Rješavanje problema raspoređivanja primjenom značajki krajolika dobrote genetskoga programiranja

Čorić, Rebeka

Sveučilište u Zagrebu

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Rebeka Čorić

RJEŠAVANJE PROBLEMA RASPOREĐIVANJA PRIMJENOM ZNAČAJKI KRAJOLIKA DOBROTE GENETSKOGA PROGRAMIRANJA

DOKTORSKI RAD

Zagreb, 2021.

Sveučilište u Zagrebu

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Rebeka Čorić

RJEŠAVANJE PROBLEMA RASPOREĐIVANJA PRIMJENOM ZNAČAJKI KRAJOLIKA DOBROTE GENETSKOGA PROGRAMIRANJA

DOKTORSKI RAD

Mentor: Prof. dr. sc. Domagoj Jakobović

Zagreb, 2021.

University of Zagreb

FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Rebeka Čorić

SOLVING SCHEDULING PROBLEMS WITH GENETIC PROGRAMMING FITNESS LANDSCAPE FEATURES

DOCTORAL THESIS

Supervisor: Professor Domagoj Jakobović, PhD

Zagreb, 2021

Doktorski rad izrađen je na Sveučilištu u Zagrebu Fakultetu elektrotehnike i računarstva, na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave

Mentor: prof. dr. sc. Domagoj Jakobović

Doktorski rad ima: 171 stranicu

Doktorski rad br.: _____

O mentoru

Domagoj Jakobović rođen je 1973. godine u Našicama. Diplomirao je 1996. godine na Fakultetu elektrotehnike i računarstva u Zagrebu, gdje je 2001. godine i magistrirao te 2005. godine doktorirao s temom doktorske disertacije "Raspoređivanje zasnovano na prilagodljivim pravilima".

Od 1997. godine zaposlen je na Zavodu za elektroniku, mikroelektroniku, računalne i inteligentne sustave na Fakultetu elektrotehnike i računarstva u Zagrebu, gdje je u siječnju 2018. izabran u zvanje redovitog profesora. Sudjelovao je na više znanstvenih projekata od kojih je trenutno aktivan "Hyperheuristic Design of Dispatching Rules" (financiran od strane Hrvatske zaklade za znanost) na kojemu je voditelj. Objavio je oko 100 radova u časopisima i zbornicima konferencija u područjima raspoređivanja, optimizacije, paralelnog programiranja, stohastičkih algoritama i strojnog učenja.

Prof. dr. sc. Domagoj Jakobović član je Hrvatske akademije tehničkih znanosti te stručnih udruga IEEE i ACM. Dodatno, član je uredništva znanstvenog časopisa, član je programskog odbora više međunarodnih konferencija, a kao recenzent sudjeluje u radu više od 15 inozemnih časopisa.

About the Supervisor

Domagoj Jakobović was born in 1973 in Našice. He graduated in 1996 at University of Zagreb, Faculty of Electrical Engineering and Computing, where he also obtained M.Sc. in 2001 and Ph.D. in 2005 with doctoral thesis "Scheduling based on adaptive rules".

From April 1997 he is working at the Department of electronics, microelectronics, computer and intelligent systems at Faculty of Electrical Engineering and Computing, where he was promoted to Full Professor in January 2018. He participated in several scientific projects of which the project "Hyperheuristic Design of Dispatching Rules" (funded by the Croatian Science Foundation) is currently active and where he is the lead researcher. He published around 100 papers in journals and conference proceedings in the areas of scheduling, optimization, parallel programming, stohastic algorithms and machine learning.

Prof. Jakobović is a member of Croatian Academy of Engineering, IEEE and ACM. Additionally, he is a member of a journal editorial board, program committee of several international conferences and he serves as a technical reviewer for more than 15 international journals.

Zahvala

Prvo želim zahvaliti svom mentoru prof. dr. sc. Domagoju Jakoboviću na trudu i vremenu uloženom u vođenju izrade ove disertacije i što je uvijek bio spreman pomoći kada treba.

Zahvaljujem i Odjelu za matematiku u Osijeku koji mi je omogućio da radim posao koji volim i upravi koja je uvijek imala razumijevanja i davala podršku za izvršavanje svih obveza koje je doktorski studij zahtijevao. Hvala i mentoru izv. prof. dr. sc. Domagoju Matijeviću koji me usmjerio na upisivanje ovog doktorskog studija i koji je uvijek bio dostupan za razgovor kada je bilo potrebno.

Hvala svim kolegama i kolegicama kako sa Odjela za matematiku tako i sa Fakulteta elektrotehnike i računarstva na svim kavama, razmijenjenim mailovima, porukama i riječima podrške.

Zahvaljujem i svojim prijateljima uz koje sam s vremena na vrijeme mogla zaboraviti na poslovne obveze i koji su mi bili podrška kroz svo ovo vrijeme.

Posebnu zahvalu želim uputiti svojoj kolegici i prijateljici Mateji koja je sve ovo prolazila sa mnom. Puno je lakše bilo ispunjavati sve obveze kada znaš da se imaš kome obratiti. Hvala i što si uvijek korak ispred mene pa te mogu gnjaviti sa pitanjima oko administracije na koja uvijek strpljivo odgovaraš.

Veliko hvala mojim roditeljima, Suzani i Igoru koji su me uvijek usmjeravali da radim ono što volim i što sam uvijek imala njihovu potpunu podršku. Hvala i sestri Roberti koja mi uvijek pokazuje da treba biti svoj i ponekad razmišljati o neozbiljnim stvarima.

Najveće hvala mom mužu Zoranu koji je morao trpiti sve moje promjene raspoloženja kada nešto nije išlo po planu. Hvala ti za svu podršku i razumijevanje. Hvala i sinu Petru koji mi je presložio prioritete u životu i koji mi svaki dan pokazuje da se treba radovati malim stvarima.

Sažetak

Problemi raspoređivanja su NP teški problemi čije se primjene mogu pronaći u raznim područjima. Jedna od hiperheuristika koja se koristi za njihovo rješavanje je genetsko programiranje. Snaga genetskog programiranja leži u prilagodljivosti parametara promatranom problemu kako bi se dobili bolji rezultati. Određivanje vrijednosti parametara je dugotrajno pa se često koristi automatizirano određivanje vrijednosti parametara. Kako bi se dobio uvid u strukturu problema, može se iskoristiti analiza krajolika dobrote koja na temelju relevantnih značajki može razlikovati instance problema i zatim se za svaku grupu instanci mogu odrediti odgovarajući parametri.

Ova disertacija se bavi istraživanjem značajki krajolika dobrote problema raspoređivanja, određivanjem grupa sličnih instanci problema na temelju odabranih značajki te određivanjem parametara za genetsko programiranje u svakoj od dobivenih grupa. Prvi dio disertacije je usmjeren na definiranje operatora koji omogućuju računanje značajki krajolika dobrote jedinki prikazanih stablima. Drugi dio grupira instance problema na temelju izračunatih značajki i uspoređuje rezultate dobivene parametrima prilagođenim za pojedinu grupu i one dobivene ručno određenim parametrima. Treći dio je usmjeren na klasifikaciju instanci i odabir odgovarajućih parametara za pojedinu klasu. Provedena ispitivanja pokazuju da se grupiranjem instanci i korištenjem parametara prilagođenih grupi primjera problema mogu postići bolji rezultati nego korištenjem ručno određenih parametara.

Ključne riječi: problemi raspoređivanja, genetsko programiranje, analiza krajolika dobrote, stabla, grupiranje, automatizirano određivanje parametara, automatizirani odabir značajki

Extended abstract

Solving scheduling problems with genetic programming fitness landscape features

Introduction

Scheduling is a decision-making process in which resources need to be appointed to activities to optimize one or more criteria. Resources can be machines in a workshop, runways in airports, computer processors, etc., while activities can be tasks in the production process, take-offs and landings of airplanes, task execution in computers, etc. Criteria that can be optimized depend on the problem domain and they can be e.g. minimizing the makespan of a schedule, minimizing the number of tasks that exceed the deadline, etc.

Scheduling tasks are interesting because they can describe various real-life problems such as sports events scheduling, university timetabling, nurse rostering, airline scheduling, and many others. The goal of scientific research in this area is to find algorithms that can solve said problems in a reasonable amount of time while obtaining schedules that optimize given criteria. Most of the scheduling problems belong to the class of NP hard problems and that is one of the reasons why the scientific community is still working on finding the best optimization techniques for these problems.

Some scheduling problems can be solved using exact methods (which search the entire solution space and guarantee to find an optimal solution) but only when solution space is relatively small because then one can search the entire solution space in a reasonable amount of time. In most cases, exact methods are impractical for usage so metaheuristics and hyperheuristics are used for solving scheduling problems. Those methods don't guarantee the optimality of the solution, but they can obtain a good enough solution in a reasonable amount of time.

Oftentimes genetic programming (GP) is used as a hyperheuristic for obtaining a solution to scheduling problems. GP is used as a tool for obtaining priority (dispatching) rules which then help in creating a schedule that can be evaluated. GP is an evolutionary technique whose goal is to develop programs that will solve a given problem. It creates programs by using given functions and terminals. In order to get a good solution when using GP, some parameters need to be tuned, such as which mutation and crossover operators to use, what is the mutation probability and the size of the population, etc. Usually, those parameters are determined by hand based on some previous knowledge about the area in which the GP is being used or by searching the parameter space by hand which is a long-lasting procedure. In order to speed up and simplify the process of parameter tuning, many methods for automatic parameter tuning are being developed.

One of many tools that can help in automatic parameter configuration is the fitness landscape analysis (FLA). FLA can help to obtain a better understanding of a given problem and

facilitate parameter selection and tuning for hyperheuristics such as GP. Fitness landscape was first mentioned as an idea in 1932 in the paper published by Sewall Wright. Fitness landscape refers to mapping the genome of a population to their respective fitness values and visualizing that mapping. In order to describe the fitness landscape, many measures are being introduced. Some of them are modality, ruggedness, walks, neutrality, etc. Fitness landscape analysis has many applications. Some of them are using fitness landscape analysis of combinatorial optimization problems to obtain a better understanding of problem structure, using fitness landscape analysis to predict algorithm efficiency for a given problem, using fitness landscape analysis to choose a good algorithm or an operator for solving a given problem, etc. Some researchers use fitness landscape analysis to determine which parameters to use while solving a given problem using a metaheuristic or a hyperheuristic.

Additionally, when solving a set of problems, that set can be heterogeneous so it makes sense to cluster instances of those problems in similar clusters and solve problems cluster by cluster by using parameters adjusted for problem instance characteristics in the corresponding cluster. Scheduling problems usually contain many instances and those instances differ concerning various configuration parameters, so it is safe to assume that those instances can be clustered into similar clusters. Then, for every cluster, good parameters for GP can be determined to obtain good solutions. The experiments in this thesis try to find good clusters for scheduling problems and suitable GP parameters to obtain good solutions for a given problem.

Motivation

Genetic programming is used for solving various scheduling problems such as single machine scheduling, unrelated machines scheduling, resource constrained project scheduling problem, job shop scheduling, etc. But, there is no determinate way to choose good parameters for GP to obtain good solutions for given problems. Scheduling problems have various applications and that is the main motivation for the improvement of GP for solving these problems. The automatic parameter configuration enables the usage of GP for solving scheduling problems even to the people who are not experts in that area because in that case there is no need to have prior knowledge of which parameter values yield better results. Additionally, automated parameter configuration speeds up the application of GP to the desired problem because there is no need to carry out the long-lasting process of determining suitable parameters by hand.

Fitness landscape analysis can give insight into the structure of a given problem, so it is needed to determine relevant features of the fitness landscape in scheduling problems. Many papers research fitness landscapes of metaheuristics, but for hyperheuristics, this area is relatively unexplored. In papers that do explore hyperheuristic fitness landscapes, individuals are represented by permutation representation and binary representation while in GP for scheduling problems individuals are often represented as trees. One of the main questions that remained unanswered in this area is how to determine fitness landscape features for tree representation.

The main problem arises in determining a random walk for trees because there is no determinate way to define a tree that has some arbitrary distance from a given tree. There exists some measures which can determine the distance between two trees, but they don't take into account that GP can have two types of nodes - function and terminal nodes, and that function nodes can have different arities. That is why it is needed to define a measure that will take those different types of nodes into account and also adjust existing fitness landscape measures so they can be calculated for tree individuals.

Furthermore, since the instances of the scheduling problem differ in properties, it is necessary to study whether the given instances can also be clustered based on the fitness landscape features. Many authors have explored which fitness landscape features can distinguish instances of the problem. Some of the open questions are which fitness landscape features to use in scheduling problems, can the same features give the same information in different scheduling problems and which clustering algorithms to use when clustering scheduling problems instances into clusters.

The automated parameters selection and configuration simplifies the usage of the algorithms used for solving given problems to end-users so the last step is to determine GP parameters that are best suited for every cluster obtained in the previous step. Those parameters can be determined by using existing frameworks or by deriving new procedures for solving the stated problem. That way the end-users can easily determine in which cluster some problem instance belongs and use predetermined parameters for that cluster.

Thesis overview

This thesis is divided into 8 chapters. Chapter 1 gives an introduction to the thesis. Chapters 2, 3, and 4 give an overview and definitions of fitness landscape analysis, scheduling problems that are used in the thesis, and genetic programming. Chapters 5, 6, and 7 deal with the main contributions of the thesis. Finally, chapter 8 gives the conclusion of this work.

Chapter 1 gives an introduction to the thesis. It describes the problems that are taken into account, how they are solved and states the importance of fitness landscape analysis. Additionally, it gives the motivation for conducted research and describes the main contributions of the thesis. An overview of the thesis sections was given at the end.

Chapter 2 describes fitness landscape analysis. It explains how one can define fitness landscape, gives an overview of fitness landscape measures and fitness landscape analysis applications.

Chapter 3 defines scheduling problems and describes their building blocks. Scheduling in a single machine environment, the unrelated machines environment and the resource constrained project scheduling problem are described in more detail and some papers which have those three scheduling problems in focus are listed.

Chapter 4 gives building blocks of genetic programming and explains how the algorithm

works. Additionally, it explains which individual representation is used as well as how to initialize the individuals and which stopping criteria can be used in GP.

In chapter 5, the insert, edit and delete operators are introduced and described which enable us to move through syntactic tree individuals. Additionally, the heuristic based on the introduced operators is defined which calculates the distance between any two trees. At the end of the chapter, an example of heuristic usage to determine the distance between two arbitrary trees is given.

The beginning of chapter 6 gives an overview of the literature in which GP is used for solving scheduling problems, the literature which connects GP and fitness landscape analysis, and finally the literature which uses clustering and automated parameter selection and configuration. After the literature overview, it is described how the problem instances are defined and how were the experiments conducted. Fitness landscape features that were used are described and it is explained how the parameters for obtained cluster were determined. The results obtained for the single machine scheduling environment, the unrelated machines environment, and the resource constrained scheduling problem are given in corresponding subsections. At the end of the section, a procedure of choosing fitness landscape features for clustering is explained and a brief review of the execution time of all the procedures in this chapter is given.

Chapter 7 describes the procedure of GP parameter selection for classes obtained in the classification process where classification is based on fitness landscape features of the resource constrained project scheduling problem. The grouping association scheme is described which creates learning set for the classification of instances and it is explained how to compare obtained parameters for given classes. In the end, the results are given.

Finally, chapter 8 contains the main contributions obtained while making this thesis and it lists possible ways forward in future research.

Conclusion

The main goal of this thesis is to use fitness landscape features of GP to cluster the instances of scheduling problems into similar clusters and use automatic parameter configuration to obtain suitable GP parameters for every cluster. The following original scientific contributions were made in the thesis:

- Operator set for syntactic trees in order to obtain random walk in search space.
- Grouping of scheduling problems based on fitness landscape characteristics of search space of genetic programming.
- Using fitness landscape analysis in order to determine suitable parameters for genetic programming.
- Learning model which selects or construct fitness landscape characteristics in order to determine optimal parameters for solving scheduling problems by using genetic programming.

This dissertation used automatic parameter selection for solving scheduling problems with GP. First, the fitness landscape analysis was used to study the differences between instances of a given scheduling problem and to cluster similar instances into the same clusters. After that, GP parameters were determined automatically for every obtained cluster and the results obtained with these parameters were compared to the results obtained by parameters determined manually.

The first problem was calculating fitness landscape features for a problem where individuals are represented as trees. To solve this problem, insert, edit, and delete operators were introduced as well as a heuristic that can calculate the distance between two arbitrary trees which enabled the calculation of features. Those operators can be expanded to other sets of functions and terminals.

Clustering based on fitness landscape features showed that for different scheduling problems different clustering algorithms have to be used to get better clusters. For single machine and unrelated machines environment, the k-means algorithm yields better clusters while in the resource constrained project scheduling problem EM algorithm yields better clusters. Additionally, in different scheduling problems, the usage of different fitness landscape features yields better clusters. Automatic parameter configuration yielded different GP parameters for different clusters in every clustering of conducted experiments which means that clustering based on fitness landscape features can differentiate instances of a given problem. Comparison of the results obtained by GP with automatically determined parameters for corresponding cluster and the results obtained manually for the entire set of instances show that better results are obtained when using parameters adjusted for the cluster to which an instance belongs. Better improvements could be obtained by using feature selection to choose fitness landscape features that can better differentiate instances of a given problem. Additionally, the described procedure reduces the time needed to set up the GP to solve a given problem.

The last section of the thesis showed that instances of the problem can be classified into predetermined classes based on fitness landscape features. Although this procedure doesn't offer improvement in the terms of results, it significantly reduces the time needed to set up GP to solve scheduling problems.

Experiments conducted in this thesis show that fitness landscape analysis can improve GP usage for solving scheduling problems and they opened many further interesting topics that can be explored.

Keywords: scheduling problems, genetic programming, fitness landscape analysis, trees, clustering, automatic parameter configuration, automatic feature selection

Sadržaj

Poglavlje 1

Uvod

Raspoređivanje je proces donošenja odluka u kojemu je potrebno alocirati sredstva za zadatke koji se trebaju izvršiti, pri čemu je cilj optimizirati jedan ili više željenih kriterija [1]. Sredstva koja se upotrebljavaju mogu se odnositi na strojeve u radionici, piste na aerodromima, procesore u računalu, dok zadaci mogu biti radnje u procesu proizvodnje, polijetanje i slijetanje zrakoplova, izvršavanje računalnih programa, itd. Kriteriji koji se trebaju optimizirati ovise o domeni u kojoj se problem nalazi pa mogu biti npr. minimizacija vremena završetka posljednjeg zadatka, minimizacija broja zadataka koji završavaju nakon svog roka za izvršavanje, itd.

Problemi raspoređivanja zanimljivi su jer se uz pomoć njih mogu opisati mnogi problemi iz stvarnog svijeta, poput raspoređivanja sportskih događaja [2], izrade rasporeda predavanja na sveučilištima [3], raspoređivanje smjena medicinskih sestara [4], izrade rasporeda slijetanja u zrakoplovnim lukama [5], itd. pa je cilj znanstvenih istraživanja pronaći što bolje algoritme za rješavanje ovih problema. Jedan od razloga za neprestano razvijanje novih načina rješavanja je taj što većina problema raspoređivanja pripada skupini NP teških problema, odnosno ne postoje algoritmi koji mogu u polinomijalnom vremenu pronaći njihovo optimalno rješenje.

Problemi raspoređivanja se mogu rješavati egzaktnim metodama (koje pretražuju cijeli prostor stanja i time garantiraju pronalazak optimalnog rješenja), ali samo kada se radi o vrlo malim instancama problema u kojima je moguće u razumnoj količini vremena pretražiti cijeli prostor stanja [6] [7] [8] [9]. U većini slučajeva je korištenje egzaktnih metoda nepraktično pa se za rješavanje problema raspoređivanja koriste metaheuristike [10] [11] i hiperheuristike [12] [13] koje nemaju jamstvo optimalnosti, ali uspijevaju u razumnoj količini vremena doći do dovoljno dobrog rješenja promatranog problema.

Često se za rješavanje problema raspoređivanja kao hiperheuristika koristi i genetsko (genetičko) programiranje (GP). GP se često koristi kao alat za razvijanje prioritetnih pravila koja zatim omogućuju stvaranje rasporeda koji se može evaluirati [14] [15] [16]. GP je tehnika iz skupine evolucijskih algoritama čiji je cilj razviti programe koji će rješavati dani problem [17]. GP na temelju danih funkcija i značajki stvara programe koji se mogu primijeniti na promatrani

problem. Pri tome GP ima određene parametre, poput izbora operatora križanja i mutacije, vjerojatnosti mutacije, veličine populacije, i drugih, koji utječu na rad samog algoritma. Odabirom dobrih parametara, GP može pronaći jako dobra rješenja. Obično se parametri odabiru ručno na temelju prethodnog znanja o području u kojem se algoritam primjenjuje ili ručnom pretragom prostora parametara koja je dugotrajan proces. Iz tog razloga razvijaju se metode za automatizirano određivanje parametara za GP kako bi se sam proces pripreme GP-a za rješavanje danog problema ubrzao i pojednostavio [18] [19] [20].

Jedan od alata koji može pomoći u razumijevanju strukture promatranog problema, a samim time i olakšati odabir dobrih parametara za hiperheuristike poput GP-a, je analiza krajolika dobrote. Krajolik dobrote se kao ideja javlja 1932. u radu Sewalla Wrighta [21] i odnosi se na mapiranje genoma populacije jedinki na njihove dobrote te vizualizaciju tog mapiranja [22]. Kako bi se krajolik mogao opisati, uvode se razne mjere krajolika poput modaliteta, hrapavosti, šetnji, neutralnosti i mnogih drugih [23]. Krajolik dobrote ima mnoge primjene; neke od njih su analiza krajolika dobrote kombinatoričkih problema u svrhu boljeg razumijevanja strukture problema, analiza krajolika dobrote u svrhu određivanja težine problema, analiza krajolika dobrote u svrhu predviđanja učinovitosti algoritma, analiza krajolika dobrote u svrhu određivanja dobrog algoritma ili operatora za rješavanje danog problema, itd. Analiza krajolika dobrote se može koristiti i kao alat za pomoć prilikom određivanja parametara za metaheuristike i hiperheuristike pa tako autori u [24] koriste analizu krajolika dobrote za određivanje parametara za robusnu tabu pretragu, dok autori u [25] koriste analizu krajolika dobrote kako bi odabrali operator križanja za GP.

Dodatno, kada se rješava skup problema, može se dogoditi da oni imaju različita svojstva pa bi imalo smisla grupirati primjerke problema u grupe međusobno sličnih primjeraka te ih tako odvojeno rješavati prilagođenim algoritmima. Upravo tom idejom su se vodili autori u [26] koji su na temelju značajki krajolika dobrote rasporedili instance NK-problema u slične grupe i zatim određivali parametre za memetički algoritam za svaku od dobivenih grupa. S obzirom da problemi raspoređivanja sadrže mnogo instanci koje se razlikuju po raznim konfiguracijskim parametrima, može se pretpostaviti da se i u tim problemima instance mogu rasporediti u grupe sličnih instanci i da se onda u tim grupama može primijeniti GP kako bi se dobili što bolji završni rasporedi. Upravo tim problemima se bave eksperimenti u ovom radu.

1.1 Istraživačka motivacija

GP se koristi za rješavanje raznih problema raspoređivanja poput raspoređivanja na jednom stroju [27] [28], raspoređivanja na nesrodnim strojevima [29] [15], raspoređivanja s ograničenim sredstvima [30] [16], problema proizvoljne obrade [31], itd. No, ne postoji jednoznačan način kako odabrati dobre parametre za GP kako bi dobiveni rezultati u konačnici bili što bolji.

Raširenost primjena problema raspoređivanja, kao i korištenje GP-a za rješavanje tih problema, glavna su motivacija za unaprjeđenje korištenja GP-a u tom području. Automatizirani odabir parametara omogućuje korištenje GP-a i korisnicima koji nisu stručnjaci u danom području jer nema potrebe za predznanjem o tome koji parametri rezultiraju dobrim vrijednostima u promatranim problemima. Isto tako, automatizirani odabir parametara ubrzava primjenu algoritma na željenom problemu jer nema potrebe prije samog pokretanja vršiti dugotrajan proces ručnog određivanja parametara.

S obzirom da analiza krajolika dobrote daje uvid u strukturu promatranih problema, potrebno je odrediti relevantne značajke krajolika dobrote u problemima raspoređivanja. Za metaheuristike postoji mnoštvo radova u kojima se istražuju krajolici dobrote, no za hiperheuristike je to područje relativno neistraženo. Ochoa i suradnici su u [32] i [33] definirali krajolik dobrote hiperheuristike, ali se u tim radovima za prikaz jedinki koristi permutacijski prikaz i binarni prikaz pa je za njih već definirano mnoštvo značajki krajolika dobrote koje se mogu uporabiti. U ovom radu se za prikaz jedinki u GP-u koriste stabla te je jedno od otvorenih pitanja kako odrediti značajke krajolika dobrote u slučaju kada su jednike prikazane stablima. Najveći izazov pri tome predstavljaju slučajne šetnje jer nije jednoznačno određeno kako dobiti susjeda na proizvoljnoj udaljenosti od trenutno promatrane jedinke. U prijašnjim istraživanjima su definirane mjere kojima se određuje udaljenost između stabala [34] [35] [36], ali one ne uzimaju u obzir da u GP-u čvorovi u stablima mogu biti funkcijski čvorovi ili mogu sadržavati značajke pa treba definirati mjeru koja će i te informacije uzeti u obzir. Uz definiranje metrike za udaljenost, potrebno je i prilagoditi definirane mjere tako da se mogu izračunati za stabla.

Nadalje, s obzirom da se instance problema raspoređivanja razlikuju po svojstvima, potrebno je proučiti mogu li se dane instance grupirati i na temelju značajki krajolika dobrote. Mnogi autori u svojim su radovima istraživali koje značajke krajolika dobrote mogu razlikovati instance problema. Autori u [37] istražuju svojstva instanci koje dolaze iz 6 različitih kombinatoričkih problema i donose zaključke koliko su instance različite i teške za rješavanje pojedinim algoritmom. U radu [38] istražuje se po kojim se svojstvima instance problema biparticije grafa (engl. *graph bipartitioning problem*) razlikuju i koji od ponuđenih algoritama bolje djeluje na kojoj instanci. Kao što je već spomenuto, autori u [26] su došli do zaključka kako bi imalo smisla grupirati instance problema u grupe temeljene na značajkama krajolika dobrote pa je otvoreno pitanje koje značajke krajolika dobrote upotrijebiti u problemima raspoređivanja, daju li jednake značajke u različitim problemima raspoređivanja jednake informacije i koje algoritme za grupiranje upotrijebiti.

Automatizirani odabir parametara za algoritme koji se upotrebljavaju prilikom rješavanja željenih problema znatno olakšava uporabu takvih algoritama krajnjim korisnicima pa preostaje za svaku od dobivenih grupa iz prethodnog koraka, korištenjem već razvijenih okruženja ili stvaranjem novih postupaka, odrediti koji parametri za GP su najbolji za koju grupu. Na taj

način će krajnji korisnici na temelju značajki instance ili podvrste problema koju žele riješiti, moći odrediti kojoj grupi instanca pripada i upotrijebiti unaprijed određene parametre.

1.2 Glavni doprinosi

Osnovna ideja ovog rada je upotrijebiti značajke krajolika dobrote GP-a kako bi se instance problema raspoređivanja grupirale u slične grupe, a zatim automatiziranim odabirom parametara odrediti odgovarajuće parametre za svaku od dobivenih grupa.

Istraživanje se provodilo u tri dijela. Prvi dio istraživanja fokusirao se na definiranje operatora nad sintaksnim stablima koji omogućuju kretanje kroz prostor pretraživanja i računanje značajki krajolika dobrote. Definirani su operatori umetanja, zamjene i brisanja čvorova koji uzimaju u obzir skup funkcija i značajki koji se koriste u GP-u. Ovi operatori omogućuju dobivanje jedinki metodom slučajne šetnje koje se tada mogu evaluirati i temeljem kojih se mogu izračunati i druge značajke krajolika dobrote. Dodatno, s obzirom da ne postoji mjera koja bi za ovakva stabla dala informaciju o tome kako definirati jedinku na proizvoljnoj udaljenosti od trenutno promatrane jedinke, definirana je heuristika temeljena na uvedenim operatorima koja omogućuje prikupljanje informacija o susjedstvu u sintaksnom prostoru genotipa.

Drugi dio istraživanja bavio se značajkama krajolika dobrote za probleme raspoređivanja, poput koeficijenta autokorelacije, mjera neutralnosti krajolika, informacijskih mjera, itd. Prvi korak bio je pronalaženje skupa značajki krajolika dobrote koje se mogu relativno brzo i jednostavno izračunati. Zatim su se pokušale pronaći one značajke koje daju informaciju o različitosti instanci promatranog problema, odnosno one na temelju kojih se instance promatranog problema mogu grupirati. Za grupiranje su se koristili algoritmi k - srednjih vrijednosti i algoritam maksimizacije očekivanja. Grupiranje instanci problema pokušalo se učiniti na temelju svake od izračunatih značajki posebno, a zatim i temeljem kombinacije više značajki krajolika dobrote. Cilj ovog dijela istraživanja bio je pronaći odgovarajuće značajke krajolika dobrote koje se mogu brzo izračunati, a koje daju dobre informacije koje omogućuju grupiranje danih instanci problema raspoređivanja.

Posljednji dio istraživanja fokusirao se na pronalaženje optimalnih parametara GP-a za rješavanje promatranih problema raspoređivanja u dobivenim grupama. Za svaku od dobivenih grupa su se upotrebom okruženja *irace* odredili parametri za GP koji su se zatim usporedili s parametrima određenima ručno za cijeli skup problema. Dodatno, definirana je heuristika koja promatrane probleme svrstava u klase i zatim metodama nadziranog učenja klasificira dotad neviđene instance problema u odgovarajuće klase te na njima primijenjuje unaprijed definirane parametre za GP u odgovarajućoj klasi. Na ovaj način dobio se alat kojim se može brzo izračunati koje su značajke krajolika dobrote neke nove instance problema, na temelju značajki odrediti u koju grupu ta instanca pripada te primijeniti dobivene parametre za odgovarajuću

grupu na promatranu instancu. Proces optimizacije se time ubrzava jer nema potrebe za svaku instancu problema posebno određivati parametre za GP.

Na temelju do sada navedenih dijelova istraživanja, doprinos ove disertacije se sastoji od sljedećih točaka:

1. Skup operatora za sintaktička stabla u svrhu prolaska kroz prostor pretraživanja slučajnom šetnjom.
2. Grupiranje problema raspoređivanja temeljeno na značajkama krajolika dobrote prostora rješenja genetskog programiranja.
3. Određivanje prikladnih parametara genetskog programiranja temeljem krajolika dobrote.
4. Model učenja koji odabirom ili konstrukcijom značajki krajolika dobrote određuje optimalne parametre za rješavanje problema genetskim programiranjem.

1.3 Organizacija rada

Ova disertacija sastoji se od 8 poglavlja. Poglavlje 1 daje uvod u disertaciju. Poglavlja 2, 3 i 4 daju pregled i definicije analize krajolika dobrote, promatranih problema raspoređivanja, i GP-a. Poglavlja 5, 6 i 7 sadrže doprinose ove disertacije. Na poslijetku, poglavlje 8 zaključuje ovaj rad.

Poglavlje 1 sadrži uvod u disertaciju. Opisani su promatrani problemi, načini njihovog rješavanja i važnost upotrebe analize krajolika dobrote. Uz to dana je motivacija za provedena istraživanja te su opisani glavni doprinosi. Na kraju poglavlja dan je pregled dijelova disertacije.

Poglavlje 2 daje opis krajolika dobrote. Objašnjeno je kako se krajolik definira, navedene su neke od mjera te su nabrojane primjene krajolika dobrote.

U poglavlju 3 je dana definicija i opis problema raspoređivanja općenito. Nabrojane su i opisane glavne komponente problema raspoređivanja. Dodatno, problem raspoređivanja u okolini jednog stroja, problem raspoređivanja u okolini nesrodnih strojeva i problem raspoređivanja s ograničenim sredstvima koji se koriste u eksperimentima su pobliže opisani i dani su neki od radova kojima su ti problemi u fokusu.

Poglavlje 4 prikazuje algoritam GP-a, nabraja osnovne komponente algoritma, te objašnjava koji se prikaz rješenja koristi, kako inicijalizirati jedinke i koji se kriteriji zaustavljanja mogu koristiti prilikom izvršavanja GP-a.

U poglavlju 5 dane su definicije i primjeri operatora umetanja, zamjene i brisanja koji omogućuju kretanje kroz prostor sintaksnih stabala. Dodatno, definirana je i heuristika temeljena na uvedenim operatorima koja omogućuje računanje udaljenosti između dva proizvoljna stabla. Na kraju je dan primjer računanja udaljenosti između dva proizvoljna stabla korištenjem uvedene heuristike.

Poglavlje 6 na početku daje pregled literature u kojoj se koristi GP za rješavanje problema

raspoređivanja, zatim literature koja povezuje GP i analizu krajolika dobrote, te literature koja koristi grupiranje i automatizirani odabir parametara. Zatim je dan opis oblikovanja ispitnih primjera, odnosno na koji način su provedeni eksperimenti. Opisane su značajke krajolika dobrote koje su se računale te je objašnjeno na koji način se određuju parametri u određenim grupama. Nakon toga su u pojedinim odjeljcima predstavljeni rezultati dobiveni za okolinu jednog stroja, okolinu nesrodnih strojeva te dvije verzije problema raspoređivanja s ograničenim sredstvima. Na kraju poglavlja prikazan je postupak odabira značajki krajolika dobrote na temelju kojih se obavlja grupiranje te je dan kratak osvrt na vrijeme izvršavanja pojedinih dijelova eksperimenata.

U poglavlju 7 je opisan postupak odabira parametara za GP za pojedinu klasu koja se dobije temeljem klasifikacije problema raspoređivanja s ograničenim sredstvima. Dan je postupak pridruživanja grupiranjem koji od instanci problema grupiranja stvara skup za učenje na temelju kojeg se mogu učiti razni klasifikatori. Opisan je postupak usporedbe dobivenih parametara te su dani rezultati.

Konačno, poglavlje 8 sadrži glavne zaključke dobivene izradom ove disertacije te daje moguća otvorena pitanja koja nisu bila u dometu izrade disertacije.

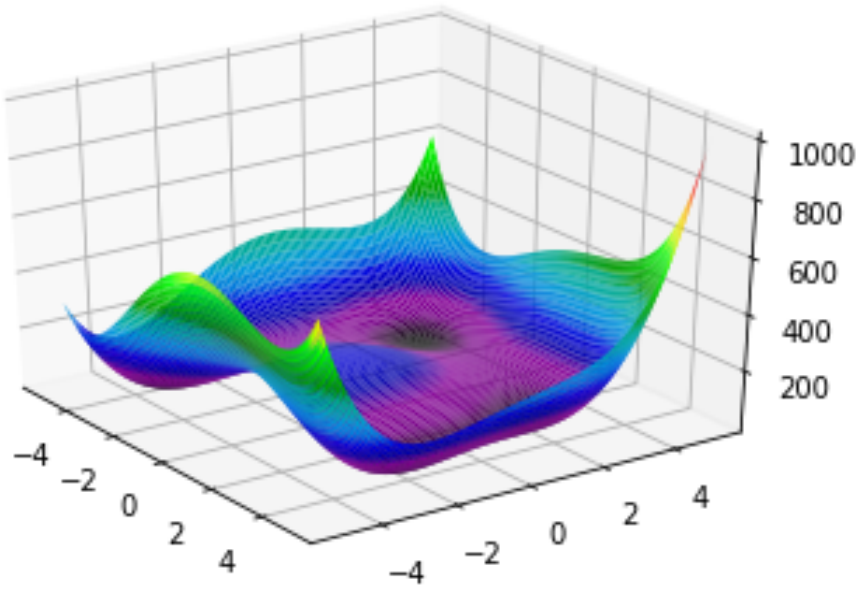
Poglavlje 2

Analiza krajolika dobrote

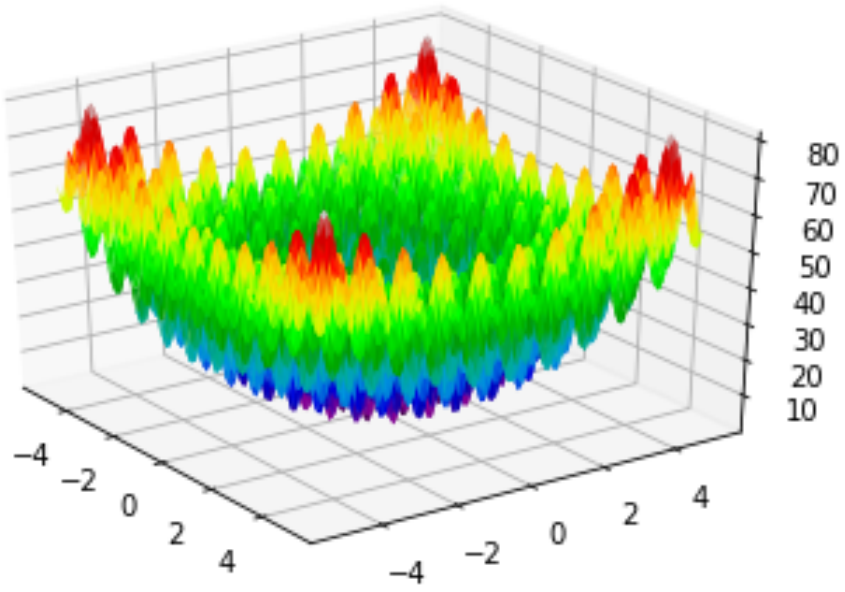
Krajolik dobrote (engl. *fitness landscape*) se kao ideja prvi puta spominje u radu Sewalla Wrigta iz 1932. [21]. Iako se u tom radu zapravo nigdje ne spominje izraz "krajolik dobrote", Wright se smatra začetnikom te ideje. Koncept krajolika dobrote se odnosi na preslikavanje genoma populacije jedinki na njihove dobrote te vizualizaciju tog preslikavanja [22]. Ukoliko se promatra dvodimenzionalan problem, vrlo je jednostavno prikazati krajolik dobrote. Jedinke se prikazuju u ravnini (na *x*-*y* osima), dok se njihova dobrota prikazuje u trećoj dimenziji (na *z* osi) i na taj se način dobija prikaz krajolika u kojemu se mogu uočiti vrhovi (engl. *peaks*) i doline (engl. *valleys*). Iz toga prikaza se može uočiti gdje se nalaze lokalni optimumi za promatrani problem, gdje se nalazi globalni optimum, je li sam krajolik hrapav ili gladak, itd. Na slici 2.1a se može vidjeti Himmelblau funkcija [39] koja ima gladak krajolik dobrote, ali ima četiri lokalna minimuma, dok se na slici 2.1b može vidjeti Rastriginova funkcija [40] koja ima hrapav krajolik dobrote, ali ima samo jedan globalni minimum.

Ukoliko imamo više dimenzija, ovakav je prikaz nemoguć pa je potrebno formalno definirati krajolik dobrote kako bi se mogle mjeriti značajke krajolika i izvlačiti potrebni zaključci. Krajolik dobrote se može definirati kao uređena trojka $F = (S, f, d)$ gdje je S skup svih kandidata za rješenje danog problema, f funkcija dobrote te d funkcija udaljenosti između elemenata iz skupa S [41]. Iz ove definicije se vidi da krajolik dobrote ne ovisi samo o funkciji i vrijednostima dobrote, nego i o međusobnom odnosu između kandidata za rješenja iz skupa S. Ukoliko gledamo krajolik dobrote prilikom rješavanja nekog problema evolucijskim algoritmima, njegov izgled ovisi o izboru prikaza rješenja i metoda koje se koriste za prelazak iz jednog rješenja u drugo, odnosno metoda za pretraživanje susjedstva nekog rješenja.

Pojam krajolik dobrote potječe iz područja evolucijske biologije gdje se promatralo kako genotip utječe na fenotip i u kakvoj su oni povezanosti sa uspjehom (dobrotom) reprodukcije neke jedinke [42]. Iz tog područja se pojam krajolika dobrote proširio i u polje računarstva u svrhu opisivanja značajki krajolika koji nastaje primjenom određenog algoritma na neki optimizacijski problem.



(a) Himmelblau funkcija, $f(x,y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$



(b) Rastriginova funkcija u 2 dimenzije, $f(x,y) = 20 + x^2 - 10\cos(2\pi x) + y^2 - 10\cos(2\pi y)$, $x \in [-5.12, 5.12], y \in [-5.12, 5.12]$

Slika 2.1: Prikaz krajolika dobrote za dvije funkcije

Prvi primjer krajolika dobrote u računalnom smislu je predložio Kauffman i nazvao ga NK model, odnosno NK krajolik (engl. *NK fitness landscape*) [43] [44]. U tom modelu hrapavost krajolika se može kontrolirati korištenjem parametara N i K, gdje je N duljina jedinke, a K označava broj jedinki koje utječu na ponašanje svake jedinke. Ukoliko se K postavi na vrijednost jednaku nuli, dobiva se gladak (engl. *smooth*) krajolik, a ukoliko se postavi na vrijednost jednaku $N-1$, odnosno da na vrijednost dobrote svake jedinke u nekoj mjeri utječu i sve ostale jedinke u populaciji, dobiva se jako hrapav (engl. *rugged*) krajolik. Nakon uvođenja ovog modela, pojavili su se razni radovi koji su analizirali različita svojstva takvog krajolika (npr. [45] [46]) i na taj način dobili uvid u dinamiku procesa evolucije. Pokazalo se da krajolik dobrote dosta utječe na ponašanje algoritama i na put kojim neki algoritam dolazi do krajnjeg rješenja. Ta saznanja su postala temelj za proučavanje krajolika i stvaranje raznih mjera koje mogu opisati krajolik dobrote. Osim proučavanja NK krajolika, počeli su se proučavati i krajolici dobrote genetskih algoritama, evolucijskog programiranja i evolucijskih strategija [47] [48] [49] [50].

U nastavku će biti opisane neke od značajki koje se koriste za opisivanje samog krajolika i bit će navedene neke primjene krajolika dobrote.

2.1 Značajke (mjere) krajolika dobrote

Analiza krajolika dobrote (engl. *fitness landscape analysis*) je postupak istraživanja značajki (ili mjera) krajolika dobrote kako bi se dobio uvid u strukturu problema i kako bi se te informacije iskoristile za bolje razumijevanje samog problema ili za bolje oblikovanje postupkaka rješavanja problema. Prilikom analize krajolika dobrote istražuju se razne značajke koje daju određene informacije o problemu i kao takve se neprestano razvijaju. Ovdje će biti opisane neke od značajki krajolika dobrote koje se koriste u istraživanjima.

Modalitet (engl. *modality*). Ovaj pojam dolazi iz statistike gdje se distribucija s jednom vrijednošću koja ima najveću vjerojatnost ostvarivanja naziva unimodalna, dok se distribucija s više vrijednosti koje imaju jednaku najveću vrijednost ostvarivanja naziva multimodalna distribucija [23]. U svijetu krajolika dobrote ovaj pojam najčešće označava koliko promatrani krajolik ima lokalnih optimuma i taj broj se označava sa n_O. Također se može gledati i frekvencija lokalnih optimuma u odnosu na veličinu prostora rješenja pa se i taj podatak naziva modalitetom. Važne informacije o krajoliku dobrote se mogu iščitati i iz međusobne udaljenosti između lokalnih optimuma ili iz njihove distribucije.

Bazeni privlačnosti (engl. *basins of attraction*). Bazeni privlačnosti su područja oko lokalnih optimuma koja vode direktno prema promatranom lokalnom optimumu kada se spuštamo prema rješenju (ako se radi o minimizacijskom problemu). No, s obzirom da može postojati više mogućnosti za spust prema rješenju, bazeni se dijele na bazene bezuvjetne ili jake privlačnosti i na bazene uvjetne ili slabe privlačnosti [51]. Bazeni jake privlačnosti su oni u kojima

bilo koji oblik spusta vodi do promatranog optimuma, a bazeni slabe privlačnosti oni u kojima lokalni optimum u kojem će se završiti ovisi o tome koji se algoritam ili operator koristi prilikom spusta. Ako promatramo put prema dolje (engl. *downward path*) $p_{\downarrow}$ između neka dva kandidata za rješenja x_0 i x_n definiran kao povezan niz kandidata za rješenja $\{x_i\}_{i=0}^{n}$ koji zadovoljavaju $(\forall i < j)x_i \geq x_j, x_0 > x_n, x_{i+1} \in N(x_i)$, pri čemu je sa $N(x_i)$ označeno susjedstvo od x_i, onda se i formalno mogu definirati bazeni privlačnosti na sljedeći način: bazen slabe privlačnosti $b(o) := \{x | x \in S, p_{\downarrow}(x,o)\}$, bazen jake privlačnosti: $\hat{b}(o) := \{x | x \in b(o), (\nexists o' \neq o \in \mathscr{O})x \in b(o')\}$. Oznaka $\mathscr{O}$ predstavlja skup svih lokalnih optimuma.

Šetnje (engl. *landscape walks*). Šetnje su jedan od bitnih alata za analiziranje krajolika dobrote. Šetnja označava pomicanje za neku malu udaljenost od jednog kandidata za rješenje do drugog kandidata za rješenje u blizini i promatranje što se događa s dobrotama kandidata prilikom tog pomicanja. Ukoliko se promatra diskretan prostor pretraživanja, tada postoji konačan broj kandidata za novo rješenje na nekoj odabranoj udaljenosti ε od trenutnog kandidata za rješenje. Ukoliko se promatra kontinuirani prostor pretraživanja, treba eksplicitno odabrati duljinu koraka ε ili vjerojatnosnu distribuciju susjednih kandidata za rješenje. U oba slučaja može postojati više kandidata za rješenje koji su za ε udaljeni od trenutno promatranog kandidata za rješenje pa o ovisnosti o kriteriju odabira sljedećeg kandidata razlikujemo nekoliko vrsta šetnji [23]. Slučajne šetnje (engl. *random walk*) su one u kojima se sljedeći kandidat bira na slučajan način od svih koji su na udaljenosti ε od trenutnog. Kod adaptivnih šetnji (engl. *adaptive walk*) se bira kandidat iz skupa svih kandidata na udaljenosti ε od trenutnog, koji ima bolju vrijednost funkcije dobrote nego trenutno promatrani kandidat za rješenje. S druge strane, kod obrnuto adaptivnih šetnji (engl. *reverse adaptive walk*) se bira kandidat iz skupa svih kandidata na udaljenosti ε od trenutnog, koji ima lošiju vrijednost funkcije dobrote nego trenutno promatrani kandidat za rješenje. Šetnja uzbrdo - nizbrdo (engl. *"uphill - downhill" walk*) prvo primjenjuje adaptivnu šetnju sve dok se više ne može postići poboljšanje, a zatim obrnuto adaptivnu šetnju sve dok se više ne može pronaći kandidat sa lošijom vrijednošću funkcije dobrote. Neutralne šetnje (engl. *neutral walk*) odabiru rješenja koja imaju jednaku vrijednost funkcije dobrote kao trenutno promatrani kandidat (npr. u svrhu povećanja udaljenosti od početne točke). Kada se koriste šetnje, najbolje ih je kombinirati jer različite šetnje istražuju različita područja krajolika. Slučajne šetnje istražuju šire područje dok npr. adaptivne šetnje istražuju "zanimljivije" područje krajolika dobrote.

Hrapavost (engl. *ruggedness*). Hrapavost se može opisati kao učestalost mijenjanja nagiba iz "uzbrdo" u "nizbrdo" i jedan je od prvih pojmova koji se koristio za mjerenje težine optimizacijskog problema [22] [52]. No, postavlja se pitanje: kako mjeriti hrapavost krajolika? Weinberger je u [45] definirao autokorelaciju i duljinu autokorelacije kao mjere za hrapavost. Autokorelacija ρ se definira kao prosječna korelacija vrijednosti dobrote na udaljenosti ε, tj. $\rho(\varepsilon) = \frac{E(f_t \cdot f_{t+\varepsilon}) - E(f_t)E(f_{t+\varepsilon})}{Var(f_t)}$. Duljina korelacije se definira kao prosječna udaljenost između je-

dinki na kojoj one postaju "nekorelirane". Duljina korelacije τ se najčešće računa kao $1/\rho(1)$. Što je vrijednost τ manja, to je krajolik hrapaviji.

Korelacija udaljenosti i dobrote (engl. *fitness distance correlation - FDC*). Ova mjera je dizajnirana kako bi se dobio globalan pogled na krajolik dobrote. Ona stavlja u odnos udaljenost između rješenja i razlike u vrijednostima dobrote tih rješenja. U maksimizacijskim problemima bi bilo idealno da kako se povećava vrijednost dobrote, da se tako smanjuje i udaljenost trenutno promatranog rješenja od optimalnog rješenja. Često se ova mjera koristi kako bi se predvidjelo koliko je promatrani problem težak za rješavanje korištenjem genetskog algoritma. U prvoj verziji, za računanje FDC-a bilo je potrebno poznavati globalni optimum, ali se ubrzo zaključilo da je to nerealno za očekivati pa se umjesto globalnog optimuma cijelog prostora, za računanje FDC-a koristi globalni optimum nekog reprezentativnog uzorka prostora rješenja. FDC se računa kao: $\text{FDC} = \frac{\frac{1}{n}\sum_{i=1}^{n}(f_i-\bar{f})(d_i-\bar{d})}{\sigma_F \sigma_D}$, gdje je s x_O označen globalni optimum, s $\{f_i\}_{i=1}^n = \{f(x_i)\}_{i=1}^n$ vrijednosti dobrota za odabranih n rješenja iz prostora rješenja te s $\{d_i\}_{i=1}^n = \{d(x_i,x_O)\}_{i=1}^n$ udaljenosti odabranih rješenja do globalnog optimuma. Oznake $\bar{f}$ i $\bar{d}$ predstavljaju srednje vrijednosti nizova $\{f_i\}_{i=1}^n$, odnosno $\{d_i\}_{i=1}^n$. Autori u [23] navode da je zbog jednostavnosti i dobre diskriminatorne moći u mnogim stvarnim problemima, ovo jedna od najkorištenijih mjera u prošlosti.

Neutralnost (engl. *neutrality*). Neutralnost se može shvatiti kao stupanj do kojeg određeni krajolik sadrži povezana područja jednake vrijednosti dobrote. Postoji više mjera za mjerenje neutralnosti, a neke od njih su [23]:

- volumen ili veličina, tj. broj susjednih kandidata za rješenje koji imaju jednaku funkciju dobrote kao trenutni kandidat ili funkciju dobrote s jako malom razlikom u odnosu na trenutnog kandidata,
- broj različitih vrijednosti dobrote kandidata za rješenja koji okružuju neutralno područje,
- maksimalna udaljenost između dva člana neutralnog područja,
- prosječna udaljenost između dva člana neutralnog područja,
- stopa neutralnih susjeda koja se računa kao $\frac{|NN|}{l}$ gdje je sa $|NN|$ označen broj neutralnih susjeda u slučajnoj šetnji duljine l.

Prema [23], neutralnost igra važnu ulogu u radu i performansama heurističkih algoritama [53] [54] i ne može se opisati nekim drugim mjerama [55].

Mjere probiranja (engl. *probing measures*). Ove mjere se računaju na temelju slučajnog odabira vrijednosti dobrote i daju općenit pregled krajolika dobrote [56]. Prilikom računanja promatra se l odabranih jedinki i njihove vrijednosti funkcije dobrote. Kako bi se dobile mjere neosjetljive na skaliranje, od dobivenih vrijednosti funkcija dobrote prave se tri različita skupa. Prvi skup se označava s M_X i sadrži sve vrijednosti dobrote dobivene evaluacijom odabranih jedinki. Drugi skup je označen s M_{iqr} i u njega pripadaju sve vrijednosti funkcije dobrote koje se nalaze između donjeg i gornjeg kvartila vrijednosti iz skupa M_X. Treći skup je označen s M_{LU} i

u njega pripadaju sve vrijednosti koje se nalaze između donje granice L i gornje granice U. Donja granica je definirana kao $L = Q_1 + \frac{1}{2}(Q_M - Q_1)$, a gornja kao $U = Q_M + \frac{1}{2}(Q_3 - Q_M)$, gdje je s Q_M označen medijan, a s Q_1 i Q_3 donji i gornji kvartil vrijednosti iz skupa M_X. Nakon definiranja ova tri skupa za koje vrijedi $M_X \supset M_{iqr} \supset M_{LU}$, određuje se koliko elemenata se nalazi u pojedinom skupu. Na temelju dobivenih vrijednosti računaju se tri mjere, odnosno značajke krajolika dobrote: minimum slučajnog probiranja ($\mu_{RP.Min}$), maksimum slučajnog probiranja ($\mu_{RP.Max}$) i raspon slučajnog probiranja ($\mu_{RP.Range}$). Minimum slučajnog probiranja računa se na temelju linearnog modela koji opisuje vezu između broja vrijednosti u svakom od navedena tri skupa i minimalne vrijednosti u odgovarajućem skupu. Nagib dobivenog pravca se dijeli sa interkvartilom skupa M_X i ta dobivena vrijednost je $\mu_{RP.Min}$. Na sličan način dobiju se i vrijednosti maksimuma i raspona slučajnog probiranja, samo se umjesto minimalnih vrijednosti u odgovarajućim skupovima promatraju maksimalne vrijednosti, odnosno razlike između maksimalnih i minimalnih vrijednosti u odgovarajućim skupovima.

Informacijske mjere (engl. *information measures*). Ove mjere su inspirirane time da sadržaj informacije o skupu jedinki može poslužiti kao mjera težine opisivanja toga skupa [57]. Definiraju se tri mjere: sadržaj informacije (engl. *information content*), djelomični sadržaj informacije (engl. *partial information content*) i stabilnost informacije (engl. *information stability*). Za računanje sve tri mjere potrebno je načiniti slučajnu šetnju neke duljine l. Sadržaj informacije karakterizira distribuciju lokalnih optimuma u šetnji dok djelomični sadržaj informacije daje broj lokalnih optimuma u šetnji. Stabilnost informacije je najveća postignuta razlika u vrijednostima funkcije dobrote za dvije susjedne jedinke u šetnji. Ukoliko sa $\{f_t\}_{t=0}^{l}$ označimo niz vrijednosti, koje dolaze iz intervala I, dobivenih evaluacijom jedinki u slučajnoj šetnji duljine l, možemo definirati znakovni niz $S(\varepsilon) = s_1 s_2 s_3 \cdots s_l$ simbola $s_i \in \{\bar{1}, 0, 1\}$ koji se računaju kao

$$s_i = \begin{cases} \bar{1} & \text{ako je } f_i - f_{i-1} < -\varepsilon \\ 0 & \text{ako je } |f_i - f_{i-1}| \leq \varepsilon \\ 1 & \text{ako je } f_i - f_{i-1} > \varepsilon \end{cases}$$

za neki fiksan broj ε. Vrijednost za ε se uzima iz intervala $[0, d_I]$ gdje d_I označava duljinu intervala I. Vrijednost parametra ε određuje točnost prilikom računanja znakovnog niza $S(\varepsilon)$. Ukoliko je on jednak nuli, $S(\varepsilon)$ će biti određen jako precizno jer će vrijednosti s_i biti jako osjetljive na vrijednosti dobrote, a ukoliko je ε jednak d_I, $S(\varepsilon)$ će biti niz nula. Sadržaj informacije se računa kao $H(\varepsilon) = -\sum_{p \neq q} P_{[pq]} \log_6 P_{[pq]}$ gdje su pq blokovi duljine 2 iz znakovnog niza $S(\varepsilon)$, a P označava vjerojatnost koja se računa kao $P_{[pq]} = \frac{n_{[pq]}}{l}$ gdje je $n_{[pq]}$ broj pojavljivanja bloka pq u znakovnom nizu $S(\varepsilon)$. U izrazu za računanje sadržaja informacije se uzima logaritam sa bazom 6 jer je to broj različitih blokova koji se mogu sastaviti od simbola iz skupa $\bar{1}, 0, 1$. Za računanje djelomičnog sadržaja informacije definira se novi znakovni niz $S'(\varepsilon)$ koji se dobije

na sljedeći način: $S'(\varepsilon)$ je prazan ukoliko se $S(\varepsilon)$ sastoji samo od nula. U suprotnom se definira kao $S'(\varepsilon) = s_{i_1}s_{i_2}\cdots s_{i_\mu}$, gdje je $s_{i_j} \neq 0$ i $s_{i_j} \neq s_{i_{j-1}}$ za $j > 1$. U znakovnom nizu $S'(\varepsilon)$ se nalaze samo elementi oblika $\bar{1}$ ili 1. Duljina μ niza $S'(\varepsilon)$, daje modalnost krajolika dobivenog šetnjom. Duljina μ skalirana na interval $[0,1]$ se označava sa $M(\varepsilon) = \frac{\mu}{n}$ i naziva djelomični sadržaj informacije. Stabilnost informacije je najmanja vrijednost ε^* za koju $S(\varepsilon^*)$ postaje niz nula. Odnosno, stabilnost informacije je najmanja vrijednost za koju će prema ovim mjerama promatrani krajolik biti gladak.

Mjere temeljene na udaljenosti (engl. *lengthscale measures*). Mjera temeljena na udaljenosti mjeri promjenu u funkciji cilja uzimajući u obzir i udaljenost između parova kandidata za rješenja [58]. Ako se sa x_i i x_j označe dva različita kandidata za rješenje, pri čemu su $f(x_i)$ i $f(x_j)$ njihove odgovarajuće vrijednosti dobrote, mjera temeljena na skali udaljenosti se računa kao: $r = \frac{|f(x_i)-f(x_j)|}{d(x_i,x_j)}$, gdje je $d(x_i,x_j)$ udaljenost između ta dva kandidata za rješenje. Ostaje pitanje kako učiniti uzorkovanje populacije da bi se izračunala ova mjera. Autori u [59] predlažu da se uzmu u obzir svi parovi kandidata za rješenje. Ukoliko početni uzorak ima m jedinki, treba se načiniti svih $\binom{m}{2}$ kombinacija parova iz kojih se dobije $\frac{m(m-1)}{2}$ mjera temeljenih na udaljenosti.

U ovom radu su se za opisivanje krajolika dobrote računale slučajne šetnje duljine 1000 jedinki, zatim prosječna vrijednost dobrote slučajne šetnje, koeficijent autokorelacije, stopa neutralnih susjeda u slučajnoj šetnji, mjere probiranja, informacijske mjere i mjere temeljene na udaljenosti.

2.2 Primjene analize krajolika dobrote

Često se javlja pitanje zašto uopće analizirati krajolik dobrote ako se to vrijeme i resursi mogu iskoristiti za rješavanje samog problema koji se promatra? Ukoliko se treba riješiti samo jedna instanca nekog danog problema, onda bi možda imalo smisla resurse uložiti u rješavanje baš te instance. No, ukoliko se rješava problem koji pripada nekoj klasi problema, ima smisla učiniti analizu krajolika dobrote kako bi se bolje razumjelo kakva svojstva taj problem ima i kakvi bi postupci rješavanja mogli iskoristiti ta svojstva i na drugim problemima sličnima onom promatranom [23]. Razni autori se u svojim radovima bave opisivanjem krajolika dobrote za neke poznate i zanimljive probleme kako bi dobili uvid u strukturu i težinu problema za neke algoritme. U nastavku će biti opisane neke od primjena analize krajolika dobrote.

