju ili smanjiti broj nepridruženih studenata, ili se koristiti za stvaranje novih grupa koje sadržavaju studente grupirane prema nekim pravilima, a takve grupe bez uporabe ekspertnih znanja ne bi se moglo postići.

5.1.1. Ekspertno znanje o sličnosti ocjena studenta

Ekspertno znanje o ocjeni studenata predstavlja novi neizraziti pristup rangiranju studenata za glavni popis rangiranih studenata. Temelj ovog algoritma je ekspertno znanje kojim se pretpostavlja da dva studenta imaju jednake ocjene čak i kada postoje manje razlike u njima.

Zanemarivanjem malih razlika u ocjenama, odnosno u konačnim prosječnim ocjenama studenata, dozvoljena je zamjena pozicije dva studenta na popisu rangiranih studenata i kada to, strogo matematički gledano dva studenta imaju različite ocjene. Zamjena većeg broja studenata omogućava generiranje većeg broja glavnih popisa rangiranih studenata čime se znatno povećava i broj mogućih pridruživanja odnosno manji broj nepridruženih studenata.

Parametar *udaljenost* (eng. distance) određuje najveću dozvoljenu promjenu ocjene prilikom zamjene pozicije dva studenta. Kada se za rangiranje studenata koristi prosječna ocjena najveća vrijednost ovog parametra iznosi 3,0 s obzirom na to da je to razlika između najbolje (5,0) i najniže prolazne ocjene (2,0).

Kad parametar *udaljenost* ima najmanju vrijednost (0,0) svi se studenti rangiraju isključivo temeljem njihove prosječne ocjene studija. Kada je vrijednost ovog parametra najveća, tada prosječne ocjene ne utječu na rangiranje i studenti s boljim prosjekom nemaju nikakvu prednost prilikom pridruživanja grupi.

5.1.2. Ekspertno znanje o ocjeni najuspješnijeg studenta u grupi

Ekspertno znanje o ocjeni najuspješnijeg studenta u grupi koristi se kada je potrebno grupirati već formiranu grupu studenata, a želi se postići da su nove grupe sadrže studente sličnih ocjena ili da su studenti sličnih ocjene raspršeni u različite grupe.

Kada se želi postići da su studenti dio iste grupe, tada se studenti rangiraju prema ocjeni najuspješnijeg studenta u grupi i onda se studenti redom pridružuju u prvu grupu do njenog

maksimalnog popunjena, kad se prelazi na idući grupu. Grupe se pune redom dok se ne pridruže svi studenti. Time će se dobiti grupe koje sadrže studente sličnih rangova.

Također ovo znanje se može koristiti i za raspršivanje studenata u različite grupe. Tada se treba odrediti koliko je grupa potrebno, te se grupa s najbolje rangiranim studentom pridružuje prvoj grupi, grupa s idućim najbolje rangiranim studentom u drugu grupu i tako redom. Kada se dođe do zadnje grupe, grupe studenata se ponovno pridružuju od prve grupe.

5.1.3. Ekspertno znanje o području grupe

Ekspertno znanje o području grupe se također koristi kada je potrebno grupirati već formirane grupe studenata u nove, veće grupe. Može koristiti kako bi se grupe s istim ili sličnim područjem grupirale zajedno u veću prezentacijsku grupu. Time studenti iz sličnih područja mogu raditi zajedno u jednoj grupi.

Najčešće će broj grupa u pojedinim područjima biti različit, te neće biti moguće izbjeći da se u novu grupu pridružuju grupe iz različitih područja. Da bi se to riješilo, najbolje je odrediti redoslijed područja, kako bi se postojeće grupe mogle poredati prema tom redoslijedu. Time se može postići da su studenti pridruženi u nove grupe koje se sastoje od više sličnih područja.

5.1.4. Ekspertno znanje o prosječnoj ocjeni grupe

Ekspertno znanje o prosječnoj ocjeni grupe, koristi se kada je potrebno grupirati već formirane grupe studenata u nove grupe. S ovim znanjem može se kombinirati da nove grupe sadrže kombinaciju dobrih i loših studenata.

Jedan od načina na koji se to može postići je da se grupe poredaju prema prosječnoj ocjeni grupe pri čemu je grupa s najvećim prosjekom na početku, a s najmanjim prosjekom na završetku liste. Potom se kreira prioritetna lista grupa, koja se koristi za pridruživanje, tako što se izmjenično stavljaju najuspješnija, pa najmanje uspješna grupa.

Kroz ovaj način bi sve grupe trebale imati od izvrsnih do loših studenata.

5.1.5. Ekspertno znanje o nazivu studenata, projekta ili voditelja grupe

Znanje o nazivu studenta, projekta ili voditelja grupe formirat će grupe slično nasumičnom redoslijedu. Kod ovog pridruživanja mogu se dobiti grupe koje je lako definirati. Na primjer, studenti čija prezimena počinju od A do Ca su grupa 1, Cb do De su grupa 2 itd.

Ovim načinom se ne mogu postići ciljevi u kojima se grupa treba sastojati od studenata prema nekom kriteriju, kao što je to navedeno kada se koristi ekspertno znanje o ocjeni najuspješnijeg studenta, području grupe i slično.

Abecedno ili nasumično pridruživanje ne koristi ekspertno znanje koje bi utjecalo na pridruživanje. Projektne se grupe poredaju se abecedno u glavnom popisu koji se dalje koristi za pridruživanje.

5.2. Metoda pridruživanja nepridruženih elemenata

Nepotpune studentske prioritetne liste mogu rezultirati nepridruženim studentima. U radovima navedenim u Poglavlju 4 predlažu se različite strategije rješavanja problema pridruživanja nepridruženih studenata. U jednom od radova predlaže se uporaba nasumičnog pridruživanja studenata grupama [34], a u više se radova predlaže uvođenje drugog kruga odabira [30], [32], [36], [37].

Metoda pridruživanja nepridruženih elemenata u nastavnim procesima temeljem zaključaka izvedenih iz konteksta odabira predložena ovim radom koristi drugačiji pristup pridruživanju nepridruženih studenata. Ona ne smanjuje broj nepridruženih studenata, već se oni pridružuju onim odabirima koje bi sami najvjerojatnije odabrali da su izradili dužu prioritetnu listu.

Metoda se primjenjuje za svakog studenta, a rezultat ove metode je izrada nove liste na kojoj su grupe koje student nije odabrao, a poredane su onako kako bi ih student najvjerojatnije sam poredao. Ovu novu listu nazivamo *proširena prioritetna lista* (eng. *extended preference list*).

Izrada proširene prioritetne liste inspirirana je idejom preporučiteljskih sustava (eng. recommender systems) [45]–[50]. Preporučiteljski sustavi temelje se na dva ulazna podatka:

(i) Odabir koji je napravila osoba za koju se rade preporuke;

(ii) Odabir koji su napravili drugi.

Rezultat rada preporučiteljskih sustava je lista novih, do tada neodabranih podataka. Preporučiteljski sustavi popularni su u sustavima *online* kupovine, gdje kupcu predlažu artikle temeljem dotadašnjih odabira i pregleda.

Za razliku od preporučiteljskih sustava kod kojeg su odabiri drugih osoba poznati u trenutku odabira pojedinog kupca, a temeljeni su na dužim periodima praćenja, podatak o odabirima koje su napravili drugi studenti, nije dostupan tijekom studentskog odabira grupa.

Grupe se mijenjaju svake godine, zbog čega podaci prikupljeni tijekom prethodne godine nisu upotrebljivi u idućoj. Nedostatak ovog ulaznog podatka razlog je zašto se studentima ne predlažu grupe tijekom samog procesa njihovog odabira, ali se mogu koristiti na kraju procesa kada su podaci odabira poznati.

Ova proširena prioritetna lista koristi se kada nije moguće pridružiti studenta nijednom odabiru s prioritetne liste. Bez korištenja prioritetne liste, uz dodjelu grupe u jednom krugu, studentima bi se grupa pridruživala nasumičnim redoslijedom. Uporabom proširene prioritetne liste studentu se stoga pridružuje grupa koja bi mu trebala bolje odgovarati.

Tablica 1 – Primjer prioritetnih lista pet studenata

Prioritetna lista STUDENTA S1	Prioritetne liste USPOREDNIH STUDENATA			
	STUDENT S2	STUDENT S3	STUDENT S4	STUDENT S5
Grupa P1	Grupa P3	Grupa P11	Grupa P1	Grupa P7
Grupa P2	Grupa P8	Grupa P3	Grupa P8	Grupa P8
Grupa P3	Grupa P5	Grupa P2	Grupa P7	Grupa P6
Grupa P4	Grupa P2	Grupa P9	Grupa P4	Grupa P9
Grupa P5	Grupa P10	Grupa P1	Grupa P10	Grupa P5
Grupa P6	Grupa P1	Grupa P7	Grupa P6	
	Grupa P7	Grupa P10	Grupa P9	
		Grupa P8		

Primjer: U tablici 1 dan je primjer pet prioritetnih lista koje su izradili studenti. Studenti su označeni oznakama S1 do S5. U primjeru se određuje proširena prioritetna lista za studenta S1.

5.2.1. Sličnost prioritetne liste

Sličnost prioritetne liste određuje se temeljem četiri vrijednosti sličnosti:

1. C1 – sličnost prvog odabira;
2. C3 – sličnost prvih tri odabira;
3. C5 – sličnost prvih pet odabira;
4. CMax – sličnost prvih N odabira.

Koraci (1, 3, 5 i Max) su određeni eksperimentalno temeljem najčešće odabranog broja grupa. Studenti su najčešće birali između pet i dvadeset grupa. Kada bi prosječni broj odabranih

grupa u nepotpunoj prioritetnoj listi studenata bio znatno veći, tada bi i predloženi koraci mogli biti drugačiji.

Sličnost odabira (C) određuje se kao postotak grupa koji je zajednički studentu koji se promatra i studentu s kojim se uspoređuje. Vrijednost sličnosti odabira računa se kao postotak zajedničkih grupa oba studenta na prvih 1, 3, 5 odnosno Max mjesta, pri čemu je Max broj grupa koje je odabrao promatrani student. Ako je broj grupa na jednoj strani manji od traženog broja grupa za usporedbu algoritam koristi taj manji broj grupa za usporedbu.

X = { 1, 3, 5, Max}

Max = broj grupa koje odabrao promatrani student

Cx = (broj zajedničkih grupa na prvih X mjesta) / X

Vrijednost sličnosti odabira je najveća (100%) kada obje liste imaju jednake odabrane grupe, pri čemu se zanemaruje njihov poredak na popisima. Vrijednost odabira je najmanja (0%) kada nema niti jedne zajedničke grupe.

Prioritetne liste studenata smatraju se sličnijima kada su iste grupe visoko rangirane. Zbog toga izračun sličnosti koristi težinske faktore. Suma svih težinskih faktora je 1 (100%). Način određivanja ovih faktora detaljno je objašnjen u dijelu 9.2, a predložene vrijednosti težinskih faktora su:

1. F1 = 0,30;
2. F3 = 0,28;
3. F5 = 0,25;
4. FMax = 0,17.

Predložene vrijednosti težinskih faktora su rezultat usporedbe stvarnih studentskih prioritetnih lista i proširenih prioritetnih lista.

Vrijednost *sličnosti prioritetnih lista* (PS) se određuje kao:

*PS = C1*F1 + C3*F3 + C5*F5 + WMax*FMax*

Tablica 2 – Sličnost prioritetnih lista studenta S1 s ostalim prioritetnim listama

Tablica sličnost prioritetnih lista	USPOREDNI STUDENTI Vrijednost sličnosti prioritetnih lista (PS)				Faktor
	STUDENT S2	STUDENT S3	STUDENT S4	STUDENT S5	
First choice (C1)	0%	0%	100%	0%	F1 = 0,3
First 3 choices (C3)	33%	67%	33%	0%	F3 = 0,28
First 5 choices (C5)	60%	60%	40%	20%	F5 = 0,25
Max choices (CMax)	67%	50%	50%	33%	FMax = 0,17
Total value of similarity with student S1	**36%**	**42%**	**58%**	**11%**	**SUM(F1..FMax) = 1**

Primjer: U tablici 2 prikazana je usporedba prioritetne liste promatranog studenta S1 s prioritetnim listama drugih studenata.

Sličnost prva tri odabira (C3) sa studentom S2 je 33% (1/3) jer oba studenta imaju zajednički odabir samo grupe P3 na prva tri mjesta njihovih prioritetnih lista. Ukupna vrijednost sličnosti prioritetne liste promatranog studenata S1 sa studentom S2 je:

PS = 0*0,3+0,33*0,28+0,6*0,25+0,67*0,17 = 36

5.2.2. Važnost grupe

Važnost grupe računa se za sve grupe koje su drugi studenti odabrali. Pri tome se dodjeljuje veća važnost grupama koje su:

(i) Više rangirane na prioritetnoj listi studenta s kojim se uspoređuje;

(ii) Rangirane na kraćoj prioritetnoj listi – kraća lista se smatra pažljivije odabranom od one na kojoj su, primjerice, poredane sve grupe.

Najveća vrijednost koju može imati *važnost grupe* je 1 kada student s kojim se uspoređuje odabere samo jednu grupu, a smanjuje se prema 0 za grupe koje su rangirane na posljednjem mjestu dugačke prioritetne liste. Na izračun *važnosti grupe* ne utječe prosječna ocjena promatranog studenta.

Važnost grupe (PF – eng. Group factor), određuje se za grupu i studenta kao:

PF = (Max – PP + 1) / Max

Pri čemu je:

Max = broj grupa koje je student odabrao i rangirao;

PP = Pozicija grupe na prioritetnoj listi studenta s kojim se uspoređuje. Za najviše rangiranu grupu je vrijednost 1, za iduću 2 i tako dalje.

Tablica 3 – Redoslijed ocjene za grupe koje student S1 nije odabrao na svojoj prioritetnoj listi u svrhu kreiranja proširene prioritetne liste

Tablica važnosti grupe za STUDENTA S1	USPOREDNI STUDENTI Važnost grupe (PF)			
	STUDENT S2	STUDENT S3	STUDENT S4	STUDENT S5
Grupa P7	0,14	0,38	0,71	1,00
Grupa P8	0,86	0,13	0,86	0,80
Grupa P9		0,63	0,14	0,40
Grupa P10	0,43	0,25	0,43	
Grupa P11		1,00		

Primjer: Tablica 3 prikazuje popis grupa, P7 do P11 koje student S1 nije odabrao na svojoj prioritetnoj listi. Grupe, P1 do P6 koje je student S1 odabrao ne analiziraju se za proširenu prioritetnu listu.

Važnost grupe P7 za studenta S2 određuje se na sljedeći način: grupa je studentu S2 rangirana na sedmom mjestu (PP = 7), broj odabranih grupa je sedam (Max = 7). Stoga slijedi:

PF = (7 - 7 + 1) / 7 = 0,14

Proširena prioritetna lista računa se kombinacijom sličnosti *liste prioriteta* (PS) iz Tablice 2 i rezultata izračuna Važnosti grupe (PF) za sve grupe iz Tablice 3.

5.2.3. Proširena prioritetna lista

Proširena prioritetna lista (eng. *extended preference list*) određuje se algoritmom kao ona u kojoj su grupe:

(i) Više rangirane ako je njihov rang grupe viši – *sličnost prioritetne liste* (PS);

(ii) Više rangirane ako su ih više rangirali studenti s većom *važnosti grupe* (PF).

Ako student s kojim se uspoređuje nema grupu navedenu na prioritetnoj listi tada PF ima vrijednost 0.

Grupe na proširenoj prioritetnoj listi rangiraju se upotrebljavajući *vrijednosti utjecaja grupe* (PIV) (eng. group impact value). Za n uspoređenih studenata, PIV se određuje kao:

$$PIV = \sum_{i=0}^{n} PS_i * PF_i$$

Grupe koje imaju veću vrijednost utjecaja grupe rangiraju se više. Algoritam na proširenu prioritetnu listu dodaje samo grupe koje imaju vrijednost utjecaja grupe veću od nule.

Tablica 4 – Primjer izračuna vrijednosti utjecaja grupe (PIV) na primjeru studenta S1

Tablica utjecaja grupe	USPOREDNI STUDENTI (PS * PF)				Ukupni utjecaj grupe (PIV)	Rang grupe
	STUDENT S2	STUDENT S3	STUDENT S4	STUDENT S5		
Grupa P7	0,05	0,16	0,41	0,11	0,73	2
Grupa P8	0,31	0,05	0,50	0,09	0,94	**1**
Grupa P9	0,00	0,26	0,08	0,04	0,39	5
Grupa P10	0,15	0,11	0,25	0,00	0,51	3
Grupa P11	0,00	0,42	0,00	0,00	0,42	4

Primjer: *Vrijednost sličnosti prioritetne liste* (PS) računa se u Tablici 2, a vrijednost važnosti grupe (PF) u Tablici 3. *Važnost grupe* P8 za studenta S2 računa se na sljedeći način:

PS (36%) * PF (0,86) = 0,31

Vrijednost utjecaja grupe (PIV) računa se kao zbroj svih vrijednosti utjecaja:

za grupu P8: 0,31 + 0,05 + 0,5 + 0,09 = 0,94.

Za primjer prikazan Tablicom 4 proširenu prioritetnu listu čine grupe rangirane prema vrijednosti utjecaja grupe (PIV):

P8 (najveća vrijednost PIV), P7, P10, P11 i P9 (najmanja vrijednost PIV).

5.3. Primjena metoda u algoritmima pridruživanja s prioritetnim listama

5.3.1. Algoritam strogog jednostrukog pridruživanja s glavnim popisom

Predloženi algoritam strogog jednostrukog pridruživanja s glavnim popisom rješava problem pridruživanja u nastavnim procesima u kojima:

- Studenti trebaju biti pridruženi točno jednoj grupi, čak i kada grupa nije na studentovoj prioritetnoj listi, što se u ovoj disertaciji naziva *strogo jednostruko pridruživanje*;
- Studenti imaju nepotpune prioritetne liste;
- Broj grupa koje je moguće odabrati može biti iznimno velik (veći od 100);
- Broj slobodnih mjesta u grupama može biti iznimno mali, a ukupan broj svih slobodnih mjesta jednak broju studenata;
- Koristi se glavni popis rangiranih studenata;

- Studenti s boljim prosjekom ocjena imaju prednost te se, kada je to moguće, pridružuju kvalitetnijim grupama.

Nepotpune prioritetne liste u kombinaciji s ograničenim brojem slobodnih mjesta u grupama rezultira nepridruženim studentima. U cilju pridruživanja svih studenata u samo jednom krugu, u ovom se radu predlaže korištenje zaključaka izvedenih iz konteksta odabira za kreiranje proširene prioritetne liste (eng. extended preference list) kao novi pristup u rješavanju problema nepridruženih studenata.

Zbog zahtjeva da studenti s boljim ocjenama imaju prednost u pridruživanju, sve grupe imaju istu, zajedničku, listu prioriteta – glavni popis rangiranih studenata. Na tom popisu su svi studenti koji se pridružuju grupama, a poredani su prema ocjenama. Studenti s boljim prosjekom ocjena su na početku popisa, a njih slijede studenti s lošijim prosjekom ocjena. Studenti dijele mjesto na glavnom popisu rangiranih studenata kada imaju jednake prosječne ocjene – takvi studenti stvaraju niz mogućih rješenja pridruživanja jer je potrebno u obzir uzeti sve slučajeve mogućih strogih poredaka. U izvedbi algoritma, gdje dva studenta imaju istu prosječnu ocjenu, izrađuju se oba popisa – jedan u kojem je prvi student rangiran više, i drugi u kojem je taj student rangiran niže. Time se dobivaju dva ili više mogućih rješenja te treba odrediti koje rješenje je bolje.

U većini opisanih radova, najboljim rješenjem smatra se ono u kojem je broj studenata koji su ostali nepridruženi najmanji. Ako postoji više rješenja s istim brojem nepridruženih studenata, tada se boljim smatra ono u kojem su studenti pridruženi u grupe koje su pozicionirali više na svojoj prioritetnoj listi. U njihovom pristupu glavni popis rangiranih studenata je strogo poredan – nema studenata koji imaju isti rang, a promjena ranga nije dozvoljena čak niti u slučaju kada je razlika u uvjetu rangiranja – u ovom slučaju to je prosječna ocjena – toliko mala da se može zanemariti. Razlika uvjeta rangiranja poštuje se čak i kada bi njena varijacija mogla rezultirati značajno boljim rezultatima.

Kada proces ne bi bio automatiziran i kada bi se voditeljima grupa dozvolilo ručno rangiranje studenata, pri čemu bi im se dozvolilo pozicioniranje studenata na isti rang, nastavnici bi studente s malom razlikom u ocjeni izjednačavali – davali im jednaki rang. U automatiziranim procesima, ocjene se često gledaju strogo matematički i ne upotrebljavaju se nikakva dodatna znanja. Gledano matematički, student S1 koji ima prosjek 4,495 zaokružen na dvije decimale imat će prosjek 4,50 i smatra se boljim od studenta koji ima prosjek 4,494 jer je njegov prosjek zaokružen na dvije decimale manji, 4,94. Da su prvi i drugi student imali za

0,001 veću ocjenu, tada bi oba studenta imali različitu ocjenu gledano na 3 decimale, ali jednaku na dvije decimale.

Ovim radom predlaže se uvođenje novog algoritma izrade glavnog popisa rangiranih studenata koji koristi ekspertna znanja u nastavnim procesima za određivanje redoslijeda studenata uporabom neizrazitog pristupa (eng. *fuzzy approach*). Sâm algoritam inspiriran je načelima neizrazite logike (eng. *fuzzy logic*) [51]–[53]. Neizraziti pristup omogućava algoritmu promjenu prosječne ocjene studenta, što je temelj izrade glavnog popisa rangiranih studenata, unutar dozvoljenih granica. Time se zamagljuju dosadašnje stroge granice ocjena i omogućava se zamjena pozicija dvaju studenata na popisu iako se njihove ocjene razlikuju. Ovim pristupom simulira se ljudska procjena u kojoj se studenti s malom razlikom u ocjeni smatraju jednako dobrima.

Uvođenjem neizrazitog pristupa i proširene prioritetne liste mijenja se definicija najboljeg rezultata. U kontekstu pridruživanja studenata projektima, a u svrhu algoritma, najboljim rezultatom smatra se onaj u kojem preostaje najmanji broj nepridruženih studenata, pri čemu se prednost daje studentima koji imaju bolji prosjek ocjena – odnosno pridruženi su više na glavnom popisu rangiranih studenata.

Novi *algoritam strogog jednostrukog pridruživanja s glavnim popisom* kombinira nekoliko novih ili poboljšanih algoritama određivanja najboljeg pridruživanja:

1. **Algoritam izrade glavnog popisa rangiranih studenata**
   (eng. *Master preference list generation algorithm*) koji koristi novopredloženu metodu odabira redoslijeda elemenata pri rješavanju problema pridruživanja s prioritetnim listama u nastavnim procesima uporabom ekspertnog znanja;
2. **Algoritam izrade proširene prioritetne liste**
   (eng. *Extended preference list generation algorithm*) koji koristi novopredloženu metodu za pridruživanje nepridruženih elemenata u nastavnim procesima temeljem zaključaka izvedenih iz konteksta odabira;
3. **Algoritam strogog jednostrukog pridruživanja**
   (eng. *Student allocation algorithm*);
4. **Algoritam ocjene strogog jednostrukog pridruživanja**
   (eng. *Allocation evaluation algorithm*).

Najbolje rješenje pridruživanja može se pronaći ako se izrade i ocijene svi mogući glavni popisi rangiranih studenata. Način rješavanja ovim pristupom može biti vrlo dugotrajan

s obzirom na broj mogućih glavnih popisa rangiranih studenata koji u najgorem slučaju može biti N! za N studenata koji svi imaju jednaki rang.

Za generiranje glavnog popisa rangiranih studenata algoritam koristi neizraziti pristup u kojem varijacijom prosječne ocjene studenta kreira dodatni broj glavnih popisa rangiranih studenata. Generiranje i ocjena svih takvih popisa može zahtijevati jako puno vremena. Zbog toga se kombiniraju dva pristupa kako bi se brže i bolje pronašlo rješenje:

- algoritam se izvodi u raspodijeljenom okruženju;
- za odabir podskupa koji će se analizirati koristi se **genetski algoritam** pretraživanja (eng. *genetic algorithm*) a koji je primjenjiv u ovim vrstama problema [54][12].

Raspodjela algoritma na više računala omogućava širu pretragu genetskog algoritma u kraćem periodu. Navedena je karakteristika korisna kada je potrebno provesti iznimno veliki broj pretraga.

5.3.1.1. Glavni genetski algoritam

Ovaj algoritam izvodi se na svakom klijentu. Algoritam, temeljem pripremljenih podataka poslužiteljske strane, računa i traži najbolje rješenje. Za pretragu rješenja koristi genetski algoritam.

Na svakom klijentu, prvo se generira proširena prioritetna lista svakog studenta, upotrebljavajući *Algoritam izrade proširene prioritetne liste*. Algoritam nastavlja iterativno i tijekom jedne iteracije izrađuje i ocjenjuje jedno rješenje sljedećim koracima:

(i) Kreira glavni popis rangiranih studenata uporabom *Algoritma za izradu glavnog popisa rangiranih studenata;*

(ii) Pridružuje studente uporabom glavnog popisa rangiranih studenata i proširene prioritetne liste upotrebljavajući *Algoritam pridruživanja studenata;*

(iii) Ocjenjuje dobiveno pridruživanje uporabom funkcije dobrote (eng. *fitness*) upotrebljavajući *Algoritam ocjene jednostrukog pridruživanja.*

Sva su rješenja koja algoritam generira validna i uvijek zadovoljavaju sljedeće:

1. Rješenje je stabilno;
2. Poštuju se ograničenja donjih i gornjih granica slobodnih mjesta u grupama;

3. Grupe su pridružene studentima temeljem pozicije studenta na glavnom popisu rangiranih studenata – grupa se prvo pridružuje studentu koji je na početku glavnog popisa, zatim idućem i tako dalje;
4. Studenti su rangirani na glavnom popisu rangiranih studenta prema prosječnoj ocjeni – uspjehu na studiju;
5. Pozicija dva studenta na glavnom popisu rangiranih studenata može se zamijeniti ako je njihova razlika prosječne ocjene manja ili jednaka dozvoljenoj udaljenosti zamjene.

Svako rješenje ocjenjuje se uporabom algoritma ocjene jednostrukog pridruživanja koji računa funkciju dobrote rješenja. Rješenje se smatra boljim kada ima više studenata koji ispunjavaju navedene uvjete – uvjeti su poredani prema važnosti:

1. Student je pridružen grupi temeljem vlastite prioritetne liste, a ne putem proširene prioritetne liste;
2. Student je pridružen grupi koja je više rangirana na prioritetnoj listi;
3. Student je pridružen grupi na proširenoj prioritetnoj listi, a ne nasumičnim pridruživanjem;
4. Student je pridružen grupi koja je više rangirana na proširenoj prioritetnoj listi.

Genetski algoritam implementiran je na sljedeći način:

1. Glavni popis rangiranih studenata predstavljen je vektorom prosječnih ocjena studenata i bilježi se kao jedan kromosom;
2. Početna populacija kreira nasumične glavne popise rangiranih studenata uporabom parametra algoritma *PopulationSize* koji određuje *veličinu populacije;*
3. Za odabir roditelja koristi se odabir kraćenjem (eng. *truncation selection)*. Nakon što je kreirano *PopulationSize* glavnih popisa rangiranih studenata, oni se ocjenjuju algoritmom ocjene jednostrukog pridruživanja odnosno funkcijom dobrote koju implementira, a najbolja dva rješenja upotrebljavaju se kao roditelji za generiranje iduće populacije;
4. Za generiranje djece, koristi se jednolično križanje (eng. *uniform crossover*) kojim se prosječna ocjena svakog od studenata odabire temeljem jednog od roditelja s jednakom vjerojatnosti;
5. Mutacija koristi parametar *ChanceOfMutation* za *vjerojatnost mutacije* gena prilikom generiranja djece. Svakih N iteracija, gdje je N zadan vrijednosti parametra

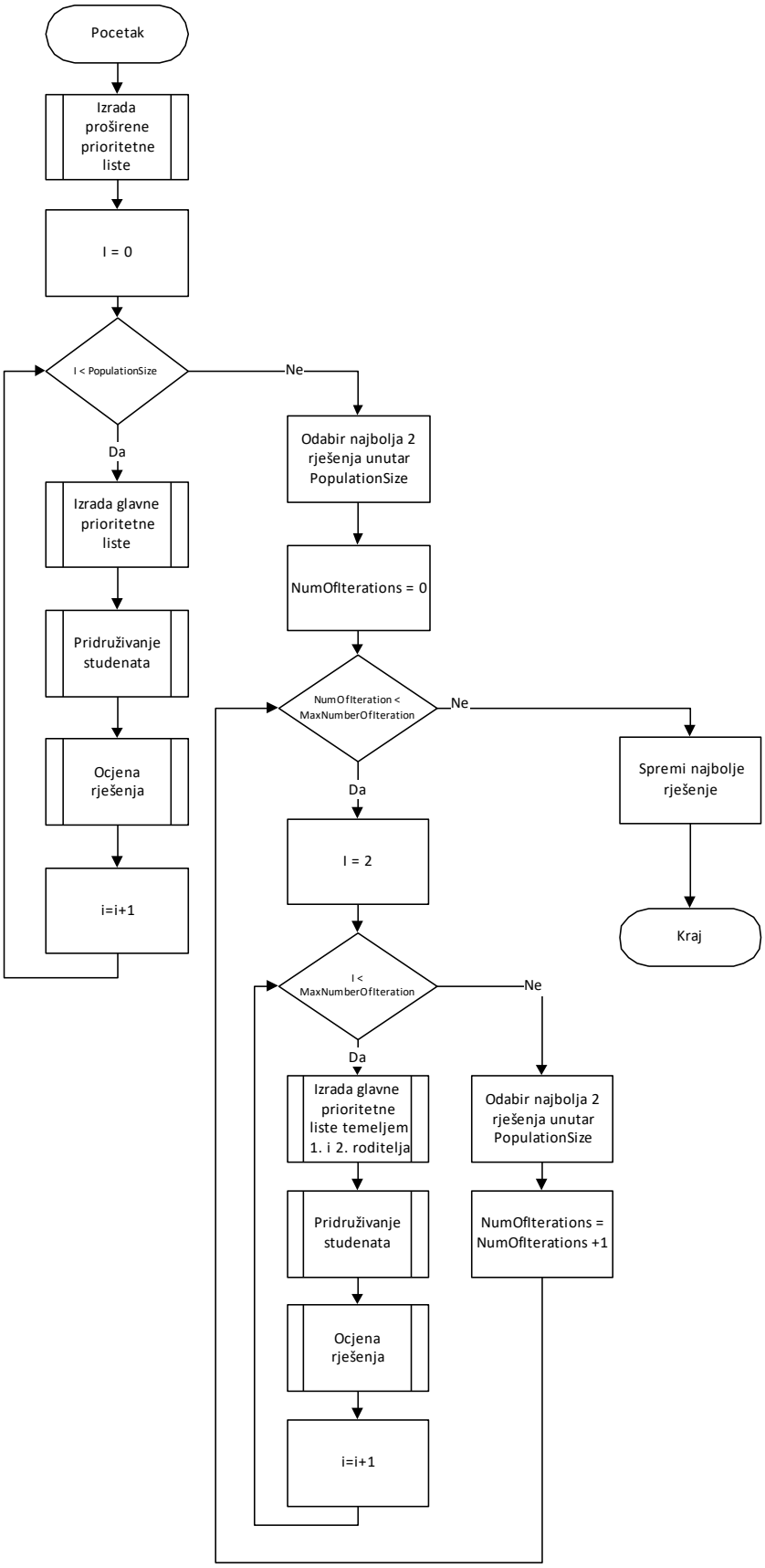
IncreasedMutationCycle, mutacija se povećava za 50% populacije na zadanu vrijednost parametra *MaxChanceOfMutation*;

6. Tijekom mutacije, algoritam mijenja osnovnu prosječnu ocjenu studenta do najviše vrijednosti zadane parametrom *Distance* (*udaljenost),* čime se kreira novi glavni popis rangiranih studenata;
7. Algoritam ocjene jednostrukog pridruživanja računa funkciju dobrote kao jedinstveni cilj (eng. *single objective*) genetskog algoritma;
8. Genetski algoritam završava nakon što je dosegnut broj iteracija zadan parametrom algoritma *MaxNumberOfIteration* (*najveći broj iteracija)*.

Algoritam **1** predstavlja pseudokôd (eng. *pseudocode*) glavnog algoritma koji se izvodi na jednom klijentu, a uključuje pozive ostalih metoda algoritma detaljno objašnjenih u nastavku ovog rada.

```
FUNCTION MAIN GENETIC ALGORITHM (DatabaseData, AllInputData)

  AllStudents := AllInputData.AllStudents
  AllGroups := AllInputData.AllStudents
  Distance := AllInputData.AllStudents

  //Generate Extended preference list for all students
  FOREACH Student IN AllStudents DO
    Student.ExtendedPreferenceList :=
            GENERATE EXTENDED PREFERENCE LIST
            (Student, AllStudents, AllGroups)
  END FOREACH

  //Generate first set - initial population – random
  FOR i:=0 TO PopulationSize
    Allocations[i].MasterList :=
            GENERATE AN INITIAL MASTER PREFERENCE LIST
            (AllStudents, Distance)
    Allocations[i].Allocations :=
            ALLOCATE STUDENTS
            (AllStudents,AllGroups,Allocations[i].MasterList)
    Allocations[i].Score :=
            EVALUATE ALLOCATIONS
            (Allocations[i].StudentsAllocations)
  END FOR

  Allocations:= Allocations descending ordered by score
  BestAllocation.Score := Allocations[0].Score
  NumOfIteration := 0

  WHILE NumOfIteration < AllInputData.MaxNumberOfIteration DO
    //Generate new set – based on genetic
    //Parents (first two – best) + Children
    FOR i:=2 TO AllInputData.PopulationSize

      RandKoef := AllInputData.ChanceOfMutation;

      IF(NumOfIteration% AllInputData.IncreasedMutationCycle == 0 && i%2==0) THEN
        RandKoef := AllInputData.MaxChanceOfMutation;
      END IF

      Allocations[i].MasterList :=
              GENERATE A MASTER PREFERENCE LIST FROM PARENTS
              (AllStudents,Allocations[0].MasterList,
                    Allocations[1].MasterList, AllInputData.MutationProbability)
      Allocations[i].Allocations =
              ALLOCATE STUDENTS
              (AllStudents,AllGroups,Allocations[i].MasterList)
      Allocations[i].Score =
              EVALUATE ALLOCATIONS
              (Allocations[i].StudentsAllocations)
    END FOR

    //Check if better allocation is found
    Allocations:= Allocations desc. ordered by score
    IF Allocations[0].Score > BestAllocation.Score THEN
      BestAllocation := Allocations[0]
    END IF
  END WHILE

  //Save results
  SAVE BESTALLOCATION (DatabaseData, BestAllocation)

END FUNCTION
```

Algoritam 1 – Pseudokôd glavnog genetskog algoritma



Slika 1 – Dijagram tijeka glavnog algoritma

Ulazni parametri algoritma (broj u zagradi označava vrijednost koja se pokazala najboljom tijekom eksperimenata):

1. *AllStudents* – Popis studenata, njihovih prioritetnih lista i prosječnih ocjena;
2. *AllGroups* – Popis dostupnih grupa;
3. *Distance* – Prihvatljiva promjena prosječne ocjene studenta za uporabu neizrazite generacije glavnog popisa rangiranih studenata;
4. *ChanceOfMutation* – Vjerojatnost mutacije kromosoma djeteta – promjene prosječne ocjene studenta (najbolje: 10);
5. *MaxChanceOfMutation* – Povećana vrijednost mutacije (najbolje: 80);
6. *IncreasedMutationCycle* – Povećani ciklus mutacije svakih N iteracija;
7. *MaxNumberOfIteration* – Dozvoljeni broj iteracija – nakon zadanog broja iteracija algoritam završava s radom (najbolje: 100.000);
8. *PopulationSize* – Veličina jedne populacija (najbolje: 15).

Izlazni podaci:

- *BestAllocation* – Popis pridruživanja studenata grupama.

5.3.1.2. Algoritam izrade glavnog popisa

Algoritam izrade glavnog popisa koristi *metodu odabira redoslijeda elemenata* pri rješavanju problema pridruživanja s prioritetnim listama u nastavnim procesima uporabom ekspertnog znanja. Glavni genetski algoritam mora ocijeniti sve glavne popise rangiranih studenata da bi našao najbolje pridruživanje, ono u kojem je najmanji broj nepridruženih studenata i u kojem se osigurava prednost studentima s boljim prosjekom ocjena.

S obzirom na to da se u jednom trenutku može ocijeniti samo jedan glavni popis rangiranih studenata, za ocjenu više glavnih popisa koristi se iterativni proces. U svakoj iteraciji odabire se novi glavni popis rangiranih studenata temeljem njihovih prosjeka ocjena. Porastom broja studenata, povećava se i broj mogućih glavnih popisa rangiranih studenata. Najveći broj glavnih popisa rangiranih studenata za N studenata iznosi N!, što se događa kada svi studenti imaju jednake prosječne ocjene.

Parametar *udaljenost* koristi se kako bi promijenio prosječnu ocjenu svakog studenta pri čemu u obzir uzima najveću (5,0) i najmanju (2,0) prosječnu ocjenu koju student može ostvariti. Nakon određivanja novih ocjena, temeljem parametra *udaljenost*, algoritam generira

nove glavne popise rangiranih studenata redanjem prema novoodređenim prosječnim ocjenama.

Glavni popis rangiranih studenata koji je rezultat izvođenja algoritma, ujedno predstavlja kromosom genetskog algoritma. Algoritam se poziva iz dvije funkcije unutar glavnog programa:

(i) Generiranje početne populacije pozivom funkcije iz glavnog dijela programa: *GENERATE AN INITIAL MASTER PREFERENCE LIST;*

(ii) Generiranje populacije djece pozivom funkcije iz glavnog dijela programa: *GENERATE A MASTER PREFERENCE LIST FROM PARENTS.*

Generiranje početne populacije (i) temelji se na izradi niza različitih početnih glavnih popisa rangiranih studenata uporabom Algoritma 2. U skladu sa zahtjevima genetskog algoritma, najbolja se pridruživanja odabiru kao roditelji populacije djece.

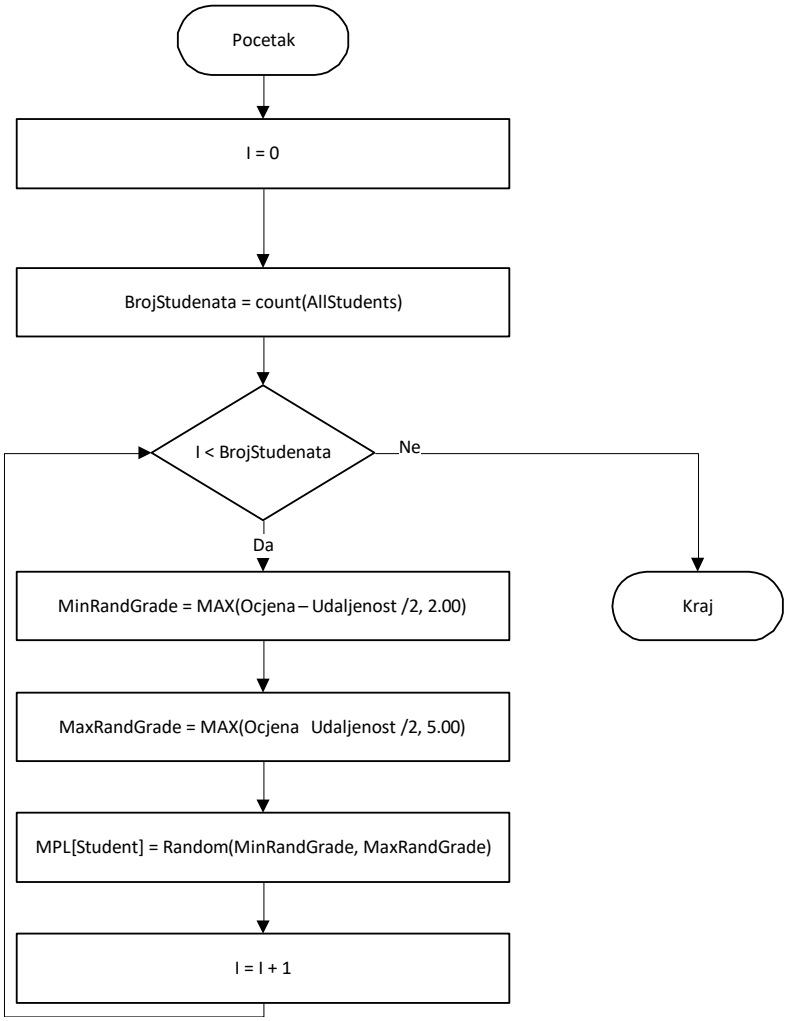
Algoritam zahtijeva sljedeće ulazne parametre:

1. *AllStudents* (svi studenti) – Popis svih studenata, njihovih prioritetnih lista i njihovih prosječnih ocjena;
2. *Distance* (udaljenost) – Prihvatljiva promjena ocjene za neizrazitu izradu glavnog popisa rangiranih studenata.

Izlaz ovog algoritma je nasumično generiran glavni popis rangiranih studenata.

```
FUNCTION GENERATE AN INITIAL MASTER PREFERENCE LIST

  FOREACH Student IN AllStudents DO

    //Fuzzy approach - change student grade
    MinRandGrade :=
         MAX(Student.StudyGrade – Distance/2, 2.00)
    MaxRandGrade :=
         MIN(Student.StudyGrade + Distance/2, 5.00)
    MPL[Student.ID] :=
         RANDOM(MinRandGrade, MaxRandGrade)

  END FOREACH

  MPL:= SORT LIST DESCENDING BY STUDENT STUDY GRADES (MPL)

  RETURN MPL

END FUNCTION
```

Algoritam 2 – Pseudokôd generiranja izrade glavnog popisa rangiranih studenata



Slika 2 – Dijagram tijeka generiranja izrade glavnog popisa rangiranih studenata

Algoritam generiranja populacije djece (ii) koristi mehanizme križanja i mutacije u cilju stvaranja nove populacije temeljene na odabranim roditeljima. Pristup algoritma u optimizaciji križanja i mutacije je uporaba vrlo male vjerojatnosti mutacije tijekom većine procesa kako je određeno parametrom *ChangeOfMutation* (vjerojatnost mutacije). To omogućava algoritmu bolje istraživanje lokalnih ekstrema u cilju pronalaska najboljeg rješenja.