Analiza krajolika dobrote kombinatoričkih problema. U radovima koji se bave istraživanjem krajolika dobrote, može se pronaći mnoštvo onih koji proučavaju strukturu kombinatoričkih problema. Ideja tih istraživanja je dobiti uvid u strukturu krajolika dobrote pojedinog problema kako bi ta saznanja pomogla u stvaranju novih ili poboljšanju već postojećih algoritama i heuristika za rješavanje danih problema. Istražuju se i opisuju krajolici dobrote problema

maksimalne zadovoljivosti (engl. *maximum satisfiability*) [60] [61], problema kvadratnog dodjeljivanja (engl. *quadratic assignment problem*) [62] [63], problema trgovačkog putnika (engl. *traveling salesman problem*) [64] [65], problema bojanja grafa (engl. *graph-coloring problem*) [66] [67], problema ruksaka (engl. *knapsack problem*) [68] [69], problema usmjeravanja vozila (engl. *vehicle routing problem*) [70] [71], problema raspoređivanja uz obradu tijeka (engl. *flowshop scheduling*) [72] [73], problema raspoređivanja s ograničenim sredstvima (engl. *resource constrained project scheduling problem*) [74], itd. Neki od radova, poput [75], uspoređuju krajolike različitih problema kako bi vidjeli postoji li poveznica između njih kako bi se stvorila heuristika koja bi dobro djelovala na sve odabrane probleme.

Analiza krajolika dobrote u svrhu određivanja težine problema. U mnogim slučajevima se analiza krajolika dobrote koristi kako bi se odredile značajke krajolika dobrote koje utječu na težinu rješavanja nekog problema. Autori su u [22] došli do zaključka da usporedba raspona dubina bazena privlačnosti u krajoliku dobrote na kraju adaptivnih šetnji ima dobru korelaciju s težinom problema na velikom skupu istraženih problema. Isto tako, hrapavost je jedna od prvih mjera koja se koristila za određivanje težine problema pa se u nekim radovima veća hrapavost povezuje s većom težinom promatranog problema optimizacije. U dosta radova se istražuje poveznica upravo između FDC-a i težine promatranog problema [76] [77] [78] [79].

Analiza krajolika dobrote u svrhu predviđanja učinkovitosti algoritma. S obzirom da heuristike ne garantiraju optimalnost dobivenog rješenja, jako je teško predvidjeti kako će se one ponašati prilikom rješavanja nekog danog problema. Zbog toga bi bilo zanimljivo prije samog pokretanja algoritma imati ideju kako bi se promatrani algoritam mogao ponašati na nekom danom problemu. Zbog toga se u radovima istražuje kako neke značajke krajolika dobrote utječu na rad pojedinih algoritama na promatranom problemu. Npr. autori u [52] i [80] promatraju koje značajke najviše utječu na rad algoritma roja čestica (engl. *particle swarm optimization*) na problemima kontinuirane optimizacije, dok se u [48] [81] istražuje koje značajke najviše utječu na rad genetskog algoritma. Dolazi se do zaključka da nijedna značajka sama nije dovoljna za predviđanje kako će algoritam raditi na nekom problemu, ali da bi kombinacija raznih značajki mogla dovesti do puno boljih zaključaka.

Analiza krajolika dobrote u svrhu određivanja dobrog algoritma ili operatora za rješavanje problema. Kada se promatra neki problem, treba odlučiti koji algoritam odabrati za njegovo rješavanje kako bi se postigli što bolji rezultati. Neki autori u tu svrhu koriste analizu krajolika dobrote, tj. analiziraju značajke koje daju indikaciju koji od ponuđenih algoritama odabrati za rješavanje. Npr. autori u [82], koristeći razne značajke krajolika dobrote poput autokorelacije, duljine korelacije, informacija o dubini bazena, odabiru koji od ponuđena dva algoritma će bolje riješiti zadanu instancu problema kvadratnog dodjeljivanja. Isto tako, ako se koriste algoritmi u kojima postoje neki posebni operatori, mogu se istražiti značajke krajolika dobrote kako bi se odredilo koji operatori daju bolja krajnja rješenja. Autori u [25] koriste ana-

lizu krajolika dobrote kako bi odredili koji operator križanja se treba odabrati u radu genetskog algoritma kako bi se postigli bolji rezultati. Nadalje, prilikom korištenja metaheuristika, često za sam rad algoritama treba odrediti koji će se parametri za rad algoritma koristiti prilikom rješavanja. Autori u [24] koriste analizu krajolika dobrote kako bi odredili koji parametri se trebaju koristiti za robusnu tabu pretragu (engl. *robust taboo search*) kako bi se što bolje riješio problem kvadratnog dodjeljivanja.

Poglavlje 3

Problemi raspoređivanja

Kako bi se mogle odrediti značajke krajolika dobrote, potrebno je definirati probleme na kojima će se značajke određivati. U ovom radu, naglasak je stavljen na probleme raspoređivanja. Ovo poglavlje će dati kratak opis problema raspoređivanja u općenitom smislu, kao i problema raspoređivanja na jednom stroju, problema raspoređivanja na nesrodnim paralelnim strojevima i problema raspoređivanja s ograničenim sredstvima.

3.1 Definicija i opis problema raspoređivanja

Raspoređivanje je proces donošenja odluka u kojem je potrebno alocirati sredstva za zadatke koji se trebaju izvršiti kroz neko vrijeme, s ciljem optimizacije jednog ili više željenih kriterija [1]. Pri tome sredstva i zadaci mogu biti raznoliki. Sredstva mogu biti strojevi u radionici, procesori u računalu, piste na aerodromu, itd. Zadaci mogu biti radnje u procesu proizvodnje, izvršavanja računalnih programa, polijetanja i slijetanja zrakoplova itd. Kriteriji koje treba optimizirati biraju se na temelju promatranog problema pa mogu biti npr. minimizacija vremena završetka posljednjeg zadatka, minimizacija broja zadataka koji završavaju nakon svog roka za izvršavanje, itd.

Općenito, kada se opisuje neki problem raspoređivanja, postoji n poslova i m strojeva pri čemu (u klasičnoj teoriji raspoređivanja) niti jedan stroj ne može obrađivati više od jednog posla odjednom, niti se neki posao može izvršavati na dva ili više strojeva u isto vrijeme. Svaki posao ima određena svojstva. Oznaka p_{ij} označava vrijeme izvršavanja (engl. *processing time*) posla j na i - tom stroju. S r_j je označeno vrijeme u kojem posao j postaje dostupan za izvođenje (engl. *release date*). Zatim, d_j označava željeno vrijeme završetka (engl. *due date*) posla j, dok $\bar{d}_j$ označava vrijeme nužnog završetka posla j (engl. *deadline*). Svaki posao ima i neku težinu, koja se označava sa w_j (engl. *weight*).

Gotovo svaki problem raspoređivanja se može opisati trojkom $\alpha|\beta|\gamma$ kako je definirano u [83]. Polje α označava okruženje stroja i može sadržavati samo jednu vrijednost. Polje β

daje detalje o svojstvima izvršavanja i ograničenjima ukoliko ona postoje. Ovo polje može biti prazno, ali može sadržavati i jednu ili više vrijednosti. Polje γ opisuje cilj koji se treba optimizirati i najčešće sadrži samo jednu vrijednost [1].

Polje α se naziva okolina strojeva, a neke od najčešće korištenih vrijednosti koje se u tom polju pojavljuju će biti navedene u nastavku. U zagradi pokraj naziva svake okoline je oznaka za navedenu okolinu koja se stavlja u polje α.

Okolina jednog stroja (1) (engl. *single machine*) je najjednostavnija od svih okolina strojeva u kojoj postoji samo jedan stroj na kojemu se trebaju izvršiti svi poslovi. Ova okolina je specijalan slučaj svih ostalih složenijih okolina.

Okolina paralelnih identičnih strojeva (P_m) (engl. *identical machines in parallel*) je okolina u kojoj postoji m identičnih strojeva koji rade paralelno. Svaki posao se sastoji samo od jedne operacije i može se izvoditi na bilo kojem od m ponuđenih strojeva. Pri tome je trajanje izvođenja posla j jednako na svim strojevima.

Okolina jednolikih strojeva (Q_m) (engl. *machines in parallel with different speeds, uniform machines*) je okolina u kojoj m strojeva radi paralelno, ali je za svaki stroj definirana neka brzina v_i. Vrijeme koje posao j provede na stroju i se označava sa p_{ij} i računa se kao p_j/v_i. Ukoliko je brzina jednaka na svim strojevima, ova okolina je identična prethodno opisanoj.

Okolina nesrodnih strojeva (R_m) (engl. *unrelated machines in parallel*) je dodatna generalizacija prethodne okoline. Ovdje također postoji m strojeva koji rade paralelno, ali je za svaki par posao - stroj definirana brzina kojom se određeni posao izvršava na određenom stroju. Ukoliko su brzine neovisne o poslovima koji se izvode na strojevima, ova okolina je identična prethodno opisanoj.

Okolina obrada tijeka (F_m) (engl. *flow shop*) je okolina u kojoj se svaki posao mora izvršiti na svakom od strojeva i pri tome moraju slijediti određenu rutu.

Okolina proizvoljne obrade (J_m) (engl. *job shop*) je okolina u kojoj se poslovi moraju izvoditi određenim redoslijedom i na određenim strojevima, ali se pri tome posao može izvoditi više puta na nekom stroju ili nijednom na nekom drugom stroju.

Okolina otvorene obrade (O_m) (engl. *open shop*) je okolina u kojoj se svaki posao mora izvršiti na svakom od strojeva, ali ne postoji određen redoslijed kojim se to mora obaviti.

Kao što je već navedeno, u polju β se zapisuju svojstva izvršavanja i ograničenja poslova. U nastavku će biti opisane neke od najčešćih mogućnosti za ovo polje.

Vremena pripravnosti (r_j) - ukoliko se ovaj simbol pojavi u polju β, on označava da j - ti posao ne može započeti sa izvršavanjem prije njegovog vremena pripravnosti r_j.

Prekidivost zadataka (*prmp*) (engl. *preemptions*) - označava da se neki posao smije djelomično izvršiti na nekom stroju, zaustaviti pa dovršiti kasnije u nekom vremenskom trenutku na istom ili nekom drugom stroju.

Ograničenja u redoslijedu (*prec*) (engl. *precedence constraints*) - ukoliko se ova oznaka

pojavi u polju β, to znači da se neki posao ne može izvršiti sve dok nisu izvršeni svi njegovi prethodnici. Ako svaki posao ima samo jednog prethodnika i jednog sljedbenika, onda se u polje β stavlja oznaka *chains*. Ako svaki posao ima najviše jednog sljedbenika, u β se stavlja oznaka *intree*, a ako svaki posao ima najviše jednog prethodnika, onda se stavlja oznaka *outtree*.

Trajanja postavljanja (s_{jk}) - ukoliko se ova oznaka nađe u polje β ona označava da postoji određeno vrijeme koje je potrebno da se stroj pripremi za izvršavanje posla k ako se prije toga na stroju izvršavao posao j.

Dozvoljeno čekanje (*no idleness*) - ukoliko se prilikom izvršavanja pojavi situacija da su strojevi i sredstva potrebna za izvršavanje nekog posla dostupni, ali znamo da će u skorijoj budućnosti neki posao koji ima veću važnost postati dosupan, potrebno je odlučiti hoće li se posao manje važnosti, koji se može početi izvršavati u danom trenutku, staviti na izvršavanje ili će cijeli sustav otići u mirovanje i čekati da se pojavi posao veće važnosti pa sredstva preusmjeriti na njega. Ako postoji mogućnost čekanja, onda se govori o dozvoljenom čekanju (engl. *inserted idleness*), a ukoliko ne postoji mogućnost čekanja, govori se o nedozvoljenom čekanju (engl. *no inserted idleness*). Općenito, ako ništa nije navedeno, pretpostavlja se da je čekanje dozvoljeno.

Čekanje među procesorima (*nwt*) - (engl. *no wait*) ova oznaka se može pojaviti u okolini obrade tijeka ukoliko se svaki posao mora, čim završi s izvršavanjem na jednom stroju, prebaciti na sljedeći. U tom slučaju, vrijeme početka izvršavanja posla na prvom stroju se mora odgoditi do onog trenutka kada se može osigurati da jednom započeti posao može dovršiti izvršavanje na svim strojevima redom kojim se treba izvršiti.

Polje γ sadrži informaciju o funkciji cilja koju je potrebno minimizirati prilikom rješavanja problema raspoređivanja. U ovom polju se može nalaziti jedan ili više kriterija. Ukoliko ih je više, govori se do višekriterijskoj optimizaciji. Neke od najčešće korištenih funkcija cilja će biti nabrojane u nastavku, no prije toga je potrebno definirati vrijednosti koje se mogu naći na izlazu sustava jer se pomoću tih vrijednosti definiraju funkcije cilja. Oznaka C_j obilježava vrijeme završetka j - tog posla (engl. *completion time*). Sa F_j se označava protjecanje (engl. *flow time*), odnosno količina vremena koju je posao j proveo u sustavu i računa se kao $F_j = C_j - r_j$. Kašnjenje L_j (engl. *lateness*) se definira kao $L_j = C_j - d_j$ i označava razliku između vremena završetka posla i vremena željenog završetka posla. Oznakom T_j se označava zaostajanje (engl. *tardiness*) posla j, odnosno pozitivan iznos kašnjenja nekog posla. Ukoliko je posao završio prije svog željenog vremena završetka, T_j će biti jednak nuli. Formula kojom se računa T_j je dana sa $T_j = max\{0, L_j\}$. Zakašnjelost (engl. *unit penalty*) se označava sa U_j i govori je li neki posao prekoračio željeno vrijeme završetka. U_j poprima vrijednost 1 ukoliko je $T_j > 0$ i vrijednost nula ukoliko je $T_j = 0$.

Uz korištenje vrijednosti koje se mogu naći na izlazu sustava, neke od najčešćih funkcija cilja su:

- ukupna duljina rasporeda (C_{max}) (engl. *makespan*) - posljednje vrijeme dovršenja svih poslova u sustavu,
- najveće kašnjenje (L_{max}) (engl. *maximum lateness*) - najveća vrijednost kašnjenja koju postiže neki od poslova; $L_{max} = max\{L_j\}$,
- težinska ukupna duljina rasporeda ($\sum w_j C_j$) (engl. *total weighted completion time*)- suma težinskih vremena završetaka svih poslova; u literaturi se često označava i sa $\sum w_j F_j$ i naziva se težinsko protjecanje (engl. *total weighted flow time*),
- težinsko zaostajanje ($\sum w_j T_j$) (engl. *total weighted tardiness*) se računa kao težinska suma zaostajanja svih poslova,
- težinski zbroj zaostalih poslova ili težinska zakašnjelost (U_w) (engl. *weighted number of tardy jobs*) se računa kao težinska suma svih zaostalih poslova, odnosno $\sum w_j U_j$.

Uz ove navedene kriterije optimizacije, može se definirati proizvoljan broj drugih kriterija koji su specifični za sam problem koji se promatra.

Prilikom rješavanja problema raspoređivanja mogu se koristiti različiti pristupi i algoritmi. Najčešće se metode za rješavanje problema raspoređivanja dijele na egzaktne algoritme, aproksimacijske algoritme i heurističke algoritme [1]. Pri tome se heuristički algoritmi mogu podijeliti na dvije skupine: unaprjeđivačke heuristike koje kreću od nekog gotovog rješenja i kroz iteracije poboljšavaju rješenje te konstruktivne heuristike koje kreću od praznog rasporeda pa dodaju aktivnost po aktivnost dok ne dođu do završnog rasporeda. Uz navedene vrste algoritama, može se koristiti i hiperheuristički pristup, odnosno heuristika koja služi za odabir ili generiranje najbolje heuristike koja će riješiti promatrani problem. U ovom radu se prilikom rješavanja problema raspoređivanja koristi genetsko programiranje kao hiperheuristika koja razvija prioritetna pravila koja se koriste prilikom pravljenja samog rasporeda nekom od metoda generiranja rasporeda. Više o samim metodama generiranja rasporeda će biti navedeno u poglavlju 6.

3.2 Problem raspoređivanja na jednom stroju

Kao što je već napomenuto, raspoređivanje na jednom stroju je najjednostavniji oblik raspoređivanja i može se smatrati jednim od specijalnih slučajeva svih ostalih složenijih okolina raspoređivanja. U ovom problemu postoji samo jedan stroj na kojemu se mora izvršiti svih n danih poslova. Pri tome se u polju β mogu naći različite vrijednosti u ovisnosti o tome postoje li neka dodatna ograničenja ili posebni uvjeti u samom postupku.

S obzirom da je ovaj problem specijalan slučaj složenijih problema, i u novije vrijeme se aktivno istražuje. Autori u [84] istražuju robusnu verziju postupaka rješavanja problema raspoređivanja na jednom stroju i predlažu hiperheuristiku temeljenu na simuliranom kaljenju za rješavanje ovog problema. Robusnom verzijom postupaka rješavanja problema bave se i autori

u [85], ali u tom radu trajanja dolaze iz određenih intervala i za kriterij optimizacije se koristi ukupno težinsko zaostajanje. Navedeni problem se rješava koristeći miješano cjelobrojno linearno programiranje (engl. *mixed - integer linear programming*) i dva aproksimacijska algoritma. I autori u [86] se bave robusnom verzijom problema, ali koriste kriterij ukupne duljine rasporeda pri čemu vremena kada poslovi postaju dostupni nisu unaprijed poznata. Za rješavanje ovog problema predlažu dva algoritma koji rješenje daju u polinomijalnom vremenu. Autori u [6] se bave efektom učenja kod problema raspoređivanja na jednom stroju u svrhu minimiziranja kriterija zasnovanih na vremenu završetka posla pri čemu u problemu postoje vremena kada poslovi postaju dostupni. Ovaj problem je u navedenom radu riješen koristeći dvije heuristike, dok se za male instance koristi algoritam grananja i ograđivanja (engl. *branch and bound algorithm*). U [7] se istražuje problem raspoređivanja na jednom stroju u kojem poslovi imaju zadano vrijeme kada postaju dostupni, zadana su vremena do kojih poslovi moraju završiti s izvršavanjem, postoji trajanje pripreme stroja između dva posla i postoji mogućnost odbijanja izvršavanja nekog posla. Autori definiraju egzaktan algoritam i polinomijalnu aproksimacijsku shemu (engl. *fully-polynomial time approximation scheme*) za rješavanje ovakve vrste problema. Autori u [87] minimiziraju krajnje vrijeme završetka svih poslova pri čemu postoje vremena kada poslovi postaju dostupni, svaki od poslova ima utjecaj (pozitivan ili negativan) na centralno stanje inventara u skladištu pa su zadana i ograničenja na inventar. U tu svrhu koriste četiri algoritma, pri čemu se pokazalo da u ovom slučaju algoritam pogađanja i provjere zasnovan na dinamičkom programiranju daje najbolje rješenje. U [88] se istražuje energetski efikasan problem raspoređivanja na jednom stroju u kojem postoje ograničenja na pouzdanost stroja. Za rješavanje ovog problema definiran je model matematičkog programiranja s ciljem minimiziranja zaostajanja i utrošene energije, dok se za rješavanje problema iz stvarnog svijeta koristio modificirani mravlji algoritam. I autori u [89] se bave energetski efikasnim problemom raspoređivanja na jednom stroju, ali oni u ovom radu daju analizu kompleksnosti ove vrste problema i uvode donje granice za općenit slučaj ovog problema.

Iz ovog kratkog pregleda najnovijih radova u području raspoređivanja na jednom stroju vidi se da je ovaj problem aktualan i da ima smisla istraživati njegova svojstva i u budućnosti.

U ovom radu će se promatrati problem raspoređivanja na jednom stroju sa sljedećim svojstvima:

- svi podaci o poslovima su unaprijed poznati,
- raspoređivanje se obavlja tijekom rada sustava (dinamički),
- poslovi se ne smiju prekidati jednom kad započnu s izvršavanjem,
- stroj je uvijek raspoloživ.

Ova svojstva vrijede u većini prikaza problema raspoređivanja na jednom stroju u literaturi pa se iz tog razloga i ovdje uzimaju u obzir.

3.3 Problem raspoređivanja na nesrodnim paralelnim strojevima

Problem raspoređivanja na nesrodnim paralelnim strojevima je problem u kojem postoji n poslova koji se trebaju izvršavati na jednom od m dostupnih strojeva. Pri tome svaki stroj ima određenu brzinu kojom se svaki posao na njemu može izvršavati i te brzine nisu nužno jednake za različite kombinacije posao - stroj pa treba pronaći i odgovarajuće uparivanje poslova sa strojevima, a zatim poslove koji su dodijeljeni nekom stroju rasporediti kako bi se dobio krajnji raspored i minimizirao željeni kriterij.

Ovaj problem se može pronaći u raznim situacijama u stvarnom svijetu, poput prizemljivanja zrakoplova u zračnim lukama, raspoređivanja operativnih zahvata u operacijske sale u bolnicama, u industriji boja i plastike, itd. Upravo iz tog razloga se i ovaj problem aktivno istražuje što pokazuje i mnoštvo novijih znanstvenih radova na tu temu. Kratak pregled novijih radova koji se bave problemom raspoređivanja na nesrodnim strojevima će biti dan u nastavku.

Autori u [90] daju usporedbu sedam različitih reprezentacija rješenja za rješavanje problema raspoređivanja na nesrodnim paralelnim strojevima korištenjem metaheuristika. Rad [29] daje usporedbu shema za generiranje rasporeda korištenjem prioritetnih pravila za genetsko programiranje. Autori u [91] koriste metaheuristike (genetski algoritam i simulirano kaljenje) za rješavanje problema raspoređivanja na nesrodnim paralelnim strojevima sa slučajnom mrežom u svrhu minimiziranja očekivanog težinskog zaostajanja. U radu [92] se analizira kako djeluje varijabilni spust po susjedstvu (engl. *variable neighborhood descent*) kao operator lokalne pretrage za rješavanje problema raspoređivanja na nesrodnim strojevima, pri čemu se minimizira ukupno težinsko zaostajanje. Autor u [8] predlaže matematički model za opisivanje problema raspoređivanja na nesrodnim paralelnim strojevima uz ograničenja na dostupna sredstva i postojanje trajanja pripreme stroja između obavljanja dvaju poslova. Predložen je i egzaktan algoritam za rješavanje navedenog problema. Autori u [9] proučavaju energetski efikasan problem raspoređivanja na nesrodnim strojevima u kojem se posebno promatraju različite tarife za obračunavanje potrošnje energije. Za rješavanje ovog problema predložen je model miješanog cjelobrojnog programiranja (engl. *mixed - integer programming*) i jedna heuristička metoda te je pokazano da predloženi modeli dobro rješavaju promatrani problem. Rad [93] koristi algoritam kolonije pčela (engl. *artificial bee colony algorithm*) za rješavanje distribuiranog problema raspoređivanja na nesrodnim strojevima u kojem postoji preventivno održavanje strojeva. Autori u [94] daju konstruktivne heuristike za rješavanje problema raspoređivanja s nesrodnim strojevima u kojem strojevi nisu dostupni u svakoj vremenskoj jedinici i postoji određeno vrijeme pripreme stroja između obavljanja dvaju poslova. Autori u [95] daju pregled prioritetnih pravila za dinamički problem raspoređivanja na nesrodnim strojevima. Napravljena je analiza performansi za 26 pravila prilikom optimiziranja 9 različitih kriterija.

U ovom radu će se promatrati problem raspoređivanja na nesrodnim strojevima sa sljedećim svojstvima:

- podaci o poslovima postaju poznati tek kad se posao pojavi u sustavu,
- raspoređivanje se obavlja tijekom rada sustava (dinamički),
- poslovi se ne smiju prekidati jednom kad započnu s izvršavanjem,
- svi strojevi su uvijek raspoloživi.

3.4 Problem raspoređivanja s ograničenim sredstvima

Problem raspoređivanja s ograničenim sredstvima (engl. *resource - constrained project scheduling problem* - RCPSP) je uveo Kelley u radu iz 1963. godine [96]. U ovom problemu se razmatraju sredstva koji imaju ograničenu količinu i/ili dostupnost u nekim vremenskim intervalima i poslovi, odnosno aktivnosti za koje se zna kolika je njihova duljina izvršavanja i koliko kojeg sredstva je potrebno za njihovo izvršavanje. Za svaki posao je dana i lista njegovih prethodnika i sljedbenika, odnosno, poznati su uvjeti prednosti. U osnovnom obliku RCPSP-a poslovi se ne smiju prekidati jednom kada njihovo izvršavanje započne, sredstva se obnavljaju u svakom vremenskom trenutku i kriterij koji treba minimizirati je vrijeme završetka projekta.

Formalno, RCPSP se može definirati kao problem kombinatorne optimizacije u obliku uređene sedmorke [97]:

$$RCPSP = (A, E, p, R, B, D, c).$$

Prvi element zapisa označava skup svih aktivnosti koje se nalaze u projektu i definira se kao $A = \{A_0, A_1, \cdots, A_n, A_{n+1}\}$, gdje su A_0 i A_{n+1} fiktivne aktivnosti (engl. *dummy*) duljine nula koje predstavljaju početak i završetak projekta. Oznakom A' označavamo skup aktivnosti iz kojih su izbačene fiktivne aktivnosti A_0 i A_{n+1}.

Drugi element daje uvjete prednosti (engl. *precedence constraints*). Skup E se sastoji od uređenih parova oblika $(A_i, A_j), i, j \in 0, 1, \cdots, n, n+1$ koji označavaju da je aktivnost A_i prethodnik aktivnosti A_j. Aktivnost A_0 je prethodnik svim ostalim aktivnostima, dok je aktivnost A_{n+1} sljedbenik svih aktivnosti u projektu.

Vektorom p se označava trajanje svih aktivnosti u projektu. Vektor p je vektor prirodnih brojeva (u njemu se dodatno smije nalaziti i nula) duljine $n+2$ u kojemu se na i - toj poziciji nalazi vrijeme potrebno za izvršavanje aktivnosti A_i.

Oznakom $R = \{R_1, \cdots, R_m\}$ su označena obnovljiva sredstva čija je dostupnost u svakom vremenskom trenutku dana vektorom $B \in \mathbb{N}^m$. Sa $D \in \mathbb{N}_0^{(n+2)\times m}$ je označena matrica u kojoj se na odgovarajućim pozicijama nalaze zahtjevi pojedine aktivnosti za pojedinim sredstvom. Aktivnosti A_0 i A_{n+1} nemaju nikakve zahtjeve na sredstva, odnosno u matrici D u retcima koji odgovaraju tim aktivnostima se nalaze samo vrijednosti 0.

Zadnji element zapisa, c predstavlja funkciju cilja definiranu kao $c : \chi \to \mathbb{R}$. Ova funkcija

rasporedima iz skupa svih mogućih rasporeda pridružuje neku realnu vrijednost. Ukoliko u zapisu problema nema funkcije, pretpostavlja se da je cilj minimizirati vrijeme završetka projekta.

Rješenje RCPSP-a se zapisuje vektorom $S \in \chi \subseteq \mathbb{R}^{n+2}$ u kojem i - ta komponenta predstavlja vrijeme početka izvršenja aktivnosti A_i. Vektor S treba biti takav da je funkcija cilja optimalna.

Ukoliko s $A_t = \{A_i \in A : S_i \leq t \leq S_i + p_i\}$ označimo skup svih aktivnosti koje se izvršavaju u trenutku t, RCPSP možemo zapisati kao problem kombinatorne optimizacije u kojem je cilj pronaći raspored S tako da su zadovoljeni uvjeti:

$$S_j - S_i \geq p_i, \quad \forall (A_i, A_j) \in E \tag{3.1}$$

$$\sum_{A_i \in A(t)} D_{ij} \leq B_j, \quad \forall t \geq 0, \forall R_j \in R \tag{3.2}$$

Uvjet 3.1 osigurava da se aktivnosti izvršavaju dobrim redoslijedom (bez narušavanja uvjeta prednosti), dok uvjet 3.2 osigurava da se poštuju ograničenja na dostupnu količinu sredstva.

S obzirom da se ovaj model može primijeniti na mnoge probleme iz stvarnog života, i on se aktivno istražuje kao i prethodna dva problema raspoređivanja. U nastavku će biti nabrojani neki od novijih radova koji se bave istraživanjem i rješavanjem problema raspoređivanja s ograničenim sredstvima.

Autori u [98] predlažu hiperheurističkipristup za rješavanje stohastičke inačice ovog problema korištenjem ansambala i genetskog programiranja. U stohastičkoj verziji problema raspoređivanja s ograničenim sredstvima, vremena izvršavanja se biraju na slučajan način. Rad [99] proučava prekide u promjenama modula (engl. *mode change disruption*) kod problema raspoređivanja s ograničenim sredstvima s više modula (engl. *Multi-mode Resource Constrained Project Scheduling Problem*). Ukoliko se dogode prekidi, raspored se može znatno pogoršati pa se predlažu reaktivni matematički modeli i heuristike koji će popraviti nastalo pogoršanje. Autori u [100] koriste klasifikator kako bi odredili koje je prioritetno pravilo najbolje za specifičan projekt iz skupa problema raspoređivanja s ograničenim sredstvima. Pokazalo se da se na taj način dobivaju bolji rezultati nego korištenjem samo jednog prioritetnog pravila za sve instance problema. Autori u [101] definiraju algoritam diferencijske evolucije (engl. *differential evolution algortihm*) u dvije faze sa više operatora u svrhu rješavanja RCPSP-a, dok autori u [102] koriste memetički algoritam baziran na genetskom algoritmu. Rad [103] se bavi istraživanjem prioritetnih pravila za stohastički problem raspoređivanja s više projekata pri čemu, za vrijeme trajanja izvršavanja, postoji mogućnost dolazaka novih projekata koje treba riješiti. Autori u [104] predlažu novu *rollout* heuristiku koja poboljšava postojeća rješenja za dobivanje rasporeda u problemu raspoređivanja s ograničenim sredstvima u kojemu postoje aktivnosti koje se preklapaju. Autori u [16] koriste genetsko programiranje kako bi razvili prioritetna pravila za rješavanje RCPSP-a.

Iz ovog kratkog pregleda radova se može vidjeti da se za rješavanje problema raspoređivanja s ograničenim sredstvima koriste razni algoritmi i da se različite inačice ovog problema pojavljuju u primjenama. U ovom radu će se promatrati RCPSP u kojem se aktivnost koja je počela s izvršavanjem ne smije prekinuti, a sredstva su obnovljiva i jednaka količina nekog sredstva je dostupna u svakom vremenskom intervalu.

O samim instancama koje se koriste u ovom radu za sva tri navedena problema će biti riječi u poglavlju 6.

Poglavlje 4

Genetsko (genetičko) programiranje

Genetsko programiranje (GP) je tehnika koja pripada skupini evolucijskih algoritama čiji je cilj razviti programe koji će rješavati zadani problem. Pri tome korisnik ne mora znati ili unaprijed odrediti oblik ili strukturu rješenja problema. Najopćenitije, GP je sistematična metoda neovisna o domeni, pomoću koje računala rješavaju probleme automatski, počevši od općenitog ("high-level") objašnjenja što točno treba riješiti. [17]

Osnove za genetsko programiranje su postavljene 1990-ih godina u radovima J.R.Koze [105] [106] i kasnije su dopunjene novim saznanjima u dodatnim knjigama [107] [108] [109]. Od uvođenja principa genetskog programiranja do danas mnogi su znanstvenici iskoristili uvedene principe za rješavanje problema u raznim područjima. Autori u [17] navode kako se GP pokazao kao dobrim alatom za rješavanje u područjima koja imaju neka od sljedećih svojstava:

- veze između bitnih varijabli su nepoznate ili nedovoljno razumljive;
- pronalazak veličine i oblika krajnjeg rješenja je veliki dio problema;
- velika količina podataka je dostupna u obliku koji računalo može iskoristiti;
- postoje dobri simulatori za testiranje izvršavanja privremenih rješenja nekog problema, ali ne postoje dobre metode za dobivanje dobrih rješenja;
- rješenja se ne mogu dobiti analitički;
- aproksimacijsko rješenje je prihvatljivo;
- mala poboljšanja u učinku se rutinski mjere i vrlo su važna.

Iz nabrojanih svojstava je vidljivo dase mnogi problemi mogu svrstati u područja u kojima GP dobro funkcionira i upravo to je razlog što se GP i dalje aktivno istražuje u znanstvenom svijetu.

Jedno od aktivnijih područja gdje se GP koristi je problem simboličke regresije, tj. traženja funkcije čije će vrijednosti imati neko željeno svojstvo (npr. vrijednosti funkcije će odgovarati unaprijed danim vrijednostima u nekim točkama). Simbolička regresija je bilo jedno od prvih područja u kojima se korstio GP [106], ali se nastavlja koristiti i u novijim radovima iz tog područja [110] [111] [112] [113].

Nadalje, GP se koristi za dobivanje rezultata usporedivih sa rezultatima dobivenim rješe-

njima koja su osmislili stručnjaci iz određenog područja. Neke od primjena u toj domeni su: stvaranje kvantnih algoritama, uključujući i algoritam za pretraživanje baza podataka [114], stvaranje algoritma za problem identifikacije segmenta membrane kod proteina [107] [108], stvaranje pravila za klasifikaciju većine koje je bolje od svih dotad ljudski definiranih pravila [115], dizajniranje antene za korištenje u NASA-i [116], stvaranje rješenja za praćenje brzine kod mobilnih robota [117], primjena na revezibilne transformacije invarijantne na pomak [118], automatsko otkrivanje test-statistika [119] i mnoge druge.

Još neka od područja u kojima se GP aktivno koristi su i obrada slike i signala [120] [121] [122], ekonomsko modeliranje i financije [123] [124], medicina, biologija i bioinformatika [125] [126], računalne igre [127] [128], umjetnost [129] [130] itd.

Zanimljivo je da se genetsko programiranje koristi i kao hiperheuristika, odnosno heuristika koja stvara ili odabire druge heuristike. Razlika između metaheuristike i hiperheuristike je u tome što metaheuristika izravno rješava neki dani problem, dok hiperheuristika pretražuje prostor heuristika koje bi mogle riješiti dani problem i pokušava pronaći najbolju. Neka od područja u kojima se GP dosta uspješno koristi kao hiperheuristika su: stvaranje rješenja za probleme zadovoljivosti (engl. *satisfiability problems*) [131], stvaranje rješenja za algoritme roja čestica [132] [133], stvaranje rješenja za problem trgovačkog putnika [134].

Kao što je već napomenuto u poglavlju 3, u ovom radu će se koristiti problemi raspoređivanja kao bazni problemi na kojima će se pokazati kako radi predloženi pristup. I za ove tipove problema GP se uspješno koristi kao hiperheuristika. U novije vrijeme neki od radova u kojima se GP koristi za traženje dobrih heuristika za rješavanje problema raspoređivanja bave se problemima raspoređivanja na jednom stroju [27] [28], problemima raspoređivanja u okružju nesrodnih strojeva [29] [15], problemom raspoređivanja s ograničenim sredstvima [30] [16] [135], problemom proizvoljne obrade [31] i mnogim drugima.

U ostatku ovog poglavlja će biti opisan osnovni algoritam za GP, kao i svi njegovi građevni blokovi.

4.1 Osnovni algoritam

Genetsko programiranje prilikom rješavanja nekog problema ne radi sa samo jednim rješenjem, nego koristi populaciju rješenja (jedinki) na kojima izvršava operatore selekcije (odabira), križanja i mutacije kako bi se na kraju dobilo najbolje moguće rješenje.

Osnovni koraci GP-a su dani u algoritmu 1. Algoritam počinje stvaranjem početne populacije jedinki. Postoje razni načini za generiranje jedinki što će biti objašnjeno malo dalje u ovom poglavlju. Broj jedinki ovisi o unaprijed određenoj veličini populacije koju zadaje korisnik. Nakon stvaranja početne populacije, odabiru se jedinke koje će se zatim iskoristiti za križanje kako bi se stvorile nove jedinke i zatim se te jedinke mogu mutirati da bi se dobila još

različitija rješenja. Novodobivene jedinke se uspoređuju sa odabranima na temelju vrijednosti funkcije cilja i ukoliko su bolje, stavljaju se u novu populaciju. Ovaj postupak se ponavlja dok se ne dostigne neki od kriterija zaustavljanja poput maksimalnog broja generacija, maksimalnog broja evaluacija, dovoljno dobre vrijednosti funkcije cilja najbolje jedinke, itd.

Algoritam 1 Osnovni algoritam genetskog programiranja

```
1: inicijaliziraj jedinke za početnu populaciju
2: ponavljaj
3:     odaberi potreban broj jedinki na osnovu vrijednosti funkcije cilja
4:     koristeći operatore mutacije i križanja stvori nove jedinke od odabranih
5: dok nije zadovoljen neki od kriterija zaustavljanja
6: vrati najbolju jedinku iz trenutne populacije kao rješenje
```

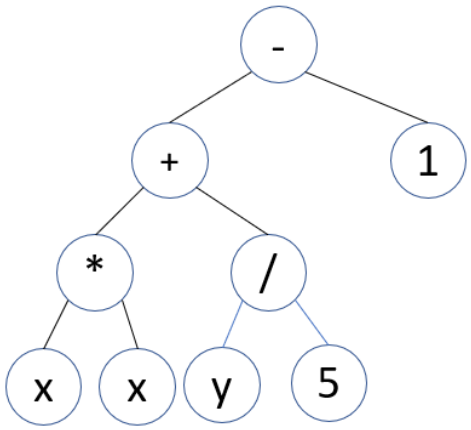
Na temelju načina na koji se nove jedinke stavljaju u populaciju razlikujemo eliminacijski i generacijski GP. Kod eliminacijskog GP-a selekcijom se odabere određeni broj jedinki, dvije najbolje se odabiru da budu roditelji za križanje, pri čemu se stvori nova jedinka. Ta jedinka zamjenjuje najgoru od prethodno odabranih jedinki u trenutnoj populaciji. Kod generacijskog GP-a, nova jedinka ne zamjenjuje jedinke iz trenutne populacije, nego se stavlja u posebnu populaciju i taj se postupak selekcije i stvaranja novih jedinki ponavlja sve dok veličina nove populacije ne dosegne veličinu prethodne populacije. U generacijskom GP-u, s obzirom da se svaki put stvara cijela nova populacija, može doći do gubitka najboljih jedinki iz prethodne populacije. Kako bi se to izbjeglo uvodi se elitizam. Elitizam je postupak u kojem se jedna ili više najboljih jedinki iz prethodne generacije automatski prebacuje u novu generaciju jedinki kako se ne bi izgubio trenutno najbolji genetski materijal.

4.2 Prikaz rješenja

Genetsko programiranje najčešće za prikaz jedinke koristi stabla. Razlog tome je što su stabla struktura podataka koja može reprezentirati neke složenije izraze. Cilj korištenja GP-a je od nekih jednostavnijih građevnih blokova doći do rješenja koje će reprezentirati možda i neki složeniji izraz, a stabla upravo to i omogućuju. Prikaz jedne jedinke u obliku stabla se može vidjeti na slici 4.1. Ovo stablo predstavlja izraz $x^2 + \frac{y}{5} - 1$. Ovaj izraz se može zapisati i u tzv. prefiksnoj notaciji koja se često koristi prilikom zapisivanja stabala. Stablo za slike u prefiksnoj notaciji ima oblik `(- (+ (* x x) (/ y 5)) 1)`.

Kako bi stabla mogla tvoriti rješenja, potrebno im je dati građevne blokove koji se nazivaju primitivi. Primitivi se dijele na funkcije i značajke (terminale).

Skup funkcija i značajki koji će se koristiti uvelike ovisi o domeni u kojoj se GP primjenjuje. Skup značajki se najčešće sastoji od varijabli koje u određenim situacijama poprimaju određene vrijednosti, konstanti i nekih funkcija bez argumenata, poput funkcije `random()` koja će svaki



Slika 4.1: Prikaz jedinke koja predstavlja izraz: $x^2 + \frac{y}{5} - 1$

put kada se nađe u čvoru stabla generirati neku slučajnu vrijednost. Značajke se mogu pojaviti samo u listovima stabla kako bi stablo bilo dopustivo.

Skup funkcija se može sastojati od klasičnih aritmetičkih operacija zbranja, oduzimanja, množenja i dijeljenja (pri tome se kod dijeljenja posebno pazi na slučaj dijeljenja s nulom), zatim od matematičkih funkcija poput sinusa, kosinusa, logaritma, Booleovih izraza (`AND, OR, NOT`), grananja (`IF-THEN-ELSE`), petlji (`FOR, REPEAT`), raznih naredbi specifičnih za područje u kojem se GP koristi (npr. kretanje naprijed kod robotskog usisavača), itd. Funkcije se ne mogu nalaziti u listovima stabla s obzirom da one uvijek moraju primiti neke argumente. Poseban slučaj su one funkcije koje ne primaju argumente (npr. funkcija koja govori robotu da se pomakne unaprijed) pa se u tom slučaju funkcije mogu nalaziti u listovima stabla.

Prilikom odabira skupa značajki i funkcija za rješavanje nekog problema bitno je odabrati dovoljno velik skup značajki tako da se u njemu nalazi sve potrebno kako bi se optimalno rješenje moglo prikazati. No, treba paziti da skup ne bude prevelik jer nepotrebne značajke ili funkcije mogu dovesti do loših rješenja i povećati prostor pretraživanja GP-a, a time i znatno usporiti konvergenciju algoritma.

Najčešće se kod korištenja stabala kao jedinki za GP ograničava maksimalna dubina stabla ili maksimalan broj čvorova u stablu kako bi se izbjeglo nagomilavanje nepotrebnih informacija. Što su stabla veća, to je teže baratati njima, javlja se problem pretjeranog rasta (engl. *bloat*) prilikom stvaranja novih jedinki koje većinom nemaju poboljšanje u vrijednosti funkcije dobrote u odnosu na puno manja stabla. Isto tako, što je stablo veće, to je teže interpretirati rješenje koje to stablo predstavlja. Još uvijek nije do kraja poznato zašto se točno javlja problem pretjeranog rasta pa se taj problem aktivno pokušava shvatiti i njegovo se pojavljivanje pokušava što više smanjiti, što dokazuju i mnoga istraživanja poput [106] [136] [137] [138].

Iako je prikaz stablom najčešći prikaz jedinki u GP-u, koriste se i neki drugi prikazi poput: linearnog GP-a (engl. *linear GP*) [139] [140], GP-a zasnovanog na grafu (engl. *graph-based GP*) [141], GP-a zasnovanog na gramatici (engl. *grammar-based GP*) [142] [143] i kartezijevog

prikaza (engl. *Cartesian GP*) [144] [145].

4.3 Inicijalizacija

Kako je navedeno u osnovnom algoritmu za GP, prvi korak je inicijalizirati populaciju jedinki na kojima će se onda izvršavati operatori odabira, križanja i mutacije u svrhu dobivanja boljih jedinki. Sama inicijalizacija se većinom radi na slučajan način koristeći neku od metoda osmišljenih za stvaranje stabala.

Dvije najjednostavnije i najraširenije metode za generiranje stabala su: metoda potpunosti (engl. *full method*) i metoda rasta (engl. *grow method*). Kako bi se smanjile negativne značajke koje se javljaju kada se koristi samo jedna od tih dviju metoda, uvodi se i pojačana pola-pola metoda (engl. *ramped half-and-half method*) koja je kombinaciju prvih dviju. U nastavku će biti opisane navedene tri metode.

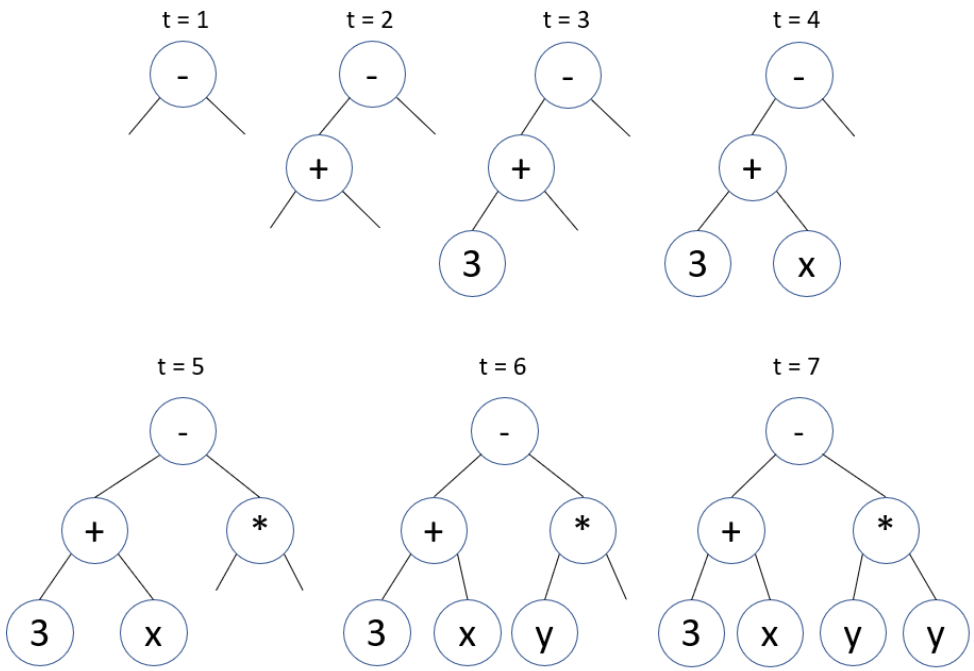
U sve tri metode su početna stabla generirana tako da ne prelaze neku unaprijed određenu dubinu stabla. Dubina nekog čvora u stablu je broj bridova koji se trebaju prijeći da bi se došlo od korijena stabla do tog čvora. Dubina stabla je tada maksimalna dubina koja se postiže na nekom od listova stabla.

Metoda potpunosti je dobila to ime jer se pomoću nje stvaraju potpuna stabla, odnosno stabla kojima su svi listovi na maksimalnoj dozvoljenoj dubini. U ovoj metodi se elementi iz skupa funkcija odabiru na slučajan način, krenuvši od korijenskog čvora, sve dok se ne dođe do maksimalne dubine stabla. Kada se dođe do maksimalne dubine, za čvorove se mogu odabrati samo elementi iz skupa značajki. Slika 4.2 prikazuje nastanak jednog stabla korištenjem metode potpunosti. Na slici slovo t označava vrijeme, a maksimalna dozvoljena dubina stabla je 2.

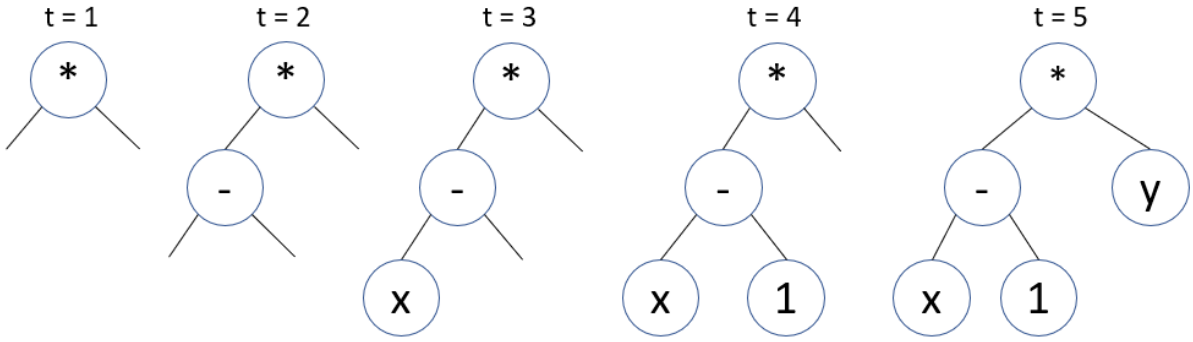
Iako su sva stabla nastala ovom metodom jednake dubine, to ne znači da su ona jednakog oblika. Različiti oblici se mogu dobiti ukoliko funkcije iz skupa funkcija imaju različit broj djece. No, i dalje su stabla dobivena ovom metodom u konačnici dosta slična. Kako bi se povećala raznolikost u veličinama i oblicima stabala, koristi se metoda rasta.

Kod metode rasta se za čvorove stabala može odabrati bilo koji element iz skupa primitiva sve dok se ne dođe do maksimalne dubine stabla gdje se čvorovi kao i kod metode potpunosti mogu odabrati samo iz skupa značajki. Slika 4.3 prikazuje nastanak jednog stabla korištenjem metode rasta. Jednako kao kod metode potpunosti, t označava vrijeme, a maksimalna dozvoljena dubina stabla je 2.

S obzirom da niti metoda rasta niti metoda potpunosti same ne postižu veliku raznolikost u oblicima i veličinama stabala koje generiraju, uvodi se pojačana pola-pola metoda koja kombinira navedene dvije metode. U ovoj metodi se pola stabala generira korištenjem metode potpunosti, a pola stabala se generira metodom rasta. Pri tome se maksimalne dubine za svako stablo biraju iz skupa dubina od 1 do maksimalne dubine stabla koja je zadana na početku. Na taj



Slika 4.2: Prikaz nastajanja stabla korištenjem metode potpunosti



Slika 4.3: Prikaz nastajanja stabla korištenjem metode rasta

način se dobivaju stabla različitih dubina, veličina i oblika.

Važno je napomenuti da izgled stabala ovisi i o broju značajki i funkcija u skupu primitiva. Ako je skup značajki znatno veći od skupa funkcija, metoda rasta će generirati veoma kratka stabla neovisno o zadanoj maksimalnoj dubini stabla. S druge strane, ako je skup funkcija znatno veći od skupa značajki, metoda rasta će generirati stabla slična stablima koja su generirana koristeći metodu potpunosti.

Iako se najčešće početna populacija generira na slučajan način, ukoliko postoje neka dodatna saznanja o svojstvima željenih rješenja, stabla se u početnoj populaciji mogu prilagoditi tim svojstvima kako bi se što prije dobila neka bolja rješenja.

4.4 Funkcija dobrote (evaluacija)

U osnovnom algoritmu za genetsko programiranje je u trećem koraku navedeno da treba odabrati broj jedinki na osnovu vrijednosti funkcije cilja. To znači da je potrebno definirati neki mehanizam kojim će se moći razlikovati "dobre" od "loših" jedinki. Funkcija dobrote je upravo taj mehanizam.

Funkcija dobrote je funkcija koja svakoj jedinki pridružuje neku numeričku vrijednost kako bi se dobrote jedinki mogle međusobno uspoređivati. Dobrota se može mjeriti u raznim terminima, npr. kao pogreška između dobivenog i željenog rješenja, kao vrijeme potrebno za dobivanje željenog krajnjeg rješenja uporabom jedinke iz GP-a, kao točnost koju program postiže u prepoznavanju uzoraka ili prilikom klasifikacije objekata, itd. U ovisnosti o tome je li promatrani problem minimizacijski ili maksimizacijski, reći ćemo da su bolje jedinke one koje postižu manju vrijednost funkcije dobrote (za minimizacijske probleme) ili one koje postižu veću vrijednost funkcije dobrote (za maksimizacijske probleme).

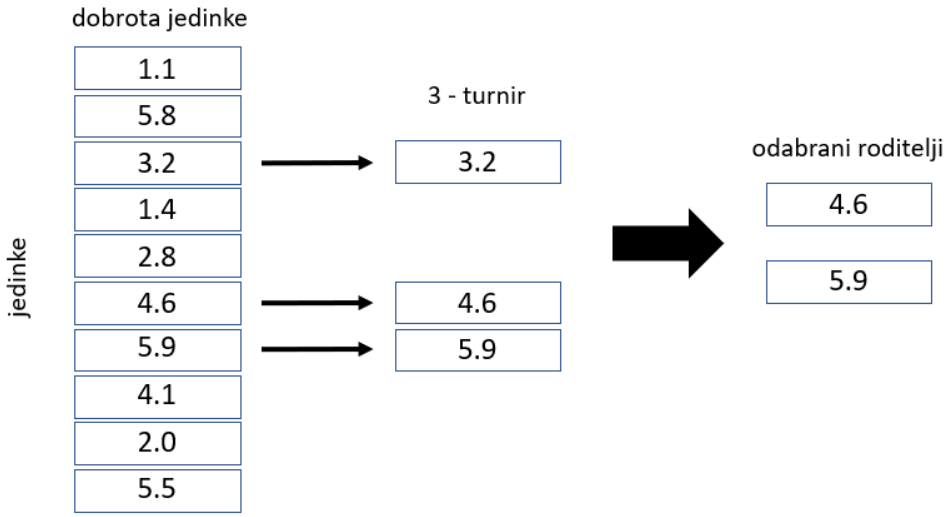
Kod genetskog programiranja se rješenja iz populacije najčešće moraju evaluirati na više instanci problema u svakom koraku. Zbog toga se vrijeme izvršavanja algoritma značajno povećava povećavanjem broja instanci u skupu za učenje. Trebalo bi odabrati takav skup za učenje koji će biti dovoljno velik da dobro reprezentira svojstva problema, a s druge strane dovoljno malen da ne utječe previše na vrijeme izvršavanja cijelog algoritma.

4.5 Odabir jedinki (selekcija)

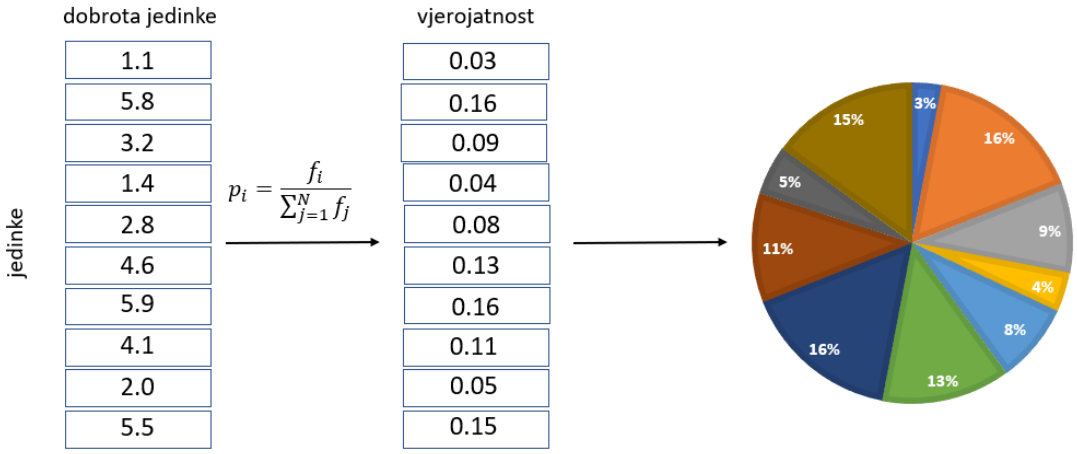
Kako bi se od postojećih jedinki u trenutnoj populaciji mogle dobiti bolje jedinke korištenjem operatora križanja i kasnije mutacije, potrebno je na neki način odabrati jedinke iz trenutne populacije koje će proslijediti svoj genetski materijal dalje. Logično je da se pri tome odabiru što bolje jedinke kako bi se dobar genetski materijal sačuvao u sljedećim generacijama. Postoje razni oblici odabira jedinki, a neki od najpoznatijih su turnirski odabir (engl. *tournament selection*) i jednostavni odabir (engl. *roulette wheel selection, fitness proportionate selection*).

Kod turnirskog odabira k jedinki se odabire na slučajan način iz trenutne populacije. S obzirom da je za križanje potrebno dvije jedinke, od k odabranih se odabiru dvije s najboljim vrijednostima funkcije dobrote. Ukoliko se radi o generacijskom GP-u, dijete koje nastane križanjem tih dvaju roditelja se stavlja u novu populaciju, a ukoliko se radi o eliminacijskom GP-u, dijete koje nastane križanjem tih dvaju roditelja se stavlja u trenutnu populaciju umjesto najlošije od k odabranih jedinki. Zbog slučajnog odabira jedinki u ovom slučaju, može se dogoditi da i lošije jedinke iz populacije budu odabrane kao roditelji u operatoru križanja čime se čuva raznolikost u populaciji. Slika 4.4 prikazuje turnirski odabir na jednoj maloj populaciji.

U jednostavnom odabiru, svaka jedinka ima vjerojatnost da bude odabrana za roditelja pro-



Slika 4.4: Prikaz 3-turnirskog odabira na populaciji od 10 jedinki



Slika 4.5: Prikaz jednostavnog odabira na populaciji od 10 jedinki

porcionalnu svojoj vrijednosti funkcije dobrote. U tom postupku će jedinke bolje dobrote češće biti odabrane za križanje i stvaranje novih jedinki. Ukoliko sa f_i označimo vrijednost funkcije dobrote i-te jedinke iz populacije od ukupno N jedinki, onda se vjerojatnost da i-ta jedinka bude odabrana može izračunati izrazom: $p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$. Najveći problem ove vrste odabira je što se za svaku promjenu u populaciji vjerojatnosti za odabir svake jedinke moraju ponovo računati pa to može dosta usporiti rad algoritma. Slika 4.5 prikazuje jednostavan odabir na jednoj maloj populaciji. Jednostavan odabir možemo zamisliti poput kruga na kojemu su označene površine za svaku jedinku iz populacije proporcionalne vjerojatnostima da ta jedinka bude odabrana. Pokraj kruga se fiksira neka točka i krug se zavrti. Kada krug stane, odabire se jedinka na čiju površinu fiksirana točka pokazuje. Iz toga se slikovito vidi da uistinu jedinke s boljom vrijednošću funkcije dobrote imaju veću vjerojatnost da budu odabrane za prijenos genetskog materijala na nove jedinke.

Osim ove dvije navedene vrste odabira, postoje i druge koje se koriste u raznim problemima.

Neke od njih su: stohastičko univerzalno uzorkovanje (engl. *stohastic universal sampling*), odabir prema rangu (engl. *rank selection*), slučajni odabir (engl. *random selection*), odabir odsijecanjem (engl. *truncation selection*). Više o navedenim vrstama odabira se može pronaći u radovima [146] i [147].

4.6 Operatori križanja i mutacije

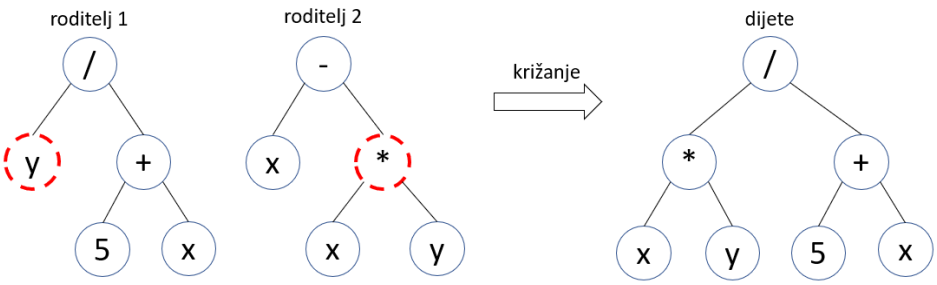
Prilikom rješavanja nekog problema korištenjem GP-a potrebno je na neki način istražiti prostor stanja. Genetički operatori služe kako bi se kombinacijom dviju ili više jedinki (križanje), ili promjenom neke odabrane jedinke (mutacija) došlo do novih jedinki i na taj način prošlo kroz prostor stanja danog problema. Kako je u GP-u jedinka najčešće prikazana u obliku stabla, bilo je potrebno definirati operatore križanja i mutacije koji će znati raditi sa stablima.

4.6.1 Operatori križanja

Cilj operatora križanja (engl. *crossover operator*) je prijenos genetskog materijala s odabranih jedinki koje predstavljaju roditelje na novu jedinku koja predstavlja dijete. Pri tome jedinka koja predstavlja dijete dobije dio genetskog materijala od jednog roditelja i dio genetskog materijala od drugog roditelja. Glavni cilj je da se takvim kombinacijama dobije jedinka koja ima bolju vrijednost funkcije dobrote od već postojećih. Postoje razni operatori križanja za jedinke prikazane u obliku stabla, a neki od njih će biti opisani u nastavku. Detaljnije o operatorima križanja se može pronaći u [17].

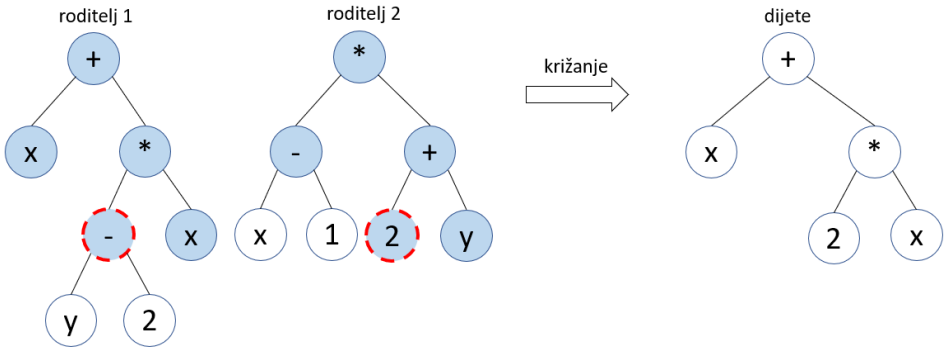
Križanje podstabala (engl. *subtree crossover*) je operator križanja koji se najčešće koristi u GP-u. Za svako roditeljsko stablo ovaj operator slučajnim odabirom i neovisno jedno o drugom odabere jednu točku križanja (engl. *crossover point*). Zatim stvori potomka (dijete) tako da podstablo s korijenom u točki križanja odabranoj u prvom stablu zamijeni sa podstablom kojem je korijen točka križanja odabrana u drugom stablu [17]. Moguće je definirati i slučaj da se umjesto jednog ovim križanjem dobiju dva djeteta, ali ta inačica nije često korištena. Prilikom korištenja ovog operatora, često se uvodi i ograničenje da se za točku križanja u 90% slučajeva odabire funkcijski čvor u stablu, a u preostalih 10% slučajeva čvor koji sadrži značajku kako bi se izbjeglo izmijenjivanje jako malog i nebitnog dijela genetskog materijala [106]. Slika 4.6 prikazuje primjer križanja podstabala. Točke križanja su zaokružene crvenom isprekidanom linijom.

Križanje s jednom točkom prekida (engl. *one-point crossover*) odabire zajedničku točku prekida i zamjenjuje stablo s korijenom u odabranoj točki prekida u prvom roditelju sa stablom s korijenom u odabranoj točki prekida u drugom roditelju [17]. Kako bi se mogla odabrati zajednička točka prekida, stabla roditelji se analiziraju kako bi se otkrila zajednička područja.



Slika 4.6: Prikaz operatora križanja podstabala

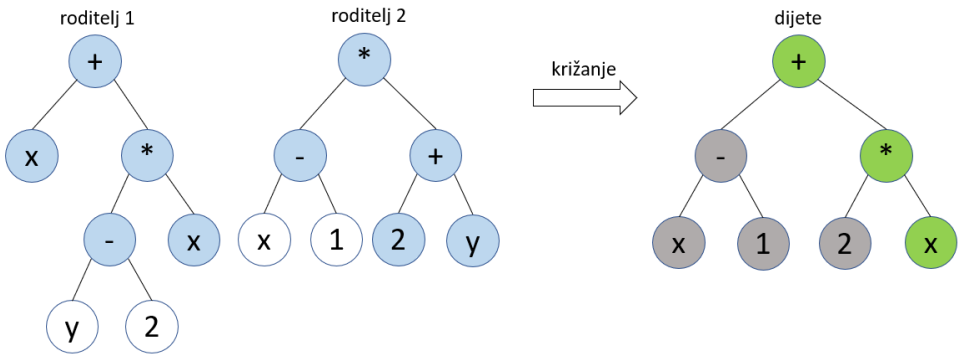
Zajednička područja u stablima su ona u kojima stabla imaju jednak oblik. Slika 4.7 daje prikaz zajedničkog područja za odabrana dva stabla, zajedničke točke prekida i samog postupka križanja. Čvorovi stabala koji pripadaju zajedničkom području su obojani plavom bojom, a zajednička točka prekida je zaokružena crvenom isprekidanom linijom.



Slika 4.7: Prikaz operatora križanja s jednom točkom prekida

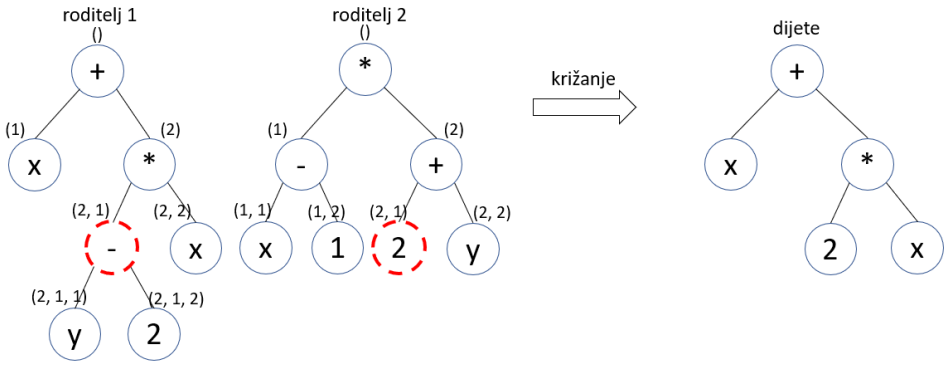
Uniformno križanje (engl. *uniform crossover*) kao i križanje s jednom točkom prekida prvo treba odrediti zajednička područja stabala roditelja [17]. Zatim se prolazi svakim čvorom zajedničkog područja i na slučajan način se odabire hoće li stablo dijete taj čvor naslijediti od prvog ili drugog roditelja. Ukoliko je čvor koji dijete treba naslijediti na rubu zajedničkog područja funkcijski, tada dijete nasljeđuje i cijelo podstablo ukorijenjeno u tom čvoru. Ovaj oblik križanja omogućuje veću raznolikost u prijenosu genetskog materijala nego drugi operatori križanja. Slika 4.8 daje prikaz uniformnog križanja. Čvorovi stabala koji pripadaju zajedničkom području su obojani plavom bojom. U jedinki koja predstavlja dijete, zelenom bojom su označeni čvorovi koji su preuzeti iz prvog roditelja, a sivom bojom čvorovi koji su preuzeti iz drugog roditelja.

Križanje koje čuva kontekst (engl. *context - preserving crossover*) definira koordinate za svaki čvor u stablu u obliku izraza $(b_1, b_2, \cdots, b_n)$ gdje n predstavlja dubinu čvora, a b_i koja je grana odabrana na dubini i kako bi se došlo do trenutnog čvora [148]. Grane se obično numeriraju s lijeva nadesno počevši od broja 1, dok se korijenski čvor označava praznim zagradama. Prilikom odabira točke križanja, mogu se odabrati samo one točke koje imaju jednake koordinate. Nakon odabira točke križanja, križanje se vrši kao i kod križanja podstabala. Slika 4.9



Slika 4.8: Prikaz operatora uniformnog križanja

daje prikaz ovog križanja. Kraj svakog čvora u roditeljskim stablima je naznačeno koja je točno oznaka pojedinog čvora. Za točku križanja je odabrana točka s koordinatama $(2, 1)$.

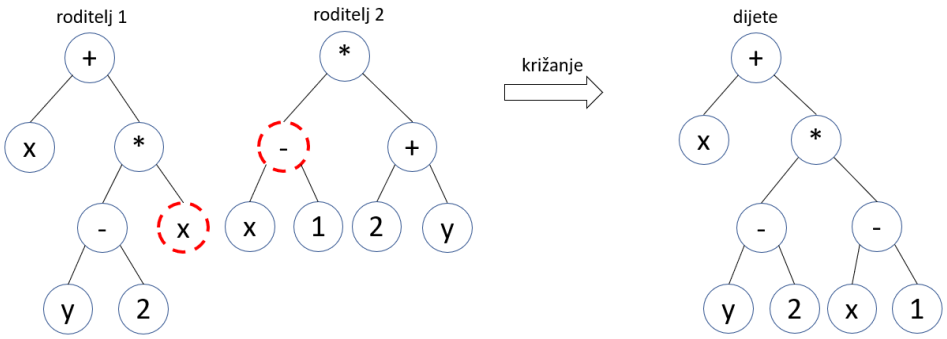


Slika 4.9: Prikaz operatora križanja koje čuva kontekst

Križanje s pravednom veličinom (engl. *size-fair crossover*) je križanje u kojem se prva točka križanja odabire nasumično u prvom roditelju, a zatim se računa veličina podstabla koja se treba ukloniti iz prvog roditelja [149]. Zatim se izračuna koja je najveća veličina podstabla koje smije biti odabrano iz drugog roditelja za preslikati u prvog roditelja. Ukoliko se sa s označi veličina podstabla koje je odabrano za micanje iz prvog roditelja, najveća veličina stabla koje smije biti odabrano iz drugog stabla je $n = 1 + 2 * s$. Takav postupak osigurava da podstablo iz drugog roditelja neće biti preveliko u odnosu na izbačeno podstablo pa se ni novodobivena jedinka neće previše povećati u odnosu na roditelje. Slika 4.10 daje prikaz ovog križanja. Podstablo odabrano u prvom roditelju je veličine 1 pa iz drugog roditelja za križanje smije biti odabrano samo podstablo veličine 3 ili manje.

4.6.2 Operatori mutacije

Cilj operatora mutacije (engl. *mutation operators*) je u nekoj mjeri izmijeniti genetski materijal odabrane jedinke kako bi se povećala raznolikost u populaciji. U prvim inačicama GP-a se mutacija često izostavljala i to je razlog zašto se i danas prilikom primjene GP-a na neki problem,

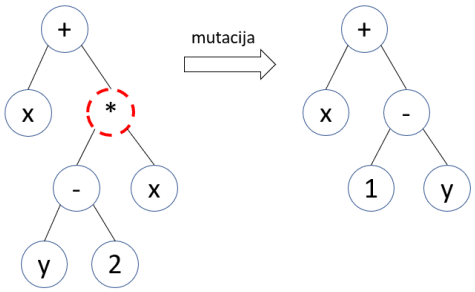


Slika 4.10: Prikaz operatora križanja s pravednom veličinom

mutacija zanemaruje. No autori u [150] su došli do zaključka da relativan učinak operatora križanja i mutacije ovisi i o problemu na koji se GP primjenjuje i o detaljima samog GP sustava. Zbog toga ipak ima smisla koristiti i mutaciju s nekom određenom vjerojatnošću.

Mutacija se najčešće izvodi na jedinkama dobivenim križanjem i to s nekom određenom vjerojatnošću. Korištenjem mutacije se ne dobiju uvijek bolje jedinke, ali ako se koristi samo operator križanja, populacija s vremenom može konvergirati u neki lokalni optimum pa mutacija omogućuje rješenju da se pomakne iz lokalnog optimuma [17]. Kao i kod operatora križanja, i za mutaciju stabala su definirani mnogi operatori, a neki od njih će biti opisani u nastavku.

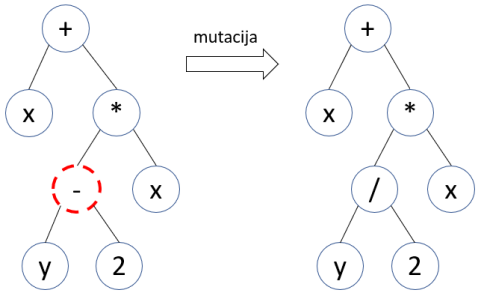
Mutacija podstabla (engl. *subtree mutation*) je najčešće korišten oblik mutacije kod stabala [17]. U ovoj mutaciji se na slučajan način odabire točka mutacije u stablu i zatim se podstablo ukorijenjeno u toj točki zamjenjuje sa slučajno generiranim podstablom. Slika 4.11 daje prikaz ove vrste mutacije. Na slici je čvor koji predstavlja korijen podstabla koje će se zamijeniti slučajno generiranim podstablom, zaokružen crvenom isprekidanom linijom.



Slika 4.11: Prikaz operatora mutacije podstabla

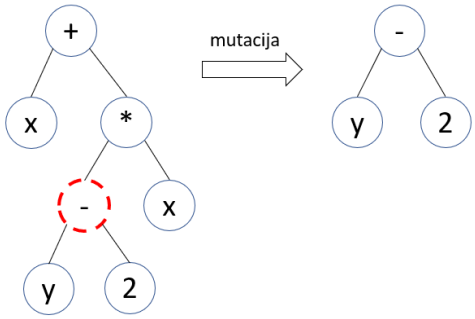
Mutacija zamjenom čvora (engl. *point mutation, node replacement mutation*) na slučajan način odabire neki čvor u stablu i zamjenjuje ga nekim od odgovarajućih primitiva iz skupa primitiva [17]. Ukoliko se u odabranom čvoru nalazi značajka, iz skupa primitiva se odabire neka druga značajka za zamjenu, a ukoliko se u odabranom čvoru nalazi funkcija, iz skupa primitiva se odabire neka druga funkcija koja prima jednak broj argumenata kao i odabrana funkcija. U ovoj mutaciji se ne mijenja samo jedan čvor nego se promjena može s nekom vjerojatnošću dogoditi na svakom od čvorova u stablu. Slika 4.12 daje prikaz mutacije zamjenom čvorova.

Može se vidjeti da je operator − zamijenjen operatorom /.



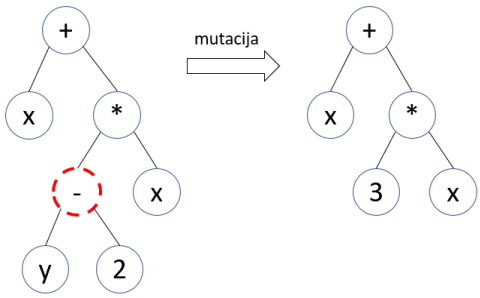
Slika 4.12: Prikaz operatora mutacije zamjenom čvora

Hoist mutacija (engl. *hoist mutation*) odabire na slučajan način podstablo trenutnog stabla i to podstablo postaje nova jedinka [17]. Na ovaj način se u populaciju ne uvodi nikakav novi genetski materijal, ali se veličina jedinke smanjuje što pomaže u smanjenju pretjeranog rasta. Slika 4.13 daje prikaz ove vrste mutacije.



Slika 4.13: Prikaz operatora hoist mutacije

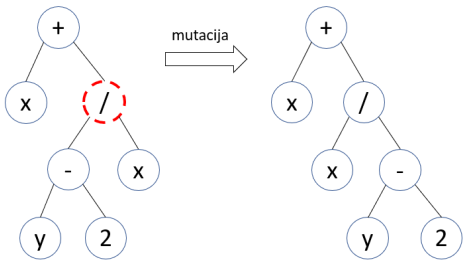
Mutacija smanjivanjem (engl. *shrink mutation*) zamjenjuje slučajno odabrano podstablo samo jednom značajkom [17]. Kao i kod prethodne mutacije, cilj ovog postupka je smanjiti veličinu jedinke i tako smanjiti pretjerani rast (engl. *bloat*). Slika 4.14 daje prikaz mutacije smanjivanjem. Sa slike se može vidjeti da je cijelo podstablo s korijenom − zamijenjeno značajkom koja u sebi sadrži konstantu 3.



Slika 4.14: Prikaz operatora mutacije smanjivanjem

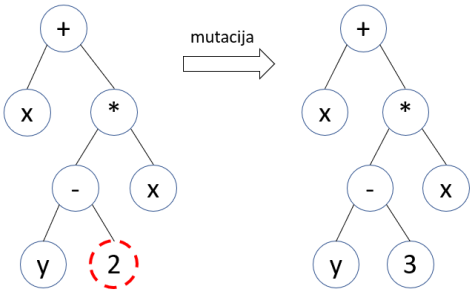
Mutacija permutacijom (engl. *permutation mutation*) odabire funkcijski čvor u stablu i zamjenjuje redoslijed njegovih podstabala [17]. Slika 4.15 daje prikaz mutacije permutacijom.

Čvor koji predtavlja korijen podstabla kojemu će se zamijeniti redoslijed grana je označen crvenom bojom.



Slika 4.15: Prikaz operatora mutacije permutacijom

Gaussova mutacija (engl. *Gauss mutation*) se može primijeniti samo na onim čvorovima koji sadrže neku numeričku konstantu [17]. Ova mutacija odabire jedan od takvih čvorova i na vrijenodts spremljenu u njemu nadoda slučajno generiranu vrijednost iz Gaussove distribucije. Slika 4.16 daje prikaz ove vrste mutacije.

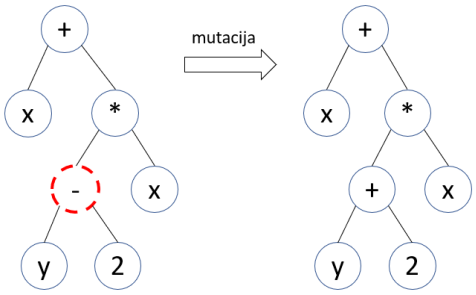


Slika 4.16: Prikaz operatora Gaussove mutacije

Mutacija komplementarnih čvorova (engl. *node complement mutation*) radi po jednakom principu kao i mutacija zamjenom čvora, ali uz dodatna ograničenja. Kod ove vrste mutacije se definiraju međusobno komplementarni čvorovi koji se smiju zamijeniti jedan drugim. Komplementarni čvorovi se mogu odrediti proizvoljno ako korisnik želi, no najčešće se sastoje od čvorova koji predstavljaju komplementarne operacije poput zbrajanja i oduzimanja te množenja i dijeljenja. Prilikom definiranja komplementarnih čvorova treba paziti na to da oni trebaju imati jednak broj argumenata kako bi rezultirajuće stablo i dalje bilo dopustivo. Slika 4.17 daje prikaz ove vrste mutacije. Ovdje je operator $-$ zamijenjen njegovom komplementarnom operacijom, operatorom $+$.

4.7 Kriterij zaustavljanja

Prilikom korištenja heuristika za rješavanje nekog problema, ne postoji garancija da će algoritam završiti u konačnom vremenu niti da će rezultirati optimalnim rješenjem. GP pripada



Slika 4.17: Prikaz operatora mutacije komplementarnih čvorova

skupini heuristika pa prilikom njegova korištenja treba odrediti u kojem slučaju algoritam treba prestati sa izvršavanjem.

Pri tom odabiru treba biti oprezan jer ako algoritam završi prerano, dobiveno rješenje vjerojatno neće biti dobre kvalitete. S druge strane, ako pustimo algoritam da se predugo izvršava, moglo bi doći do prenaučenosti za trenutno promatrani problem pa se gubi mogućnost generalizacije, tj. uspješne primjene evoluiranog rješenja na dosad neviđene (slične) probleme.

Neki od uobičajenih kriterija zaustavljanja su: maksimalan broj iteracija, maksimalan broj generacija, maksimalan broj funkcijskih evaluacija, vrijeme izvršavanja, stagnacija (u nekom određenom broju koraka nije došlo do poboljšanja u funkciji cilja najbolje jedinke), postizanje neke unaprijed zadane vrijednosti, itd.

Poglavlje 5

Skup operatora za promjenu sintaktičkih stabala

Prilikom rješavanja nekog problema korištenjem genetskog programiranja, najčešće se za prikaz jedinke koriste stabla. Takav prikaz je koristan i u problemima raspoređivanja jer se iz njega lako mogu izračunati rješenja promatranog problema.

Ukoliko je potrebno generirati npr. slučajnu šetnju u prostoru pretraživanja gdje su jedinke prikazane stablima, potrebno je definirati mehanizme koji će omogućiti pomak do neke susjedne jedinke. Isto tako, trebalo bi moći definirati kada je udaljenost između neka dva stabla jednaka 1, tj. kada su neka dva stabla prvi susjedi.

Do sada su već definirane razne mjere udaljenosti između dvaju stabala, poput *tree edit distance* [34] [35] [36], ali nijedna od mjera ne uzima u obzir ograničenja koja nameću funkcije koje se uključuju kao parametri genetskog programiranja. Npr. ne može se učiniti zamjena funkcije nekom drugom funkcijom koja nema jednak broj argumenata, jer u tom slučaju rezultirajuće stablo nije dopustivo. Isto tako, nijedna od do sada definiranih mjera udaljenosti stabala ne daje informaciju o tome kako točno odrediti što je susjed na udaljenosti 1 od trenutno promatranog stabla.

Upravo ovi problemi su bili motivacija za definiranje tri nova operatora nad stablima koji će omogućiti kretanje po stablima slučajnom šetnjom. Ovi operatori će omogućiti da se definira što je to točno susjed na udaljenosti 1 od nekog promatranog stabla. Isto tako, uz pomoć predloženih operatora, definira se i heuristika koja može izračunati udaljenost između bilo koja dva proizvoljna stabla, što omogućuje računanje mjera krajolika dobrote poput mjera temeljenih na udaljenosti.

U nastavku će biti opisani operatori umetanja (engl. *insert*), zamjene (engl. *edit*) i brisanja (engl. *delete*) koji uzimaju u obzir vrstu čvorova u stablima (funkcija ili značajka), kao i broja djece u funkcijskim čvorovima.

U opisima svih operatora pretpostavlja se da su stabla zapisana u prefiksnoj notaciji. Ope-

ratori su definirani za predefinirani podskup funkcija koji se koristi u problemima u ovom radu, ali mogu se lako proširiti na dodatne funkcije koristeći ista pravila. Tablica 5.1 sadrži nazive i opise korištenih funkcija.

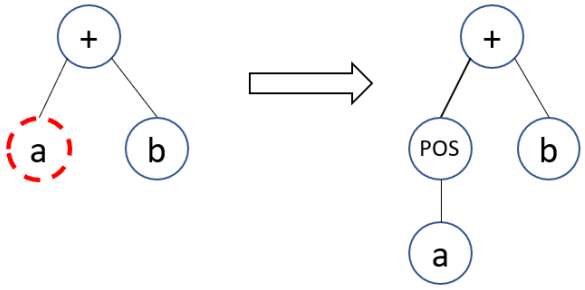
Tablica 5.1: Podskup korištenih funkcija

Funkcija	**Opis**
+, -, *	zbrajanje, oduzimanje i množenje
/	Zaštićeno dijeljenje: $DIV(a,b) = \begin{cases} 1, & \lvert b\rvert < 0.000000001 \\ \frac{a}{b}, & inače. \end{cases}$
MAX	$MAX(a) = \begin{cases} a, & a > 0 \\ 0, & inače. \end{cases}$
POS	$POS(a) = \begin{cases} a, & a > 0 \\ -a, & inače. \end{cases}$
NEG	$NEG(a) = -a$
IF	$IF(a,b,c) = \begin{cases} b, & a > 0 \\ c, & inače. \end{cases}$

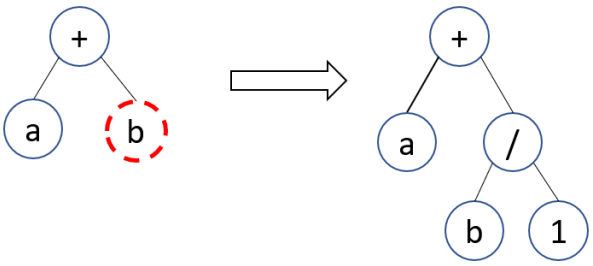
5.1 Operator umetanja (*insert*)

Operator umetanja dodaje novi čvor na proizvoljno mjesto u stablu. Ovaj operator dozvoljava umetanje samo funkcijskog čvora. Razlog tomu je što ako se doda značajka (terminal), dopustivost (engl. *feasibility*) stabla će biti narušena i ne postoji neki očigledan način kako popraviti dobiveno stablo da bi ponovno postalo dopustivo. Procedura umetanja čvora je sljedeća: odabere se neki čvor u stablu ispred kojeg će se dodati novi čvor. Ukoliko se umeće čvor koji sadrži funkciju koja ima samo jedno dijete, onda se novi čvor samo zapiše ispred odabranog. Primjer za ovaj slučaj se može vidjeti na slici 5.1, gdje je *POS* funkcija koja ima samo jedno dijete. Čvor ispred kojeg se umeće funkcija *POS* je označen crvenom bojom. U prefiksnoj notaciji ovo umetanje ima oblik: `+ a b` $\rightarrow$ `+ POS a b`.

Ukoliko čvor koji se umeće sadrži funkciju koja ima dva djeteta, novi funkcijski čvor se zapiše ispred odabranog i zatim, ovisno o vrijednosti novog čvora, 0 ili 1 se zapisuje iza sve djece odabranog čvora. Ukoliko je novi funkcijski čvor $+$ ili $-$, tada se 0 zapisuje iza djece odabranog čvora jer je 0 neutralni element za zbrajanje i oduzimanje. Ako je novi funkcijski čvor $*$ ili $/$, tada se iza djece odabranog čvora zapisuje 1, jer je 1 neutralni element za množenje i dijeljenje. Primjeri umetanja čvora koji sadrži funkciju koja ima dva elementa se mogu vidjeti na slikama 5.2 i 5.3. Kao i u prethodnom primjeru, odabrani čvor ispred kojeg se treba



Slika 5.1: Umetanje funkcije koja ima jedno dijete

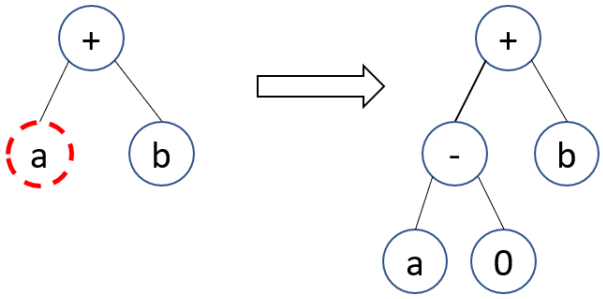


Slika 5.2: Umetanje čvora /

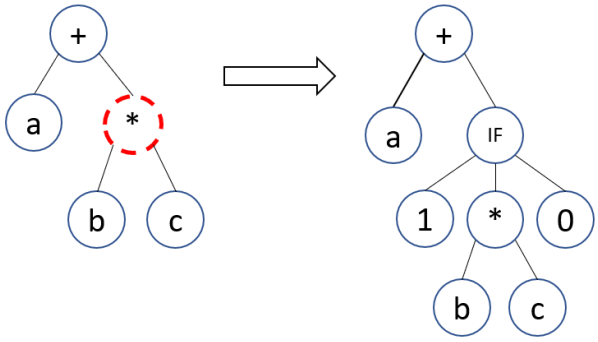
umetnuti novi čvor, je označen crvenom bojom. Odgovarajući zapisi u prefiksnoj notaciji su `+ a b` $\rightarrow$ `+ a / b 1` za prvi slučaj i `+ a b` $\rightarrow$ `+ - a 0 b` za drugi slučaj.

Ukoliko se umeće čvor koji sadrži funkciju *IF* (jedina funkcija u skupu odabranih koja ima 3 djeteta), odabrani čvor i sva njegova djeca se stavljaju kao srednja grana novog čvora, dok se s lijeve strane stavlja 1, a s desne 0. Primjer se može vidjeti na slici 5.4. Čvor ispred kojeg se stavlja funkcija *IF* je označen crvenom bojom. Umetanje sa slike se može prikazati kao `+ a * b c` $\rightarrow$ `+ a IF 1 * b c 0` u prefiksnom zapisu.

Dodatno ograničenje ovog operatora umetanja je da se funkcijski čvorovi ne mogu umetati kao listovi stabla.



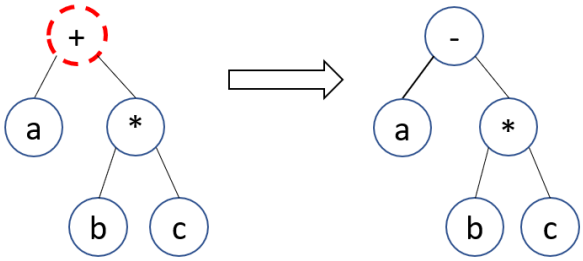
Slika 5.3: Umetanje čvora –



Slika 5.4: Umetanje funkcije koja ima tri djeteta

5.2 Operator zamjene (*edit*)

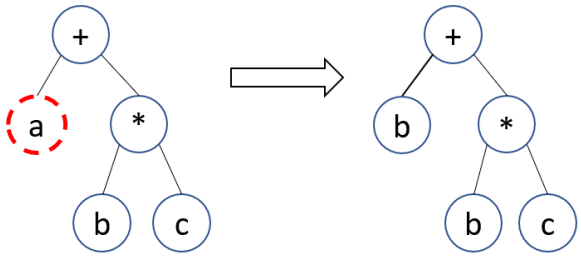
Operator zamjene mijenja odabrani čvor s nekim od dozvoljenih čvorova. Prvo se odabere jedan čvor u trenutnom stablu. Ukoliko odabrani čvor sadrži funkciju, može se zamijeniti drugom funkcijom, ali samo ako nova funkcija ima jednak broj djece kao ona u odabranom čvoru. Iz odabranog skupa funkcija, funkcije *MAX*, *POS* i *NEG* se mogu međusobno mijenjati jer imaju po jedno dijete. Isto tako, funkcije $+$, $-$, $*$ i $/$ se mogu međusobno mijenjati jer imaju po dva djeteta. Ukoliko odabrani čvor sadrži značajku, a ne funkciju, tada se može zamijeniti bilo kojom drugom značajkom iz unaprijed definiranog skupa značajki. Slika 5.5 prikazuje operator zamjene ukoliko odabrani čvor sadrži funkciju, a slika 5.6 prikazuje operator zamjene ukoliko odabrani čvor sadrži značajku. U oba prikaza čvor koji je odabran za zamjenu u početnom stablu je označen crvenom bojom. Odgovarajući prikazi u prefiksnoj notaciji su dani s `+ a * b c` $\rightarrow$ `- a * b c` za sliku zamjene funkcije, odnosno `+ a * b c` $\rightarrow$ `+ b * b c` za sliku zamjene značajke.



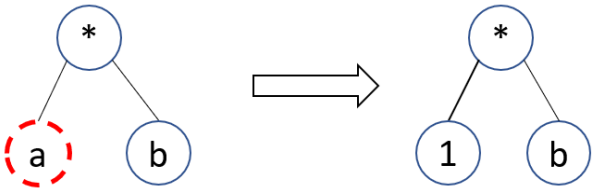
Slika 5.5: Zamjena čvora koji sadrži funkciju ($+$ u $-$)

5.3 Operator brisanja (*delete*)

Operator brisanja može obrisati bilo koji čvor osim korijenskog. Ukoliko bi se obrisao korijenski čvor, tada više ne bismo imali nikakvo stablo, što ne bi imalo smisla. Prilikom brisanja, prvo se odabere čvor u trenutnom stablu koji će se izbrisati. Ukoliko je odabran čvor koji sadrži



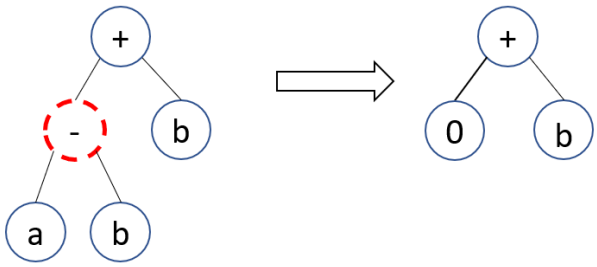
Slika 5.6: Zamjena čvora koji sadrži značajku (*a* u *b*)



Slika 5.7: Brisanje čvora koji sadrži značajku (*a*)

značajku, on se briše, a umjesto njega se stavlja čvor koji sadrži 0 ili 1 kako bi se očuvala dopustivost stabla. Ako je roditelj odabranog čvora $+$, $-$, *IF* ili bilo koji od funkcijskih čvorova koji imaju samo jedno dijete, onda se umjesto obrisanog čvora stavlja 0. Ukoliko je roditelj odabranog čvora $*$ ili $/$, umjesto obrisanog čvora se stavlja 1. Primjer brisanja čvora koji sadrži značajku dan je na slici 5.7. Čvor koji je odabran za brisanje je označen crvenom bojom. Odgovarajući prikaz ovog brisanja u prefiksnoj notaciji je `* a b` $\rightarrow$ `* 1 b`.

Ukoliko čvor koji je odabran za brisanje sadrži funkciju, taj čvor i sva njegova djeca se brišu iz stabla. Nakon toga, na mjesto odabranog čvora se stavlja 0 ili 1 po istim pravilima kao kad se briše čvor koji sadrži značajku. Primjer brisanja čvora koji sadrži funkciju dan je na slici 5.8. Čvor koji je odabran za brisanje je označen crvenom bojom. Odgovarajući prikaz ovog brisanja u prefiksnoj notaciji je `* - a b b` $\rightarrow$ `+ 0 b`. Ako je za brisanje odabran čvor koji sadrži funkciju sa samo jednim djetetom, onda se briše samo ta funkcija, tj. čvor dijete zamjenjuje obrisani čvor.



Slika 5.8: Brisanje čvora koji sadrži funkciju ($-$)

5.4 Heuristika za određivanje udaljenosti između stabala

U praksi je važno imati odgovarajuć način za mjerenje udaljenosti između bilo koja dva stabla kako bi se mogle izračunati određene značajke krajolika dobrote. Heuristika koja to omogućuje je opisana u nastavku.

Za bilo koja dva stabla zapisana u prefiksnoj notaciji, stablo sa više čvorova se označava kao *stablo*1, a drugo stablo kao *stablo*2 (ukoliko stabla imaju jednak broj čvorova, onda bilo koje stablo može biti *stablo*1). Ta dva stabla će biti ulaz za heuristiku čiji je pseudokod zapisan u Algoritmu 2.

Algoritam 2 Pseudokod heuristike za računanje udaljenosti između stabala

```
1:  udaljenost ← 0
2:  ako je stablo1 == stablo2
3:      vrati udaljenost
4:  ako je stablo1[0] ≠ stablo2[0]
5:      ako je broj djece od stablo1[0] == broj djece od stablo2[0]
6:          ZAMIJENI stablo1[0] sa stablo2[0]
7:      u suprotnom
8:          UMETNI stablo2[0] kao korijen od stablo1
9:      udaljenost ← 1
10: ponavljaj za i ← 1 do duljine od stablo1
11:     ako je stablo1[i] ≠ stablo2[i]
12:         ako je stablo1[i] značajka i stablo2[i] značajka
13:             ZAMIJENI stablo1[i] sa stablo2[i]
14:         ako je stablo1[i] funkcija i stablo2[i] funkcija
15:             ako je broj djece od stablo1[i] == broj djece od stablo2[i]
16:                 ZAMIJENI stablo1[i] sa stablo2[i]
17:             u suprotnom
18:                 UMETNI stablo2[i] kao i-ti čvor u stablo1
19:         ako je stablo1[i] značajka i stablo2[i] funkcija
20:             UMETNI stablo2[i] kao i-ti čvor u stablo1
21:         ako je stablo1[i] funkcija i stablo2[i] značajka
22:             OBRIŠI stablo1[i]
23:             i ← i-1
24:         udaljenost ← udaljenost + 1
25: vrati udaljenost
```

Heuristika kreće od korijena oba stabla i uspoređuje ih. Ukoliko su različiti, primijeni odgovarajući operator i poveća udaljenost za jedan. Zatim uspoređuje svaki sljedeći par čvorova i u ovisnosti o tome nalaze li se u njima funkcije ili značajke, primjenjuje jedan od tri uvedena operatora. Specifičan je slučaj ako se u prvom stablu na mjestu koje se trenutno promatra nalazi funkcija, a na drugom značajka. U tom slučaju funkciju i svu djecu te funkcije treba obrisati pa se onda 0 ili 1 koji se nađu na mjestu te funkcije (po pravilima operatora brisanja) treba usporediti sa značajkom koja se nalazi na promatranom mjestu u drugom stablu. Iz tog razloga

se iterator smanjuje za jedan, kako bi se promatrani čvor koji se brisanjem promijenio, ponovno usporedio za odgovarajućim čvorom iz drugog stabla. Svaki put kada se mora primijeniti jedan od operatora, udaljenost se povećava za 1 i na kraju se dobija ukupna udaljenost između dva proslijeđena stabla.

Ovaj opisani postupak je heuristika jer ne garantira da će se njegovom uporabom doći do optimalnog rješenja, odnosno do najmanje udaljenosti između zadana dva stabla. Ali, ovako definiran postupak će uvijek na isti način računati udaljenost između dva proizvoljna stabla pa je iz tog razloga pogodan za uporabu prilikom uspoređivanja udaljenosti između zadanih stabala.

U nastavku je dan primjer rada navedene heuristike na dva proizvoljna stabla. Funkcije koje se koriste u primjeru su: $+$,$/$,$*$,*POS*, a značajke: *RS*, *D*, *SPD*, *TSC*, *RR*. U svakom koraku primjera, par čvorova koji se uspoređuje je podvučen i označen podebljanim fontom.

S1: **/** + RS TSC + POS RS POS * D RR
S2: ***** POS / RS D + POS SPD RS
ZAMIJENI / sa * $\rightarrow$ udaljenost = 1

S1: * **+** RS TSC + POS RS POS * D RR
S2: * **POS** / RS D + POS SPD RS
UMETNI POS ispred + $\rightarrow$ udaljenost = 2

S1: * POS **+** RS TSC + POS RS POS * D RR
S2: * POS **/** RS D + POS SPD RS
ZAMIJENI + sa / $\rightarrow$ udaljenost = 3

S1: * POS / **RS** TSC + POS RS POS * D RR
S2: * POS / **RS** D + POS SPD RS
jednake značajke u čvorovima $\rightarrow$ udaljenost = 3

S1: * POS / RS **TSC** + POS RS POS * D RR
S2: * POS / RS **D** + POS SPD RS
ZAMIJENI TSC sa D $\rightarrow$ udaljenost = 4

S1: * POS / RS D **+** POS RS POS * D RR
S2: * POS / RS D **+** POS SPD RS
jednake funkcije u čvorovima $\rightarrow$ udaljenost = 4

S1: * POS / RS D + **POS** RS POS * D RR
S2: * POS / RS D + **POS** SPD RS
jednake funkcije u čvorovima $\rightarrow$ udaljenost = 4

S1: * POS / RS D + POS **RS** POS * D RR
S2: * POS / RS D + POS **SPD** RS

ZAMIJENI RS sa SPD → udaljenost = 5

S1: * POS / RS D + POS SPD **POS** * D RR
S2: * POS / RS D + POS SPD **RS**
OBRIŠI POS → udaljenost = 6

S1: * POS / RS D + POS SPD ***** D RR
S2: * POS / RS D + POS SPD **RS**
OBRIŠI * → udaljenost = 7

S1: * POS / RS D + POS SPD **0**
S2: * POS / RS D + POS SPD **RS**
ZAMIJENI 0 sa RS → udaljenost = 8

Može se vidjeti da je potrebno 8 koraka kako bi se iz stabla1 prešlo u stablo2 pa je udaljenost između ova dva stabla jednaka 8.

Poglavlje 6

Primjena analize krajolika dobrote u svrhu određivanja prikladnih parametara za genetsko programiranje

Glavna ideja ovog rada je iskoristiti značajke krajolika dobrote kako bi se instance problema raspoređivanja grupirale u slične klastere i kako bi se onda za te klastere odredili najbolji parametri za GP koji rješava spomenute probleme.

U ovom poglavlju će biti dan pregled dosadašnjih radova slične tematike, bit će opisan postupak oblikovanja primjera za učenje i testiranje, zatim će biti opisan postupak grupiranja i prikazani odgovarajući rezultati te će se na kraju pokazati rezultati koji se dobiju kada se za svaku od dobivenih grupa odrede parametri za GP.

6.1 Pregled literature

U literaturi se dosta pozornosti posvećuje istraživanju krajolika dobrote evolucijskih algoritama i metaheuristika općenito [151] [152] [153] [154]. No, za hiperheuristike poput genetskog programiranja se krajolik dobrote nije toliko istraživao.

U radu [22] promatraju se dvije klase problema koje se žele riješiti korištenjem GP-a. U prvoj klasi problema trebaju se evoluirati algoritmi sortiranja za bilo koji niz brojeva, dok se u drugoj klasi problema promatraju problemi jednake parnosti (engl. *Boolean even parity problems*). Za obje klase problema se definira slučajna šetnja duljine 1000 koraka i promatra se autokorelacija dobivenih šetnji. Dobivene vrijednosti autokorelacije za promatrane probleme se dovode u korelaciju sa težinom rješavanja promatranih problema korištenjem GP-a. Zatim se istražuju adaptivne šetnje i promatra se koji je prosječan broj koraka potreban za postizanje optimalnog rješenja. Taj broj koraka se opet dovodi u korelaciju s težinom rješavanja problema, ali u manjoj mjeri nego vrijednosti autokorelacije.

Vanneschi i suradnici u nizu radova [155] [156] [157] [158] [159] istražuju krajolik dobrote genetskog programiranja. U [155] se definira koeficijent negativnog spusta (engl. *negative slope coefficient*) kao mjera koja može pomoći u karakteriziranju krajolika dobrote GP-a, odnosno koliko je teško riješiti neki problem korištenjem GP-a. Koeficijent je testiran na nekoliko primjera poput problema jednake parnosti i problema umjetnog mrava (engl. *the artificial ant on the Santa Fe trail*) i postigao je dobru korelaciju između vrijednosti koeficijenta i težine rješavanja danog problema korištenjem GP-a. U [156] se koristi udaljenost između stabala temeljena na križanju podstabala kako bi se dobio izgled krajolika dobrote GP-a u kojemu su jedinke prikazane stablima. Ova udaljenost zapravo nije prava mjera udaljenosti, nego se promatra kao vjerojatnost da se iz nekog stabla T_1 može doći do stabla T_2 zamjenom podstabla iz T_1 nekim podstablom iz cijele populacije kako bi se došlo do stabla T_2. Radovi [157] [158] se bave istraživanjem neutralnosti u krajolicima dobrote za binarni problem parnosti koji se rješava uporabom GP-a. U tu svrhu su definirane tri nove mjere i novi genetički operatori koji omogućuju definiranje susjedstva u ovom prikazu. I u ovim radovima se dovodi u korelaciju težina rješavanja problema GP-om i vrijednosti izračunatih mjera. U radu [159] se također istražuju mjere neutralnosti, ali za problem multipleksora i izračunate mjere se uspoređuju sa koeficijentom negativnog spusta.

Moraglio i suradnici su u [160] uveli pojam geometrijskog semantičkog genetskog programiranja (engl. *geometric semantic genetic programming*) u kojemu se umjesto sintaktičkog prostora stabala pretražuje semantički prostor dobivenih programa, odnosno pretražuje se prostor vrijednosti dobivenih izvršavanjem dobivenih programa. Pokazalo se da je krajolik dobrote operatora koji rade na semantičkom prostoru uvijek unimodalan za probleme u kojima na temelju danih ulaza treba pronaći funkcija koja najbolje opisuje dane izlaze.

Ochoa i suradnici su u [32] i [33] definirali što je to krajolik dobrote hiperheuristike. U radu [32] je opisan hiperheuristički pristup za rješavanje hibridnog problema raspoređivanja u obradi tijeka. Autori navode da ovdje postoje dva prostora pretraživanja. Prvi je prostor pretraživanja heuristika (engl. *heuristic space*, HS), a drugi je prostor pretraživanja rješenja problema. Unatoč tome, na kraju se dobija samo jedan krajolik dobrote. Naime, u prostoru pretraživanja heuristika vrijednost funkcije dobrote može se izračunati tek nakon što se odabrana (ili generirana) heuristika primijeni na problem te se evaluira odgovarajuća točka u prostoru pretraživanja rješenja. Za potrebe formalne definicije krajolika dobrote hiperheuristike, definiraju se dva preslikavanja, g iz liste heuristika u odgovarajuće rješenje i f iz odgovarajućeg rješenja u vrijednost funkcije dobrote u tom rješenju. Tada se funkcija cilja na heurističkom prostoru može definirati kao kompozicija od f i g, odnosno, $f(g(\hat{A})) : HS \rightarrow \mathbb{R}$, gdje je $\hat{A}$ niz heuristika. Uz ovako definiranu funkciju cilja, hiperheuristički krajolik je trojka $(HS, f(g), V)$, gdje je sa V označeno susjedstvo inducirano operatorom najmanjeg mogućeg pomaka u prostoru pretraživanja heuristika. U radu [33] se koristi isti pristup, samo se umjesto susjedstva V koristi Hammin-

gova udaljenost jer se radi o problemu raspoređivanja nastavnih aktivnosti (engl. *educational timetabling problem*) koji se rješava uporabom hiperheurističkog pristupa baziranog na grafu (engl. *graph - based hyper-heuristic*). U prvom radu se radi o unaprjeđivačkoj heuristici, dok se u drugom radu promatra konstruktivna hiperheuristika, no u oba slučaja su dobivene strukture krajolika dobrote veoma slične.

U domeni grupiranja se javljaju neki radovi koji koriste značajke krajolika dobrote kako bi grupirali instance problema i na taj način olakšali pronalazak rješenja algoritmima koji se koriste za rješavanje danih problema.

Rad [161] se bavi metodom hijerarhijskog grupiranja koje omogućuje grupiranje prostora pretraživanja tako da algoritmi koji dobro rade samo na unimodalnim funkcijama mogu, koristeći nastale grupe, riješiti i multimodalan problem. Prilikom definiranja grupa, koriste se vrijednosti funkcije dobrote kako bi se odredili predstavnici grupa. Autori se u [162] također bave grupiranjem podataka oko potencijalnih lokalnih optimuma, samo se u ovom radu koristi rekurzivno usrednjenje (engl. *recursive middling*) za određivanje predstavnika u grupama. Rad [163] predlaže grupiranje temeljeno na brdima i dolinama (engl. *hill - valley clustering*) koje se prilagođava multimodalnosti krajolika dobrote. U ovom grupiranju se grupe stvaraju tako da svaki lokalni optimum pripada jednoj grupi, pa se onda svaka grupa posebno optimizira nekim odabranim algoritmom.

Nadalje, s obzirom da se kod korištenja GP-a prije samog pokretanja trebaju odrediti parametri poput veličine populacije, broja generacija, vjerojatnosti mutacije, maksimalne dubine stabla i slično, razvijaju se pristupi za automatsko određivanje parametara algoritma koji će proizvesti bolja rješenja u konačnici. Neki od radova koji se bave ovim problemom navedeni su u nastavku.

Autori u [18] opisuju metodologiju za određivanje parametara za GP koristeći faktorski dizajn (engl. *factorial design*), jednofaktorski dizajn (engl. *one-factor design*) i multilinearnu regresiju (engl. *multiple linear regression*). Pokazuje se da se faktorski dizajni mogu koristiti kako bi se odredilo koji parametri imaju najveći utjecaj na izvršavanje algoritma. Kada se odredi koji su to parametri, prostor pretraživanja parametara se može znatno smanjiti ako se proces određivanja parametara fokusira samo na najbitnije parametre. Autori u [164] istražuju krajolik dobrote samog problema pretraživanja dobrih parametara. Istražuju se značajke krajolika dobrote problema određivanja parametara za optimizaciju algoritmom roja čestica i dolazi se do zaključka da je u ovom slučaju krajolik konfiguracije parametara unimodalan, ali ga ne mora nužno biti jednostavno pretražiti. Rad [165] istražuje kako odrediti dobre lokalne strategije za rješavanje problema obrade tijeka. U samom procesu određivanja koriste se značajke krajolika dobrote kako bi se instance problema mogle razlikovati. Dolazi se do zaključka da osim odabiranja dobre strategije, u nekim se slučajevima može odabrati i dobra konfiguracija za odabrani algoritam koja će dovesti do boljeg rješenja.

Razvijaju se i razna okruženja koja se mogu primijeniti za određivanje parametara za odabrane algoritme. Autori su u [166] definirali *ParamILS* - okruženje za automatsko određivanje parametara za algoritam koje se temelji na iterativnoj lokalnoj pretrazi (engl. *iterated local search*) konfiguracije parametara. Ova procedura kreće od jedne konfiguracije parametara, radi lokalno pretraživanje kako bi došla do lokalnog optimuma i s nekom vjerojatnošću radi perturbaciju dobivene konfiguracije kako bi izašla iz tog lokalnog optimuma i nastavila pretraživanje sve dok se ne postigne zadani kriterij zaustavljanja. Autori su pokazali da ova metoda dobro funkcionira za rješavanje problema CPLEX-om u kojemu je potrebno odrediti 80-ak parametara prije samog rješavanja. Rad [167] uvodi novu metodu za konfiguraciju algoritama za instance problema (engl. *instance - specific algorithm configuration*). U ovom pristupu se za svaku instancu posebno promatra koja kombinacija parametara će u danom algoritmu proizvesti najbolje rješenje za promatranu instancu problema. Prilikom rada, instance se grupiraju u slične grupe koristeći algoritam k-means, ali u kojemu se broj grupa određuje automatski, i zatim se za svaku od tih grupa traže najbolji parametri kako bi se ipak postigla određena mogućnost generalizacije. Autori su pokazali da ovaj pristup daje dobre rezultate prilikom rješavanja niza problema kombinatorne optimizacije raznim algoritmima. I autori u [168] se bave pronalaženjem najboljih parametara za željeni algoritam, ali koristeći iterativno uzorkovanje (engl. *iterated racing*) i na temelju metode, okruženje je nazvano *irace*. Iterativno uzorkovanje se sastoji od tri koraka koji se ponavljaju dok se ne postigne neki kriterij zaustavljanja: obavlja se uzorkovanje konfiguracija parametara na temelju određene distribucije, zatim se odabire najbolja konfiguracija na temelju utrkivanja i na kraju se distribucija za uzorkovanje prilagodi trenutno najboljim parametrima kako bi više u obzir uzimala bolje konfiguracije. Autori su pokazali da ovaj pristup daje dobre rezultate na primjeru algoritma kolonije mrava za rješavanje problema trgovačkog putnika, kao i za problem višekriterijske optimizacije na problemu mjerenja hipervolumena.

Autori u [26] koriste analizu krajolika dobrote kako bi poboljšali proces automatskog pretraživanja prostora parametara. Pristup se istražuje na NK krajolicima u kojima se koristi memetički algoritam za koji treba odrediti veličinu populacije, operator križanja, vjerojatnost križanja i vjerojatnost mutacije. Za sve probleme se određuje slučajna šetnja iz koje se zatim računa prosječna dobrota jedinki iz šetnje, autokorelacija i stopa neutralnih susjeda. Na temelju izračunatih značajki se instance problema raspoređuju u grupe i za svaku od tih grupa se pomoću okruženja *irace* određuju najbolji parametri za memetički algoritam. Ovaj rad se temelji na idejama iz [26], ali je proširen na probleme raspoređivanja koji se rješavaju genetskim programiranjem i istražen je veći broj značajki krajolika dobrote.

6.2 Oblikovanje ispitnih primjera

U ovom radu se genetsko programiranje koristi kako bi se riješili problemi raspoređivanja, a značajke krajolika dobrote se koriste kako bi se instance problema grupirale u slične grupe i kako bi se zatim odredili optimalni parametri za GP u svakoj od grupa. S obzirom da je GP tehnika strojnog učenja, potrebno je definirati skupove za učenje i za testiranje kako bi se mogla provjeriti naučena pravila raspoređivanja.

6.2.1 Oblikovanje ispitnih primjera u okolini jednog stroja

Za okolinu jednog stroja definirano je 100 instanci u skupu za učenje (treniranje) i 600 instanci u skupu za testiranje kao u [14]. Za svaku instancu definira se trajanje svakog posla, težina posla i željeno vrijeme završetka. Težine poslova dolaze iz skupa vrijednosti 0.01 do 1 u koracima 0.01 i biraju se uniformno. Trajanje posla dolazi iz skupa vrijednosti od 1 do 100 i za instance iz skupa za učenje se dobiju iz jednolike (uniformne) razdiobe na intervalu $[1,100]$. Za instance iz skupa za testiranje se vrijednosti trajanja poslova generiraju iz tri raspodjele: 20% vrijednosti dolazi iz jednolike razdiobe, 50% iz normalne (Gaussove), a 30% iz kvazi-bimodalne razdiobe. Dodatno su definirana dva parametra, R - područje zaostajanja (engl. *due date range*) i T - postotak zaostajanja (engl. *due date tightness*) pomoću kojih se računaju željena vremena završetka poslova. Oba parametra vrijednosti primaju iz intervala [0, 1]. Parametar T zadaje očekivani postotak zakašnjelih poslova dok parametar R određuje širinu intervala vrijednosti koje mogu poprimiti željena vremena završetka. Za svaku instancu problema, željena vremena završetka se dobiju jednolikom razdiobom iz intervala $d_j \in [\sum_{j=1}^{n} p_j(1-T-R/2), \sum_{j=1}^{n} p_j(1-T+R/2)]$ gdje je sa n označen broj poslova u ispitnom primjeru i dodatno se vrijednosti d_j ograničavaju da ne smiju biti manje od nule. Za svaku instancu problema je određen broj poslova i vrijednosti parametara T i R. Brojevi poslova poprimaju vrijednosti 12, 25, 50 i 100, dok parametri T i R poprimaju vrijednosti 0.2, 0.4, 0.6, 0.8 i 1 u raznim kombinacijama.

Kako bi se mogla evaluirati stabla koja predstavljaju prioritetno pravilo razvijeno korištenjem genetskog programiranja, potrebno je definirati i funkciju dobrote, odnosno optimizacijski kriterij, kao i način na koji se iz prioritetnog pravila generira raspored. U ovim eksperimentima se minimizira ukupno težinsko zaostajanje (engl. *total weighted tardiness*) $\sum w_j T_j$. Za računanje ukupnog težinskog zaostajanja, potrebno je imati raspored poslova u instanci, a on se dobija sljedećom shemom generiranja rasporeda: čekaj dok stroj ne postane dostupan, izračunaj prioritete svih neraspoređenih poslova (koristeći prioritetno pravilo razvijeno pomoću GP-a), odaberi posao s najboljim prioritetom i rasporedi ga na stroj. Ovaj postupak se ponavlja sve dok postoje neraspoređeni poslovi.

Konačno, kako bi se prioritetna pravila mogla generirati korištenjem genetskog programiranja, potrebno je definirati skup značajki i funkcija koje će se proslijediti GP-u kako bi od

njih mogao stvarati stabla. Značajke mogu biti statičke (njihova vrijednost je nepromijenjena za cijelo vrijeme izvršavanja sustava) i dinamičke (njihova vrijednost se mijenja u ovisnosti o trenutnom stanju aktivnosti i sredstava). Tablica 6.1 prikazuje funkcije i značajke koje se koriste u okruženju jednog stroja.

Tablica 6.1: Podskup korištenih funkcija i značajki za okruženje jednog stroja

Funkcija	**Opis**
+	zbrajanje
-	oduzimanje
*	množenje
/	zaštićeno dijeljenje: $DIV(a,b) = \begin{cases} 1, & \lvert b\rvert < 0.000000001 \\ \frac{a}{b}, & inače. \end{cases}$
POS	$POS(a) = \begin{cases} a, & a > 0 \\ 0, & inače. \end{cases}$
Značajka	**Opis**
pt	trajanje izvršavanja posla j na stroju i (p_{ij})
dd	željeno vrijeme završetka (d_j)
w	težina (w_j)
N	ukupan broj poslova
Nr	broj poslova koji nisu raspoređeni
SP	zbroj trajanja svih poslova
SPr	zbroj trajanja neraspoređenih poslova
SD	zbroj željenih vremena završetka svih poslova
SL	pozitivna dopuštena odgoda ($max\{d_j - p_j - time, 0\}$)

Značajke *pt*, *dd* i *w* daju bitne informacije o poslovima koji se trebaju rasporediti, pa je logično da su one uključene u skup značajki. Značajke *SP* i *SD* daju korisne informacije o stanju cijelog sustava. Značajke *Nr* i *SPr* su dodane u skup značajki jer se u svakom koraku prioriteti računaju iznova, pa povijest izvršavanja nema utjecaja na odluku o raspoređivanju sljedećeg posla. Značajka *SL* daje informaciju o tome koliko dugo posao može čekati prije nego što se rasporedi, a da ne prekorači svoje željeno vrijeme završetka. *SL* se može gledati kao neka vrsta hitnosti kojom se posao mora rasporediti. Uz navedene značajke, koriste se još i vrijednosti 0 i 1.

Dodatno, kod okoline jednog stroja vrijede sljedeće pretpostavke: sve informacije o poslo-

vima su poznate unaprijed, jednom kada posao započne, nije dopušteno njegovo prekidanje i stroj je neprekidno raspoloživ, odnosno nema kvarova.

6.2.2 Oblikovanje ispitnih primjera u okolini nesrodnih strojeva

Za okolinu nesrodnih strojeva, definirano je 60 primjera u skupu za učenje i 60 primjera u skupu za testiranje kao u [169]. Brojevi poslova za oba skupa poprimaju vrijednosti 12, 25, 50 ili 100, dok se broj strojeva kreće u vrijednostima 3, 6 ili 10. U oba skupa je generirano po 5 instanci za svaku kombinaciju broja poslova i broja strojeva. Za sve poslove se trajanje generira iz intervala $[1,100]$ koristeći uniformnu, normalnu i kvazi bimodalnu razdiobu, kao i kod okoline jednog stroja. Jedina razlika je što u ovoj okolini svaka razdioba ima vjerojatnost $1/3$ da bude odabrana za generiranje trajanja posla. Težine poslova dolaze iz jednolike razdiobe na intervalu $[0,1]$. U ovoj okolini novi poslovi mogu dolaziti i za vrijeme izvršavanja sustava, pa je potrebno definirati i vrijeme u kojem pojedini posao postaje dostupan za izvršavanje. Te vrijednosti se generiraju jednolikom razdiobom iz intervala $[0,\hat{p}/2]$, gdje je $\hat{p}$ definiran kao: $\hat{p} = \frac{\sum_{j=1}^{n}\sum_{i=1}^{m} p_{ij}}{m^2}$, gdje je sa p_{ij} označeno trajanje izvršavanja posla j na i - tom stroju, m je broj strojeva, a n broj poslova. Za svaku instancu problema, željena vremena završetka se dobiju jednolikom razdiobom iz intervala $d_j \in [r_j + (\hat{p} - r_j)\cdot(1 - T - R/2), r_j + (\hat{p} - r_j)\cdot(1 - T + R/2)]$, gdje parametri R i T imaju jednako značenje kao u okolini jednog stroja. I u ovoj okolini, ti parametri poprimaju vrijednosti 0.2, 0.4, 0.6, 0.8 i 1 u raznim kombinacijama. Kao i kod okoline jednog stroja, minimizira se ukupno težinsko zaostajanje. Za svaku instancu se računa funkcija cilja, ali kako instance imaju različit broj poslova i strojeva, napravljena je normalizacija kako GP prilikom razvijanja pravila ne bi više pozornosti pridavao instancama koje imaju veći utjecaj na iznos funkcije cilja. Normalizacija za kriterij težinskog zaostajanja radi se prema izrazu: $f_i = \frac{Twt}{n\overline{w_j}\overline{p}}$, gdje su $\overline{w_j}$ srednje vrijednosti težina poslova, a $\overline{p}$ srednje vrijednosti trajanja poslova. Ukupna vrijednost funkcije dobrote dobiva se sumiranjem normalizirane vrijednosti funkcije dobrote za svaku instancu.

U ovoj okolini za raspoređivanje poslova koristi se sljedeća shema generiranja rasporeda: čekaj dok barem jedan stroj i barem jedan posao ne postanu dostupni. Izračunaj prioritete za svaki dostupan neraspoređen posao na svakom dostupnom stroju. Odredi najbolji stroj za svaki neraspoređeni posao temeljem izračunatih prioriteta. Od svih poslova za koje je najbolji stroj (za taj posao) dostupan, za raspoređivanje odaberi onaj koji ima najveći prioritet. Ukoliko ne postoji posao za koji je za njega najbolji stroj dostupan, raspoređivanje se odgađa dok neki drugi posao ili stroj ne postanu dostupni. Ovaj postupak se ponavlja sve dok postoje neraspoređeni poslovi. Pseudokod opisane sheme je dan algoritmom 3.

Kao i kod okoline jednog stroja, i ovdje je potrebno definirati skup funkcija i značajki koje će GP koristiti prilikom stvaranja prioritetnih pravila. Tablica 6.2 prikazuje podskup funkcija i značajki za okolinu nesrodnih strojeva.