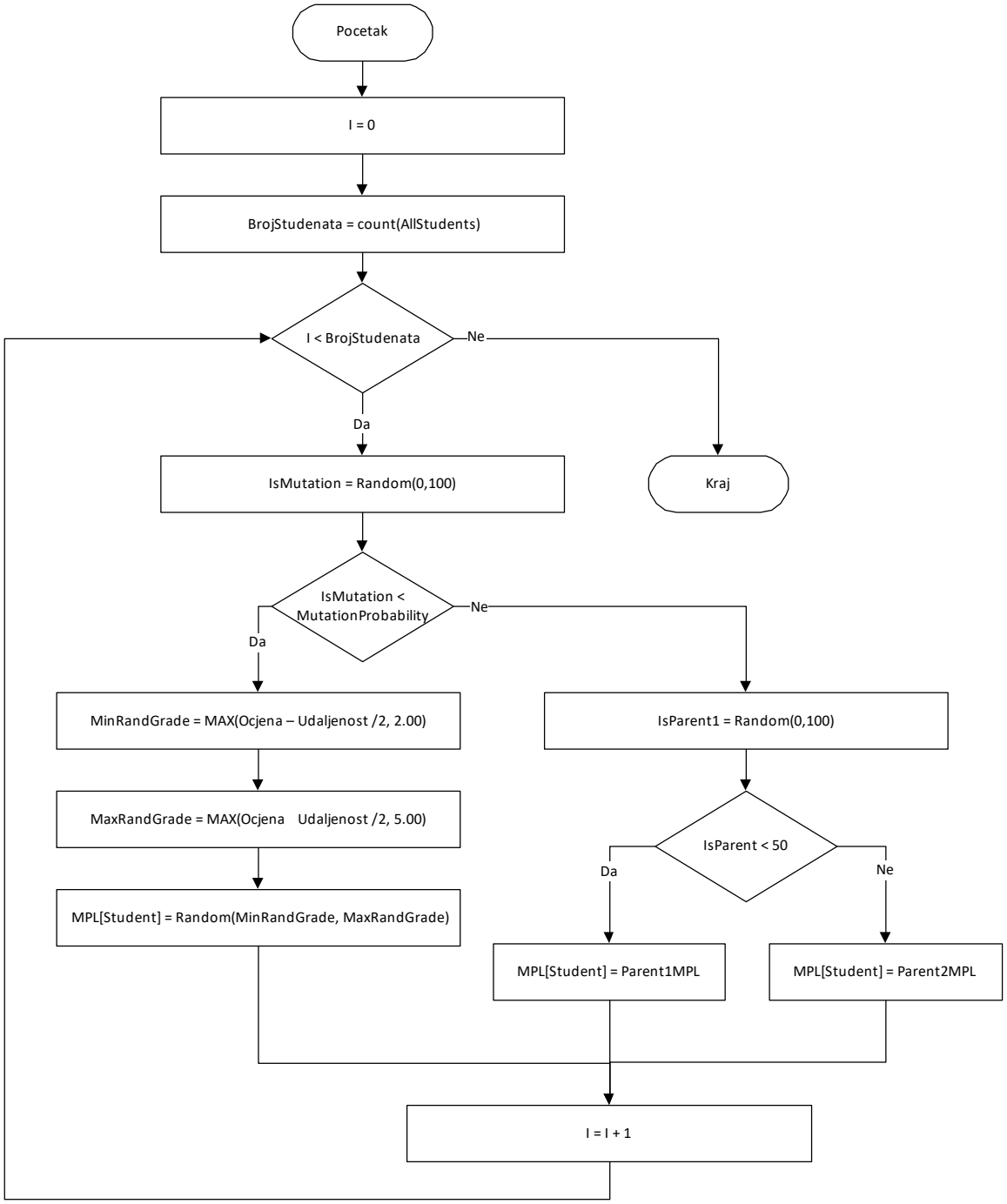
Tijekom ciklusa iteracije koji je višekratnik parametra *IncreasedMutationCycle*, 50% djece će imati uvećanu vjerojatnost mutacije u skladu s vrijednosti parametra *MaxChangeOfMutation*. Ovakav pristup omogućava udaljavanje od mogućeg lokalnog ekstrema u rješenju pridruživanja i istraživanje drugih mogućih rješenja. Vjerojatnost mutacije zadana je parametrom u algoritmu 3; redni broj iteracije i vjerojatnost mutacije određuju se u glavnom dijelu Algoritma 1.

Potrebni ulazni parametri za algoritam izgrade glavnog popisa rangiranih studenata su:

1. *AllStudents* (svi studenti) – popis studenata, njihovih prioritetnih lista i prosječnih ocjena;
2. *Distance* (udaljenost) – prihvatljiva promjena prosječne ocjene studenata za neizraziti pristup u izradi glavnog popisa rangiranih studenata;
3. *Parent1MasterList* (glavni popis rangiranih studenta prvog roditelja) – rangirani studenti prvog roditelja;
4. *Parent2MasterList* (glavni popis rangiranih studenata drugog roditelja) – rangirani studenti drugog roditelja;
5. *MutationProbability* (vjerojatnost mutacije) – vjerojatnost mutacije pojedinog kromosoma.

```
FUNCTION GENERATE A MASTER PREFERENCE LIST FROM PARENTS

  //For each student
  FOREACH Student IN AllStudents DO

    IsMutation := RANDOM(0, 100)

    //Set grade using mutation if mutation probability is given
    IF IsMutation < MutationProbability THEN

      // Set new grade is changed by max distance
      MinRandGrade :=
          MAX(Student.StudyGrade - Distance/2, 2.00)
      MaxRandGrade :=
          MIN(Sudent.StudyGrade + Distance/2, 5.00)
      MPL[Sudent.ID] :=
          RANDOM(MinRandGrade, MaxRandGrade)

    //or get grade from parents
    ELSE
       //chance of geting from parent 1
      IsParent1 := RANDOM(0, 100) < 50

      IF IsParent1 THEN
        MPL[Student.ID] := Parent1MasterList[Student.ID]
      ELSE
        MPL[Student.ID] := Parent2MasterList[Student.ID]
      END IF

    END IF
  END FOREACH

  //Sort final list
  MPL:= Sort MPL descending by student study grades
  RETURN MPL

END FUNCTION
```

Algoritam 3 – Pseudokôd generiranja glavnog popisa rangiranih studenata uz nasljeđivanje



Slika 3 – Dijagram tijeka generiranja glavnog popisa rangiranih studenata uz nasljeđivanje

5.3.1.3. Algoritam izrade proširene prioritetne liste

Algoritam izrade proširene prioritetne liste koristi metodu za pridruživanje nepridruženih elemenata u nastavnim procesima temeljem zaključaka izvedenih iz konteksta odabira. Ovaj algoritam koristi se na početku glavnog genetskog algoritma i poziva se za sve studente.

Traženi ulazni parametri algoritma su:

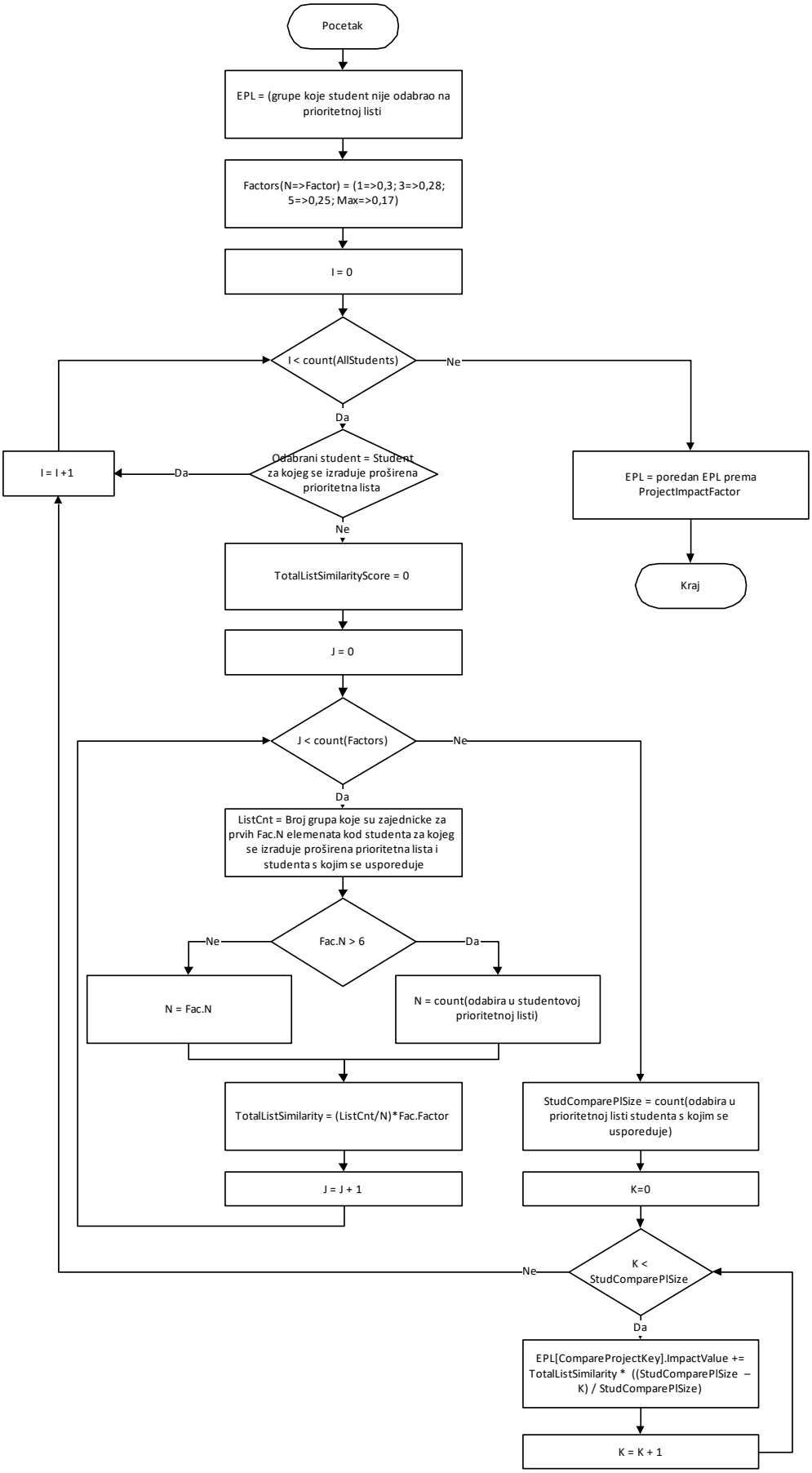
1. *Student* – prioritetna lista studenta;
2. *AllStudents* (svi studenti) – Prioritetne liste svih studenata;
3. *AllGroups* (sve grupe) – Popis svih dostupnih grupa.

Izlazni podaci:

1. *EPL* – Proširena prioritetna lista studenta.

```
FUNCTION GENERATE EXTENDED PREFERENCE LIST

  EPL := AllGroups without groups already on Student's preference list
  Factors(N=>Factor) := (1=>0.3, 3=>0.28, 5=>0.25, MAX=>0.17)

  FOREACH CompareStudent IN AllStudents DO

    IF Student.ID == CompareStudent.ID THEN CONTINUE END IF

    //Calculate total list similarities
    TotalListSimilarityScore := 0
    FOREACH Fac IN Factors DO

      LIST1 := New list as first Fac.N
               elements from a student's preference list
      LIST2 := New list as first Fac.N
               elements from a compared student's pref. list
      ListCnt := Number of same groups
                 in LIST1 AND LIST2
      N := Fac.N
      IF Fac.N > 6 THEN
        N := Student.PreferencesList.Count
      END IF

      TotalListSimilarity += (ListCnt/N)*Fac.Factor;

    END FOREACH

    //For each group on compared student pref. list
    //calculate Group Impact Value
    StudComparePLSize :=  COUNT(studCompare.PreferencesList)

    FOR i:=0 TO StudComparePLSize DO

      GroupFactor := ((StudComparePLSize - i) / StudComparePLSize)
      GroupImpact := TotalListSimilarity * GroupFactor
      CompareGroupKey := CompareStudent.PreferenceList[i].GroupKey
      EPL[CompareGroupKey].GroupImpactValue += GroupImpact;

    END FOR

  END FOREACH //STUDENTS

  EPL:= Sort EPL descending by GroupImpactValue

  RETURN EPL

END FUNCTION
```

Algoritam 4 – Pseudokôd algoritma generiranja proširene prioritetne liste



Slika 4 – Dijagram tijeka algoritma generiranja proširene prioritetne liste

Proširena prioritetna lista računa se temeljem:

i. *sličnosti prioritetne liste* i
ii. *važnosti grupe*.

Algoritam računa obje vrijednosti za svakog studenta. Porastom broja studenata, prikuplja se više podataka i povećava se kvaliteta generirane proširene prioritetne liste.

5.3.1.4. Algoritam strogog jednostrukog pridruživanja

Algoritam pridružuje studente grupama temeljem prioritetne liste koju studenti sami određuju, putem proširene prioritetne liste, koja se automatski generira za svakog studenta te putem glavnog popisa rangiranih studenata. Studenti u grupe trebaju biti pridruženi ravnomjerno, a za pridruživanje se koristi pohlepna metoda (eng. *greedy method*).

Algoritam prvo studentu na početku glavnog popisa rangiranih studenata određuje grupu, a potom prelazi na idućeg studenta na popisu. Svakog studenta algoritam pokuša prvo pridružiti prvoj dostupnoj grupi na njegovoj prioritetnoj listi. Ako niti jedna takva nema slobodnog mjesta, algoritam ga pokušava pridružiti temeljem proširene prioritetne liste. Ako niti to nije moguće, pridružuje ga nasumično odabranoj grupi u kojoj je dovoljno slobodnih mjesta.

Tijekom izvođenja algoritam prati broj slobodnih mjesta u svakoj od grupa u cilju osiguranja ravnomjernog pridruživanja na grupama.

Svaka grupa može prihvatiti dva ili tri studenta. Algoritam na početku određuje koliko grupa može imati pridruženo tri studenta. Nakon što je tijekom pridruživanja taj broj grupa dosegnut, najveći broj dostupnih mjesta na grupama postavlja se na dva. Ovim pristupom osigurava se ravnomjerno pridruživanje studenata na sve grupe.

Za potrebe funkcije dobrote (ocjene) genetskog algoritma, algoritam za svakog studenta uz podatke o pridruženoj grupi, sprema i podatke o redoslijedu promatrane grupe na studentovoj listi, te je li grupa pridružena sa studentove prioritetne ili proširene prioritetne liste.

Algoritam vraća jedno rješenje koje se temelji na podacima o studentima, odabranim prioritetnim listama i grupama te zadanim glavnim popisom rangiranih studenata. Algoritam se poziva iz glavnog genetskog algoritma s jednakim podacima o studentima i grupama, ali s drugačijim glavnim popisom rangiranih studenata. Promjenom redoslijeda studenata na glavnoj listi mijenja se i rješenje koje algoritam vraća.

Zbog ograničenog broja mjesta na grupama, redoslijed pridruživanja studenata grupama utječe na broj slobodnih mjesta na grupama prema principu „tko prvi“. Tijekom procesa

pridruživanja prva dostupna grupa se određuje uporabom metode *Find first available group* (pronađi prvu dostupnu grupu).

Traženi ulazni parametri algoritma su:

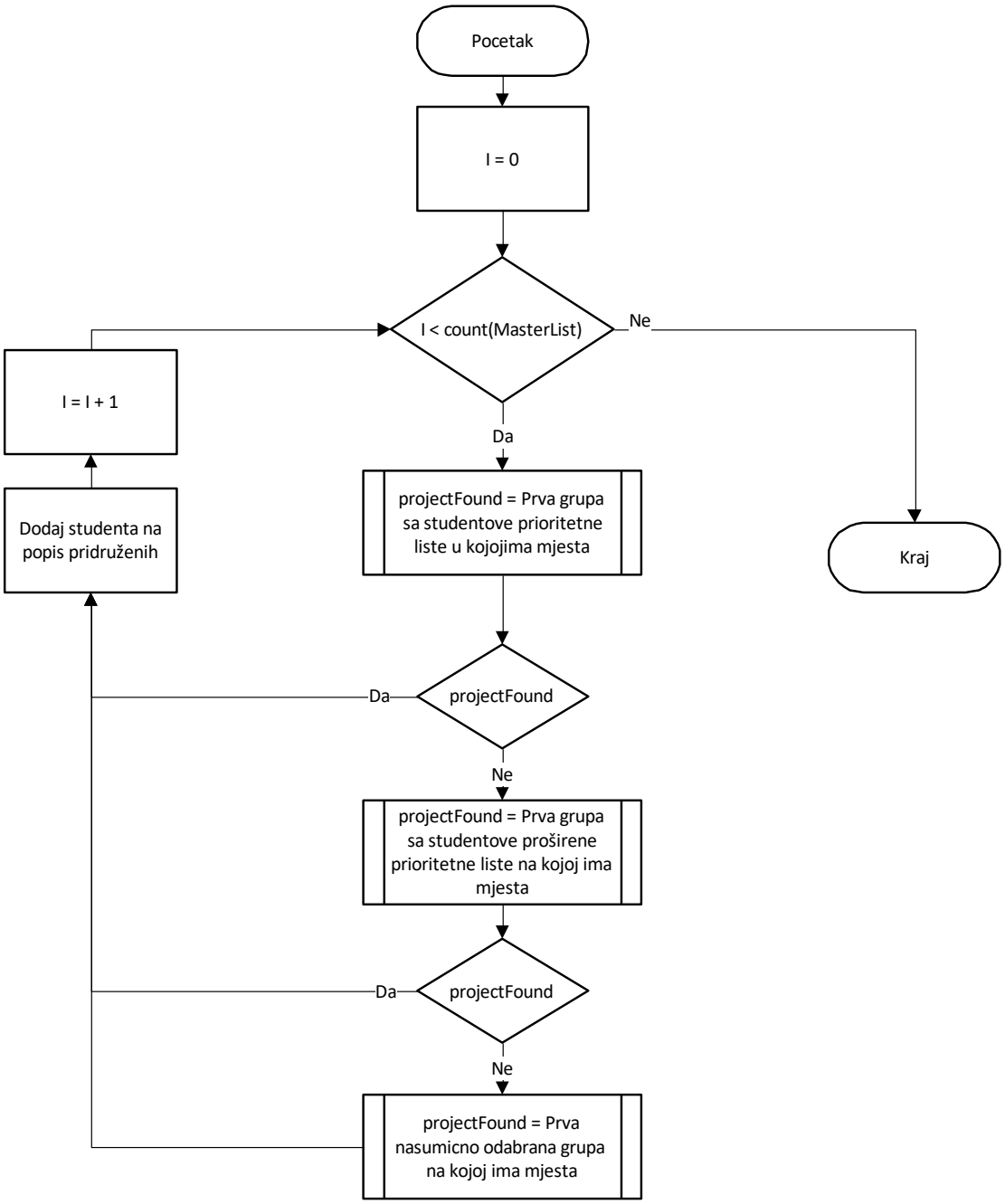
1. *AllStudents* (svi studenti) – Prioritetne liste svih studenata;
2. *AllGroups* (sve grupe) – Popis svih dostupnih grupa;
3. *MasterList* (glavni popis) – Glavni popis rangiranih studenata.

Izlazni podaci:

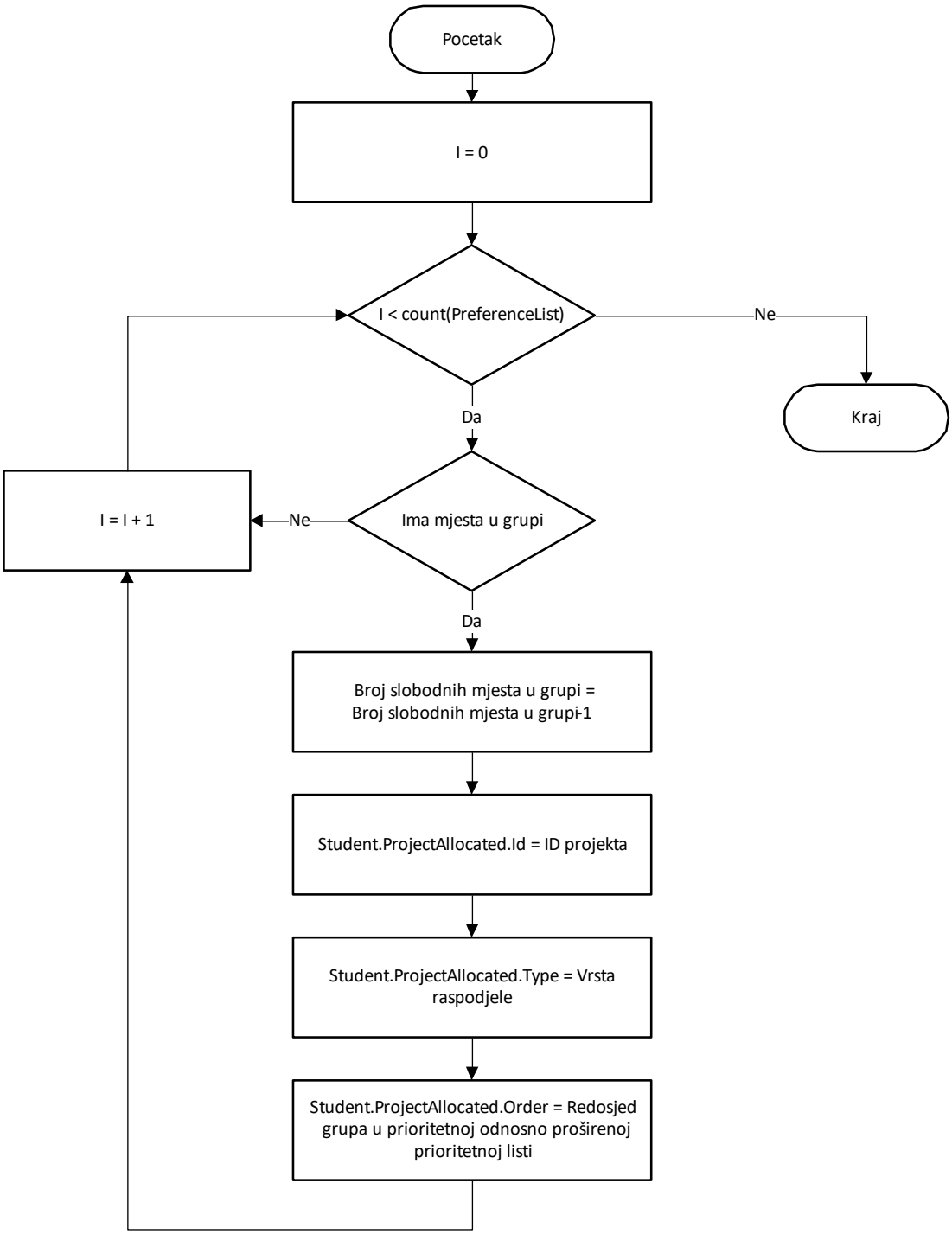
1. *StudentsAllocations* (pridruženi studenti) – Popis pridruženih studenata.

```
FUNCTION ALLOCATE STUDENTS

  FOREACH Student IN MasterList DO
    groupFound := FIND FIRST AVAILABLE GROUP
                      (OUT Student, 'Preference')
    IF NOT groupFound THEN
      groupFound:= FIND FIRST AVAILABLE GROUP
                       (OUT Student, 'Extended')
    END IF
    IF NOT groupFound THEN
          FIND FIRST AVAILABLE GROUP
               (OUT Student,'Random')
    END IF
    StudentsAllocations += Student

  END FOREACH

  RETURN StudentsAllocations

END FUNCTION

FUNCTION FIND FIRST AVAILABLE GROUP

  FOREACH Group IN PreferenceList

    IF Group have available places > 0 THEN

      UPDATE AVAILABLE PLACES (Group.ID)
      Student.GroupAllocated.ID :=Group.ID
      Student.GroupAllocated.Type := PreferenceListType
      Student.GroupAllocated.Order := Group.Order

    END IF

  END FOREACH

END FUNCTION
```

Algoritam 5 – Pseudokôd algoritma pridruživanja studenata



Slika 5 – Dijagram tijeka algoritma pridruživanja studenata

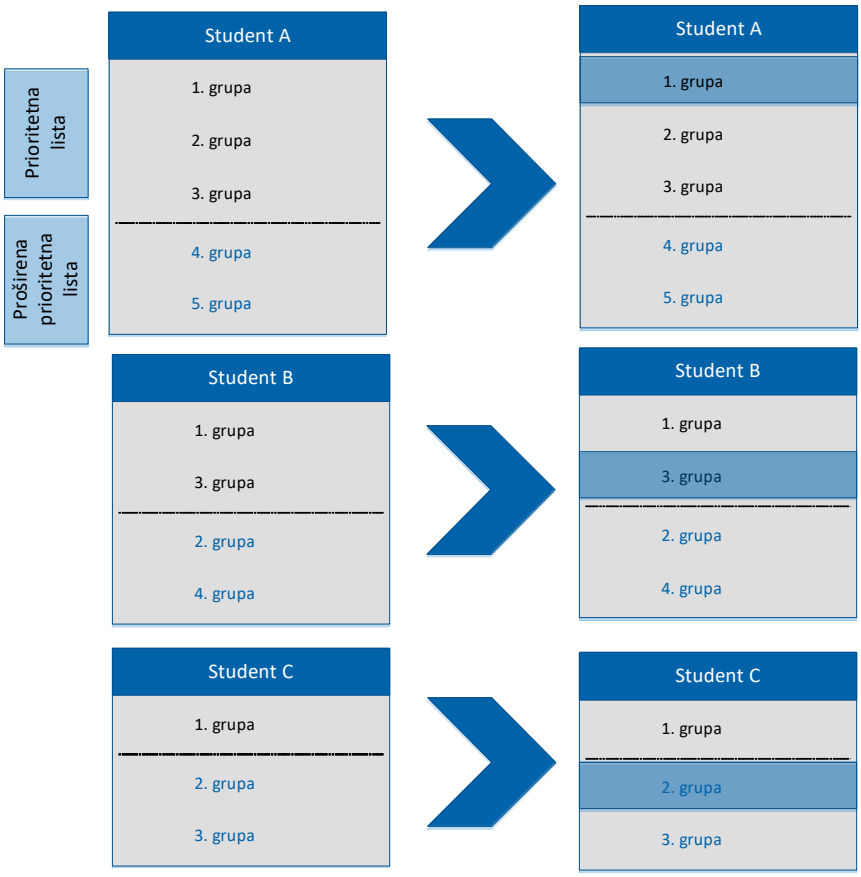


Slika 6 – Algoritma tijeka dijela algoritma pridruživanja studenata – dohvat grupe

Nakon izvođenja algoritma pridruživanja, rješenje je moguće ocijeniti uporabom algoritma ocjene pridruživanja studenata.

Na Slici **7** prikazan je primjer strogog jednostrukog pridruživanja. U primjeru sve grupe imaju jedno slobodno mjesto. Student A je na svojoj prioritetnoj listi odabrao tri grupe: 1, 2 i 3. Nakon izvođenja algoritma pridružen je u 1. grupu. Student B je na svojoj prioritetnoj listi

odabrao dvije grupe: 1. i 3. grupu. Prva je već pridružena studentu A te na njoj nema mjesta. Zbog toga ga algoritam pridružuje 2. grupi. Treći student je na svojoj prioritetnoj listi stavio samo 1. grupu. Grupa je već pridružena studentu A te mu nije moguće pridružiti niti jednu grupu s prioritetne liste. Zbog toga se pridružuje grupi iz proširene prioritetne liste koja je izrađena usporedbom odabira studenta C sa studentima A i B te se student pridružuje 2. grupi.



Slika 7 – Primjer strogog jednostrukog pridruživanja

Strogo jednostruko pridruživanje može se napraviti:

1. Nakon završetka pridruživanja svih studenata – nakon što su pridruženi svi studenti putem svoje prioritetne liste, studenti koji su ostali nepridruženi pridružuju se u slobodne grupe;
2. Prije pridruživanja idućeg studenta – kada studentu nije moguće pridružiti grupu s prioritetne liste, student se pridružuje u grupu izvan nje.

Prvi način pridruživanja, nakon završetka pridruživanja svih studenata, studentima koji se pridružuju ostavlja samo one grupe koje nisu bile zanimljive ostalim studentima. Time dobri

studenti, a koji su ostali nepridruženi, mogu biti pridruženi samo manje zanimljivim grupama. Zbog toga se u predloženom algoritmu razmatra samo drugi način pridruživanja.

5.3.1.5. Algoritam ocjene strogog jednostrukog pridruživanja

Glavni genetski algoritam kao funkciju dobrote koristi algoritam ocjene strogog jednostrukog pridruživanja (eng. *allocation evaluation algorithm*). Algoritam ocjene dodjeljuje bodove temeljem podataka o rangu grupe na studentovoj prioritetnoj listi te informacije je li mu grupa pridružena s prioritetne liste ili proširene prioritetne liste.

Za studenta kojem je pridružena grupa koja se nalazi na početku njegove prioritetne liste algoritam će pridružiti najveći broj bodova. Pridruživanjem manje prioritetne grupe, dodjeljuje se manji broj bodova. Bodovi se dodjeljuju i za pridruživanje grupama s proširene prioritetne liste, ali su manji od bodova pridruženih grupa s prioritetne liste. Time se osigurava da algoritam prioritetno pridružuje studentu grupu s prioritetne liste, a tek potom s proširene prioritetne liste.

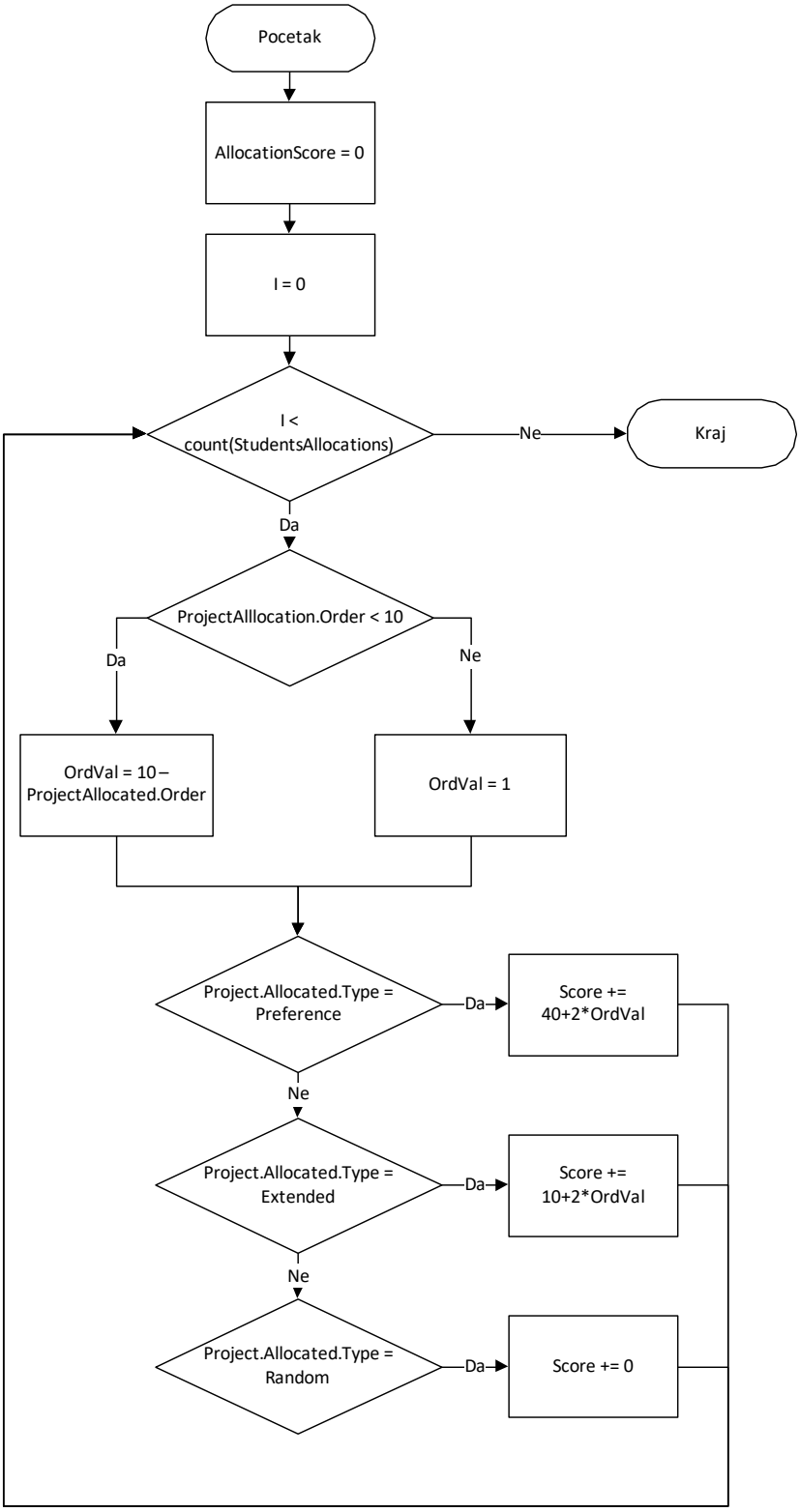
Pridruživanje studenta nasumičnoj grupi ne boduje se, jer se takvo pridruživanje smatra najmanje poželjnim. Algoritam boduje jednakim brojem bodova raspodjelu u grupe koje su pozicionirane na deseto ili niže mjesto na prioritetnim listama.

Algoritam ocjene jednostrukog pridruživanja studenata opisan je Algoritmom 6, a bodovi se računaju kao suma svih bodova za pridruživanje.

Rješenje u kojem je algoritam ocjene jednostrukog pridruživanja studenata izračunao najveći broj bodova smatra se najboljim rješenjem.

```
FUNCTION EVALUATE ALLOCATIONS
  AllocationScore := 0;

  FOREACH Student IN StudentsAllocations DO
    IF Student.GroupAllocated.Order < 10 THEN
      ORDVAL := 10 - Student.GroupAllocated.Order
    ELSE
      ORDVAL := 1
    END IF

    CASE  Student.GroupAllocated.Type OF
      'Preference': AllocationScore += 40+2*ORDVAL;
      'Extended'  : AllocationScore += 10+2*ORDVAL;
      'Random'    : AllocationScore += 0
    END CASE

  END FOREACH
END FUNCTION
```

Algoritam 6 – Pseudokôd algoritma ocjene strogog jednostrukog pridruživanja studenata



Slika 8 – Dijagram tijeka algoritma ocjene strogog jednostrukog pridruživanja studenata

Eksperimentalno je određeno da vrijednosti bodovanja prikazane Tablicom 5 daju najbolje rezultate kada su uspoređene s očekivanim rezultatima najboljeg pridruživanja i onog koje je algoritam označio kao najbolje.

Tablica 5 – Vrijednosti bodova za potrebe algoritma ocjene pridruživanja studenata

Rang	Prioritetna lista		Proširena prioritetna lista	
	Formula	Bodovi	Formula	Bodovi
1	40+2*10	60	10*2*10	30
2	40+2*9	58	10*2*9	28
3	40+2*8	56	10*2*8	26
4	40+2*7	54	10*2*7	24
5	40+2*6	52	10*2*6	22
6	40+2*5	50	10*2*5	20
7	40+2*4	48	10*2*4	18
8	40+2*3	46	10*2*3	16
9	40+2*2	44	10*2*2	14
10+	40+2*1	42	10*2*1	12

Primjer: Za studenta pridruženog grupi koja je rangirana na trećem mjestu na njegovoj prioritetnoj listi dodjeljuje se 56 bodova. Za pridruživanje grupi koju je drugi student rangirao na 12. mjestu na proširenoj prioritetnoj listi dodjeljuje se 12 bodova. Kada bi se promatrala samo ova dva studenta, tada bi ukupna ocjena bila:

56+12=68 bodova.

Drugi radovi iz ovog područja ne upotrebljavaju proširenu prioritetnu listu. Zbog toga oni ocjenjuju rješenje pridruživanja temeljeno samo na rangu grupe u studentovoj prioritetnoj listi. Ovaj algoritam ocjenjivanja proširen je da u svojem bodovanju koristi i rang grupe na proširenoj prioritetnoj listi.

5.3.2. Algoritam višestrukog pridruživanja s glavnim popisom

Algoritam višestrukog pridruživanja s glavnim popisom temelji se na Algoritmu strogog jednostrukog pridruživanja s glavnim popisom opisanim u poglavlju 5.3.1. Za razliku od Algoritma strogog jednostrukog pridruživanja, kod ovog je algoritma istog studenta moguće pridružiti u više grupa te se studenta pridružuje samo u one grupe koje su na njegovoj prioritetnoj listi.

Algoritam višestrukog pridruživanja s glavnim popisom rješava problem u kojem:

- Student treba biti pridružen u najviše N grupa koje su na njegovoj prioritetnoj listi. Broj grupa (N) zadan je odvojeno za svakog studenta. Ovaj način pridruživanja se u ovoj disertaciji naziva *višestruko pridruživanje*;
- Studenti imaju nepotpune prioritetne liste;
- Broj grupa koje je moguće odabrati može biti iznimno velik;
- Broj slobodnih mjesta u grupama može biti iznimno mali, a ukupni broj svih dostupnih mjesta u grupama može biti jednak broju studenata;
- Koristi se glavni popis rangiranih studenata.

Za razliku od Algoritma strogog jednostrukog pridruživanja, u ovom algoritmu ne koristi se kreiranje proširenih prioritetnih lista niti nasumična dodjela grupa, a umjesto algoritma za pridruživanje i ocjenu koriste se ovi algoritmi:

1. *Algoritam višestrukog pridruživanja* koji se temelji na predloženom algoritmu pridruživanja s glavnim popisom te ga proširuje mogućnošću pridruživanja istog studenta u više grupa. Svaki od studenata uz prioritetnu listu određuje i najveći broj grupa na koje može biti pridružen;
2. *Algoritam ocjene višestrukog pridruživanja* koji se temelji na predloženom algoritmu ocjene strogog jednostrukog pridruživanja s time da je proširen mogućnošću bodovanja višestrukog pridruživanja.

5.3.2.1. Algoritam višestrukog pridruživanja

Izvorni algoritam jednostrukog pridruživanja omogućava pridruživanje studenta samo jednoj grupi te osigurava da student uvijek bude pridružen grupi čak iako je nije odabrao. Zbog toga se u ovom algoritmu ne koristi algoritam izrade proširene prioritetne liste niti nasumično pridruživanje grupe studentu.

Algoritam prvom studentu na početku glavnog popisa rangiranih studenata pridružuje željene grupe, a potom prelazi na idućeg s popisa. Svakom studentu algoritam pokušava pridružiti N grupa, pri čemu je N broj grupa u koje student želi biti istovremeno pridružen. Ako niti jedna grupa na prioritetnoj listi nema slobodnog mjesta student će ostati nepridružen – neće se pridruživati nasumičnim odabirom grupe.

Traženi ulazni parametri algoritma su:

1. *AllStudents* (svi studenti) – Prioritetne liste svih studenata;

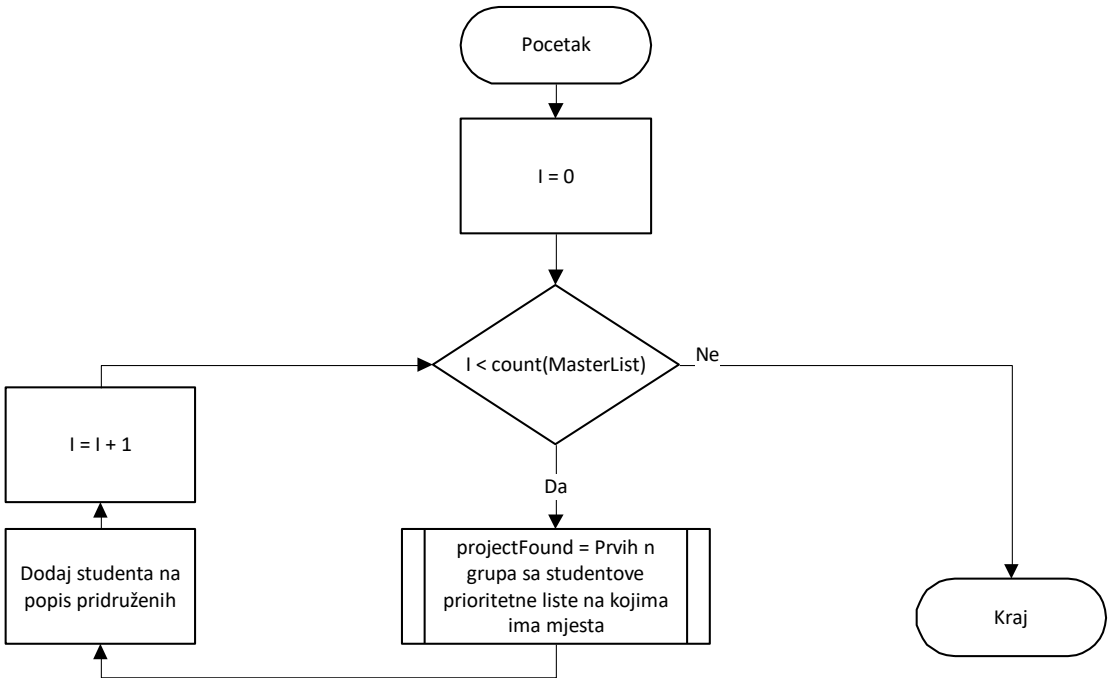
2. *AllGroups* (sve grupe) – Popis svih dostupnih grupa;
3. *MasterList* (glavni popis) – Glavni popis rangiranih studenata.

Izlazni podaci:

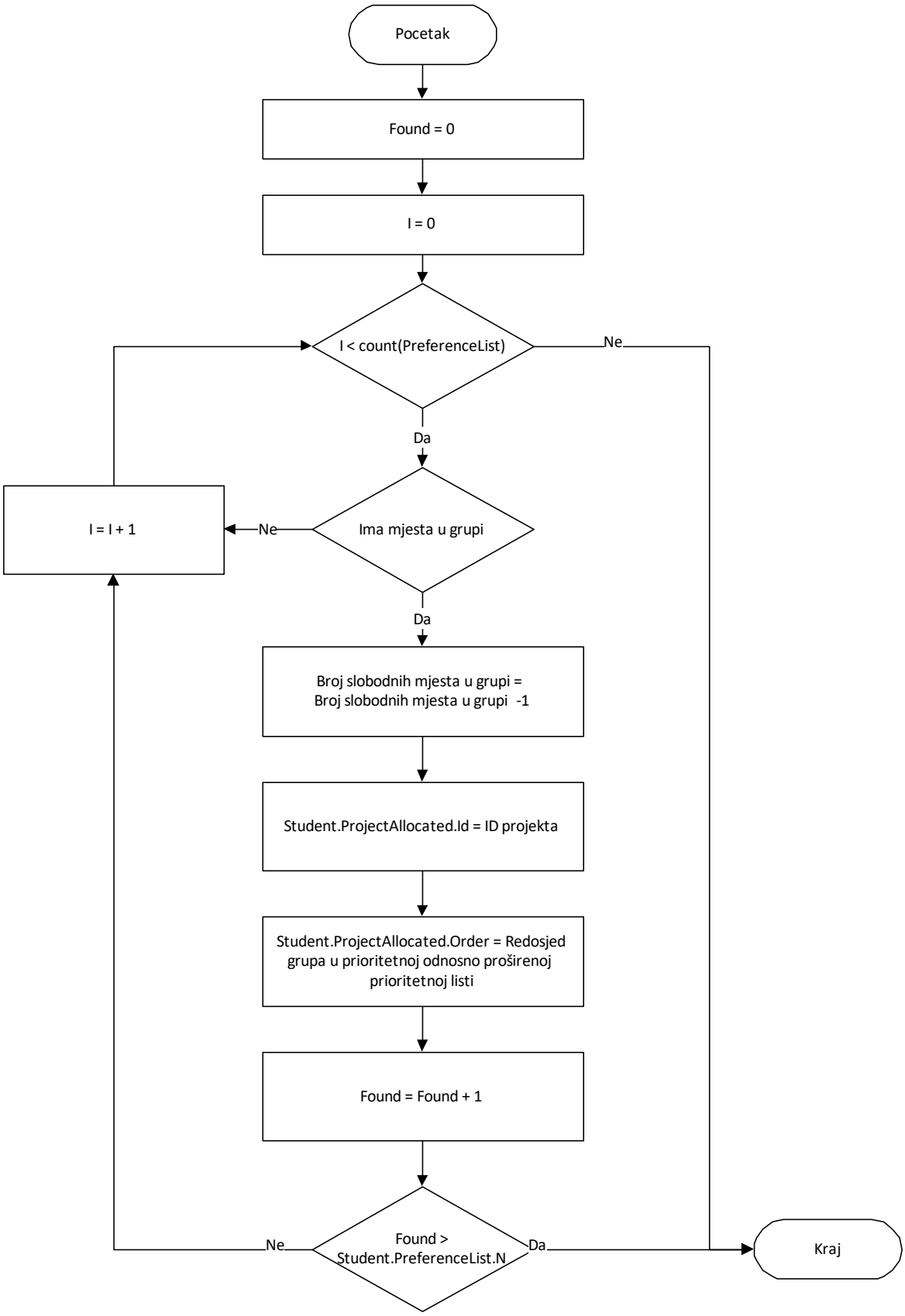
1. *StudentsAllocations* (pridruženi studenti) – Popis pridruženih studenata.

```
FUNCTION ALLOCATE STUDENTS

  FOREACH Student IN MasterList DO
      groupsFound := FIND FIRST N AVAILABLE GROUPS
                      (OUT Student)
      StudentsAllocations[] := Student
  END FOREACH

  RETURN StudentsAllocations

END FUNCTION

FUNCTION FIND FIRST N AVAILABLE GROUPS
  Found := 0
  FOREACH Group IN PreferenceList DO
    IF Group have available places > 0 THEN
      UPDATE AVAILABLE PLACES (Group.ID)
      Student.GroupAllocated[Found].ID := Group.ID
      Student.GroupAllocated[Found].Order := Group.Order
      Found++
    END IF
    IF Found >= Student.PreferenceList.N THEN
       RETURN Found
    END IF
  END FOREACH
END FUNCTION
```

Algoritam 7 – Pseudokôd algoritma višestrukog pridruživanja

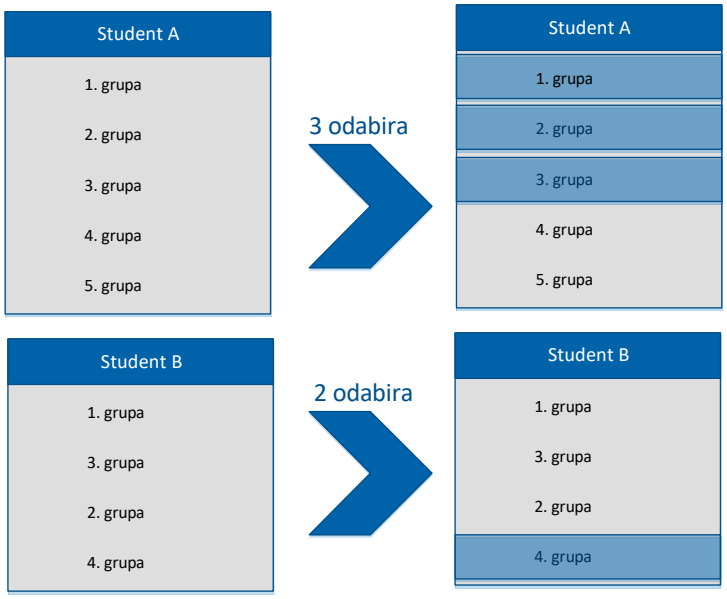


Slika 9 – Dijagram tijeka algoritma višestrukog pridruživanja



Slika 10 – Dijagram tijeka algoritma višestrukog pridruživanja za odabir N grupa

Na Slici **11** prikazan je primjer višestrukog pridruživanja. U ovom primjeru, svaka grupa ima jedno slobodno mjesto. Student A je odabrao prvu do pete grupe te želi biti pridružen u tri grupe. Student A se, kao bolje rangiran, pridružuje u 3 grupe koje je odabrao na početku svoje liste prioriteta. U primjeru su 1, 2 i 3 grupa. Student B odabrao je četiri grupe na svojoj prioritetnoj listi od čega želi biti raspodijeljen u dvije grupe. Prva, druga i treća grupa nemaju više slobodnih mjesta, te se student pridružuje samo grupi 4 iako je odabrao pridruživanje u dvije grupe.



Slika 11 – Primjer višestrukog pridruživanja

5.3.2.2. Algoritam ocjene višestrukog pridruživanja

Glavni genetski algoritam kao funkciju dobrote koristi algoritam ocjene višestrukog pridruživanja. Algoritam ocjene dodjeljuje bodove za svaku grupu pridruženu studentu.

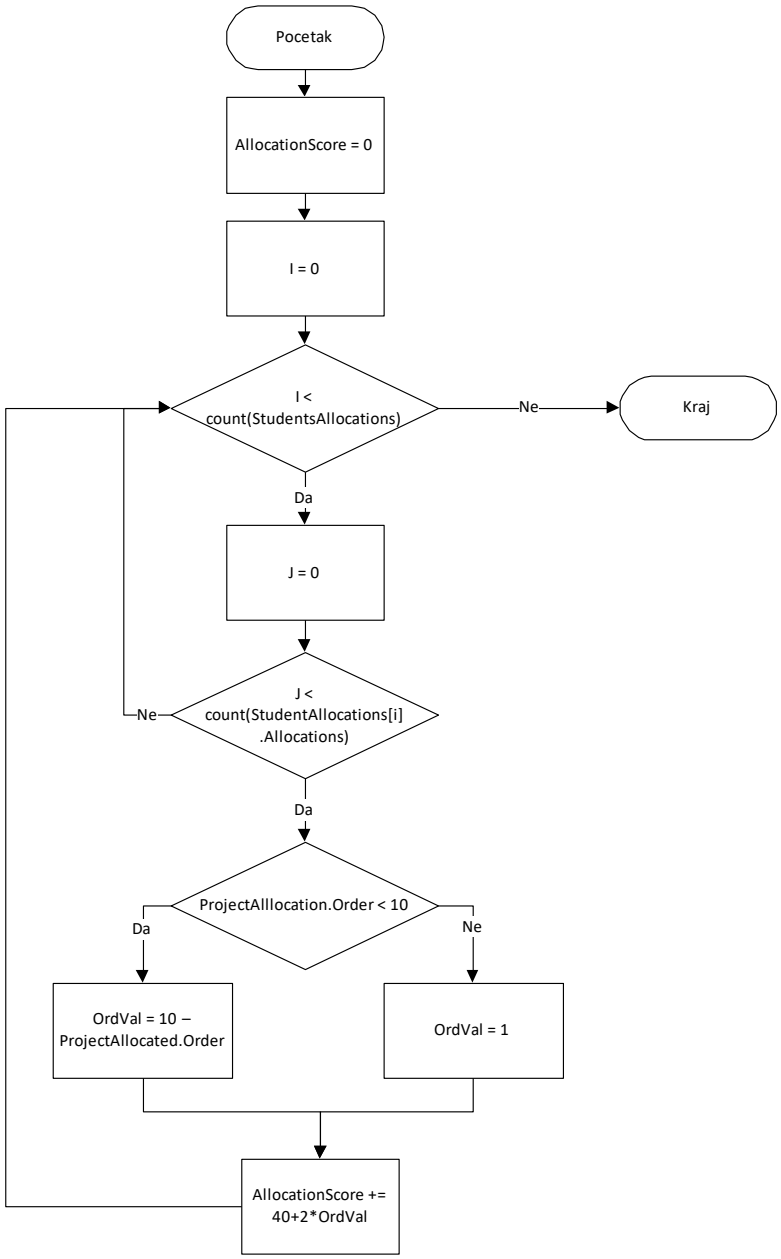
Za one koji su na početku studentove prioritetne liste algoritam će dodijeliti najveći broj bodova. Pridruživanjem niže rangirane grupe, dodjeljuje se manji broj bodova. Ukupni bodovi za jednog studenta jednaki su zbroju bodova za svaku od pridruženih grupa.

Algoritam ocjene višestrukog pridruživanje studenata opisan je Algoritmom 6, a ukupni se bodovi računaju kao suma bodova pridruživanja svih studenata. Rješenje u kojem je algoritam ocjene izračunao najveći broj bodova smatra se najboljim rješenjem.

Za bodove jednog pridruživanja koriste se jednake vrijednosti kao i u algoritmu ocjene strogog jednostrukog pridruživanja studenta za bodovanje grupe temeljem prioritetne liste.

```
FUNCTION EVALUATE N ALLOCATIONS
  AllocationScore := 0;
  FOREACH Student IN StudentsAllocations DO
    FOREACH Allocation IN Student.Allocations DO
      IF Allocation.GroupAllocated.Order < 10 THEN
        ORDVAL := 10 - Allocation.GroupAllocated.Order
      ELSE
        ORDVAL := 1
      END IF
      AllocationScore += 40+2*ORDVAL
    END FOREACH
   END FOREACH
END FUNCTION
```

Algoritam 8 – Pseudokôd algoritma ocjene algoritma višestrukog pridruživanja



Slika 12 – Dijagram tijeka algoritma ocjene algoritma višestrukog pridruživanja

5.3.3. Algoritam višestrukog pridruživanja s glavnim popisom i višeslojnom prioritetnom listom

Algoritam višestrukog pridruživanja s glavnim popisom i višeslojnom prioritetnom listom temelji se na Algoritmu strogog jednostrukog pridruživanja s glavnim popisom opisanim u poglavlju 5.3.1.

Predloženi algoritam višestrukog pridruživanja s glavnim popisom i višeslojnom prioritetnom listom rješava problem pridruživanja u nastavnim procesima u kojima:

- Studenti u odabiru koriste liste u kojima se glavna prioritetna lista sastoji od više prioritetnih lista. Glavna lista se u ovoj disertaciji naziva *višeslojna prioritetna lista*. Svaka prioritetna lista koristi princip *višestrukog pridruživanja*, a glavna prioritetna lista brine o ograničenju ukupnog broja grupa;
- Liste mogu biti nepotpune;
- Broj grupa unutar svake prioritetne liste može biti iznimno velik;
- Koristi se glavni popis rangiranih studenata.

Ovaj algoritam također upotrebljava glavni genetski algoritam. Ne koristi algoritam proširenih prioritetnih lista te se umjesto algoritma za pridruživanje i ocjenu pozivaju sljedeći algoritmi:

1. Algoritam pridruživanja s višeslojnom prioritetnom listom;
2. Algoritam ocjene višestrukog pridruživanja s višeslojnom prioritetnom listom.

5.3.3.1. Algoritam pridruživanja s višeslojnom prioritetnom listom

Ovaj algoritam temelji se na predloženom algoritmu višestrukog pridruživanja s glavnim popisom te ga proširuje mogućnošću pridruživanja temeljem višeslojne prioritetne liste. Višeslojne prioritetne liste sastoje se od niza prioritetnih lista. Pridruživanje za svaku od prioritetnih lista može biti višestruko, a sve prioritetne liste unutar višeslojne prioritetne liste poredane su prioritetno.

Kod višeslojnog pridruživanja, iterativno se prolaze prioritetne liste redom kojim su rangirane na višeslojnoj prioritetnoj listi. Za svaku prioritetnu listu algoritam određuje koje grupe će se pridružiti. Za sve grupe koje su rezultat pridruživanja, brišu se s prioritetnih lista koje su niže rangirane. Uz sve grupe koje su rezultat pridruživanja, bilježi se njihov redni broj na prioritetnoj listi i identifikator prioritetne liste.

Višeslojna prioritetna lista ima ograničenje u ukupnom broju elemenata koji su rezultat pridruživanja. Stoga se pridruživanje provodi samo za prvih N grupa koje su vraćene kao rezultat pridruživanja svih prioritetnih lista.

Ovaj algoritam provodi se iterativno za sve studente, prvo za studenta na početku glavnog popisa rangiranih studenata, potom za sljedećeg studenta na popisu i dalje redom.

Ulazni parametri algoritma su:

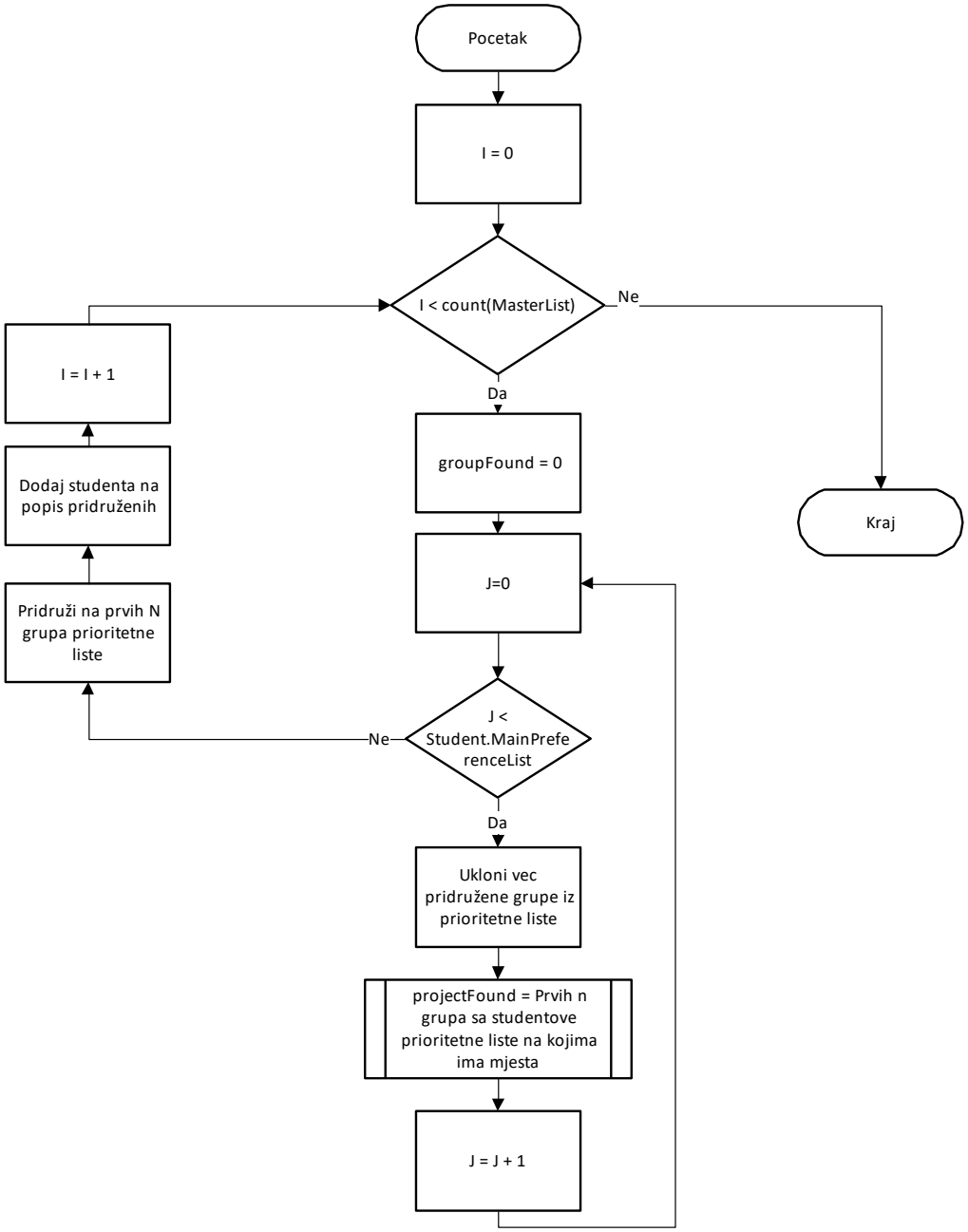
1. *AllStudents* (svi studenti) – Prioritetne liste svih studenata;
2. *AllGroups* (sve grupe) – Popis svih dostupnih grupa;
3. *MasterList* (glavni popis) – Glavni popis rangiranih studenata.

Izlazni podaci:

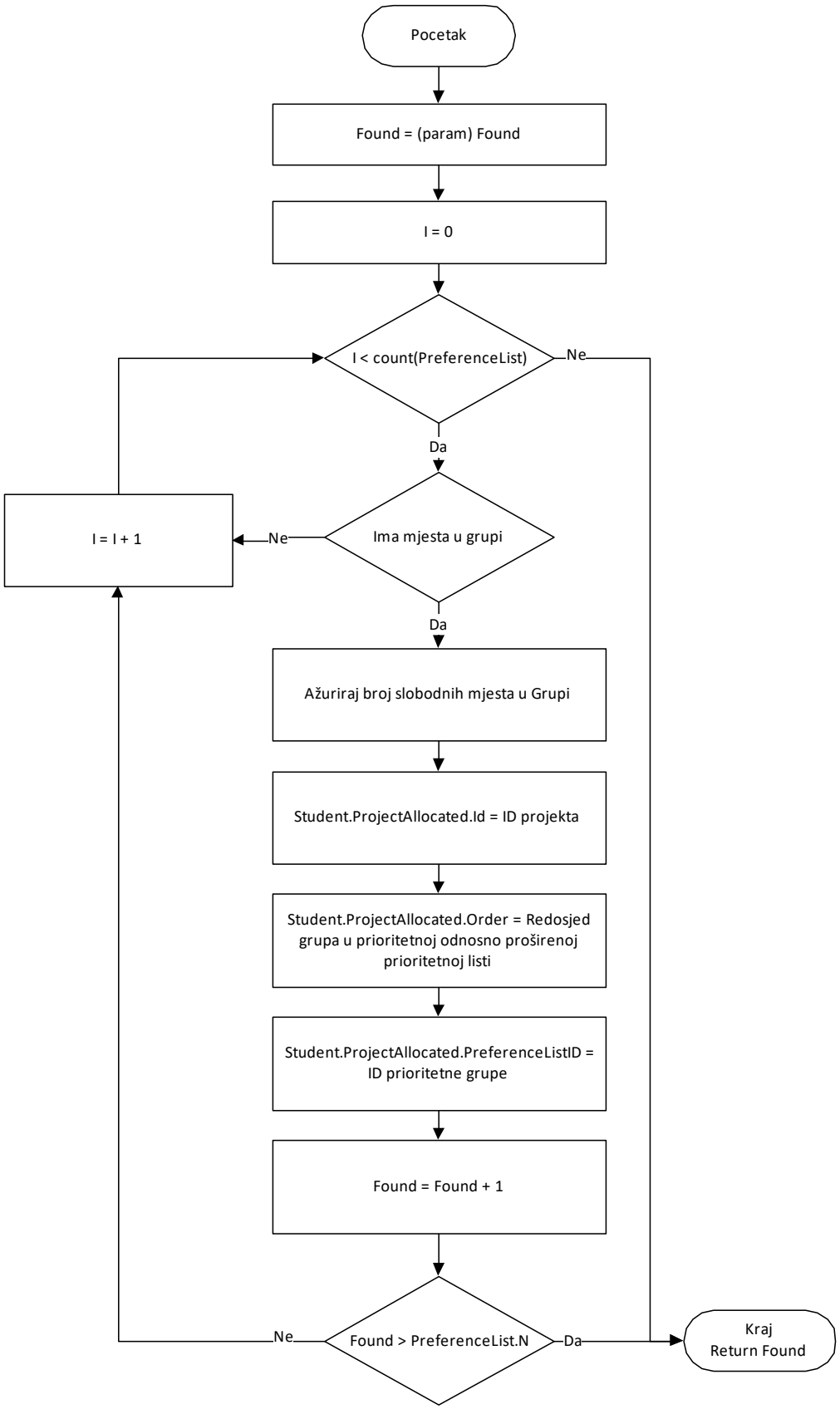
1. *StudentsAllocations* (pridruženi studenti) – Popis pridruženih studenata.

```
FUNCTION ALLOCATE STUDENTS
  FOREACH Student IN MasterList DO
    Found := 0
    FOREACH PreferenceList IN Student.MainPreferenceList DO
      REMOVE ALLOCATED GROUPS FROM PREFERENCE LIST (Student, Preference )
      Found := FIND FIRST N AVAILABLE GROUPS
                        (OUT Student, PreferenceList, Found)
    END FOREACH

    Student.GroupAllocated :=
        Student.groupAllocated.GetFirst(Student.MainPreferenceList.N)
    StudentsAllocations[] := Student
  END FOREACH

  RETURN StudentsAllocations
END FUNCTION

FUNCTION FIND FIRST N AVAILABLE GROUPS

  FOREACH Group IN PreferenceList

    IF GROUP HAVE AVAILABLE PLACES (Group.ID) > 0 THEN
      UPDATE AVAILABLE PLACES (Group.ID)
      Student.GroupAllocated[Found].ID := Group.ID
      Student.GroupAllocated[Found].Order := Group.Order
      Student.GroupAllocated[Found].PreferenceListID := PreferenceList.ID
      Found++
    END IF

    IF Found >= PreferenceList.N THEN
      RETURN Found
    END IF

  END FOREACH
END FUNCTION
```

Algoritam 9 – Pseudokôd algoritma s višeslojnom prioritetnom listom

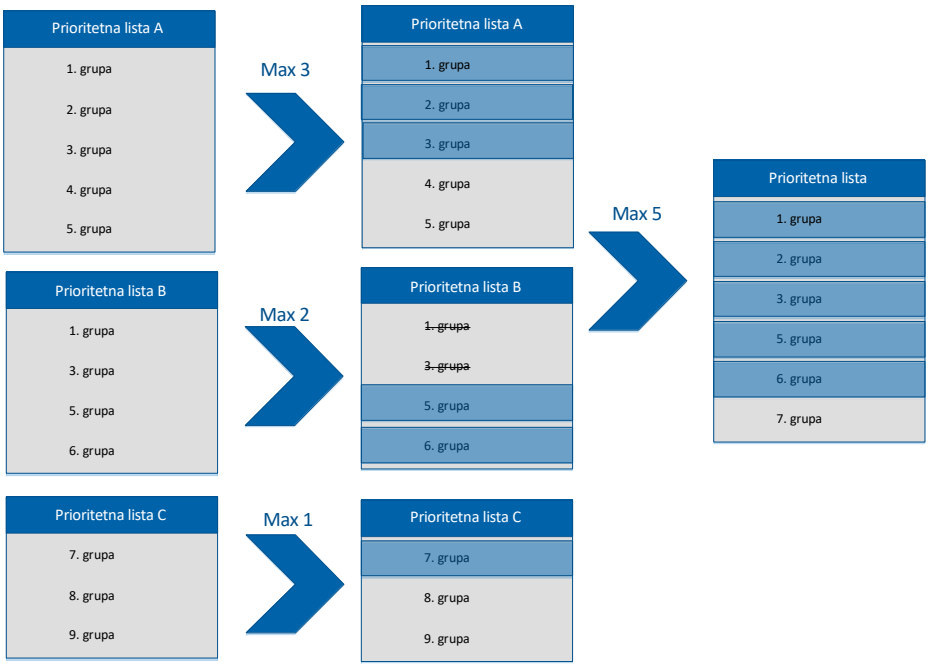


Slika 13 – Dijagram tijeka algoritma s višeslojnom prioritetnom listom



Slika 14 – Dijagram tijeka algoritma s višeslojnom prioritetnom listom – odabir rješenja

Na Slici **15** prikazan je primjer pridruživanja višeslojnom prioritetnom listom. Studentova višeslojna prioritetna lista sastoji se od tri prioritetne liste: A, B i C. Na prioritetnoj listi A odabrano je pet grupa od čega student treba biti pridružen u najviše tri grupe, na prioritetnoj listi B odabrano je četiri grupe od čega student treba biti najviše pridružen u dvije, te na prioritetnoj listi C odabrane su tri grupe od čega student treba biti pridružen u najviše jednu. Ukupno student treba biti pridružen u najviše pet grupa. Algoritam s prve liste studentu pridružuje prve tri grupe, nakon čega s prioritetne liste B i C briše pridružene grupe. U primjeru se brišu 1. i 3. grupa s prioritetne liste B s obzirom na to da su one već pridružene unutar prioritetne liste A. S druge liste, algoritam pridružuje dvije grupe, a s treće prioritetne liste jednu. Ukupno bi studentu bilo pridruženo šest grupa, ali s obzirom na to da je glavnom prioritetnom listom određeno da studentu treba dodijeliti najviše pet grupa sa svih prioritetnih listi, konačno je studentu pridruženo pet grupa.



Slika 15 – Primjer pridruživanja višeslojnom prioritetnom listom

5.3.3.2. Algoritam ocjene višestrukog pridruživanja s višeslojnom prioritetnom listom

Glavni genetski algoritam kao funkciju dobrote koristi algoritam ocjene višestrukog pridruživanja s višeslojnom prioritetnom listom. Algoritam ocjene dodjeljuje bodove jednako kao i u algoritmu ocjene višestrukog pridruživanja s time da se za poredak pridružene grupe

koristi poredak unutar prioritetne liste u koje je grupa pridružena. Bodovi se dodjeljuju neovisno o poretku samih prioritetnih lista unutar višeslojne prioritetne liste.

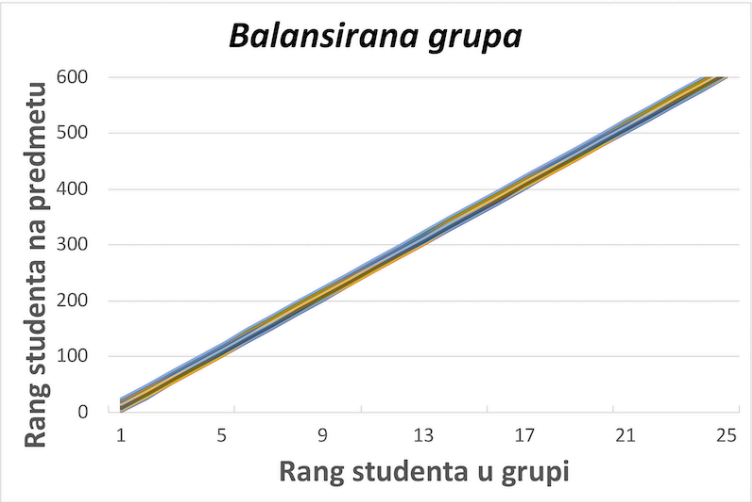
Algoritam ocjene višestrukog pridruživanje studenata opisan je Algoritmom 6.

5.3.4. Algoritam jednostrukog pridruživanja grupiranih elemenata

Za razliku od prethodnih algoritama, *algoritam jednostrukog pridruživanja grupiranih elemenata* rješava problem pridruživanja već grupiranih elemenata, u kojem vrijedi sljedeće:

- Studenti su pridruženi u nedjeljive grupe koje treba pridružiti novim grupama;
- Studenti nemaju prioritetne liste;
- Studenti su pridruženi u grupe s približno jednakim brojem studenata;
- Grupe studenata treba pridružiti u grupe zadane najveće veličine;
- Novoformirane grupe, trebaju sadržavati reprezentativan uzorak studenata svih rangova, od studenata s najvećom prosječnom ocjenom studija do onih s najmanjom. U ovoj disertaciji takve grupe nazivamo *balansirane grupe*.

Na Slici 16 prikazana je željena balansirana grupa. Ukupno 600 studenata pridruženo je u 24 grupe od 25 studenata. Na Y-osi studenti su poredani prema uspjehu unutar ukupnog broja studenata (600). Na X-osi studenti su poredani prema uspjehu unutar grupe (25 studenata). Balansirana grupa sastoji se od jednakog omjera studenata s izvrsnim, dobrim i lošim uspjehom, odnosno od jednog studenta koji je među prvih 24 rangiranih studenata na predmetu, jednog koji je među prvih 48, te na kraju od jednog koji je među zadnjih 24 rangiranih studenata.



Slika 16 – Grafički prikaz balansirane (idealne) raspodjele studenata

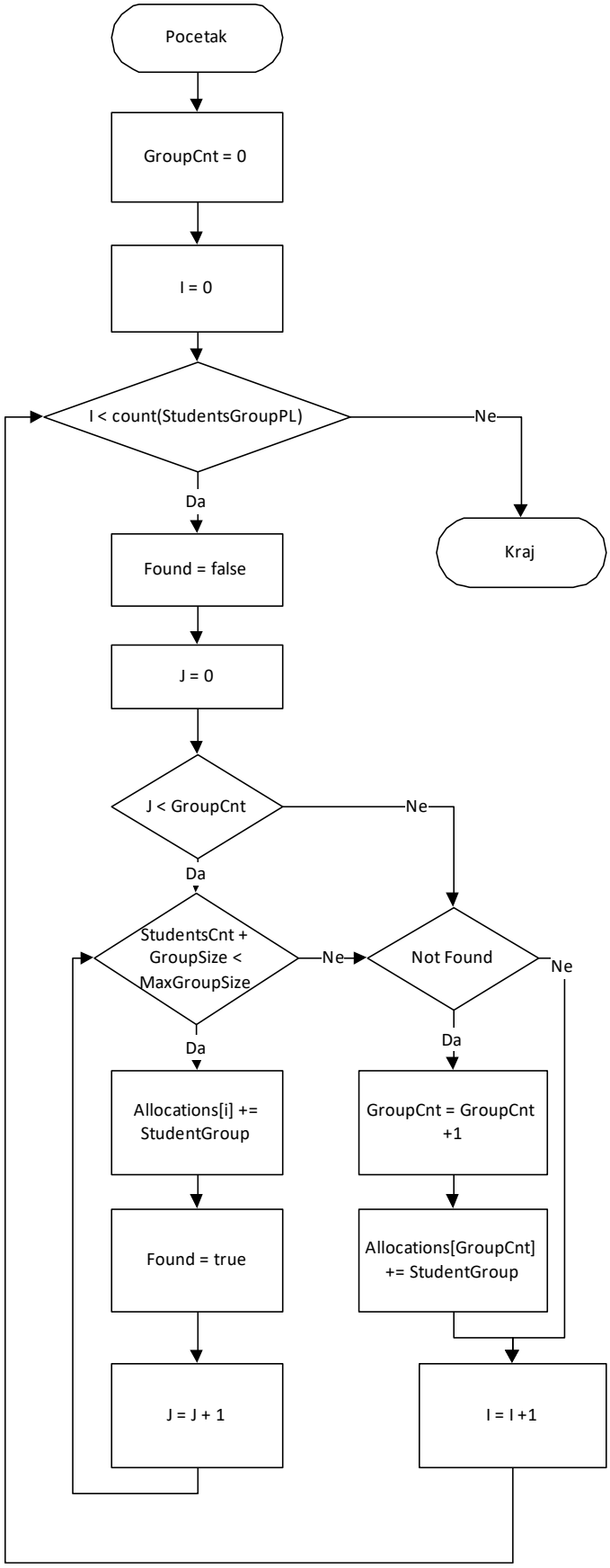
Ovaj algoritam također koristi metodu odabira redoslijeda elemenata pri rješavanju problema pridruživanja s prioritetnim listama u nastavnim procesima uporabom ekspertnog znanja o ocjeni i području grupe.

Prilikom pridruživanja, može se koristiti jednostavan algoritam u kojem se za kreiranje prve prezentacijske grupe uzme toliko poredanih projektnih grupa da broj studenata bude unutar željene veličine grupe. Iduća grupa se formira na isti način, uporabom sljedećih članova grupe.

Algoritam **10** prikazuje pseudokôd ovog algoritma. Ovisno o vrsti odabranog ekspertnog znanja algoritam se poziva s drugačije poredanim grupama na prioritetnoj listi.

```
FUNCTION ALLOCATE STUDENTS
  GroupCnt := 0
  //Go throught master list
  FOREACH StudentGroup IN MasterList DO

    Found := false
    FOR (i=0; i<GroupCnt ;i++)
      IF StudentsCnt+Groupsize < MaxGroupSize THEN
        Allocations[i] += StudentGroup
        Found := true
        BREAK
      END IF
    END FOR

    IF NOT Found THEN
      GroupCnt ++
      Allocations[GroupCnt] += StudentGroup
    END IF
  END FOREACH

  SAVE Allocations
END FUNCTION
```

Algoritam 10 – Pseudokôd algoritma jednostrukog pridruživanja grupiranih elemenata



Slika 17 – Dijagram tijeka algoritma jednostrukog pridruživanja grupiranih elemenata

6. MODEL RASPODIJELJENOG RJEŠAVANJA PROBLEMA PRIDRUŽIVANJA

Svi predloženi algoritmi s funkcijom ocjene algoritma predviđaju mogućnost raspodijeljenog rješavanja [55]–[59]. Model raspodijeljenog rješavanja problema pridruživanja u nastavnim procesima izvodi se u okruženju klijentske arhitekture. Tako se zadatak može raspodijeliti na više računala i može se doći do boljeg rješenja u kraćem vremenu.

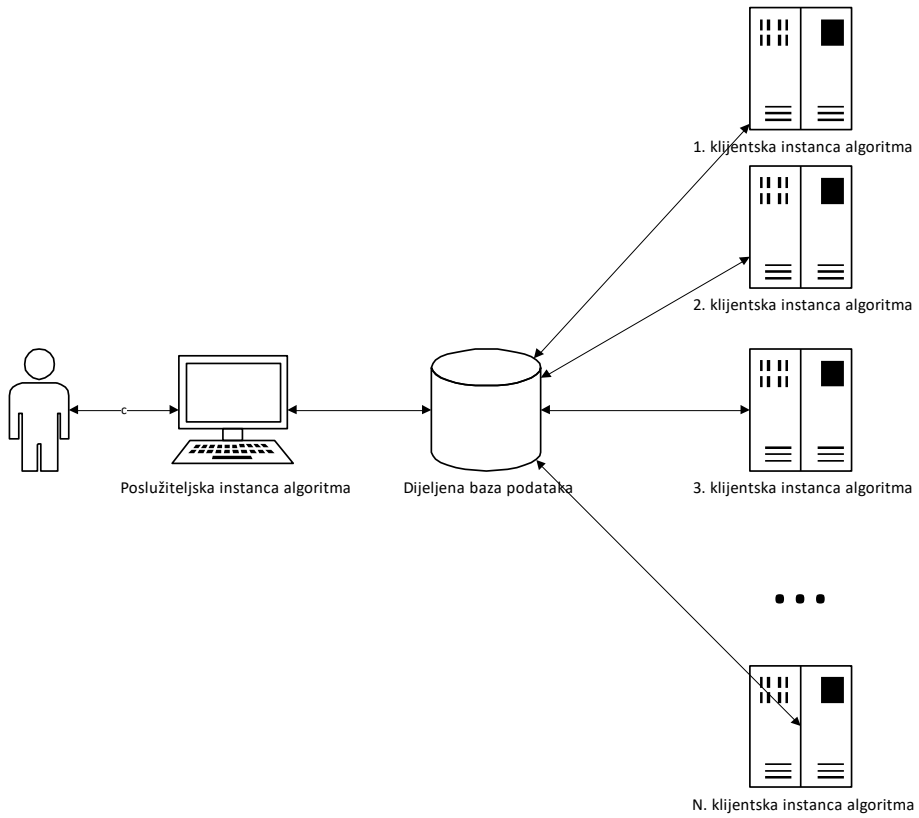
U predloženom modelu, podaci se razmjenjuju putem dvosmjerne veze za koji se može koristiti zajednička baza podataka. Poslužitelj priprema podatke i parametre za izvršavanje algoritma, dok se na svakom klijentu izvodi genetski algoritam koji određuje vlastito najbolje rješenje.

Broj klijenata nije ograničen dizajnom, a svaki od njih prilikom pokretanja izračuna preuzima podatke o zadatku i parametrima izvršavanja. Poslužiteljska verzija aplikacije (eng. *server*) prilikom pokretanja utvrđuje postoje li sve potrebne tablice u bazi, te ih po potrebi kreira. Postojeći podaci se brišu.

Klijentske verzije moguće je pokrenuti u bilo kojem trenutku, a one periodično provjeravaju jesu li se pojavili novi zadaci izračuna. U trenutku kada poslužitelj postavi oznaku spremnog zadatka, klijentska računala preuzimaju zadatak te pokreću glavni genetski algoritam koji traži najbolje rješenje. Glavni genetski algoritam po završetku izvođenja sprema najbolje rješenje u bazu podataka koja se koristi za razmjenu informacija, ako je pronađeno rješenje bolje od onih koja su do sada pronađena.

Na Slici 18 prikazan je način razmjene podataka u raspodijeljenoj programskoj potpori. U prikazu lijevo korisnik zadaje zahtjev za izračun poslužitelju. Poslužitelj preuzima i priprema podatke te ih sprema u dijeljenu bazu podataka. Klijenti, prikazani s desne strane baze podataka, aktiviraju se, obrađuju podatke i spremaju najbolje rješenje koje poslužitelj isporučuje korisniku. Klijenti također preuzimaju trenutno najbolje rješenje koje mogu koristiti kako bi poboljšali svoje.

Cijeli algoritam može se izvoditi na više procesora, ako se svaki poziv glavnog genetskog algoritma pozove u novoj dretvi (eng. *threads*). Kao najbolje rješenje u bazu podataka sprema se najbolje dobiveno izvođenjem svih dretvi.



Slika 18 – Grafički prikaz razmjene podataka u raspodijeljenoj programskoj potpori

Ulazni parametri algoritma su:

- *Server* – koristi li se algoritam kao poslužitelj ili klijent;
- *DatabaseAddress* – IP adresa baze podataka putem koje se razmjenjuju podaci;
- *DatabasePort* – Port na kojem se vrti baza podataka;
- *DatabaseName* – Naziv baze;
- *DatabaseLogin* – Korisničko ime za spajanje na bazu;
- *DatabasePassword* – Lozinka za spajanje na bazu;
- *TimeOut* – Vrijeme za čekanje do dolaska najboljeg rješenja, -1 označava čekanje završetka rada svih klijenata.

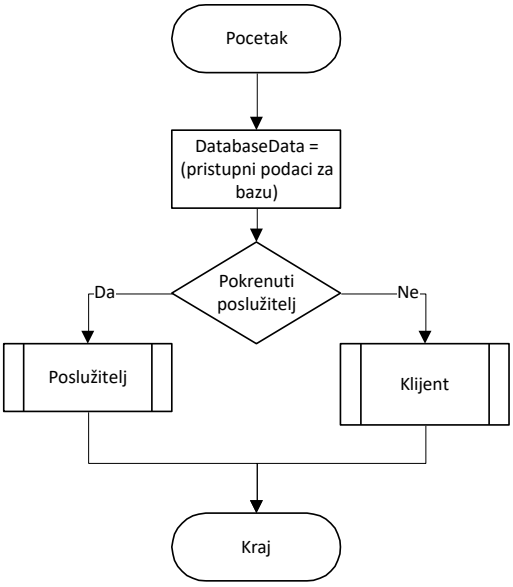
Uz ove podatke algoritam učitava i podatke iz datoteka na disku o:

- Studentima;
- Prioritetnim listama grupa za svakog studenta;
- Grupama.

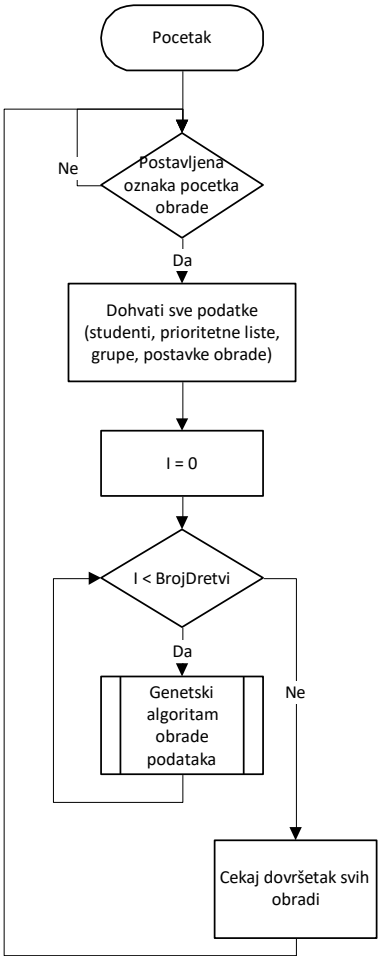
Algoritam 11 prikazuje pseudokôd raspodijeljenog rješavanja problema pridruživanja u nastavnim procesima. Prikazani algoritam može se pozvati u poslužiteljskom ili klijentskom načinu rada. Algoritam najbolje rješenje sprema na disk te ga ispisuje na zaslon.

```
 DatabaseData :=
     {DatabaseAddress, DatabasePort, DatabaseName, DatabaseLogin, DatabasePassword}

//Check if working in Server or Clientmode
IF (Server == true) THEN
    SERVER (DatabaseData)
ELSE
    CLIENT (DatabaseData)
END

FUNCTION SERVER
  //Check if exists table in database, and create tables if not
  CREATE TABLES (DatabaseData)

  //Load all data into database
  LOAD STUDENTS DATA INTO DATABASE (Students, DatabaseData)
  LOAD STUDENT PREFERENCE LISTS DATA INTO DATABASE (Students, DatabaseData)
  LOAD GROUPS DATA INTO DATABASE (Groups, DatabaseData)
  LOAD SETTINGS INTO DATABASE (DatabaseData)

  SET READY SIGNAL FOR CLIENTS (DatabaseData)

  //Waiting for best Solution to Calculate.
  //Clients work in endless loop, so we wait for TimeOut time
  WAIT (10 sec)

  WAIT ALL CLIENTS FINISH (DatabaseData)

  //Show best Solution data.
  BestSolution := GET BEST SOLUTION FROM DATABASE (DatabaseData)
  SAVE TO DISK (BestSolution)
  PRINT (BestSolution)

END FUNCTION

FUNCTION CLIENT
  //Run in Continuoous loop
  WHILE (true)

    //Check if new data exist for calculate
    IF (!READY SIGNAL (DatabaseData))
      WAIT (600)
      CONTINUE
    END IF

    //If exist new data, calculate solution using the main genetic algorithm
    AllInputData := GET ALL DATA (DatabaseData)

    FOR (i=0; i<NumberOfThreads; i++)
      MAIN GENETIC ALGORITHM (DatabaseData, AllInputData)
    END FOR

  END WHILE
END FUNCTION
```

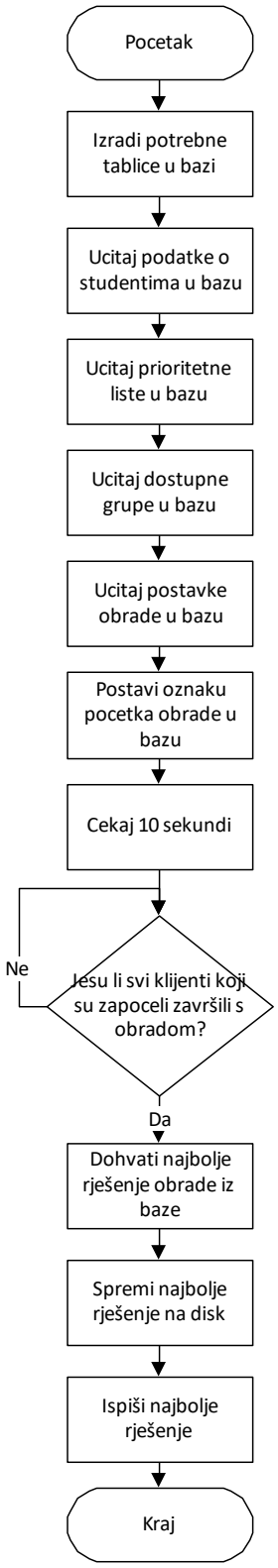
Algoritam 11 – Pseudokôd algoritma za izvođenje u raspodijeljenoj programskoj podršci



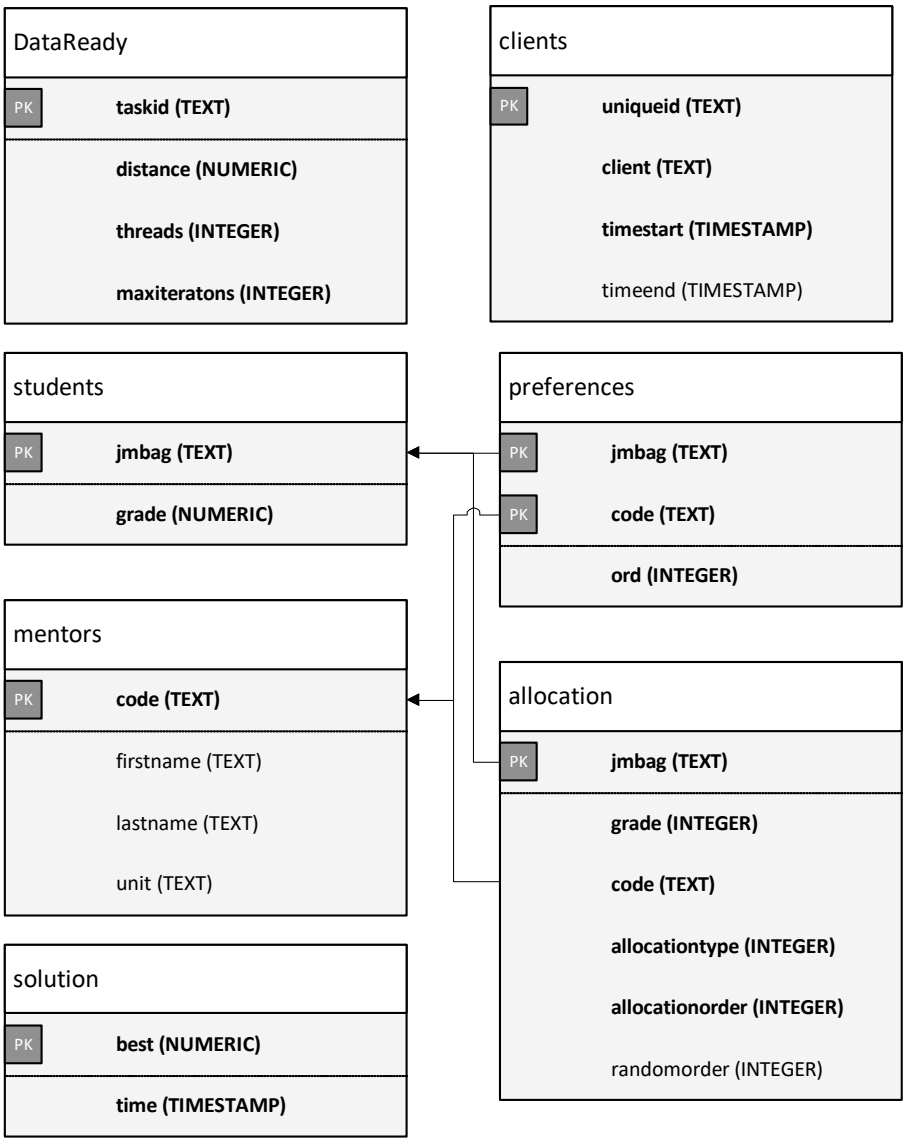
Slika 19 – Dijagram tijeka algoritma u raspodijeljenoj programskoj podršci – glavni



Slika 20 – Dijagram tijeka algoritma u raspodijeljenoj programskoj podršci – klijent



Slika 21 – Dijagram tijeka algoritma u raspodijeljenoj programskoj podršci – poslužitelj



Slika 22 – Dijagram baze podataka za razmjenu podataka između poslužitelja i klijenata

7. PROTOTIP PROGRAMSKOG SUSTAVA

Prototip programskog sustava rješavanja problema pridruživanja s prioritetnim listama koristi nove metode odabira redoslijeda elemenata i pridruživanja nepridruženih elemenata te model raspodijeljenog rješavanja problema pridruživanja.