Algoritam 3 Shema generiranja rasporeda u okolini nesrodnih strojeva

```
1: dok postoje dostupni neraspoređeni poslovi ponavljaj
2:     čekaj dok barem jedan posao i jedan stroj ne postanu dostupni
3:     za svaki dostupan posao i za svaki dostupan stroj ponavljaj
4:         izračunaj prioritet π_ij raspoređivanja posla j na stroju i
5:     za svaki dostupan posao ponavljaj
6:         odredi najbolji stroj (onaj na kojemu se postiže najbolja vrijednost za π_ij)
7:     dok postoje poslovi za koje je najbolji stroj dostupan ponavljaj
8:         odredi najbolje prioritete za te poslove
9:         rasporedi posao s najboljim prioritetom na odgovarajući stroj
```

Prve četiri značajke jednake su kao i kod okoline jednog stroja. U okolini nesrodnih strojeva postoji više od jednog stroja, pa informacije o vremenu izvršenja na različitim strojevima (*pmin* i *pavg*) mogu dati korisne informacije GP-u prilikom razvijanja prioritetnog pravila. Značajka *PAT* sadrži informaciju o količini vremena potrebnoj da stroj za koji trenutno promatrani posao daje najkraće vrijeme izvršavanja, postane dostupan. Značajka *MR* označava količinu vremena potrebnu da trenutni stroj postane dostupan. Posljednja značajka, *age*, sadrži informaciju o vremenu koje je posao proveo u sustavu kako bi se bolje moglo odlučiti koji posao rasporediti sljedeći, tako da nijedan posao ne čeka predugo na raspoređivanje. Kao i kod okoline jednog stroja, i ovdje su u skup značajki dodane vrijednosti 0 i 1.

Kod okoline nesrodnih strojeva vrijede sljedeće pretpostavke: informacije o poslovima nisu poznate unaprijed, poslovi postaju dostupni za raspoređivanje u nekom vremenskom trenutku, pa se koristi dinamičko raspoređivanje; jednom kada posao započne s izvršavanjem, ne smije se prekidati; svi strojevi su dostupni cijelo vrijeme, odnosno nema kvarova.

6.2.3 Oblikovanje ispitnih primjera za problem raspoređivanja s ograničenim sredstvima

Instance za problem raspoređivanja s ograničenim sredstvima dolaze iz biblioteke *PSPLIB*[*]. Instance iz ove biblioteke su podijeljene u 4 skupine, u ovisnosti o broju aktivnosti koje sadrže. Prva skupina sadrži primjere sa 30 aktivnosti, druga sa 60, treća sa 90 i posljednja sa 120 aktivnosti. Za svaku skupinu problema definirani su parametri - kompleksnost mreže (*NC*), faktor sredstva (*RF*) i jakost sredstva (RS_r) koji se mijenjaju prilikom generiranja instanci. Za probleme s 30, 60 i 90 instanci kompleksnost mreže poprima vrijednosti 1.5, 1.8 i 2.1, faktor sredstva 0.25, 0.5, 0.75 i 1, a jakost sredstva 0.2, 0.5, 0.7 i 1. Na osnovu ovih parametara postoji 48 različitih kombinacija i za svaku od tih kombinacija je generirano po 10 instanci. Kod problema koji sadrže 120 aktivnosti, kompleksnost mreže i faktor sredstva poprimaju jednake vrijednosti kao i kod problema sa 30, 60 i 90 instanci, dok jakost sredstva poprima vrijednosti

[*]dostupno na: http://www.om-db.wi.tum.de/psplib/

Tablica 6.2: Podskup korištenih funkcija i značajki za okruženje nesrodnih strojeva

Funkcija	**Opis**
+	zbrajanje
-	oduzimanje
*	množenje
/	zaštićeno dijeljenje: $DIV(a,b) = \begin{cases} 1, & \lvert b\rvert < 0.000000001 \\ \frac{a}{b}, & inače. \end{cases}$
POS	$POS(a) = \begin{cases} a, & a > 0 \\ 0, & inače. \end{cases}$
Značajka	**Opis**
pt	trajanje izvršavanja posla j na stroju i (p_{ij})
dd	željeno vrijeme završetka (d_j)
w	težina (w_j)
SL	pozitivna dopuštena odgoda ($max\{d_j - p_j - time, 0\}$)
pmin	minimalno vrijeme izvršavanja posla na svim strojevima ($min_i(p_{ij})$)
pavg	prosječno vrijeme izvršavanja posla na svim strojevima
PAT	strpljenje (engl. *patience*)
MR	spremnost stroja (engl. *machine ready*)
age	vrijeme koje je posao proveo u sustavu ($vrijeme - r_j$)

0.1, 0.2, 0.3, 0.4 i 0.5. Na osnovu ovih parametara postoji 60 različitih kombinacija i jednako kao za prethodne probleme, generirano je po 10 instanci za svaku kombinaciju parametara. Prema tome, ukupan broj instanci u PSPLIB-u iznosi 2040. U ovom radu korištene su dvije inačice problema raspoređivanja s ograničenim sredstvima. U jednoj su u obzir uzeti samo instance problema koje sadrže 90 aktivnosti, a u drugoj se gledaju svi problemi, tj. problemi koji sadrže i 30 i 60 i 90 i 120 aktivnosti. U prvoj inačici su odabrani samo poslovi sa po 90 aktivnosti kako bi se vidjelo postoje li i u problemima koji se ne razlikuju po broju aktivnosti grupe koje se mogu pronaći temeljem značajki krajolika dobrote. Odabrani su poslovi s 90 aktivnosti jer je taj broj aktivnosti dovoljno velik da poslovi ne budu prejednostavni za rješavanje, a opet dovoljno malen da eksperimenti ne traju vremenski predugo. U drugoj inačici su odabrani svi problemi dostupni u PSPLIB-u kako bi se vidjelo kako različit broj aktivnosti u poslovima utječe na grupiranje. Dodatno, taj skup problema se koristi u novijim radovima koji se bave problemom raspoređivanja s ograničenim sredstvima. U oba slučaja, instance se dijele na skup

za učenje, skup za validaciju i skup za testiranje. Ta tri skupa moraju biti disjunktna kako bi rezultati bili valjani. Korištena je implementacija iz [170], u kojoj skup za validaciju služi kako bi se odredilo koje naučeno pravilo će se najbolje ponašati na dosad neviđenim primjerima. Za probleme koji sadrže samo instance s 90 aktivnosti, skup za učenje sadrži 192 instance, skup za validaciju 96, dok su u skupu za testiranje preostale 192 instance problema. U slučaju kada se koriste sve instance problema, skup za učenje se sastoji od 816 instanci, skup za validaciju od 408, a u skupu za testiranje je preostalih 816 instanci problema. U svakom skupu se nalazi jednak postotak primjera s 30, 60, 90 i 120 aktivnosti od ukupnog broja instanci.

U ovoj okolini se za kriterij optimizacije uzima funkcija kojoj je cilj minimizirati krajnje vrijeme završetka projekta, ali je modificirana tako da instance koje su manje ili više teške za rješavanje otprilike jednako pridonose ukupnoj funkciji cilja. Dobrota na i - toj instanci se dobija kao: $f_i = \frac{C_i}{p_i^{avg} \cdot \sqrt{n_i}}$, gdje je C_i vrijeme završetka i - tog projekta, p_i^{avg} prosječno trajanje aktivnosti unutar projekta, a n_i broj aktivnosti unutar projekta. S obzirom da se pravila razvijaju na skupu instanci, ukupna funkcija cilja je aritmetička sredina vrijednosti f_i svih instanci koje se nalaze u promatranom skupu.

Prilikom pravljenja rasporeda korištenjem dobivenih prioritetnih pravila, moguće je koristiti slijednu shemu generiranja rasporeda ili usporednu shemu generiranja rasporeda. U ovom radu se po uzoru na [170] koristi usporedna shema bez odgode. U njoj se prvo na temelju uvjeta prednosti izračunava skup dostupnih aktivnosti koje se u danom vremenskom trenutku mogu rasporediti. Zatim se uzima aktivnost sa najboljom vrijednošću prioritetnog pravila i ako za nju postoji dovoljna količina sredstva, ona se stavlja u raspored. U suprotnom se raspoređuje aktivnost koja je druga po redu na temelju izračunatih prioriteta. Postupak se ponavlja sve dok postoje neraspoređene aktivnosti.

Kao što je već napomenuto, ovdje se koriste dvije inačice RCPSP-a i osim što se razlikuju po izgledu skupova na kojima se radi raspoređivanje, razlikuju se i po funkcijama i značajkama koji se prosljeđuju GP-u za generiranje prioritetnih pravila. Funkcije koje se koriste u prvoj verziji gdje se promatraju samo instance sa po 90 aktivnosti su prikazane u tablici 5.1, dok su za drugu verziju, u kojoj se promatraju instance s različitim brojem aktivnosti, funkcije dodatno modificirane i prikazane su u tablici 6.3. Značajke koje se koriste u prvoj verziji (sa po 90 aktivnosti) su prikazane u tablici 6.4. Za drugu verziju se koristi samo podskup od skupa značajki navedenog u tablici 6.4, točnije, koriste se samo: ARU, DPC, GRPW, LF, LS, NSP, NUA, TD i TNA. Dodatno, uz sve navedene značajke, u skup značajki se u oba slučaja dodaju vrijednosti 0 i 1.

Značajke iz tablice 6.4 su podijeljene na one karakteristične projektu, i one karakteristične aktivnosti. Dinamičke značajke su: NUA, NAA, NPA, SUD, SAD, SPD, NSP i SL, dok su sve ostale statičke. Značajke karakteristične projektu sadrže informacije koje se tiču cijelog promatranog projekta. Značajke RF, RS i TNA se mogu dobiti iz konfiguracijske datoteke za

Tablica 6.3: Podskup korištenih funkcija za RCPSP s 30, 60, 90 i 120 aktivnosti

Funkcija	**Opis**
+	zbrajanje
-	oduzimanje
*	množenje
/	zaštićeno dijeljenje: $DIV(a,b) = \begin{cases} 1, & \lvert b\rvert < 0.000000001 \\ \frac{a}{b}, & inače. \end{cases}$
MAX	$MAX(a,b) = \begin{cases} a, & a > b \\ b, & inače. \end{cases}$
MIN	$MIN(a,b) = \begin{cases} a, & a < b \\ b, & inače. \end{cases}$
APS	apsolutna vrijednost
NEG	$NEG(a) = (-1) \cdot a$

instancu RCPSP problema. Ukupna duljina trajanja projekta (TD) se dobija sumiranjem trajanja svih aktivnosti. Za vrijeme izvođenja programa, aktivnosti mogu biti raspoređene, neraspoređene, ili aktivne - one za koje su zadovoljeni uvjeti, ali još nisu raspoređene. Značajke NUA, NAA, NPA, SUD, SAD i SPD se bave tim vrstama aktivnosti, odnosno njihovim brojevima i trajanjima izvršavanja. Značajke specifične aktivnosti sadrže informacije koje se mogu izvući iz pojedine aktivnosti. Trajanje izvršavanja neke aktivnosti je bitna informacija prilikom određivanja prioriteta, pa je zato značajka D dodana u skup. Značajke RR, RRT i ARU se bave ograničenjima na sredstva, dok se DPC, DSC, TPC, TSC, SPC i SSC bave uvjetima prednosti. Prilikom određivanja prioriteta je važno znati koliko resursa je potrebno za izvršavanje aktivnosti i koliko prethodnika i sljedbenika pojedina aktivnost ima. Aktivnosti s više sljedbenika bi vjerojatno trebale imati veći prioritet od aktivnosti s manje sljedbenika, kako bi se i ti sljedbenici mogli što prije rasporediti. Značajka GRPW* daje najveći težinski pozicijski rang sljedbenika promatrane aktivnosti. Značajke ES, EF, LS i LF daju informaciju o tome kada neka aktivnost najranije i najkasnije može započeti i završiti kako bi se znalo u kojem vremenskom rasponu aktivnost mora biti raspoređena. NSP daje informaciju o broju prethodnika, dok je SL jednak kao i u prethodne dvije okoline.

Tablica 6.4: Podskup korištenih značajki za RCPSP s 90 aktivnosti

Kategorija	Značajka	Opis
Karakteristične projektu	RF	faktor sredstva
	RS	jakost sredstva
	TNA	broj svih aktivnosti (bez fiktivnih)
	TD	ukupno trajanje projekta
	NUA	broj neizvršenih aktivnosti
	NAA	broj trenutno aktivnih aktivnosti
	NPA	broj izvršenih aktivnosti
	SUD	zbroj vremenskog trajanja neizvršenih aktivnosti
	SAD	zbroj vremenskog trajanja trenutno aktivnih aktivnosti
	SPD	zbroj vremenskog trajanja izvršenih aktivnosti
Karakteristične aktivnosti	D	trajanje aktivnosti
	RR	broj potrebnih sredstava
	RRT	RR pomnožen s potrebnom količinom pojedinog sredstva
	ARU	prosječna uporaba sredstva
	DPC	broj direktnih prethodnika
	DSC	broj direktnih sljedbenika
	TPC	ukupan broj prethodnika
	TSC	ukupan broj sljedbenika
	SPC	broj razina u stablu prethodnika
	SSC	broj razina u stablu sljedbenika
	GRPW*	najveći težinski pozicijski rang sljedbenika
	ES	najraniji trenutak početka aktivnosti
	EF	najraniji trenutak završetka aktivnosti
	LS	najkasniji trenutak početka aktivnosti
	LF	najkasniji trenutak završetka aktivnosti
	NSP	broj raspoređenih prethodnika
	SL	vrijeme čekanja

6.3 Oblikovanje eksperimenata

U svakoj okolini su se za svaku instancu problema odredile značajke krajolika dobrote. Za svaku instancu se na slučajan način generiralo po 30 početnih jedinki. Iz svake početne jedinke je napravljena slučajna šetnja duljine 1000 na način da se od početne jedinke slučajnim odabirom jednog od tri uvedena operatora iz poglavlja 5 dobija sljedeća jedinka u šetnji i zatim se iz te jedinke slučajnim odabirom jednog od tri uvedena operatora dobija sljedeća jedinka itd. Uz slučajan odabir operatora, u svakom koraku se na slučajan način odabire i čvor na kojemu će se operator izvršiti. Za svaku jedinku iz slučajne šetnje je određena vrijednost funkcije dobrote i na temelju tih vrijednosti su izračunate odgovarajuće značajke. Prva značajka koja se određuje je prosječna vrijednost dobrote koja se dobije kao prosjek vrijednosti dobrote svih jedinki iz slučajne šetnje. Za svaku slučajnu šetnju se dobija po jedna prosječna vrijednost, pa kako bi se dobila krajnja vrijednost značajke, izračunata je aritmetička sredina i standardna devijacija dobivenih 30 vrijednosti i te 2 vrijednosti predstavljaju značajke za prosječnu vrijednost dobrote.

Zatim se određuje koeficijent autokorelacije koji se računa po izrazu: $\hat{r}(1) = \frac{\sum_{t=1}^{l-1}(f(x_t)-\overline{f})\cdot(f(x_{t+1})-\overline{f})}{\sum_{t=1}^{l}(f(x_t)-\overline{f})^2}$, gdje je $f(x_t)$ vrijednost funkcije dobrote jedinke x_t, $\overline{f}$ je aritmetička sredina svih dobrota u slučajnoj šetnji, a $l = 1000$ je duljina slučajne šetnje. Na ovaj način se za svaku instancu problema dobija 30 vrijednosti za koeficijent autokorelacije. Kako bi se dobile vrijednosti koje predstavljaju koeficijent autokorelacije za promatranu instancu problema, odredila se aritmetička sredina i standardna devijacija dobivenih 30 vrijednosti, i te dvije vrijednosti predstavljaju koeficijent autokorelacije za pojedinu instancu.

Na isti način određene su dvije vrijednosti koje predstavljaju stopu neutralnih susjeda. Za svaku slučajnu šetnju prebrojan je broj susjeda koji imaju jednaku vrijednost funkcije dobrote i taj broj je podijeljen duljinom šetnje. Za dobivenih 30 vrijednosti po instanci, određena je aritmetička sredina i standardna devijacija.

Nakon toga su izračunate značajke temeljene na skali udaljenosti, koje se za svaku instancu dobiju kao $r = \frac{|f(x_i)-f(x_j)|}{d(x_i,x_j)}$, gdje je $f(x_i)$ vrijednost funkcije dobrote jedinke x_i, a $d(x_i,x_j)$ je udaljenost između jedinki x_i i x_j. Vrijednost r se računa za svaki par jedinki x_i i x_j iz slučajne šetnje, pa se na taj način dobija $\frac{1000\cdot 999}{2}$ vrijednosti za svaku slučajnu šetnju. Tada se za svaku slučajnu šetnju određuje minimum, donji kvartil, medijan, gornji kvartil, maksimum, aritmetička sredina i standardna devijacija dobivenih vrijednosti. Na taj način se dobije po 30 vrijednosti za navedene mjere za svaku instancu, pa se još dodatno za svaku mjeru određuje aritmetička sredina i standardna devijacija. Iz toga proizlazi 14 vrijednosti koje opisuju značajke temeljene na udaljenosti. S obzirom da u svim promatranim primjerima na ovaj način minimum ispada nula (jer za promatrane probleme postoje neutralna područja u krajoliku u kojima su postignute vrijednosti funkcije cilja jedinki jednake), dodatno se isti postupak ponovio

s podacima iz kojih su izbačene nule, pa se dobija još 14 značajki koje se temelje na udaljenosti bez nula. Ukoliko u nekoj instanci problema sve vrijednosti ove mjere ispadnu nula, onda se u tom slučaju nule ne izbacuju kako se ipak ne bi izgubila informacija o toj instanci.

Nakon toga se određuju mjere probiranja izrazima opisanim u poglavlju 2. Vrijednost l je ovdje jednaka duljini slučajne šetnje, odnosno jednaka je 1000. Za svaku slučajnu šetnju se određuje minimum, maksimum i raspon slučajnog probiranja, pa se na svakoj instanci dobije po 30 vrijednosti za svaku od te tri mjere. Na kraju se za svaku mjeru određuje aritmetička sredina i standardna devijacija, što rezultira sa 6 značajki.

Zadnje mjere koje se određuju su informacijske mjere. Ove se mjere računaju kako je opisano u poglavlju 2. I za njih se dobije po 30 vrijednosti za svaku instancu, pa se ponovno računaju aritmetička sredina i standardna devijacija kako bi se dobile samo 2 vrijednosti po instanci. S obzirom da se za različite vrijednosti ε dobija različit pogled na krajolik, ovdje se izračunala duljina intervala iz koje dolaze vrijednosti funkcije dobrote za jedinke iz slučajne šetnje kao $d_I = \lceil \max(f(x_i)) \rceil - \lfloor \min(f(x_i)) \rfloor$, pa su se vrijednosti sadržaja informacije i djelomičnog sadržaja informacije računale koristeći sljedeće vrijednosti za ε: $0.02 \cdot d_I$, $0.1 \cdot d_I$, $0.2 \cdot d_I$ i $0.5 \cdot d_I$. Za stabilnost informacije, vrijednosti ε se gledaju u intervalu $[0, d_I]$ s korakom 0.01.

Na kraju se ovim postupkom za svaku instancu dobija 58 značajki krajolika dobrote koje se koriste za grupiranje instanci problema.

S obzirom da se skupovi problema sastoje od instanci s različitim svojstvima (broj poslova, vrijeme izvršavanja, broj strojeva, itd.), može se pretpostaviti da će za različite instance GP s različitim parametrima dati najbolje rezultate. Iz tog razloga, instance u svakom problemu se mogu grupirati u klastere sličnih instanci za koje će GP s prilagođenim parametrima dati bolja rješenja. U ovom radu se za grupiranje koriste algoritam k-means i algoritam EM [171] koji koristi Gaussove jezgre. Za svaku izračunatu značajku, grupiranje se radilo na temelju aritmetičke sredine i standardne devijacija za pojedinu značajku i na temelju samo aritmetičke sredine ili samo standardne devijacije svake značajke. Na taj način se dobilo 29 grupiranja temeljenih na dvije varijable, i 58 grupiranja temeljenih na jednoj varijabli. No, za svako od tih grupiranja je potrebno odrediti koji broj grupa je najbolji za promatrani problem. Kako bi se dobio najbolji broj grupa kod EM grupiranja, grupiranje na temelju određenih varijabli se ponovilo 10 puta i za svako to grupiranje se za vrijednosti od jednog do 6 klastera izračunao Bayesov informacijski kriterij (BIC) [172] i Akaike informacijski kriterij (AIC) [173]. Nakon računanja vrijednosti za oba kriterija u svim slučajevima, odabire se model koji daje najmanji BIC kriterij i model koji daje najmanji AIC kriterij. Između ta dva modela, za grupiranje se odabire onaj koji daje manji broj klastera kao rezultat. Za okolinu jednog stroja i okolinu nesrodnih strojeva algoritam EM nije davao dobre rezultate, pa se u tim slučajevima za grupiranje koristio algoritam k-means. Za određivanje broja klastera kod algoritma k-means, računala su se tri mjerila:

silhouette [174], Davies - Bouldin [175] te Calinski i Harabasz mjera [176] za broj klastera od 2 do 10. Mjera *silhouette* se računa kao razlika između srednje vrijednosti udaljenosti unutar klastera i srednje vrijednosti udaljenosti između najbližih klastera, podijeljena sa maksimumom te dvije vrijednosti za svaki uzorak. Vrijednosti ove mjere variraju od -1 do 1, i što su one veće, to je grupiranje bolje. Mjera Davies - Bouldin se definira kao prosječna mjera sličnosti svakog klastera s njemu najsličnijim klasterom, gdje se sličnost gleda kao omjer udaljenosti unutar klastera i udaljenosti između klastera. U ovom mjerilu niže vrijednosti označavaju bolje grupiranje. Mjera Calinski i Harabasz se definira kao omjer između raspršenosti unutar klastera i raspršenosti između različitih klastera. Kao i kod prve mjere, i ovdje veće vrijednosti označavaju bolje grupiranje.

Za svaku okolinu koja se promatrala u sljedećim će poglavljima biti prikazano kako su se točno instance grupirale i kako izgledaju vrijednosti odgovarajućih mjera.

Kao što je već napomenuto, ideja grupiranja se javlja kako bi se instance problema koje su najsličnije jedne drugima stavile u istu grupu jer se skup početnih instanci razlikuje po određenim svojstvima, pa se onda za te grupe treba odrediti koji su parametri GP-a najbolji, odnosno kojim parametrima će se na kraju dobiti najbolje vrijednosti funkcije cilja. Za svaku dobivenu grupu, odnosno klaster, parametri su se određivali automatizirano koristeći *irace*. *irace* je okruženje u kojemu se pomoću iterativnog uzorkovanja (engl. *iterated racing*) dolazi do najbolje konfiguracije parametara za željeni problem. Iterativno uzorkovanje je metoda automatske konfiguracije parametara koja se sastoji od tri koraka: uzorkovanje novih konfiguracija na temelju određene distribucije, odabir najboljih konfiguracija temeljem uspoređivanja ("utrkivanja") iz skupa dobivenog u prvom koraku te ažuriranje distribucije za uzorkovanje tako da bude pristranija boljim rješenjima dobivenim u drugom koraku. Ova tri koraka se ponavljaju sve dok se ne ispuni neki od kriterija zaustavljanja. U okruženju *irace*, svaki od parametara koji treba odabrati ima svoju distribuciju koja je neovisna o distribucijama ostalih parametara koji se biraju. Distribucije su odrezana normalna distribucija za kontinuirane parametre ili diskretna distribucija za kategoričke parametre [168]. Prilikom trećeg koraka, u normalnoj distribuciji se mijenjaju srednja vrijednost i standardna devijacija kojima je distribucija određena, a kod diskretne distribucije se mijenjaju vrijednosti vjerojatnosti odabira određenih vrijednosti parametra. Nakon što se načini uzorak konfiguracija, provodi se uspoređivanje i to tako da se na nekoj instanci problema evaluiraju sve uzorkovane konfiguracije. Nakon nekog određenog broja koraka, konfiguracije se uspoređuju Friedmanovim testom i one koje su statistički značajno lošije od ostalih se izbacuju iz promatranja, pa se postupak nastavlja na "preživjelim" (preostalim) konfiguracijama na nekoj drugoj instanci. Postupak se nastavlja sve dok se ne dođe do minimalnog broja preživjelih konfiguracija, maksimalnog broja instanci koje se mogu koristiti ili dok se ne dostigne maksimalno ukupno vrijeme izvršavanja ili maksimalan broj eksperimenata. Moguće je uvesti i elitizam, odnosno da se najbolja dosad pronađena konfiguracija mora prenijeti u slje-

deću iteraciju.

Kako bi *irace* mogao odrediti najbolje parametre za neki skup problema, potrebno je definirati parametre. Prosljeđuje se naziv parametra, tip parametra (kontinuirani, cjelobrojni, kategorički), a ukoliko je potrebno, za određene parametre se mogu dodati i uvjeti. Zatim se u konfiguracijskoj datoteci prosljeđuje izvršna datoteka koja pokreće algoritam za koji treba konfigurirati parametre, maksimalan broj eksperimenata, putanje do instanci na kojima se program treba izvršavati, kao i broj elitističkih konfiguracija. Za svaki od promatranih problema će u sljedećim poglavljima biti navedeno koji su parametri korišteni u *irace*-u.

Na kraju, kako bi se dobiveni rezultati mogli usporediti sa nekim vrijednostima, za sve instance (prije) grupiranja se određuje jedan skup parametara koji je najbolji za cijeli skup za učenje. Ovaj postupak se obavlja "ručno", odnosno GP se pokreće s raznim kombinacijama parametara dok se ne dobiju parametri za koje GP postiže najbolje rješenje na promatranom skupu za učenje. U ovom postupku se jedan parametar mijenja, a ostali se fiksiraju dok se ne odredi za koju vrijednost promatranog parametra GP da najbolje rješenje. Nakon toga se taj parametar fiksira s ostalima, a sljedeći se mijenja dok se ne istraže svi parametri. Ovaj postupak je jako dugotrajan, pa i zbog toga postoji potreba za automatskom konfiguracijom parametara.

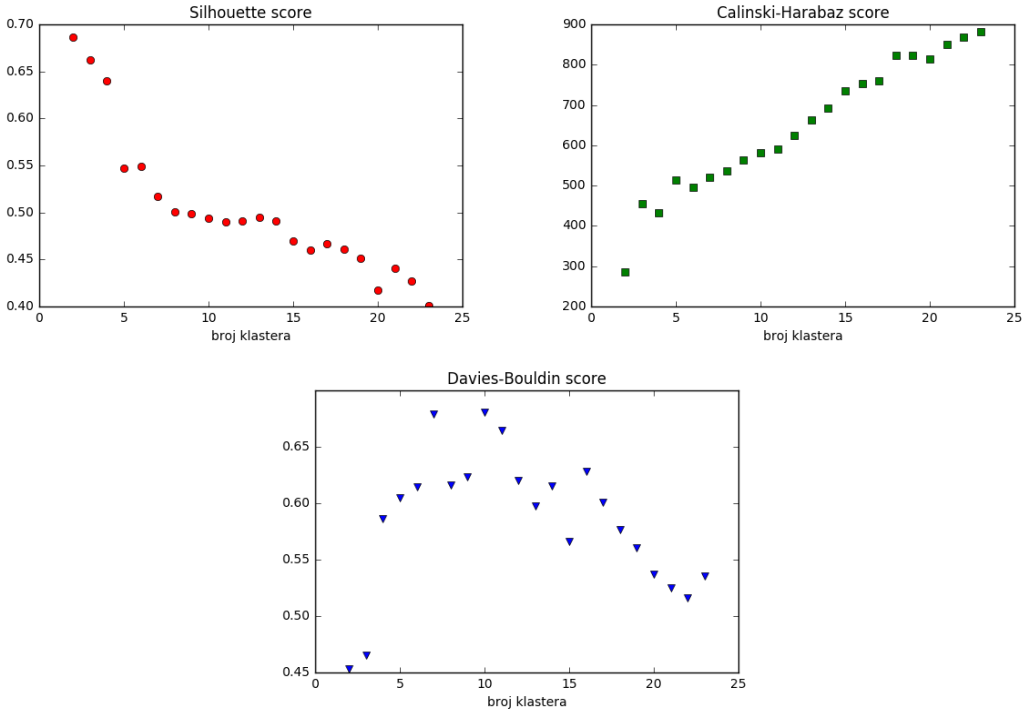
Kada se odrede najbolji parametri za cijeli skup za učenje i najbolji parametri za pojedine grupe dobivene grupiranjem, GP se pokreće 30 puta s obje kofiguracije parametara na odgovarajućim skupovima za testiranje i uspoređuju se dobiveni rezultati, što će biti vidljivo u nastavku ovog poglavlja.

6.4 Rezultati u okolini jednog stroja

U ovom odjeljku će biti prikazani rezultati eksperimenata napravljenih za okolinu jednog stroja. Prvo će biti prikazan postupak grupiranja i na koji način su se odabrali parametri na temelju kojih se radi grupiranje, a nakon toga će biti prikazani rezultati dobiveni automatskom i ručnom konfiguracijom parametara za pojedini klaster te će se učiniti njihova usporedba.

6.4.1 Grupiranje

U ovoj okolini prvo su obavljeni eksperimenti u kojima se grupiranje radilo koristeći algoritam EM, no na klasterima koji su se dobili tim algoritmom, automatska konfiguracija parametara nije dala bolje rezultate od parametara određenih ručno. Iz tog razloga se pokušalo promotriti hoće li algoritam k-means dati bolje grupiranje, odnosno hoće li u ovom slučaju bolje razdvojiti dane instance u klastere. Kao što je navedeno u odjeljku 6.3, koristile su se tri mjere kako bi se za svaki parametar, odnosno značajku (srednju vrijednost i standardnu devijaciju značajke) odredilo koja daje najbolje grupiranje.



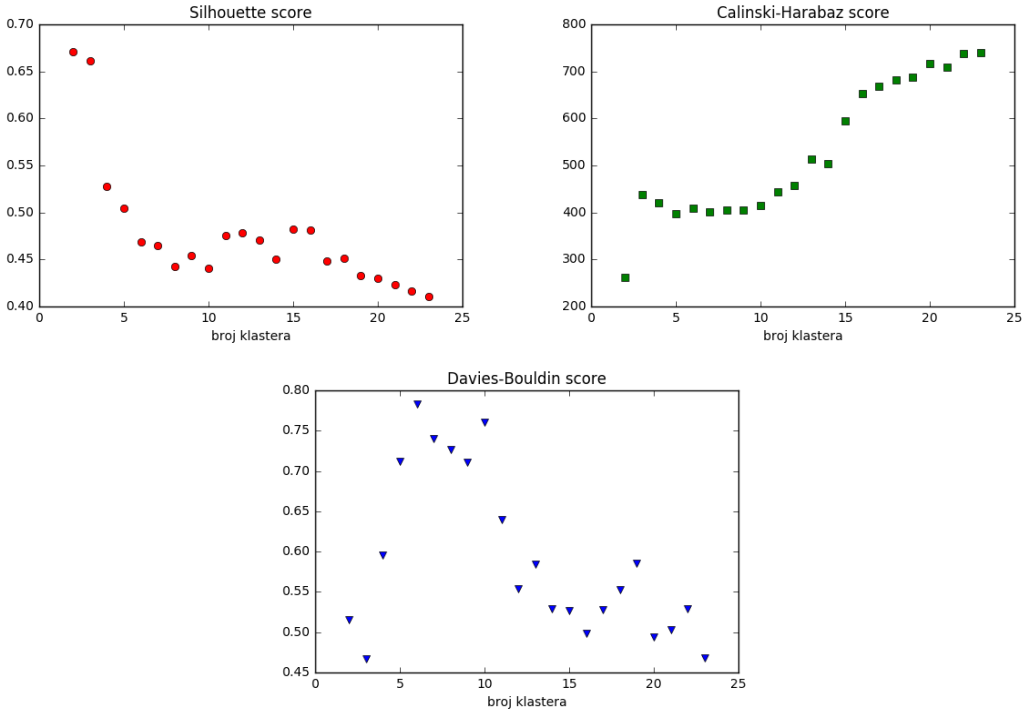
Slika 6.1: Prikaz dobivenih mjera za grupiranje temeljeno na prosjeku mjere temeljene na udaljenosti u okolini jednog stroja

Za svaku značajku je napravljeno grupiranje na broj grupa iz cjelobrojnog intervala $[2, 23]$ i za svaki broj grupa su nacrtane odgovarajuće mjere. Najbolji rezultati (na temelju mjera) su postignuti za prosjek mjere temeljene na udaljenosti (na slikama oznaka lengthscale_avg prema engleskom *lengthscale average*) i za maksimum mjere temeljene na udaljenosti bez nula (na slikama oznaka lengthscale_bn_max). Na slici 6.1 su prikazane mjera *silhouette*, mjera Davies - Bouldin i mjera Calinski Harabaz dobivene za prosjek mjere temeljene na udaljenosti, dok su na slici 6.2 prikazane vrijednosti tih triju mjera za maksimum mjere temeljene na udaljenosti bez nula.

U slučaju prosjeka mjere temeljene na udaljenosti, odabrano je da se podaci trebaju grupirati u 3 klastera. Vidljivo je da je mjera *silhouette* najbolja za 2 i 3 klastera, kod mjere Davis-Bouldin je vrijednost nešto malo bolja za 2 nego za 3 klastera, ali je kod mjere Calinski Harabaz vrijednost za 3 klastera znatno bolja nego za 2 klastera, pa je iz tog razloga u ovom slučaju odabrano grupiranje instanci u 3 klastera.

I u slučaju maksimuma mjere temeljene na udaljenosti bez nula, odabrano je da se podaci trebaju grupirati u 3 klastera. Ovdje su u mjeri *silhouette* rezultati podjednako dobri za 2 i 3 klastera dok se u mjeri Davis - Bouldin najbolji rezultat postigne za 3 klastera. Mjera Calinski Harabaz također pokazuje da je 3 klastera bolji izbor nego 2 za promatrane instance.

Tablica 6.5 daje prikaz broja instanci u skupu za učenje i skupu za testiranje koje su se našle u pojedinom klasteru za prosjek mjere temeljene na udaljenosti. Isto tako, vidljivo je i koji su predstavnici klastera, pri čemu prvi element uređenog para predstavlja srednju vrijednost dane



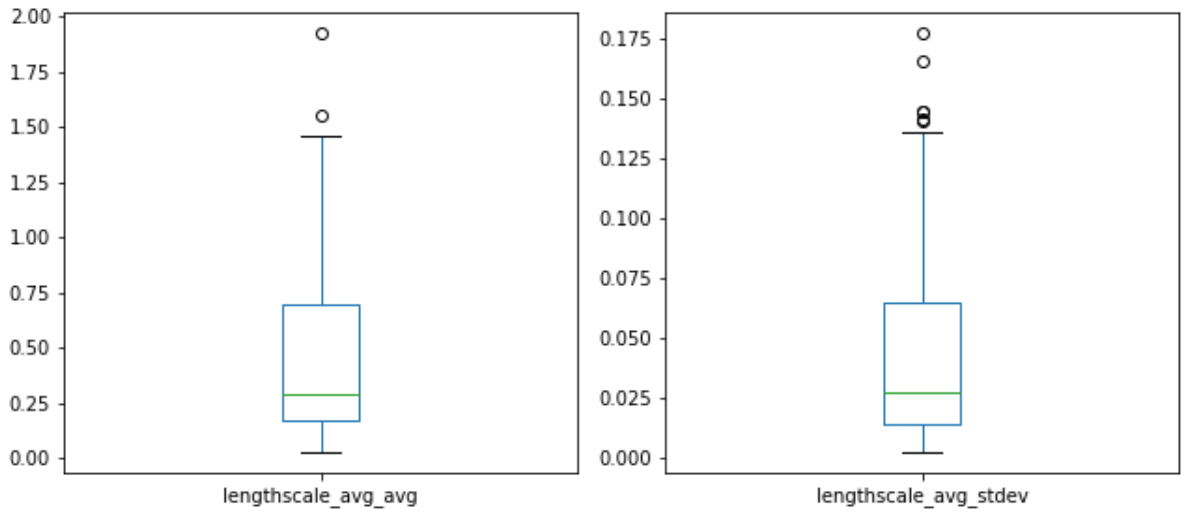
Slika 6.2: Prikaz dobivenih mjera za grupiranje temeljeno na maksimumu mjere temeljene na udaljenosti bez nula u okolini jednog stroja

značajke, a drugi element standardnu devijaciju dane značajke. Zanimljivo je vidjeti da u skupu za učenje, na temelju kojega se i radilo dobiveno grupiranje, prvi klaster ima najviše elemenata, a drugi najmanje, dok je u skupu za testiranje situacija obrnuta. Razlog tomu vjerojatno leži u načinu generiranja instanci u skupu za učenje i testiranje. Kod skupa za testiranje se trajanje poslova generira iz uniformne, bimodalne i normalne razdiobe, za razliku od skupa za učenje gdje se trajanja poslova generiraju samo iz uniformne razdiobe, pa je vjerojatno više instanci u skupu za testiranje generirano tako da im značajke odgovaraju trenutnom drugom klasteru.

Tablica 6.5: Broj instanci u klasterima i predstavnici klastera za prosjek mjere temeljene na udaljenosti u okruženju jednog stroja

	klaster1	**klaster2**	**klaster3**
# instanci (skup za učenje)	60	17	23
# instanci (skup za testiranje)	40	436	124
predstavnik klastera	(0.1869, 0.0168)	(1.3132, 0.1253)	(0.6555, 0.0590)

Slika 6.3 daje prikaz raspona vrijednosti za aritmetičku sredinu i standardnu devijaciju prosjeka mjere temeljene na udaljenosti. Vidljivo je da su i za aritmetičku sredinu i za standardnu devijaciju vrijednosti, kao i medijan na nižoj strani intervala u kojem se nalaze sve vrijednosti. Maksimumi koji se pojavljuju su već stršeće vrijednosti, pa je za očekivati da će klasteri koji grupiraju niže vrijednosti imati više instanci nego oni koji grupiraju više vrijednosti.

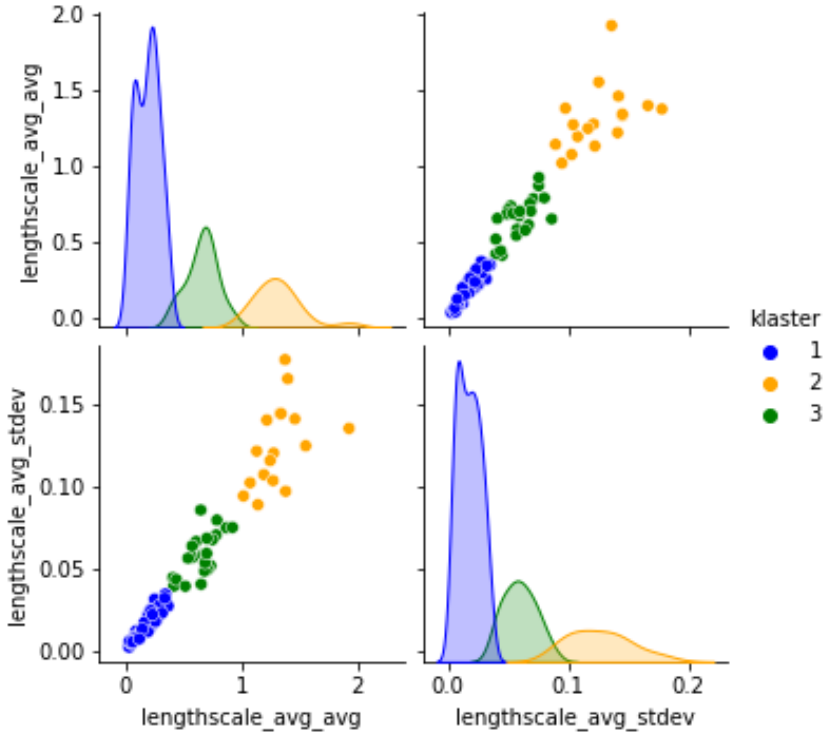


Slika 6.3: Kutijasti dijagram vrijednosti prosjeka mjere temeljene na udaljenosti za okruženje jednog stroja

Slika 6.4 daje prikaz dijagrama raspršenosti za aritmetičku sredinu i standardnu devijaciju prosjeka mjere temeljene na udaljenosti. Na glavnoj dijagonali se nalaze razdiobe za svaki od dobivenih klastera, označene različitim bojama, dok je na sporednoj dijagonali prikazana veza između aritmetičke sredine i standardne devijacije dane mjere. Vidljivo je da u prvi klaster ulaze podaci koji imaju nisku vrijednost aritmetičke sredine i standardne devijacije prosjeka mjere temeljene na udaljenosti, u drugom klasteru su visoke vrijednosti za obje značajke, dok su se u treći klaster grupirale vrijednosti iz sredine raspona. Isto tako, iz ovog dijagrama je vidljivo da su aritmetička sredina i standardna devijacija prosjeka mjere temeljene na udaljenosti pozitivno korelirane.

I za maksimum mjere temeljene na udaljenosti bez nula je situacija slična kao kod prethodne mjere. Tablica 6.6 daje prikaz broja instanci u skupu za učenje i skupu za testiranje koje su se našle u pojedinom klasteru za maksimum mjere temeljene na udaljenosti bez nula. I ovdje je vidljivo koji su predstavnici klastera, pri čemu prvi element uređenog para predstavlja srednju vrijednost dane značajke, a drugi element standardnu devijaciju dane značajke. Kao i kod prethodne mjere, zanimljivo je vidjeti da u skupu za učenje, na temelju kojega se i radilo dobiveno grupiranje, prvi klaster ima najviše elemenata, a drugi najmanje, dok je u skupu za testiranje situacija obrnuta.

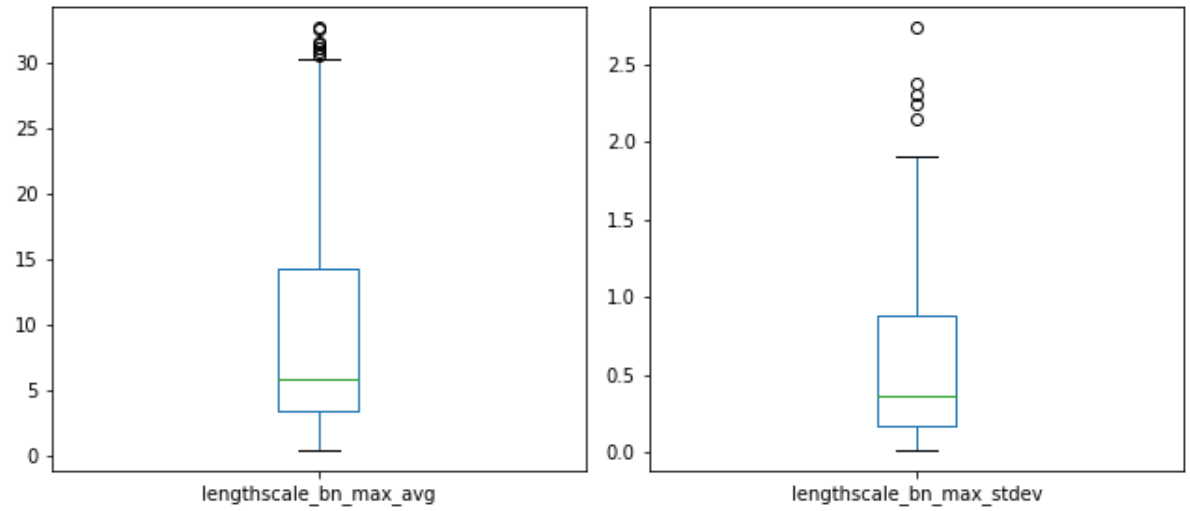
Slika 6.5 daje prikaz raspona vrijednosti za aritmetičku sredinu i standardnu devijaciju maksimuma mjere temeljene na udaljenosti bez nula. I ovdje je, kao i za prethodnu mjeru, vidljivo da su i za aritmetičku sredinu i za standardnu devijaciju dobivene vrijednosti na nižoj strani intervala u kojem se nalaze sve vrijednosti, iako je ovdje interval iz kojeg dolaze vrijednosti puno veći nego za prethodnu mjeru. Isto tako, medijan je bliži minimumu nego maksimumu. Maksimumi koji se pojavljuju su već stršeće vrijednosti, pa je za očekivati da će klasteri koji



Slika 6.4: Dijagram raspršenja za prosjek mjere temeljene na udaljenosti u okruženju jednog stroja

Tablica 6.6: Broj instanci u klasterima i predstavnici klastera za maksimum mjere temeljene na udaljenosti bez nula u okruženju jednog stroja

	klaster1	**klaster2**	**klaster3**
# instanci (skup za učenje)	63	13	24
# instanci (skup za testiranje)	27	471	102
predstavnik klastera	(4.0555, 0.2429)	(29.8464, 1.9569)	(14.9306, 0.9109)



Slika 6.5: Kutijasti dijagram vrijednosti maksimuma mjere temeljene na udaljenosti bez nula za okruženje jednog stroja

grupiraju niže vrijednosti imati više instanci nego oni koji grupiraju veće vrijednosti, kao što je to slučaj i kod prethodne mjere.

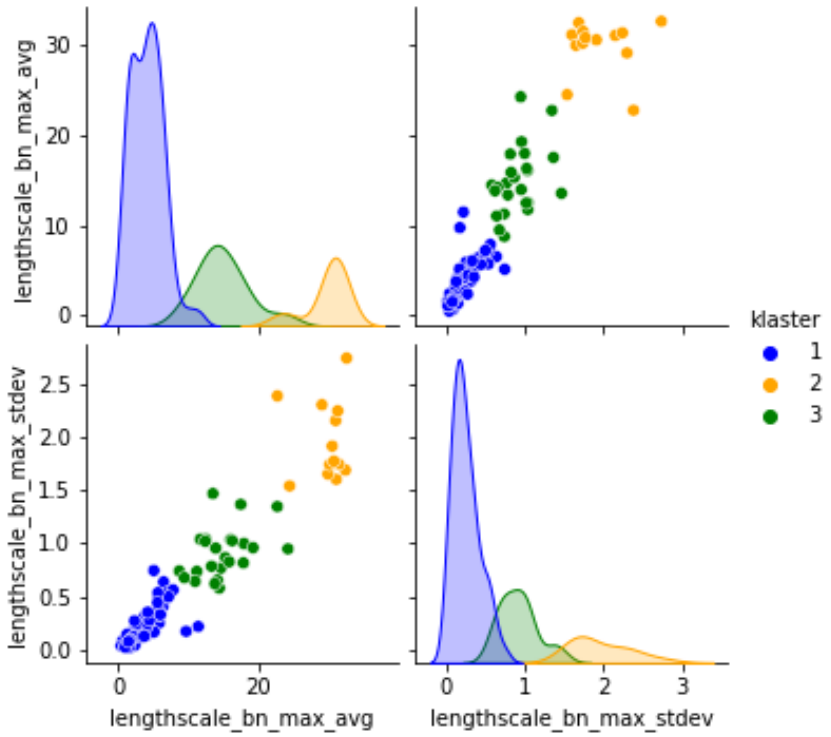
Slika 6.6 daje prikaz dijagrama raspršenosti za aritmetičku sredinu i standardnu devijaciju maksimuma mjere temeljene na udaljenosti bez nula. Vidljivo je da u prvi klaster ulaze podaci koji imaju nisku vrijednost aritmetičke sredine i standardne devijacije maksimuma mjere temeljene na udaljenosti, u drugom klasteru su visoke vrijednosti za obje značajke, dok su se u treći klaster grupirale vrijednosti iz sredine raspona. I ovdje su aritmetička sredina i standardna devijacija pozitivno korelirane, ali je dijagram ipak nešto više raspršen nego kod prosjeka mjere temeljene na udaljenosti.

6.4.2 Određivanje parametara

U svim promatranim okruženjima, prvi korak je odrediti parametre za GP na cijelom skupu za učenje, prije grupiranja instanci u klastere kako bi se dobili parametri koji će služiti za usporedbu sa parametrima dobivenim korištenjem *irace* okruženja u klasterima dobivenim grupiranjem.

Parametri za GP koji se trebaju odabrati u ovom slučaju su: broj generacija, veličina populacije, vjerojatnost mutacije, maksimalna dubina stabla i veličina turnira u turnirskoj selekciji. Početne vrijednosti za svaki od navedenih parametara su dane u tablici 6.7.

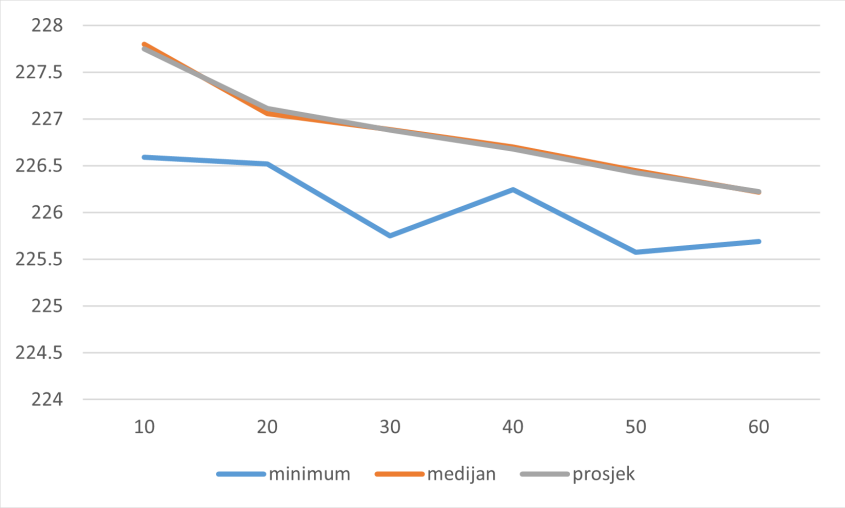
Prvi parametar za koji se određivala vrijednost je broj generacija. Promatrane su vrijednosti iz skupa {10, 20, 30, 40, 50, 60}, dok su vrijednosti svih ostalih parametara bile fiksirane na one iz tablice 6.7. Za svaku od vrijednosti iz promatranog skupa napravljeno je 30 pokretanja GP-a i zabilježena je najbolja dobivena vrijednost funkcije dobrote u svakom pokretanju. Slika 6.7 prikazuje kretanje minimalne vrijednosti, medijana i prosječne vrijednosti dobivenih vri-



Slika 6.6: Dijagram raspršenja za maksimum mjere temeljene na udaljenosti bez nula u okruženju jednog stroja

Tablica 6.7: Početni parametri za GP u okruženju jednog stroja

naziv parametra	**vrijednost**
broj generacija	20
veličina populacije	1000
vjerojatnost mutacije	0.3
maksimalna dubina stabla	5
veličina turnira	3



Slika 6.7: Prikaz kretanja minimuma, medijana i aritmetičke sredine za broj generacija između 10 i 60 u okolini jednog stroja

jednosti funkcije dobrote za odabrani broj generacija, dok su u tablici 6.8 prikazani minimum, maksimum, medijan, aritmetička sredina i standardna devijacija dobivenih vrijednosti. U ovom slučaju je za maksimalan broj generacija odabrana vrijednost 30, jer je vidljivo da se minimalna vrijednost u 30 generacija ne razlikuje znatno od minimalne vrijednosti postignute sa 50 i 60 generacija, a medijan postignut korištenjem 30 generacija nije puno lošiji od medijana postignutog korištenjem 50 ili 60 generacija. Isto tako, iz tablice 6.8 je vidljivo da je standardna devijacija za 30 i 60 generacija podjednaka, dok je za 30 generacija puno bolja nego za 50 generacija. Iako su rezultati postignuti sa 60 generacija možda malo bolji od rezultata postignutih sa 30 generacija, dodatno vrijeme koje bi bilo potrebno za računanje ostalih eksperimenata korištenjem 60 umjesto 30 generacija, ne opravdava uporabu većeg broja generacija.

Tablica 6.8: Vrijednosti funkcije dobrote za broj generacija između 10 i 60 u okolini jednog stroja

broj generacija	**10**	**20**	**30**	**40**	**50**	**60**
min	226.591	226.519	225.754	226.244	225.575	225.69
max	228.368	227.982	227.554	227.248	227.588	226.807
medijan	227.801	227.058	226.89	226.702	226.447	226.22
prosjek	227.752	227.1158	226.8852	226.6817	226.4287	226.225
stdev	0.543706	0.360239	0.357875	0.266255	0.397607	0.345431

Nakon odabira broja generacija, sljedeći parametar za koji je potrebno odrediti vrijednost je veličina populacije, odnosno broj jedinki u populaciji. Taj parametar se birao iz skupa {250, 500, 750, 1000, 1250, 1500}. Jednako kao i kod broja generacija, napravljeno je 30 pokretanja GP-a sa svakom od navedenih vrijednosti i rezultati minimuma, maksimuma, medijana, srednje vrijednosti i standardne devijacije su vidljivi u tablici 6.9.

Tablica 6.9: Vrijednosti funkcije dobrote za veličinu populacije iz skupa {250, 500, 750, 1000, 1250, 1500} u okolini jednog stroja

vel. Populacije	250	500	750	1000	1250	1500
min	226.594	226.326	226.208	225.882	225.969	226.032
max	228.634	228.089	227.855	227.57	227.364	227.121
medijan	227.642	226.79	226.987	226.794	226.7165	226.6175
prosjek	227.5481	226.9468	227.0408	226.809	226.7461	226.6245
stdev	0.481007	0.388482	0.427908	0.399319	0.303537	0.299452

Dodatno, za svaku veličinu populacije je nacrtan kutijasti dijagram koji prikazuje medijane i raspršenosti podataka, vidljiv na slici 6.8. Iz tablice 6.9 i spomenutog dijagrama, zaključeno je da treba odabrati veličinu populacije od 1250 jedinki. Iako je medijan manji za populaciju od 1500 jedinki, podaci su mnogo raspršeniji u tom slučaju nego za veličinu populacije od 1250 jedinki. Isto tako, za populaciju od 1250 jedinki se postiže bolja minimalna vrijednost nego za 1500 jedinki. Iz tih razloga, u daljnjim eksperimentima je veličina populacije postavljena na 1250 jedinki.

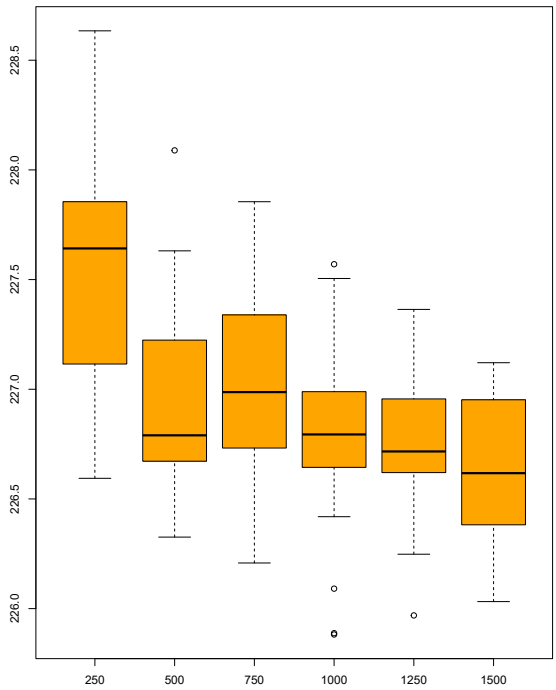
Nakon odabira veličine populacije, sljedeći parametar čiju vrijednost treba odabrati je vjerojatnost mutacije. U ovom slučaju su u obzir uzete vrijednosti iz skupa {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}. Tablica 6.10 prikazuje minimum, maksimum, srednju vrijednost, medijan i standardnu devijaciju dobivene pokretanjem GP-a sa odgovarajućim parametrima 30 puta. Dodatno, kao i za veličinu populacije, slika 6.9 daje kutijaste dijagrame koji prikazuju izgled dobivenih vrijednosti za odgovarajuće parametre.

Tablica 6.10: Vrijednosti funkcije dobrote za vjerojatnost mutacije iz skupa vrijednosti 0.1 - 0.9 u okolini jednog stroja

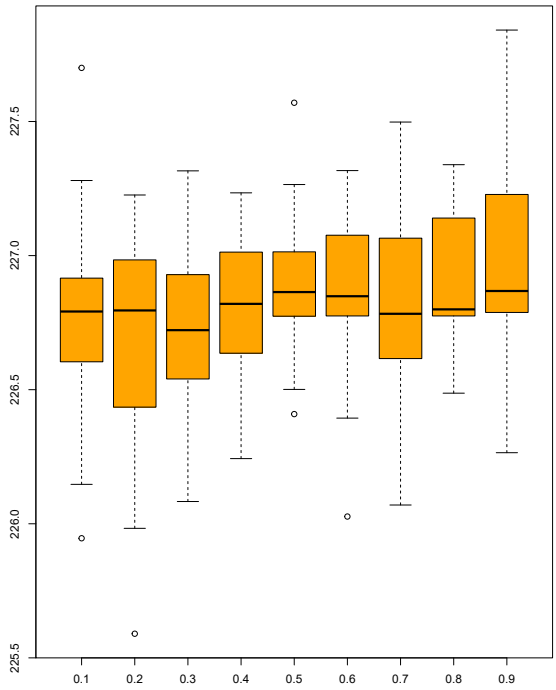
vjer. mutacije	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
min	225.95	225.59	226.08	226.24	226.41	226.03	226.07	226.49	226.27
max	227.70	227.23	227.32	227.23	227.57	227.32	227.50	227.34	227.84
medijan	226.79	226.80	226.72	226.82	226.86	226.85	226.78	226.80	226.87
prosjek	226.77	226.70	226.73	226.79	226.88	226.86	226.80	226.90	226.96
stdev	0.35	0.38	0.29	0.26	0.25	0.28	0.34	0.24	0.33

Iz tablice 6.10 i slike 6.9 se može zaključiti da je najbolje odabrati vjerojatnost mutacije jednaku 0.3 jer se za tu vrijednost postiže najmanji medijan i podaci su puno manje raspršeni nego kod npr. vrijednosti od 0.2 za koju se postiže najbolji minimum.

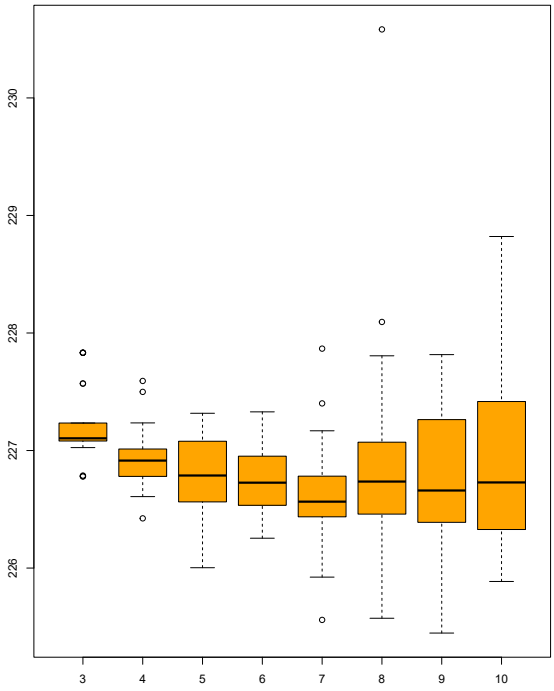
Pretposljednji parametar za koji treba odrediti vrijednost je maksimalna dubina stabla. Za



Slika 6.8: Kutijasti dijagrami za veličine populacija iz skupa {250, 500, 750, 1000, 1250, 1500} u okolini jednog stroja



Slika 6.9: Kutijasti dijagrami za vjerojatnost mutacije iz skupa vrijednosti 0.1 - 0.9 u okolini jednog stroja

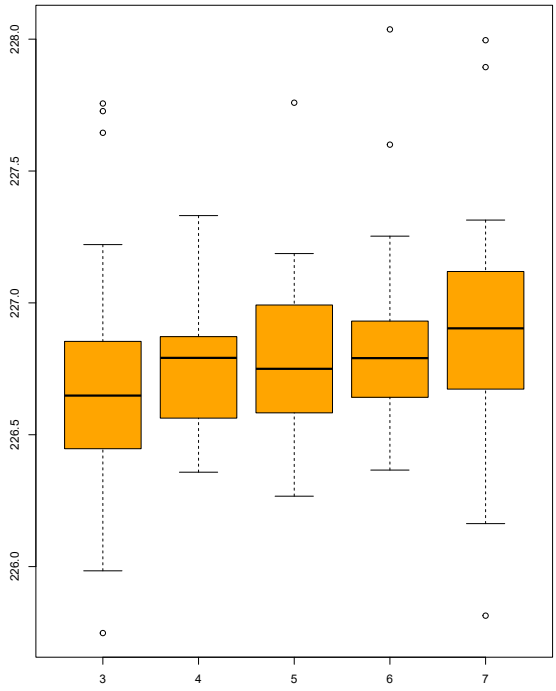


Slika 6.10: Kutijasti dijagrami za maksimalnu dubinu stabla iz skupa vrijednosti 3 - 10 u okolini jednog stroja

taj parametar vrijednosti se odabiru iz skupa {3, 4, 5, 6, 7, 8, 9, 10}. Tablica 6.11 prikazuje minimum, maksimum, srednju vrijednost, medijan i standardnu devijaciju dobivene pokretanjem GP-a sa odgovarajućim parametrima 30 puta, dok slika 6.10 daje kutijaste dijagrame koji prikazuju izgled dobivenih vrijednosti za odgovarajuće parametre. Za ovaj parametar je zaključeno da je najbolje odabrati vrijednost 7. Razlog tomu je što su i minimalna vrijednost i medijan postigli najbolje vrijednosti upravo za maksimalnu dubinu stabla jednaku 7. Kutijasti dijagrami sa slike 6.10 dodatno potkrjepljuju taj izbor jer se na njima vidi da je za vrijednost 7 raspršenost dobivenih vrijednosti funkcije dobrote najmanja.

Tablica 6.11: Vrijednosti funkcije dobrote za maksimalnu dubinu stabla iz skupa vrijednosti 3 - 10 u okolini jednog stroja

max dubina stabla	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
min	226.78	226.42	226.00	226.25	225.56	225.57	225.45	225.89
max	227.83	227.59	227.32	227.33	227.87	230.58	227.82	228.82
medijan	227.10	226.91	226.79	226.73	226.57	226.74	226.66	226.73
prosjek	227.22	226.92	226.78	226.75	226.62	226.89	226.80	226.89
stdev	0.30	0.25	0.33	0.25	0.44	0.87	0.57	0.73



Slika 6.11: Kutijasti dijagrami za veličinu turnira iz skupa vrijednosti 3 - 7 u okolini jednog stroja

Posljednji parametar za koji treba odrediti odgovarajuću vrijednost je veličina turnira. Veličina turnira se bira iz skupa vrijednosti {3, 4, 5, 6, 7}. Tablica 6.12 daje minimum, maksimum, medijan, srednju vrijednost i standardnu devijaciju vrijednosti funkcije dobrote dobivene u 30 pokretanja GP-a sa odgovarajućim vrijednostima promatranog parametra. Najbolji minimum i medijan se dobiju u eksperimentima u kojima se koristi veličina turnira 3, a iz kutijastih dijagrama sa slike 6.11 je također vidljivo da je vrijednost 3 najbolji izbor za ovaj parametar.

Tablica 6.12: Vrijednosti funkcije dobrote za veličinu turnira iz skupa vrijednosti 3 - 7 u okolini jednog stroja

veličina turnira	**3**	**4**	**5**	**6**	**7**
min	225.748	226.358	226.267	226.366	225.814
max	227.756	227.331	227.759	228.037	227.996
medijan	226.6485	226.7915	226.75	226.7905	226.9035
prosjek	226.7105	226.7406	226.8059	226.847	226.8781
stdev	0.460254	0.236806	0.306223	0.355977	0.44426

Tablica 6.13 daje prikaz završnih parametara za GP dobivenih opisanim postupkom. Kao što je već napomenuto, ovi parametri će se koristiti za pokretanje GP-a na odgovarajućim testnim primjerima u dobivenim klasterima kako bi se dobili rezultati koji će se zatim usporediti s

```
# name          switch              type      values
tsize           "--tsize "            i       (3,7)
maxdepth        "--maxdepth "         i       (3,10)
popsize         "--popsize "          i       (200,1500)
mutProb         "--mutprob "          r       (0.01, 0.99)
```

Slika 6.12: Parametri za *irace* u okolini jednog stroja

vrijednostima dobivenim pokretanjem GP-a s parametrima dobivenim korištenjem okruženja *irace* za odgovarajući klaster.

Tablica 6.13: Parametri za GP u okruženju jednog stroja, dobiveni ručnim pretraživanjem prostora parametara

naziv parametra	**vrijednost**
broj generacija	30
veličina populacije	1250
vjerojatnost mutacije	0.3
maksimalna dubina stabla	7
veličina turnira	3

Nakon ručnog određivanja parametara za GP, potrebno je u svakom klasteru dobivenom kao što je opisano u prethodnom odjeljku, korištenjem okruženja *irace* odrediti parametre za GP. Kako bi se *irace* mogao koristiti, potrebno je proslijediti odgovarajuće parametre. Prvo treba proslijediti koji je maksimalan broj eksperimenata. Taj broj označava koliko se puta najviše, prilikom određivanja optimalnih parametara, smije pozvati proslijeđeni algoritam za koji se određuju parametri. U okolini jednog stroja maksimalan broj elemenata je jednak $600/broj\_klastera$. Isto tako, potrebno je odrediti hoće li se koristiti elitizam i koliko konfiguracija se prenosi u sljedeću iteraciju *irace*-a ukoliko se on koristi. U ovom okruženju se koristi elitizam i samo se najbolja konfiguracija prenosi u sljedeću iteraciju. Zatim je potrebno definirati datoteku u kojoj će biti nazivi parametara za algoritam koji se promatra, tipovi tih parametara i vrijednosti iz kojih ti parametri mogu doći. Za okolinu jednog stroja, prikaz takve datoteke je dan na slici 6.12.

U prvom stupcu se nalazi naziv parametra, u drugom oznaka kojom se taj parametar može dohvatiti unutar *irace*-a, treći stupac označava tip određenog parametra i četvrti vrijednosti koje određeni parametar može poprimiti. Prvi parametar naveden ovdje je *tsize* koji predstavlja veličinu turnira. Oznaka *i* u trećem stupcu označava da je ovo cjelobrojni tip podatka, dok oznaka $(3,7)$ označava da veličina turnira može poprimiti cjelobrojne vrijednosti između 3 i 7 (uključujući i te dvije vrijednosti). Maksimalna dubina stabla je označena sa *maxdepth*, također poprima cjelobrojne vrijednosti i to između 3 i 10. Oznaka *popsize* predstavlja veličinu

```
(popsize > 1300) & (maxdepth > 5)
(popsize > 1200) & (maxdepth > 6)
(popsize > 1100) & (maxdepth > 7)
(popsize > 1000) & (maxdepth > 8)
```

Slika 6.13: Zabranjene konfiguracije parametara za *irace* u okolini jednog stroja

populacije koja može poprimiti cjelobrojne vrijednosti iz raspona od 200 do 1500. Posljednji parametar koji *irace* treba odrediti je vjerojatnost mutacije. Oznaka za taj parametar je *mutProb* i on može poprimiti vrijednosti između 0.01 i 0.99. Oznaka *r* u trećem stupcu označava da je ovaj parametar realan broj. U *irace*-u se dodatno može odabrati na koliko decimala će se uzimati realni parametri i ovdje je odabrano da će se koristiti samo dvije decimale.

U jednoj skupini eksperimenata se i broj generacija proslijedio kao parametar za *irace*, ali je utvrđivanje tog parametra samo produžilo vrijeme izvršavanja, a nije dalo bolje rezultate nego kad se taj parametar fiksira na vrijednost od 30 generacija.

Dodatno, *irace* može primiti i listu zabranjenih konfiguracija ukoliko neka kombinacija parametara nije dopustiva, ako unaprijed znamo da neka konfiguracija neće dati zadovoljavajuće rješenje, ako znamo da će neka kombinacija parametara zahtjevati previše resursa za izvršavanje, itd. U ovoj okolini su korištene zabranjene konfiguracije i njihova je lista dana na slici 6.13.

U zabranjenim konfiguracijama su se našle one kombinacije veličine populacije i maksimalne dubine stabla koje zauzimaju previše računalnih resursa, pa jako usporavaju ili čak onemogućuju daljnji rad sustava.

Uz već navedene parametre: broj generacija, veličinu populacije, vjerojatnost mutacije, maksimalnu dubinu stabla i veličinu turnira, za uporabu GP-a potrebno je definirati i koji će se operatori mutacije i križanja koristiti u eksperimentima. U okolini jednog stroja, za operator križanja se koristi križanje podstabala, pri čemu je vjerojatnost da se za točku križanja odabere funkcijski čvor jednaka 90%. Za operatore mutacije se koriste mutacija podstabla, mutacija zamjenom čvora i mutacija smanjivanjem (kao što je navedeno u odjeljku 4.6), sa jednakom vjerojatnoću izbora bilo koje od te tri mutacije.

Kao što je već rečeno, za svako grupiranje temeljem neke značajke prikazano u prethodnom odjeljku, i za svaki klaster u tom grupiranju, *irace* je uz prethodno navedene parametre odredio parametre za GP. Zatim se u svakom klasteru GP pokrenuo na skupu za učenje 30 puta s ručno određenim parametrima i 30 puta s parametrima dobivenim okruženjem *irace*. Pravila koja su naučena tim postupcima su zatim evaluirana na skupu za testiranje za odgovarajući klaster i dobiveni rezultati će biti dani u nastavku.

Prvo grupiranje je rađeno na temelju prosjeka mjere temeljene na udaljenosti. Tablica 6.14 daje prikaz veličine populacije, vjerojatnosti mutacije, maksimalne dubine stabla i veličine tur-

nira koje je *irace* naučio za svaki od 3 dobivena klastera.

Tablica 6.14: Parametri za GP dobiveni *irace*-om za svaki od klastera temeljen na prosjeku mjere temeljene na udaljenosti u okolini jednog stroja

parametar	klaster1	klaster2	klaster3
veličina populacije	1492	1444	660
vjerojatnost mutacije	0.44	0.02	0.49
max dubina stabla	5	5	6
veličina turnira	7	7	5

Iako su vrijednosti veličine populacije, maksimalne dubine stabla i veličina turnira jednake ili gotovo jednake za prvi i drugi klaster, vjerojatnost mutacije se znatno razlikuje. Možda je to indikator da se u drugom klasteru nalazi manje lokalnih optimuma, pa GP nema potrebe koristiti mutaciju kako bi izašao iz lokalnog optimuma ukoliko se nađe u njemu u tijeku pretraživanja prostora stanja. Iz tablice 6.5 se vidi da u prvom klasteru u skupu za učenje ima više instanci nego u skupu za učenje drugog klastera, pa nije čudno da se u prvom klasteru može pronaći više lokalnih optimuma iz kojih onda GP većom vjerojatnošću mutacije treba izaći ako u njima zapne. Za treći klaster je *irace* odredio najmanju veličinu populacije, što automatski rezultira najkraćim vremenom izvršavanja, pa samo određivanje pravila GP-om traje kraće nego određivanje parametrima koji su određeni ručno. Vidljivo je da je *irace* za različite klastere rezultirao različitim vrijednostima parametara što znači da postoje određene razlike u instancama koje se nalaze u dobivenim klasterima.

Tablica 6.15 daje osnovne statističke mjere (minimum, prvi kvartil, medijan, aritmetičku sredinu, treći kvartil, maksimum i standardnu devijaciju) vrijednosti funkcije dobrote dobivene primjenom prethodno evoluiranih pravila na skupu za ispitivanje određenog klastera s odgovarajućim parametrima. Dodatno, u tablici je dana i p-vrijednost dobivena Mann - Whitney - Wilcoxonovim (MWW) statističkim testom. Ovaj test uzima u obzir dva uzorka i procjenjuje medijan razlike između uzorka dobivenog iz prvog vektora podataka i uzorka dobivenog iz drugog vektora podataka. MWW je neparametarski test koji ne zahtjeva da ulazni podaci dolaze iz normalne razdiobe. Nul - hipoteza u danom testu je da je medijan razlike između dva promatrana uzorka jednak nuli.

Iz dane tablice se može primijetiti da u sva tri dobivena klastera vrijednosti funkcije dobrote dobivene korištenjem GP-a s parametrima koje je dao *irace* imaju manji medijan i minimum nego vrijednosti funkcije dobrote dobivene korištenjem GP-a s ručno određenim parametrima. Dodatno, u sva tri slučaja je standardna devijacija manja u stupcu koji daje rezultate korištenjem parametara iz *irace*-a, što znači da su ovi parametri više prilagođeni klasterima nego parametri određeni ručno, pa je i manja raspršenost dobivenih rješenja. Slika 6.14 daje usporedbu kutijas-

Tablica 6.15: Usporedba vrijednosti funkcije dobrote za klastere temeljene na prosjeku mjere temeljene na udaljenosti u okolini jednog stroja

	klaster 1		klaster 2		klaster 3	
	ručno	irace	ručno	irace	ručno	irace
min	5.7855	5.7766	1984.2700	1980.2300	68.0418	67.7454
1. kv.	5.9196	5.8265	1987.7725	1987.2875	68.9759	69.1241
medijan	5.9542	5.9212	1989.8850	1988.6400	70.0290	69.9288
prosjek	5.9766	5.9114	1996.1197	1990.5907	69.9437	69.7885
3. kv.	6.0337	5.9463	1996.6100	1990.7050	70.6530	70.5151
max	6.1838	6.0935	2039.4900	2010.2000	71.7920	72.2765
stdev	0.1016	0.0865	15.4478	6.8721	1.1421	1.1247
p-vrijednost	0.006105		0.129		0.2413	

tih dijagrama za vrijednosti funkcije dobrote dobivene korištenjem ručno određenih parametara za GP i parametara dobivenih korištenjem *irace*-a.

U okolini jednog stroja je grupiranje uz prosjek mjere temeljene na udaljenosti rađeno i na temelju maksimuma mjere temeljene na udaljenosti bez nula. U nastavku će biti dani rezultati za tu mjeru. Tablica 6.16 daje vrijednost parametara za GP dobivenih korištenjem *irace*-a u svakom od tri dobivena klastera.

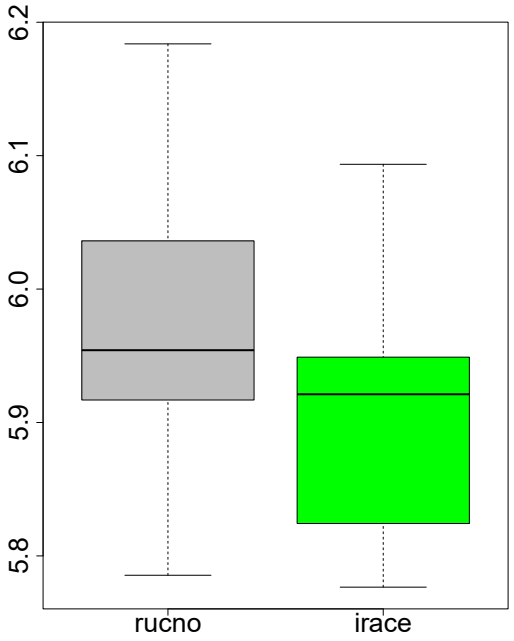
Tablica 6.16: Parametri za GP dobiveni *irace*-om za svaki od klastera temeljen na maksimumu mjere temeljene na udaljenosti bez nula u okolini jednog stroja

parametar	klaster1	klaster2	klaster3
veličina populacije	850	955	1141
vjerojatnost mutacije	0.46	0.87	0.97
max dubina stabla	5	3	4
veličina turnira	6	8	7

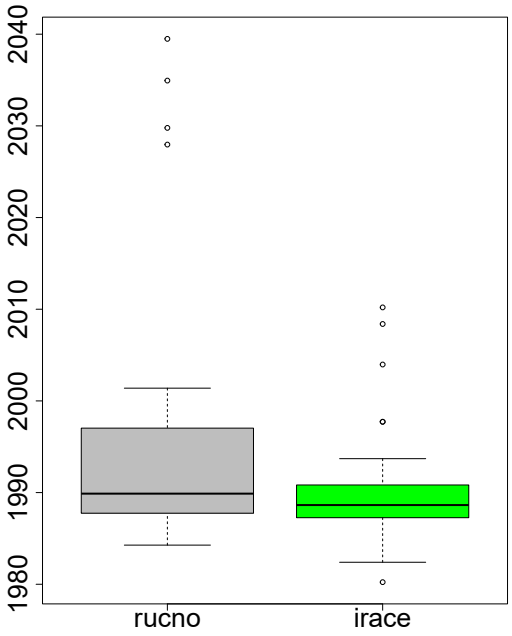
U ovom slučaju su parametri različiti za svaki od klastera, što znači da je grupiranje na temelju ove značajke krajolika dobrote uspjelo rasporediti instance problema u klastere koji se razlikuju. Može se uočiti i da je u sva tri klastera vjerojatnost mutacije visoka, što znači da je *irace* u postupku pretraživanja prostora rješenja zaključio da postoji potreba za izlaskom iz lokalnih optimuma kada se jednom u njima zaglavi.

Tablica 6.17 daje osnovne statističke mjere vrijednosti funkcije dobrote dobivene pokretanjem GP-a s ručno određenim i s automatski određenim parametrima.

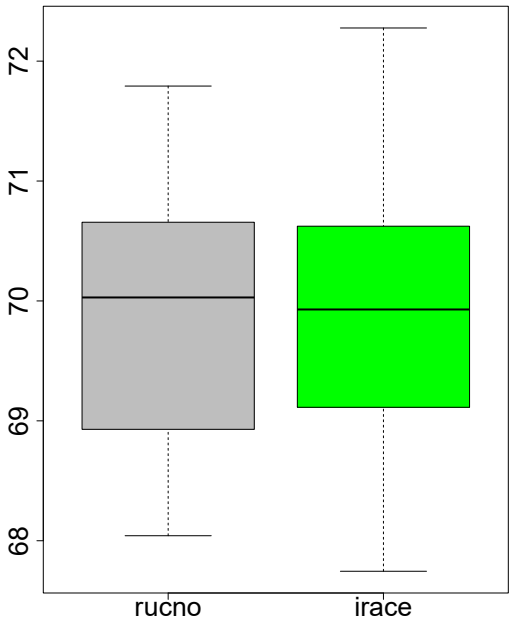
Iz dane tablice može se uočiti da su u prvom i trećem klasteru vrijednosti medijana bolje



(a) klaster 1



(b) klaster 2



(c) klaster 3

Slika 6.14: Prikaz kutijastih dijagrama dobivenih na temelju aritmetičke sredine i standardne devijacije prosjeka mjere temeljene na udaljenosti u okolini jednog stroja

Tablica 6.17: Usporedba vrijednosti funkcije dobrote za klastere temeljene na maksimumu mjere temeljene na udaljenosti bez nula u okolini jednog stroja

	klaster 1		klaster 2		klaster 3	
	ručno	irace	ručno	irace	ručno	irace
min	1.1590	1.1828	2020.4300	2022.9000	34.7672	34.7283
1. kv.	1.2357	1.2282	2024.1825	2026.2575	35.0685	35.1283
medijan	1.2878	1.2562	2028.2050	2030.2200	35.4068	35.2731
prosjek	1.2950	1.2516	2036.9843	2037.4463	35.4420	35.4733
3. kv.	1.3446	1.2841	2034.9375	2047.7800	35.7262	35.7781
max	1.5156	1.3475	2229.9300	2069.9500	36.3611	37.1530
stdev	0.0794	0.0458	37.4135	14.0066	0.4688	0.5657
p-vrijednost	0.0101		0.9614		0.4735	

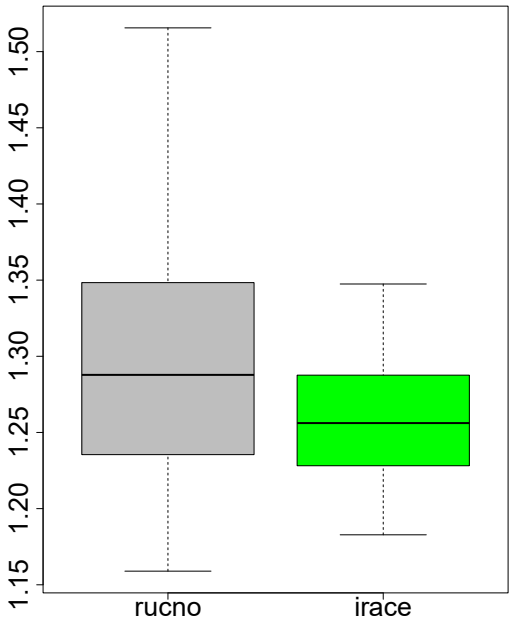
za vrijednosti funkcije dobrote dobivene korištenjem GP-a s automatski određenim parametrima, dok je u drugom klasteru vrijednost medijana malo bolja za vrijednosti funkcije dobrote dobivene korištenjem GP-a s ručno određenim parametrima. No, u slučajevima kada je *irace* dao bolje rezultate, poboljšanje u medijanu je 2.52% u prvom klasteru i 0.38% u trećem klasteru, dok je pogoršanje medijana u drugom klasteru tek 0.099%. Dakle, poboljšanje dobiveno korištenjem automatske konfiguracije parametara je veće nego pogoršanje, stoga ovi rezultati pokazuju da se korištenjem automatske konfiguracije parametara u klasterima može dobiti bolji rezultat, što i je cilj korištenja hiperheuristika. Odgovarajuće usporedbe kutijastih dijagrama za ovu značajku su dane na slici 6.15.

6.5 Rezultati u okolini nesrodnih strojeva

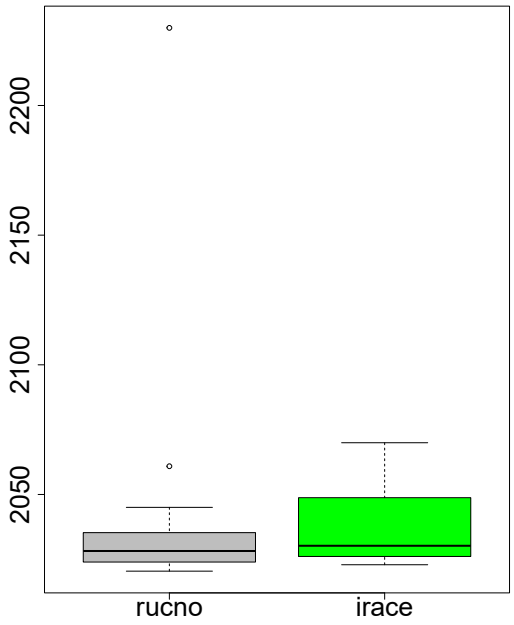
U ovom odjeljku će biti prikazani rezultati eksperimenata za okolinu nesrodnih strojeva. Kao i za okolinu jednog stroja, i ovdje će prvo biti prikazan postupak grupiranja i na koji način su se odabrali parametri na temelju kojih se radi grupiranje, a nakon toga će biti prikazani rezultati dobiveni automatskom i ručnom konfiguracijom parametara za pojedini klaster te će se učiniti njihova usporedba.

6.5.1 Grupiranje

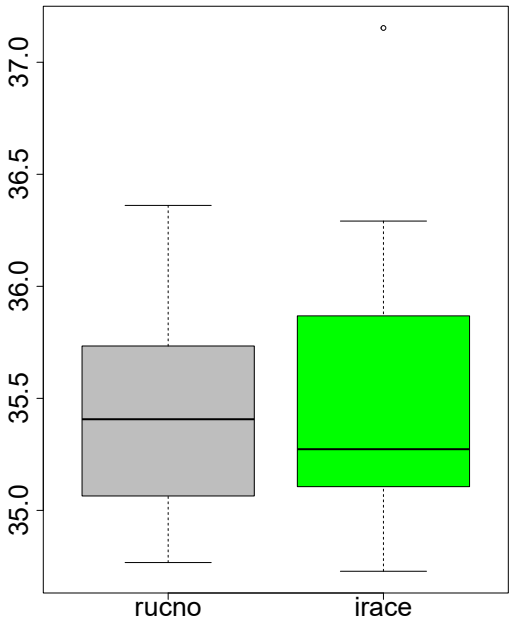
Kod okoline nesrodnih strojeva, kao ni kod okoline jednog stroja, u klasterima dobivenim algoritmom EM, automatska konfiguracija parametara nije proizvela bolje rezultate od parametara određenih ručno. Iz tog razloga se i u ovoj okolini grupiranje radilo koristeći algoritam k-means.



(a) klaster 1

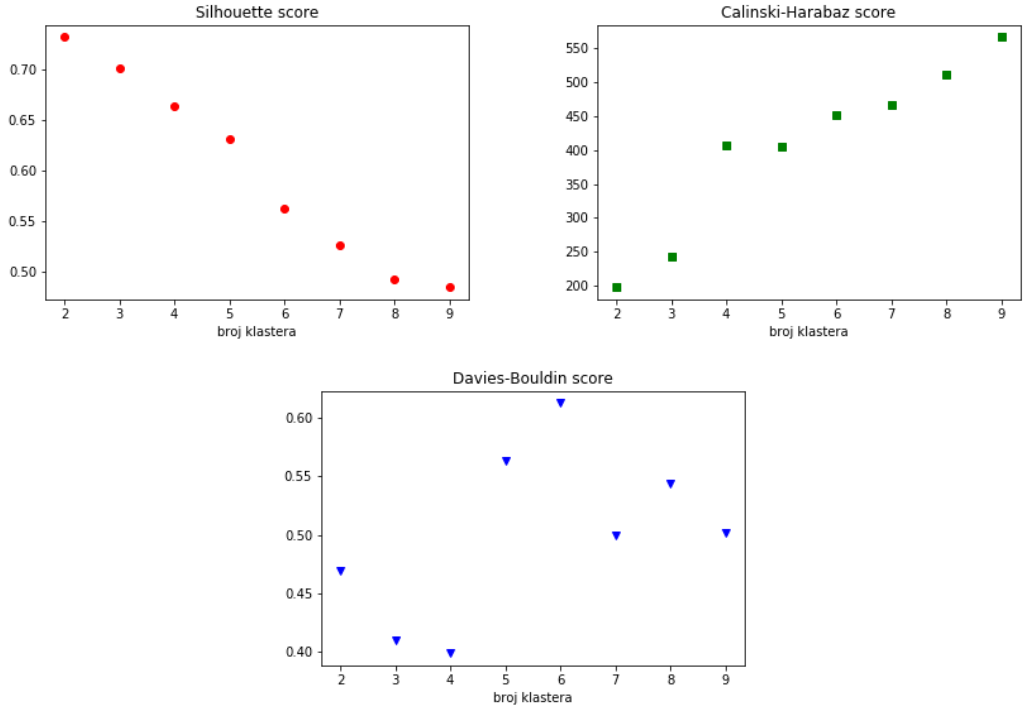


(b) klaster 2



(c) klaster 3

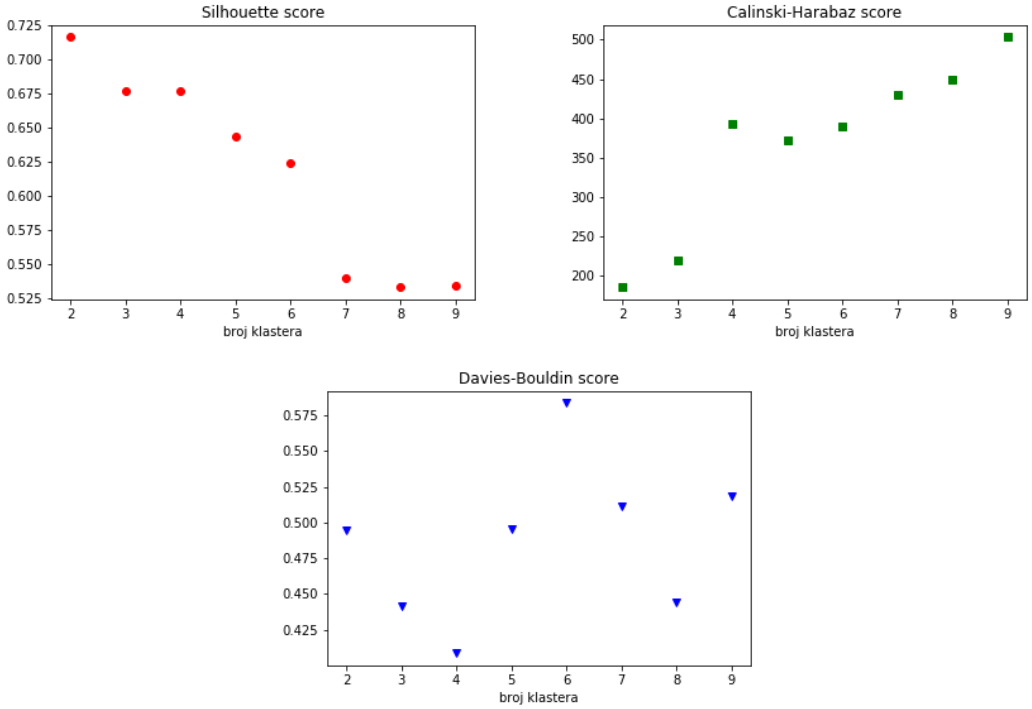
Slika 6.15: Prikaz kutijastih dijagrama dobivenih na temelju aritmetičke sredine i standardne devijacije maksimuma mjere temeljene na udaljenosti bez nula u okolini jednog stroja



Slika 6.16: Prikaz dobivenih mjera za grupiranje temeljeno na gornjem kvartilu mjere temeljene na udaljenosti bez nula u okolini nesrodnih strojeva

S obzirom da je ovdje definirano samo 60 instanci u skupu za učenje, nisu se promatrali brojevi grupa iz intervala $[2, 24]$, nego $[2, 9]$. Prvi skup grupiranja je na temelju *silhouette*, Davis - Bouldin te Calinski i Harabasz mjere odredio da je najbolje uzeti 4 klastera i to za gornji kvartil mjere temeljene na udaljenosti bez nula i za prosjek mjere temeljene na udaljenosti. Slike 6.16 i 6.17 prikazuju dobivene mjere za navedene dvije značajke. U oba slučaja vrijednost mjere Calinski Harabsz se poveća prelaskom sa 3 na 4 klastera, mjera Davis Bouldin je najniža za 4 klastera, dok u mjeri *silhouette* vrijednost za 4 klastera nije puno lošija od vrijednosti u 2 i 3 klastera.

No, za grupiranja kojima su rezultirale ove dvije mjere, rezultati postignuti automatskom konfiguracijom parametara nisu bolji od rezultata postignutih korištenjem ručno odabranih parametara u GP-u. Ukoliko se usporede značajke korištene u okolini jednog stroja, gdje ovaj postupak rezultira boljim vrijednostima funkcije cilja u slučaju automatske konfiguracije parametara za pojedinu grupu, vidi se da u okolini nesrodnih strojeva postoje i značajke koje se prilikom evaluacije jedinke mijenjaju u ovisnosti o tome u kojem se vremenu izvršavanja trenutno nalazi sustav, čega u okolini jednog stroja nema. Iz tog razloga, postupak grupiranja je ponovljen, ali prilikom stvaranja početnih 30 jedinki i slučajnih šetnji iz njih, korištene su samo značajke *pt*, *dd*, *w*, *SL*, *pmin*, *pavg* te vrijednosti 0 i 1. Kod ovako definiranih jedinki, algoritmom k-means se dobilo da je najbolje grupirati instance u 6 klastera na temelju standardne devijacije mjere temeljene na udaljenosti bez nula te u 4 klastera na temelju stabilnosti informacije.



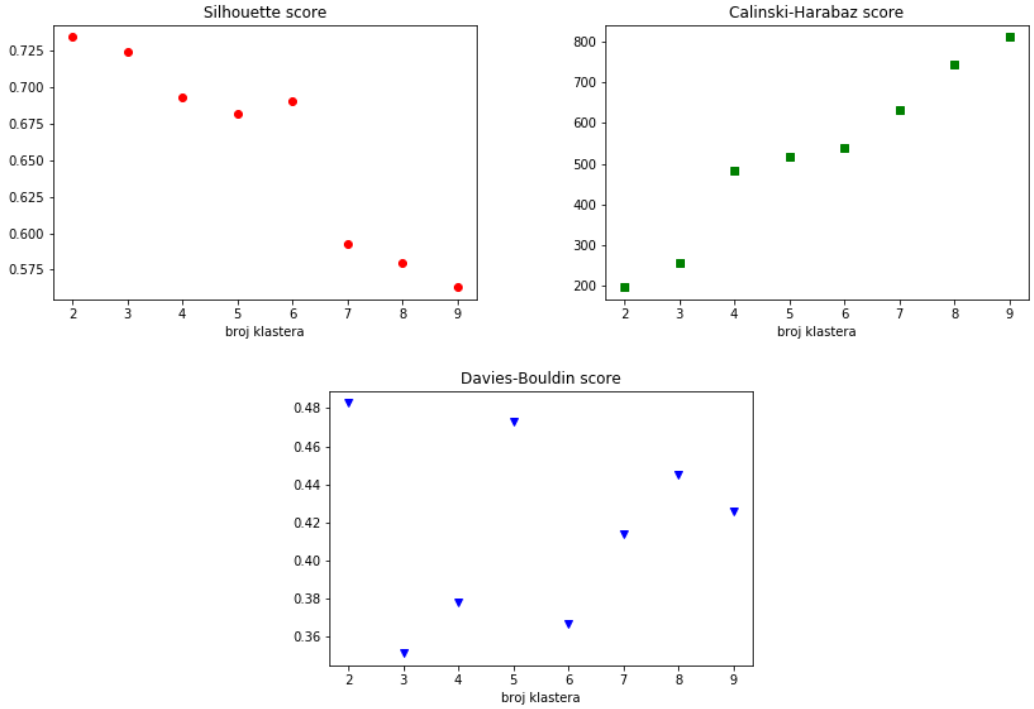
Slika 6.17: Prikaz dobivenih mjera za grupiranje temeljeno na prosjeku mjere temeljene na udaljenosti u okolini nesrodnih strojeva

Slika 6.18 prikazuje dobivene mjere za standardnu devijaciju mjere temeljene na udaljenosti bez nula. Sa slike je vidljivo da je mjera Davis - Bouldin podjednako dobra za 3 i 6 klastera, mjera *silhouette* je bolja za 3 klastera, iako se dobivena vrijednost ne razlikuje puno od vrijednosti za 6 klastera, no mjera Calinski - Harabasz pokazuje da je vrijednost za 6 klastera znatno bolja nego za 3 klastera, pa se iz tog razloga u ovom slučaju odabralo grupiranje u 6 klastera.

Tablica 6.18 daje prikaz broja instanci u skupu za učenje i skupu za testiranje koje su se našle u pojedinom klasteru za standardnu devijaciju mjere temeljene na udaljenosti bez nula. Za razliku od eksperimenata u okruženju jednog stroja, ovdje je vidljivo da su se instance u podjednakom broju rasporedile u klastere i u skupu za učenje i u skupu za testiranje. Odnosno, u okruženju nesrodnih strojeva, korištenjem standardne devijacije mjere temeljene na udaljenosti bez nula dobija se grupiranje koje na jednak način raspoređuje i instance iz skupa za učenje i instance iz skupa za testiranje.

Slika 6.19 daje prikaz raspona vrijednosti za standardnu devijaciju i aritmetičku sredinu standardne devijacije mjere temeljene na udaljenosti bez nula. Na oba dijagrama je vidljivo da je većina vrijednosti niža dok se maksimalne vrijednosti pojavljuju više kao stršeće vrijednosti. Stoga ne čude predstavnici iz tablice 6.18, gdje predstavnik s najmanjim vrijednostima aritmetičke sredine i standardne devijacije ima najviše instanci u svom klasteru.

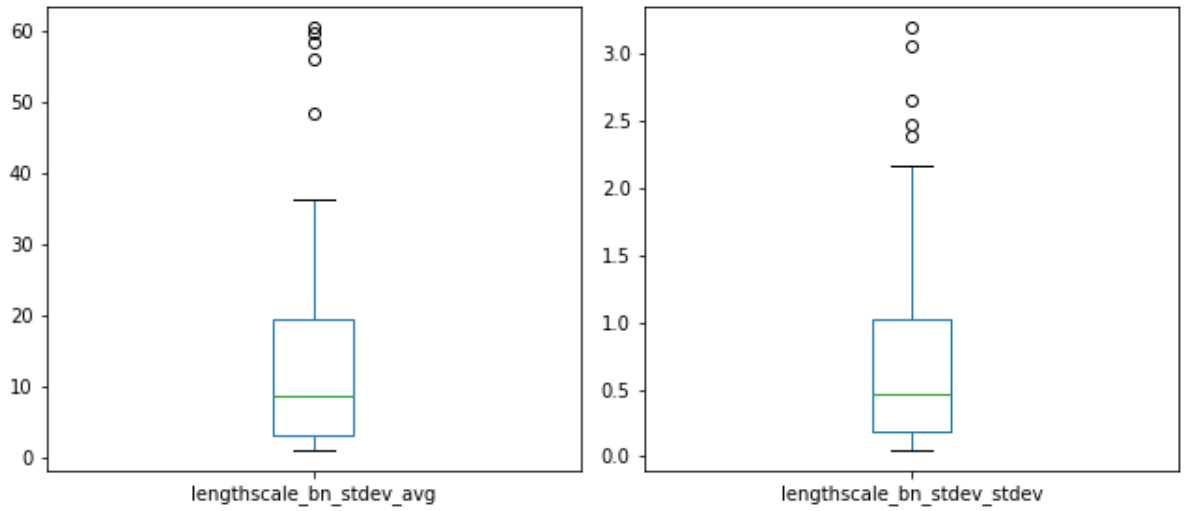
Slika 6.20 daje prikaz dijagrama raspršenosti za standardnu devijaciju i aritmetičku sredinu standardne devijacije mjere temeljene na udaljenosti bez nula. I ovdje je vidljivo da se najbrojniji klasteri nalaze na nižim vrijednostima aritmetičke sredine i standardne devijacije. Kao i kod



Slika 6.18: Prikaz dobivenih mjera za grupiranje temeljeno na standardnoj devijaciji mjere temeljene na udaljenosti bez nula u okolini nesrodnih strojeva

Tablica 6.18: Broj instanci u klasterima i predstavnici klastera za standardnu devijaciju mjere temeljene na udaljenosti bez nula u okruženju nesrodnih strojeva

	klaster1	**klaster2**	**klaster3**
# instanci (skup za učenje)	30	2	3
# instanci (skup za testiranje)	24	2	2
predstavnik klastera	(3.7801, 0.2072)	(35.8467, 2.1542)	(54.3660, 2.5073)
	klaster4	**klaster5**	**klaster6**
# instanci (skup za učenje)	15	8	2
# instanci (skup za testiranje)	21	8	3
predstavnik klastera	(13.5121, 0.7013)	(29.6890, 1.4994)	(60.1904, 3.1326)



Slika 6.19: Kutijasti dijagram vrijednosti standardne devijacije mjere temeljene na udaljenosti bez nula za okruženje nesrodnih strojeva

okoline jednog stroja, i ovdje su aritmetička sredina i standardna devijacija promatrane značajke pozitivno korelirane, ali se sa prikazanog dijagrama raspršenosti može uočiti da su klasteri 3, 4, 5 i 6 fizički poprilično dobro razdvojeni.

Nadalje, slika 6.21 prikazuje mjere grupiranja dobivene za sadržaj informacije.

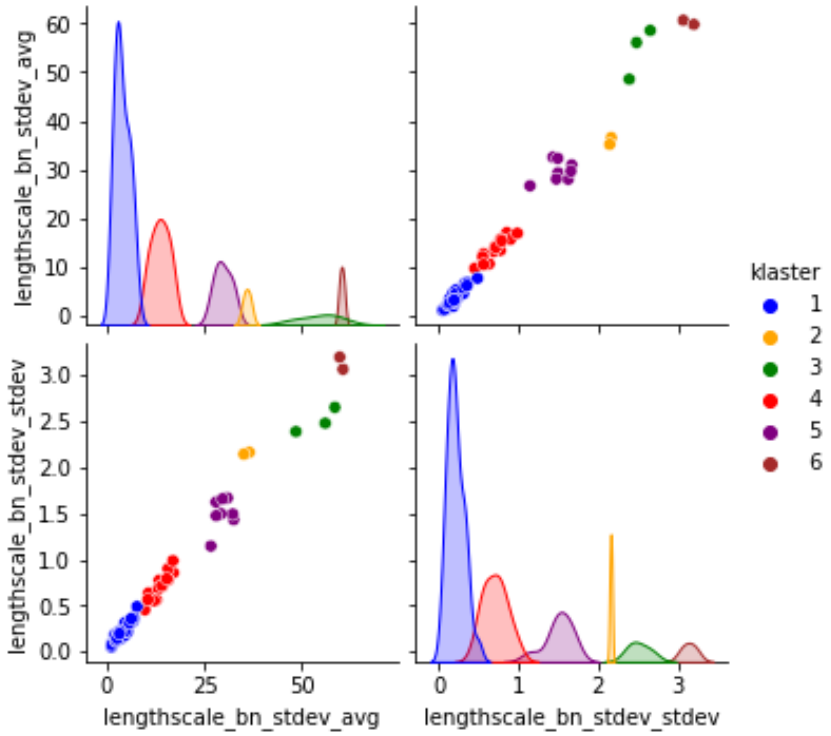
Sa slike je vidljivo da su za 3 i 4 klastera vrijednosti mjere Davies - Bouldin jednake, kod mjere *silhouette* je vrijednost malo bolja za 3 nego za 4 klastera, dok je kod mjere Calinski Harabazs vrijednost bolja za 4 nego za 3 klastera. No, razlika u mjeri *silhouette* je puno manja nego razlika u mjeri Calinski Harabazs, pa se iz tog razloga odabralo 4 klastera za grupiranje instanci.

Tablica 6.19 daje prikaz broja instanci u skupu za učenje i skupu za testiranje koje pripadaju pojedinom klasteru na temelju stabilnosti informacije. Kao i kod prethodne značajke, i ovdje se u svakom klasteru nalazi podjednak broj instanci problema u skupu za učenje i skupu za testiranje.

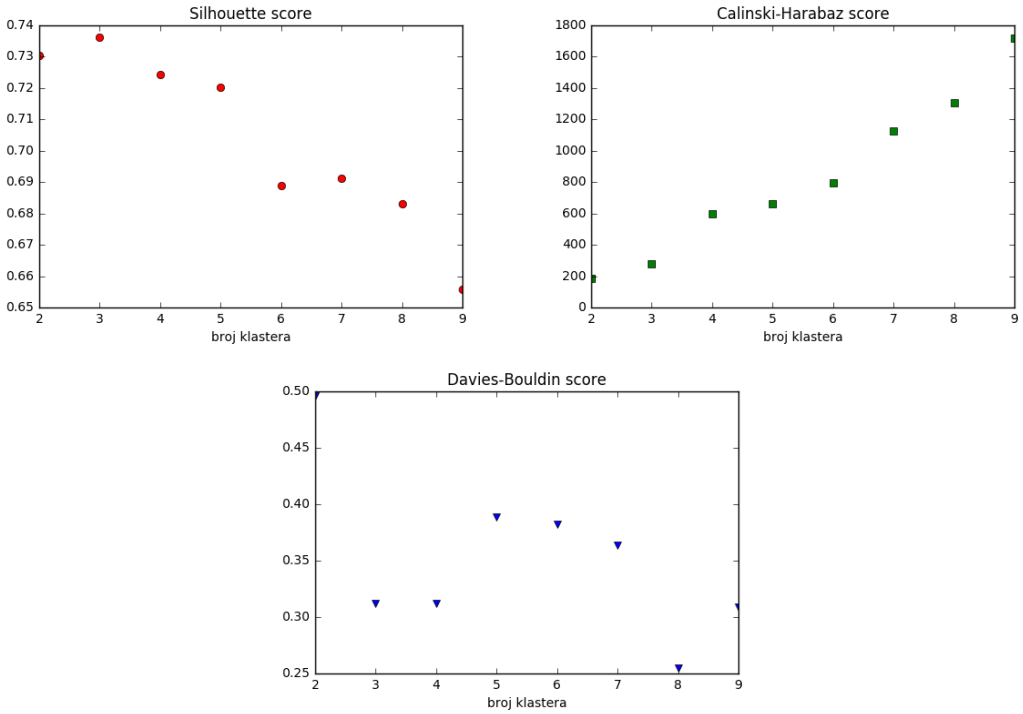
Tablica 6.19: Broj instanci u klasterima i predstavnici klastera za stabilnost informacije u okruženju nesrodnih strojeva

	klaster1	**klaster2**	**klaster3**	**klaster4**
# instanci (skup za učenje)	29	10	5	16
# instanci (skup za testiranje)	22	10	5	23
predstavnik klastera	(42.3269, 0.0)	(327.2180, 0.0)	(597.1180, 0.0)	(143.2213, 0.0)

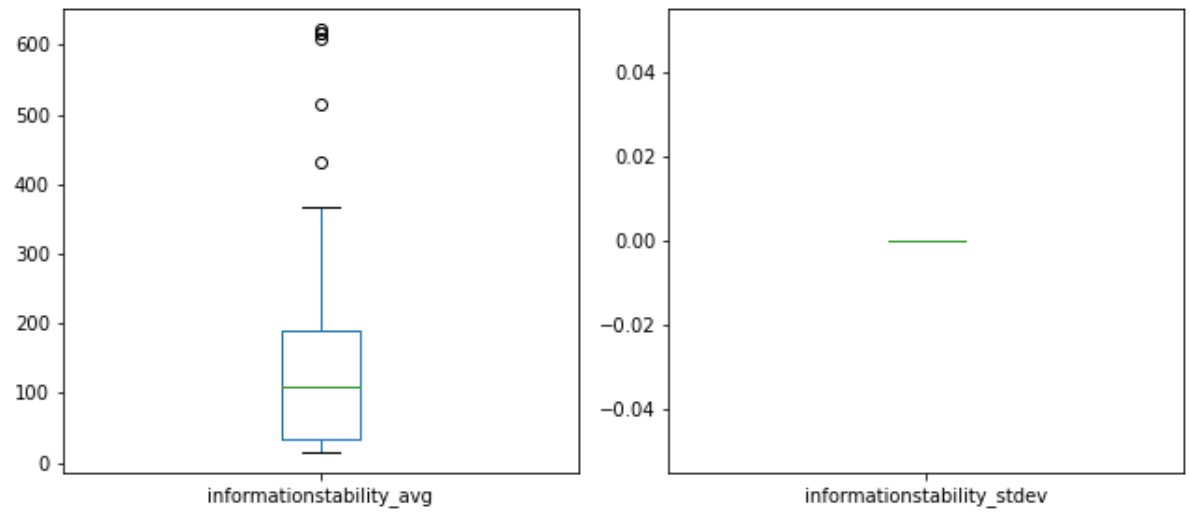
Kao i u većini ostalih značajki, najviše instanci problema nalazi se u klasteru u kojem predstavnik ima najmanju vrijednost aritmetičke sredine i standardne devijacije promatranih značajki. Ukoliko se promatra raspon vrijednosti za aritmetičku sredinu i standardnu devijaciju



Slika 6.20: Dijagram raspršenja za standardnu devijaciju mjere temeljene na udaljenosti bez nula u okruženju nesrodnih strojeva



Slika 6.21: Prikaz dobivenih mjera za grupiranje temeljeno na stabilnosti informacije u okolini nesrodnih strojeva



Slika 6.22: Kutijasti dijagram vrijednosti stabilnosti informacije za okruženje nesrodnih strojeva

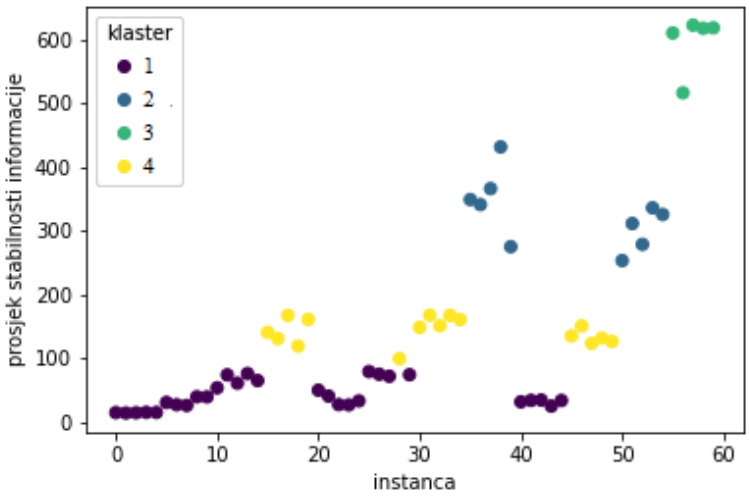
stabilnosti informacije na slici 6.22, ne čudi da je najviše elemenata u klasteru s najmanjim predstavnikom s obzirom da se većina vrijednosti aritmetičke sredine stabilnosti informacije nalazi bliže minimumu nego maksimumu. Sa dane slike se može uočiti i da su sve vrijednosti standardne devijacije promatrane značajke jednake nuli, pa je ovdje grupiranje rađeno zapravo samo na temelju jednog parametra i to aritmetičke sredine stabilnosti informacije.

Na kraju, slika 6.23 daje prikaz dijagrama raspršenosti za aritmetičku sredinu stabilnosti informacije. Na x-osi se nalazi redni broj instance, dok je na y-osi vrijednost aritmetičke sredine stabilnosti informacije. Sa slike je jasno vidljivo da su u prvom klasteru one instance koje imaju najniže vrijednosti promatrane značajke, u drugom su one instance s najvećim vrijednostima promatrane značajke (njih je ujedno i najmanje), a u sredini su se značajke rasporedile na klaster 3 i 4. Može se uočiti da su klaster 1 i klaster 4 dosta bliski, dok su preostala dva klastera dosta dobro razdvojena međusobno i od prva dva klastera.

6.5.2 Određivanje parametara

Kao i kod okoline jednog stroja, i u okolini nesrodnih strojeva je potrebno odrediti parametre za GP koji su najbolji za cijeli skup za učenje kako bi se vrijednosti dobivene korištenjem tih parametara za GP mogle usporediti s vrijednostima dobivenim parametrima kojima rezultira *irace* za odgovarajući klaster. S obzirom da su instance problema za ovu okolinu preuzete iz [169], a u tom radu su već određeni najbolji parametri za GP na promatranim instancama problema, ovdje su ti parametri samo preuzeti i prikazani su u tablici 6.20.

Uz ručno određene parametre, potrebno je odrediti i parametre koristeći *irace* u svakom od dobivenih klastera. Za svaki klaster, maksimalan broj ekperimenata je postavljen na $600/broj\_klastera$. Korišten je elitizam i to u obliku da se samo najbolja konfiguracija prenosi u sljedeću iteraciju. Postavke za parametre koje treba odrediti korištenjem okruženja *irace*



Slika 6.23: Dijagram raspršenja za stabilnost informacije u okolini nesrodnih strojeva

Tablica 6.20: Parametri za GP u okruženju nesrodnih strojeva, dobiveni ručnim pretraživanjem prostora parametara

naziv parametra	**vrijednost**
broj generacija	30
veličina populacije	1000
vjerojatnost mutacije	0.3
maksimalna dubina stabla	5
veličina turnira	3

```
# name          switch              type      values
tsize           "--tsize "            i       (3,7)
maxdepth        "--maxdepth "         i       (3,10)
popsize         "--popsize "          i       (200,1500)
mutProb         "--mutprob "          r       (0.01, 0.99)
```

Slika 6.24: Parametri za *irace* u okolini nesrodnih strojeva

```
(popsize > 1300) & (maxdepth > 5)
(popsize > 1200) & (maxdepth > 6)
(popsize > 1100) & (maxdepth > 7)
(popsize > 1000) & (maxdepth > 8)
```

Slika 6.25: Zabranjene konfiguracije parametara za *irace* u okolini nesrodnih strojeva

dane su na slici 6.24.

Kao i kod okoline jednog stroja, *tsize* označava veličinu turnira u turnirskom odabiru, što je cjelobrojni parametar koji poprima vrijednosti iz raspona od 3 do 7. Parametar *maxdepth* je cjelobrojni parametar koji označava maksimalnu dubinu stabla, a poprima vrijednosti iz raspona od 3 do 10. Sljedeći parametar koji *irace* treba odrediti je označen sa *popsize* i predstavlja veličinu populacije. Ovaj parametar je također cjelobrojan i poprima vrijednosti iz raspona od 200 do 1500. Posljednji parametar za GP koji se određuje pomoću *irace*-a je vjerojatnost mutacije, u oznaci *mutProb*. Ovaj parametar je realan i poprima vrijednosti iz intervala $[0.01, 0.99]$ uz točnost od dvije decimale.

I u ovoj okolini postoje zabranjene konfiguracije za *irace* te su one vidljive na slici 6.25. Ukoliko je npr. veličina populacije veća od 1200, a maksimalna dubina stabla veća od 6, dobit će se mnogo stabala velike dubine koja će zauzeti puno memorije u računalu i baratanje njima će postati otežano.

Kao i kod okoline jednog stroja, i ovdje je uz već navedene parametre potrebno definirati i koji operatori križanja i mutacije će se koristiti prilikom razvijanja prioritetnih pravila. U okolini nesrodnih strojeva se za operatore križanja koriste križanje podstabala (vjerojatnost da se odabere funkcijski čvor kao točka križanja je 90%), križanje koje čuva kontekst, križanje s pravednom veličinom te uniformno križanje. Svi navedeni operatori križanja imaju jednaku vjerojatnost biti izabrani u postupku stvaranja stabala u GP-u. Za operatore mutacije se koriste mutacija podstabla, mutacija permutacijom, Gaussova mutacija, hoist mutacija, mutacija komplementarnih čvorova, mutacija zamjenom čvorova i mutacija smanjivanjem. Jednako kao i kod operatora križanja, i ovi navedeni operatori mutacije se s jednakom vjerojatnošću mogu izabrati za modifikaciju stabala u GP-u.

U prethodnom odjeljku navedeno je da je za okolinu nesrodnih strojeva grupiranje rađeno prvo na temelju standardne devijacije mjere temeljene na udaljenosti bez nula i to grupiranje je razdvojilo instance problema u 6 klastera. U nastavku će biti prikazani rezultati dobiveni za

svaki od klastera korištenjem ručno određenih parametara i parametara određenih automatski korištenjem *irace*-a.

Tablica 6.21 daje prikaz veličine populacije, vjerojatnosti mutacije, maksimalne dubine stabla i veličine turnira optimiranih za svaki od 6 dobivenih klastera.

Tablica 6.21: Parametri za GP dobiveni *irace*-om za svaki od klastera temeljen na standardnoj devijaciji mjere temeljene na udaljenosti bez nula u okolini nesrodnih strojeva

parametar	klaster1	klaster2	klaster3	klaster4	klaster5	klaster6
veličina populacije	1165	1163	964	995	1165	610
vjerojatnost mutacije	0.42	0.3	0.11	0.46	0.03	0.86
max dubina stabla	7	5	8	5	7	8
veličina turnira	3	6	5	3	5	7

I u ovom slučaju je vidljivo da su za različite klastere dobivene različite vrijednosti parametara za GP. Vidljivo je da vjerojatnost mutacije varira od vrlo niskih do vrlo visokih vrijednosti. To sugerira da su krajolici dobrote dobivenih klastera uistinu različiti jer u nekima postoji potreba za izlaskom iz lokalnih minimuma, pa je vjerojatnost mutacije visoka, dok u nekima nema potrebe za izlaskom iz lokalnih minimuma jer samo pretraživanje vrlo teško može zapeti u nekom od minimuma, pa je i vjerojatnost mutacije niska.

Tablica 6.22 daje prikaz osnovnih statističkih mjera vrijednosti funkcija dobrote dobivenih u 30 pokretanja GP-a korištenjem ručno određenih parametara i u 30 pokretanja funkcije dobrote korištenjem automatski određenih parametara za GP.

Iz dane tablice je vidljivo da je u 4 od 6 klastera medijan dobivenih vrijednosti funkcije dobrote bolji u slučaju GP-a koji koristi automatski određene parametre nego u slučaju GP-a koji koristi ručno određene parametre. U trećem klasteru su dobivene vrijednosti medijana jednake i iznose 0, što znači da se u tom klasteru nalaze one instance problema koje se mogu riješiti tako da nema zakašnjelih poslova. Dakle, za većinu instanci se ipak postiže određeno poboljšanje prilikom korištenja automatski određenih parametara, a ponekad je prilikom korištenja hiperheuristika i malo poboljšanje dovoljno kako bi se isplatilo korištenje nekog postupka. Slika 6.26 daje prikaz kutijastih dijagrama za sve klastere osim za treći u kojem su dijagrami u obliku samo jedne linije. Za klaster 6 je y-os ograničena na vrijednosti do 0.6, pa stršeća vrijednost koja se dobiva prilikom izvršavanja GP-a s ručno određenim parametrima nije prikazana na ovoj slici.

Druga značajka na temelju koje je rađeno grupiranje u okolini nesrodnih strojeva je stabilnost informacije. Tablica 6.23 daje prikaz veličine populacije, vjerojatnosti mutacije, maksimalne dubine stabla i veličine turnira koje je *irace* naučio za svaki od 4 dobivena klastera.

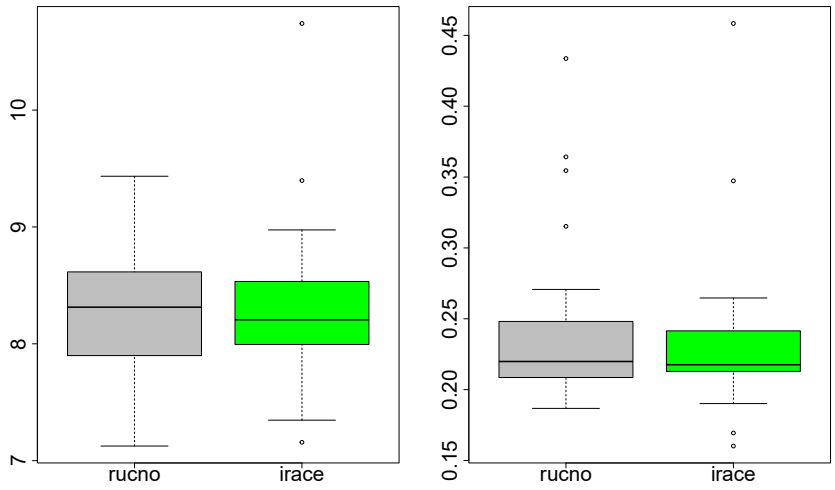
Iz tablice je vidljivo da su za različite klastere naučene različite vrijednosti parametara što ide u prilog pretpostavci da se u dobivenim klasterima instance problema uistinu razlikuju.