Izrađen je u programskom jeziku C# koristeći programsko okruženje .Net Core 2.1. Izvedba omogućava izvršavanje programskog sustava na platformama MS Windows, MacOS i GNU/Linux bez potrebe izmjene istog. U raspodijeljenom načinu rada prototip algoritma koristi bazu PostgreSQL, a razmjena podataka provodi se putem programske biblioteke Npgsql.

Prototip je optimiran u cilju racionalne upotrebe memorije, te se pri njegovom izvođenju koristi oko 13 MB memorije, za izračun pridruživanja više od 500 studenata u više od 200 grupa uz više od 3.000 studentskih odabira. Vrijeme rada prototipnog algoritma za navedeni broj studenata, grupa i odabira je 200 iteracija / sekundi na standardnom PC računalu u produkcijskom načinu rada (eng. *release*), dok rad u razvojnom (eng. *debug*) uspori izračun na oko 100 iteracija / sekundi za svaku dretvu odvojeno pri čemu broj dretvi (eng. *thread*) ne prelazi broj dostupnih jezgri računala (eng. *CPU core*).

7.1. Načini rada prototipa programskog sustava

Prototip je moguće pokrenuti na tri načina:

- Samostalan rad (parametar pokretanja: -*noserver*);
- Kao poslužitelj (parametar pokretanja: -*client*);
- Kao klijent (parametar pokretanja: -*server*).

Samostalan rad prototipa namijenjen je okolini u kojoj je dostupno samo jedno računalo ili se ne želi koristiti paralelan rad na više računala. U samostalnom pokretanju moguće je parametrima navesti broj paralelnih dretvi u kojima će se izvoditi (parametar pokretanja: -*threads*), broj iteracija (parametar pokretanja: -*iterations*) te dozvoljenu udaljenost (parametar pokretanja: -*distance*).

Pokretanje prototipa kao poslužitelja omogućava korištenje u raspodijeljenom okruženju. Pri tome samo jedna instanca može biti poslužitelj, a broj klijenata nije ograničen. U poslužiteljskom radu prototip priprema dijeljenu bazu podataka te u nju pohranjuje sve potrebne podatke obrade. Nakon što klijenti završe sa svojim radom odnosno po isteku predviđenog vremena za izračun, ovisno o postavkama prototipa, ispisat će se najbolje rješenje

koje ostaje zapisano i u zajedničkoj bazi podataka. Poslužiteljska instanca programa koristi minimalne resurse računala s obzirom na to da ne provodi nikakve izračune.

Klijentski rad prototipa se od prototipa u samostalnom radu razlikuje u načinu učitavanja podataka. Dok se kod samostalnog rada programa podaci učitavaju iz lokalne datoteke u klijentskom načinu se i podaci i parametri rada prototipa preuzimaju iz zajedničke baze podataka. Pronalaskom najboljeg rješenja ono se sprema u zajedničku bazu podataka, a ne u lokalnu datoteku kao što je to slučaj kod samostalnog rada programa.

7.2. Struktura podataka

Podaci su u memoriji prototipa pohranjeni u dva oblika:

1. *Hash lista* ulaznih podataka;
2. Jednodimenzionalno polje strukture rješenja.

Hash lista ulaznih podataka organizirana je u dva svojstva:

(i) *_students* – Podaci o studentima i njihovim odabirima;

(ii) *_mentors* – Podaci o grupama u koje ih se pridružuje.

Jednodimenzionalno polje strukture rješenja organizirano je kroz strukturu:

(iii) *SingleSolutionSimple* – Podaci ocjene rješenja te polje studenata, grupa i podataka pridruživanja studenta grupi.

7.2.1. Struktura _students

Struktura *_students* organizirana je kao hash lista kojoj je ključ JMBAG studenta dok je vrijednost predstavljena klasom *StudentsClass* s elementima:

- *Jmbag* (string) – JMBAG studenta;
- *Grade* (double) – Učitana ocjena studenta;
- *StudentId* (uint) – Jedinstveni identifikator studenta u obliku cijelog broja bez predznaka;
- Dvije hash liste (prioritetna lista i proširena prioritetna lista) kojima je ključ redni broj odabir, a vrijednost jedinstvena oznaka grupe.

Nakon učitavanja podataka iz datoteke u strukturu *_student* programski sustav određuje:

- *StudentId* – jedinstveni identifikator studenta u algoritmu. Studenti se poredaju prema jedinstvenom identifikatoru – JMBAG, te se redom dodjeljuju identifikatori iz skupa 0, 1, 2, 3, … N-1, gdje je N ukupni broj učitanih studenata;
- Hash lista – proširena prioritetna lista (*PreferenceExtended*) – izračunava se za svakog studenta odvojeno, temeljem izračuna Algoritma izrade proširene prioritetne liste.

Primjer deklaracije strukture *StudentsClass* kao klase u programskom jeziku C# prikazan je na primjeru Programskog kôda 1.

```
public class StudentsClass {
    public string Jmbag { get; set; }
    public double Grade { get; set; }

    public uint StudentID { get; set; }

    public Dictionary<int, string> Preferences { get; set; }
    public Dictionary<int, string> PreferencesExtended { get; set; }
}
```

Programski kôd 1 – Struktura klase *StudentsClass*

7.2.2. Struktura *_mentors*

Struktura *_mentors* organizirana je kao hash lista kojoj je ključ jedinstvena oznaka grupe dok je vrijednost predstavljena klasom *MentorsClass* s elementima:

- *Code* (string) – jedinstvena oznaka grupe;
- *FirstName* (string) – naziv grupe, neobavezno polje, ne koristi se njena vrijednost u izračunu i obradi;
- *LastName* (string) – opis grupe, neobavezno polje, ne koristi se njena vrijednost u izračunu i obradi;
- *Unit* (string) – oznaka pripadnosti, neobavezno polje, ne koristi se njena vrijednost u izračunu i obradi;
- *MentorId* (uint) – jedinstveni identifikator grupe u algoritmu u obliku cijelog broja bez predznaka, a određuje se tako što se poredaju sve jedinstvene oznake grupe abecedno te se redom dodjeljuju identifikatori iz skupa 0, 2, 3, … N-1, gdje je N ukupni broj grupa.

Primjer deklaracije strukture *MentorsClass* kao klase u programskom jeziku C# prikazan je na primjeru Programskog kôda 2.

```
public class MentorsClass {
    public string Code { get; set; }

    public string FirstName { get; set; }
    public string LastName { get; set; }
    public string Unit { get; set; }
    public uint MentorId { get; set; }
}
```

Programski kôd 2 – Struktura klase *MentorsClass*

7.2.3. Polje rješenja

Svako rješenje sprema se u strukturi *SingleSolutionSimple* koja predstavlja kromosom genetskog algoritma. Izvedba strukture u programskom jeziku C# prikazana je na primjeru Programskog kôda 3, a sastoji se od dva dijela:

a) *StudentOrder* (StudentIdGrade[]) – jednodimenzionalno polje strukture StudentIdGrade koje se sastoji od jedinstvenog poretka studenata s njihovim ocjenama i dodjelama grupa;

b) *SolutionGrade* (*float*) – ukupna ocjena rješenja.

```
public struct SingleSolutionSimple
{
    public StudentIdGrade[] StudentOrder;
    public float SolutionGrade;
}
```

Programski kôd 3 – Struktura klase *SingleSolutionSimple*

Izvedba strukture *StudentIdGrade* prikazana je na primjeru Programskog kôda 4, a sadrži elemente:

- *StudentId* (*uint*) – jedinstveni identifikator studenta u algoritmu;
- *StudentGrade* (*int*) – prosječna ocjena studenta prikazana kao cijeli broj. Prosjek je uvećan za 100 puta. Ako je student imao prosjek 4,45 tada *StudentGrade* iznosi 455, a za ocjenu 5 iznosi 500. Najmanji prosjek je 2 odnosno najmanji *StudentGrade* iznosi 200;
- *MentorId* (*uint*) – jedinstveni identifikator studenta u algoritmu;
- *AssignmentType* (*uint*) – vrsta pridruživanja studenta, za pridruživanje s prioritetne liste vrijednost je 1, s proširene prioritetne liste je 2, a za slučajni odabir 3;
- *AssignmentOrder* (*uint*) – redoslijed kojim je studentu pridružena grupa, ako je pridružena grupa na početku prioritetne liste tada je vrijednost 0, ako je druga tada je 1 i tako dalje;
- *RandomId* (*uint*) – nasumični identifikator svakog studenta.

```
public struct StudentIdGrade
{
    public uint StudentId;
    public int StudentGrade;
    public uint MentorId;
    public uint AssignmentType;
    public uint AssignmentOrder;
    public uint RandomId;
}
```

Programski kôd 4 – Struktura *StudentIdGrade*

7.3. Optimizacija brzine izvođenja prototipne izvedbe algoritma

7.3.1. Prilagodba organizacije podataka u memoriji

U prototipu su podaci organizirani na način kojim se osigurava najmanji broj zauzimanja i oslobađanja memorije. Time se značajno pridonosi brzini izvođenja.

Memorija se zauzima dinamički, korištenjem hash lista, samo prilikom jednokratnog učitavanja podataka o studentima, grupama i prioritetnim listama. Nakon toga se sva potrebna memorija zauzima samo jednom, te se iste memorijske lokacije koriste za sva kasnija izvođenja algoritma. Za izvedbu navedenog koriste se jednodimenzionalne strukture pohrane pojedinog rješenja u polju.

Ovaj način osigurao je gotovo dvostruko brže izvođenje u odnosu na implementaciju u kojoj se memorija cijelo vrijeme zauzimala dinamički. Dodatno ubrzanje postignuto je korištenjem cijelih brojeva za oznake studenata i grupa umjesto tekstualnog polja kojim se uobičajeno predstavljaju – u ovom slučaju ubrzanje je posljedica brže usporedbe podataka u računalu. Ovo ubrzanje nije bilo toliko značajno u odnosu na prethodno.

7.4. Programska izvedba prototipa

7.4.1. Izvedba glavnog algoritma

Prototip pokreće glavni genetski algoritam koji je implementiran u programskom jeziku C#, prema opisanom pseudokôdu u poglavlju 5.3.1.1., a sastoji se od sljedećih koraka:

1. Inicijalizacija memorije i izrada početne populacije;
2. Izvođenje algoritma u petlji, odnosno zadani broj iteracija.

U svakom koraku petlje algoritam:

1. Odabire najbolja dva rješenja (roditelje);
2. Ako je najbolje rješenje bolje od dosadašnjeg najboljeg, sprema ga kao najbolje;
3. Dva najbolja rješenja poreda prema jedinstvenim oznakama studenata. U procesu mutacije i nasljeđivanja, dva elementa jednodimenzionalne liste moraju biti pozicionirani na istim mjestima zbog optimizacije vremena izvođenja;
4. Za sva preostala rješenja osim prva dva postupkom mutacije i nasljeđivanja izrađuju se nova potencijalna rješenja.

Izvedba glavnog algoritma u sklopu prototipne izvedbe algoritma prikazana je na primjeru Programskog kôda 5. Prikazani kôd sadrži i dio algoritma koji se odnosi na klijentski način rada, a koji se aktivira kada je definirana varijabla *_clientMode*. Rad u klijentskom načinu rada nešto je sporiji u odnosu na samostalni rad programa zbog dodatnog vremena pohrane podataka u bazu podataka.

```
private void ThreadSolution(double distance, int? threadId, int maxIterations) {
    // Broj generiranih rješenja
    const int generatedSolutions = 15;

    // Inicijalizacija polja rješenja
    var studentSolutions = new SingleSolutionSimple[generatedSolutions];
    for (var i = 0; i < generatedSolutions; i++){
        studentSolutions[i].StudentOrder = new StudentIdGrade[_students.Count];
        studentSolutions[i].SolutionGrade = -1;
    }

    // Početna populacija - generatedSolutions različitih rješenja
    for (var i = 0; i < generatedSolutions; i++){
        studentSolutions[i].StudentOrder = GetRankedListOfStudents(distance,
                                                studentSolutions[i].StudentOrder);
        studentSolutions[i] = GenerateNewSingleSolution(studentSolutions[i]);
    }

    var iteration = 1;
    if (_iteration[threadId ?? 0] == null){
        _iteration[threadId ?? 0] = new IterationValue
            {Iteration = iteration};
        for (var i = 0; i < generatedSolutions; i++){
            _iteration[threadId ?? 0].ScoreValue.Add(studentSolutions[i].SolutionGrade);
        }
    }

    // Ovo je sada beskonačna petlja
    while (true){
        _mutex.WaitOne();
        // Odaberi dva najbolja rješenja i njih zadrži
        if (studentSolutions[0].SolutionGrade > _bestSolutionSimple.SolutionGrade)
        {
            _bestSolutionSimple = studentSolutions[0];
            ShowBestSolution(iteration, _bestSolutionSimple, threadId);

            // Ako se koristi kao klijentski način onda ažuriraj i na serveru najbolje rješenje
            if (_clientMode)
            {
                // Ako je bolje od lokalnog najboljeg onda provjeri je li bolje i od serverskog best
                var remoteBest = ClientGetBestSolutionValue();

                // Ako je izračunato bolje od serverskog spremi ga i na server
                if (studentSolutions[0].SolutionGrade > remoteBest)
                {
                    ClientSaveBestSolution(_bestSolutionSimple, _studentCode, _mentorCode);
                }
            }
        }
        _mutex.ReleaseMutex();

        // Odabir dva najbolja rješenja
        GetBestSolutions(studentSolutions, generatedSolutions);

        // Poredaj najbolja dva rješenja prema studentId (da se mogu uspoređivati)
        Array.Sort(studentSolutions[0].StudentOrder, (i1, i2) =>
                                                i1.StudentId.CompareTo(i2.StudentId));
        Array.Sort(studentSolutions[1].StudentOrder, (i1, i2) =>
                                                i1.StudentId.CompareTo(i2.StudentId));

        // Za ostale generiraj rješenja prema djeci
        for (var i = 2; i < generatedSolutions; i++){
            studentSolutions[i].StudentOrder =
                GetRankedListOfStudentsFromParents(distance, studentSolutions[0].StudentOrder,
                    studentSolutions[1].StudentOrder, studentSolutions[i].StudentOrder,
                    (iteration % 300 == 0 && i % 2 == 0) ? 80 : 10);
            studentSolutions[i] = GenerateNewSingleSolution(studentSolutions[i]);
        }

        iteration++;
        _iteration[threadId ?? 0].Iteration = iteration;
        _iteration[threadId ?? 0].Iteration = iteration;
        for (var i = 0; i < generatedSolutions; i++){
            _iteration[threadId ?? 0].ScoreValue[i] = studentSolutions[i].SolutionGrade;
        }

        if (iteration > maxIterations)
            break;
    }

    // Snimi u bazu generiranih rješenja
    SaveAssignmentsSimple(iteration, studentSolutions[0], threadId ?? 0);
}
```

Programski kôd 5 – Glavni genetski algoritam

7.4.2. Izvedba izrade proširene prioritetne liste

Glavni (genetski) algoritam, za sve studente poziva izradu proširene prioritetne liste. Prototipna izvedba algoritma uključuje izradu proširene prioritetne liste u skladu s algoritmom koji definira pseudokod. Programski kôd 6 izvodi se za svakog studenta odvojeno i određuje proširenu prioritetnu listu odabranog studenta.

```
private void ExtendedPreferenceListForStudent(StudentsClass student)
{
    // Ako je prazna lista onda ništa
    if (student.Preferences.Count == 0) return;

    double varukScore;
    var newMentors = new Dictionary<string, double>();
    var factors = new List<FactorClass> {
        new FactorClass {N = 1, Factor = 0.3},
        new FactorClass {N = 3, Factor = 0.28},
        new FactorClass {N = 5, Factor = 0.25},
        new FactorClass {N = _mentors.Count, Factor = 0.17}
    };

    // Inicijalizacija nove liste mentora
    foreach (var ment in _mentors) newMentors.Add(ment.Value.Code, 0);

    // Iz popisa mičemo sve koje je već odabrao na svojoj listi
    foreach (var prefs in student.Preferences) newMentors.Remove(prefs.Value);

    foreach (var studCompare in _students) {
        // Ne uspoređuj sam sa sobom
        if (studCompare.Key == student.Jmbag) continue;
        if (studCompare.Value.Preferences.Count == 0) continue;

        varukScore = 0;

        foreach (var fac in factors){
            var list1 = student.Preferences.Take(fac.N).Select(row => row.Value);
            var list2 = studCompare.Value.Preferences.Take(fac.N).Select(row => row.Value);
            var listCmp = list1.Intersect(list2);

            var min = fac.N < 6 ? fac.N : student.Preferences.Count;
            varukScore += (Convert.ToDouble(listCmp.Count()) / min) * fac.Factor;
        }

        //If there are no similarities, go to next student.
        if (Math.Abs(varukScore) < 0.000001) continue;

        var cnt = studCompare.Value.Preferences.Count;
        var i = 0;
        foreach (var pref in studCompare.Value.Preferences) {
            if (newMentors.ContainsKey(pref.Value))
                newMentors[pref.Value] +=
                    varukScore * ((double)(cnt - i) / cnt);
            i++;
        }
    }

    // Sortira sve mentore prema ocjeni koju su dobili, zanemari one koje su dobili 0 bodova
    var list = newMentors
        .OrderByDescending(row => row.Value)
        .Where(row => row.Value > 0.00001)
        .ToList();

    // Dodaj kao popis u Extended listu mentora
    var index = 0;
    foreach (var item in list)
        _students[student.Jmbag].PreferencesExtended[index++] = item.Key;
}
```

Programski kôd 6 – Algoritam izrade proširene prioritetne liste

Prototip prvo inicijalizira popis dostupnih grupa koje student nije odabrao na svojoj prioritetnoj listi. Zatim se provodi usporedba sa svim ostalim studentima izračunom vrijednosti sličnosti prvih N elemenata, pri čemu je N = { 1, 3, 5 i Max } te se izračunava ukupna vrijednost svake od sličnosti. Za studente s kojima postoji sličnost izračunavaju se vrijednosti za svaku od grupa. Konačna proširena lista prioriteta dobiva se nakon što je usporedba napravljena sa svim ostalim studentima poretkom prema najvećem broju bodova svaku od grupa.

7.4.3. Izvedba izrade glavnog popisa studenata

Prototip, kao i pseudokod, implementira dva načina izrade glavnog popisa studenata. To su: (i) generiranje novog glavnog popisa studenata koji se koristi prilikom određivanja inicijalne populacije genetskog algoritma te (ii) algoritam generiranja glavnog popisa studenata temeljem podataka roditelja koji koristi metode mutacije i nasljeđivanja genetičkog algoritma. Prilikom nasljeđivanja, nova se populacija generira temeljem karakteristike dva najbolja rješenja prethodne populacije.

7.4.3.1. Izvedba generiranja novog glavnog popisa studenata

Novi glavni popis studenata generira se kod početnog generiranja populacije izradom novog potencijalnog rješenja. Inicijalno rješenje generira se promjenom prosječne ocjene studenta unutar vrijednosti zadanog parametra udaljenosti (distance). Promjenom prosječne ocjene algoritam ima implementirana ograničenja najveće i najmanje moguće ocjene (2,0 i 5,0) kako bi generirana rješenja bila ispravna i moguća. Sve ocjene prikazane su kao cijeli brojevi, s obzirom na to da se tako brže izvode operacije izračuna, te radi dodatne preciznosti.

Svakom studentu se ujedno dodjeljuje i jedinstvena nasumična vrijednost koja se koristi u procesu redanja studenata prema prosječnim ocjenama gdje se studenti koji imaju jednake ocjene potom redaju prema nasumično dodijeljenoj vrijednosti. Ovim se osigurava da su studenti uvijek jednako poredani, čak i kada imaju iste ocjene.

Izvedba algoritma početnog generiranja rješenja prikazana je na primjeru Programskog kôda 7.

```
private StudentIdGrade[] GetRankedListOfStudents(double distance,
                                                 StudentIdGrade[] studentGrades)
{
    var random = new Random();
    var len = _studentOrderBaseGrades.Length;
    var maxDistance = (int) (distance * 100);

    for (var i = 0; i < len; i++){
        studentGrades[i].StudentId = _studentOrderBaseGrades[i].StudentId;

        // Ocjena je: Osnovna - offset + (0 - 2*offset)
        // Min jer ocjena ne smije biti manja od 2.00 ni veća od 5.00
        var minRandGrade = Math.Max(_studentOrderBaseGrades[i].StudentGrade –
                                    maxDistance, 200);
        var maxRandGrade = Math.Min(_studentOrderBaseGrades[i].StudentGrade +
                                    maxDistance, 500);

        // +1 jer random ide do max maxRandGrade (ne uključujući MaxRandGrade)
        studentGrades[i].StudentGrade = random.Next(minRandGrade, maxRandGrade + 1);

        // Jedinstveni redni broj studenta, za raspodjelu studenata koji imaju iste
        // ocjene
        studentGrades[i].RandomId = (uint) random.Next(0, 100000);
    }

    return studentGrades;
}
```

Programski kôd 7 – Algoritam izrade glavnog popisa – početna populacija

7.4.3.2. Izvedba generiranja glavnog popisa studenata temeljem podataka roditelja

Za razliku od generiranja početne populacije genetskog algoritma, u svakom se od koraka nova rješenja generiraju temeljem podataka roditelja. U tom procesu koriste se metode genetskog algoritma: mutacija i nasljeđivanje.

Za svakog studenta algoritam prvo određuje hoće li se koristiti metoda mutacije ili nasljeđivanja. Metoda mutacije prati izvedbu inicijalnog generiranja glavnog popisa studenata dok metoda nasljeđivanje određuje roditelja od kojeg će se naslijediti vrijednost ocjene.

Izvedba algoritma generiranja glavnog popisa studenata temeljem podataka roditelja prikazana je na primjeru Programskog kôda 8.

```
private StudentIdGrade[] GetRankedListOfStudentsFromParents(double distance,
                        StudentIdGrade[] parent1, StudentIdGrade[] parent2,
                        StudentIdGrade[] result, int mutationProbability) {
    var len = parent1.Length;
    var random = new Random();
    var maxDistance = (int)(distance * 100);

    for (var i = 0; i < len; i++){
        var rand = random.Next(0, 100);
        var isMutation = rand < mutationProbability;
        result[i].StudentId = (uint)i;

        // Mutacija
        if (isMutation){
            var s = _studentOrderBaseGrades[i].StudentId;
            var minRandGrade = Math.Max(_studentOrderBaseGrades[i].StudentGrade –
                                    maxDistance, 200);
            var maxRandGrade = Math.Min(_studentOrderBaseGrades[i].StudentGrade +
                                    maxDistance, 500);
            result[s].StudentGrade = random.Next(minRandGrade, maxRandGrade + 1);
            result[s].RandomId = (uint)random.Next(0, 100000);
        }else{
            // Nasljeđivanje (50:50) je vjerojatnost nasljeđivanja 1. ili 2. roditelja
            var chooseParent = random.Next(0, 100);
            result[i].StudentGrade = chooseParent < 50 ? parent1[i].StudentGrade :
                                        parent2[i].StudentGrade;
            result[i].RandomId = (uint)random.Next(0, 100000);
        }
    }
    return result;
}
```

Programski kôd 8 – Algoritam izrade glavnog popisa – temeljem nasljeđivanja i mutacije

7.4.4. Izvedba algoritma novog rješenja

Programski kôd 9 primjer je izvedbe algoritma novog rješenja, a implementiran je metodom *GenerateNewSingleSolution* koja poziva dvije odvojene metode:

1. *GetSolution* – generira novo potencijalno rješenje;
2. *GradeSolution* – algoritam evaluacije algoritma.

```
private SingleSolutionSimple GenerateNewSingleSolution(SingleSolutionSimple
                                                        rankedListOfStudents){

  rankedListOfStudents.StudentOrder = GetSolution(rankedListOfStudents.StudentOrder);
  rankedListOfStudents.SolutionGrade = GradeSolution(rankedListOfStudents.StudentOrder);
  return rankedListOfStudents;
}
```

Programski kôd 9 – Izračun novog rješenja

Programski kôd 10 primjer je izvedbe algoritma izrade novog potencijalnog rješenja, a izvedba je pohlepnog algoritma (eng. *greedy algorithm*):

1. Studenti se poredaju prema svojim ocjenama (prvi pozicioniran na glavnom popisu studenata je student s najvišim prosjekom);

2. Algoritam prolazi studenta po studenta i pokušava ga prvo pridružiti u grupu na: prioritetnoj listi, potom na proširenoj prioritetnoj listi te potom nasumičnim odabirom – pri tome se bilježi lista s koje je odabir napravljen te redoslijed grupe na toj listi.

```
private StudentIdGrade[] GetSolution(StudentIdGrade[] rankedList)
{
    var mentorFreeSpaces = new int[_mentors.Count];
    var numberOfFreeSpaces = (int) Math.Ceiling
                                    ( (double) _students.Count / _mentors.Count);
    var numberOfMentorsMax = (_students.Count % _mentors.Count);
    var localListOfAllMentors = new uint[_listOfAllMentors.Length];
    var lenMentors = _listOfAllMentors.Length;
    for (var i = 0; i < lenMentors; i++)
    {
        localListOfAllMentors[i] = _listOfAllMentors[i];
    }

    // Poredak polja prema ocjenama studenata
    Array.Sort(rankedList, CompareStudentGrades);

    var len = rankedList.Length;
    for (var i = 0; i < len; i++){
        // Prvi odabir studenta
        var mentorAssignmentType = 1;
        var mentorFound = FindFirstAvailableMentor(mentorFreeSpaces,
            _studentMentorPreferences[rankedList[i].StudentId],
            numberOfFreeSpaces, out var mentorAssignedIndex);

        // Proširena lista
        if (mentorFound == -1){
            mentorFound = FindFirstAvailableMentor(mentorFreeSpaces,
                _studentMentorPreferencesExtended[rankedList[i].StudentId],
                numberOfFreeSpaces, out mentorAssignedIndex);
            mentorAssignmentType = 2;
        }

        // Random odabir
        if (mentorFound == -1){
            Randomize(localListOfAllMentors);
            mentorFound = FindFirstAvailableMentor(mentorFreeSpaces,
                localListOfAllMentors, numberOfFreeSpaces, out mentorAssignedIndex);
            mentorAssignmentType = 3;
        }

        if (mentorFound == -1)
            throw new Exception("Unable to find mentor.");

        // Mentora označi kao da ima studenta
        mentorFreeSpaces[mentorFound]++;

        if (mentorFreeSpaces[mentorFound] == numberOfFreeSpaces){
            numberOfMentorsMax--;
            if (numberOfMentorsMax == 0) numberOfFreeSpaces--;
        }

        rankedList[i].MentorId = (uint) mentorFound;
        rankedList[i].AssignmentType = (uint) mentorAssignmentType;
        rankedList[i].AssignmentOrder = (uint) mentorAssignedIndex;
    }

    return rankedList;
}
```

Programski kôd 10 – Algoritam strogog jednostrukog pridruživanja

Izvedba funkcije ocjene, prikazana je na primjeru Programskog kôda 11, a implementirana je na način koji definira pseudokod, a za svakog od studenata se radi ocjena rješenja temeljem liste iz koje je studentu dodijeljena grupa te pozicije grupe na toj listi.

```
private static uint GradeSolution(StudentIdGrade[] solution){
    uint solutionScore = 0;
    foreach (var student in solution) {
        uint val = 0;
        switch (student.AssignmentType){
            case 1: // Studentov odabir
                val = 40 + 2 * (student.AssignmentOrder < 10 ? (10 -
                                    student.AssignmentOrder) : 1);
                break;
            case 2: // Prošireni odabir
                val = 10 + 2 * (student.AssignmentOrder < 10 ? (10 –
                                    student.AssignmentOrder) : 1);
                break;
            case 3: // Random odabir
                val = 0;
                break;
        }

        solutionScore += val;
    }

    return solutionScore;
}
```

Programski kôd 11 – Algoritam ocjene strogog jednostrukog pridruživanja

7.5. Izvršavanje prototipa programskog sustava

7.5.1. Primjer ulaznih datoteka

Prototipna izvedba algoritma za ulaz koristi tri datoteke:

- *Students.txt* – Popis studenata;
- *Mentors.txt* – Popis raspoloživih grupa;
- *Preference.txt* – Studentske prioritetne liste.

Izvedba predviđa mogućnost da jedna datoteka sadrži podatke za više godina, pri čemu se algoritmu zadaje akademska godina za koju će se podaci obrađivati. Sve datoteke su tekstualne, podaci su unutar jednog retka odvojeni oznakom tabulatora, a novi red predstavljen je oznakom \n. Sve ulazne datoteke imaju prvi redak zaglavlja kojim se određuju podaci koji se nalaze u datoteci. Poredak podataka u datotekama nije važan, a prototipna izvedba sama će raditi poredak prema zahtjevima pojedinog algoritma.

Datoteka studenata sadrži podatke:

- *Year* – akademska godina na koju se podaci odnose;
- *Jmbag* – JMBAG studenta;
- *Ocjena* – ocjena studenta u decimalnom obliku uporabom decimalne točke.

Primjer ulazne datoteke sa šest studenata (SA, SB, SC, SD, SE i SF) prikazan je na Slici 23. Za sve studente se umjesto stvarnih vrijednosti JMBAG-a mogu koristiti i zamjenske oznake sukladno pravilima koje propisuje Opća uredba o zaštiti podataka (GDPR), a što prototipna izvedba algoritma podržava.

```
year    jmbag    ocjena
2017    SA       4.2
2017    SB       4
2017    SC       3.9
2017    SD       3.2
2017    SE       3
2017    SF       2.9
```

Slika 23 – Primjer ulazne datoteke studenata

Datoteka grupa sadrži podatke:

- *Year* – akademska godina na koju se podaci odnose;
- *Code* – jedinstvena oznaka grupe;
- *Ime_djel* – naziv grupe (algoritam ne koristi ovu vrijednost za izračun);
- *Prez_djel* – oznaka grupe (algoritam ne koristi ovu vrijednost za izračun);
- *Krat_orgjed* – pripadnost grupe (algoritam ne koristi ovu vrijednost za izračun).

Ulazna datoteka grupa sadržava podatak o svim dostupnim grupama na koje je moguće pridružiti studenta. Prototipni algoritam implementiran je tako da odabiri grupa u prioritetnim listama studenata, a koje nisu navedene u datoteci grupa, budu automatski zanemareni. Također, u datoteci grupa mogu se nalaziti i one grupe koje nisu u datoteci odabira. To su grupe koje niti jedan student nije odabrao, a na koje će se također, ovisno o algoritmu koji se koristi, pridruživati studenti.

Uz podatak o jedinstvenoj oznaci grupe, nudi se mogućnost dodatna tri polja koja korisnik može koristiti za vlastite potrebe, a koja ostaju sadržana u rezultatima izvođenja algoritma. U ovom radu algoritam ne koristi ova polja, a u primjeru ulazne datoteke može se vidjeti da su sva polja zamijenjena drugim vrijednostima čime je izvedba usklađena s pravilima GDPR-a.

U primjeru ulazne datoteke prikazane na Slici 24 navedene su samo četiri grupe M1, M2, M3 i M4 u koje studenti trebaju biti pridruženi.

```
year    code  ime_djel  prez_djel   krat_orgjed
2017    M1    I1        P1          O1
2017    M2    I1        P1          O1
2017    M3    I1        P1          O1
2017    M4    I1        P1          O1
```

Slika 24 – Primjer ulazne datoteke grupa

Datoteka prioritetnih lista sadrži podatke:

- *Year* – akademska godina na koju se podaci odnose;
- *JMBAG* – jedinstvena oznaka studenta;
- *Code* – jedinstvena oznaka grupe;
- *Ord* – redni broj odabira u prioritetnoj listi.

Datoteka prioritetnih lista sadrži sve odabire studenata koji su ih napravili. Za studente koji nisu navedeni u ovoj datoteci, smatrat će se da nisu napravili svoj odabir te će biti pridruženi nasumičnim odabirom. Odabiri grupa koje nisu dostupne u datoteci grupa će se zanemariti. Odabir studenata koji nisu navedeni u datoteci studenata će se također zanemariti.

U primjeru ulazne datoteke na Slici 25 prikazani su odabiri svih studenata. Student SA odabrao je grupu M1 i M2, dok je student SF odabrao samo grupu M3.

```
year    jmbag   mentor   ord
2017    SA      M1       1
2017    SA      M2       2
2017    SB      M1       1
2017    SC      M1       1
2017    SD      M3       1
2017    SD      M4       2
2017    SE      M3       1
2017    SF      M3       1
```

Slika 25 – Primjer ulazne datoteke prioritetnih lista studenata

7.5.2. Proširena prioritetna lista

Prototipna izvedba algoritma izračunava proširenu prioritetnu listu za sve studente. Za prikazane ulazne datoteke vrijednosti grupa proširene prioritetne liste su navedene u Tablici 6.

Tablica 6 – Izračun proširenih prioritetnih lista za primjer ulaznih datoteka

Student	Grupe proširene prioritetne liste
SA	(nema)
SB	M2
SC	M2
SD	(nema)
SE	M4
SF	M4

7.5.3. Rezultat izvođenja – pridruživanje studenata grupama

Za prikazani primjer ulaznih podataka studenata, grupa i odabira prototipna izvedba je odredila pridruživanje prikazano u Tablici 7. Izvedba prototipnog algoritma ispravno je odredila da se u grupe M1 i M3 pridruže dva studenta, dok će se u grupe M2 i M4 pridružiti po jedan student. Također je vidljivo i pridruživanje putem prioritetne liste studenta SC i SF koji su odabrali grupe u kojima nema dovoljno mjesta, a pridruženi su grupama koje su sličnije studentima koji su napravili odabir sličniji njima. Nijedan student nije pridružen nasumično.

Tablica 7 – Rezultat izvođenja prototipa – pridruživanje studenata grupama

Student	Prosječna ocjena	Pridružena grupa	Prioritetna lista	Proširena prioritetna lista	Redoslijed odabira
SA	4,20	M1	Da		1
SB	4,00	M1	Da		1
SC	3,90	M2		Da	1
SD	3,20	M3	Da		1
SE	3,00	M3	Da		1
SF	2,90	M4		Da	1

7.6. Prikaz zaslona prilikom izvršavanja prototipa

Na Slici 26 prikazano je izvođenje poslužiteljskog dijela prototipa. On prvo učitava podatke i sprema ih u zajedničku bazu podataka. Zatim se u zadanim vremenskim intervalima na zaslon ispisuju statusne informacije o najboljem do sada pronađenom rješenju. Sve statusne

informacije dohvaćaju se iz zajedničke baze podataka, a koje su zapisale klijentske instance koje se paralelno izvode.

```
Select Administrator: Developer Command Prompt for VS 2017 - dotnet  solver.dll -server localhost 54320
Loading mentors...
Loading students...


  #0 Best: -1 #Clients: 3 Time: 16.10.2019. 20:22:56
 Found: -1


  #1 Best: -1 #Clients: 4 Time: 16.10.2019. 20:22:57
 Found: -1
.

  #3 Best: 20618 #Clients: 4 Time: 16.10.2019. 20:22:59
 Found: 20618


  #4 Best: 21040 #Clients: 4 Time: 16.10.2019. 20:23:00
 Found: 21040


  #5 Best: 21246 #Clients: 4 Time: 16.10.2019. 20:23:01
 Found: 21246


  #6 Best: 21660 #Clients: 4 Time: 16.10.2019. 20:23:02
 Found: 21660


  #7 Best: 21958 #Clients: 4 Time: 16.10.2019. 20:23:03
 Found: 21958
```

Slika 26 – Izvršavanje algoritma na poslužiteljskom računalu

Na Slici 27 prikazano je izvođenje algoritma na klijentskom računalu. Algoritam odgađa pokretanje do evidencije zadatka u bazi. Nakon što su podaci spremni, dohvaća ih i računa rješenje. Ako je našao bolje rješenje od onoga koje je do sada zapisano u bazi, rješenje se sprema i označava se kraj izračuna. Tijekom izvođenja svako klijentsko računalo ispisuje statusne informacije o nađenom najboljem rješenju unutar svoje instance.

```
Select Administrator: Developer Command Prompt for VS 2017 - dotnet ..\solver.dll -client localhost 54320
Starting the client application...
  Waiting for data .........

 Starting the evaluation
 Starting the calculation:
    Distance:   0,5
    Paralel:    4
    Iterations: 100


Invalid projects (28)
  #2 Best: 20618 Iteration: 1 Time: 16.10.2019. 20:22:59
      Sel: 0: 153  1: 65  2: 40  3: 23  4: 15  5: 13  6: 4  7: 6  8: 1  9: 2  10: 4
      Ext: 0: 6  1: 2  2: 2  3: 6  4: 1  5: 5  6: 0  7: 4  8: 3  9: 2  10: 109
      Rnd: 0: 6  1: 5  2: 3  3: 3  4: 2  5: 2  6: 0  7: 1  8: 3  9: 0  10: 28
  #1 Best: 20690 Iteration: 1 Time: 16.10.2019. 20:22:59
      Sel: 0: 164  1: 62  2: 32  3: 26  4: 18  5: 7  6: 6  7: 7  8: 3  9: 0  10: 3
      Ext: 0: 3  1: 1  2: 1  3: 4  4: 6  5: 0  6: 0  7: 2  8: 0  9: 0  10: 125
      Rnd: 0: 9  1: 2  2: 1  3: 1  4: 3  5: 2  6: 0  7: 5  8: 3  9: 0  10: 23
  #3 Best: 20946 Iteration: 1 Time: 16.10.2019. 20:22:59
      Sel: 0: 167  1: 64  2: 32  3: 26  4: 16  5: 9  6: 6  7: 5  8: 5  9: 0  10: 3
      Ext: 0: 2  1: 3  2: 5  3: 4  4: 2  5: 0  6: 1  7: 0  8: 0  9: 3  10: 117
      Rnd: 0: 6  1: 0  2: 7  3: 2  4: 2  5: 3  6: 3  7: 3  8: 3  9: 0  10: 20
  #2 Best: 21040 Iteration: 2 Time: 16.10.2019. 20:23:00
      Sel: 0: 168  1: 69  2: 32  3: 24  4: 15  5: 11  6: 4  7: 5  8: 1  9: 4  10: 2
      Ext: 0: 2  1: 1  2: 2  3: 7  4: 5  5: 3  6: 1  7: 0  8: 3  9: 1  10: 106
      Rnd: 0: 8  1: 3  2: 5  3: 2  4: 4  5: 2  6: 3  7: 0  8: 2  9: 4  10: 20
```

Slika 27 – Izvršavanje algoritma na klijentskom računalu

Slika 28 prikazuje izvođenje prototipa u samostalnom načinu rada u kojem se koristi raspodijeljena programska potpora koristeći četiri jezgre. Višedretvenost je izvedena kroz dijeljenje podataka u algoritmima. Na zaslonu se ispisuje dretva koja je došla do boljeg rješenja. Svaka od jezgri se označava kao #ID gdje je ID broj od 0 do N.

```
Administrator: Developer Command Prompt for VS 2017
Starting the calculation:
   Distance:   0,5
   Paralel:    4
   Iterations: 100


Invalid projects (28)
  #0 Best: 20636 Iteration: 1 Time: 16.10.2019. 20:20:23
      Sel: 0: 180  1: 60  2: 33  3: 22  4: 7  5: 7  6: 5  7: 6  8: 0  9: 2  10: 2
      Ext: 0: 3  1: 0  2: 1  3: 5  4: 2  5: 1  6: 1  7: 4  8: 2  9: 1  10: 125
      Rnd: 0: 7  1: 4  2: 2  3: 0  4: 2  5: 3  6: 2  7: 3  8: 0  9: 3  10: 24
  #1 Best: 21016 Iteration: 1 Time: 16.10.2019. 20:20:24
      Sel: 0: 181  1: 61  2: 29  3: 27  4: 13  5: 5  6: 4  7: 2  8: 5  9: 3  10: 5
      Ext: 0: 2  1: 0  2: 1  3: 4  4: 3  5: 2  6: 0  7: 4  8: 1  9: 0  10: 117
      Rnd: 0: 9  1: 2  2: 7  3: 3  4: 0  5: 2  6: 2  7: 0  8: 1  9: 3  10: 21
  #0 Best: 21060 Iteration: 2 Time: 16.10.2019. 20:20:24
      Sel: 0: 179  1: 56  2: 40  3: 20  4: 17  5: 7  6: 1  7: 8  8: 2  9: 3  10: 3
      Ext: 0: 3  1: 1  2: 1  3: 3  4: 1  5: 1  6: 2  7: 4  8: 0  9: 1  10: 116
      Rnd: 0: 5  1: 1  2: 3  3: 2  4: 4  5: 2  6: 2  7: 1  8: 1  9: 6  10: 23
  #2 Best: 21290 Iteration: 2 Time: 16.10.2019. 20:20:25
      Sel: 0: 173  1: 68  2: 37  3: 24  4: 16  5: 9  6: 3  7: 6  8: 1  9: 2  10: 0
      Ext: 0: 2  1: 2  2: 3  3: 7  4: 1  5: 1  6: 0  7: 2  8: 0  9: 2  10: 109
      Rnd: 0: 6  1: 10  2: 2  3: 4  4: 1  5: 2  6: 3  7: 1  8: 2  9: 1  10: 19
  #0 Best: 21450 Iteration: 3 Time: 16.10.2019. 20:20:25
      Sel: 0: 176  1: 72  2: 35  3: 23  4: 12  5: 11  6: 3  7: 7  8: 2  9: 2  10: 3
      Ext: 0: 1  1: 2  2: 1  3: 2  4: 1  5: 0  6: 2  7: 3  8: 0  9: 2  10: 107
      Rnd: 0: 9  1: 8  2: 1  3: 1  4: 3  5: 3  6: 3  7: 2  8: 1  9: 1  10: 20
  #3 Best: 21470 Iteration: 3 Time: 16.10.2019. 20:20:26
      Sel: 0: 172  1: 64  2: 37  3: 26  4: 19  5: 8  6: 6  7: 3  8: 4  9: 1  10: 5
      Ext: 0: 2  1: 1  2: 3  3: 2  4: 6  5: 2  6: 2  7: 2  8: 3  9: 1  10: 101
      Rnd: 0: 11  1: 3  2: 4  3: 1  4: 3  5: 1  6: 2  7: 0  8: 1  9: 1  10: 22
  #0 Best: 21596 Iteration: 4 Time: 16.10.2019. 20:20:26
      Sel: 0: 181  1: 66  2: 36  3: 22  4: 17  5: 9  6: 4  7: 6  8: 1  9: 3  10: 3
      Ext: 0: 1  1: 2  2: 2  3: 2  4: 3  5: 1  6: 1  7: 2  8: 0  9: 3  10: 104
      Rnd: 0: 11  1: 6  2: 0  3: 2  4: 1  5: 1  6: 1  7: 3  8: 1  9: 4  10: 20
  #2 Best: 21598 Iteration: 5 Time: 16.10.2019. 20:20:27
      Sel: 0: 167  1: 72  2: 39  3: 26  4: 21  5: 9  6: 5  7: 5  8: 0  9: 3  10: 0
      Ext: 0: 3  1: 2  2: 4  3: 5  4: 1  5: 3  6: 0  7: 3  8: 0  9: 2  10: 96
      Rnd: 0: 9  1: 2  2: 5  3: 0  4: 1  5: 2  6: 3  7: 1  8: 2  9: 1  10: 27
  #0 Best: 21618 Iteration: 5 Time: 16.10.2019. 20:20:28
```

Slika 28 – Višedretvenost – prikaz izvršavanje algoritma

Na slici **29** prikazana je implementacija prototipa koje koristi sučelje u obliku web stranice. Na njemu je moguće pratiti rad svake dretve istovremeno. Uz svaku dretvu prikazano je i koliko iteracija je do sada prošla. Također je označeno do sada najbolje pronađeno rješenje.

Aktivnosti

localhost:8081/

Najbolji rezultat: 22678 Udaljenost: 0.5

Iteracija:

ThreadID	Iteracija	Bodovi	Iter. / sek
#0	67	22360,22334,21864,22142,21998,22018,21724,22090,22004,21776,21894,22016,22040,22042,21984,0,0,0,0,0	10
#1	62	22624,22620,22520,22244,21720,22224,22454,22114,21914,22314,21972,22272,21968,22186,22340,0,0,0,0,0	8
#2	64	22336,22304,21814,21892,21906,21596,21610,21722,21642,21896,22080,22258,21882,21914,21876,0,0,0,0,0	10
#3	52	22678,22536,22106,22184,22210,22362,22286,22324,22128,22406,22228,22330,22044,22288,22096,0,0,0,0,0	10

Statistika:

ThreadID	Score	Iteration	Time	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.+
				173	78	30	30	13	8	8	6	4	3	2
#0	21904	9	0d 00:00:06	3	0	3	4	3	1	2	2	4	1	91
				8	1	4	1	1	3	2	1	2	2	25
				176	71	39	32	19	17	7	4	1	4	3
#1	22624	58	0d 00:00:15	4	4	1	2	3	0	1	0	0	3	76
				8	3	4	4	1	3	1	1	5	1	21
				176	73	35	27	20	6	3	6	2	3	1
#2	21810	7	0d 00:00:06	3	2	1	1	5	1	1	3	1	2	96
				6	3	2	2	3	4	3	2	2	0	24
				176	75	36	33	16	15	8	5	3	3	5
#3	22678	52	0d 00:00:15	3	2	3	2	3	2	0	1	0	1	75
				6	3	1	3	2	3	3	4	1	1	25

Slika 29 – Višedretvenost – sučelje nadzora izvođenja algoritma

8. PRIMJENA METODA I MODELA U NASTAVNIM PROCESIMA

U ovom poglavlju je navedena izvedba predloženih algoritama na primjerima problema pridruživanja u nastavnim procesima. Problemi pridruživanja opisani su u 2. poglavlju, a predloženi algoritmi u 5. poglavlju.

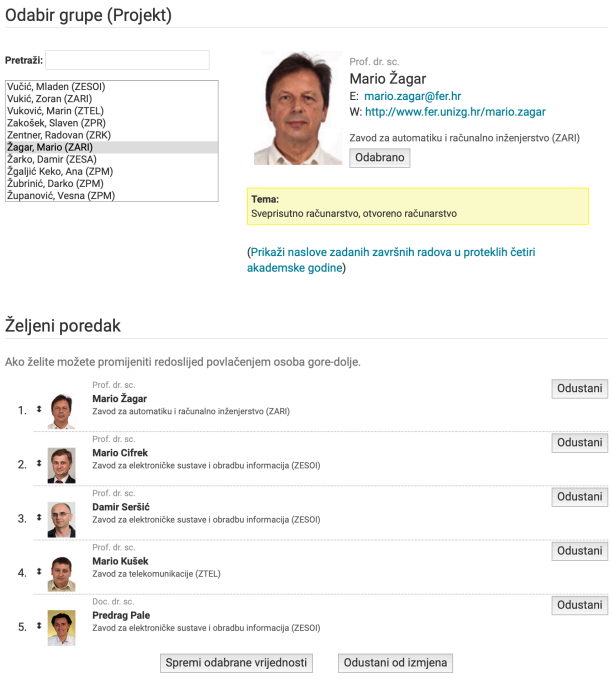
8.1. Pridruživanje studenta projektu

Algoritam strogog jednostrukog pridruživanja s glavnim popisom u potpunosti se koristi prilikom pridruživanja studenata projektu na predmetu „Seminar“ Fakulteta elektrotehnike i računarstva.

Nastavnici svake godine određuju područja svojih projekata. Svaki je nastavnik voditelj jednog projekta. Studenti prilikom upisa predmeta rangiraju izabrane projekte kojima žele biti pridruženi. Svi podaci o prioritetnim listama prikupljaju se i spremaju u bazu podataka.

Na početku akademske godine, po završetku upisa, nakon što su poznati svi studenti upisani na predmet, izvodi se algoritam za pridruživanje studenata na projekte. Najbolje dobiveno rješenje koristi se kao konačni prijedlog pridruživanja studenata u projekte.

Na Slici 30 prikazana je izvedba odabira prioritetne liste projekata. S obzirom na to da svaki nastavnik vodi jednu grupu, studenti na prioritetnu listu stavljaju nastavnike. Uz svakog nastavnika prikazane su i kratke informacije o području projekta te temama nastavnika. Na slici, student je odabrao pet projekata.



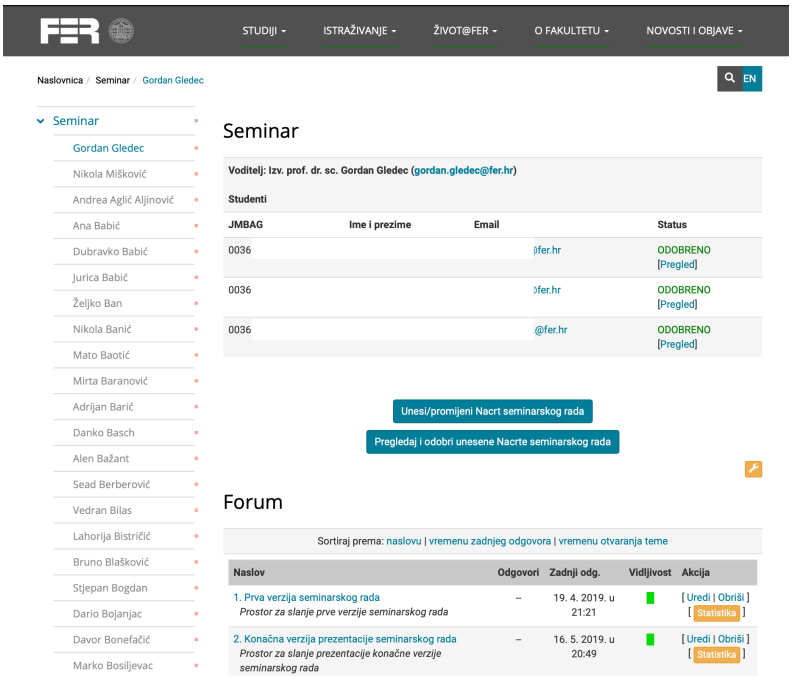
Slika 30 – Primjer sučelja odabira prioritetne liste projekata

Na Slici 31 prikazan je primjer tablice u kojoj se spremaju podaci o prioritetnim listama studenata. Uz svakog studenata zabilježena je grupa i njen redoslijed na prioritetnoj listi.

	jmbag	ord	mentor
1	0006032679	1	LB011
2	0006032679	2	SG006
3	0006032679	3	AS038
4	0006032679	4	ŽC000
5	0006032679	5	ID001
6	0006032679	6	DJ010
7	0006032679	7	ZK020
8	0006032679	8	IP020
9	0006032679	9	MM036
10	0006032679	10	SB025
11	0006032679	11	TB046
12	0006032679	12	TP007
13	0006032679	13	MŠ026
14	0006032679	14	IL008
15	0006032679	15	GJ004
16	0006032679	16	DS013
17	0006032679	17	JP085
18	0006032679	18	AN006
19	0006032679	19	DP024
20	0006032679	20	DŠ012
21	0023120117	1	SŠ014
22	0023120117	2	MK056

Slika 31 – Primjer spremljenih prioritetnih lista

Na Slici 32 prikazan je rezultat izvođenja algoritma na primjeru pridruživanja studenta projektu. Svaka projektna grupa ima vlastitu web-stranicu grupe gdje nastavnik i studenti mogu razmjenjivati materijale.



Slika 32 – Primjer web-stranice jedne projektne grupe

8.2. *Pridruživanje studenta mentoru*

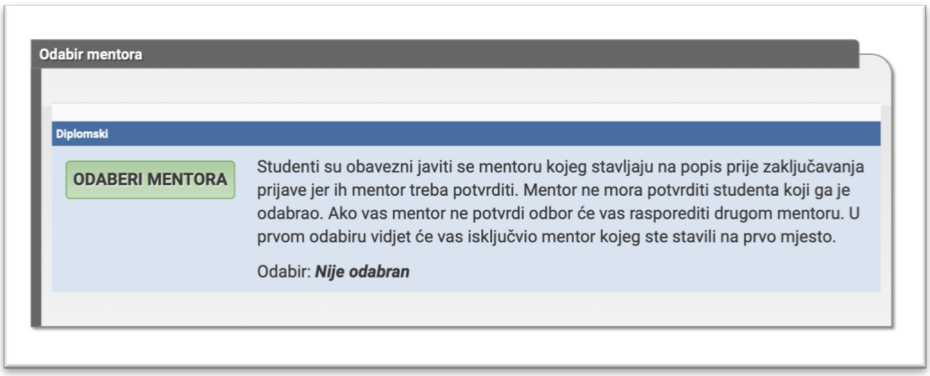
Pridruživanje studenta mentoru vrlo je sličan problem pridruživanja studenta projektu, te se može primijeniti jednaki algoritam njegovog rješavanja.

Na Fakultetu elektrotehnike i računarstva, studenti mentore odabiru putem prioritetne liste. Njihov odabir potvrđuju mentori, a sva pridruživanja odobravaju odbori.

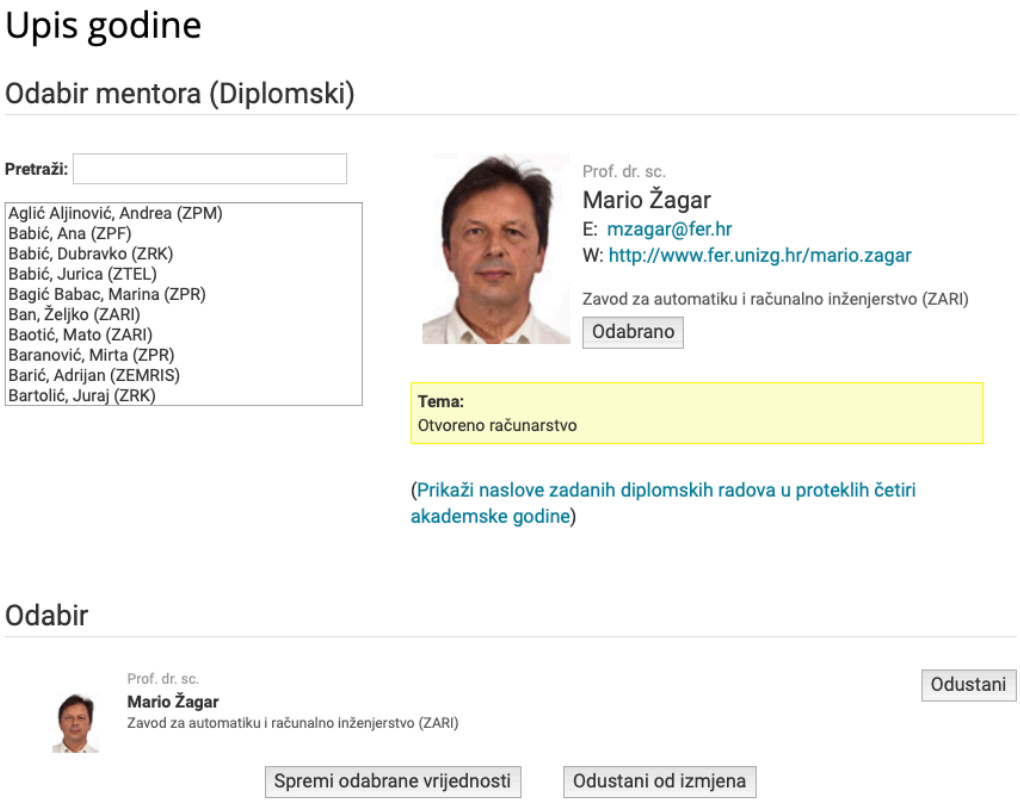
Nakon što mentori potvrde studente, nad svim studentima kojima mentori nisu odbili niti potvrdili niti jednog studenta koristi se algoritam strogog jednostrukog pridruživanja. Studenti se pridružuju samo prvom odabiru s njihove prioritetne liste, a mentorima se pridružuje najviše šest studenata.

Odbor pridružuje sve nepridružene studente. Odbor može pridružiti studenta temeljem njegovog drugog ili sljedećeg odabira s prioritetne liste ili popisa nastavnika njegovog smjera. Predložena metoda za pridruživanje nepridruženih elemenata koja se koristi kao podloga algoritma izrade proširene liste, može se koristiti i kao preporučiteljski sustav u kojem su nastavnici umjesto abecedno, poredani temeljem proširene prioritetne liste. Tako bi Odbor mogao koristiti ne samo svoje znanje o mentorima koje pridružuje nepridruženim studentima već i ekspertno znanje proizašlo iz nove predložene metode pridruživanja nepridruženih elemenata u nastavnim procesima temeljem zaključaka izvedenih iz konteksta odabira.

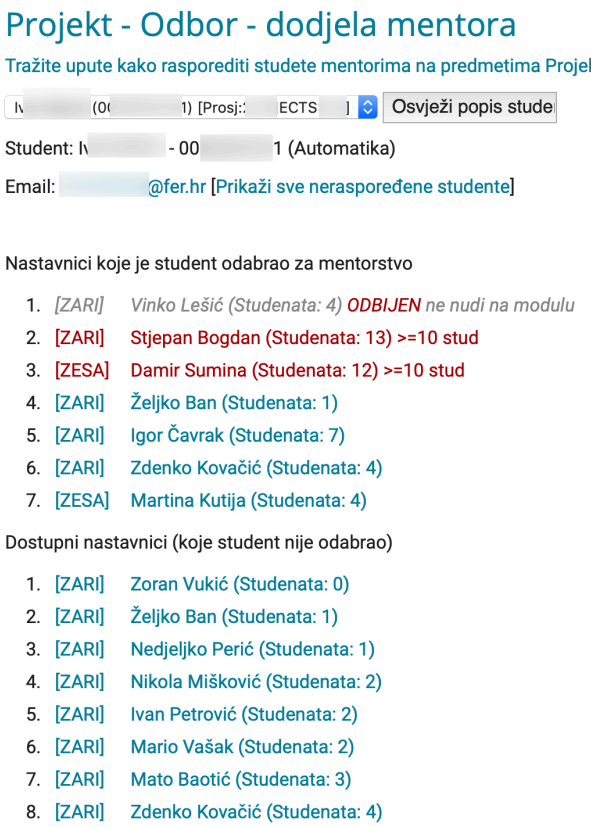
S obzirom na to da su promjene mentora tijekom godina zanemarive, algoritam izrade proširene liste mogao bi se koristiti kao preporučiteljski sustav prilikom odabira mentora. Kada student odabere minimalno pet mentora, mogla bi se predložiti proširena prioritetna lista temeljem odabira koje su napravili studenti prethodnih godina. U slučaju korištenja preporučiteljskog sustava potrebno je obratiti posebnu pažnju na nove nastavnike jer oni još ne bi bili u preporučiteljskom sustavu pa je moguće da bi ih studenti manje odabirali.



Slika 33 – Izvedba glavnog zaslona odabira mentora na diplomskom studiju



Slika 34 – Izvedba sučelja kreiranja prioritetne liste mentora na diplomskom studiju

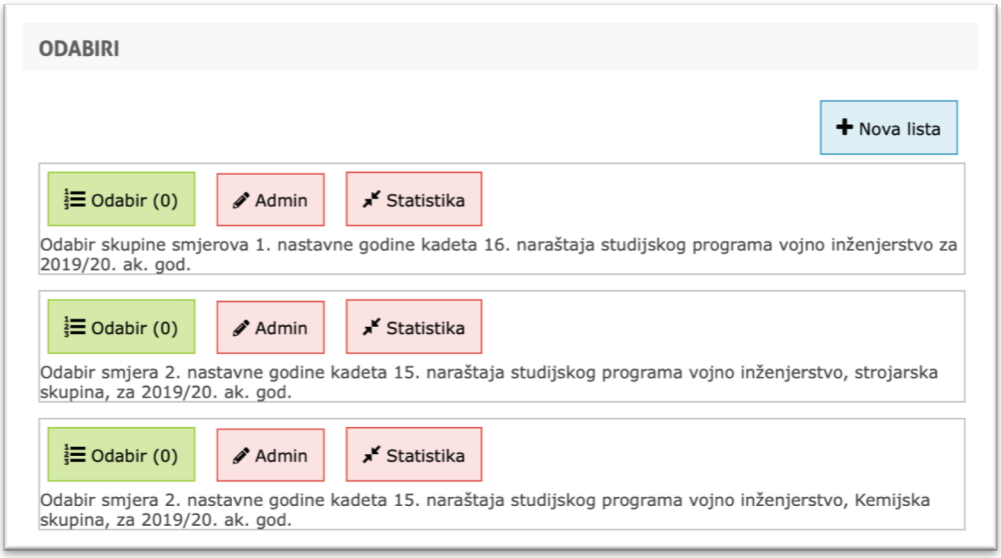


Slika 35 – Izvedba sučelja odobravanja mentorstva u radu Odbora

8.3. Odabir smjera

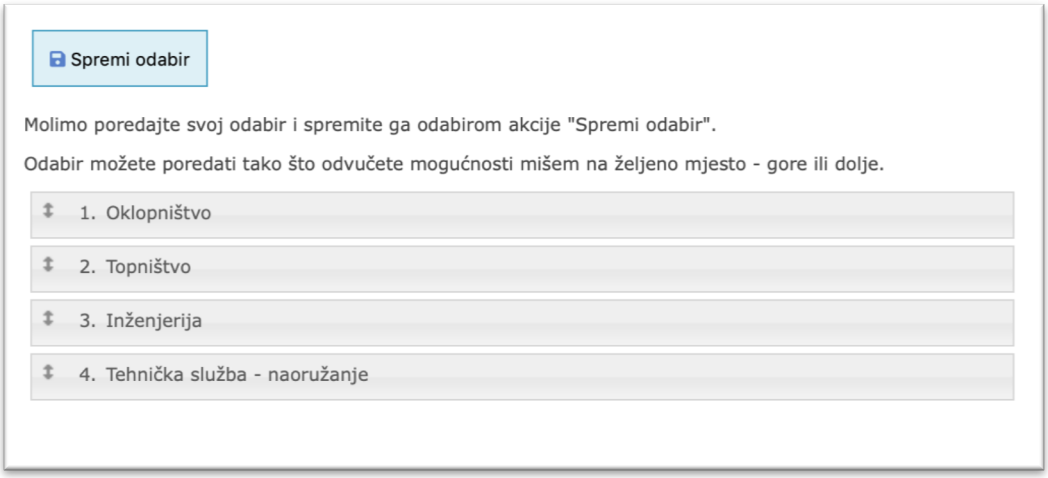
Pridruživanje studenata smjeru na vojnim studijskim programima „Vojno inženjerstvo“ i „Vojno vođenje i upravljanje“ u potpunosti je implementirano algoritmom strogog jednostrukog pridruživanja s glavnim popisom.

Studenti, ovisno o smjeru i nastavnoj godini, vide jednu od lista odabira prikazanih na Slici **36**. Na slici je prikazan administratorski pogled koji vidi sva pridruživanja koja su u tijeku.



Slika 36 – Primjer glavnog zaslona odabira – administratorski pogled

Studenti rangiraju smjerove na svojoj prioritetnoj listi. Slika **37** prikazuje izvedbu sučelja izrade prioritetne liste na primjeru jednog studenta.



Slika 37 – Primjer kreiranja prioritetne liste usmjerenja – studentov pogled

Nakon što završe odabiri, u sustav se učita glavni popis studenata, čime se pokreće i algoritam izračuna. S obzirom na mali broj podataka Algoritam se ne izvodi raspodijeljeno, a za vrijednost udaljenosti koristi se nula s obzirom na to da se prioritet zadan bodovima mora poštovati.

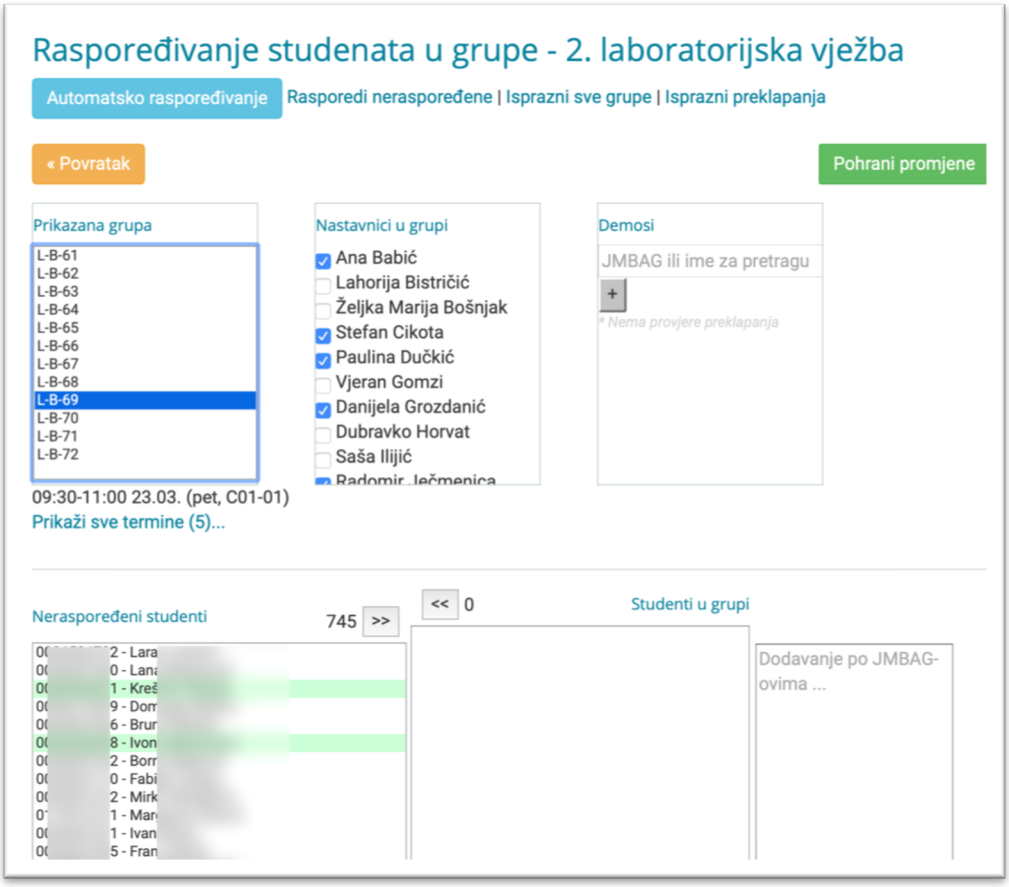
8.4. Pridruživanje studenata u grupe s određenim terminima održavanja

Pridruživanje studenata u grupe s određenim terminima održavanja u potpunosti koristi algoritam višestrukog pridruživanja s glavnim popisom. Za razliku od pridruživanja studenata projektu ili mentoru te odabira smjera, kod pridruživanja studenata u grupu student se smije pridružiti isključivo u grupu koja je na njegovoj prioritetnoj listi.

U rješavanju ovog problema sve prioritetne liste generiraju se automatski. Na glavnom popisu rangiranih studenata studenti se rangiraju prema broju grupa u koje student može biti raspodijeljen. Student koji može biti pridružen u najmanji broj grupa rangiran je na početku liste. Svakom studentu generira se prioritetna lista grupa na koju se stavljaju samo one grupe u kojima student može sudjelovati, odnosno čiji termini nisu u preklapanju s njegovim dotadašnjim rasporedom. Grupa u koju može se pridružiti najmanji broj studenata rangirana je na početku studentovog popisa prioriteta.

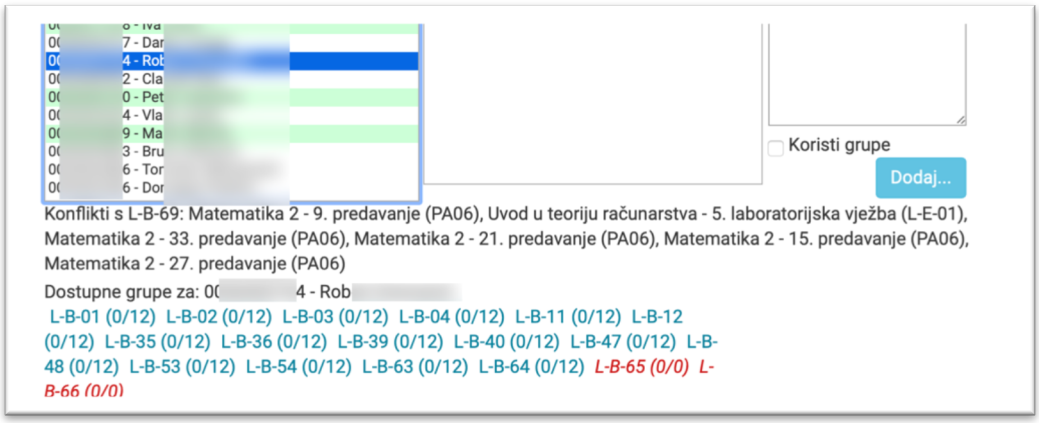
Studenti se na glavnom popisu rangiraju prema broju grupa kojima mogu biti pridruženi. Zbog toga parametar udaljenosti može biti od 0 do ukupnog broja grupa.

Na Slici **38** prikazana je izvedba sučelja pridruživanja studenata u grupe. U danom primjeru 745 studenata pridruženo je u 72 grupe. Na prikazu je odabrana grupa L-B-68, u koju se mogu pridružiti dva studenta koji su zeleno označeni u donjem lijevom izborniku.

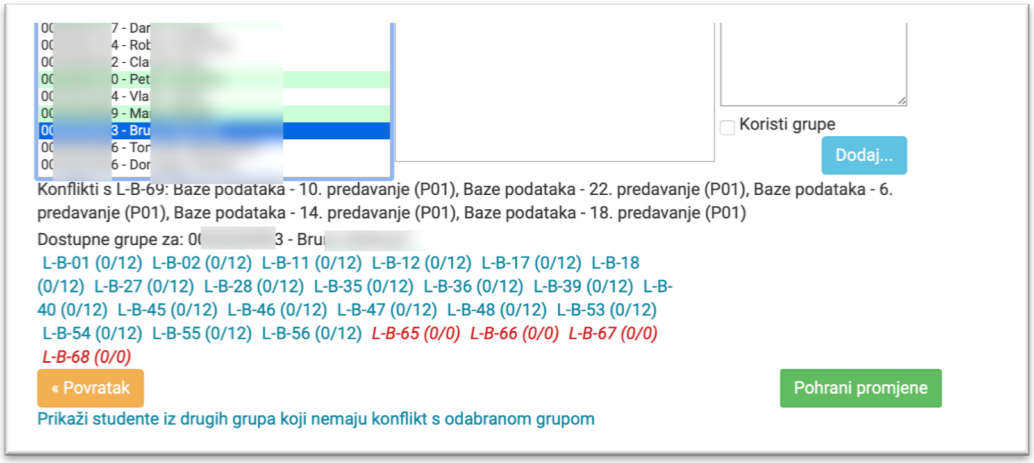


Slika 38 – Zaslon pridruživanja studenata u grupe

Na Slici **39** vidljiv je prikaz dostupnih grupa jednog studenta (L-B-01, L-B-02, L-B-11, itd). Studentu je od 72 grupe dostupno njih 16. Plavo su označene grupe u kojima ima mjesta i student nema preklapanja, a crveno su grupe u kojima nema mjesta. Usporedno, na Slici **40** vidljivo, drugi odabrani student može se pridružiti u 20 grupa (L-B-01, L-B-02, L-B-12).

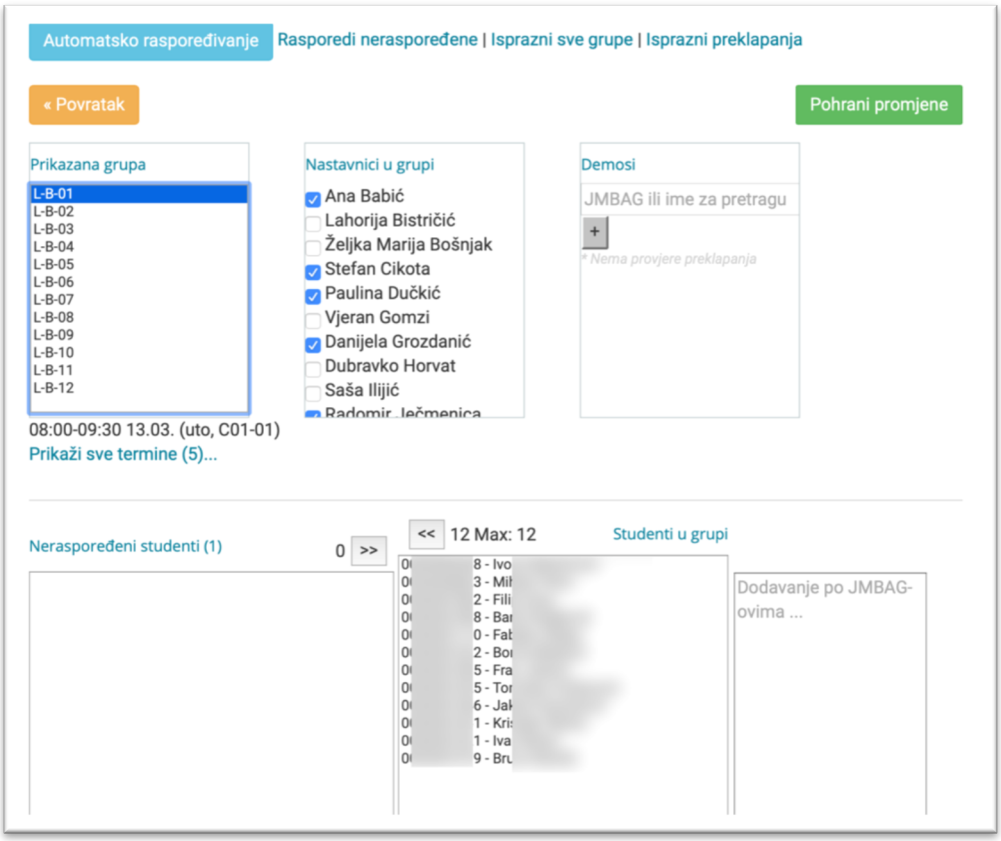


Slika 39 – Prikaz slobodnih grupa u koje student može biti pridružen – Student 1



Slika 40 – Prikaz slobodnih grupa u koje student može biti pridružen – Student 2

Na Slici 41 prikazan je rezultat izvršavanja algoritma. Nije ostao niti jedan nepridružen student – u donjem lijevom dijelu nema niti jednog studenta.



Slika 41 – Prikaz zaslona pridruživanja nakon pridruživanja

8.5. *Pridruživanje studenata na studentske prakse*

Pridruživanje studenata na studentske prakse može se provoditi putem razvijenog prototipa pridruživanja, a temeljem stvarnih odabira studenata.

Studenti na prioritetnu listu stavljaju pozicije na koje bi htjeli biti pridruženi. Nakon što završe odabiri, tvrtke potvrđuju studente. Studenti se pridružuju samo na one pozicije koje su na njihovoj prioritetnoj listi, a potvrdila ih je tvrtka.

Pridruživanje se može provoditi putem algoritma višestrukog pridruživanja s glavnim popisom. Način mogućeg rada ovog algoritma prikazan je u nastavku. Studenti se rangiraju prema prosječnoj ocjeni studija te se pridružuju na pozicije. Student se može pridružiti najviše na jednu poziciju, što se koristi kao gornje ograničenje u višestrukom pridruživanju.

Na Slici 42 prikazana je izvedba odabira pozicija. Student je u primjeru odabrao tri pozicije ispod kojih mu se nalaze druge ponuđene.

Odabrali ste sljedeće teme

Redosljed	Naziv tvrtke	Tema	Akcija
1.	d.d.	Projektant i elektrotehnolog asinkronih motora	POGLEDAJ
2.	d.o.o.	Back-end developer	POGLEDAJ
3.	d.o.o.	Ljetna praksa na razvoju machine learning rješenja	POGLEDAJ

Ponuda tema

Tvrtka	Mjesta	Prijava prvi odabir	Tema	Akcija
d.o.o. [Web]	1	1	Back-end developer	POGLEDAJ
d.o.o. [Web]	1	2	Front-end developer	POGLEDAJ
d.o.o. [Web]	1	0	Ljetna praksa -	POGLEDAJ
d.o.o. [Web]	1	11	Ljetna praksa na razvoju machine learning rješenja	POGLEDAJ
d.o.o. [Web]	1	1	Ljetna praksa u odjelu „Business intelligence"	POGLEDAJ

Slika 42 – Izvedba kreiranja prioritetne liste

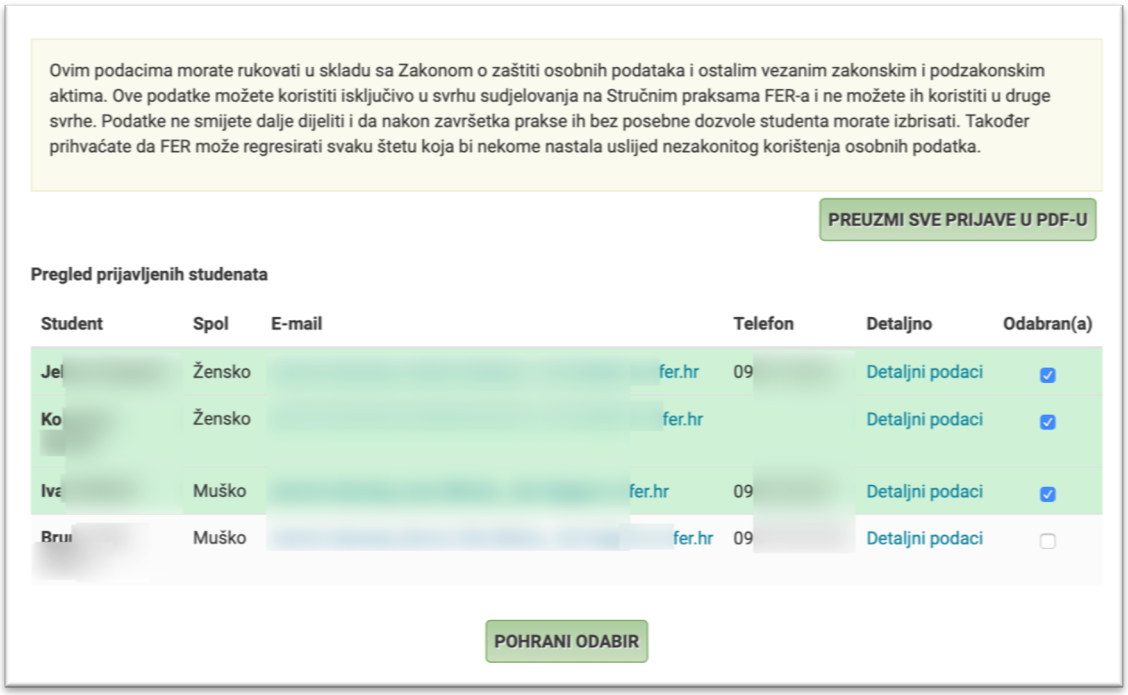
Pregledom pozicije, otvara se ekran s popisom svih pozicija te tvrtke što je vidljivo na Slici **43**. Jedna tvrtka može ponuditi više pozicija, a studentu se uz svaku poziciju ispisuje i koliko studenata će tvrtka prihvatiti na toj poziciji te koliko se do sad studenata prijavilo. Na primjer u kategoriji „C#/C++ developer" prijavio se jedan student, a tvrtka ih ukupno prihvaća šest (1 / 6).

Tema	Mjesta	Termin	#Prijava	Akcija
Backend Developer	2	01.07. - 01.09.2019.	**0** / 4	PRIJAVE
C#/C++ developer	2	01.07. - 30.09.2019.	**1** / 6	PRIJAVE
C++ Developer	1	01.08. - 30.09.2019.	**0** / 8	PRIJAVE
C#/Java programer	2	01.07. - 30.09.2019.	**3** / 18	PRIJAVE
Data Scientist	1	01.07. - 01.09.2019.	**1** / 22	PRIJAVE
DevOps Engineer	2	01.07. - 01.09.2019.	**0** / 0	PRIJAVE
Frontend Developer	2	01.07. - 01.09.2019.	**0** / 6	PRIJAVE
Fullstack developer	1	01.07. - 01.09.2019.	**0** / 1	PRIJAVE
Python developer	3	15.08. - 30.09.2019.	**0** / 16	PRIJAVE
Python Developer	1	01.08. - 30.09.2019.	**0** / 9	PRIJAVE
QA Engineer	1	01.07. - 01.09.2019.	**1** / 4	PRIJAVE
Web developer	2	01.07. - 30.09.2019.	**1** / 5	PRIJAVE

Ukupno odabrano studenata na svim pozicijama: 7.

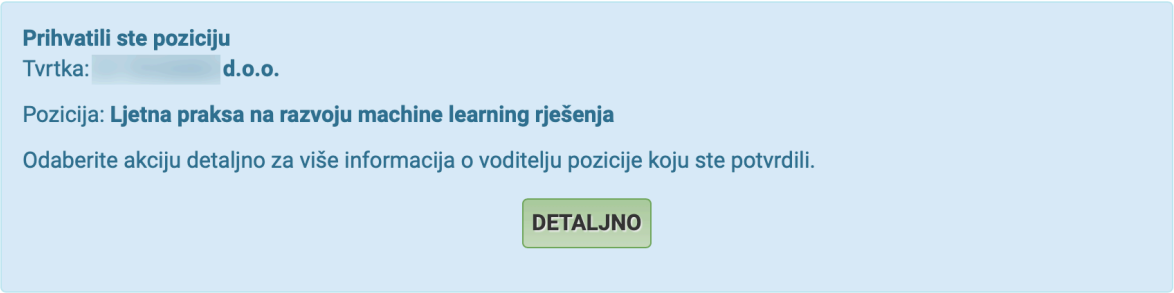
Slika 43 – Izvedba dodavanja pozicije na prioritetnu listu

Na Slici **44** prikazana je izvedba dijela potvrđivanja studenata od strane tvrtke. Tvrtka može pregledati detaljne podatke o svakom studentu te odabrati koje studente potvrđuje.



Slika 44 – Izvedba potvrđivanja studenata na pozicijama

Na Slici **45** prikazan je rezultat izvršavanja algoritma. Studenti se rangiraju prema prosječnoj ocjeni studija na glavnom popisu te se za njihove potvrđene odabire računa pozicija. Nakon što algoritam odredi najbolje rješenje, studentu se prikazuje na koju poziciju je potvrđen.



Slika 45 – Rezultat pridruživanja studenata na pozicije

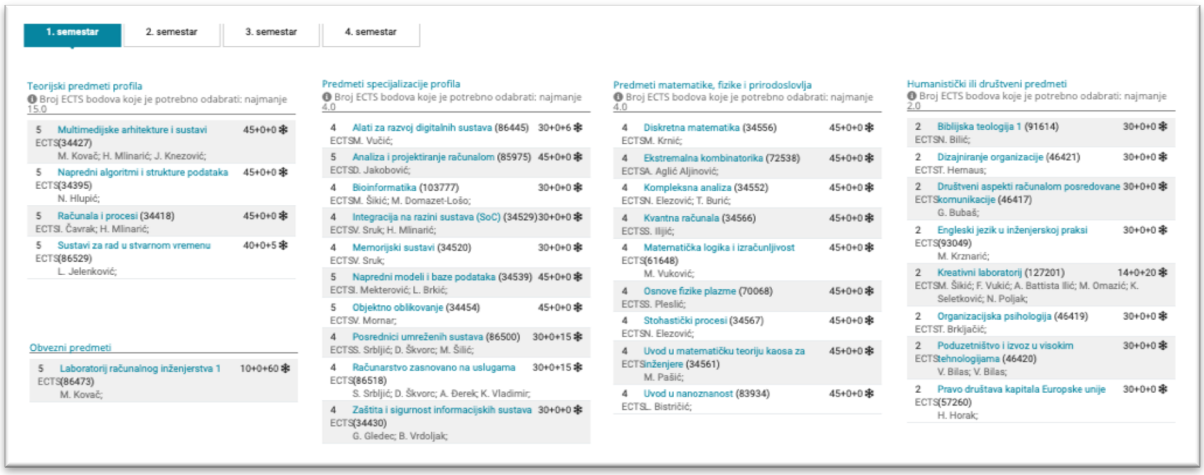
8.6. Odabir predmeta za upis s višeslojnom prioritetnom listom

U odabiru predmeta prilikom upisa semestra ili više godine na Fakultetu elektrotehnike i računarstva koristi se višeslojna lista. Konačan popis predmeta koji se upisuje pojedinom studentu određuje algoritam višestrukog pridruživanja s glavnim popisom.

Za razliku od standardnog procesa upisa u kojem studenti svoje predmete upisuju nakon završetka ispitnih rokova – u tjednu prije početka nastave, prednost korištenja ovog algoritma je što se proces upisa može odvijati puno ranije. Time su svi sudionici puno ranije upoznati s brojem studenata koje mogu očekivati, što uvelike olakšava izradu satnice kao i pripremu nastavnika za predavanja i vježbe.