Tablica 6.22: Usporedba vrijednosti funkcije dobrote za klastere temeljene na standardnoj devijaciji mjere temeljene na udaljenosti bez nula u okolini nesrodnih strojeva

	klaster 1		klaster 2		klaster 3	
	ručno	irace	ručno	irace	ručno	irace
min	7.1251	7.1572	0.1868	0.1602	0.0000	0.0000
1. kv.	7.9143	8.0130	0.2087	0.2132	0.0000	0.0000
medijan	8.3133	8.2053	0.2199	0.2176	0.0000	0.0000
prosjek	8.2554	8.3198	0.2406	0.2325	0.0000	0.0014
3. kv.	8.5964	8.5061	0.2480	0.2413	0.0000	0.0000
max	9.4337	10.7410	0.4336	0.4584	0.0000	0.0426
stdev	0.5794	0.6438	0.0571	0.0546	0.0000	0.0078
p-vrijednost	0.4383		0.3951		0.8493	
	klaster 4		klaster 5		klaster 6	
	ručno	irace	ručno	irace	ručno	irace
min	4.2349	4.3166	1.1111	1.1214	0.0652	0.0652
1. kv.	4.3655	4.5003	1.1893	1.1991	0.0758	0.0720
medijan	4.5386	4.5756	1.2731	1.2432	0.0995	0.0784
prosjek	4.5258	4.5863	1.2841	1.2731	0.7115	0.1222
3. kv.	4.6359	4.6422	1.3400	1.3147	0.1213	0.1140
max	4.9737	5.1485	1.5318	1.6375	17.7828	0.5752
stdev	0.1800	0.1657	0.1143	0.1172	3.2257	0.1024
p-vrijednost	0.8915		0.3050		0.1084	

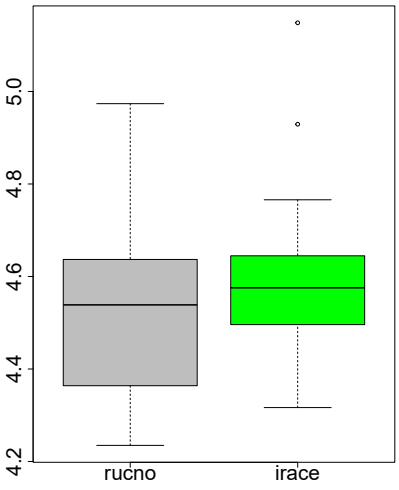
Tablica 6.23: Parametri za GP dobiveni *irace*-om za svaki od klastera temeljen na stabilnosti informacije u okolini nesrodnih strojeva

parametar	klaster1	klaster2	klaster3	klaster4
veličina populacije	1091	1161	1438	971
vjerojatnost mutacije	0.07	0.01	0.71	0.18
max dubina stabla	8	6	3	6
veličina turnira	4	6	6	3

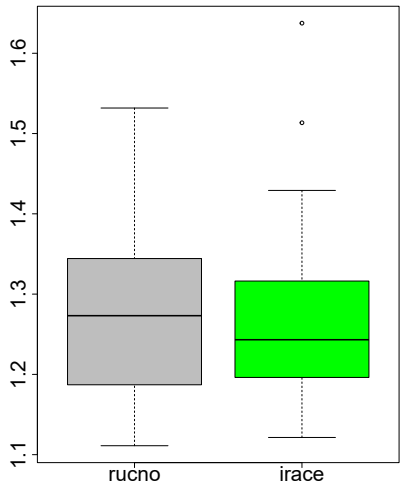


(a) klaster 1

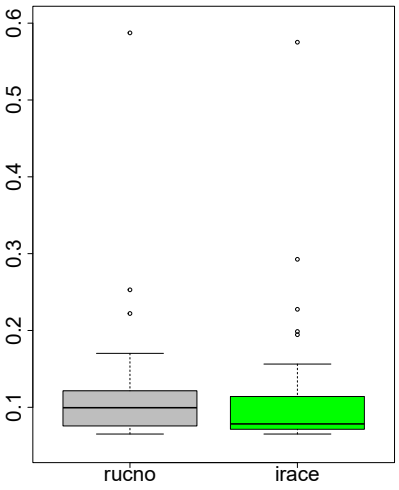
(b) klaster 2



(c) klaster 4



(d) klaster 5



(e) klaster 6

Slika 6.26: Prikaz kutijastih dijagrama dobivenih na temelju standardne devijacije mjere temeljene na udaljenosti bez nula u okolini nesrodnih strojeva

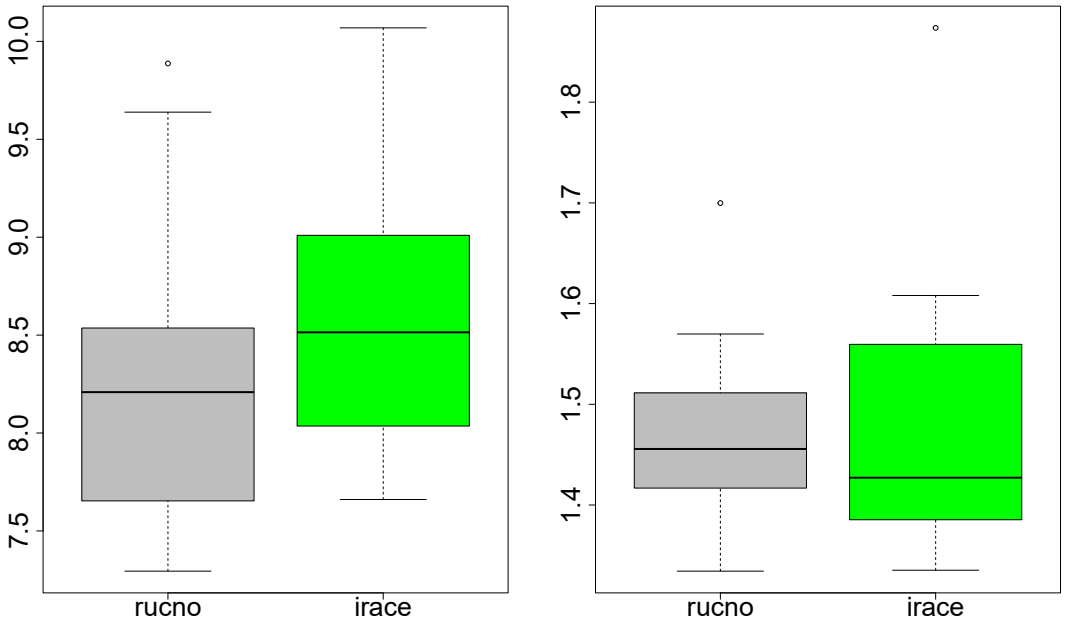
Zanimljivo je primijetiti da je u gotovo svim klasterima vrijednost mutacije niska, odnosno dobiveni krajolik dobrote je gladak. U trećem klasteru, gdje je vjerojatnost mutacije visoka, nalaze se instance problema s najvećom vrijednošću aritmetičke sredine stabilnosti informacije i taj klaster ujedno ima i najmanje elemenata. Za ove elemente je dobiveni krajolik takav da postoji više lokalnih optimuma, pa je potrebna veća vjerojatnost mutacije kako bi se iz njih moglo izaći ukoliko pretraga zapne u nekom od njih.

Tablica 6.24 daje prikaz osnovnih statističkih mjera vrijednosti funkcija dobrote dobivenih u 30 pokretanja GP-a korištenjem ručno određenih parametara i u 30 pokretanja funkcije dobrote korištenjem automatski određenih parametara za GP.

Tablica 6.24: Usporedba vrijednosti funkcije dobrote za klastere temeljene na stabilnosti informacije u okolini nesrodnih strojeva

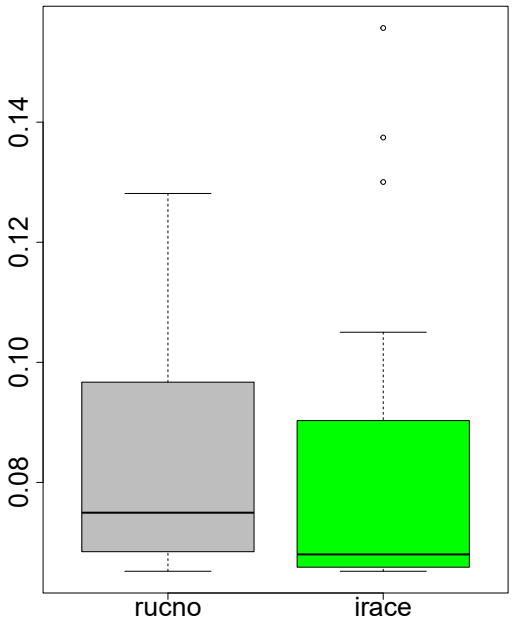
	klaster 1		klaster 2		klaster 3		klaster 4	
	ručno	irace	ručno	irace	ručno	irace	ručno	irace
min	7.2946	7.6606	1.3343	1.3353	0.0652	0.0652	4.2549	4.0825
1. kv.	7.6714	8.0647	1.4182	1.3891	0.0687	0.0659	4.4947	4.4524
medijan	8.2088	8.5143	1.4557	1.4271	0.0750	0.0680	4.6034	4.5805
prosjek	8.2161	8.6592	1.4675	1.4629	0.0842	0.0813	4.8396	4.6030
3. kv.	8.5241	8.9927	1.5112	1.5457	0.0951	0.0903	4.7359	4.7000
max	9.8873	10.0690	1.6998	1.8738	0.1281	0.1557	8.4738	5.5026
stdev	0.6727	0.6937	0.0736	0.1164	0.0202	0.0242	0.8137	0.2655
p-vrijednost	0.9912		0.1935		0.0718		0.2649	

U 3 od 4 klastera je medijan dobivenih vrijednosti dobrote bolji u slučaju GP-a koji koristi automatski određene parametre za pojedini klaster nego u slučaju GP-a koji koristi ručno određene parametre. Raspršenost je podjednaka u oba slučaja za sve klastere osim četvrtog gdje GP s ručno određenim parametrima ima puno veće oscilacije u vrijednostima od GP-a koji koristi parametre određene baš za taj klaster. Slika 6.27 daje prikaz kutijastih dijagrama za sva četiri dobivena klastera. Na slikama se vidi poboljšanje vrijednosti medijana (osim u prvom klasteru), što je dobar pokazatelj za hiperheuristiku jer se postiže neka vrsta poboljšanja u odnosu na prethodno korišteni pristup.

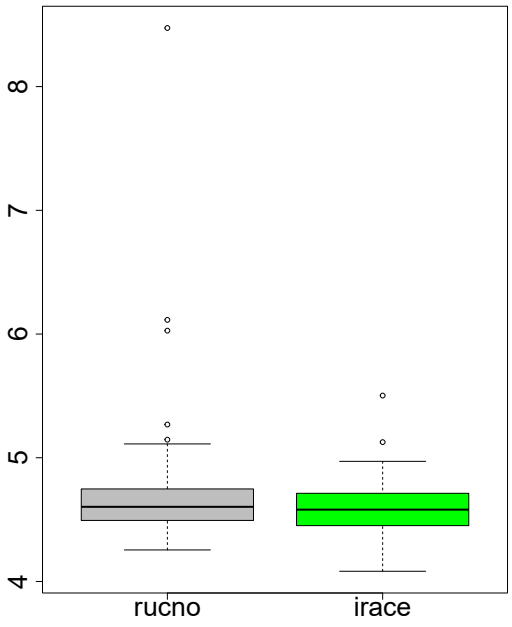


(a) klaster 1

(b) klaster 2



(c) klaster 3



(d) klaster 4

Slika 6.27: Prikaz kutijastih dijagrama dobivenih na temelju stabilnosti informacije u okolini nesrodnih strojeva

6.6 Rezultati za problem raspoređivanja s ograničenim sredstvima, 90 aktivnosti

Ovaj odjeljak prikazuje rezultate dobivene za problem raspoređivanja s ograničenim sredstvima, ali samo na instancama koje zadrže po 90 aktivnosti. Prvo će se opisati rezultati grupiranja, a zatim će se opisati postupak određivanja parametara ručno i korištenjem okruženja *irace* te će se dati usporedba vrijednosti funkcija dobrote dobivene korištenjem tih parametara.

6.6.1 Grupiranje

U problemu raspoređivanja s ograničenim sredstvima u kojem se promatraju samo instance s 90 aktivnosti, grupiranje je rađeno korištenjem algoritma EM. Tablica 6.25 daje vrijednosti logaritamskih vjerodostojnosti dobivenih za dani broj grupa u prikazanim značajkama krajolika dobrote. U tablici su prikazane samo one značajke za koje je postupak grupiranja opisan u odjeljku 6.3 rezultirao s više od jednog klastera. Mjere temeljene na udaljenosti su u tablici nazvane *lengthscale*, dok mjere temeljene na udaljenosti bez nula u tablici imaju oznaku *lengthscale bn*.

Tablica 6.25: Logaritamske vjerodostojnosti dobivene algoritmom EM za značajke u problemu raspoređivanja s ograničenim sredstvima s 90 aktivnosti

FL značajka	# klastera	log vjerodostojnost
lengthscale medijan	4	9.891009425
lengthscale bn donji kvartil	3	9.519606422
lengthscale srednja vrijednost	3	9.209791325
lengthscale standardna devijacija	3	8.853975226
lengthscale bn standardna devijacija	3	8.760618422
lengthscale bn medijan	3	8.732951897
lengthscale bn aritmetička sredina	3	8.688751344
lengthscale gornji kvartil	4	8.424637449
lengthscale bn gornji kvartil	3	8.17710097
stopa neutralnih susjeda	3	7.254059803

Iz tablice se može vidjeti da većina mjera dane instance problema raspoređuje u 3 klastera, dok medijan mjere temeljene na udaljenosti i gornji kvartil mjere temeljene na udaljenosti raspoređuju instance u 4 klastera. Zanimljivo je uočiti da se gotovo sve mjere u prikazanoj tablici odnose na mjere temeljene na udaljenosti. Iz toga se može zaključiti da su u slučaju ovog problema mjere temeljene na udaljenosti one koje najbolje razlučuju različite instance problema.

Za daljnje eksperimente su odabrane tri značajke koje su imale najbolje vrijednosti logaritamske vjerodostojnosti: medijan mjere temeljene na udaljenosti, donji kvartil mjere temeljene na udaljenosti bez nula i srednja vrijednost mjere temeljene na udaljenosti.

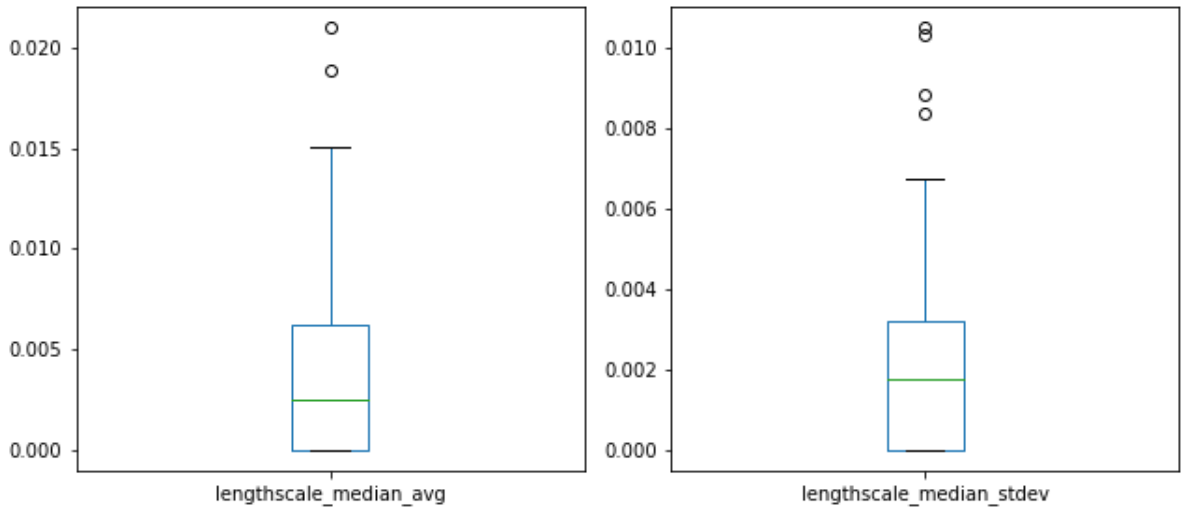
Tablica 6.26 daje prikaz broja instanci u skupovima za učenje, validaciju i testiranje za svaki od klastera dobivenih korištenjem medijana mjere temeljene na udaljenosti. Isto tako, dane su vrijednosti aritmetičke sredine i standardne devijacije promatrane mjere za predstavnika svakog klastera. Zanimljivo je primijetiti da i u ovom problemu, kao i kod okruženja jednog stroja, u jednom klasteru ima jako malo instanci u skupu za učenje, dok se u tom istom klasteru u skupu za testiranje našao puno veći broj instanci. Ovdje se to ne može pripisati različitom načinu generiranja trajanja poslova kao kod okruženja jednog stroja, ali se može pretpostaviti da se ovakva situacija dobiva zbog slučajnosti raspodjele ispitnih primjera u skup za učenje, validaciju i testiranje na samom početku eksperimenta, prije generiranja klastera. S obzirom da instance nisu birane ni po kakvom pravilu za raspoređivanje, vjerojatno se dogodilo da više instanci u skupu za testiranje ima značajke koje pripadaju trećem klasteru nego što je to slučaj kod instanci skupa za učenje i validaciju.

Tablica 6.26: Broj instanci u klasterima i predstavnici klastera za medijan mjere temeljene na udaljenosti u problemu raspoređivanja s ograničenim sredstvima s 90 aktivnosti

	klaster1	**klaster2**	**klaster3**	**klaster4**
# instanci (skup za učenje)	84	37	3	67
# instanci (skup za validaciju)	45	21	1	30
# instanci (skup za testiranje)	93	3	61	35
predstavnik klastera	(0.0002,0.0002)	(0.0097, 0.0044)	(0.0170, 0.0095)	(0.0042, 0.0024)

Slika 6.28 daje prikaz raspona vrijednosti za aritmetičku sredinu i standardnu devijaciju medijana mjere temeljene na udaljenosti. Vidljivo je da je većina vrijednosti smještena bliže minimumu u oba slučaja i da su maksimalne vrijednosti zapravo stršeće vrijednosti.

Slika 6.29 daje prikaz dijagrama raspršenosti za aritmetičku sredinu i standardnu devijaciju medijana mjere temeljene na udaljenosti. Najveći broj instanci se nalazi u prvom klasteru i to su one vrijednosti koje imaju najnižu aritmetičku sredinu i standardnu devijaciju promatrane mjere. Ovakav raspored ne čudi ako pogledamo kutijaste dijagrame sa slike 6.28 gdje je jasno vidljivo da se najviše dobivenih vrijednosti nalazi upravo u predjelu minimuma. Isto tako, vidljivo je zašto algoritam EM za ovu mjeru daje 4 umjesto 3 klastera, iako se na taj način u jednom klasteru dobivaju samo 4 instance (3 u skupu za učenje i 1 u skupu za validaciju). Te 4 vrijednosti su dosta odvojene od ostalih i vjerojatno bi se odabirom 3 klastera i smještanjem tih vrijednosti u neki od postojećih klastera, dobila puno manja vrijednost logaritamske vjerodostojnosti. I u ovom slučaju aritmetička sredina i standardna devijacija pokazuju pozitivnu koreliranost, ali su vrijednosti ipak malo više raspršene nego što je to bio slučaj kod okruženja



Slika 6.28: Kutijasti dijagram vrijednosti medijana mjere temeljene na udaljenosti za problem raspoređivanja s ograničenim sredstvima s 90 aktivnosti

jednog i okruženja nesrodnih strojeva.

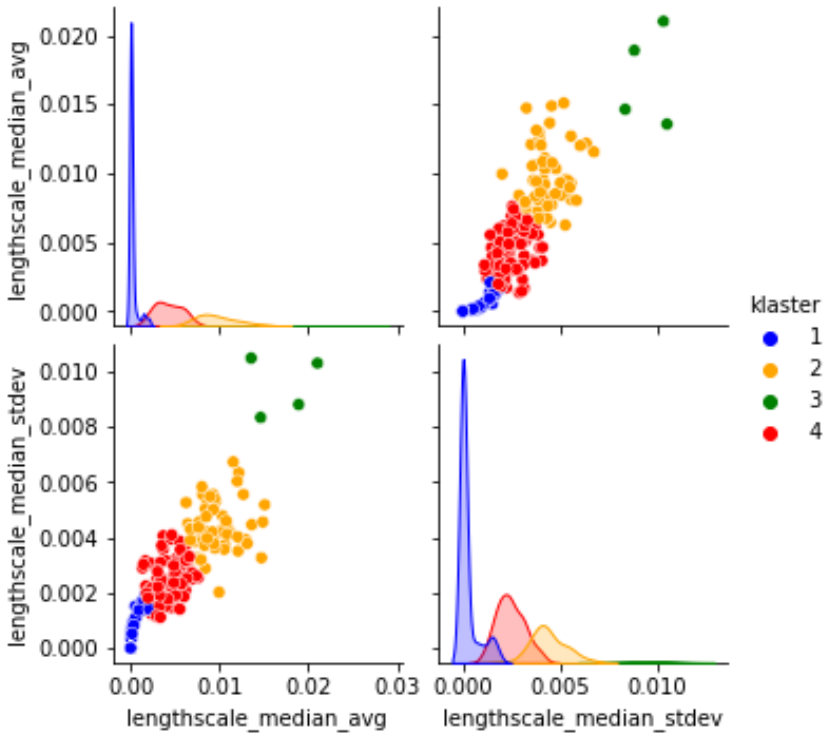
Tablica 6.27 daje prikaz broja instanci u skupovima za učenje, validaciju i testiranje za svaki od klastera dobivenih korištenjem donjeg kvartila mjere temeljene na udaljenosti bez nula. Uz brojeve instanci po skupovima, dane su i vrijednosti aritmetičke sredine i standardne devijacije promatrane mjere za predstavnika svakog klastera. U ovom slučaju su prvi i treći klaster podjednako veliki po pitanju broja instanci u njima, dok je drugi klaster nešto veći. No, i ovdje se dogodila situacija da onaj klaster koji u skupu za učenje ima najmanje instanci, u skupu za testiranje ima najviše, i obratno.

Tablica 6.27: Broj instanci u klasterima i predstavnici klastera za donji kvartil mjere temeljene na udaljenosti bez nula u problemu raspoređivanja s ograničenim sredstvima s 90 aktivnosti

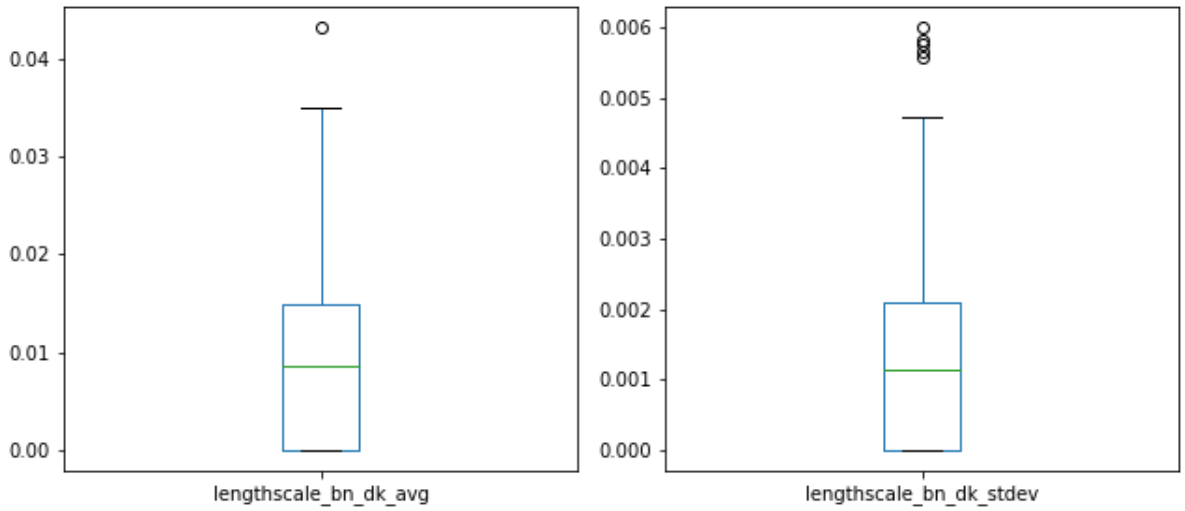
	klaster1	**klaster2**	**klaster3**
# instanci (skup za učenje)	51	95	46
# instanci (skup za validaciju)	24	47	25
# instanci (skup za testiranje)	100	42	50
predstavnik klastera	(0, 0)	(0.0089, 0.0012)	(0.0197, 0.0029)

Slika 6.30 daje prikaz raspona vrijednosti za aritmetičku sredinu i standardnu devijaciju donjeg kvartila mjere temeljene na udaljenosti bez nula. I ovdje je, kao i kod prethodne mjere, većina vrijednosti smještena bliže minimumu u oba slučaja.

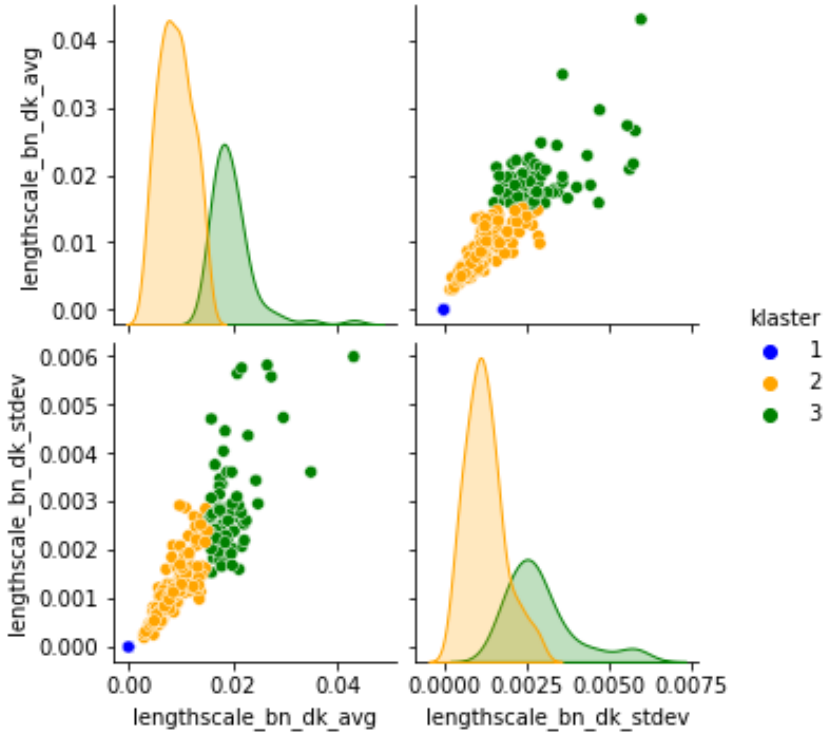
Slika 6.31 daje prikaz dijagrama raspršenosti za aritmetičku sredinu i standardnu devijaciju donjeg kvartila mjere temeljene na udaljenosti bez nula. U ovom slučaju je algoritam EM sve one instance iz kojih je bilo nemoguće izbaciti nule smjestio u prvi klaster, pa se u tom klasteru nalaze samo one instance kojima su i aritmetička sredina i standardna devijacija promatrane



Slika 6.29: Dijagram raspršenja za medijan mjere temeljene na udaljenosti za problem raspoređivanja s ograničenim sredstvima s 90 aktivnosti



Slika 6.30: Kutijasti dijagram vrijednosti donjeg kvartila mjere temeljene na udaljenosti bez nula za problem raspoređivanja s ograničenim sredstvima s 90 aktivnosti



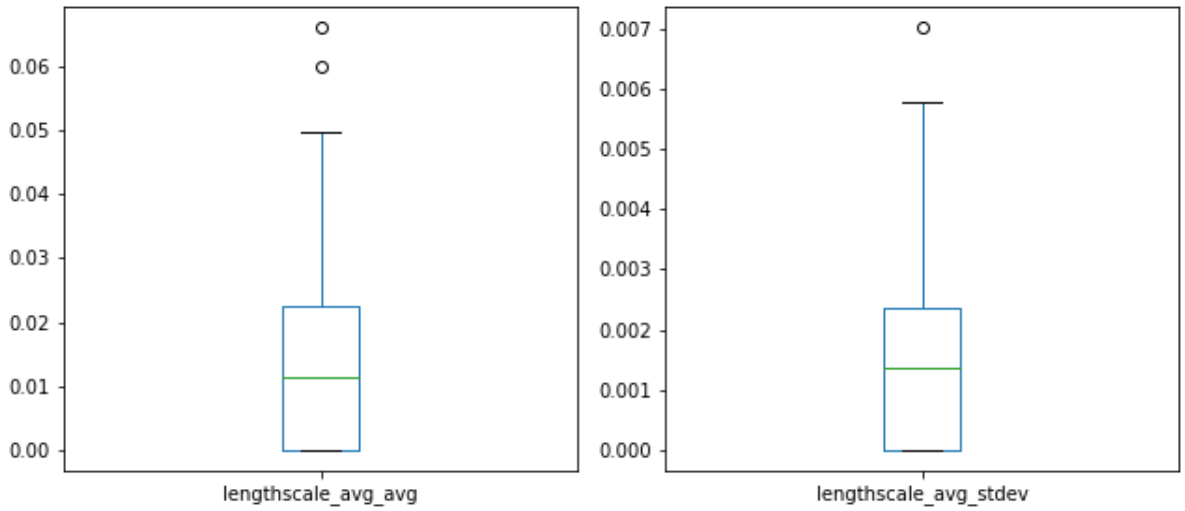
Slika 6.31: Dijagram raspršenja za donji kvartil mjere temeljene na udaljenosti bez nula za problem raspoređivanja s ograničenim sredstvima s 90 aktivnosti

mjere jednake nuli (kao što je to vidljivo i po predstavniku prvog klastera u tablici 6.27). U drugi klaster su se tada smjestile vrijednosti koje se nalaze na sredini raspona aritmetičke sredine i standardne devijacije promatrane mjere, dok su se u trećem klasteru našle instance kojima su vrijednosti aritmetičke sredine i standardne devijacije najveće među promatranim vrijednostima.

Tablica 6.28 daje prikaz broja instanci u skupovima za učenje, validaciju i testiranje te predstavnike za svaki od klastera dobivenih korištenjem prosjeka mjere temeljene na udaljenosti. Za ovu mjeru su instance podjednako raspoređene u sva tri klastera, kako u skupu za učenje, tako i u skupu za testiranje, iako se opet pojavljuje slučaj da klaster s namanjim brojem instanci u skupu za učenje ima najveći broj instanci u skupu za testiranje.

Tablica 6.28: Broj instanci u klasterima i predstavnici klastera za prosjek mjere temeljene na udaljenosti u problemu raspoređivanja s ograničenim sredstvima s 90 aktivnosti

	klaster1	**klaster2**	**klaster3**
# instanci (skup za učenje)	82	53	57
# instanci (skup za validaciju)	43	28	25
# instanci (skup za testiranje)	57	86	49
predstavnik klastera	(0.0119, 0.0014)	(0.0306, 0.0032)	(0.0002, 0.00003)



Slika 6.32: Kutijasti dijagram vrijednosti prosjeka mjere temeljene na udaljenosti za problem raspoređivanja s ograničenim sredstvima s 90 aktivnosti

Slika 6.32 daje prikaz raspona vrijednosti za aritmetičku sredinu i standardnu devijaciju prosjeka mjere temeljene na udaljenosti. Vrijednosti su pretežno grupirane na nižem dijelu raspona, dok se stršeće vrijednosti nalaze oko maksimuma. U ovom slučaju ima manje stršećih vrijednosti nego kod prethodne dvije mjere za RCPSP s 90 aktivnosti.

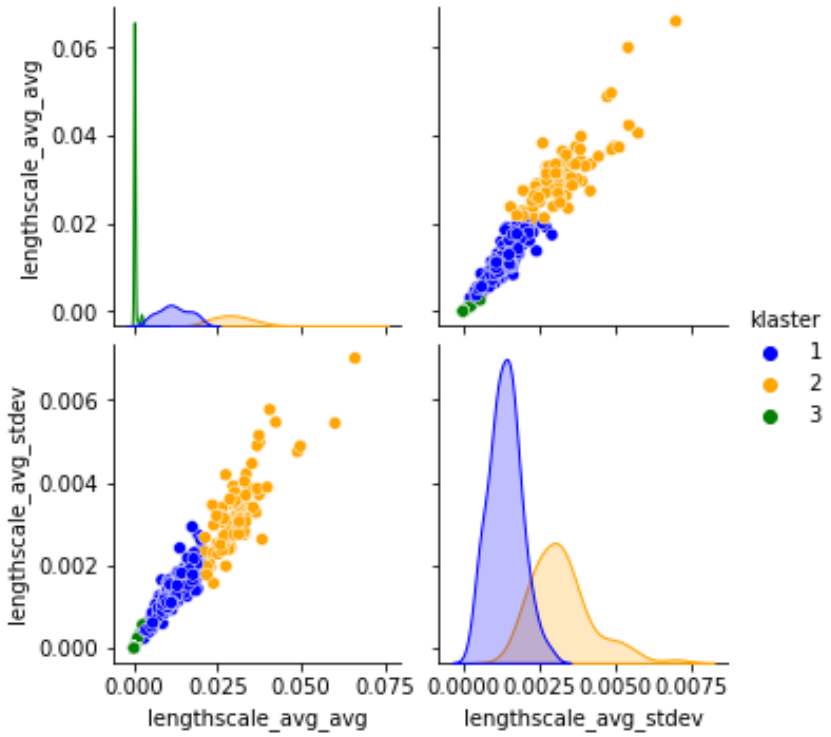
Slika 6.33 daje prikaz dijagrama raspršenosti za aritmetičku sredinu i standardnu devijaciju prosjeka mjere temeljene na udaljenosti. Na ovom dijagramu se može vidjeti da ne postoji jasna granica između prvog i drugog klastera jer su podaci dosta zbijeni. U sljedećem odjeljku će se ispitati utječe li to na kvalitetu dobivenih rješenja.

6.6.2 Određivanje parametara

Prvi korak kod određivanja parametara je odrediti parametre za GP koji daju najbolja rješenja na cijelom skupu za učenje i to ručnim pretraživanjem prostora parametara. Za ovaj problem, parametri za koje treba odrediti vrijednosti su: veličina populacije, vjerojatnost mutacije, maksimalna dubina stabla i veličina turnira u turnirskoj selekciji. Broj generacija je postavljen na 25 kao u [170].

Prilikom određivanja vrijednosti za pojedini parametar, vrijednost tog parametra se bira iz nekog određenog skupa dok su vrijednosti svih ostalih parametara fiksirane. Za svaku vrijednost trenutnog parametra se GP pokreće 30 puta i na temelju dobivenih vrijednosti se odabire vrijednost trenutno promatranog parametra. Početni parametri od kojih kreće pretraživanje su dani u tablici 6.29.

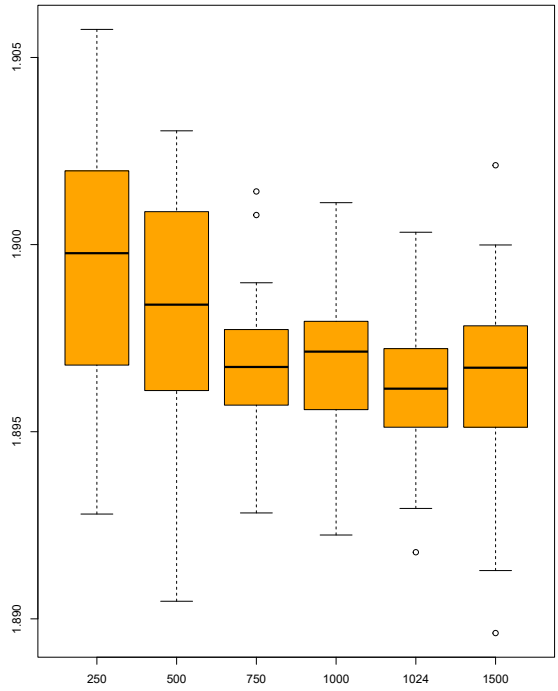
Prvi parametar koji je potrebno odabrati za ovaj problem je veličina populacije, odnosno broj jedinki u populaciji. Broj jedinki se bira iz skupa {250, 500, 750, 1000, 1024, 1500}. Tablica 6.30 prikazuje minimum, maksimum, medijan, prosječnu vrijednost i standardnu devi-



Slika 6.33: Dijagram raspršenja za prosjek mjere temeljene na udaljenosti za problem raspoređivanja s ograničenim sredstvima s 90 aktivnosti

Tablica 6.29: Početni parametri za GP u RCPSP-u sa 90 aktivnosti

naziv parametra	**vrijednost**
broj generacija	25
veličina populacije	1000
vjerojatnost mutacije	0.3
maksimalna dubina stabla	5
veličina turnira	3



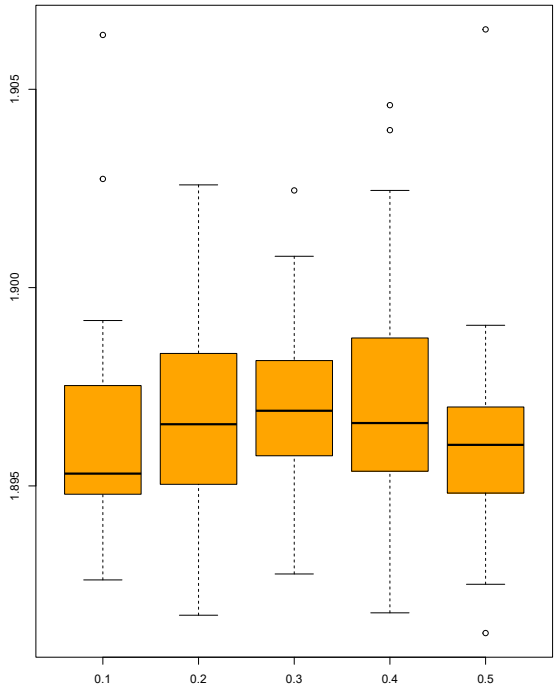
Slika 6.34: Kutijasti dijagrami za veličine populacija iz skupa {250, 500, 750, 1000, 1024, 1500} za RCPSP s 90 aktivnosti

jaciju dobivene pokretanjem GP-a sa odgovarajućim parametrima 30 puta. Slika 6.34 prikazuje kutijaste dijagrame za odgovarajuću vrijednost veličine populacije. Iz tablice je vidljivo da se za veličinu od 1024 jedinke postigne najmanji medijan i gotovo najmanja minimalna vrijednost. Slika 6.34 potvrđuje da je veličina populacije od 1024 jedinke najbolji izbor jer se na njoj vidi da je medijan najmanji i da je raspršenost podataka među najmanjima od svih promatranih vrijednosti.

Tablica 6.30: Vrijednosti funkcije dobrote za veličinu populacije iz skupa {250, 500, 750, 1000, 1024, 1500} za RCPSP s 90 aktivnosti

vel. Populacije	**250**	**500**	**750**	**1000**	**1024**	**1500**
min	1.8928	1.89047	1.89283	1.89224	1.89178	1.88962
max	1.90575	1.90304	1.90142	1.90112	1.90033	1.90212
medijan	1.89977	1.898395	1.89673	1.89714	1.89615	1.89671
prosjek	1.89961	1.898222	1.896753	1.896786	1.896216	1.896231
stdev	0.003443	0.003263	0.001886	0.002069	0.00191	0.00257

Nakon odabira veličine populacije, potrebno je odabrati vjerojatnost mutacije. Za RCPSP s 90 aktivnosti, vjerojatnost mutacije se bira iz skupa {0.1, 0.2, 0.3, 0.4, 0.5}. Tablica 6.31



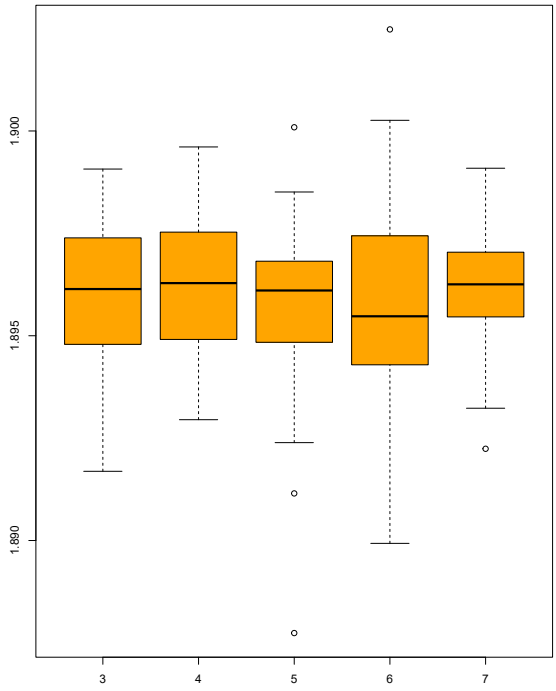
Slika 6.35: Kutijasti dijagrami za vjerojatnost mutacije iz skupa vrijednosti 0.1 - 0.5 za RCPSP s 90 aktivnosti

daje prikaz minimuma, maksimuma, medijana, prosječne vrijednosti i standardne devijacije vrijednosti funkcije dobrote dobivenih u 30 pokretanja GP-a s odgovarajućim parametrima. U tablici se vidi da su dobivene vrijednosti dosta slične za svaki od ponuđenih odabira vjerojatnosti mutacije. Najmanje vrijednosti medijana se postižu za vjerojatnost mutacije 0.1 i 0.5. Najmanja minimalna vrijednost se dobija za 0.5 Ukoliko pogledamo kutijaste dijagrame koji su prikazani na slici 6.35, vidi se da od te dvije vrijednosti, vjerojatnost od 0.5 ima manju raspršenost, pa se iz tog razloga za vjerojatnost mutacije u ovom problemu odabire upravo vrijednost 0.5.

Tablica 6.31: Vrijednosti funkcije dobrote za vjerojatnost mutacije iz skupa vrijednosti 0.1 - 0.5 za RCPSP s 90 aktivnosti

vjer. mutacije	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
min	1.893	1.892	1.893	1.892	1.891
max	1.906	1.903	1.902	1.905	1.907
medijan	1.895	1.897	1.897	1.897	1.896
prosjek	1.896	1.897	1.897	1.897	1.896
stdev	0.003	0.003	0.002	0.003	0.003

Sljedeći parametar za koji se određuje vrijednost je maksimalna dubina stabla, i ovdje se



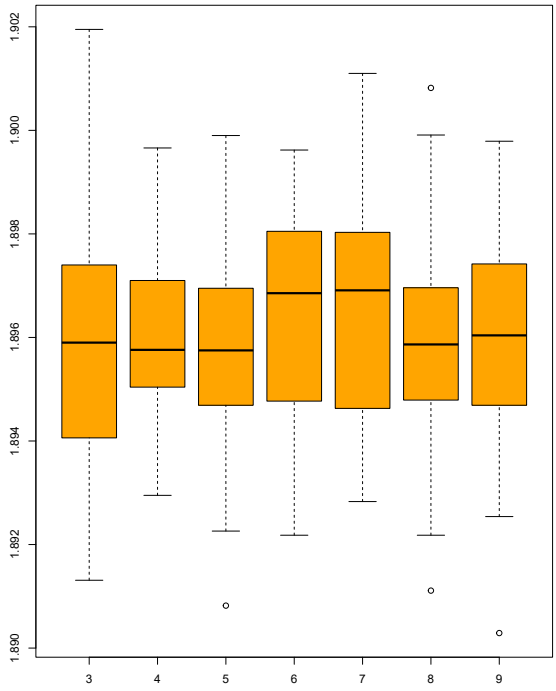
Slika 6.36: Kutijasti dijagrami za maksimalnu dubinu stabla iz skupa vrijednosti 3 - 7 za RCPSP s 90 aktivnosti

ona bira iz skupa vrijednosti {3, 4, 5, 6, 7}. Tablica 6.32 daje minimum, maksimum, medijan, prosječnu vrijednost i standardnu devijaciju vrijednosti funkcije dobrote dobivene pokretanjem GP-a sa odgovarajućim vrijednostima ovog parametra. U ovom slučaju su medijani gotovo jednaki za sve odabire dubine stabla, no najbolja minimalna vrijednost se postiže za maksimalnu dubinu stabla jednaku 5. Ukoliko promotrimo odgovarajuće kutijaste dijagrame sa slike 6.36, vidimo da maksimalna dubina stabla 5 ima jednu od najmanjih raspršenosti među svim promatranim vrijednostima, pa se ta dubina stabla odabire za daljnje eksperimente.

Tablica 6.32: Vrijednosti funkcije dobrote za maksimalnu dubinu stabla iz skupa vrijednosti 3 - 7 za RCPSP s 90 aktivnosti

max dubina stabla	**3**	**4**	**5**	**6**	**7**
min	1.892	1.893	1.888	1.89	1.892
max	1.899	1.9	1.9	1.902	1.899
medijan	1.896	1.896	1.896	1.895	1.896
prosjek	1.896	1.896	1.896	1.896	1.896
stdev	0.002	0.002	0.002	0.003	0.002

Na poslijetku se odabire vrijednost za veličinu turnira u turnirskoj selekciji. U ovom slučaju



Slika 6.37: Kutijasti dijagrami za veličinu turnira iz skupa vrijednosti 3 - 9 za RCPSP s 90 aktivnosti

se ta vrijednost bira iz skupa {3, 4, 5, 6, 7, 8, 9}. Ovdje se u tablici 6.33 koja daje prikaz minimuma, maksimuma, medijana, srednje vrijednosti i standardne devijacije funkcije dobrote dobivene pokretanjem GP-a s odgovarajućom vrijednošću parametra, vidi da su sve dobivene vrijednosti dosta slične, pa se trebaju pogledati kutijasti dijagrami sa slike 6.37 kako bi se moglo odlučiti koju vrijednost odabrati za ovaj parametar. Iz kutijastih dijagrama se može vidjeti da se za veličine turnira 5 i 9 postižu najmanje minimalne vrijednosti, ali je za veličinu turnira 5 medijan nešto niži nego za veličinu turnira 9 i raspršenost podataka je manja za veličinu turnira 5. Iz tih razloga je upravo 5 veličina turnira koja se odabire za daljnje eksperimente.

Tablica 6.33: Vrijednosti funkcije dobrote za veličinu turnira iz skupa vrijednosti 3 - 9 za RCPSP s 90 aktivnosti

veličina turnira	**3**	**4**	**5**	**6**	**7**	**8**	**9**
min	1.891	1.893	1.891	1.892	1.893	1.891	1.89
max	1.902	1.9	1.9	1.9	1.901	1.901	1.9
medijan	1.896	1.896	1.896	1.897	1.897	1.896	1.896
prosjek	1.896	1.896	1.896	1.896	1.897	1.896	1.896
stdev	0.002	0.002	0.002	0.002	0.002	0.002	0.002

Popis vrijednosti svih parametara koji su odabrani prethodno opisanim postupkom je vidljiv

```
# name          switch              type      values
tsize           "--tsize "            i       (3,7)
maxdepth        "--maxdepth "         i       (3,10)
popsize         "--popsize "          i       (200,2000)
mutProb         "--mutprob "          r       (0.01, 0.99)
```

Slika 6.38: Parametri za *irace* za RCPSP s 90 aktivnosti

u tablici 6.34.

Tablica 6.34: Parametri za GP u RCPSP-u s 90 aktivnosti, dobiveni ručnim pretraživanjem prostora parametara

naziv parametra	**vrijednost**
broj generacija	25
veličina populacije	1024
vjerojatnost mutacije	0.5
maksimalna dubina stabla	5
veličina turnira	5

Kada se parametri za GP određe ručno, potrebno je konfigurirati *irace* tako da se pomoću njega za svaki od dobivenih klastera mogu odrediti parametri za GP u tom klasteru. U problemu raspoređivanja s ograničenim sredstvima s 90 aktivnosti maksimalan broj iteracija u *irace*-u je za svaki klaster određen kao $\lceil \frac{200}{broj\_klastera} \rceil$. Korišten je elitizam u *irace*-u i to tako da je samo najbolja konfiguracija zadržana za sljedeću iteraciju.

Kao i kod prethodne dvije okoline, potrebno je definirati datoteku u kojoj će biti nabrojani svi parametri koji se trebaju odrediti pomoću *irace*-a, kao i njihovi tipovi te vrijednosti koje oni mogu poprimiti. Slika 6.38 daje prikaz tih informacija za RCPSP s 90 aktivnosti.

Prvi parametar, *tsize*, označava veličinu turnira i poprima cjelobrojne vrijednosti iz raspona od 3 do 7. Parametar *maxdepth* određuje maksimalnu dubinu stabla. Kao i prethodni parametar, i ovaj parametar je cjelobrojan te on poprima vrijednosti iz raspona od 3 do 10. Veličina populacije je označena parametrom *popsize* koji je, kao i prethodna dva parametra, cjelobrojan i poprima vrijednosti iz raspona od 200 do 2000. Posljednji parametar koji je potrebno odrediti je vjerojatnost mutacije koji je ovdje označen s *mutProb*. Ovaj parametar poprima realne vrijednosti iz intervala $[0.01, 0.99]$ i to uz točnost od dvije decimale. Za problem raspoređivanja s ograničenim sredstvima nisu definirane zabranjene konfiguracije parametara.

U nastavku će biti prikazani rezultati usporedbe vrijednosti funkcije dobrote dobivenih u 30 pokretanja GP-a s ručno određenim parametrima i onih vrijednosti dobivenih u 30 pokretanja GP-a s parametrima određenim pomoću *irace*-a na odgovarajućim testnim skupovima za klastere dobivene na temelju značajki krajolika dobrote spomenutih u prethodnom odjeljku.

Prva značajka krajolika dobrote na temelju koje je rađeno grupiranje je medijan mjere temeljene na udaljenosti. Tablica 6.35 daje vrijednosti veličine populacije, vjerojatnosti mutacije, maksimalne dubine stabla i veličine turnira dobivene *irace*-om na skupu za učenje u odgovarajućem klasteru.

Tablica 6.35: Parametri za GP dobiveni *irace*-om za svaki od klastera temeljen na medijanu mjere temeljene na udaljenosti za RCPSP s 90 aktivnosti

parametar	klaster1	klaster2	klaster3	klaster4
veličina populacije	733	1187	1320	1480
vjerojatnost mutacije	0.65	0.36	0.66	0.83
max dubina stabla	10	7	5	3
veličina turnira	3	3	7	5

U ovom slučaju je također vidljivo da za različite klastere *irace* pronalazi različite parametre za GP. To znači da značajka krajolika dobrote koja se ovdje promatra uspijeva razlikovati instance problema i grupirati ih u slične klastere. Isto tako, zanimljivo je primijetiti da, što je veća veličina populacije, to je manja maksimalna dubina stabla koja se dobija za određeni klaster.

Tablica 6.36 daje prikaz osnovnih statističkih mjera za vrijednosti funkcije dobrote dobivene uporabom GP-a s ručno određenim parametrima i GP-a s automatski određenim parametrima u odgovarajućem klasteru.

Tablica 6.36: Usporedba vrijednosti funkcije dobrote za klastere temeljene na medijanu mjere temeljene na udaljenosti za RCPSP s 90 aktivnosti

	klaster 1		klaster 2		klaster 3		klaster 4	
	ručno	irace	ručno	irace	ručno	irace	ručno	irace
min	1.6945	1.6941	2.5687	2.5623	1.8472	1.8431	2.5355	2.5325
1. kv.	1.6953	1.6959	2.5949	2.6157	1.8500	1.8479	2.5538	2.5465
medijan	1.6965	1.6969	2.6581	2.6481	1.8531	1.8501	2.5591	2.5498
prosjek	1.6969	1.6976	2.6507	2.6512	1.8559	1.8527	2.5580	2.5527
3. kv.	1.6978	1.6986	2.6873	2.6861	1.8604	1.8560	2.5628	2.5590
max	1.7007	1.7073	2.7554	2.7812	1.8840	1.8768	2.5702	2.5817
stdev	0.0018	0.0026	0.0546	0.0525	0.0082	0.0079	0.0080	0.0103
p-vrijednost	0.8263		0.4646		0.0312		0.0056	

Može se vidjeti da je za gotovo svaki klaster medijan manji u slučaju korištenja GP-a s automatski određenim parametrima. U klasteru 1 je medijan nešto manji za GP s ručno određenim parametrima, ali je minimum koji se postiže automatski određenim parametrima manji

od minimuma koji se postiže s ručno određenim parametrima. Zanimljivo je primijetiti i da je u svakom klasteru minimalna vrijednost postignuta s automatski određenim parametrima manja od minimalne vrijednosti postignute s ručno određenim parametrima. Slika 6.39 daje usporedbu kutijastih dijagrama za vrijednosti funkcije dobrote u slučaju korištenja ručno određenih parametara i korištenja automatski određenih parametara za svaki od klastera.

Tablica 6.37 daje prikaz parametara za GP dobivenih automatskim odabirom korištenjem *irace*-a za tri klastera dobivena na temelju donjeg kvartila mjere temeljene na udaljenosti bez nula.

Tablica 6.37: Parametri za GP dobiveni *irace*-om za svaki od klastera temeljen na donjem kvartilu mjere temeljene na udaljenosti bez nula za RCPSP s 90 aktivnosti

parametar	klaster1	klaster2	klaster3
veličina populacije	884	1061	1911
vjerojatnost mutacije	0.35	0.28	0.4
max dubina stabla	8	6	7
veličina turnira	7	3	5

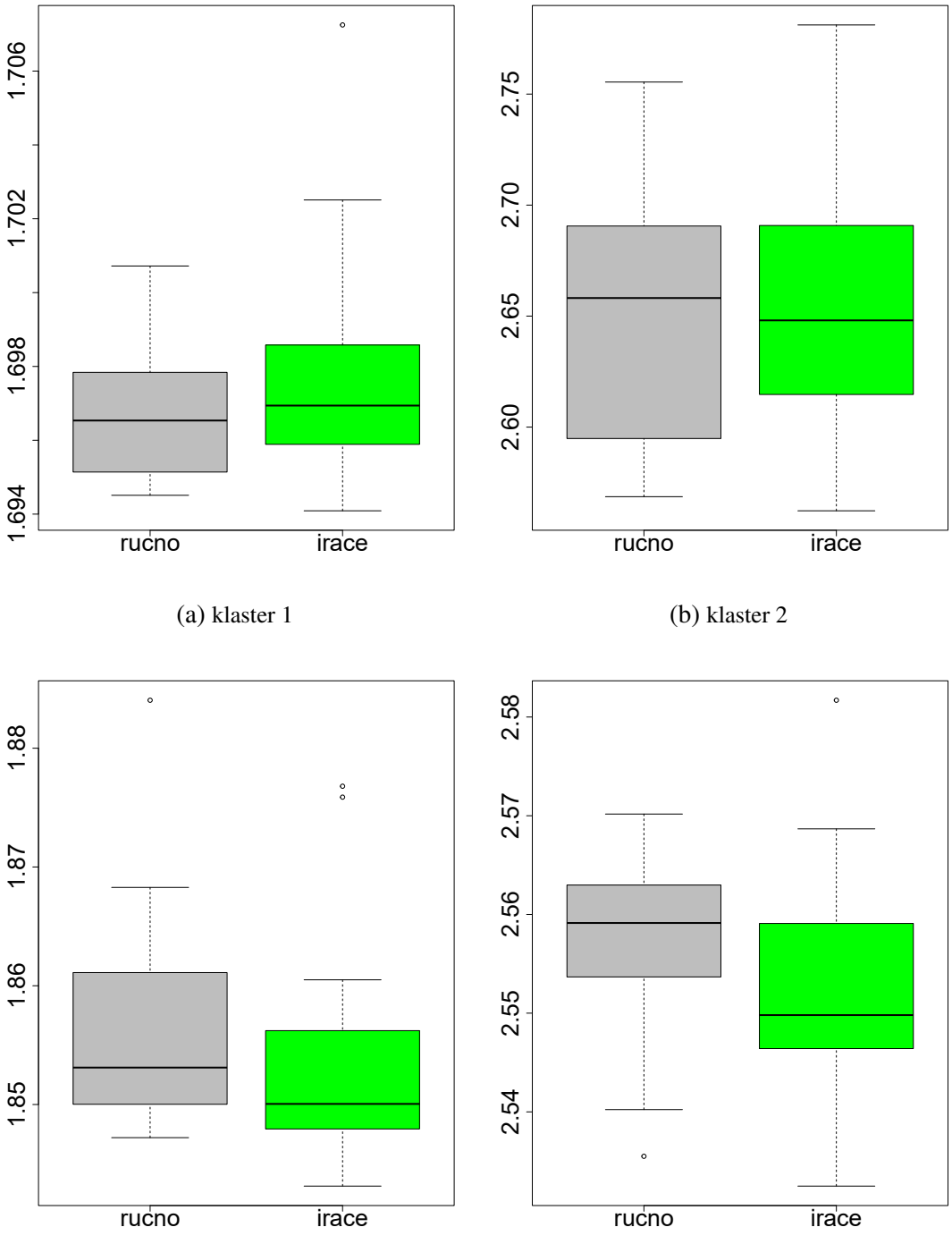
Kao i u prethodnim slučajevima, dobiveni parametri se razlikuju za dobivene klastere. Može se uočiti da su u klasterima dobivenim ovom značajkom vjerojatnosti mutacije manje nego kod klastera dobivenih na temelju medijana mjere temeljene na udaljenosti, što znači da, iako se grupiraju jednake instance problema u oba slučaja, svaka od ove dvije značajke promatra te instance iz svoje perspektive, na drugačiji ih način grupira i time se stvara različit izgled krajolika dobrote za pojedinu grupu instanci.

Tablica 6.38 daje minimum, medijan, prvi i treći kvartil, prosjek, standardnu devijaciju i maksimum vrijednosti funkcije dobrote dobivenih korištenjem GP-a s ručno određenim parametrima i GP-a s automatski određenim parametrima.

Iz tablice je vidljivo da je medijan vrijednosti dobivenih pokretanjem s parametrima dobivenim *irace*-om manji ili jednak medijanu vrijednosti dobivenom s parametrima određenima ručno. U ovom slučaju je standardna devijacija podjednaka u oba slučaja za sve klastere, što znači da za ovakvo grupiranje vrijednosti funkcije dobrote za instance unutar klastera ne osciliraju previše. U klasteru 3 su se našle one instance koje se mogu riješiti optimalno, pa i GP s ručno određenim parametrima i GP s parametrima određenim pomoću *irace*-a daju jednake rezultate u tom slučaju. Slika 6.40 daje kutijaste dijagrame za prvi i drugi klaster dobiven korištenjem donjeg kvartila mjere temeljene na udaljenosti bez nula.

Posljednja značajka na temelju koje je rađeno grupiranje u ovom problemu je prosjek mjere temeljene na udaljenosti i u nastavku će biti prikazani rezultati za dobiveno grupiranje.

Tablica 6.39 daje vrijednosti parametara za GP određenih korištenjem *irace*-a za odgovara-



(a) klaster 1

(b) klaster 2

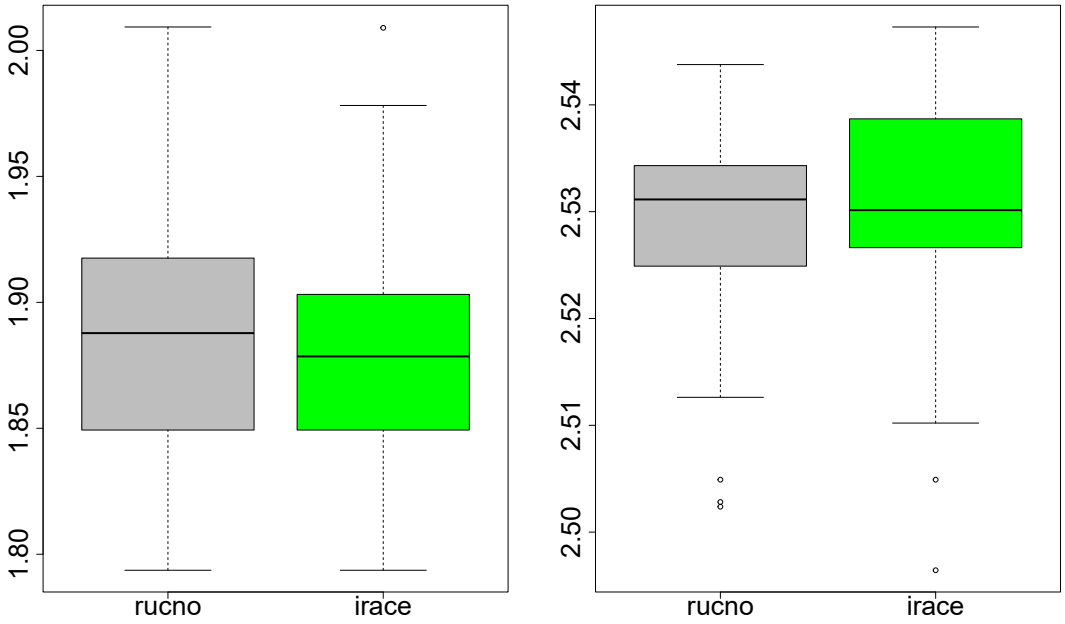
(c) klaster 3

(d) klaster 4

Slika 6.39: Prikaz kutijastih dijagrama dobivenih na temelju medijana mjere temeljene na udaljenosti za RCPSP s 90 aktivnosti

Tablica 6.38: Usporedba vrijednosti funkcije dobrote za klastere temeljene na donjem kvartilu mjere temeljene na udaljenosti bez nula za RCPSP s 90 aktivnosti

	klaster 1		klaster 2		klaster 3	
	ručno	irace	ručno	irace	ručno	irace
min	1.7936	1.7936	2.5024	2.4964	1.6555	1.6555
1. kv.	1.8493	1.8493	2.5249	2.5267	1.6555	1.6555
medijan	1.8878	1.8785	2.5311	2.5301	1.6555	1.6555
prosjek	1.8945	1.8813	2.5282	2.5292	1.6555	1.6555
3. kv.	1.9172	1.9031	2.5343	2.5379	1.6555	1.6555
max	2.0093	2.0090	2.5438	2.5473	1.6555	1.6555
stdev	0.0508	0.0485	0.0112	0.0116	0.0000	0.0000
p-vrijednost	0.2041		0.5617		1	



(a) klaster 1

(b) klaster 2

Slika 6.40: Prikaz kutijastih dijagrama dobivenih na temelju donjeg kvartila mjere temeljene na udaljenosti bez nula za RCPSP s 90 aktivnosti

juće klastere dobivene prosjekom mjere temeljene na udaljenosti.