Studenti su na glavnom popisu rangirani prema prosječnoj ocjeni studija. Svaki predmet određen je i brojem ECTS-bodova. Odabir predmeta za upis provodi se v*išeslojnom prioritetnom listom.* Predmeti su poredani u skladu sa strukturom studija koja ih grupira u izborne grupe. Svaka izborna grupa određuje broj predmeta koji će se upisati kroz broj potrebnih ECTS-bodova. Student može odrediti i vlastito gornje ograničenje u broju upisanih predmeta kroz ograničenje ECTS-bodova.

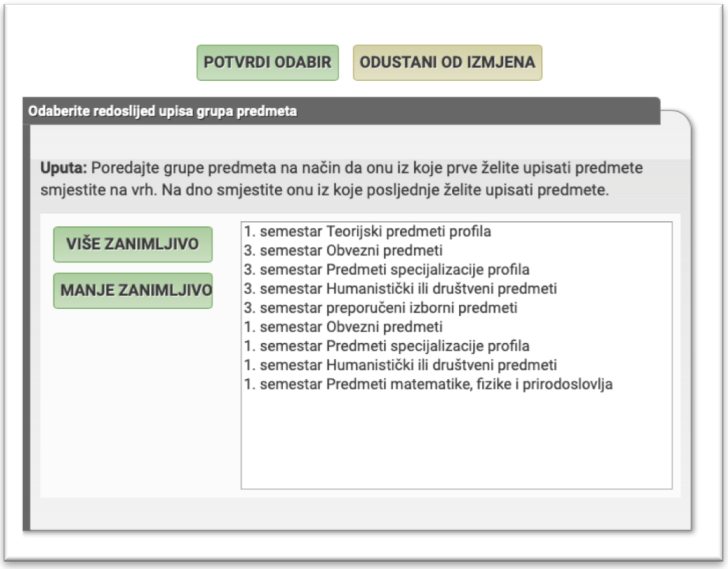
Na Slici 46 prikazan je primjer strukture studija prvog semestra na diplomskom studiju Računarstva, na profilu Računalno inženjerstvo na FER-u. Student treba odabrati tri predmeta (15 ECTS-a) iz grupe Teorijski predmeti profila, sve predmete iz grupe Obvezni predmeti, jedan predmet (4 ECTS-a) iz grupe Predmeti specijalizacije profila i tako dalje.



Slika 46 – Primjer strukture studija

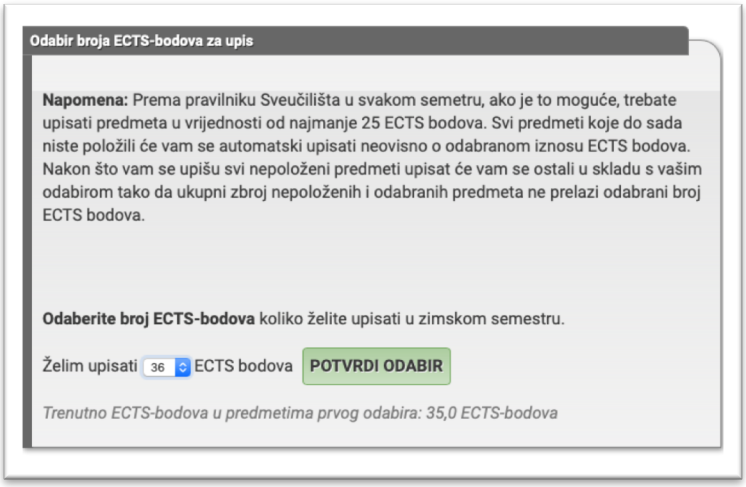
Odabir predmeta za upis putem višeslojne prioritetne liste implementiran je kroz *prioritetnu listu grupa predmeta* u kojoj se svaka grupa sastoji od *prioritetne liste predmeta*. Svaka lista predmeta određuje drugi broj odabranih predmeta. Ujedno se određuje i najveći broj ECTS-bodova koji se želi upisati.

Na Slici 47 prikazana je izvedba odabira prioriteta grupa *prioritetne liste grupe predmeta*. Promjenom redoslijeda grupa predmeta student može odrediti koja grupa predmeta ima prioritet pri upisu. Studentu iz primjera prvo će se upisati predmeti iz grupe predmeta 1. semestar Teorijski predmeti profila, pa zatim iz 1. semestar Obvezni predmeti i tako dalje. Svim studentima je redoslijed grupa inicijalno postavljen prema unaprijed određenoj strukturi studija.



Slika 47 – Prioritetna lista grupa predmeta

Na Slici 48 prikazana je izvedba odabira ograničenja ukupnog broja ECTS-bodova glavne višeslojne prioritetne liste, odnosno prioritetne liste grupe predmeta.

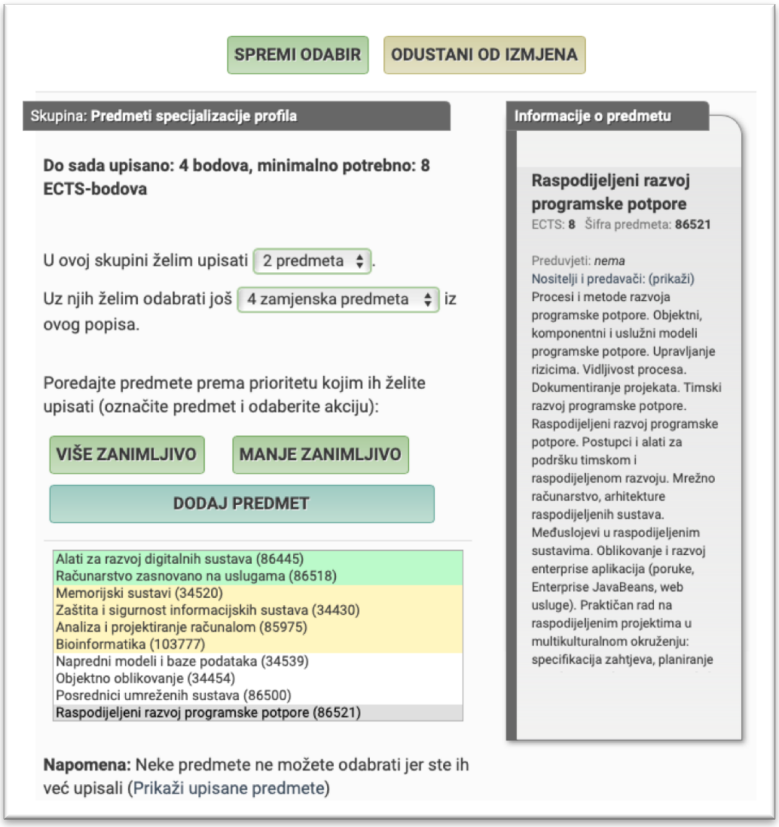


Slika 48 – Ograničenje ukupnog broja upisanih predmeta

Na Slici 49 prikazana je izvedba odabira predmeta unutar jedne izborne grupe, odnosno *prioritetne liste predmeta*. Student odabire:

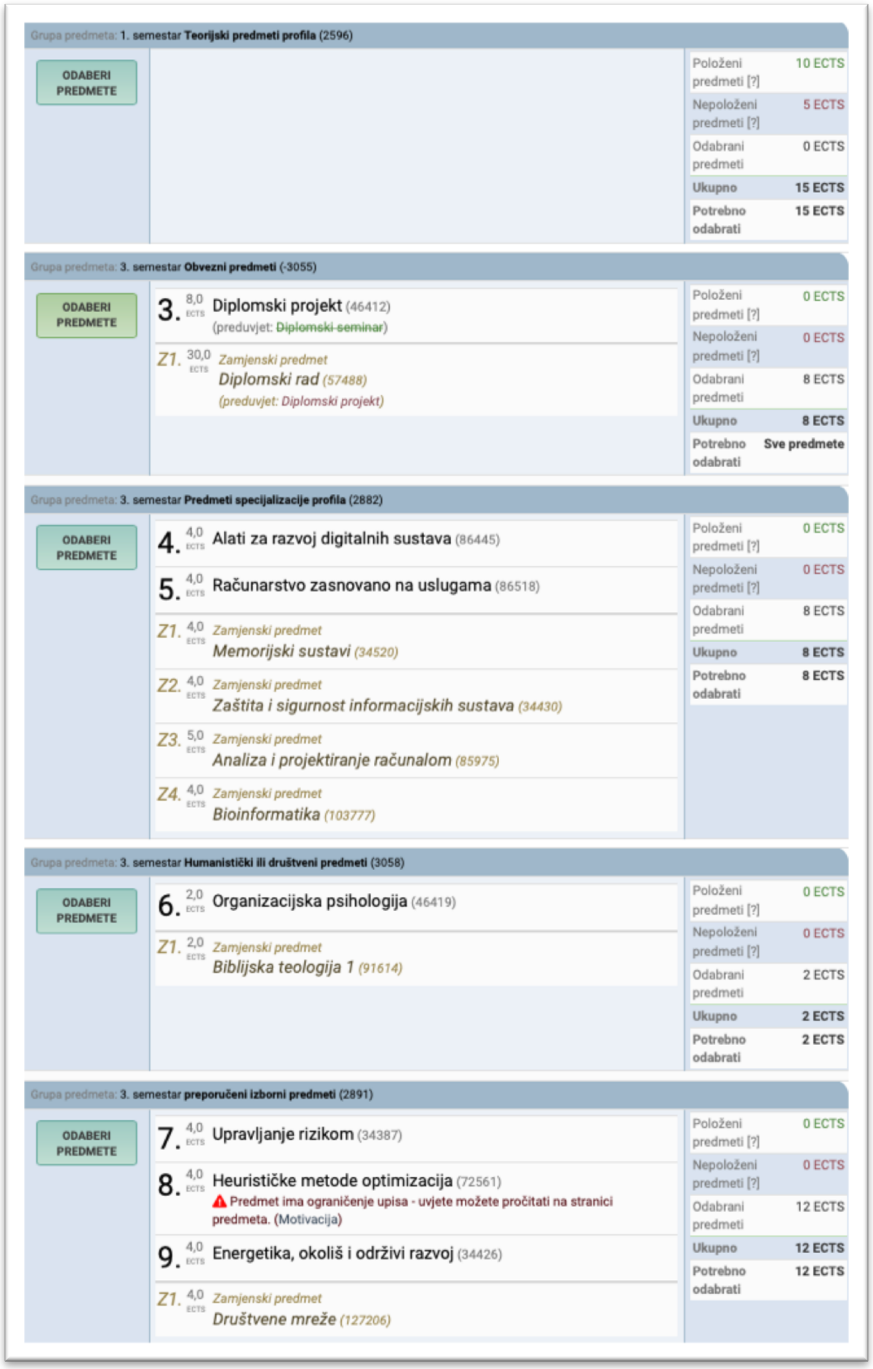
- koliko predmeta želi upisati u toj grupi (na Slici 49 odabrana su dva predmeta);
- koliko zamjenskih predmeta želi imati (na Slici 49 odabrana su tri zamjenska).

Odabirom ovih predmeta, student je ograničio prioritetnu listu predmeta na ukupno pet predmeta, od kojih bi student htio upisati najviše dva predmeta.



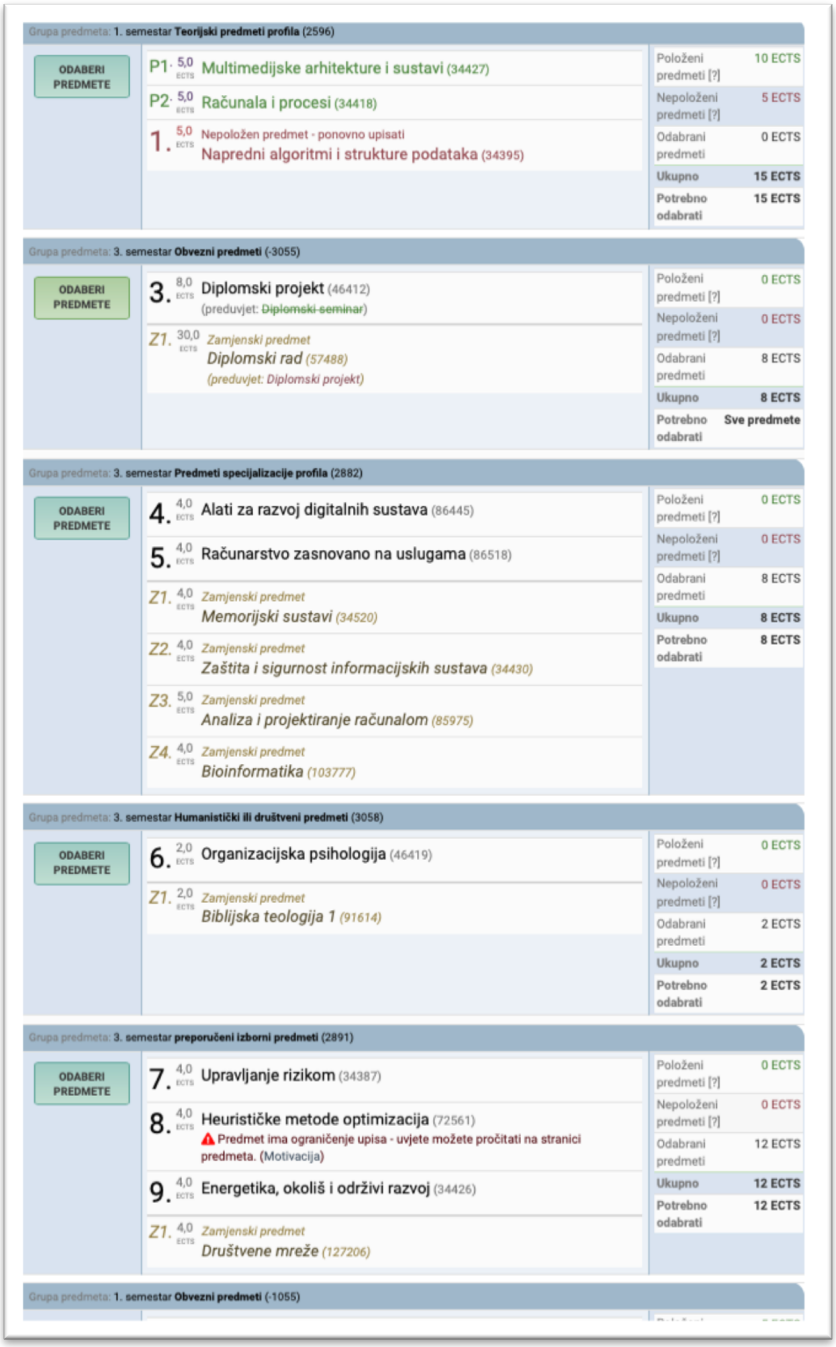
Slika 49 – Prioritetna lista grupe predmeta

Na Slici 50 prikazan je konačni prikaz višeslojne prioritetne liste. Nju čine prioritetne liste izbornih grupa, te prioritetne liste predmeta u svakoj od njih. U primjeru, student u izbornoj grupi predmeta Teorijski predmeti profila želi upisati tri od četiri odabrana predmeta u grupi. U grupi obveznih predmeta, odabrani su svi predmeti. U grupi Humanističkih ili društvenih predmeta student je odabrao dva predmeta od čega želi upisati jedan. Studentu se u desnom dijelu prikazuju i informacije o broju predmeta, odnosno ECTS-bodova, koje mora odabrati kako bi zadovoljio strukturu studija.



Slika 50 – Konačni odabir višeslojne prioritetne liste predmeta za upis

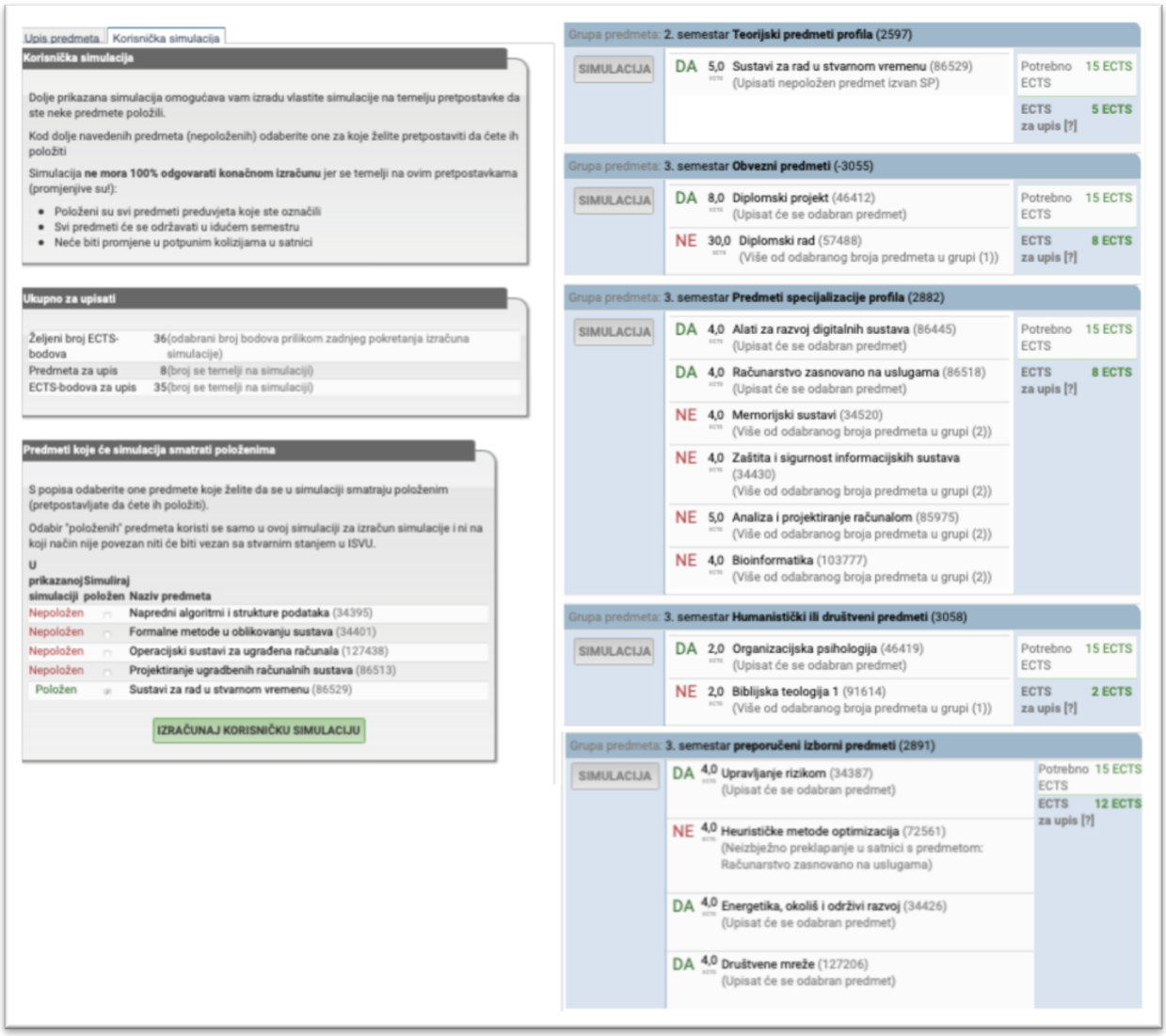
Na prioritetnoj listi student može također vidjeti i predmete koje još nije položio. Ti predmeti mu se dodaju na prioritetnu listu bez obzira na odabir kao ponovni upis. Prikazani su i predmeti koje je položio. Na Slici 51 prikazan je primjer takve liste. Predmet „Multimedijske arhitekture i sustavi“ kao i „Računala i procesi“ primjeri su već položenih predmeta. Predmet „Napredni algoritmi i strukture podataka“ je primjer predmeta koji nije položen.



Slika 51 – Konačna višeslojna prioritetna lista predmeta s prikazom predmeta

Svaki student može vidjeti rezultat izvođenja algoritma pridruživanja predmeta kao i rezultat izvođenja ovisno o predmetima koje planira položiti.

U primjeru na Slici 52 prikazana je korisnička simulacija jednog studenta. U lijevom dijelu studentu ispisani su predmeti koji nisu položeni, a u desnom dijelu je prikaz simulacije upisa predmeta temeljem svih uvjeta koje upis predmeta treba zadovoljiti.



Slika 52 – Korisnička simulacija upisa predmeta na primjeru jednog studenta

Osim korisničke simulacije, algoritam se pokreće više puta dnevno tijekom upisa. Algoritam temeljem aktualnih podataka računa koji studenti bi se upisali na koji predmet.

Na Slici 53 prikazan je rezultat izvođenja algoritma kojeg koriste nastavnici kako bi unaprijed znali očekivani broj studenata na predmetu. Uz svaki predmet prikazan je broj studenata koji će se upisati na pojedini predmet temeljem izračuna aktualne simulacije. U slučaju predmeta „Strojno učenje“ 323 studenta ostvarila su uvjet upisa predmeta, od čega njih 247 upisuje prvi put, a 73 ponavlja. Razlozi zbog kojih se pojedini studenti neće upisati, važan su dio zbirnog upisa simulacije predmeta jer se na vrijeme mogu uočiti potencijalni problemi i temeljem toga reagirati na vrijeme.

		izboru															
Diplomski	Strojno učenje	preporučeni izborni predmeti, Teorijski predmeti profila	103973	313	323	247	0	73	3	34	0	0	23	0	11	0	0
Diplomski	Računarstvo zasnovano na uslugama	Predmeti specijalizacije profila	86518	312	317	312	0	5	0	99	0	0	80	18	1	0	0
Preddiplomski	Okoliš, održivi razvoj i ublažavanje klimatskih promjena	Transverzalni predmeti	183409	291	292	292	0	0	0	305	0	0	305	0	0	0	0
Preddiplomski	Inženjerska ekonomika 1	Transverzalni predmeti	183404	257	257	257	0	0	0	340	0	0	340	0	0	0	0
Diplomski	Analiza i projektiranje računalom	Predmeti specijalizacije profila, preporučeni izborni predmet, Teorijski predmeti profila	85975	228	232	206	0	26	0	68	0	0	67	0	1	0	0
Diplomski	Laboratorij računarske znanosti 1	Obvezni predmeti	35224	220	223	209	0	11	3	5	0	0	2	0	0	0	3
Diplomski	Upravljanje rizikom	preporučeni	34387	214	216	214	0	0	2	30	0	0	25	1	4	0	0

Slika 53 – Zbirna simulacija izračuna broja studenata upisanih na pojedini predmet

8.7. Grupiranje grupa studenata

Grupiranje studenata koji rade u grupi kod istog nastavnika u prezentacijske grupe implementirano je algoritmom jednostrukog pridruživanja grupiranih elemenata.

Na preddiplomskom predmetu „Seminar“ koristi se ekspertno znanje o najuspješnijem studentu, a na predmetu „Diplomski seminar“ koristi se ekspertno znanje o području grupe.

Na Slici 54 prikazan je rezultat izvođenja algoritma jednostrukog pridruživanja grupiranih elemenata na primjeru predmeta „Seminar“. Studenti su pridruženi u prezentacijske grupe od 17 do 20 studenata.

Slika 54 – Rezultat izvođenja algoritma jednostrukog pridruživanja grupiranih elemenata

9. OCJENA PREDLOŽENIH METODA I MODELA

Ocjena predloženih metoda i modela pridruživanja napravljena je temeljem problema pridruživanja studenta projektu s obzirom na to da se on rješava algoritmom koji obuhvaća obje navedene metode. Ocjena je napravljena pomoću podataka nastalih kao rezultat izvođenja predmeta „Seminar“ na Fakultetu elektrotehnike i računarstva kroz četiri akademske godine. Ocjena rješenja provodila se nad podacima u akademskoj godini 2018./2019. tijekom koje je predmet upisalo 519 studenta, uz dostupna 244 projekta.

Prioritetne liste studenata prikupljane su prilikom upisa studenata na predmet. Ocjene studenata, potrebne za rangiranje studenata u glavnom popisu rangiranih studenata, preuzete su iz službene evidencije studentskih ocjena. Svi prikupljeni podaci korišteni su kao ulazni podaci u algoritam koji je izveden u programskom jeziku C#.

Program se izvodio na jednom računalu, s 100.000 iteracija izvođenja svake od udaljenosti (od 0,0 do 3,0) s povećanjem udaljenosti od 0,1. Tijekom prethodnih eksperimenata, udaljenosti manje od 0,1 nisu pokazivale dovoljno značajna odstupanja da bi se dalje razmatrala u ovoj disertaciji.

U prosjeku, algoritam je pronašao najbolje rješenje unutar 25.000 izvođenja, a izvođenje algoritma svake od udaljenosti trajalo je oko jedan sat na standardnom stolnom računalu. Najbolje ocijenjeno pridruživanje koristilo se kao referentna točka usporedbe s drugim udaljenostima.

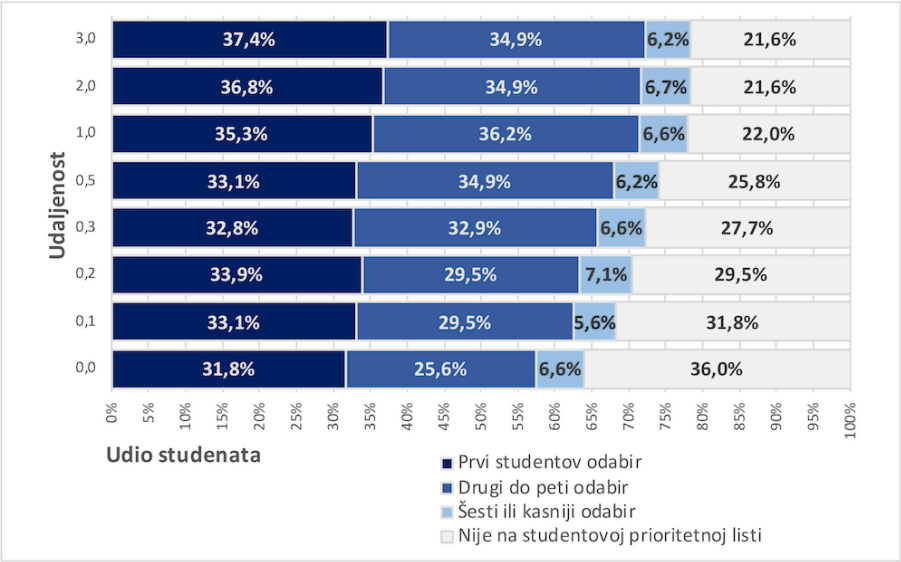
9.1. Metoda odabira redoslijeda elemenata

Metoda odabira redoslijeda elemenata pri rješavanju problema pridruživanja s prioritetnim listama u nastavnim procesima uporabom ekspertnog znanja koristi se u izradi glavnog popisa studenata. Pri tom se koristi parametar *udaljenost* (eng. distance), kojim se određuje koliko ocjene dva studenta smiju biti udaljene da bi se još uvijek smatrale jednakima. Analizira se kako povećanje udaljenosti utječe na nepridružene studente.

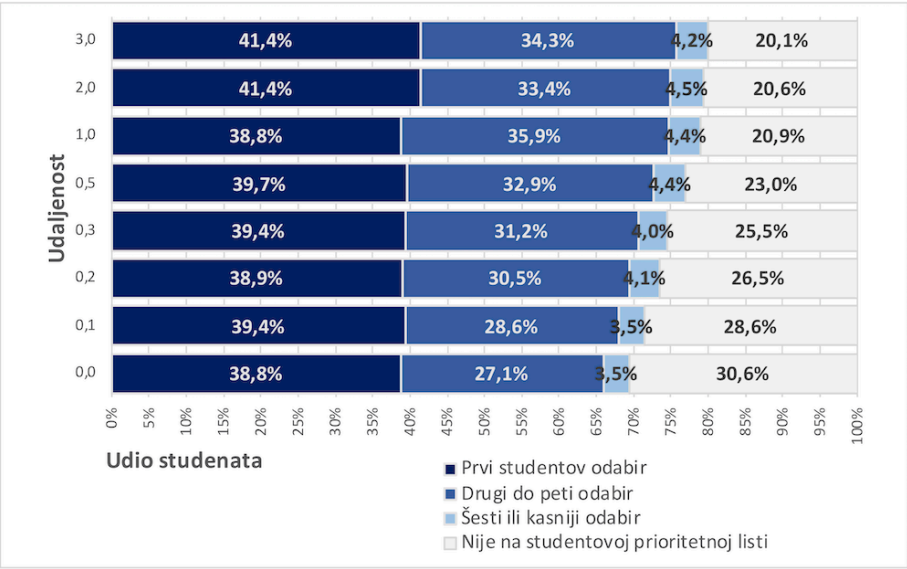
Povećanjem udaljenosti, broj dozvoljenih permutacija studenata u glavnom popisu rangiranih studenata raste. Posljedično, algoritam ima veći prostor pretrage, ali može dati i bolju ocjenu većeg broja rješenja. Sustav traži rješenje u kojem ima najmanje nepridruženih studenata, odnosno ono u kojem je najviše studenata pridruženo na svoj najviše željeni odabir. Povećanjem dozvoljene udaljenosti smanjuje se i broj nepridruženih studenata što je i jedan od ciljeva algoritma.

Utjecaj udaljenosti na broj nepridruženih studenata prikazan je na slikama 55 do 58. U akademskoj godini 2018./2019. povećanjem udaljenosti s 0 na 0,1 broj nepridruženih studenata smanjuje se za 4,2%. Dok je kod udaljenosti 0 broj nepridruženih studenata bio 36%, kod udaljenosti 0,1 smanjen je na 31,8%. Broj studenata pridruženih njihovim prvim odabirima na prioritetnoj listi povećava se s 31,8% na 33,1%. Daljnjim povećanjem udaljenosti na najveću vrijednost od 3,0 - broj nepridruženih studenata smanjuje se za 14,4% (s 36% na 21,6%).

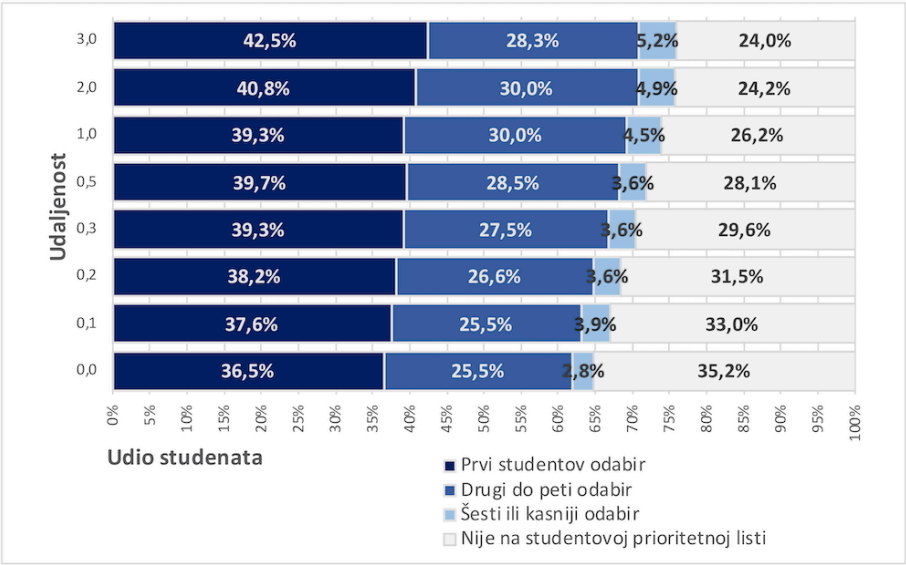
Sličan utjecaj se pojavljuje i u prethodnim godinama. Povećanjem udaljenosti od 0,1 broj nepridruženih studenata se smanjuje za gotovo 3%, a kod udaljenosti vrijednosti 3,0 smanjuje se do 12%.



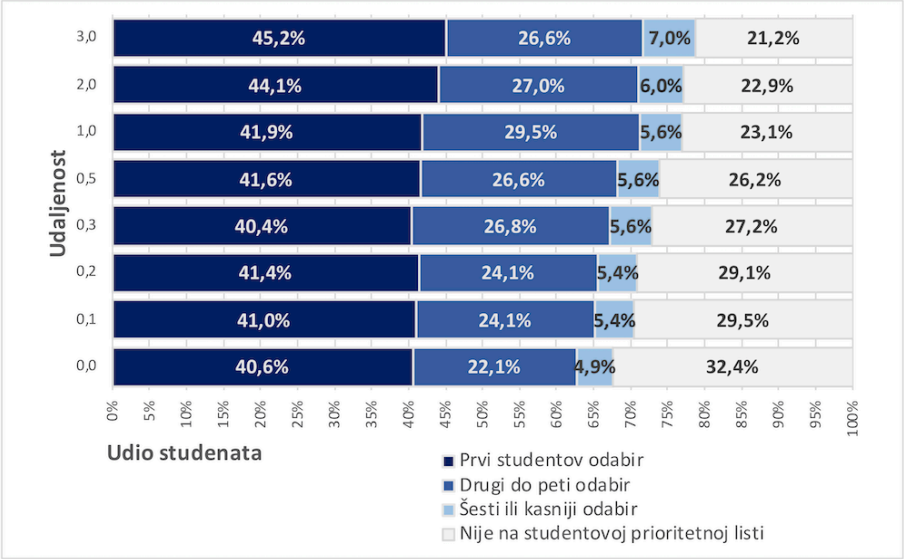
Slika 55 – Utjecaj udaljenosti na pridruživanje studenata u ak. god. 2018./2019.



Slika 56 – Utjecaj udaljenosti na pridruživanje studenata u ak. god. 2017./2018.



Slika 57 – Utjecaj udaljenosti na pridruživanje studenata u ak. god. 2016./2017.

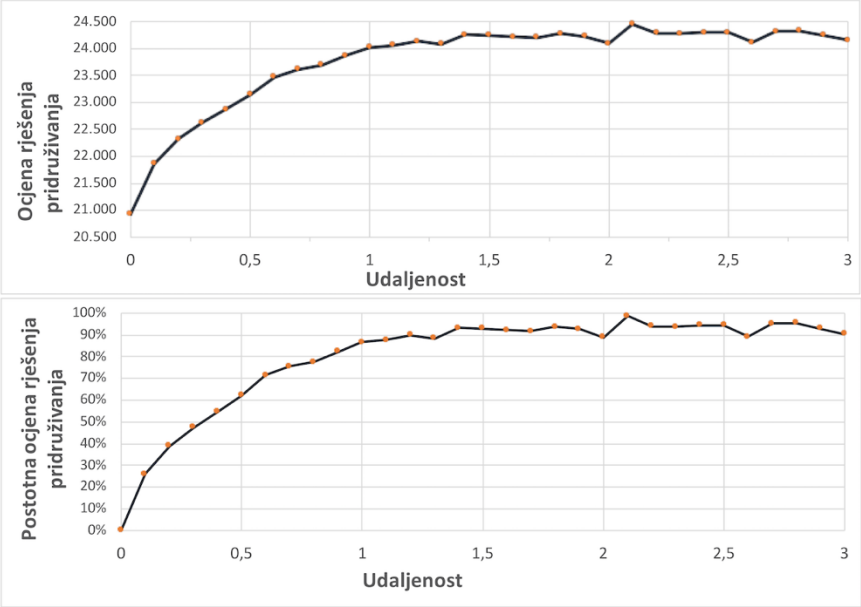


Slika 58 – Utjecaj udaljenosti na pridruživanje studenata u ak. god. 2015./2016.

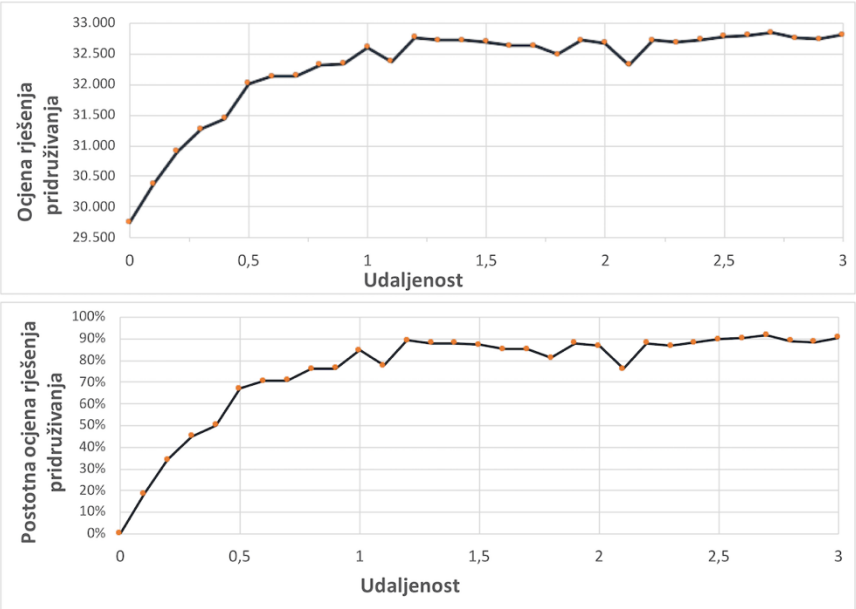
Čak i uz najveće povećanje udaljenosti, kod kojega ocjena studenta nema utjecaj već je važno samo što više studenata pridružiti grupi, i dalje ostaje nešto više od 20% nepridruženih studenata. Veliki postotak nepridruženih studenata posljedica je više utjecaja. Smanjivanje pojedinog utjecaja može smanjiti i broj nepridruženih studenata, ali mogućnost primjene rješenja može ovisiti o kontekstu u kojem se algoritam primjenjuje. Ovo su neki od prijedloga kako se navedeni utjecaji mogu smanjiti – u obimu u kojem je to dozvoljeno:

- Oko 2,5% studenata nije napravilo nikakav odabir;
    - Rješenje: proglasiti odabir obaveznim.
- Broj mjesta u grupama jednak je broju studenata;
    - Rješenje: povećati broj slobodnih mjesta.
- Studenti su strogo pridruženi temeljem njihovih prosječnih ocjena studija što rezultira pridruživanjem studenata s višim ocjenama u njima zanimljive grupe;
    - Rješenje: povećati vrijednost parametra udaljenosti na što veću vrijednost pri čemu se ne narušava traženi poredak.
- Studenti koji su niže rangirani odabiru grupe koje su popularnije;
    - Rješenje: utjecati na studente koji su niže rangirani da uz taj odabir naprave i odabir između grupa koje bira manje studenata. Studentima bi se uz grupe moglo ispisati koliko studenata ih je do sada biralo.
- Oko 14% grupa nije odabrao niti jedan student;
    - Rješenje: pokušati što ranije prepoznati takve grupe i utjecati na njihov sadržaj kako bi bili što atraktivniji i prepoznati među studentima čime bi ih studenti više birali.
- Broj odabranih grupa je premali;
    - Rješenje: utjecati na studente da odaberu što veći broj grupa.

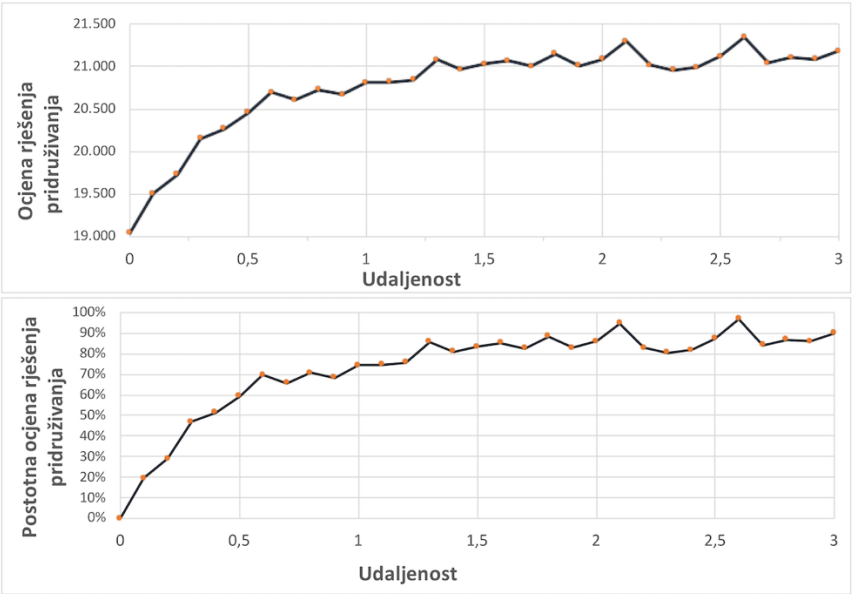
Za druga mjeru ocjene algoritma promatra se vrijednost najveće ocjene koju je izračunao algoritam, ovisno o udaljenosti, što je prikazano na slikama 59 do 62. Već i malim povećanjem udaljenosti za 0,1 značajno se povećava postotak pridruživanja. Kod vrijednosti udaljenosti veće od 1,0 promjena je zanemariva i u većoj je mjeri rezultat pronalaska najboljeg rješenja genetičkog algoritma nego stvarnog utjecaja udaljenosti na rezultate. Višestrukim ponavljanjem ovog eksperimenta graf je uvijek zadržavao prikazani oblik, a vidi se da ima i jednaku krivulju podataka kroz više godina.



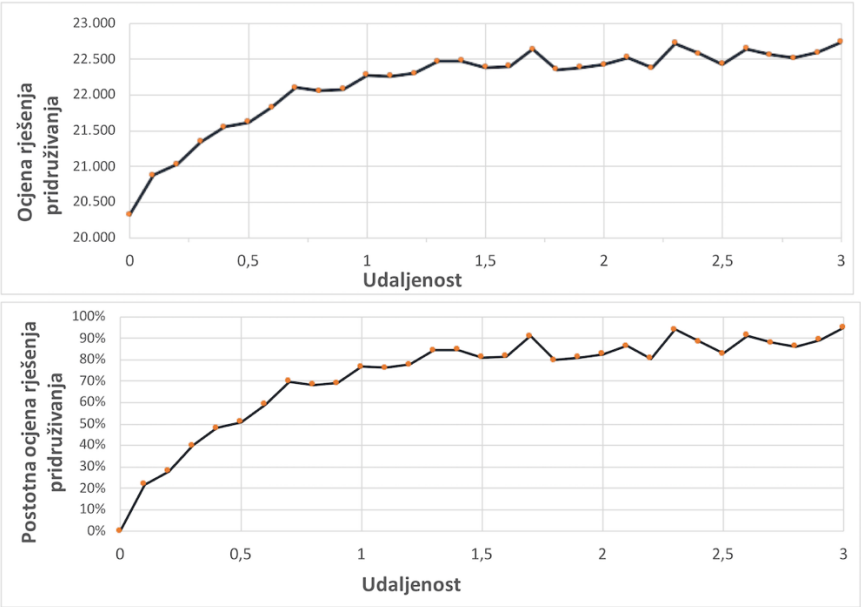
Slika 59 – Utjecaj udaljenosti na postotak pridruživanja studenata u ak. god. 2018./2019.



Slika 60 – Utjecaj udaljenosti na postotak pridruživanja studenata u ak. god. 2017./2018.



Slika 61 – Utjecaj udaljenosti na postotak pridruživanja studenata u ak. god. 2016./2017.



Slika 62 – Utjecaj udaljenosti na postotak pridruživanja studenata u ak. god. 2015./2016.

Jedan od zahtjeva pri pridruživanju studenata u grupe na primjeru predmeta „Seminar“ je da studenti s boljim ocjenama imaju prioritet u pridruživanju. Postavljanjem vrijednosti parametra *udaljenost* na najveću vrijednost (3,0), prosječna ocjena koju je student ostvario do tada na studiju gubi svoj utjecaj, a to je u suprotnosti s navedenim zahtjevom.

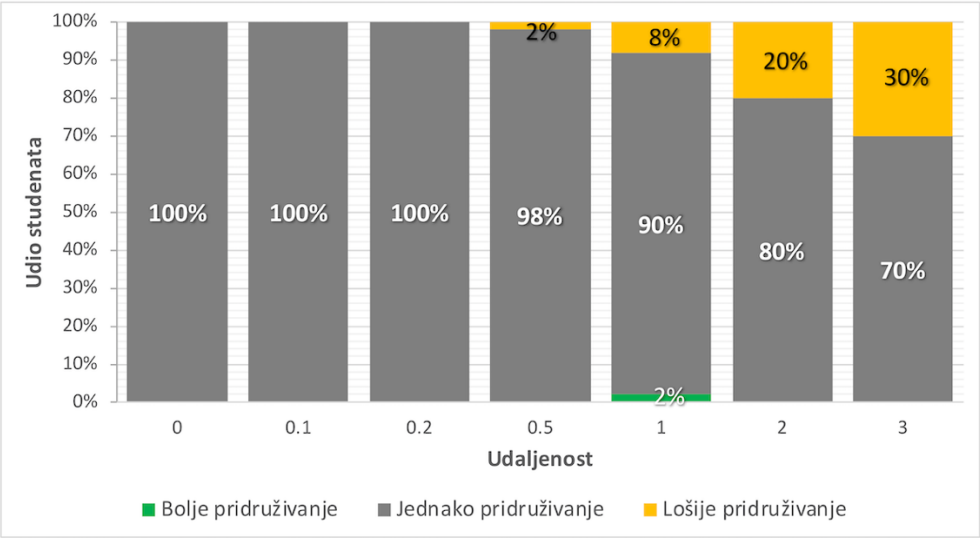
Ispunjavanje uvjeta zadržavanja prednosti studenta s boljim ocjenama postiže se odabirom manje vrijednosti parametra udaljenosti kako ne bi utjecao na odabir najbolje rangiranih studenata. Istovremeno, parametar udaljenost potrebno je postaviti što većim tako

da se smanji broj nepridruženih studenata i više studenata bude pridruženo projektu sa svoje prioritetne liste.

Eksperimentalno se vrijednost *udaljenosti* može utvrditi na sljedeći način:

1. Računa se raspodjela studenata dobivena uz vrijednost 0;
2. Udaljenost se povećava toliko dugo dok je raspodjela najboljih 10% studenata u usporedbi s izračunatom raspodjelom kada je vrijednost udaljenosti 0 i dalje nepromijenjena.

Rezultat takve analize, nad podacima u akademskoj godini 2018./2019., prikazan je na Slici 63. Kada je vrijednost parametra udaljenosti 0,5 utječe se na raspodjelu samo 2% od najboljih 10% studenata (0,2% ukupnog broja studenata), što je u ovom slučaju prihvatljiv utjecaj s obzirom na to da se istovremeno broj nepridruženih studenata smanjio za 10% ukupnog broja studenata. Kada se kao vrijednost parametra udaljenosti odabere 1, postoji jedan student (2% najboljih studenata) koji je pridružen u grupu bolje rangiranu na prioritetnoj listi. Istovremeno, 8% najboljih studenata je pridruženo u grupu lošije rangiranu od dosadašnje zbog čega ova vrijednost udaljenosti nije prihvatljiva.



Slika 63 – Utjecaj parametra udaljenosti u pridruživanju najboljih 10% studenata

9.2. Metoda pridruživanja nepridruženih elemenata

Metoda pridruživanja nepridruženih elemenata u nastavnim procesima temeljem zaključaka izvedenih iz konteksta odabira koristi se u algoritmu izrade proširene prioritetne liste. Ova metoda koristi se kako bi bolji, nepridruženi, studenti bili pridruženi u kvalitetniju grupu nego što bi bili pridruživanjem slučajnim odabirom.

Ova metoda ne smanjuje izravno broj nepridruženih studenata, ali mijenja način na koji se nepridruženi studenti pridružuju u grupe. Kada se student ne može pridružiti nijednoj grupi s njegove prioritetne liste, algoritam će ga pokušati pridružiti uporabom proširene prioritetne liste. Ako se student ne može pridružiti nijednoj grupi ni s ove liste, tada će se pridružiti nasumičnim odabiru grupe u kojoj ima slobodnih mjesta. Proširena prioritetna lista računa se za studente koji su napravili najmanje jedan odabir.

Eksperimentalnim pridruživanjem studenata u grupe, uporabom proširene prioritetne liste, pridruženo je oko 90% studenata, a samo oko 10% studenata je pridruženo nasumičnim odabirom. Bez korištenja ove nove predložene metode do 35% studenata ostaje nepridruženo. Navedena metoda značajno poboljšava pridruživanje studenata u usporedbi s prethodnim načinima u kojima se raspodjela provodila isključivo nasumičnim odabirom.

Ocjena algoritma izrade proširene prioritetne liste provedena je nad svim studentima na predmetu „Seminar“ u akademskoj godini 2018./2019. a koji su odabrali između pet i deset grupa uporabom metoda sličnih onima koje se upotrebljavaju u strojnom učenju – skraćivanje liste. Ocjena obuhvaća više od 50% studenata koji su napravili odabir.

Skraćena lista dobivena je uklanjanjem zadnjeg odabira studenta na prioritetnoj listi. Ostali odabiri i njihov redoslijed su zadržani. Liste koje su imale više od deset grupa nisu predstavljale dobar model usporedbe s obzirom na to da je većina studenata odabrala između pet i deset grupa. Dodatno, tijekom ručne provjere takvih lista utvrđeno je da je velik broj studenata odabirao nepovezane grupe. Skraćivanje liste kraće od pet grupa nije pružilo dovoljno podataka algoritmu pronalaska odgovarajućeg para.

Analiza izrade proširene prioritetne liste proveden je na sljedeći način:

(i) Algoritam izrade proširene prioritetne liste poziva se dva puta. Prvi puta je ulazni parametar cijela prioritetna lista, a drugi puta prioritetna lista bez zadnjeg odabira;

(ii) Uspoređuje se zadnji odabir iz prvog poziva (cijela prioritetna lista) i prva grupa u proširenoj prioritetnoj listi iz drugog poziva (skraćena prioritetna lista);

(iii) Ako su grupe u oba slučaja jednake, podudaranje se smatralo uspješnim.

Ova usporedba koristila se u izračunu najboljih težinskih faktora kod izračuna prioritetne liste. Algoritam koristi četiri vrijednosti sličnosti C1, C3, C5 i CMax. Svaka od vrijednosti se određuje uporabom težinskih faktora F1, F3, F5 i FMax. Kod određivanja vrijednosti težinskih faktora prvo se provela eksperimentalna usporedba odnosa kombinacija težinskih faktora uz vrijednosti: 1, 1.000 i 10.000. Ove vrijednosti odabrane su u cilju

naglašavanja međusobnog utjecaja faktora. Druge razmatrane vrijednosti nisu značajno utjecale na rezultate analize.

Slika 64 prikazuje rezultate analize s ovim faktorima. Postotak sličnosti prikazan je u svakoj od ćelija pri čemu crveno predstavlja manju postotnu vrijednost, a zeleno veću.

		F5 = 1			F5 = 1,000			F5 = 10,000		
		FMAX = 1	FMAX = 1,000	FMAX = 10,000	FMAX = 1	FMAX = 1,000	FMAX = 10,000	FMAX = 1	FMAX = 1,000	FMAX = 10,000
F1 = 1	F3 = 1	89,1%	70,7%	70,7%	80,8%	78,2%	71,6%	80,8%	81,2%	78,2%
	F3 = 1,000	71,6%	82,5%	72,9%	84,3%	84,7%	74,2%	82,1%	83,8%	79,5%
	F3 = 10,000	71,6%	76,9%	82,5%	76,0%	79,0%	83,0%	84,3%	86,0%	84,7%
F1 = 1,000	F3 = 1	41,5%	79,9%	72,5%	84,7%	81,7%	72,5%	81,7%	83,0%	78,2%
	F3 = 1,000	72,5%	86,9%	73,8%	90,0%	89,1%	75,5%	83,8%	86,0%	79,0%
	F3 = 10,000	72,1%	78,6%	83,0%	76,0%	79,0%	83,4%	84,3%	87,3%	85,2%
F1 = 10,000	F3 = 1	41,5%	58,1%	79,9%	51,5%	65,9%	79,9%	84,7%	86,5%	81,7%
	F3 = 1,000	49,8%	62,0%	80,3%	55,9%	69,9%	82,1%	88,2%	89,5%	82,1%
	F3 = 10,000	72,5%	76,4%	86,9%	75,1%	80,8%	88,2%	90,0%	91,3%	89,1%

Slika 64 – Rezultati analize uz težinske faktore F1, F3 i F5 = { 1; 1.000; 10.000 }

Iz analize prikazane na Slici 64 vidljiva je najbolja korelacija težinskih vrijednosti. Zbog toga se provodi i drugi krug usporedbe, u kojem se pretražuje područje koje se pokazalo najboljom kod određivanja vrijednosti težinskih faktora. Najbolji rezultati ostvaruju se kada je F1 ≈ F3 ≈ F5, a FMax ima manju vrijednost od drugih težinskih faktora.

Algoritam računa sličnost prioritetnih lista studenata kao postotak (0% do 100%) i zbog toga zbroj svih težinskih vrijednosti mora biti jednak 1. Da bi se to postiglo, faktor FMax određen je kao:

$$FMax = 1 - F1 - F3 - F5.$$

Za ostvarivanje svih gore navedenih uvjeta, F1, F3 i F5 trebaju biti oko 0,3, jer tada tražimo vrijednosti u kojima su navedeni faktori podjednaki, a Fmax će biti manji od njih.

Slika 65 prikazuje rezultate analize nekoliko težinskih faktora. To su vrijednosti: 0,25; 0,28; 0,3 i 0,33. Uporabom izračunatih vrijednosti težinskih faktora (F1=0,3; F3=0,28; F5=0,25; Fmax=0,17) postignuta je uspješnost sličnosti od 93,1%. Također su promatrane i druge vrijednosti, ali njihova sličnost bila je niža od navedenog rezultata.

		FMAX = 1 - F1 - F3 - F5			
		F5 = 0,25	F5 = 0,28	F5 = 0,3	F5 = 0,33
F1 = 0,25	F3 = 0,25	88,3%	89,2%	89,2%	90,5%
	F3 = 0,28	89,2%	90,9%	90,0%	91,8%
	F3 = 0,3	90,5%	91,8%	91,3%	92,2%
	F3 = 0,33	90,9%	92,2%	91,8%	90,9%
F1 = 0,28	F3 = 0,25	90,0%	91,3%	90,5%	90,9%
	F3 = 0,28	91,8%	91,8%	92,2%	92,2%
	F3 = 0,3	92,6%	91,8%	91,3%	91,3%
	F3 = 0,33	91,3%	91,8%	91,8%	90,9%
F1 = 0,3	F3 = 0,25	90,5%	91,8%	91,3%	91,3%
	F3 = 0,28	93,1%	91,8%	90,9%	91,3%
	F3 = 0,3	90,9%	90,9%	90,9%	91,3%
	F3 = 0,33	91,8%	92,2%	92,2%	90,5%
F1 = 0,33	F3 = 0,25	91,3%	92,2%	91,8%	91,3%
	F3 = 0,28	92,6%	91,8%	91,8%	92,2%
	F3 = 0,3	92,2%	91,8%	91,3%	91,8%
	F3 = 0,33	92,6%	91,3%	92,6%	89,6%

Slika 65 – Tablica težinskih vrijednosti F1, F3, F5 = { 0,25; 0,28; 0,3; 0,33 }

Provedena je i ručna analiza pogrešaka u izračunu proširene prioritetne liste temeljem koje je zaključeno da je glavni razlog neuobičajen odabir grupa od strane studenata. U većini slučajeva, rezultat pogreške je odabir nepovezane zadnje grupe na studentovoj prioritetnoj listi u odnosu na područja prethodno odabranih grupa.

Težinske vrijednosti ovise o odabiru koji su studenti napravili uz drugačiji studentski odabir težinske vrijednosti mogu se razlikovati u cilju dobivanja najboljeg rješenja. Sama metoda određivanja najboljih težinskih vrijednosti neovisna je o odabirima koje studenti rade i može se koristiti pri određivanju težinskih vrijednosti ali i drugih studentovih odabira.

9.3. Primjena metoda u algoritmima

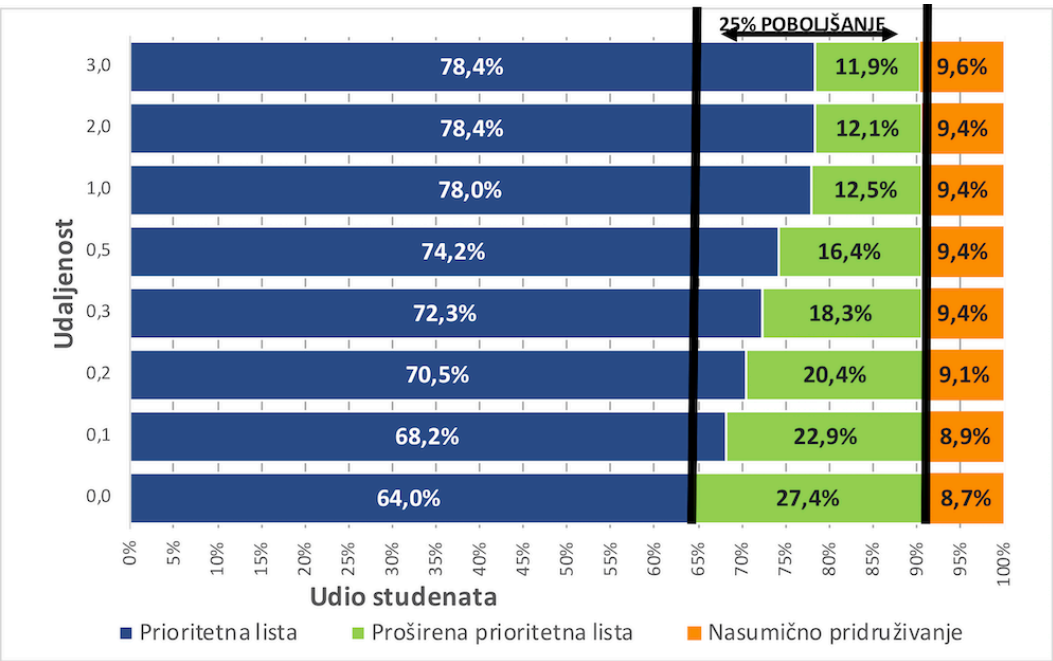
Kombiniranjem obje metode unutar predloženih algoritama, odnosno algoritma strogog jednostrukog pridruživanja s glavnim popisom, na primjeru pridruživanja studentu grupe na predmetu „Seminar“, pokazuje se poboljšanje od 20% do 25% u smanjenju broja nepridruženih studenata odnosno u broju studenata koji su nasumično pridruženi.

Kada je udaljenost nula, tada se poboljšanje temelji na metodi izrade proširene liste. Povećanjem udaljenosti smanjuje se broj nepridruženih studenata zbog nove metode odabira redoslijeda elemenata.

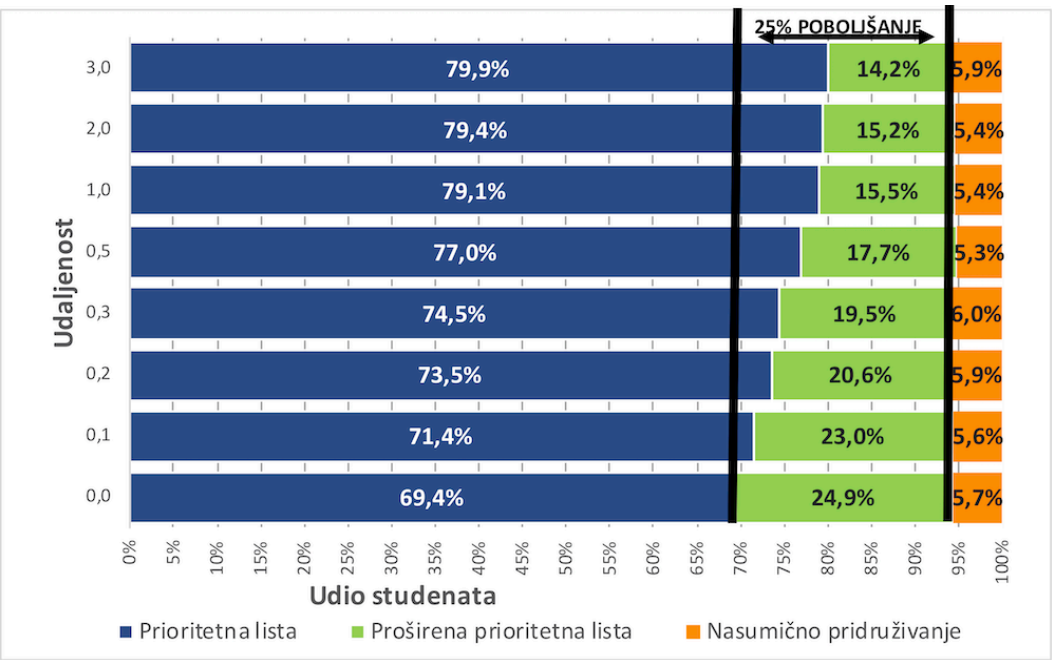
Utjecaj kombinacije korištenja ovih metoda u predloženom algoritmu strogog jednostrukog pridruživanja s glavnim popisom u odnosu na različite vrijednosti parametra udaljenosti prikazana je na slikama 66 do 69.

U primjeru akademske godine 2018./2019., uz vrijednost parametra udaljenosti od 0,5, kod koje nema značajne promjene u pridruživanju za najboljih 10% studenata, broj nepridruženih studenata smanjio se za više od 10%, a dodatnih 16,4% studenata pridruženo je uporabom proširene prioritetne liste, čime je utjecaj kombinacije ovih dviju metoda povećalo kvalitetu pridruživanja za 25% studenata.

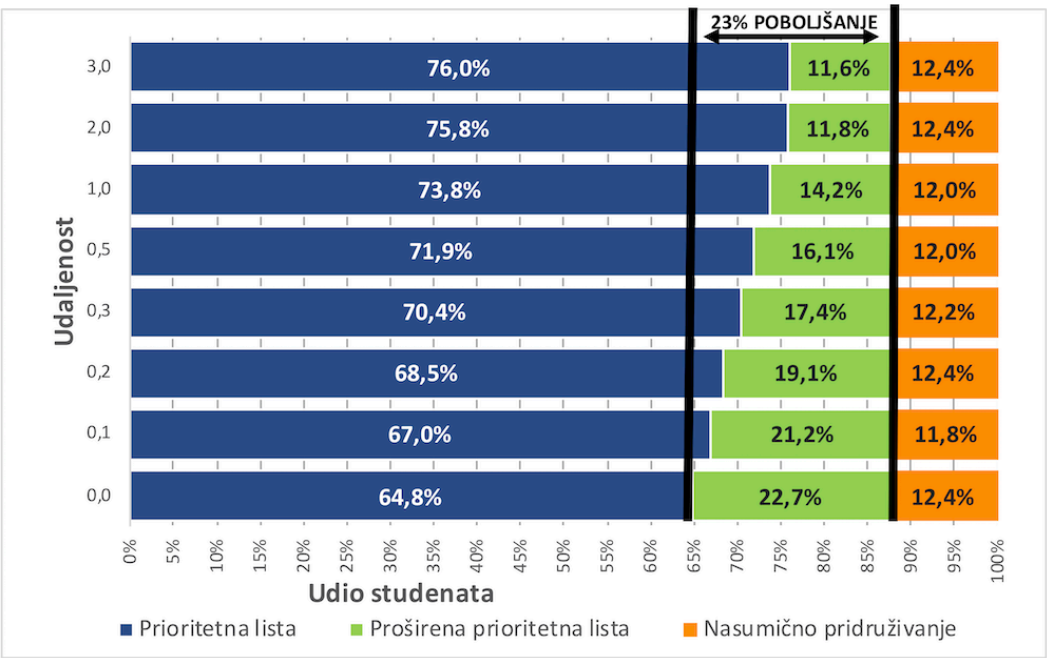
U ostalim akademskim godinama ostvareni su slični rezultati, te je ukupno povećanje iznosilo do 22%.



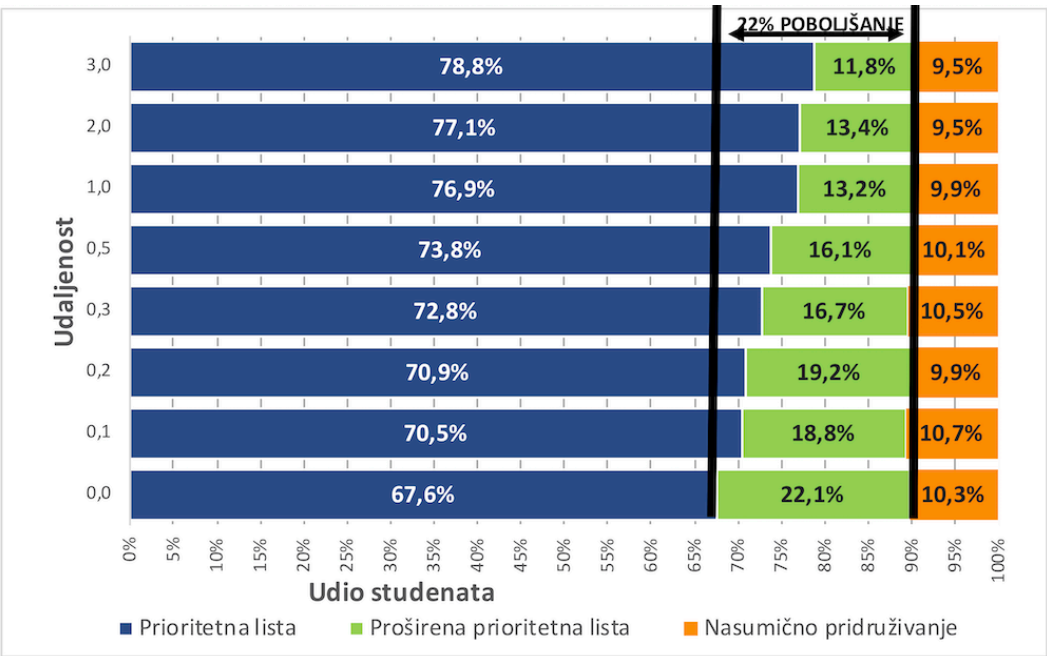
Slika 66 – Raspodjela temeljem prioritetnih i proširenih lista u ak. god. 2018./2019.



Slika 67 – Raspodjela temeljem prioritetnih i proširenih lista u ak. god. 2017./2018.



Slika 68 – Raspodjela temeljem prioritetnih i proširenih lista u ak. god. 2016./2017.



Slika 69 – Raspodjela temeljem prioritetnih i proširenih lista u ak. god. 2015./2016.

9.4. Usporedba predloženih algoritama s Algoritmom odgođenog prihvaćanja

Rad predloženih algoritama uspoređen je s Algoritmom odgođenog prihvaćanja (eng. *Deferred Acceptance algorithm, Gale-Shapley algorithm*) [60]. Kao model predloženih algoritama koristi se Algoritam strogog jednostrukog pridruživanja s glavnim popisom s obzirom na to da je to ujedno i temeljni algoritam, a i koristi obje predložene metode: metodu odabira redoslijeda elemenata i metodu za pridruživanje nepridruženih elemenata.

Algoritam odgođenog prihvaćanja zahtijeva jednak broj studenata i slobodnih mjesta u grupama. Za razliku od njega, predloženi algoritam, može ispravno raditi i u uvjetima kada je studente potrebno ravnomjerno pridružiti, a broj slobodnih mjesta je nešto veći od broja studenata. U cilju ujednačavanja ulaznih parametara oba algoritma unaprijed je određen broj slobodnih mjesta u svakoj od grupa. Kod određivanja broja mjesta, predloženi algoritam je izvršen jednom, te je njegov rezultat pridruživanja uzet za određivanje broja slobodnih mjesta po grupama. Time je osigurano da popularnije grupe imaju više mjesta, te da oba algoritma rade u jednakim uvjetima.

U algoritmu odgođenog prihvaćanja svake grupe, prioritetna lista studenata je automatski generirana temeljem glavne prioritetne liste. Algoritam zahtijeva strogo poredanu listu zbog čega su studenti s istim ocjenama za potrebe usporedbe algoritama poredani nasumično unutar istih ocjena. S obzirom na to da poredak studenata utječe na rezultate

izvođenja algoritma odgođenog prihvaćanja ovaj algoritam je pokrenut 15.000 puta pri čemu su svaki puta studenti poredani drugačije. Za potrebe usporedbe odabrano je rješenje koje je kao rezultat dalo najmanji broj nepridruženih studenata.

Kako bi algoritam bio usporediv s predloženim, u predloženom se ne koristi algoritam proširene prioritetne liste niti nasumična raspodjela studenata. Time studenti kojima nije moguće pridružiti grupu ostaju nepridruženi. Uz ove dvije promjene, rezultat izvođenja predloženog algoritma s vrijednosti udaljenosti nula i algoritma odgođenog prihvaćanja bi trebao dati podjednak rezultat.

	Algoritam odgođenog prihvaćanja	Udaljenost = 0	Udaljenost = 0.1	Udaljenost = 0.2	Udaljenost 0.3	Udaljenost = 0.5	Udaljenost 1	Udaljenost 2	Udaljenost = 3
Broj studenata alociranih u jednake grupe kao i algoritmom odgođenog prihvaćanja	100,0%	95,7%	84,8%	79,4%	77,2%	70,6%	58,6%	46,7%	46,4%
Nepridruženi studenti	24,1%	23,3%	20,6%	18,7%	17,5%	16,2%	14,5%	14,1%	13,9%
Allocated students									
1. mjesto	36,0%	36,2%	37,0%	37,2%	37,4%	38,3%	38,9%	39,3%	38,9%
2. mjesto	16,6%	16,8%	17,9%	17,9%	16,6%	17,1%	16,4%	18,9%	16,6%
3. mjesto	9,1%	8,9%	11,0%	10,4%	10,8%	9,8%	10,6%	8,7%	10,4%
4. mjesto	5,6%	5,8%	5,2%	6,0%	6,7%	7,3%	6,7%	5,2%	7,5%
5. mjesto	3,7%	3,7%	3,3%	4,2%	4,8%	4,2%	4,2%	4,6%	3,7%
6. mjesto	2,1%	2,1%	2,1%	2,1%	1,5%	1,5%	3,1%	2,5%	2,7%
7. mjesto	1,0%	1,0%	0,6%	0,8%	1,3%	1,2%	1,9%	2,9%	1,9%
8. mjesto	1,0%	1,2%	1,2%	1,7%	1,7%	1,9%	1,5%	1,9%	2,9%
9. mjesto	0,2%	0,4%	0,4%	0,4%	0,6%	1,0%	1,2%	0,8%	0,8%
10 i više mjesto	0,8%	0,8%	0,8%	0,6%	1,0%	1,3%	1,0%	1,2%	0,8%
Ukupno pridruženo	**75,9%**	**76,7%**	**79,4%**	**81,3%**	**82,5%**	**83,8%**	**85,5%**	**85,9%**	**86,1%**

Slika 70 – Usporedba rezultata s Algoritmom odgođenog prihvaćanja

Slika 70. prikazuje raspodjelu studenata uporabom oba algoritma. Usporedba s Algoritmom odgođenog prihvaćanja napravljena je za nekoliko vrijednosti parametra udaljenosti: 0; 0,1; 0,2; 0,5 i 3. Kada je vrijednost parametra udaljenosti 0, rezultati su slični onima koje ostvaruje Algoritam odgođenog prihvaćanja – nešto više od 95% studenata je pridruženo jednakim grupama kao i kod Algoritma odgođenog prihvaćanja. Bolji rezultat je posljedica genetskog algoritma koji se koristi u predloženom algoritmu. Genetski algoritam omogućava bolju pretragu rješenja. S povećanjem parametra udaljenosti, prostor rješenja se povećava, pa i sâm algoritam pronalazi bolje rješenje.

Promjena u pridruživanju posljedica je studenata koji imaju jednake prosječne ocjene i slično rangirane grupe. Ovisno o tome kako se ti studenti poredaju na glavnom popisu rangiranih studenata, više rangirani student ima prednost u pridruživanju i time će posljedično utjecati na pridruživanje grupa drugog studenata. Drugačiji poredak samo dva studenta može utjecati na niz promjena kod niže rangiranih studenata.

U dobivenoj raspodjeli na Slici 70 predloženi algoritam daje najmanje nepridruženih studenata kada je vrijednost parametra udaljenosti najveća, ali čak i s manjim povećanjem udaljenosti od 0,5, u kojoj pridruživanje nema utjecaj na najboljih 10% studenata, broj nepridruženih studenata se značajno smanjuje, za 8%.

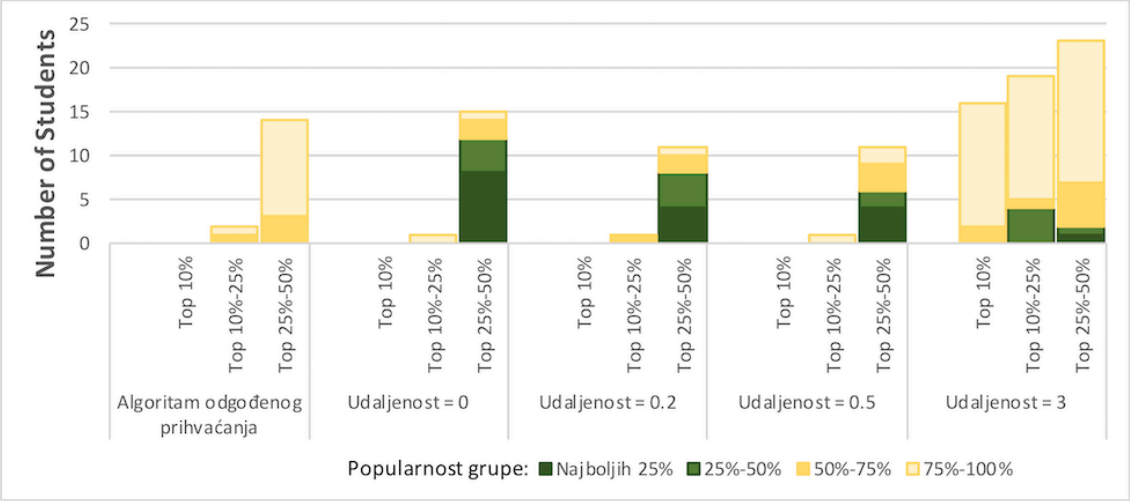
Kada se uspoređuje broj studenata koji su pridruženi u bolje rangiranu grupu na prioritetnoj listi, predloženi algoritam daje bolje rezultate od Algoritma odgođenog prihvaćanja. Predloženi algoritam više vrednuje rješenje u kojem se studenti pridružuju u grupe više rangirane na njihovoj prioritetnoj listi.

Time predloženi algoritam je bolji, jer rješenje ima manje nepridruženih studenata s više studenata koji su pridruženi u grupu bolje rangiranu na njihovoj prioritetnoj listi. Predloženi algoritam uključuje i dodatne mjere kvalitete koje nije moguće postići uz druge algoritme.

Za provjeru tih dodatnih kvaliteta napravljena je analiza pridruženih grupa studentima kojima nije moguće pridružiti grupe s njihove prioritetne liste. U postojećim algoritmima, takvi studenti se mogu samo pridružiti nasumičnim poretkom nakon što su svi studenti pridruženi. Kod predloženog algoritma raspodjela nasumičnim odabirom se radi samo za studente kojima se odabir nije mogao pridružiti pomoću prioritetne liste ili proširene prioritetne liste.

Na Slici **71**. prikazani su rezultati ove analize. Odabrani su samo studenti kojima je grupa pridružena temeljem nasumičnog odabira ili temeljem proširene prioritetne liste. Studenti su podijeljeni u skupine do 10% najboljih studenata, od 10% do 25% najbolje rangiranih studenata, te od 25% do 50% najboljih studenata.

Rang popularnosti grupe određen je brojem studenata koji su iskazali interes za navedenom grupom. Grupa koja ima više zainteresiranih studenata, neovisno o rangu na koju su je studenti postavili, smatra se popularnijom od one koju je odabralo manje studenata. Grupe su podijeljene u četiri skupine prema popularnosti: najboljih 25%, 25% do 50%, 50% do 75% te one najmanje popularne od 75% do 100%.



Slika 71 – Pridruživanje u najbolje grupe uz proširene liste i nasumičnim odabirom

Na Slici 71 vidljivo je da su studenti koji su preostali nepridruženi nakon izvršavanja Algoritma odgođenog prihvaćanja pridruženi uglavnom grupama koje su najmanje popularne (75% do 100%). To je posljedica činjenice da se grupe prvo pridružuju studentima kojima ih je moguće pridružiti putem njihove prioritetne liste, a tek nakon toga se grupe koje imaju još slobodnih mjesta pridružuju nepridruženim studentima. Na slici je vidljivo da uporabom Algoritma odgođenog prihvaćanja dobri studenti bivaju pridruženi samo u grupe koje su najmanje popularne.

Kod predloženog algoritma s udaljenosti 0; 0,2 i 0.5, vidljivo je da su studentima pridružene uglavnom bolje rangirane grupe. To je posljedica rada predloženog algoritma u kojoj algoritam pokušava prvo pridružiti grupu sa studentove prioritetne liste, zatim s proširene prioritetne liste. Ako nije uspio s niti jednom navedenom listom, onda grupu pridružuje nasumičnim odabirom. Time se boljim studentima omogućava pridruživanje popularnijoj grupi, a ne samo nekoj od preostalih nakon početnog pridruživanja studenata.

Kada je udaljenost najveća, tada studenti s boljim ocjenama nemaju prioritet u pridruživanju, već algoritam teži pridružiti što većem broju studenata grupu s prioritetne liste.

9.5. Algoritam jednostrukog pridruživanja grupiranih elemenata

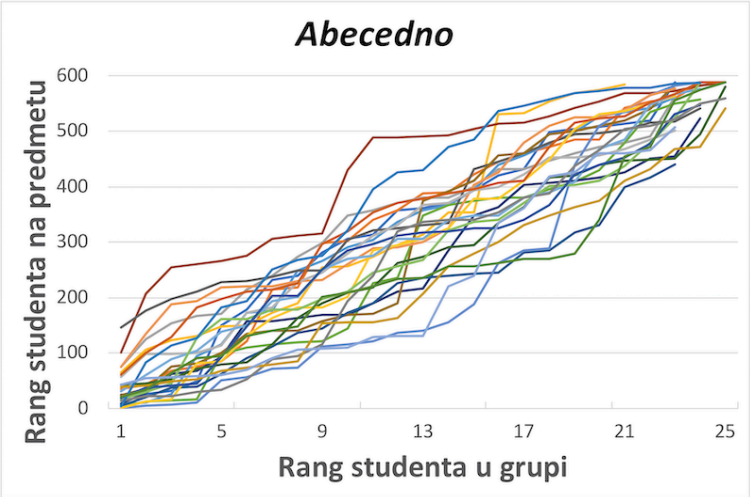
Algoritam jednostrukog pridruživanja koristi metodu odabira redoslijeda elemenata pri rješavanju problema pridruživanja s prioritetnim listama u nastavnim procesima uporabom ekspertnog znanja.

Za ocjenu ovog algoritma provedena je analiza dobivenih balansiranih grupa, kako bi se utvrdilo kako različita ekspertna znanja utječu na balansiranost grupe. U ocjeni su se koristili podaci studenata na predmetu „Seminar“ u akademskoj godini 2018./2019. Studenti su pridruženi u projektne grupe, te se tako grupirani pridružuju u prezentacijske grupe.

Na prikazanim slikama, jedna linija predstavlja jednu prezentacijsku grupu. Na X-osi predstavljeni su studenti poredani prema rangu unutar pridružene prezentacijske grupe. Na Y-osi prikazan je rang studenata na predmetu. U ishodištu grafa najbolje su rangirani studenti.

9.5.1. Pridruživanje temeljem ekspertnog znanja o nazivu projekta

Rezultat abecednog ili nasumičnog grupiranja vidljiv je na Slici 72. Ovaj način pridruživanja koristi se kao usporedni za ostale dobivene uporabom ekspertnog znanja. Usporedbom dvije grupe, pri čemu se uspoređuju po tri studenta iz svake grupe, student koji unutar grupe ima najveću prosječnu ocjenu, medijan student te student s najmanjom prosječnom ocjenom uočljivo je znatno odstupanje u usporedbi prema ukupnom rangu na predmetu. U idealnom slučaju pozicija sva tri studenta u obje grupe bila bi bez značajnih odstupanja. Ista značajna odstupanja vidljiva su i istovjetnom usporedbom drugih studenata u obje grupe. Time ovo pridruživanje daje loše balansirane grupe.



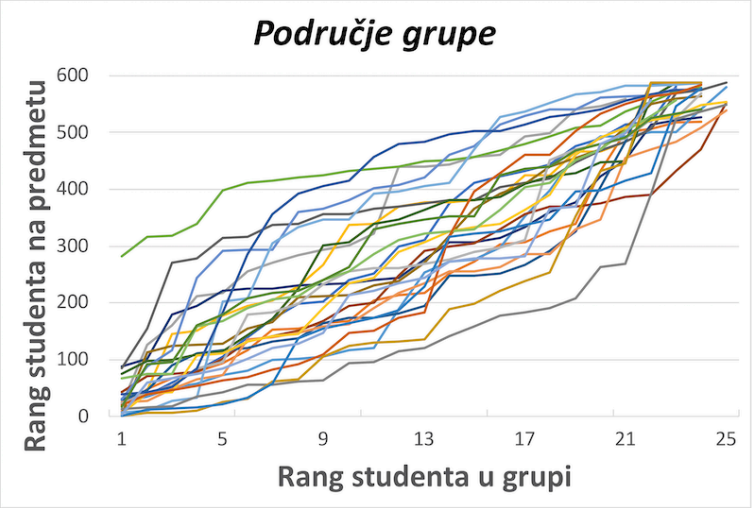
Slika 72 – Grafički prikaz kriterija prema abecedi

9.5.2. Pridruživanje temeljem ekspertnog znanja o području grupe

Za potrebe ovog rada, na predmetu „Seminar“ projekti su ručno pridruženi u ista područja, temeljem poznavanja organizacijskih jedinica voditelja grupa, ali i podskupina pojedinih organizacijskih jedinica.

Na Slici 73, prikazana je raspodjela studenata na predmetu „Seminar“ uporabom ekspertnog znanja o području grupe. Rezultat raspodjele sličan je abecednoj ili nasumičnoj raspodjeli. Ovim pristupom prezentacijske grupe također nisu balansirane. U prikazu se vide i dvije grupe koje sadrže velik broj loše rangiranih studenata odnosno dobro rangiranih studenata. Prva, najviša grupa, prikazuje grupu koja se sastoji od srednje rangiranih studenata do onih lošijih, a nema odličnih studenata. Druga, najniža grupa, prikazuje grupu koja se sastoji od više od 20 studenata koji su natprosječni, među prvih 300 studenata na rangu, a samo nekoliko studenata s lošijim rangom. Ostale se grupe nalaze unutar ova dva ekstrema.

Ovaj način raspodjele bolji je od abecednog ili nasumičnog kada se želi postići sličnost područja grupa. Ovaj način raspodjele se također koristi na diplomskom studiju Fakulteta elektrotehnike i računarstva na predmetu „Diplomski seminar“ gdje se uzima u obzir nerazmjer grupa, a studentima se želi produbiti znanje iz sličnog područja.

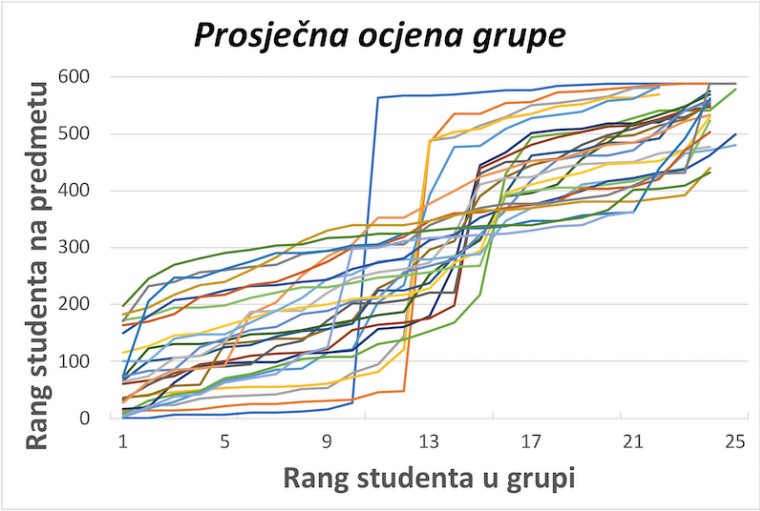


Slika 73 – Grupiranje studenata prema područjima

9.5.3. Pridruživanje temeljem ekspertnog znanja o prosječnoj ocjeni grupe

Na Slici 74 prikazan je rezultat raspodjele temeljem ekspertnog znanja o prosječnoj ocjeni grupe. Ovaj način raspodjele sadrži velik broj grupa koje se sastoje od izvrsnih i loših studenata ali nema dobrih studenata. Takve grupe vidljive su kao linije u „S-obliku“.

Ovaj način pridruživanja kreirao je suprotne rezultate od željenih i sadrži grupe koje su iznimno nebalansirane.

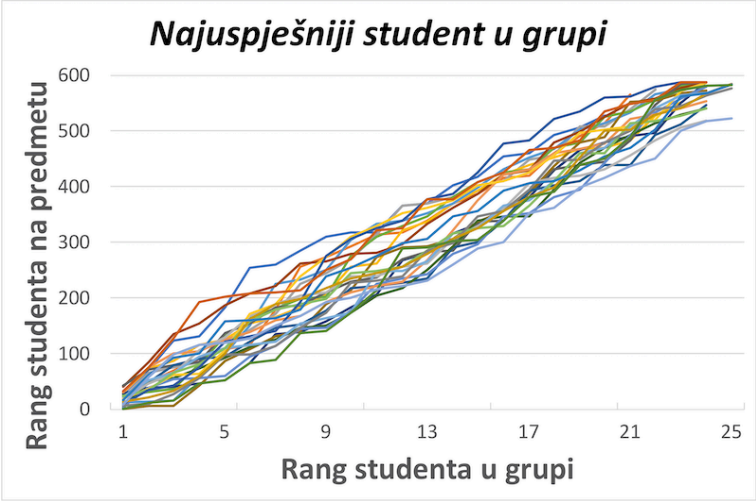


Slika 74 – Grafički prikaz kriterija prema rangu grupe

9.5.4. Pridruživanje temeljem ekspertnog znanja o ocjeni najuspješnijeg studenta u grupi

Za razliku od prethodne metode u kojoj se uzima prosjek uspjeha studenata u projektnoj grupi, u ovoj metodi koristi se ocjena najuspješnijeg studenta u grupi.

Rezultat raspodjele vidljiv je na Slici 75 kao suženje u lijevom donjem dijelu gdje se nalaze izvrsni studenti, i lagano raspršenje u desnom dijelu grafa gdje su manje uspješni studenti. Ovom metodom osiguralo se da su grupe balansirane i da imaju podjednako raspršene najuspješnije studente prema grupama.



Slika 75 – Grafički prikaz kriterija prema najuspješnijim studentima

Ova metoda odabrana je za grupiranje studenata u prezentacijske grupe na predmetu „Seminar“ s obzirom na to da kao rezultat kreira balansirane grupe.

9.6. Model raspodijeljenog rješavanja problema pridruživanja

Uporabom algoritma u raspodijeljenoj programskoj potpori povećava se prostor pretraživanja najboljeg rješenja ili skraćuje vrijeme dolaska do najboljeg rješenja. Verifikacija modela raspodijeljenog rješavanja problema pridruživanja u raspodijeljenoj programskoj potpori koristi Algoritam strogog jednostrukog pridruživanja s glavnim popisom.