Tablica 6.39: Parametri za GP dobiveni *irace*-om za svaki od klastera temeljen na prosjeku mjere temeljene na udaljenosti za RCPSP s 90 aktivnosti

parametar	klaster1	klaster2	klaster3
veličina populacije	1690	1844	278
vjerojatnost mutacije	0.89	0.79	0.18
max dubina stabla	3	5	3
veličina turnira	5	3	4

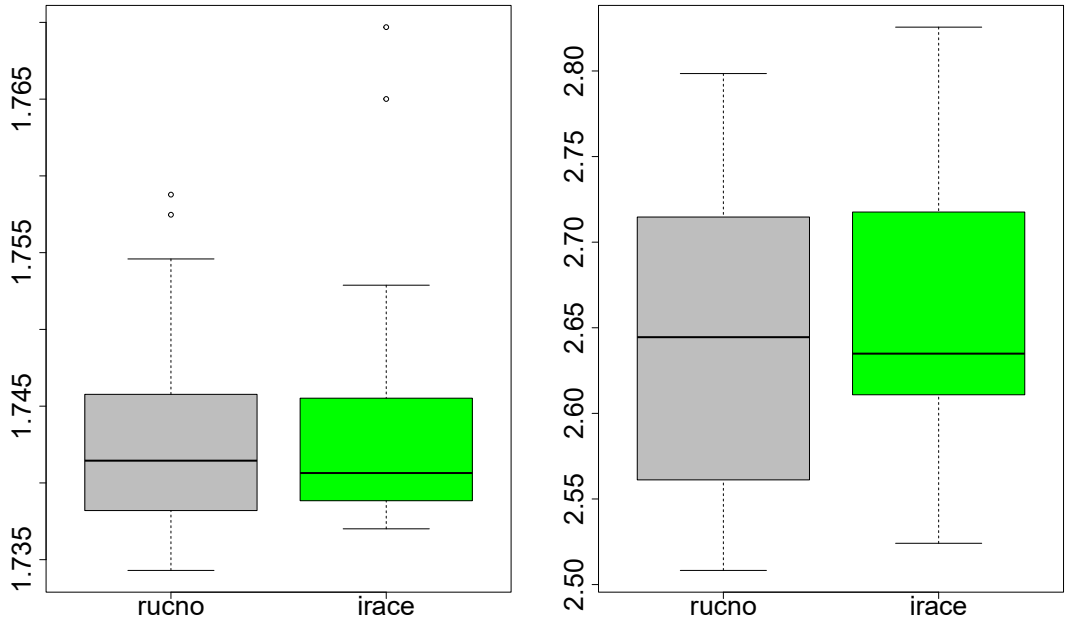
Vidljivo je da je za sva tri klastera maksimalna dubina stabla relativno mala i da je u klasteru 3 određen vrlo mali broj jedinki u populaciji. To vjerojatno pokazuje da su u tom klasteru instance problema jako slične, pa nema potrebe koristiti previše jedinki kako bi se pretražio prostor stanja s obzirom da se s vrlo malim brojem jedinki može doći do odgovarajućih rješenja.

Tablica 6.40 daje osnovne statističke mjere za vrijednosti funkcije dobrote dobivene pokretanjem GP-a s odgovarajućim parametrima po 30 puta.

Tablica 6.40: Usporedba vrijednosti funkcije dobrote za klastere temeljene na prosjeku mjere temeljene na udaljenosti za RCPSP s 90 aktivnosti

	klaster 1		klaster 2		klaster 3	
	ručno	irace	ručno	irace	ručno	irace
min	1.6566	1.6566	1.7343	1.7370	2.5083	2.5241
1. kv.	1.6570	1.6570	1.7383	1.7389	2.5637	2.6109
medijan	1.6570	1.6570	1.7415	1.7406	2.6445	2.6349
prosjek	1.6570	1.6570	1.7428	1.7438	2.6445	2.6585
3. kv.	1.6570	1.6570	1.7457	1.7455	2.7118	2.7153
max	1.6574	1.6577	1.7588	1.7697	2.7985	2.8256
stdev	0.0002	0.0002	0.0064	0.0077	0.0874	0.0797
p-vrijednost	0.4763		0.6049		0.6848	

U sva tri klastera je vrijednost medijana dobivena pokretanjem GP-a sa automatski određenim parametrima manja od vrijednosti medijana dobivene pokretanjem GP-a sa ručno određenim parametrima. Zanimljivo je primijetiti da se u klasteru 3 dobije bolja vrijednost medijana u slučaju automatski određenih parametara, a veličina populacije je znatno manja nego kod ručno određenih parametara. To znači da se ovim postupkom u nekim situacijama može znatno ubrzati rješavanje problema s obzirom da su dobivena rješenja dobra, u nekim slučajevima i bolja



(a) klaster 2 (b) klaster 3

Slika 6.41: Prikaz kutijastih dijagrama dobivenih na temelju prosjeka mjere temeljene na udaljenosti za RCPSP s 90 aktivnosti

kada se koristi grupiranje, a poboljšanje je ostvareno korištenjem manjeg broja jedinki u populaciji što automatski povlači i kraće vrijeme izvršavanja GP-a. Odgovarajući kutijasti dijagrami za klaster 2 i klaster 3 su vidljivi na slici 6.41.

6.7 Rezultati za problem raspoređivanja s ograničenim sredstvima, sve aktivnosti

Ovaj odjeljak prikazuje rezultate dobivene za problem raspoređivanja s ograničenim sredstvima za instance s 30, 60, 90 i 120 aktivnosti. Kao i za sve prethodno promatrane probleme, prvo će se opisati rezultati grupiranja, a zatim će se opisati postupak određivanja parametara ručno i korištenjem okruženja *irace*.

6.7.1 Grupiranje

Za RCPSP u kojemu su u obzir uzete instance s brojevima aktivnosti iz skupa 30, 60, 90, 120, grupiranje je rađeno korištenjem algoritma EM. Tablica 6.41 daje vrijednosti logaritamskih vjerodostojnosti dobivene za dani broj grupa u prikazanim značajkama krajolika dobrote. Pored informacijskih mjera u zagradi navedeno je za koji ε je dobivena dana vrijednost, gdje je ε parametar koji je opisan u odjeljku 6.3. Kao i kod problema raspoređivanja s ograničenim

sredstvima u kojima su promatrane samo instance sa po 90 aktivnosti, u ovoj tablici je mjera temeljena na udaljenosti označena sa *lengthscale*, a mjera temeljena na udaljenosti bez nula sa *lengthscale bn*.

Iako u ovoj tablici djelomični sadržaj informacije sa $\varepsilon = 0.5$, minimum mjere temeljene na udaljenosti bez nula i sadržaj informacije sa $\varepsilon = 0.5$ imaju najbolje vrijednosti logaritamske vjerodostojnosti, ove mjere nisu uzete u obzir za daljnje eksperimente jer su klasteri dobiveni grupiranjem na temelju tih mjera jako nesrazmjerni u pogledu broja instanci koje sadrže. Za djelomični sadržaj informacije sa $\varepsilon = 0.5$ dobivaju se 3 klastera u kojima se nalazi 46, 1156 i 22 instance; za minimum mjere temeljene na udaljenosti bez nula se dobiju 2 klastera u kojima se nalazi 12 i 1212 instanci, dok se za sadržaj informacije sa $\varepsilon = 0.5$ dobiju tri klastera u kojima se nalazi 1114, 25 i 85 instanci. Vidljivo je da ovakva grupiranja baš nemaju smisla jer gotovo sve instance svrstavaju se u samo jedan klaster, pa se iz tog razloga ta tri dobivena grupiranja ne koriste. Eksperimenti su rađeni za medijan mjere temeljene na udaljenosti, djelomični sadržaj informacije sa $\varepsilon = 0.2$ i donji kvartil mjere temeljene na udaljenosti bez nula te će za njih u nastavku biti prikazane odgovarajuće tablice i grafovi.

Tablica 6.42 daje prikaz broja instanci u skupovima za učenje, validaciju i testiranje za svaki od klastera dobivenih korištenjem medijana mjere temeljene na udaljenosti, kao i vrijednosti aritmetičke sredine i standardne devijacije promatrane mjere za predstavnika svakog klastera. Prvi i drugi klaster su nešto veći od četvrtog, dok je u treći klaster raspoređeno tek nekoliko instanci problema. No, u ovom slučaju i instance u testnom skupu prate ovakvu raspodjelu, pa se u dva klastera najveća po broju instanci u skupu za učenje, nalazi najveći broj instanci iz skupa za testiranje, a isto tako se u najmanjem klasteru ako se gleda skup za učenje, nalazi najmanji broj instanci iz skupa za testiranje. Može se zaključiti da u ovom slučaju promatrana značajka na jednak način grupira instance i iz skupa za učenje i iz skupa za testiranje.

Slika 6.42 daje prikaz raspona vrijednosti za aritmetičku sredinu i standardnu devijaciju medijana mjere temeljene na udaljenosti. Kao i u svim ostalim primjerima, i ovdje je vidljivo da je većina vrijednosti smještena blizu minimumu, dok su veće vrijednosti zapravo stršeće vrijednosti.

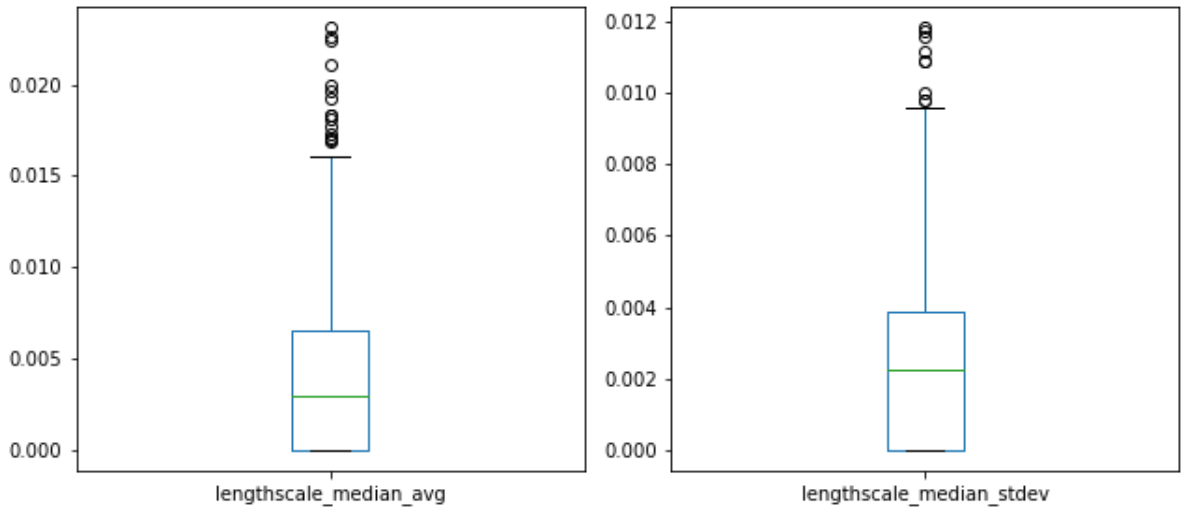
Slika 6.43 daje prikaz dijagrama raspršenosti za aritmetičku sredinu i standardnu devijaciju medijana mjere temeljene na udaljenosti. U drugom klasteru su one instance koje imaju najmanje vrijednosti aritmetičke sredine i standardne devijacije promatrane mjere, dok su u trećem klasteru one koje imaju najveće vrijednosti aritmetičke sredine i standardne devijacije. S obzirom na izgled kutijastih dijagrama sa slike 6.42, ne čudi da treći klaster sadrži najmanje elemenata od svih dobivenih klastera. U prvi i četvrti klaster smjestile su se instance koje su u sredini po vrijednosti aritmetičke sredine i standardne devijacije. Iz prikaza na sporednoj dijagonali može se vidjeti pozitivna koreliranost aritmetičke sredine i standardne devijacije promatrane mjere, ali ipak postoji i određen stupanj raspršenosti u promatranim instancama.

Tablica 6.41: Logaritamske vjerodostojnosti dobivene algoritmom EM za značajke u problemu raspoređivanja s ograničenim sredstvima s 30, 60, 90 i 120 aktivnosti

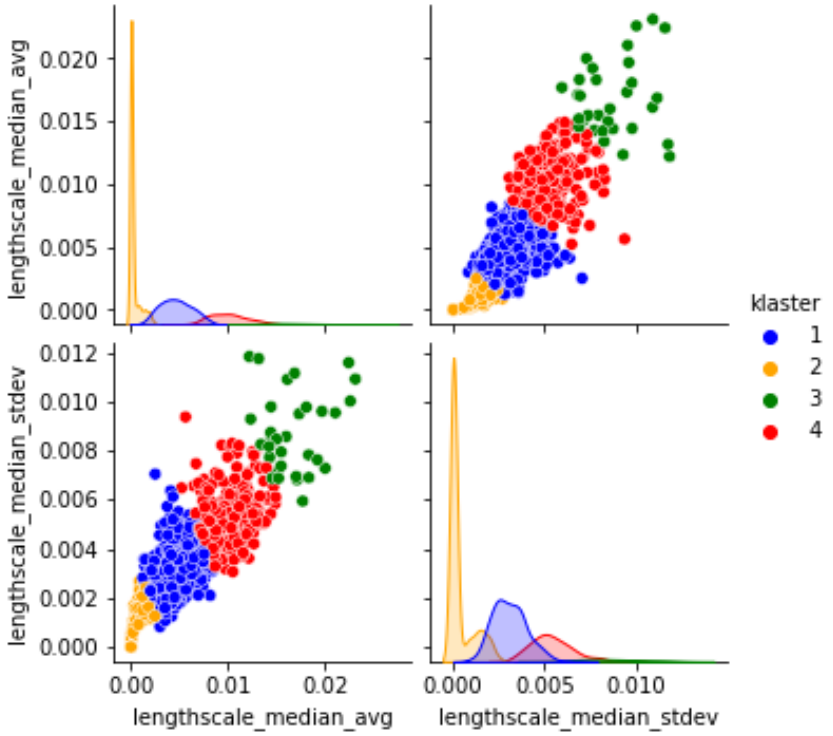
FL značajka	# klastera	log vjerodostojnost
djelomični sadržaj informacije ($\varepsilon = 0.5$)	3	11.52539163
lengthscale bn minimum	2	11.50061514
sadržaj informacije ($\varepsilon = 0.5$)	3	11.26166123
lengthscale medijan	4	9.68747132
djelomični sadržaj informacije ($\varepsilon = 0.2$)	3	9.111323928
lengthscale bn donji kvartil	4	9.060979829
lengthscale srednja vrijednost	4	8.901746061
lengthscale standardna devijacija	4	8.482530922
sadržaj informacije ($\varepsilon = 0.2$)	3	8.369587528
lengthscale bn standardna devijacija	4	8.360491481
lengthscale bn srednja vrijednost	4	8.254307332
djelomični sadržaj informacije ($\varepsilon = 0.02$)	3	8.204378339
lengthscale bn medijan	5	8.16621868
djelomični sadržaj informacije ($\varepsilon = 0.1$)	3	8.13865423
lengthscale gornji kvartil	4	7.7968803
sadržaj informacije ($\varepsilon = 0.02$)	3	7.667948551
lengthscale bn gornji kvartil	4	7.536553753
sadržaj informacije ($\varepsilon = 0.1$)	3	7.492981755
stopa neutralnih susjeda	4	7.063226633
stabilnost informacije	3	6.576897322
koeficijent autokorelacije	4	6.16937600
raspon slučajnog probiranja	6	5.309351184
minimum slučajnog probiranja	5	4.566355327
maksimum slučajnog probiranja	6	4.253746382
lengthscale bn maksimum	5	3.957309002
lengthscale maksimum	5	3.95712105
prosječna dobrota	4	3.637821527

Tablica 6.42: Broj instanci u klasterima i predstavnici klastera za medijan mjere temeljene na udaljenosti u problemu raspoređivanja s ograničenim sredstvima s 30, 60, 90 i 120 aktivnosti

	klaster1	**klaster2**	**klaster3**	**klaster4**
# instanci (skup za učenje)	303	358	23	132
# instanci (skup za validaciju)	136	180	10	82
# instanci (skup za testiranje)	272	372	34	138
predstavnik klastera	(0.0046, 0.0031)	(0.0002, 0.0004)	(0.0166, 0.0087)	(0.0100, 0.0054)



Slika 6.42: Kutijasti dijagram vrijednosti medijana mjere temeljene na udaljenosti za problem raspoređivanja s ograničenim sredstvima s 30, 60, 90 i 120 aktivnosti



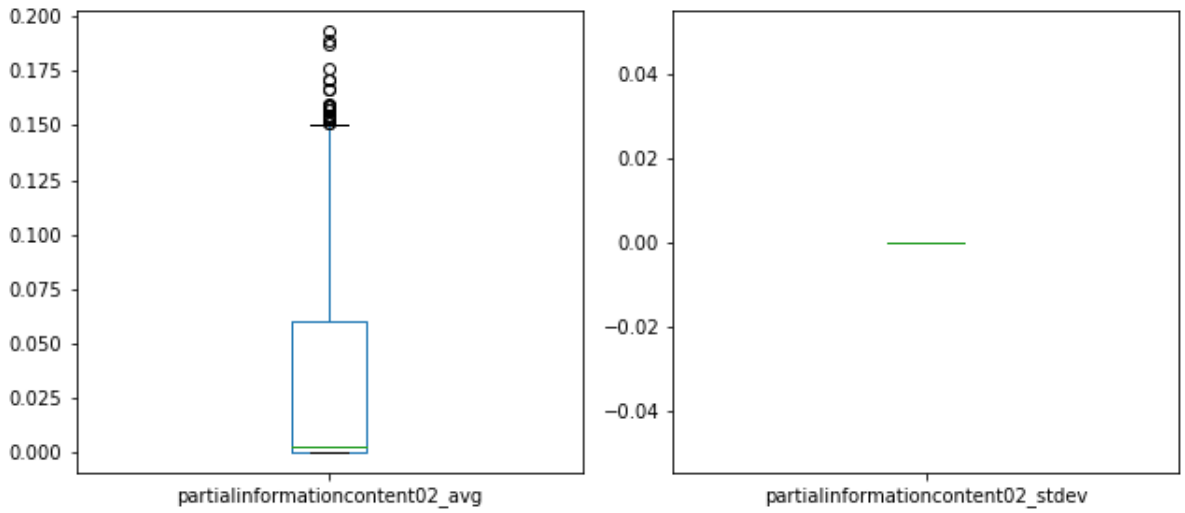
Slika 6.43: Dijagram raspršenja za medijan mjere temeljene na udaljenosti za problem raspoređivanja s ograničenim sredstvima s 30, 60, 90 i 120 aktivnosti

Tablica 6.43 daje prikaz broja instanci u skupovima za učenje, validaciju i testiranje te predstavnike za svaki od klastera dobivenih korištenjem djelomičnog sadržaja informacije sa $\varepsilon = 0.2$. U ovom slučaju prvi klaster sadrži otprilike pola promatranih instanci, dok je drugi klaster otprilike dva puta veći od trećeg. I ovdje raspodjela instanci u skupu za testiranje prati raspodjelu instanci iz skupa za učenje u smislu broja instanci u pojedinom klasteru.

Tablica 6.43: Broj instanci u klasterima i predstavnici klastera za djelomični sadržaj informacije ($\varepsilon = 0.2$) u problemu raspoređivanja s ograničenim sredstvima s 30, 60, 90 i 120 aktivnosti

	klaster1	**klaster2**	**klaster3**
# instanci (skup za učenje)	414	277	125
# instanci (skup za validaciju)	200	143	65
# instanci (skup za testiranje)	418	302	96
predstavnik klastera	(0.0002, 0.0)	(0.0851, 0.0)	(0.0132, 0.0)

Slika 6.44 daje prikaz raspona vrijednosti za aritmetičku sredinu i standardnu devijaciju djelomičnog sadržaja informacije sa $\varepsilon = 0.2$. Kod aritmetičke sredine je većina vrijednosti smještena blizu minimumu, dok su veće vrijednosti zapravo stršeće vrijednosti. Ono što je zanimljivo primijetiti je da kod standardne devijacije nema nikakve raspršenosti, odnosno, sve su vrijednosti jednake nuli. To znači da je u ovom slučaju grupiranje zapravo rađeno samo



Slika 6.44: Kutijasti dijagram vrijednosti djelomičnog sadržaja informacije ($\varepsilon = 0.2$) za problem raspoređivanja s ograničenim sredstvima s 30, 60, 90 i 120 aktivnosti

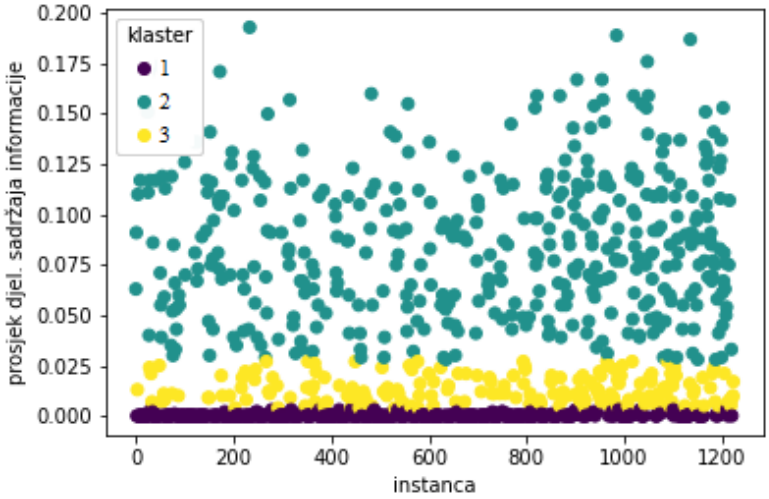
temeljem jedne vrijednosti promatrane značajke i to upravo na temelju aritmetičke sredine djelomičnog sadržaja informacije.

S obzirom da je standardna devijacija za djelomični sadržaj informacije jednaka nuli za svaku instancu, slika 6.45 daje prikaz dijagrama raspršenosti samo za aritmetičku sredinu promatrane mjere. Na x-osi se nalazi redni broj instance, dok je na y-osi vrijednost aritmetičke sredine djelomičnog sadržaja informacije sa $\varepsilon = 0.2$. Vidljivo je da su sve instance koje imaju najniže vrijednosti aritmetičke sredine (takvih je ujedno i najviše) smještene u prvi klaster. Drugi klaster sadrži one instance s najvišim vrijednostima aritmetičke sredine, dok se treći smjestio na srednjim vrijednostima. Vidljivo je i da su u drugom klasteru vrijednosti najraspršenije.

Tablica 6.44 daje prikaz broja instanci u skupovima za učenje, validaciju i testiranje za svaki od klastera dobivenih korištenjem donjeg kvartila mjere temeljene na udaljenosti bez nula. U istoj tablici su dane i vrijednosti aritmetičke sredine i standardne devijacije predstavnika svakog klastera. U četvrtom klasteru su sve one instance kojima su obje vrijednosti jednake nuli. Od preostalih instanci, najmanji broj pripada trećem klasteru u kojem se, prema vrijednostima predstavnika, nalaze instance s najvećim vrijednostima aritmetičke sredine i standardne devijacije. I ovdje, kao i kod prethodne dvije značajke promatrane u ovom problemu, raspodjela instanci u skupu za testiranje prati rasporedjelu instanci u skupu za učenje, u smislu broja instanci smještenih u pojedini klaster.

Slika 6.46 daje prikaz raspona vrijednosti za aritmetičku sredinu i standardnu devijaciju donjeg kvartila mjere temeljene na udaljenosti bez nula. Za ovu značajku krajolika dobrote su vrijednosti standardne devijacije puno manje raspršene od vrijednosti aritmetičke sredine. Maksimalne vrijednosti koje se postižu u oba slučaja su opet stršeće vrijednosti.

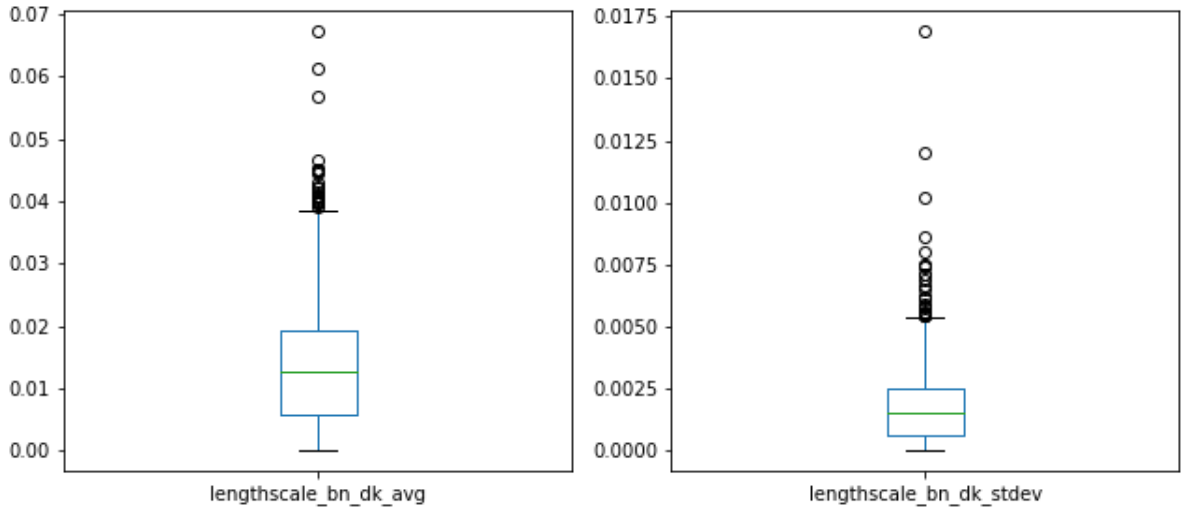
Slika 6.47 daje prikaz dijagrama raspršenosti za aritmetičku sredinu i standardnu devijaciju



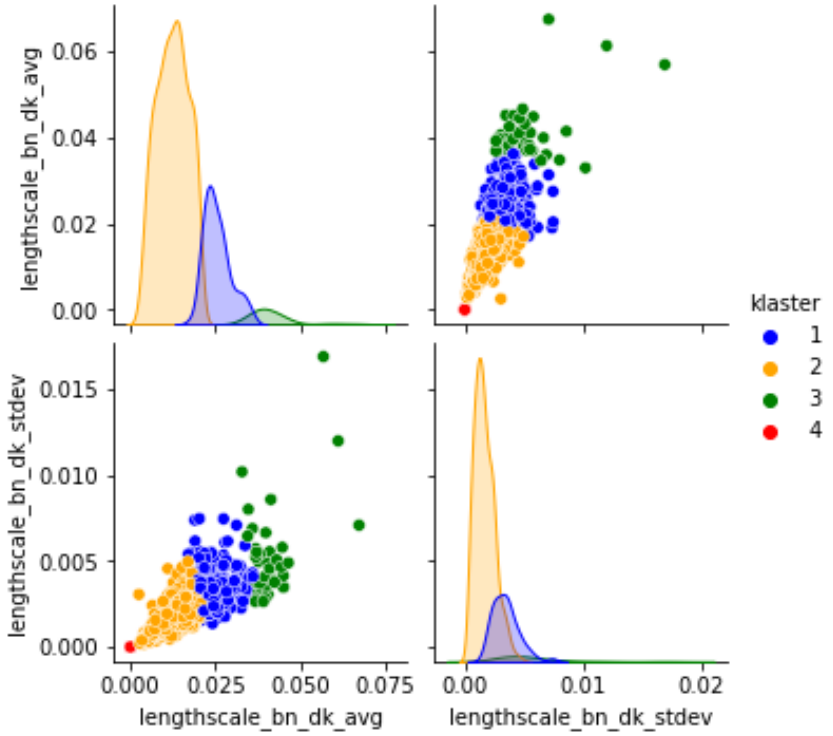
Slika 6.45: Dijagram raspršenja za djelomični sadržaj informacije ($\varepsilon = 0.2$) za problem raspoređivanja s ograničenim sredstvima s 30, 60, 90 i 120 aktivnosti

Tablica 6.44: Broj instanci u klasterima i predstavnici klastera za donji kvartil mjere temeljene na udaljenosti bez nula u problemu raspoređivanja s ograničenim sredstvima s 30, 60, 90 i 120 aktivnosti

	klaster1	**klaster2**	**klaster3**	**klaster4**
# instanci (skup za učenje)	137	496	24	159
# instanci (skup za validaciju)	84	236	13	75
# instanci (skup za testiranje)	156	473	25	162
predstavnik klastera	(0.0254, 0.0034)	(0.0123, 0.0016)	(0.0415, 0.0054)	(0.0, 0.0)



Slika 6.46: Kutijasti dijagram vrijednosti donjeg kvartila mjere temeljene na udaljenosti bez nula za problem raspoređivanja s ograničenim sredstvima s 30, 60, 90 i 120 aktivnosti



Slika 6.47: Dijagram raspršenja za donji kvartil mjere temeljene na udaljenosti bez nula za problem raspoređivanja s ograničenim sredstvima s 30, 60, 90 i 120 aktivnosti

donjeg kvartila mjere temeljene na udaljenosti bez nula. Kao što je već spomenuto, u četvrtom klasteru su samo one instance problema za koje su i aritmetička sredina i standardna devijacija promatrane značajke jednake nuli. Sa slike je vidljivo i da postoje neke vrijednosti koje poprilično odskaču od svih ostalih, pa su one smještene u treći klaster u kojem se nalaze instance problema s najvećim vrijednostima aritmetičke sredine i standardne devijacije. U klasteru 2 se nalaze instance problema kojima su aritmetička sredina i standardna devijacija najmanje, ali nisu nula. S obzirom da raspodjelu vrijednosti vidljivu na slici 6.46, ne čudi da se upravo u tom klasteru nalazi najviše instanci problema. S prikaza na sporednoj dijagonali može se uočiti blaga pozitivna koreliranost aritmetičke sredine i standardne devijacije donjeg kvartila mjere temeljene na udaljenosti bez nula.

6.7.2 Određivanje parametara

U ovom problemu se koriste instance problema jednake kao u [170], pa su i ovdje, kao i kod okoline nesrodnih strojeva, umjesto ručnog pretraživanja prostora parametara, početni parametri za GP preuzeti iz navedenog rada jer je u tom radu napravljeno detaljno pretraživanje na danim instancama. Tablica 6.45 daje prikaz parametara za GP koji će se koristiti za usporedbu.

Uz ručno određene parametre potrebno je i korištenjem okruženja *irace* odrediti parametre za svaki od dobivenih klastera. Za RCPSP sa svim aktivnostima, maksimalan broj eksperimenata dozvoljen za *irace* u svakom klasteru je izračunat kao $\lceil \frac{200}{broj\_klastera} \rceil$. I ovdje je, kao i u

Tablica 6.45: Parametri za GP u RCPSP-u s 30, 60, 90 i 120 aktivnosti, dobiveni ručnim pretraživanjem prostora parametara

naziv parametra	**vrijednost**
broj generacija	25
veličina populacije	1024
vjerojatnost mutacije	0.5
maksimalna dubina stabla	5
veličina turnira	7

```
# name          switch              type      values
tsize           "--tsize "            i       (3,7)
maxdepth        "--maxdepth "         i       (3,10)
popsize         "--popsize "          i       (200,1500)
mutProb         "--mutprob "          r       (0.01, 0.99)
```

Slika 6.48: Parametri za *irace* za RCPSP s 30, 60, 90 i 120 aktivnosti

svim prethodnim slučajevima korišten elitizam u kojem se samo najbolja konfiguracija sačuva za sljedeću iteraciju. Parametri koje *irace* određuje, kao i vrijednosti koje oni mogu poprimiti su vidljivi na slici 6.48.

Veličina turnira u turnirskom odabiru predstavljena je cjelobrojnim parametrom *tsize* koji može poprimiti vrijednosti iz raspona od 3 do 7. Drugi parametar čiju vrijednost treba odrediti je maksimalna dubina stabla. Taj parametar je ovdje označen sa *maxdepth* i poprima cjelobrojne vrijednosti iz raspona od 3 do 10. Veličina populacije (*popsize*) je također cjelobrojni parametar koji vrijednosti poprima iz raspona od 200 do 1500. Ovaj raspon je smanjen u odnosu na RCPSP sa samo 90 aktivnosti jer ovdje u svakom klasteru postoji znatno veći broj instanci, pa je za veće veličine populacije potrebno jako puno vremena kako bi GP došao do rješenja. Kao i kod svih prethodno promatranih problema, vjerojatnost mutacije je posljednji parametar čiju vrijednost treba odrediti. Ovdje je označen sa *mutProb* i može poprimiti vrijednosti iz intervala $[0.01, 0.99]$ uz točnost od dvije decimale. U ovom slučaju ne postoje zabranjene konfiguracije koje *irace* prilikom pronalaska parametara za GP treba izbjegavati.

Kao što je već prije spomenuto, za RCPSP s 30, 60, 90 i 120 aktivnosti se grupiranje radilo na temelju tri značajke krajolika dobrote: medijana mjere temeljene na udaljenosti, djelomičnog sadržaja informacije (uz $\varepsilon = 0.2$) i donjeg kvartila mjere temeljene na udaljenosti bez nula. U nastavku će biti opisani rezultati dobiveni po klasterima za svaku od navedenih značajki.

Tablica 6.46 daje vrijednosti veličine populacije, vjerojatnosti mutacije, maksimalne dubine stabla i veličine turnira za klastere dobivene na temelju medijana mjere temeljene na udaljenosti.

Zanimljivo je primijetiti da su vjerojatnosti mutacije podjednake u svim klasterima što znači da je za svaki dobiveni klaster podjednako potrebno imati mehanizam za izlazak iz lokalnog

Tablica 6.46: Parametri za GP dobiveni *irace*-om za svaki od klastera temeljen na medijanu mjere temeljene na udaljenosti za RCPSP s 30, 60, 90 i 120 aktivnosti

parametar	klaster1	klaster2	klaster3	klaster4
veličina populacije	1067	1230	914	1210
vjerojatnost mutacije	0.66	0.51	0.57	0.66
max dubina stabla	7	9	4	7
veličina turnira	5	6	3	7

optimuma ukoliko se u njemu zapne. Klaster 3 ima najmanji broj jedinki u populaciji i najmanju maksimalnu dubinu stabla što znači da je u tom klasteru dovoljno koristiti manje relativno malih stabala kako bi se došlo do zadovoljavajućeg rješenja promatranog problema.

Tablica 6.47 prikazuje osnovne statističke mjere za vrijednosti funkcije dobrote dobivene u 30 pokretanja GP-a na skupu za testiranje odgovarajućeg klastera korištenjem ručno, odnosno automatski određenih parametara.

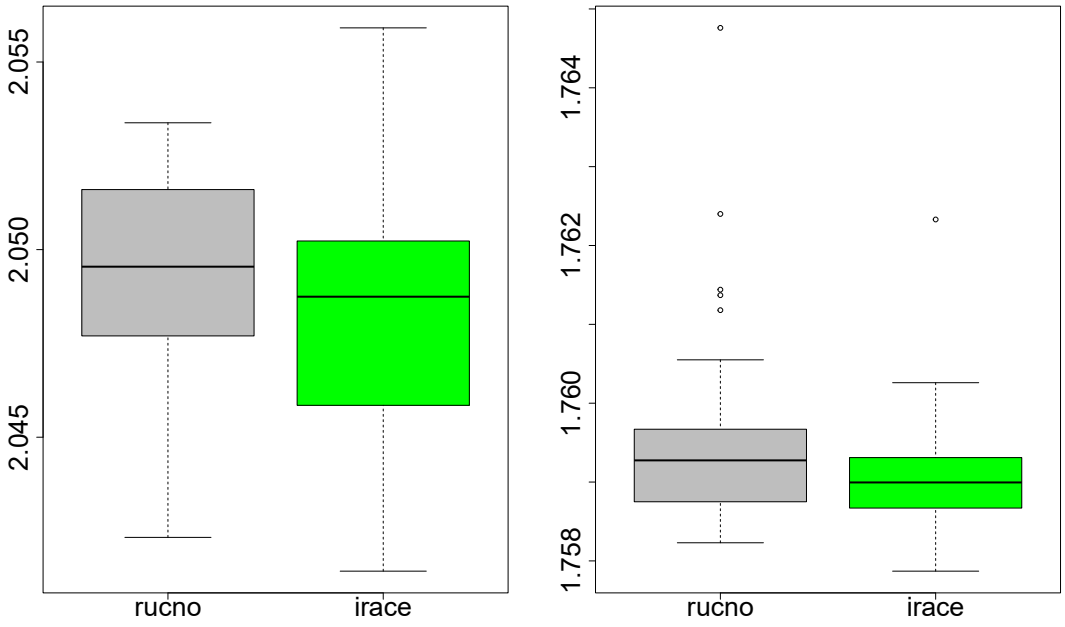
Tablica 6.47: Usporedba vrijednosti funkcije dobrote za klastere temeljene na medijanu mjere temeljene na udaljenosti za RCPSP s 30, 60, 90 i 120 aktivnosti

	klaster 1		klaster 2		klaster 3		klaster 4	
	ručno	irace	ručno	irace	ručno	irace	ručno	irace
min	2.0423	2.0414	1.7582	1.7579	3.1440	3.1492	2.5137	2.5152
1. kv.	2.0477	2.0462	1.7588	1.7587	3.1586	3.1622	2.5223	2.5212
medijan	2.0495	2.0487	1.7593	1.7590	3.1629	3.1648	2.5247	2.5244
prosjek	2.0492	2.0485	1.7596	1.7591	3.1645	3.1716	2.5246	2.5243
3. kv.	2.0515	2.0502	1.7597	1.7593	3.1688	3.1801	2.5284	2.5268
max	2.0534	2.0559	1.7648	1.7623	3.1958	3.2241	2.5389	2.5349
stdev	0.0032	0.0035	0.0014	0.0008	0.0118	0.0153	0.0049	0.0048
p-vrijednost	0.1591		0.0696		0.9631		0.3725	

Iz ove tablice je vidljivo da je u svim klasterima osim u trećem medijan vrijednosti dobivenih GP-om s automatski određenim parametrima bolji od medijana vrijednosti dobivenih sa ručno određenim parametrima. U svim klasterima je za obje skupine parametara raspršenost dosta niska. Slika 6.49 prikazuje usporedbu kutijastih dijagrama za GP s obje vrste parametara.

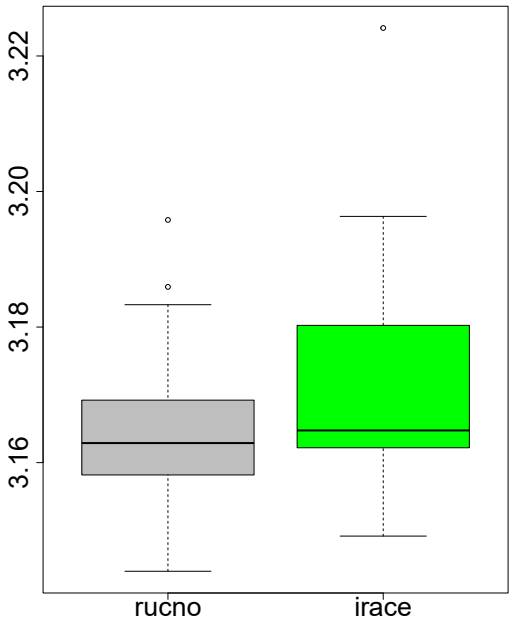
Tablica 6.48 daje vrijednosti promatranih GP parametara za klastere dobivene na temelju djelomičnog sadržaja informacije uz $\varepsilon = 0.2$.

U sva tri klastera je veličina populacije podjednaka, ali je zanimljivo primijetiti da vjerojatnost mutacije ide od jako niskih do jako visokih vrijednosti. Niska vrijednost mutacije je u

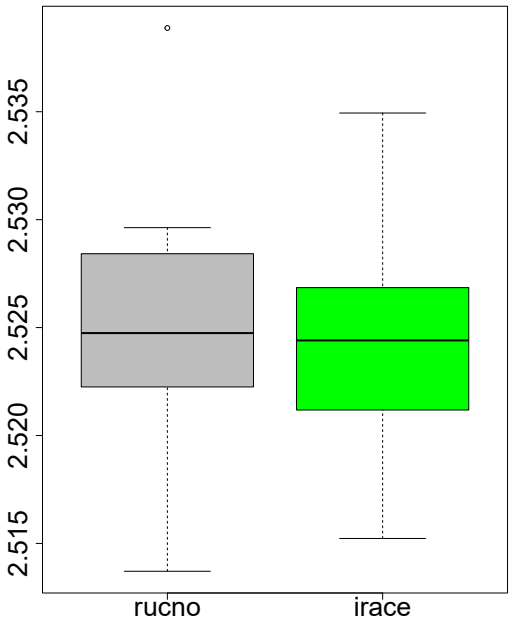


(a) klaster 1

(b) klaster 2



(c) klaster 3



(d) klaster 4

Slika 6.49: Prikaz kutijastih dijagrama dobivenih na temelju medijana mjere temeljene na udaljenosti za RCPSP s 30, 60, 90 i 120 aktivnosti

Tablica 6.48: Parametri za GP dobiveni *irace*-om za svaki od klastera temeljen na djelomičnom sadržaju informacije ($\varepsilon = 0.2$) za RCPSP s 30, 60, 90 i 120 aktivnosti

parametar	klaster1	klaster2	klaster3
veličina populacije	1465	1470	1387
vjerojatnost mutacije	0.09	0.47	0.98
max dubina stabla	9	7	9
veličina turnira	6	4	4

klasteru 1 koji ima najviše instanci, ali su to instance kojima su vrijednosti promatrane značajke krajolika dobrote jako niske, pa je očigledno krajolik dobiven u ovom klasteru gladak i nema potrebe za visokim vjerojatnostima mutacije. S druge strane, klaster 3 ima najveću vjerojatnost mutacije, a sadrži najmanje elemenata. No, ako se pogleda slika 6.45 vidljivo je da su u tom klasteru podaci najraspršeniji i imaju najrazličitije vrijednosti, pa je razumljivo da postoji veća vjerojatnost zapinjanja u lokalnom minimumu i zato vjerojatnost mutacije mora biti visoka.

Tablica 6.49 prikazuje minimum, donji i gornji kvartil, medijan, srednju vrijednost, maksimum i standardnu devijaciju za vrijednosti funkcije dobrote dobivene u 30 pokretanja GP-a na skupu za testiranje odgovarajućeg klastera korištenjem ručno određenih parametara, odnosno korištenjem parametara određenih automatski.

Tablica 6.49: Usporedba vrijednosti funkcije dobrote za klastere temeljene na djelomičnom sadržaju informacije ($\varepsilon = 0.2$) za RCPSP s 30, 60, 90 i 120 aktivnosti

	klaster 1		klaster 2		klaster 3	
	ručno	irace	ručno	irace	ručno	irace
min	1.7668	1.7663	2.4262	2.4246	2.0121	2.0132
1. kv.	1.7673	1.7671	2.4303	2.4295	2.0155	2.0169
medijan	1.7681	1.7676	2.4330	2.4324	2.0174	2.0207
prosjek	1.7682	1.7676	2.4329	2.4328	2.0182	2.0208
3. kv.	1.7683	1.7680	2.4353	2.4357	2.0216	2.0247
max	1.7751	1.7699	2.4427	2.4412	2.0276	2.0292
stdev	0.0015	0.0008	0.0036	0.0041	0.0041	0.0045
p-vrijednost	0.0367		0.3531		0.9897	

Vidljivo je da je vrijednost medijana postignuta GP-om sa automatski određenim parametrima za pojedini klaster bolja u 2 od 3 klastera nego vrijednost medijana postignuta sa ručno određenim klasterima. Jedino u klasteru 3 postignuta vrijednost nije bolja. Razlog tomu može biti što se u tom klasteru nalaze dosta različiti podaci, a ujedno ih je i najmanje, pa je *irace*-u

najteže bilo naučiti odgovarajuće parametre za njih. Možda bi više iteracija prilikom određivanja parametara dalo bolje rezultate. Slika 6.50 prikazuje usporedbu kutijastih dijagrama za GP s obje vrste parametara.

Posljednji parametar na temelju kojeg je rađeno grupiranje je donji kvartil mjere temeljene na udaljenosti bez nula. Tablica 6.50 daje GP parametre dobivene korištenjem *irace*-a za svaki od dobivenih klastera.

Tablica 6.50: Parametri za GP dobiveni *irace*-om za svaki od klastera temeljen na donjem kvartilu mjere temeljene na udaljenosti bez nula za RCPSP s 30, 60, 90 i 120 aktivnosti

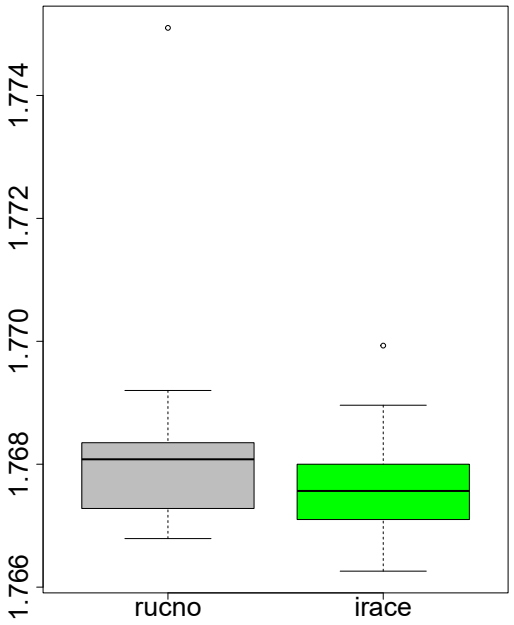
parametar	klaster1	klaster2	klaster3	klaster4
veličina populacije	864	1010	1161	1348
vjerojatnost mutacije	0.75	0.19	0.22	0.37
max dubina stabla	7	6	6	4
veličina turnira	4	3	3	4

Može se uočiti da u ovom slučaju, što je veći broj jedinki u populaciji, to je manja maksimalna dubina stabla za taj klaster. Vjerojatnosti mutacije dosta variraju od klastera do klastera, što znači da se ovom značajkom dobija grupiranje za koje u nekim klasterima nema potrebe za izlazak iz lokalnih optimuma, pa je vjerojatnost mutacije mala, dok kod nekih klastera postoje lokalni optimumi u koje GP lako može upasti, pa vjerojatnost mutacije treba biti veća kako bi postupak pretraživanja prostora stanja mogao izaći iz lokalnog optimuma u kojem je zapeo.

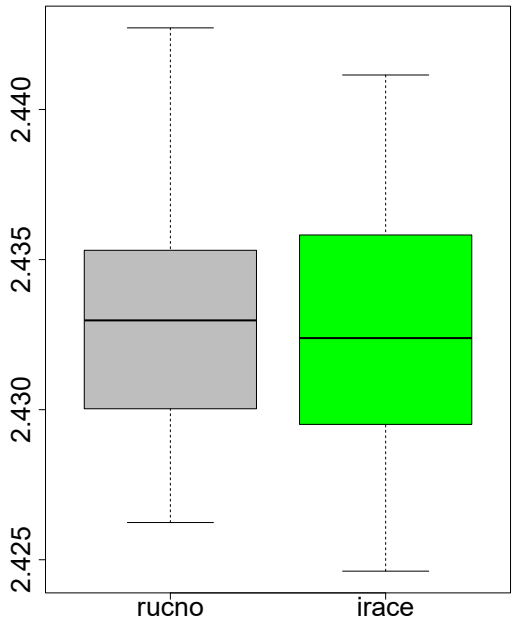
Tablica 6.51 daje usporedbu osnovnih statističkih mjera vrijednosti funkcije dobrote dobivene u 30 pokretanja GP-a sa ručno određenim i GP-a sa automatski određenim parametrima za svaki klaster.

U ovom slučaju je vidljivo da su u prvom i drugom klasteru vrijednosti medijana za GP sa ručno određenim parametrima bolje od vrijednosti medijana za GP sa automatski određenim parametrima. Samo u trećem klasteru se postiže poboljšanje korištenjem GP-a sa automatski određenim parametrima, dok je u 4. klasteru medijan jednak u oba slučaja. U 4. klasteru su grupirane one instance problema koje se mogu egzaktno riješiti, pa je GP s obje vrste parametara rezultirao jednakim vrijednostima. Može se doći do zaključka da što je lošije grupiranje (što je lošija vrijednost logaritamske vjerodostojnosti u slučaju grupiranja algoritmom EM), to će lošije rezultate dati parametri naučeni posebno za dani klaster. U tim klasterima se očigledno nisu našle instance problema koje su najsličnije jedne drugima, pa postupak automatskog odabira parametara ne može dobro naučiti koji parametri će dobro funkcionirati za takve probleme.

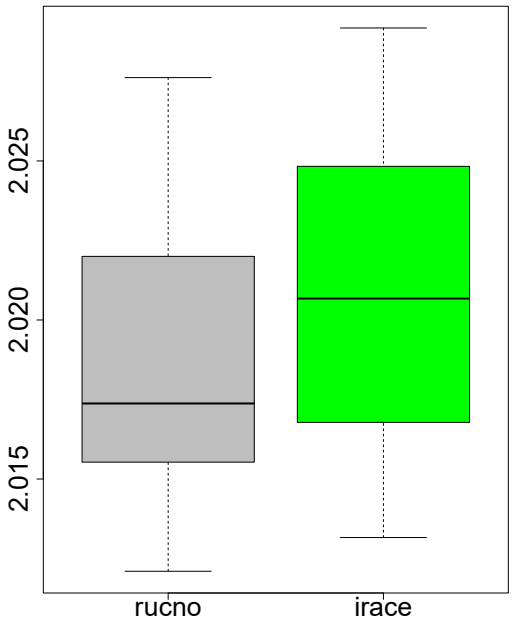
Slika 6.51 daje kutijaste dijagrame za vrijednosti dobivene GP-om sa ručno određenim parametrima i GP-om sa automatski određenim parametrima u prva tri klastera.



(a) klaster 1

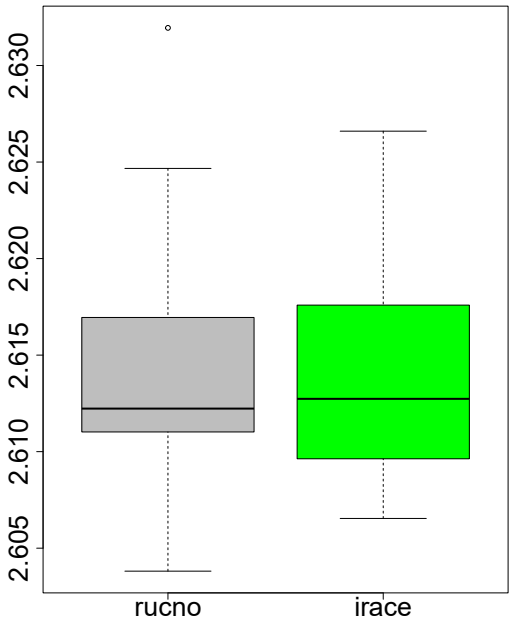


(b) klaster 2

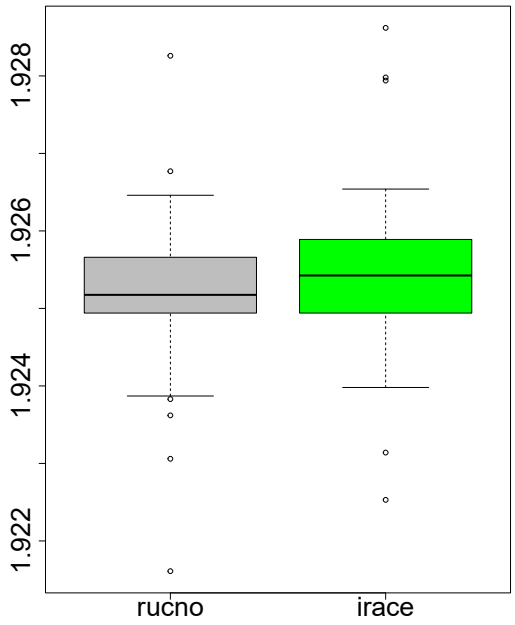


(c) klaster 3

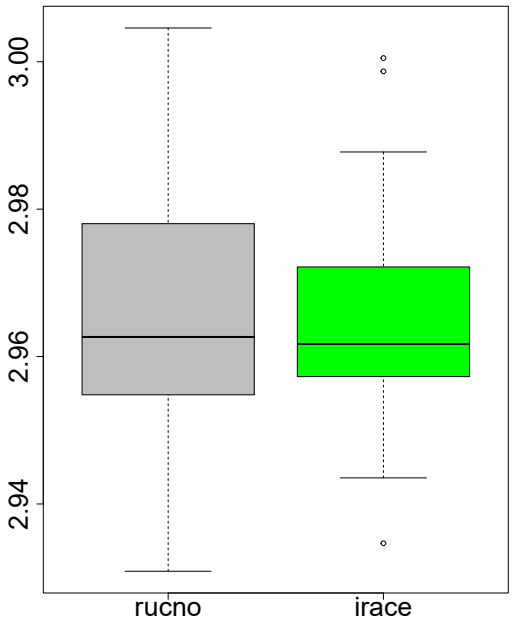
Slika 6.50: Prikaz kutijastih dijagrama dobivenih na temelju djelomičnog sadržaja informacije ($\varepsilon = 0.2$) za RCPSP s 30, 60, 90 i 120 aktivnosti



(a) klaster 1



(b) klaster 2



(c) klaster 3

Slika 6.51: Prikaz kutijastih dijagrama dobivenih na temelju donjeg kvartila mjere temeljene na udaljenosti bez nula za RCPSP s 30, 60, 90 i 120 aktivnosti

Tablica 6.51: Usporedba vrijednosti funkcije dobrote za klastere temeljene na donjem kvartilu mjere temeljene na udaljenosti bez nula za RCPSP s 30, 60, 90 i 120 aktivnosti

	klaster 1		klaster 2		klaster 3		klaster 4	
	ručno	irace	ručno	irace	ručno	irace	ručno	irace
min	2.6038	2.6065	1.9216	1.9225	2.9309	2.9347	1.7011	1.7011
1. kv.	2.6110	2.6097	1.9249	1.9249	2.9551	2.9574	1.7011	1.7011
medijan	2.6122	2.6127	1.9252	1.9254	2.9627	2.9617	1.7011	1.7011
prosjek	2.6141	2.6139	1.9251	1.9255	2.9646	2.9656	1.7011	1.7011
3. kv.	2.6169	2.6174	1.9257	1.9259	2.9774	2.9719	1.7011	1.7011
max	2.6320	2.6266	1.9283	1.9286	3.0046	3.0005	1.7011	1.7011
stdev	0.0058	0.0055	0.0012	0.0013	0.0168	0.0150	0.0000	0.0000
p-vrijednost	0.4383		0.8474		0.4853		1.0000	

6.8 Odabir značajki krajolika dobrote za grupiranje instanci problema

Na kraju prethodnog odjeljka može se vidjeti da, kako se krećemo prema lošijim grupiranjima, tako GP-u sa automatski određenim parametrima opadaju performanse, pa bi imalo smisla pokušati pronaći što bolja grupiranja. U svim provedenim eksperimentima grupiranja su rađena na temelju aritmetičke sredine i standardne devijacije samo jedne značajke krajolika dobrote. Ideja ovog odjeljka je ispitati može li se pronaći neka kombinacija parametara koja će dati bolje grupiranje, pa samim time i bolje rezultate sa automatski određenim parametrima za dobivene klastere. Kako se ne bi utrošilo previše vremena na ručno ispitivanje kombinacija izračunatih značajki krajolika dobrote, koristio se pristup automatiziranog odabira značajki za algoritam EM.

Pristupi automatiziranog odabira značajki se općenito mogu podijeliti u četiri kategorije: filter pristupi, pristupi omotača (engl. *wrappers*), hibridni pristupi i ugrađeni pristupi (engl. *embedded*) [177]. Općenito, pristupi automatskog odabira značajki mnogo se više istražuju u području nadziranog učenja (engl. *supervised learning*) nego što je to slučaj kod nenadziranog učenja (engl. *unsupervised learning*) u koje pripada i grupiranje podataka. Autori su u [177] učinili pregled metoda automatiziranog odabira značajki za grupiranje; jedna od tih metoda, definirana u [178] je korištena u ovom radu.

Ideja metode je sljedeća: implementira se sekvencijska pretraga unaprijed (engl. *sequential forward search*) koja počinje od praznog skupa značajki i dodaje jednu po jednu značajku u taj skup. Za svaku značajku koja se dodaje, pokreće se algoritam EM na instancama skupa koje

kao informaciju imaju vrijednost do sada odabranih značajki, pri čemu se za početne vrijednosti u tom algoritmu koriste vrijednosti dobivene algoritmom inicijalizacije poduzoraka. Algoritam inicijalizacije uzorkuje iz početne populacije uzoraka samo 10% jedinki i to ponavlja 10 puta. U svakoj od tih 10 iteracija, na dobivenom poduzorku se pokreće algoritam k-means sve dok se ne dobije k nepraznih klastera za taj poduzorak. Ukoliko je neki od klastera prazan nakon prvog pokretanja algoritma k-means, centroid tog klastera se postavlja na jedinku iz poduzorka koja je najudaljenija od trenutnog centroida praznog klastera. Postupak se ponavlja sve dok svi klasteri ne budu neprazni. Nakon što se za svaki poduzorak dobiju centoridi klastera, svi se kombiniraju u jedan skup podataka koji se zatim opet grupira korištenjem algoritma k-means. Onaj centroid koji maksimizira vjerodostojnost kombiniranog skupa centroida odabire se kao početna točka za pokretanje algoritma EM s početka.

Za svaki skup značajki u sekvencijskoj pretrazi unaprijed se pokreće algoritam EM i nakon konvergencije se dobivene vrijednosti kriterijske funkcije uspoređuju s vrijednostima kriterijske funkcije za svaki skup značajki koji ima jednak broj značajki kao i trenutno promatrani skup. Ona značajka koja rezultira najboljom vrijednošću kriterijske funkcije dodaje se u skup dosad odabranih značajki. Nakon toga se opet tom skupu dodaje jedna po jedna značajka i postupak se ponavlja.