Algoritam se je izvršen u 50 mjerenja na jednom računalu s jednim, dva, tri i četiri procesora, te na dva, tri i četiri računala s četiri procesora.

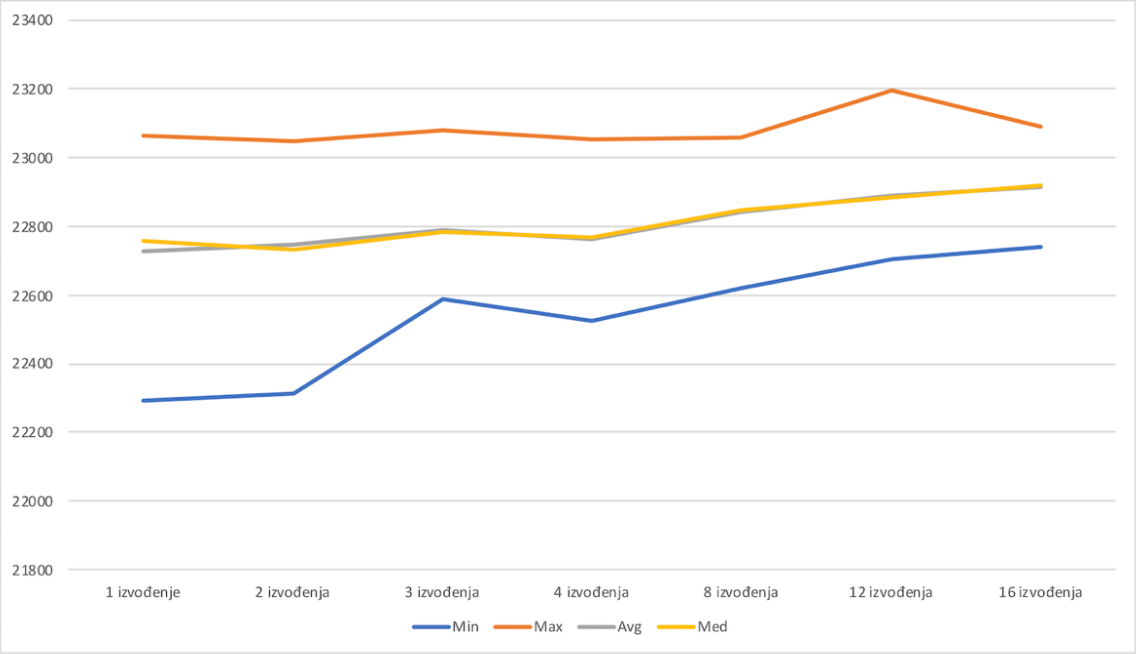
Na Slici 76 prikazana je usporedba dobivenih vrijednosti najboljeg rješenja pri izvođenju u jednakom broju iteracija (100) na svakom od procesora, odnosno računala s vrijednosti udaljenosti od 0,5 na podacima predmeta „Seminar“ u akademskoj godini 2018./2019. Iz rezultata je vidljivo značajno poboljšanje rezultata upotrebom većeg broja računala odnosno procesora.

Mjerenje	Računala: 1				Računala: 2	Računala: 3	Računala: 4
	Threadova: 1	Threadova: 2	Threadova: 3	Threadova: 4	Threadova: 4		
1	22444	22720	22874	22770	22794	22886	22994
2	22680	22702	22634	22794	22780	22956	23046
3	22628	22756	22780	22902	22838	23196	22924
4	22666	22630	22722	22830	22878	22874	22958
5	22604	22836	22764	22766	22904	22872	22970
6	22738	22700	22628	22726	22904	22868	22796
7	22366	22620	22996	22792	22806	22728	22934
8	22336	22690	22772	22920	22964	22886	22834
9	22292	22312	22642	22752	22958	22746	22868
10	22656	22494	22820	22798	22832	22946	23026
11	22784	22716	22886	22684	22884	22938	22908
12	22576	22738	22892	22800	22782	22764	22906
13	22498	22928	22918	22588	22774	22876	22902
14	22734	22822	22886	22696	22992	22900	22964
15	22816	22722	22822	22748	22972	22934	22882
16	22512	22974	22670	22846	22764	22988	22896
17	22682	22694	22726	22844	22782	22714	22992
18	22912	22514	22780	22676	23016	22888	22926
19	22806	22866	23024	22658	22968	22844	22922
20	22786	22820	22814	22808	22778	22902	23018
21	22814	22668	22902	22602	22672	22946	22752
22	22802	22800	22810	22842	22846	22896	23006
23	22834	22856	22884	22702	22720	22992	22802
24	22626	22768	22790	22768	22894	22910	22874
25	22748	22718	22756	22590	22760	22910	22866
26	22732	22744	22676	22714	22856	22944	22856
27	22852	22854	22676	22956	22850	22854	22916
28	22760	22730	22808	22800	22786	22866	22834
29	22776	22772	22834	22738	22752	22836	23090
30	22764	23048	22778	22822	22888	23036	22740
31	22784	23034	22784	22522	22888	22846	22862
32	22800	22782	22722	22638	22748	22904	22942
33	22758	22718	22590	22760	22688	22896	22982
34	22666	22584	22666	22828	22848	22854	22834
35	22786	22854	22628	22852	22806	22846	22814
36	22812	22850	22608	22802	22886	22752	23076
37	22662	22550	23080	22680	22674	22942	23074
38	23010	22680	22896	22884	22780	22978	22982
39	22660	22908	22918	22780	22900	22816	23018
40	22818	22604	22996	22914	22982	23094	22870
41	22914	22916	22686	22904	22934	22990	22998
42	22812	22682	22654	22778	22744	22876	22938
43	22814	22808	22706	22684	22752	23038	22826
44	22790	22732	22792	22636	22958	22874	22992
45	22680	22894	22846	22752	22890	22870	22942
46	22936	22680	22776	22582	23060	22890	22842
47	23066	22670	22756	22714	22880	22872	22796
48	22708	22656	22728	23052	22820	22704	22820
49	22740	22732	22816	22664	22618	22830	22924
50	22920	22810	22834	22800	22848	22824	22794

Slika 76 – Usporedba najbolje pronađenih rješenja u raspodijeljenoj programskoj potpori

Na Slici **77** prikazan je grafički prikaz prosječnih vrijednosti izvođenja algoritma. Također je vidljivo da je standardno odstupanje u rješenju značajno manje s povećanjem raspodijeljenosti samog algoritma, kao i da je medijan gotovo jednak prosječnoj vrijednosti rješenja. Iz njega je također jasno vidljivo da se bolja rješenja dobivaju uporabom algoritma u raspodijeljenoj programskoj potpori.

	1 izvođenje	2 izvođenja	3 izvođenja	4 izvođenja	8 izvođenja	12 izvođenja	16 izvođenja
Min	22292	22312	22590	22522	22618	22704	22740
Max	23066	23048	23080	23052	23060	23196	23090
Avg	22727,2	22747,1	22788,9	22763,2	22842,0	22891,8	22914,6
Med	22759	22732	22782	22769	22847	22886	22919
StdDev	156,25	135,27	112,57	105,51	96,43	91,84	86,44



Slika 77 – Prosječne vrijednosti izvođenja u raspodijeljenoj programskoj potpori

10. ZAKLJUČAK

Problemi pridruživanja u nastavnim procesima mogu riješiti niz problema pridruživanja studenata kao što su: upis studenata, pridruživanje studenta mentoru, pridruživanje studenata u projektne grupe, kreiranje grupa studenata. Drugi autori predlažu svoje pristupe u rješavanju problema pridruživanja, no nitko od njih ne razmatra pristup u kojem se prednost pri pridruživanju daje bolje rangiranim studentima, uzimajući u obzir ekspertno znanje o rangiranju.

U ovoj disertaciji pridruživanja s prioritetnim listama promatrane su kroz pridruživanje studenata u grupe, gdje grupa označava predmet, mentora, projekt, i slično. Student izrađuje svoju prioritetnu listu grupa koja je u pravilu nepotpuna jer nisu rangirane sve grupe. Za razliku od studenata, gdje svaki ima drugačiju listu, grupe koriste jednu – glavni popis. Studentima s većim brojem bodova daje se prednost u pridruživanju. Studenti s jednakim brojem bodova dijele isti rang zbog čega glavni popis nije strogo poredan. Kombinacija nepotpunih prioritetnih lista studenata i davanjem prednosti studentima s višim rangom rezultira nizom nepridruženih studenata. Za kvalitetnije rješavanje ovog problema predložene su dvije nove predložene metode: (i) *metodu odabira redoslijeda elemenata* i (ii) *metoda pridruživanja nepridruženih elemenata.*

Nova predložena *metoda odabira redoslijeda elemenata* koristi ekspertna znanja za kreiranje glavnog popisa. Neizraziti pristup izrade glavnog popisa uvodi parametar *udaljenost* kojim se određuje koliko bodovi studenata mogu biti različite a da bi studenti mogli biti jednako rangirani. S povećanjem parametra sa 0 na 0,1 broj nepridruženih studenata smanjio se za 4,2%. S dosadašnjih 36% na 31,8%. Kada se udaljenost poveća na najveću vrijednost – 3,0 broj nepridruženih studenata smanjio se za 14,4% (sa 36% na 21,6%).

Kod najveće udaljenosti bodovi studenata više ne utječu na pridruživanje. Rezultati istraživanja pokazuju da *udaljenost* 0,5 ne utječe na pridruživanje kod najbolje rangiranih 10% studenata, a istovremeno smanjuje ukupni broj nepridruženih studenata za 10%.

Predložena metoda smanjuje broj nepridruženih studenata, ali ne rješava problem nasumičnog pridruživanja preostalih studenata. Stoga se uvodi nova predložena *metoda pridruživanja nepridruženih* elemenata u nastavnim procesima temeljem zaključaka izvedenih iz konteksta odabira. Ona se koristi u izradi proširene prioritetne liste studenata pridruživanjem više rangiranih studenata više rangiranim grupama dok algoritmi drugih autora pridružuju takve studente u ispod prosječno rangirane grupe. Korištenjem ove metode više od 90% studenata pridruženo je temeljem prioritetne liste ili proširene prioritetne liste, a 10% studenata

pridruženo je nasumičnim redoslijedom. Bez predložene metode 35% studenata ostaje nepridruženo.

U disertaciji je predloženo više algoritama pridruživanja u nastavnim procesima prilagođenih posebnim zahtjevima pojedinog nastavnog procesa. Oni su izvedeni iz *algoritma strogog jednostrukog pridruživanja s glavnim popisom* koji koristi obje nove predložene metode. Analiza pokazuje porast kvaliteta pridruživanja do 25% uporabom navedenih metoda. Kvaliteta pridruživanja određena je temeljem smanjivanja nepridruženih studenata te povećanjem pridruživanja iznad prosječno rangiranih projekata iznad prosječno rangiranim studentima.

Algoritam višestrukog pridruživanja s glavnim popisom uvodi višestruko pridruživanje kojim je studenta moguće istovremeno pridružiti u više grupa. Primjenjuje se u rješavanju problema pridruživanja studenata u grupe kojima je određen terminima održavanja te kod pridruživanja studenata na studentske prakse. Kod ovih pridruživanja važno je studenta pridružiti isključivo grupama koje je odabrao na prioritetnoj listi, zbog čega se ne koristi predložena metoda pridruživanja nepridruženih elemenata.

Algoritam višestrukog pridruživanja s glavnim popisom i višeslojnom prioritetnom listom uvodi višeslojne prioritetne liste koje su važne u rješavanju problema odabira predmeta studenata pri upisu više godine ili semestra. On rješava pridruživanje kad su elementi prioritetne liste nove prioritetne liste. Elementi prioritetne liste na najnižoj razini su grupe u koje se student može pridružiti. Student rangira svaku od prioritetnih lista odvojeno.

Algoritam jednostrukog pridruživanja grupiranih elemenata uvodi mogućnost pridruživanja u balansiranim grupama. Tijekom analize razmatralo se nekoliko načina upotrebe ekspertnog znanja kreiranja glavnog popisa, a najbolje rješenje postignuto je algoritmom koji pridružuje studente prema najuspješnijem studentu u grupi. Ovakav pristup rezultirao je gotovo idealnim pridruživanjem.

U disertaciji je predložen i novi *Model raspodijeljenog rješavanja problema pridruživanja.* Središnji dio modela temelji se na predloženim algoritmima, a koristi agentsku arhitekturu bez ograničenja u broju agenata za rad u raspodijeljenim sustavima. Novi model izveden je i kao prototip, te je prikazana primjena na sedam problema u nastavnim procesima. Prototip je potvrdio pridruživanja novih predloženih metoda i modela na podacima odabira više od 500 studenata i više od 200 grupa na predmetu Seminar tijekom nekoliko godina.

Rezultati novog predloženog modela uspoređeni su s *Algoritmom odgođenog prihvaćanja* (DA) poznatim i kao Gale-Shapley algoritam temeljem: (i) *kvalitete pridruživanja* i (ii) *pozicije grupe na studentovoj prioritetnoj listi*.

Usporedba *kvalitete pridruživanja*, pokazala je usporedive rezultate pridruživanja s udaljenosti 0. S povećanjem udaljenosti na 0.5, broj nepridruženi studenata korištenjem novog predloženog modela smanjuje se za 8%. Usporedbom *pozicije grupe na studentovoj prioritetnoj listi*, predloženi model pokazuje porast u broju pridruživanja više rangiranih studenata više rangiranim projektima u odnosu na DA algoritam.

Korištenjem novih predloženog modela i metoda, porast kvalitete pridruživanja veći je za čak 25% u odnosu na rezultate dobivene upotrebom postojećeg Algoritma odgođenog prihvaćanja. Agentska arhitektura pokazala je također utjecaj pri čemu je u istom zadanom vremenu ostvaren bolji rezultat u odnosu na slučaj kad nije korištena ova arhitektura.

LITERATURA

[1] Y. Chen, M. Jiang, O. Kesten, S. Robin, and M. Zhu, "Matching in the large: An experimental study," *Games Econ. Behav.*, 2018.

[2] C. Calsamiglia, G. Haeringer, and F. Klijn, "A comment on 'School choice: An experimental study' [J. Econ. Theory 127 (1) (2006) 202-231]," *J. Econ. Theory*, 2011.

[3] Y. Chen and T. Sönmez, "School choice: An experimental study," *J. Econ. Theory*, 2006.

[4] J. Pais and Á. Pintér, "School choice and information: An experimental study on matching mechanisms," *Games Econ. Behav.*, 2008.

[5] A. E. Roth, T. Sönmez, and M. Utku Ünver, "Efficient kidney exchange: Coincidence of wants in markets with compatibility-based preferences," *Am. Econ. Rev.*, 2007.

[6] A. E. Roth, T. Sönmez, and M. U. Ünver, "A Kidney Exchange Clearinghouse in New England," *Am. Econ. Rev.*, 2005.

[7] M. Utku Unver, "Dynamic kidney exchange," *Rev. Econ. Stud.*, 2010.

[8] A. E. Roth, "The Evolution of the Labor Market for Medical Interns and Residents: A Case Study in Game Theory," *J. Polit. Econ.*, 1984.

[9] T. Sönmez, "Manipulation via capacities in two-sided matching markets," *J. Econ. Theory*, 1997.

[10] G. Prasadu, "Implementing Task and Resource Allocation Algorithm Based on Non-Cooperative Game Theory in Cloud Computing," *Int. J. Adv. Res. Comput. Sci.*, vol. 9, no. 1, pp. 660–666, 2018.

[11] X. Xu and H. Yu, "A Game Theory Approach to Fair and Efficient Resource Allocation in Cloud Computing," *Math. Probl. Eng.*, 2014.

[12] S. Caton, C. Haas, K. Chard, K. Bubendorfer, and O. F. Rana, "A social compute cloud: Allocating and sharing infrastructure resources via social networks," *IEEE Trans. Serv. Comput.*, 2014.

[13] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *Am. Math. Mon.*, 1962.

[14] L. Shapley and A. Roth, "Stable matching: Theory, evidence, and practical design.," *nobel*, 2012.

[15] D. Kokotsaki, V. Menzies, and A. Wiggins, "Project-based learning: A review of the literature," *Improv. Sch.*, 2016.

[16] D. J. Abraham, R. W. Irving, and D. F. Manlove, "The student-project allocation problem," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 2003.

[17] S. Faudzi, S. Abdul-Rahman, and R. Abd Rahman, "An Assignment Problem and Its Application in Education Domain: A Review and Potential Path," *Advances in Operations Research*. 2018.

[18] S. Olaosebikan and D. Manlove, "Super-stability in the student-project allocation problem with ties," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018.

[19] D. Manlove, D. Milne, and S. Olaosebikan, "An integer programming approach to the student-project allocation problem with preferences over projects," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018.

[20] V. Paunovic, S. Tomic, I. Bosnic, and M. Zagar, "Fuzzy Approach to Student-Project Allocation (SPA) Problem," *IEEE Access*, 2019.

[21] C. Haas, M. Hall, and S. L. Vlasnik, "Finding optimal mentor-mentee matches: A case

study in applied two-sided matching," *Heliyon*, 2018.
[22] D. Ruggiero and J. D. Boehm, "Project-based learning in a virtual internship programme: A study of the interrelated roles between intern, mentor and client," *Comput. Educ.*, 2017.
[23] G. Hein and a. Monte, "A student mentoring and development program for underrepresented groups in engineering," *34th Annu. Front. Educ. 2004. FIE 2004.*, pp. 4–8, 2004.
[24] W. C. D. Hewitt, D. L. Mount, S. K. Beyerlein, D. F. Elger, and J. Steciak, "Using developmental principles to design mentoring experiences for graduate students," *Proc. - Front. Educ. Conf.*, vol. 3, p. S3B/4-S3B/9, 2001.
[25] R. P. Badoni and D. K. Gupta, "A new algorithm based on students groupings for university course timetabling problem," in *2015 2nd International Conference on Recent Advances in Engineering and Computational Sciences, RAECS 2015*, 2016.
[26] R. Agrawal, B. Golshan, and E. Terzi, "Grouping students in educational settings," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014.
[27] R. Hbscher, "Assigning students to groups using general and context-specific criteria," *IEEE Trans. Learn. Technol.*, 2010.
[28] C. Shannon and D. McKinney, "An evolutionary algorithm for assigning students to courses," in *Proceedings of the 24th International Florida Artificial Intelligence Research Society, FLAIRS - 24*, 2011.
[29] V. Paunovic, S. Tomic, I. Bosnic, and M. Zagar, "A system for managing classes with a large number of participants," 2019.
[30] D. J. Abraham, R. W. Irving, and D. F. Manlove, "Two algorithms for the Student-Project Allocation problem," *J. Discret. Algorithms*, 2007.
[31] D. F. Manlove and G. O'Malley, "Student-Project Allocation with preferences over Projects," *J. Discret. Algorithms*, 2008.
[32] K. Iwama, S. Miyazaki, and H. Yanagisawa, "Improved approximation bounds for the Student-Project Allocation problem with preferences over projects," *J. Discret. Algorithms*, 2012.
[33] C. Y. Teo and D. J. Ho, "A systematic approach to the implementation of final year project in an electrical engineering undergraduate course," *IEEE Trans. Educ.*, 1998.
[34] A. A. Anwar and A. S. Bahaj, "Student project allocation using integer programming," *IEEE Trans. Educ.*, vol. 46, no. 3, pp. 359–367, 2003.
[35] R. Calvo-Serrano, G. Guillén-Gosálbez, S. Kohn, and A. Masters, "Mathematical programming approach for optimally allocating students' projects to academics in large cohorts," *Educ. Chem. Eng.*, 2017.
[36] D. F. Manlove and G. O'Malley, "Student-Project Allocation with preferences over Projects," *J. Discret. Algorithms*, 2008.
[37] A. Abdulkadiroğlu, Y. K. Che, and Y. Yasuda, "Expanding 'choice' in school choice," *Am. Econ. J. Microeconomics*, 2015.
[38] M. Goto, A. Iwasaki, Y. Kawasaki, R. Kurata, Y. Yasuda, and M. Yokoo, "Strategyproof matching with regional minimum and maximum quotas," *Artif. Intell.*, 2016.
[39] F. Cooper and D. Manlove, "A 3/2-approximation algorithm for the Student-Project Allocation problem," in *17th International Symposium on Experimental Algorithms (SEA-2018)*, 2018, pp. 8:1–8:13.
[40] Z. Király, "Linear time local approximation algorithm for maximum stable marriage," *Algorithms*, 2013.
[41] A. E. Roth, "The Economics of Matching: Stability and Incentives," *Math. Oper. Res.*,

2008.
[42] N. Perach, J. Polak, and U. G. Rothblum, “A stable matching model with an entrance criterion applied to the assignment of students to dormitories at the technion,” *Int. J. Game Theory*, 2008.
[43] R. W. Irving, D. F. Manlove, and S. Scott, “The stable marriage problem with master preference lists,” *Discret. Appl. Math.*, 2008.
[44] P. Biró, R. W. Irving, and I. Schlotter, “Stable matching with couples: An empirical study,” in *ACM Journal of Experimental Algorithmics*, 2011, vol. 16, pp. 1–29.
[45] R. Burke, “Hybrid web recommender systems,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2007.
[46] C. Lucchese, C. I. Muntean, R. Perego, F. Silvestri, H. Vahabi, and R. Venturini, “Recommender systems,” in *Mining User Generated Content*, 2014.
[47] *Recommender Systems Handbook*. 2011.
[48] C. C. Aggarwal and C. C. Aggarwal, “Knowledge-Based Recommender Systems,” in *Recommender Systems*, 2016.
[49] J. A. Konstan and J. Riedl, “Recommender systems: From algorithms to user experience,” *User Modeling and User-Adapted Interaction*. 2012.
[50] S. Zhang, L. Yao, A. Sun, and Y. Tay, “Deep learning based recommender system: A survey and new perspectives,” *ACM Computing Surveys*. 2019.
[51] S. Gottwald, “Fuzzy sets, fuzzy logic, fuzzy methods with applications,” *Choice Rev. Online*, vol. 34, no. 07, pp. 34-3907-34–3907, 1997.
[52] J. Krejčí, “Fuzzy set theory,” in *Studies in Fuzziness and Soft Computing*, 2018.
[53] L. A. Zadeh, “Fuzzy logic,” in *Computational Complexity: Theory, Techniques, and Applications*, 2013.
[54] P. R. Harper, V. De Senna, I. T. Vieira, and A. K. Shahani, “A genetic algorithm for the project assignment problem,” *Comput. Oper. Res.*, vol. 32, no. 5, pp. 1255–1265, 2005.
[55] W. Yeoh and M. Yokoo, “Distributed problem solving,” in *AI Magazine*, 2012.
[56] M. M. Zavlanos, L. Spesivtsev, and G. J. Pappas, “A distributed auction algorithm for the assignment problem,” in *Proceedings of the IEEE Conference on Decision and Control*, 2008.
[57] M. De Weerdt, Y. Zhang, and T. Klos, “Distributed task allocation in social networks,” in *Proceedings of the International Conference on Autonomous Agents*, 2007.
[58] M. Stillwell, D. Schanzenbach, F. Vivien, and H. Casanova, “Resource allocation algorithms for virtualized service hosting platforms,” *J. Parallel Distrib. Comput.*, 2010.
[59] D. Leong, A. G. Dimakis, and T. Ho, “Distributed storage allocations,” *IEEE Trans. Inf. Theory*, 2012.
[60] A. E. Roth, “Deferred acceptance algorithms: History, theory, practice, and open questions,” *Int. J. Game Theory*, 2008.

POPIS SLIKA

Slika 1 – Dijagram tijeka glavnog algoritma 31
Slika 2 – Dijagram tijeka generiranja izrade glavnog popisa rangiranih studenata 34
Slika 3 – Dijagram tijeka generiranja glavnog popisa rangiranih studenata uz nasljeđivanje . 36
Slika 4 – Dijagram tijeka algoritma generiranja proširene prioritetne liste 38
Slika 5 – Dijagram tijeka algoritma pridruživanja studenata 41
Slika 6 – Algoritma tijeka dijela algoritma pridruživanja studenata – dohvat grupe 42
Slika 7 – Primjer strogog jednostrukog pridruživanja 43
Slika 8 – Dijagram tijeka algoritma ocjene strogog jednostrukog pridruživanja studenata..... 45
Slika 9 – Dijagram tijeka algoritma višestrukog pridruživanja..... 48
Slika 10 – Dijagram tijeka algoritma višestrukog pridruživanja za odabir N grupa 49
Slika 11 – Primjer višestrukog pridruživanja 50
Slika 12 – Dijagram tijeka algoritma ocjene algoritma višestrukog pridruživanja 51
Slika 13 – Dijagram tijeka algoritma s višeslojnom prioritetnom listom 54
Slika 14 – Dijagram tijeka algoritma s višeslojnom prioritetnom listom – odabir rješenja..... 55
Slika 15 – Primjer pridruživanja višeslojnom prioritetnom listom 56
Slika 16 – Grafički prikaz balansirane (idealne) raspodjele studenata 57
Slika 17 – Dijagram tijeka algoritma jednostrukog pridruživanja grupiranih elemenata 59
Slika 18 – Grafički prikaz razmjene podataka u raspodijeljenoj programskoj potpori..... 61
Slika 19 – Dijagram tijeka algoritma u raspodijeljenoj programskoj podršci – glavni..... 63
Slika 20 – Dijagram tijeka algoritma u raspodijeljenoj programskoj podršci – klijent 63
Slika 21 – Dijagram tijeka algoritma u raspodijeljenoj programskoj podršci – poslužitelj 64
Slika 22 – Dijagram baze podataka za razmjenu podataka između poslužitelja i klijenata..... 65
Slika 23 – Primjer ulazne datoteke studenata..... 79
Slika 24 – Primjer ulazne datoteke grupa..... 80
Slika 25 – Primjer ulazne datoteke prioritetnih lista studenata 80
Slika 26 – Izvršavanje algoritma na poslužiteljskom računalu 82
Slika 27 – Izvršavanje algoritma na klijentskom računalu 82
Slika 28 – Višedretvenost – prikaz izvršavanje algoritma 83
Slika 29 – Višedretvenost – sučelje nadzora izvođenja algoritma 83
Slika 30 – Primjer sučelja odabira prioritetne liste projekata 84
Slika 31 – Primjer spremljenih prioritetnih lista 85
Slika 32 – Primjer web-stranice jedne projektne grupe 85

Slika 33 – Izvedba glavnog zaslona odabira mentora na diplomskom studiju 86
Slika 34 – Izvedba sučelja kreiranja prioritetne liste mentora na diplomskom studiju..... 87
Slika 35 – Izvedba sučelja odobravanja mentorstva u radu Odbora 87
Slika 36 – Primjer glavnog zaslona odabira – administratorski pogled..... 88
Slika 37 – Primjer kreiranja prioritetne liste usmjerenja – studentov pogled 88
Slika 38 – Zaslon pridruživanja studenata u grupe 90
Slika 39 – Prikaz slobodnih grupa u koje student može biti pridružen – Student 1..... 90
Slika 40 – Prikaz slobodnih grupa u koje student može biti pridružen – Student 2..... 91
Slika 41 – Prikaz zaslona pridruživanja nakon pridruživanja 91
Slika 42 – Izvedba kreiranja prioritetne liste 92
Slika 43 – Izvedba dodavanja pozicije na prioritetnu listu 93
Slika 44 – Izvedba potvrđivanja studenata na pozicijama 94
Slika 45 – Rezultat pridruživanja studenata na pozicije 94
Slika 46 – Primjer strukture studija..... 95
Slika 47 – Prioritetna lista grupa predmeta 96
Slika 48 – Ograničenje ukupnog broja upisanih predmeta 96
Slika 49 – Prioritetna lista grupe predmeta 97
Slika 50 – Konačni odabir višeslojne prioritetne liste predmeta za upis 98
Slika 51 – Konačna višeslojna prioritetna lista predmeta s prikazom predmeta..... 99
Slika 52 – Korisnička simulacija upisa predmeta na primjeru jednog studenta..... 100
Slika 53 – Zbirna simulacija izračuna broja studenata upisanih na pojedini predmet 101
Slika 54 – Rezultat izvođenja algoritma jednostrukog pridruživanja grupiranih elemenata . 101
Slika 55 – Utjecaj udaljenosti na pridruživanje studenata u ak. god. 2018./2019. 103
Slika 56 – Utjecaj udaljenosti na pridruživanje studenata u ak. god. 2017./2018. 103
Slika 57 – Utjecaj udaljenosti na pridruživanje studenata u ak. god. 2016./2017. 104
Slika 58 – Utjecaj udaljenosti na pridruživanje studenata u ak. god. 2015./2016. 104
Slika 59 – Utjecaj udaljenosti na postotak pridruživanja studenata u ak. god. 2018./2019... 106
Slika 60 – Utjecaj udaljenosti na postotak pridruživanja studenata u ak. god. 2017./2018... 106
Slika 61 – Utjecaj udaljenosti na postotak pridruživanja studenata u ak. god. 2016./2017... 107
Slika 62 – Utjecaj udaljenosti na postotak pridruživanja studenata u ak. god. 2015./2016... 107
Slika 63 – Utjecaj parametra udaljenosti u pridruživanju najboljih 10% studenata 108
Slika 64 – Rezultati analize uz težinske faktore F1, F3 i F5 = { 1; 1.000; 10.000 }..... 110
Slika 65 – Tablica težinskih vrijednosti F1, F3, F5 = { 0,25; 0,28; 0,3; 0,33 } 111

Slika 66 – Raspodjela temeljem prioritetnih i proširenih lista u ak. god. 2018./2019. 112
Slika 67 – Raspodjela temeljem prioritetnih i proširenih lista u ak. god. 2017./2018. 113
Slika 68 – Raspodjela temeljem prioritetnih i proširenih lista u ak. god. 2016./2017. 113
Slika 69 – Raspodjela temeljem prioritetnih i proširenih lista u ak. god. 2015./2016. 114
Slika 70 – Usporedba rezultata s Algoritmom odgođenog prihvaćanja 115
Slika 71 – Pridruživanje u najbolje grupe uz proširene liste i nasumičnim odabirom 117
Slika 72 – Grafički prikaz kriterija prema abecedi 118
Slika 73 – Grupiranje studenata prema područjima 119
Slika 74 – Grafički prikaz kriterija prema rangu grupe 120
Slika 75 – Grafički prikaz kriterija prema najuspješnijim studentima 120
Slika 76 – Usporedba najbolje pronađenih rješenja u raspodijeljenoj programskoj potpori . 122
Slika 77 – Prosječne vrijednosti izvođenja u raspodijeljenoj programskoj potpori 123

POPIS TABLICA

Tablica 1 – Primjer prioritetnih lista pet studenata 20
Tablica 2 – Sličnost prioritetnih lista studenta S1 s ostalim prioritetnim listama 22
Tablica 3 – Redoslijed ocjene za grupe koje student S1 nije odabrao na svojoj prioritetnoj listi u svrhu kreiranja proširene prioritetne liste 23
Tablica 4 – Primjer izračuna vrijednosti utjecaja grupe (PIV) na primjeru studenta S1 24
Tablica 5 – Vrijednosti bodova za potrebe algoritma ocjene pridruživanja studenata 46
Tablica 6 – Izračun proširenih prioritetnih lista za primjer ulaznih datoteka 81
Tablica 7 – Rezultat izvođenja prototipa – pridruživanje studenata grupama 81

POPIS ALGORITAMA

Algoritam 1 – Pseudokôd glavnog genetskog algoritma 30
Algoritam 2 – Pseudokôd generiranja izrade glavnog popisa rangiranih studenata 33
Algoritam 3 – Pseudokôd generiranja glavnog popisa rangiranih studenata uz nasljeđivanje 35
Algoritam 4 – Pseudokôd algoritma generiranja proširene prioritetne liste 37
Algoritam 5 – Pseudokôd algoritma pridruživanja studenata 40
Algoritam 6 – Pseudokôd algoritma ocjene strogog jednostrukog pridruživanja studenata.... 44
Algoritam 7 – Pseudokôd algoritma višestrukog pridruživanja..... 48
Algoritam 8 – Pseudokôd algoritma ocjene algoritma višestrukog pridruživanja 51
Algoritam 9 – Pseudokôd algoritma s višeslojnom prioritetnom listom..... 53
Algoritam 10 – Pseudokôd algoritma jednostrukog pridruživanja grupiranih elemenata 58
Algoritam 11 – Pseudokôd algoritma za izvođenje u raspodijeljenoj programskoj podršci.... 62

POPIS PROGRAMSKOG KÔDA

Programski kôd 1 – Struktura klase *StudentsClass* 68
Programski kôd 2 – Struktura klase *MentorsClass* 69
Programski kôd 3 – Struktura klase *SingleSolutionSimple* 69
Programski kôd 4 – Struktura *StudentIdGrade* 70
Programski kôd 5 – Glavni genetski algoritam 72
Programski kôd 6 – Algoritam izrade proširene prioritetne liste 73
Programski kôd 7 – Algoritam izrade glavnog popisa – početna populacija 75
Programski kôd 8 – Algoritam izrade glavnog popisa – temeljem nasljeđivanja i mutacije ... 76
Programski kôd 9 – Izračun novog rješenja 76
Programski kôd 10 – Algoritam strogog jednostrukog pridruživanja 77
Programski kôd 11 – Algoritam ocjene strogog jednostrukog pridruživanja 78

POJMOVNIK

- Pridruživanje (eng. *allocation*)
- Uparivanje (eng. *matching*)
- Prioriteti (eng. *preference*)
- Neizrazita logika (eng. *fuzzy logic*)
- Nepridružen (eng. *unallocated*)
- Nepotpuna prioritetna lista (eng. *incomplete preference list*)
- Glavni popis rangiranih studenata (eng. *master list of ranked students)*
- Stabilno pridruživanje (eng. *stable allocation*)
- Algoritam odgođenog prihvaćanja (eng. *deferred acceptance algorithm)*
- Teorija uparivanja (eng. *matching theory*)
- Problem pridruživanja (eng. *allocation problem*)
- Dvostrani algoritam uparivanja s prioritetnom listom (eng. *two-sided matching algorithm with a preference list)*
- Pridruživanje studenata projektima (eng. *Student-Project allocation*)
- Proširena prioritetna lista (eng. *extended preference list*)
- Neizraziti pristup (eng. *fuzzy approach*)
- Funkcija dobrote (eng. *fitness*)
- Jednolično križanje (eng. *uniform crossover*)
- Jedinstveni cilj (eng. *single objective*)
- Preporučiteljski sustavi (eng. *recommender systems*)
- Vrijednost utjecaja grupama (PIV) (eng. *project impact value*)
- Pohlepnu metodu (eng. greedy method)
- Algoritam izrade glavnog popisa rangiranih studenata (eng. *Master preference list generation algorithm*)
- Algoritam izrade proširene prioritetne liste (eng. *Extended preference list generation algorithm*)
- Algoritam pridruživanja studenata (eng. *student allocation algorithm*)
- Algoritam ocjene pridruživanja (eng. *allocation evaluation algorithm*)

ŽIVOTOPIS

Vlatka Paunović, dipl. ing. računarstva diplomirala je 2005. na Sveučilištu u Zagrebu Fakultetu elektrotehnike i računarstva s temom „Integracija podataka međusobno povezanih Web sustava“. Njezini profesionalni interesi uključuju otvoreno računarstvo, aplikacije otvorenog kôda i visoko dostupne aplikacije temeljene na web tehnologijama, integraciju sustava te baze podataka.

Od 2005. godine sudjeluje u osmišljavanju i provedbi integracije podataka i aplikacija na Sveučilištu u Zagrebu Fakultetu elektrotehnike i računarstva (FER) za što 2006. godine dobiva zahvalnicu FER-a: „*za predani rad na izgradnji FER-ovog sustava za upravljanje sadržajima „Quilt CMS“ i FER-ovog e-Campusa*“. Sustav se danas koristi na više desetaka fakulteta i sveučilišta u RH, te za potrebe Nacionalnog centra za e-obrazovanje u Estoniji. Od 2007. godine sudjeluje u osmišljavanju i izgradnji raznih sustava za pridruživanje u nastavi kao što su: pridruživanje studenata na nekoliko predmeta preddiplomskog i diplomskog studija, odabir mentora na preddiplomskom i diplomskom studiju, pridruživanje studenata u grupe na predmetne aktivnosti, pridruživanje studenata na pozicije u tvrtkama za potrebe studentskih praksi i slično.

Od 2005. do 2009. sudjeluje u projektu „Uporaba otvorenih programskih alata u nastavi“ pri Hrvatskoj udruzi za otvorene sustave i Internet u suradnji s Agencijom za odgoj i obrazovanje, gdje sudjeluje u organizaciji i prijenosu znanja održavanjem niza radionica besplatnih za sudionike, a namijenjenih učiteljima i nastavnicima informatike i računarstva osnovnih i srednjih škola. Koautor je nekoliko priručnika i knjige iz područja otvorenih sustava.

2009. za svoj rad dobiva priznanje Hrvatskog informatičkog zbora za „*doprinos razvitku ICT-a u Hrvatskoj*“.

Od 2014. sudjeluje kao konzultant i arhitekt idejnog rješenja za razvoj sustava eVisitor – središnjeg informacijskog sustava za prijavu i odjavu turista te praćenje i naplatu turističke pristojbe. Sustav je u potpunosti integriran i razmjenjuje informacije s više državnih tijela. Sustav eVisitor dobio je posebno priznanje Hrvatske gospodarske komore na Forumu obiteljskog smještaja, te 2018. godine na natječaju Svjetske turističke organizacije World Tourism Organisation (UNWTO) osvaja treće mjesto u kategoriji turističkih inovacija i tehnologije.

U 2014. godini sudjeluje u uspostavi vojnih studijskih programa Sveučilišta u Zagrebu „Vojno inženjerstvo“ i „Vojno vođenje i upravljanje“ koje zajednički izvodi 12 sastavnica Sveučilišta u Zagrebu. Za njen doprinos, Hrvatsko vojno učilište koje je krovna obrazovna

institucija Ministarstva obrane RH dodjeljuje joj zahvalnicu „*za osobiti doprinos tijekom pokretanja, razvoja, usvajanja studijskih programa „Vojno inženjerstvo“ i „Vojno vođenje i upravljanje“ na HVU „Petar Zrinski“ i njihove izvedbe u Zagrebačko Sveučilište*“.

Popis objavljenih radova:

Radovi u časopisima

Radovi u CC časopisima

1. **Paunović, Vlatka**; Tomić, Siniša; Bosnić, Ivana; Žagar, Mario
   Fuzzy Approach to Student-Project Allocation (SPA) Problem // IEEE access, 7 (2019), 136046-136061 doi:10.1109/access.2019.2941730 (međunarodna recenzija, članak, znanstveni)

Drugi radovi u časopisima

1. Siniša Tomić; **Vlatka Paunović**; Kristijan Zimmer
   Searching and exposing learning objects from Moodle: The ODS experience // Bulletin of the IEEE Technical Committee on Learning Technology, 17 (2015), 1-2; 18-21 (međunarodna recenzija, članak, ostalo)

Radovi u zbornicima skupova

Znanstveni radovi u zbornicima skupova

1. **Paunović, Vlatka**; Tomić, Siniša; Bosnić, Ivana; Žagar, Mario
   A system for managing classes with a large number of participants // 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)
   IEEE: IEEE, 2019. str. 1439-1444 doi:10.23919/MIPRO.2019.8756955 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)

2. Voras, Ivan; Mihaljević, Branko; Orlić, Marin; Pletikosa, Marko; Žagar, Mario; Pavić, Tomislav; Zimmer, Kristijan; Čavrak, Igor; **Paunović, Vlatka**; Bosnić, Ivana; Tomić, Siniša
   Evaluating Open-Source Cloud Computing Solutions // Proceedings of the 34th International Convention for Information and Communication Technology, Electronics and Microelectronics (MIPRO 2011) - Conference on Grid and Visualization Systems (GVS) / Biljanović, Petar ; Skala, Karolj (ur.).

Zagreb: Croatian Society for Information and Communication Technology, Electronics and Microelectronics – MIPRO, 2011. str. 238-243 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)

3. Tomić, Siniša; Zimmer, Kristijan; Žagar, Mario; **Paunović, Vlatka**; Voras, Ivan; **Living The E-Campus Dream** // EDEN 2006 Annual Conference Proceedings / Szucs, Andras ; Bo, Ingeborg (ur.).
   Vienna: European Distance and E-Learning Network, 2006. str. 644-646 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)

4. Žagar, Mario; Čavrak, Igor; Zimmer, Kristijan; Tomić, Siniša; **Paunović, Vlatka** **Distributed Learning and Collaboration in the Distributed Software Development** // Ambient and Mobile Learning / Auer E, Michael ; Auer, Ursula ; Mittermeir, Roland (ur.).
   Villach, Austria: Carinthia Tech Institute - School of Electronics, 2005. str. 1-13 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)

Autorske knjige

Knjige i poglavlja u knjigama

1. **Paunović, Vlatka**
   OpenOffice.org priručnik: Calc - tablične kalkulacije, Zagreb: Hrvatska udruga za otvorene sustave i Internet - HrOpen, Unska 3, 10000 Zagreb, http://www.open.hr/ za Središnji državni ured za e-Hrvatsku, 2008 (priručnik)

Ostale vrste radova

Bagić, Igor ; Banek, Marko ; Čavrak, Igor ; Gledec, Gordan ; Furdek, Pero ; Javor, Igor ; Jurić, Damir ; Kraljević, Hrvoje ; Krpan, Mate ; Kušek, Mario ; Lovrek, Ignac ; Ljubičić, Hrvoje ; Matijašević, Maja ; Paić-Karega, Marko ; Pandžić, Igor.S. ; **Paunović, Vlatka** ; Pejaković, Ivo ; Pintar, Damir ; Skočir, Zoran ; Sović, Dalibor ; Stipetić, Hrvoje ; Tomić, Siniša ; Vranić, Mihaela ; Vrdoljak, Boris ; Vuković, Marin ; Žagar, Mario
Poslovna inteligencija i elektroničko poslovanje, 2010. (podatak o recenziji nije dostupan, elaborat).

BIOGRAPHY

Vlatka Paunović, dipl. ing. rač graduated in 2005 at the University of Zagreb, Faculty of Electrical Engineering and Computing, with a dissertation on the topic of „Integration of data different Web based systems“. Her professional interests include open computing, open source, applications based on web technologies, system integration and databases.

Since 2005, she has been involved in the design, implementation and integrations of application at the Faculty of Electrical Engineering and Computing, University of Zagreb. For her work in 2006 she received a certificate of appreciation. Today, the system is used by more than ten faculties and universities in Croatia, as well by the national e-learning centre in Estonia. Since 2007 she has been involved in designing and implementation of systems for allocations: assigning students to groups on “Seminar” course, assigning mentors on undergraduate and graduate studies, assigning students in groups for class activities, finding internships in various companies, etc.

Between 2005 and 2009, V. Paunović participated in the project “Using Open Source Tools in Education“ of the Croatian Society for Open Systems and Internet, in cooperation with the Education and Teacher Training Agency, where she participated in the process of sharing information and skills through multiple open workshops intended for teachers of informatics and computer sciences in elementary and high schools. She is a co-author of several guidebooks and books in the field of open systems.

In 2009, she received a certificate of appreciation from Croatian Information Technology Association for her contribution towards development of ICT in Croatia.

Since 2014, she has been a consultant and conceptual solution architect in eVisitor development – the central information system for tourist check-ins and check-outs. The system is fully integrated with multiple government bodies. eVisitor was given a special recognition certificate from Croatian Chamber of Economy at the Family Accommodation Forum in 2018. The same year it is also awarded by the World Tourism Organisation (UNWTO) in tourist innovations and technologies category.

In 2014, V. Paunović participated in establishing a program for military studies in Zagreb «Military Engineering» and «Military Leadership and Management», which are realised by 12 faculties of University in Zagreb. She received a certificate of appreciation by the Croatian Military Academy.