Nakon jedne iteracije dodavanja značajki u trenutni skup značajki i dobivanja novog skupa značajki (npr. skup koji sadrži dvije značajke), postupak se ponavlja i na taj način se dobija novi skup značajki s jednom značajkom više od prethodnog skupa (npr. skup koji sadrži tri značajke). Kako bi se vrijednosti kriterijske funkcije mogle usporediti za skupove s različitim brojem značajki, koristi se metoda normalizacije unakrsnom projekcijom (engl. *cross projection normalization*). Sekvencijska pretraga unaprijed se provodi sve dok dodavanje dodatne značajke u trenutni skup značajki rezultira poboljšanjem vrijednosti kriterijske funkcije.

Još jedno pitanje koje se javlja u ovom postupku je koliko klastera bi trebalo odabrati prilikom grupiranja instanci algoritmom EM. U ovom postupku se broj klastera određuje automatizirano. Proslijeđuje se maksimalan mogući broj klastera i zatim se za svaku značajku i za svaki broj klastera od 2 do danog maksimalnog broja klastera pokreće algoritam EM kako je već opisano. S obzirom da se procjena maksimalne vjerodostojnosti povećava korištenjem više klastera, koristi se kriterijska funkcija s uključenim Bayesovim informacijskim kriterijem kao kaznom za veći broj klastera kako bi se dobivene vrijednosti kriterijske funkcije mogle uspoređivati.

Opisani postupak je proveden za RCPSP s 90 aktivnosti i u nastavku će biti opisani dobiveni rezultati.

Opisani postupak pokazao je da se instance problema trebaju grupirati u 8 klastera na temelju tri značajke krajolika dobrote: aritmetička sredina minimuma mjere temeljene na udaljenosti bez nula, standardna devijacija minimuma mjere temeljene na udaljenosti bez nula i standardna

devijacija medijana mjere temeljene na udaljenosti.

Tablica 6.52 daje prikaz broja instanci u skupovima za učenje, validaciju i testiranje za svaki od 8 klastera dobivenih korištenjem značajki dobivenih postupkom automatiziranog odabira značajki. U istoj tablici su dane i srednje vrijednosti pojedine značajke za svaki od klastera (prva vrijednost predstavlja aritmetičku sredinu minimuma mjere temeljene na udaljenosti bez nula, druga vrijednost standardnu devijaciju minimuma mjere temeljene na udaljenosti bez nula, a treća standardnu devijaciju medijana mjere temeljene na udaljenosti).

Tablica 6.52: Broj instanci u klasterima i predstavnici klastera za značajke dobivene automatiziranim odabirom značajki za RCPSP s 90 aktivnosti

	klaster1	**klaster2**	**klaster3**	**klaster4**
# instanci (skup za učenje)	11	8	42	51
# instanci (skup za validaciju)	6	4	17	24
# instanci (skup za testiranje)	16	7	23	50
predstavnik klastera	(0.0009, 0.0004, 0.0023)	(0.0008, 0.0004, 0.0029)	(0.0012, 0.0003, 0.0028)	(0.0, 0.0, 0.0)
	klaster5	**klaster6**	**klaster7**	**klaster8**
# instanci (skup za učenje)	31	18	22	9
# instanci (skup za validaciju)	16	11	12	6
# instanci (skup za testiranje)	30	21	24	21
predstavnik klastera	(0.0008, 0.0004, 0.0020)	(0.0013, 0.0005, 0.0)	(0.0009, 0.0004, 0.0042)	(0.0011, 0.0003, 0.0053)

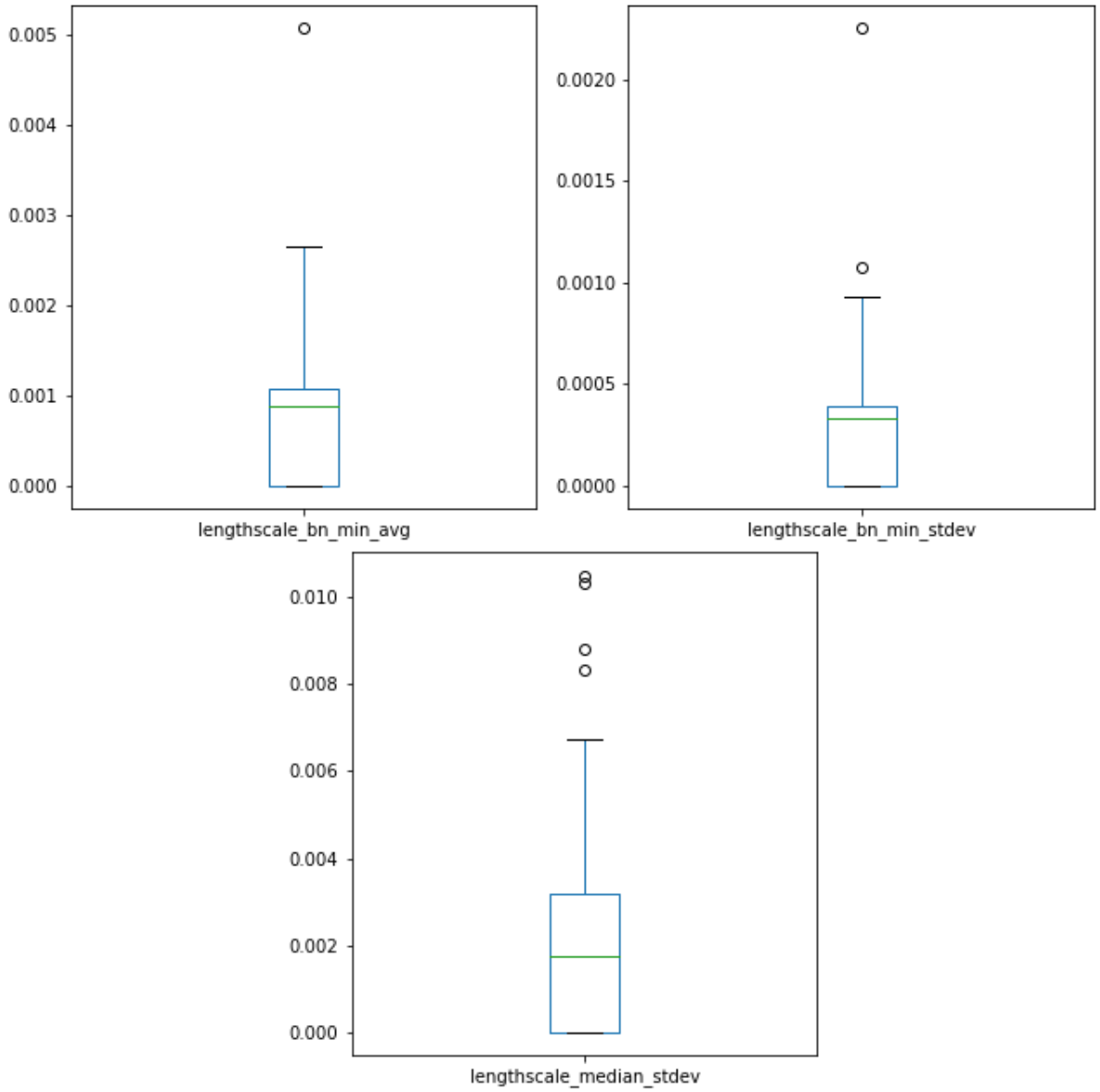
Zanimljivo je uočiti da ova vrsta grupiranja otprilike jednako grupira instance skupa za učenje i skupa za testiranje u smislu da je u svakom od klastera podjednak broj instanci u skupu za učenje i u skupu za testiranje. Klaster 4 je najbrojniji, kod njega je predstavnik uređena trojka kojoj su sve tri komponente jednake nuli, ali ukoliko se pogleda prikaz raspona vrijednosti za sve tri korištene značajke na slici 6.52, ta informacija ne čudi jer se za sve tri značajke većina vrijednosti nalazi upravo oko minimuma koji je jednak nuli. Na slici 6.52 mjera temeljena na udaljenosti je označena sa *lengthscale*, a mjera temeljena na udaljenosti bez nula sa *lengthscale_bn*.

Slika 6.53 daje prikaz dijagrama raspršenosti za sve kombinacije značajki odabranih postupkom automatiziranog odabira značajki.

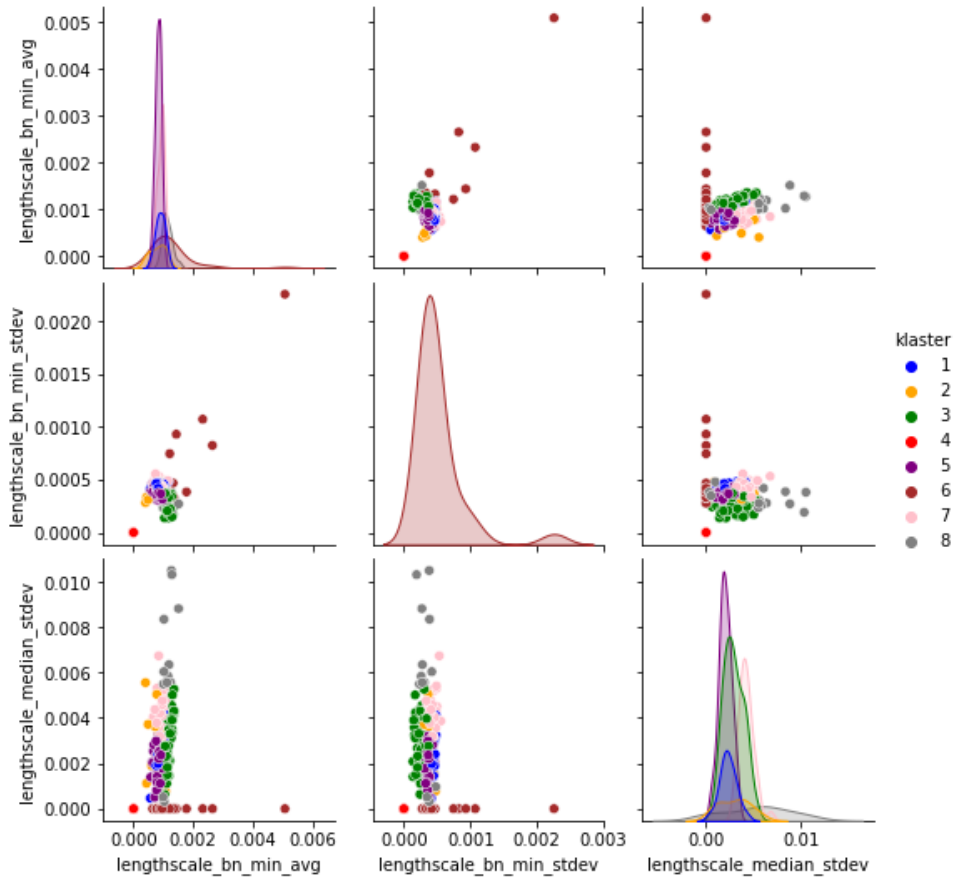
Vidljivo je da su u klasteru 4 sve instance kojima su sve tri značajke jednake nula, pa su one na ovom dijagramu prikazane samo jednom točkom. Između promatranih značajki se sa prikazanih grafova ne može uočiti neka koreliranost, što znači da je ovaj postupak odabira značajki dodavao značajke u skup tako da daju dosad neviđene informacije o instancama problema.

Nakon postupka grupiranja, potrebno je za svaki od dobivenih klastera korištenjem okruženja *irace* odrediti parametre za GP za odgovarajuće skupove za učenje. Pretpostavke za *irace* su jednake kao u pododjeljku 6.6.2.

Tablica 6.53 daje vrijednosti veličine populacije, vjerojatnosti mutacije, maksimalne dubine stabla i veličine turnira za klastere dobivene na temelju značajki odabranih postupkom automa-



Slika 6.52: Kutijasti dijagrami vrijednosti značajki dobivenih automatiziranim odabirom značajki za RCPSP s 90 aktivnosti



Slika 6.53: Dijagram raspršenja za tri značajke dobivene automatiziranim odabirom značajki za RCPSP s 90 aktivnosti

tiziranog odabira značajki.

Tablica 6.53: Parametri za GP dobiveni *irace*-om za svaki od klastera temeljen na značajkama određenim postupkom automatiziranog odabira značajki za RCPSP s 90 aktivnosti

parametar	klaster1	klaster2	klaster3	klaster4
veličina populacije	1039	505	861	809
vjerojatnost mutacije	0.95	0.16	0.44	0.31
max dubina stabla	6	3	5	7
veličina turnira	3	6	3	5
parametar	klaster5	klaster6	klaster7	klaster8
veličina populacije	1059	649	1219	840
vjerojatnost mutacije	0.34	0.63	0.78	0.47
max dubina stabla	6	8	5	9
veličina turnira	4	4	6	5

Kao i u svim prethodnim slučajevima, i ovdje je *irace* dao različite parametre za svaki od klastera što znači da se instance problema mogu razlikovati po nekim značajkama i da ima smisla za svaku skupinu instanci tražiti najbolje GP parametre. Vjerojatnosti mutacije variraju od vrlo niskih do vrlo visokih, što znači da se u nekima od klastera nalaze takve instance da postoje lokalni minimumi u kojima proces pretraživanja rješenja može zapeti, dok je u nekim klasterima dobiven krajolik gladak, pa nema potrebe za velikim vjerojatnostima mutacije jer nema opasnosti od zaglavljivanja u lokalnim minimumima.

Tablica 6.54 daje usporedbu osnovnih statističkih mjera vrijednosti funkcije dobrote dobivene u 30 pokretanja GP-a s ručno određenim parametrima (koji su jednaki kao u pododjeljku 6.6.2 jer se radi o istim instancama problema) i vrijednosti funkcije dobrote dobivene u 30 pokretanja GP-a s automatski određenim parametrima za odgovarajući klaster.

Iz tablice se vidi da je u 5 od 8 klastera vrijednost medijana dobivena automatski određenim parametrima manja od vrijednosti medijana dobivene GP-om sa ručno određenim parametrima. U klasteru 4 su vrijednosti jednake u oba slučaja. Podsjetimo se, to je klaster u kojemu se nalaze one instance problema kojima su sve tri značajke jednake nuli. Ove instance problema sudeći po rezultatima ne stvaraju problem GP-u i GP ih uspijeva riješiti optimalno u oba slučaja. U dva klastera u kojima je vrijednost medijana manja kod GP-a sa ručno određenim parametrima, razlika u medijanu je manja od 0.0006. S druge strane, u klasterima gdje je GP sa automatski određenim parametrima bolji, razlika u medijanima se kreće u vrijednostima od 0.002 do 0.0035. Iz toga je vidljivo da je poboljšanje postignuto korištenjem automatski određenih parametara puno veće nego pogoršanje u dva klastera u kojima se ono dogodi. Iako ovaj postupak

Tablica 6.54: Usporedba vrijednosti funkcije dobrote za klastere temeljene na značajkama dobivenim automatiziranim odabirom značajki za RCPSP s 90 aktivnosti

	klaster 1		klaster 2		klaster 3		klaster 4	
	ručno	irace	ručno	irace	ručno	irace	ručno	irace
min	1.8456	1.8438	2.1071	2.0929	1.9449	1.9467	1.6555	1.6555
1. kv.	1.8578	1.8564	2.1286	2.1335	1.9501	1.9506	1.6555	1.6555
medijan	1.8627	1.8615	2.1486	2.1461	1.9559	1.9565	1.6555	1.6555
prosjek	1.8628	1.8607	2.1557	2.1535	1.9569	1.9580	1.6555	1.6555
3. kv.	1.8663	1.8651	2.1671	2.1688	1.9622	1.9662	1.6555	1.6555
max	1.8849	1.8820	2.2505	2.2417	1.9783	1.9714	1.6555	1.6555
stdev	0.0087	0.0079	0.0382	0.0345	0.0088	0.0080	0.0000	0.0000
p-vrijednost	0.1996		0.5177		0.7471		1.0000	
	klaster 5		klaster 6		klaster 7		klaster 8	
	ručno	irace	ručno	irace	ručno	irace	ručno	irace
min	1.7725	1.7725	1.7487	1.7488	2.3715	2.3634	2.2505	2.2479
1. kv.	1.7787	1.7786	1.7507	1.7506	2.3788	2.3767	2.2679	2.2656
medijan	1.7802	1.7807	1.7571	1.7537	2.3832	2.3826	2.2783	2.2766
prosjek	1.7816	1.7826	1.7556	1.7549	2.3848	2.3828	2.2832	2.2835
3. kv.	1.7845	1.7857	1.7588	1.7588	2.3892	2.3863	2.2895	2.2906
max	1.7960	1.8039	1.7633	1.7691	2.4139	2.4091	2.3609	2.3609
stdev	0.0051	0.0065	0.0043	0.0049	0.0105	0.0099	0.0246	0.0273
p-vrijednost	0.5992		0.2017		0.2998		0.4267	

ne daje bolja rješenja u svim slučajevima, ipak se postiže poboljšanje, a to i je cilj korištenja hiperheuristika.

Dodatna poboljšanja mogla bi se postići korištenjem nekog drugog oblika automatiziranog odabira značajki na temelju kojih će se vršiti grupiranje ili korištenjem nekog drugog algoritma grupiranja.

6.9 Vrijeme izvršavanja

Već je prije napomenuto da se odabiru one značajke krajolika dobrote koje se mogu relativno brzo izračunati kako sâmo računanje značajki ne bi trajalo dulje od rješavanja početnog problema. U ovom odjeljku daje se kratka analiza vremena izvršavanja za problem raspoređivanja s ograničenim sredstvima s 90 aktivnosti.

Sva mjerenja vremena izvršavanja rađena su na računalu s Intel i3-5005U CPU@2.00GHz procesorom i 8GB radne memorije. Na ovom računalu, GP na negrupiranim podacima (na cijelom skupu za učenje) treba otprilike 8384 sekunde kako bi riješio dani problem uz korištenje 25 generacija i 1024 jedinke u populaciji. S druge strane, računanje 30 slučajnih šetnji za jednu instancu problema traje otprilike 1.33 sekunde, dok računanje udaljenosti između svih parova stabala u dobivenim slučajnim šetnjama traje u prosjeku 1819 sekundi. Kako bi se odredila vrijednost dobrote svih stabala u svim slučajnim šetnjama, potrebno je u prosjeku 124.49 sekundi po instanci. Nakon računanja slučajnih šetnji i odgovarajućih udaljenosti, mogu se izračunati mjere temeljene na udaljenosti. Za računanje medijana mjere temeljene na udaljenosti za jednu instancu problema potrebno je u prosjeku 50.8 sekundi. Većina tog vremena izvršavanja odlazi na učitavanje izračunatih dobrota stabala i udaljenosti između stabala u slučajnoj šetnji. Za računanje donjeg kvartila mjere temeljene na udaljenosti bez nula u prosjeku treba 48.1 sekundu po instanci, dok je za prosjek mjere temeljene na udaljenosti potrebno otprilike 50.6 sekundi. Pri svemu ovome treba imati na umu da se analiza krajolika dobrote radi samo jednom, a zatim je za sve do tada neviđene instance potrebno samo odrediti kojem one klasteru pripadaju i primijeniti GP s parametrima određenim za taj klaster.

Poglavlje 7

Odabir GP parametara klasifikacijom

Postupak u prethodnom poglavlju zahtjeva određivanje GP parametara za svaki dobiveni klaster korištenjem postupka automatiziranog određivanja. U ovom poglavlju se razmatra može li se unaprijed definirati nekoliko skupova parametara iz kojih se onda može odabrati koji skup koristiti za dani problem.

Postupak koji se koristi je baziran na ideji iz [169]. U spomenutom radu se pokušava odrediti koje od automatski razvijenih prioritetnih pravila iskoristiti za rješavanje zadane instance problema u okolini nesrodnih strojeva. Pri tome se koriste određena svojstva instance kako bi se problemi mogli klasificirati i kako bi se na temelju klase kojoj pripadaju odredilo koje prioritetno pravilo iskoristiti. Korištena svojstva se odnose na sama svojstva problema koja se mogu iščitati iz opisa instance, poput očekivanog ukupnog trajanja dobivenog rasporeda, raspona željenog vremena završetka, omjera broja strojeva i poslova i sl. U ovom poglavlju će se za svojstva promatrati značajke krajolika dobrote, a kao problem na kojem će se izvedeni pristup isprobati je odabran RCPSP s 30, 60, 90 i 120 aktivnosti.

S obzirom da se radi o klasifikaciji, potrebno je formirati skup za učenje u kojemu će se točno znati koja instanca pripada kojoj klasi. U nastavku će biti opisano kako točno se dobiva skup za učenje.

Prvi korak je razdijeliti instance RCPSP problema na tri skupa: skup za učenje na kojem će se učiniti grupiranje (ovaj skup će u nastavku teksta biti označen sa: skup za učenje *A*), skup za učenje na kojem će se definirati kako točno izgledaju klase (ovaj skup će u nastavku teksta biti označen sa: skup za učenje *B*) i skup za testiranje u kojem će biti instance za koje će se uspoređivati rezultati GP-a s unaprijed definiranim parametrima i GP-a s parametrima određenima za klasu kojoj instanca problema pripada. Kao što je već navedeno u pododjeljku 6.2.3, RCPSP s 30, 60, 90 i 120 problema ima sveukupno 2040 instanci problema. Ovdje se taj skup problema dijeli tako da se u skupu za učenje *A* nalazi 612 instanci problema, u skupu za učenje *B* se nalazi 816 instanci problema (612 u skupu za učenje i 204 u skupu za validaciju), dok se u skupu za testiranje nalazi preostalih 612 instanci problema.

Skup za učenje A služi kako bi se učinilo grupiranje problema u klastere. To se radi jer GP rijetko rješava samo jednu instancu problema. GP se većinom koristi kako bi riješio neku skupinu problema odjednom. Iz tog razloga je potrebno načiniti grupe kako bi se u skupu za učenje B našlo više sličnih problema za koje onda treba odrediti odgovarajuće parametre za GP. Za određivanje grupa je korišten postupak automatiziranog odabira značajki i grupiranja temeljenog na odabranim značajkama opisan u odjeljku 6.8. Pri tome je maksimalan broj mogućih grupa postavljen na $K = 60$. Ovaj postupak je rezultirao odabirom dvije značajke krajolika dobrote i to: standardnom devijacijom i aritmetičkom sredinom minimuma mjere temeljene na udaljenosti bez nula. Na temelju te dvije značajke, instance problema iz skupa za učenje A raspoređene su u 55 klastera.

Kako bi se mogao provesti postupak klasifikacije problema iz skupa za testiranje, potrebno je načiniti skup za učenje (u nastavku teksta ovaj skup će biti označen sa: skup za učenje C) u kojem će za svaki skup problema biti poznate vrijednosti promatranih značajki i klasa kojoj ta značajka pripada. Na temelju dvije značajke za koje je napravljeno grupiranje u skupu za učenje A, se i instance problema iz skupa za učenje B (svih 816) grupiraju u prethodno određene klastere. Na taj način se u skupu za učenje C dobije 55 grupa problema. Za svaku grupu problema se odredi predstavnik grupe, odnosno klastera i to kao prosječna vrijednost značajki na temelju kojih je rađeno grupiranje za sve instance problema u odgovarajućoj grupi. Na taj način su dobiveni podaci na kojima će se učiti postupak klasifikacije, samo je potrebno za svaki od tih podataka odrediti kojoj klasi pripada. Ovdje je situacija specifična jer će klasa zapravo označavati koji će se skup parametara primijeniti za koju skupinu problema. U tu svrhu je potrebno definirati analogne skupove parametara.

Kao i u prethodnom poglavlju, parametri koji variraju su veličina populacije, vjerojatnost mutacije, maksimalna dubina stabla i veličina turnira. Generirano je 29 različitih skupova parametara za GP u kojima su vrijednosti za veličine turnira odabrane slučajnim odabirom iz raspona od 500 do 1500 s korakom 50, vjerojatnost mutacije je birana iz raspona 0.1 do 0.9 s korakom 0.05, vrijednosti za veličinu turnira dolaze iz cjelobrojnog skupa od 3 do 7, dok se maksimalna dubina stabla dobija slučajnim odabirom iz skupa vrijednosti od 3 do 10. Dodatno je dodan još jedan skup parametara i to onaj kojemu su vrijednosti promatranih parametara ručno određene za cijeli skup problema u RCPSP-u sa 30, 60, 90 i 120 aktivnosti (tablica 6.45). Tablica 7.1 daje prikaz generiranih vrijednosti za sve korištene skupove GP parametara.

Sljedeći korak je odrediti koji od generiranih GP parametara pridružiti kojem skupu problema iz skupa za učenje kako bi se dobili podaci na kojima se može učiti klasifikacija. U tu svrhu se koristi modificirana shema pridruživanja grupiranjem (engl. *grouping association scheme* - GAS) objašnjena u [169].

Postupak za GAS je sljedeći: neka je sa P označen skup svih predstavnika klastera iz skupa za učenje B, a sa R skup svih GP parametara. Prvi korak je evaluirati GP sa svakim skupom

Tablica 7.1: Parametri korišteni prilikom kreiranja podataka za klasifikaciju

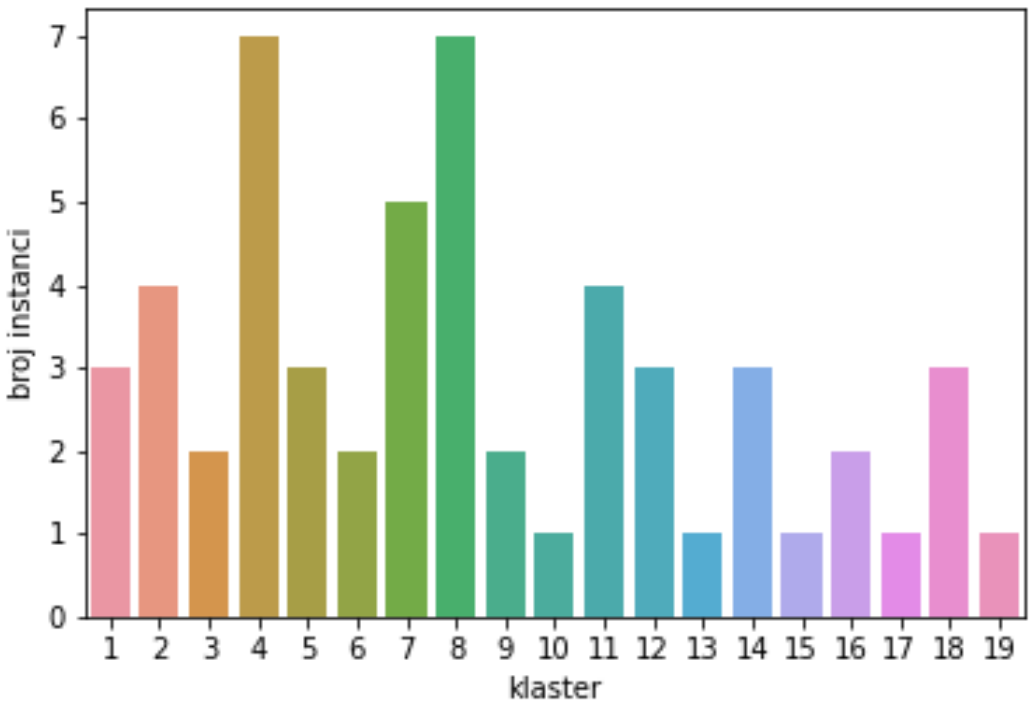
	vel. populacije	vjer. mutacije	max dubina	vel. turnira
parametri_1	1024	0.3	5	7
parametri_2	900	0.75	6	5
parametri_3	500	0.9	10	5
parametri_4	1400	0.25	4	5
parametri_5	850	0.85	5	7
parametri_6	900	0.75	7	4
parametri_7	1000	0.2	6	3
parametri_8	1150	0.2	6	3
parametri_9	1350	0.4	4	4
parametri_10	1050	0.65	7	5
parametri_11	1250	0.5	9	6
parametri_12	900	0.6	4	3
parametri_13	1200	0.65	7	7
parametri_14	900	0.4	7	6
parametri_15	1100	0.55	6	6
parametri_16	1100	0.3	6	6
parametri_17	1050	0.6	8	7
parametri_18	1350	0.3	8	4
parametri_19	1150	0.75	9	3
parametri_20	800	0.45	8	6
parametri_21	1300	0.2	9	5
parametri_22	1350	0.4	6	5
parametri_23	1400	0.6	5	5
parametri_24	1200	0.55	9	7
parametri_25	1100	0.4	9	4
parametri_26	1050	0.4	5	7
parametri_27	1300	0.6	9	3
parametri_28	900	0.1	8	6
parametri_29	1000	0.25	7	7
parametri_30	1050	0.55	7	6

parametara iz skupa R na svakom od klastera u skupu za učenje. S obzirom da je GP heuristika, i dobiveni rezultati mogu varirati u različitim pokretanjima, za svaku kombinaciju skupa parametara i klastera je GP pokrenut 10 puta i kao rezultat je odabrana prosječna vrijednost funkcije cilja dobivena u tih 10 pokretanja. Funkcija cilja koja se ovdje koristi je jednaka kao u odjeljku 6.2.3 za promatrani problem.

Za svaki od klastera se zatim definira skup *opt* u koji se spremaju svi parametri za koje se postiže najbolja vrijednost u tom klasteru. Može se dogoditi da više različitih skupova parametara da najbolje rješenje za promatrani klaster pa skup *opt* može imati jedan ili više elemenata. Ukoliko skup *opt* sadrži samo jedan element, promatrani klaster se proglašava riješenim i parametri za koje se postiže to najbolje rješenje se pridružuju promatranom klasteru. Parametri koji se pridruže nekom od klastera se zapisuju u skup *plist*. Na ovaj način se osigurava da se u *plist* spremaju svi potrebni skupovi parametara, odnosno osigurava se da za svaki od promatranih klastera u novonastalom skupu za učenje postoji skup parametara za koji se postižu najbolja rješenja. Ukoliko je skup *opt* za promatrani klaster jednočlan, a odgovarajući skup parametara se ne zapisuje u *plist*, tada se za promatrani klaster neće moći postići najbolje moguće rješenje.

Nakon što se pronađu svi klasteri za koje samo jedan skup parametara postiže najbolje rješenje, definira se skup *nonopt* onih klastera za koje je presjek skupa *opt* i skupa *plist* trenutno iskorištenih parametara, jednak praznom skupu. Odnosno, u skupu *nonopt* će se naći svi oni klasteri za koje u skupu *plist* ne postoje parametri koji omogućuju postizanje najboljeg mogućeg rješenja na promatranom klasteru. Zatim se za svaki skup parametara koji nije u skupu *plist* definira brojač koji broji za koliko se klastera iz skupa *nonopt* može dobiti najbolje rješenje korištenjem promatranog skupa parametara. Onaj skup parametara koji ima najveću vrijednost brojača se dodaje u skup *plist*. Ovime se osigurava da se u skup *plist* doda najmanji potreban broj skupova parametara koji omogućuje da se za svaki klaster može pronaći najbolje moguće rješenje, kako bi na kraju postupak klasifikacije trebao učiti klasifikaciju u manji broj klasa. Ukoliko više skupova parametara ima najveću vrijednost brojača, u *plist* se dodaje onaj skup koji ima najbolju ukupnu vrijednost funkcije cilja na svim instancama problema. Ukupna vrijednost funkcije cilja se računa kao suma vrijednosti funkcije cilja koje se dobiju primjenom odgovarajućeg skupa parametara na svaku instancu problema. Svi klasteri za koje je skup parametara s najvećom vrijednošću brojača dao najbolje rješenje uklanjaju se iz skupa *nonopt* i označavaju da su riješeni. Postupak se nastavlja sve dok je skup *nonopt* neprazan.

Nakon ovog postupka, u skupu *plist* se nalazi namanji mogući broj skupova parametara koji osigurava da se za svaki klaster može pronaći najbolje moguće rješenje uz dane parametre. No, i dalje mogu postojati oni klasteri kojima nije dodijeljen jedan od tih skupova parametara. Za sve takve klastere koji još nisu označeni kao riješeni, za svaki skup parametara iz *plist* koji daje najbolje rješenje na promatranom klasteru koji još nije riješen, treba izračunati euklidsku udaljenost između vrijednosti odabranih značjki krajolika dobrote predstavnika pro-



Slika 7.1: Broj instanci u svakoj od dobivenih klasa u skupu za učenje

matranog klastera i vrijednosti odabranih značajki krajolika dobrote svih predstavnika klastera kojima je već dodijeljen promatrani skup parametara. Onaj skup parametara koji daje najmanju prosječnu euklidsku udaljenost se dodjeljuje promatranom klasteru. Ovaj opisani postupak je zapravo jednostavan postupak grupiranja čiji je cilj dodijeliti odgovarajuće parametre promatranim klasterima i pri tome paziti da skup za učenje ostane balansiran, odnosno da se ne dogodi da se samo jedan skup parametara pridruži većini klastera.

Na ovaj način se svakom predstavniku klastera dodjeljuje skup parametara za GP koji daje najbolja rješenja za instance u klasteru. Ovim postupkom se neki od skupa parametara ne odabiru pa se dobiva skup za klasifikaciju u manje od 30 klasa. U slučaju RCPSP-a sa 30, 60, 90 i 120 aktivnosti, rezultirajući skup za učenje (C) koristi samo 19 od definiranih 30 skupova parametara. Slika 7.1 daje prikaz broja instanci u pojedinoj klasi u rezultirajućem skupu za učenje (C). Tablica 7.2 prikazuje koja oznaka klastera na slici pripada kojem skupu parametara iz tablice 7.1.

Nakon što se dobije skup podataka za učenje klasifikatora, potrebno je uporabom nekog od postupaka klasifikacije naučiti klasifikator koji će zatim svakoj instanci problema iz skupa za testiranje dodijeliti odgovarajući skup parametara. Za klasifikaciju se koristila biblioteka scikit-learn [179] u programskom jeziku Python. Prije same klasifikacije, skup problema iz skupa za učenje C se dijeli na skup za učenje klasifikatora i skup za testiranje i to tako da je na slučajan način odabrano 20% podataka iz skupa za učenje C i oni su korišteni kao skup

Tablica 7.2: Oznake klasa u skupu za učenje i odgovarajući nazivi parametara

oznaka klase	naziv parametra	oznaka klase	naziv parametra
1	parametri_22	11	parametri_5
2	parametri_4	12	parametri_23
3	parametri_27	13	parametri_14
4	parametri_19	14	parametri_25
5	parametri_18	15	parametri_30
6	parametri_8	16	parametri_13
7	parametri_9	17	parametri_28
8	parametri_12	18	parametri_15
9	parametri_17	19	parametri_10
10	parametri_26		

za testiranje klasifikatora, dok je preostalih 80% podataka korišteno za učenje klasifikatora. Isto tako, nakon razdvajanja podataka na skup za treniranje i skup za testiranje, korišten je `MinMaxScaler` koji podatke iz skupa za učenje klasifikatora skalira na vrijednosti u rasponu od 0 do 1 kako bi obje promatrane značajke jednako doprinosile klasifikaciji. Zatim se podaci iz skupa za učenje klasificiraju korištenjem *k* najbližih susjeda (engl. *k nearest neighbours - knn*) [180], pri čemu se za *k* koristi 3, 5 i 7 susjeda, Bayesovim klasifikatorima (komplementarni Bayesov klasifikator, multinomni Bayesov klasifikator) [181]; logističkom regresijom [182]; strojem potpornih vektora (engl. *support vector machine - SVM*) sa Gaussovim jezgrama [183]; klasifikatorom temeljenim na stablima odluke [184]; te višeslojnim perceptronom s 3, 5 i 7 skrivenih slojeva [185]. Točnosti naučenih klasifikatora se provjeravaju na skupu za testiranje.

Od svih korištenih klasifikatora, najbolji rezultat na skupu za testiranje daju klasifikator temeljem na stablima odluke i komplementarni Bayesov klasifikator. Postignuta točnost na skupu za testiranje u oba slučaja iznosi 0.27. Iako točnost nije velika, treba imati na umu da se skup za učenje sastoji samo od 44 podatka koji se klasificiraju u 19 klastera. Dobivena točnost je svakako bolja nego da se na slučajan način svakom od podataka pridružio neki od klastera.

Nakon što su klasifikatori naučeni na skupu za učenje klasifikatora, svaka od instanci iz skupa za testiranje s početka, se korištenjem klasifikatora raspoređuje u neku od klasa. Na taj način su se na kraju komplementarnim Bayesovim klasifikatorom instance iz skupa za testiranje rasporedile u samo 3 od 19 klasa, dok su se klasifikatorom temeljenim na stablima odluke instance iz skupa za testiranje rasporedile u 9 od ukupno 19 klasa. Tablica 7.3 prikazuje broj instanci iz skupa za testiranje u svakoj od 3 klase dobivene komplementarnim Bayesovim klasifikatorom, dok tablica 7.4 prikazuje broj instanci iz skupa za testiranje u svakoj od 9 klasa

dobivenih klasifikatorom temeljenim na stablima odluke.

Tablica 7.3: Broj instanci iz skupa za testiranje u klasama na temelju komplementarnog Bayesovog klasifikatora

klasa	broj instanci
1	121
2	276
3	215

Tablica 7.4: Broj instanci iz skupa za testiranje u klasama na temelju klasifikacije stablima odluke

klasa	broj instanci	klasa	broj instanci
1	3	8	1
3	2	9	1
4	3	11	1
5	8	14	2
7	591		

Oznake klasa se preslikavaju u skupove parametara koje je potrebno primijeniti na instance odgovarajuće klase, kako je prikazano u tablici 7.2. Iz navedenih tablica se vidi da u prvom slučaju u sve tri dobivene klase završi podjednak broj instanci, dok je u drugom slučaju klasifikacija neujednačena i gotovo svi problemi se nalaze u klasi 7.

Nakon što su određene klase i pripadnost svih instanci klasama, u svim klasama je pokrenut GP s odgovarajućim parametrima kako bi se dobili rezultati funkcije cilja za cijelu klasu. Dobivene rezultate treba usporediti s nekim vrijednostima i u tu svrhu se koristi skup parametara koji daje najbolju ukupnu vrijednost funkcije cilja na cijelom skupu za učenje (A) prije grupiranja. Taj skup parametara se dobiva tako da se svaki od 30 početnih skupova parametara iskoristi u 10 pokretanja GP-a na skupu za učenje A i kao vrijednost funkcije cilja se uzima prosječna vrijednost tih 10 pokretanja. Skup parametara koji je ovim postupkom dao najbolje rezultate je parametri_25.

U svakoj grupi instanci se zatim 30 puta pokrenuo GP s parametrima određenima za klasu kojoj pripadaju i s parametrima koji su najbolji za cijeli skup podataka. Usporedbu minimuma, donjeg i gornjeg kvartila, medijana, prosjeka, maksimuma i standardne devijacije dobivenih vrijednosti funkcije cilja za svaku od 3 klase dobivene komplementarnim Bayesovim klasifikatorom daje tablica 7.5, dok istu tu usporedbu za svaku od 9 klasa dobivenih klasifikacijom stablom odluke daje tablica 7.6. Slike 7.2 i 7.3 daju prikaz usporedbe kutijastih dijagrama vrijednosti funkcije cilja dobivenih korištenjem parametara za odgovarajuću klasu i vrijednosti

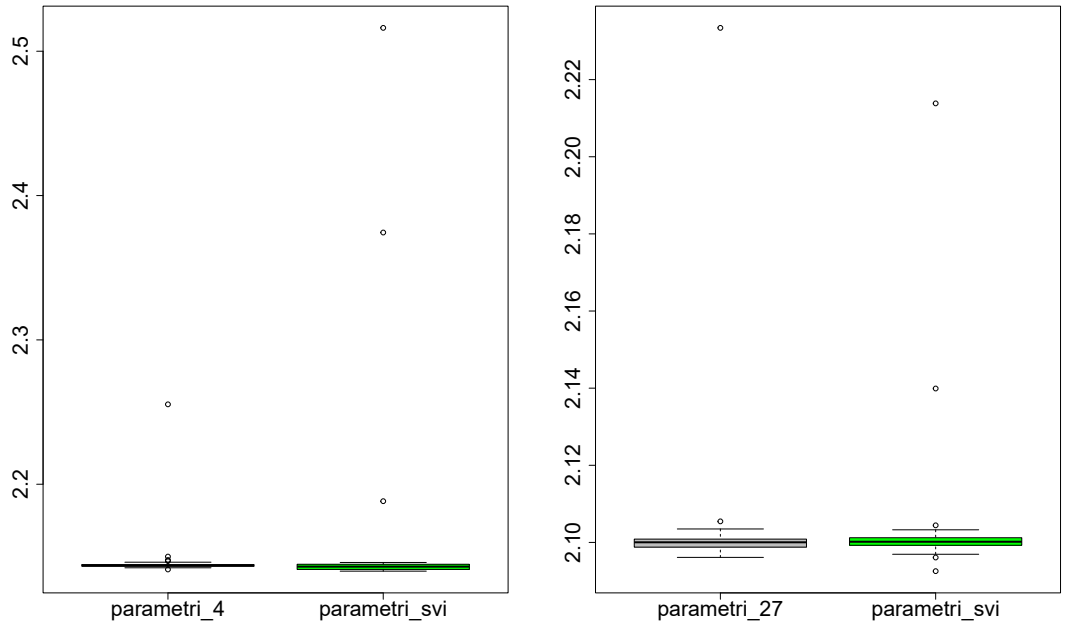
funkcije cilja dobivenih korištenjem parametara odabranih za cijeli skup instanci za prethodno navedene klasifikatore. Slika 7.2 daje prikaz odgovarajućih dijagrama samo za klasu 2 i klasu 3 dobivene komplementarnim Bayesovim klasifikatorom s obzirom da su u klasi 1 sve vrijednosti jednake. Dodatno, za ovaj klasifikator prikazani su i kutijasti dijagrami bez stršećih vrijednosti kako bi se vidjele razlike u medijanima. Na slici 7.3 se vidi da razlika u medijanima gotovo i nema, samo u pojedinim slučajevima se pojavljuju stršeće vrijednosti.

Tablica 7.5: Usporedba vrijednosti funkcije cilja dobivenih GP-om s parametrima odabranim za pojedinu klasu (temeljem klasifikacije komplementarnim Bayesovim klasifikatorom) i parametrima određenim za sve instance problema

	klasa 1		klasa 2		klasa 3	
	parametri_22	parametri_svi	parametri_4	parametri_svi	parametri_27	parametri_svi
min	1.7165	1.7165	2.1410	2.1398	2.0961	2.0926
1. kv.	1.7165	1.7165	2.1432	2.1411	2.0988	2.0993
medijan	1.7165	1.7165	2.1437	2.1429	2.1000	2.1002
prosjek	1.7165	1.7165	2.1477	2.1642	2.1044	2.1050
3. kv.	1.7165	1.7165	2.1443	2.1444	2.1008	2.1012
max	1.7165	1.7165	2.2554	2.5162	2.2334	2.2138
stdev	0.0000	0.0000	0.0204	0.0791	0.0244	0.0219

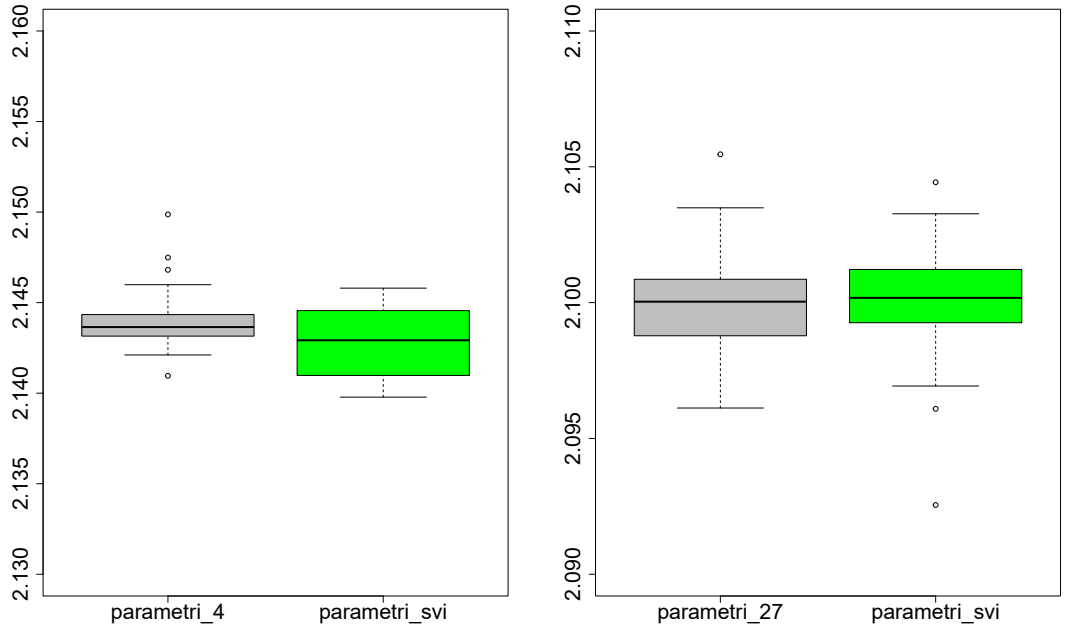
Tablica 7.6: Usporedba vrijednosti funkcije cilja dobivenih GP-om s parametrima odabranim za pojedinu klasu (temeljem klasifikacije stablima odluke) i parametrima određenim za sve instance problema

	klasa 1		klasa 3		klasa 4	
	parametri_22	parametri_svi	parametri_27	parametri_svi	parametri_19	parametri_svi
min	1.6708	1.6708	1.8504	1.8504	1.9955	1.9955
1. kv.	1.6708	1.6708	1.8504	1.8504	1.9955	1.9955
medijan	1.6708	1.6708	1.8504	1.8504	1.9955	1.9955
prosjek	1.6708	1.6807	1.8637	1.8546	2.0160	1.9955
3. kv.	1.6708	1.6708	1.8504	1.8504	1.9955	1.9955
max	1.6708	1.8488	2.1222	1.9763	2.4012	1.9955
stdev	0.0000	0.0385	0.0540	0.0230	0.0762	0.0000
	klasa 5		klasa 7		klasa 8	
	parametri_18	parametri_svi	parametri_9	parametri_svi	parametri_12	parametri_svi
min	2.0461	2.0461	2.0484	2.0459	1.6241	1.6241
1. kv.	2.0508	2.0473	2.0503	2.0488	1.6241	1.6241
medijan	2.0508	2.0508	2.0509	2.0494	1.6241	1.6241
prosjek	2.0542	2.0548	2.0615	2.0503	1.6389	1.6241
3. kv.	2.0508	2.0508	2.0516	2.0506	1.6241	1.6241
max	2.1051	2.2061	2.1603	2.0627	2.0699	1.6241
stdev	0.0118	0.0287	0.0306	0.0033	0.0814	0.0000
	klasa 9		klasa 11		klasa 14	
	parametri_17	parametri_svi	parametri_5	parametri_svi	parametri_25	parametri_svi
min	2.1202	2.1202	1.6744	1.6744	1.5604	1.5604
1. kv.	2.1202	2.1202	1.6744	1.6744	1.5604	1.5604
medijan	2.1202	2.1202	1.6744	1.6744	1.5604	1.5604
prosjek	2.1296	2.1261	1.6744	1.6744	1.5638	1.5656
3. kv.	2.1202	2.1202	1.6744	1.6744	1.5604	1.5604
max	2.4029	2.2969	1.6744	1.6744	1.6155	1.7182
stdev	0.0516	0.0323	0.0000	0.0000	0.0131	0.0288



(a) klasa 2

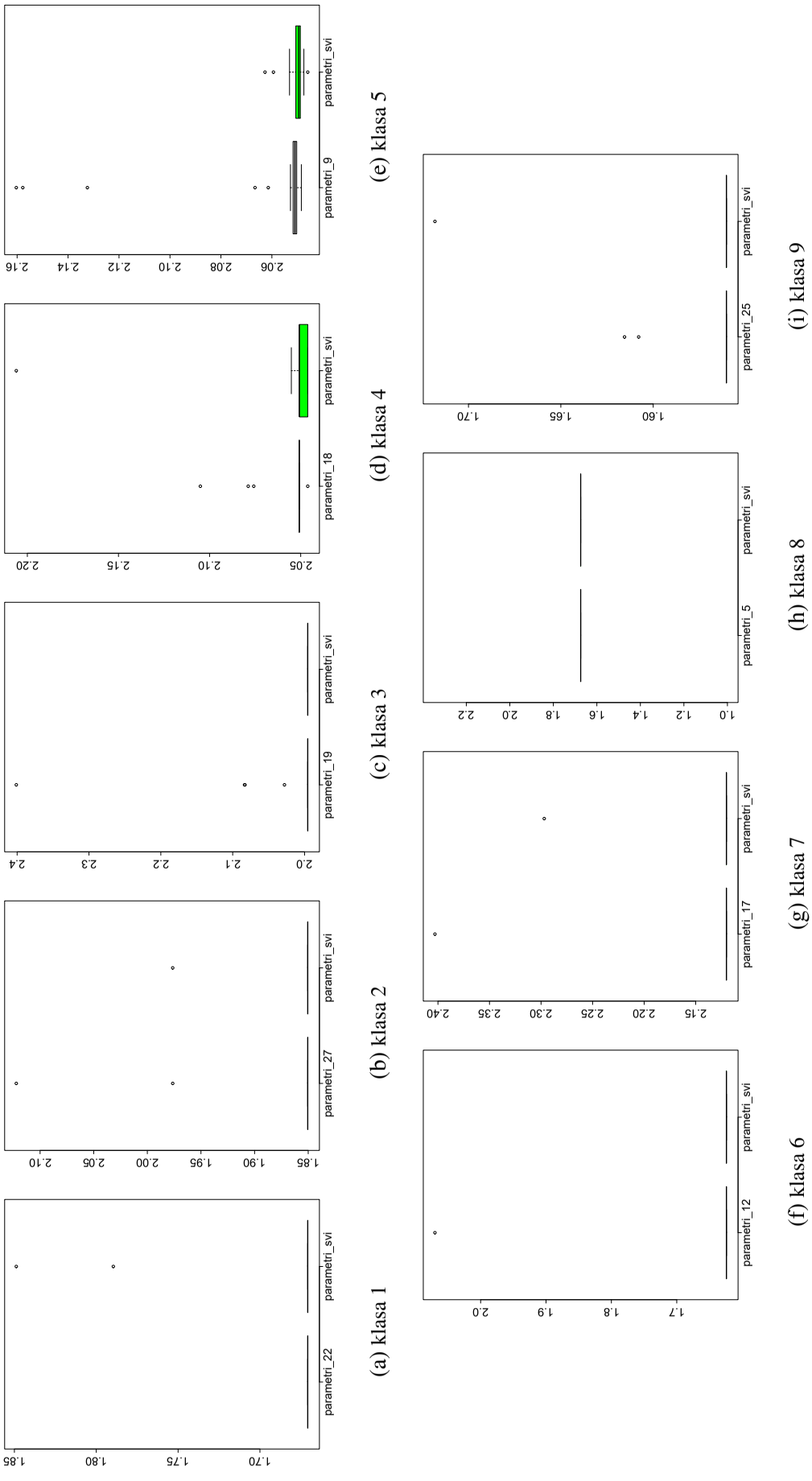
(b) klasa 3



(c) klasa 2 - bez outliera

(d) klasa 3 - bez outliera

Slika 7.2: Prikaz kutijastih dijagrama dobivenih na temelju klasifikacije komplementarnim Bayesovim klasifikatorom



(a) klasa 1 (b) klasa 2 (c) klasa 3 (d) klasa 4 (e) klasa 5

(f) klasa 6 (g) klasa 7 (h) klasa 8 (i) klasa 9

Slika 7.3: Prikaz kutijastih dijagrama dobivenih na temelju klasifikacije stablom odluke

Iz tablica 7.5 i 7.6 se vidi da u slučaju klasifikacije komplementarnim Bayesovim klasifikatorom, medijan postignutih vrijednosti funkcije cilja je u jednom klasteru bolji za GP sa parametrima određenim za odgovarajuću klasu, u jednom klasteru je bolji GP sa parametrima određenim za podatke bez grupiranja, dok je u jednom klasteru vrijednost medijana funkcije cilja jednaka. Kod klasifikacije stabilma odluke medijani su jednaki u svim klasterima osim u onom s najviše instanci. U tom klasteru GP s parametrima određenim za sve instance problema ipak daje nešto bolji medijan. Razlog tomu može ležati u načinu kreiranja skupa za učenje klasifikatora jer je moguće kako je, zbog redoslijeda kojim se popunjava *plist*, skup parametara koji je naknadno određen kao najbolji za cijeli skup podataka jednostavno preskočen. Dodatno, razlog ovakvim rezultatima može biti i relativno mali broj primjera za klasifikaciju pa klasifikatori ne mogu naučiti najbolje obrasce za podatke iz skupa za testiranje. Isto tako, možda bi imalo smisla isprobati neki drugi način grupiranja podataka na samom početku stvaranja grupa za skup za učenje.

Prednost ovog postupka je to što, kada se jednom učini klasifikacija, za probleme koji dosad nisu viđeni se vrlo brzo može odabrati skup parametara koje treba upotrijebiti za rad GP-a pa se tu skraćuje ukupno vrijeme potrebno za rješavanje problema.

Konačno, kao područje za daljnje istraživanje može se promatrati kako unaprijed odrediti raspon za svaki od parametara potrebnih za rad GP-a iz kojeg će skup za učenje birati parametre. Imalo bi smisla iskoristiti znanja stručnjaka koji iz iskustva znaju koji parametri obično daju bolje rezultate za probleme raspoređivanja pa umjesto slučajno generiranih parametara koristiti one predložene od strane stručnjaka.

Poglavlje 8

Zaključak

Problemi raspoređivanja su problemi koji su rasprostranjeni u stvarnom svijetu i čije se primjene mogu pronaći u raznim područjima. S obzirom da su ti problemi NP teški, razvijaju se razni načini za njihovo rješavanje. Uz egzaktne metode koje pretražuju cijeli prostor stanja i jamče pronalazak optimalnog rješenja, koriste se i heuristike i hiperheuristike u slučajevima kada je prostor pretraživanja velik i nemoguće je pretražiti cijeli prostor stanja. Jedna od hiperheuristika koja se uspješno primjenjuje za rješavanje problema raspoređivanja je i GP. GP se većinom upotrebljava za razvoj prioritetnih pravila koja se zatim koriste za stvaranje rasporeda aktivnosti u samom problemu.

Snaga hiperheuristika poput GP-a leži u tome da se parametri GP-a mogu prilagoditi tako da izvršavanje algoritma daje dobre rezultate na promatranim problemima. Problem koji se javlja je kako odrediti dobre parametre. Stručnjaci u pojedinim područjima iz iskustva mogu znati koji parametri otprilike dobro funkcioniraju za probleme iz tog područja. Ukoliko stručnjaci nisu dostupni, ručnim pretraživanjem prostora parametara može se doći do konfiguracije parametara koja će dobro raditi na danom problemu, ali je taj postupak dosta dugotrajan. Iz tog razloga se javlja potreba za mehanizmima automatiziranog odabira parametara koja će ubrzati cijeli proces početnog postavljanja algoritma i ukloniti potrebu za prethodnim poznavanjem područja ukoliko stručno iskustvo nije dostupno.

Jedan od načina kako se može dobiti uvid u samu strukturu problema je korištenjem analize krajolika dobrote. Krajolik dobrote se odnosi na mapiranje genoma populacije jedinki na njihove dobrote te vizualizaciju tog mapiranja. Kroz razne značajke krajolika dobrote moguće je dobiti uvid u strukturu problema i razlike između pojedinih instanci problema. Informacije o razlikama mogu se uporabiti kako bi se heterogeni skup instanci grupirao u slične klastere i tada se za svaki od tih klastera može odrediti prilagođeni skup parametara za GP kako bi se u konačnici dobila što bolja rješenja.

8.1 Glavni zaključci i doprinosi

S obzirom da automatizirani odabir parametara daje dobre rezultate u raznim područjima primjene, u ovoj disertaciji je iskorišten automatizirani odabir parametara za GP na problemima raspoređivanja. Pri tome je prvo korištena analiza krajolika dobrote kako bi se proučile razlike između instanci pojedinog problema i kako bi se na temelju dobivenih značajki dobrote instance problema mogle grupirati u klastere sličnih instanci. Zatim se za svaki od dobivenih klastera korištenjem automatiziranog odabira parametara odredilo koji parametri daju najbolje rezultate i dobiveni rezultati su uspoređeni s parametrima koji su određeni ručno za cijeli skup instanci. Pregled zaključaka za svaki od doprinosa će biti dan u nastavku.

8.1.1 Skup operatora za sintaktička stabla

U ovoj disertaciji se za prikaz rješenja u GP-u koriste stabla. Prvi korak koji treba učiniti kod analize krajolika dobrote je izračunati razne mjere, odnosno značajke koje će dati uvid u strukturu problema. Za neke od mjera potrebno je odrediti kako doći do susjeda promatrane jedinke ili koliko su međusobno udaljene dvije proizvoljne jedinke. Problem koji se javlja je što ne postoji jednoznačno određen način kako doći do susjeda jedinke ili kako izračunati udaljenost danih stabala. Iako postoje neke mjere udaljenosti između stabala, one ne uzimaju u obzir da, prilikom korištenja GP-a, stabla mogu imati funkcijske čvorove i čvorove koji sadrže značajke pa prilikom računanja udaljenosti treba uvažiti razlike u čvorovima.

Prethodno opisani problemi su riješeni tako da su definirana tri nova operatora za promjene na stablima. Definiran je operator umetanja koji u postojeće stablo umeće čvor poštujući određena pravila za funkcijske čvorove i čvorove koji sadrže značajke; operator zamjene koji zamjenjuje samo čvorove jednakog tipa; te operator brisanja koji može obrisati sve čvorove osim korijenskog, pri tome pazeći na vrstu čvora koji se briše i primjenjujući određena pravila prilikom brisanja. Uvedeni operatori daju mogućnost dobivanja susjedstva promatrane jedinke, odnosno stabla, što je bitno prilikom računanja značajki krajolika dobrote.

Dodatno, kako bi se mogla izračunati udaljenost između bilo koje dvije jednike, definirana je i heuristika koja korištenjem prethodno uvedenih operatora računa udaljenost između bilo koja dva proizvoljna stabla. Dan je i primjer računanja udaljenosti definiranom heuristikom.

Iako su operatori definirani za skup funkcija koji je korišten u eksperimentima u ovoj disertaciji, oni se vrlo jednostavno mogu proširiti i na funkcije koje se koriste u drugim problemima koristeći pravila opisana prilikom definiranja operatora.

8.1.2 Primjena analize krajolika dobrote u svrhu određivanja prikladnih parametara za GP

Problemi na koje se fokusirala ova disertacija su problemi raspoređivanja u okolini jednog stroja, problemi raspoređivanja u okolini nesrodnih strojeva i dvije inačice problema raspoređivanja s ograničenim sredstvima. Navedeno je koje se značajke krajolika dobrote koriste i kako se računaju. Zatim je objašnjeno na koji način je rađeno grupiranje u pojedinom problemu raspoređivanja.

Za svaki od navedenih problema dani su prikazi grupiranja po značajkama krajolika dobrote koje daju najbolje grupiranje. Rezultati su pokazali da za različite okoline treba koristiti različite algoritme grupiranja. Kod problema raspoređivanja u okolini jednog stroja i okolini nesrodnih strojeva k-means algoritam daje dobre rezultate, dok kod obje inačice problema raspoređivanja s ograničenim sredstvima EM algoritam daje dobre rezultate. Također, pokazalo se da u različitim okolinama grupiranja, različite značajke daju najbolje grupiranje.

Nakon određivanja grupa, za svaki od klastera se korištenjem okruženja *irace* odredilo koje vrijednosti za veličinu populacije, vjerojatnost mutacije, maksimalnu dubinu stabla i veličinu turnira treba koristiti u GP-u. Pokazalo se da ovaj postupak daje različite vrijednosti parametara za različite klastere, što znači da je postupak grupiranja na temelju značajki krajolika dobrote uspio razlikovati instance problema i prilagoditi parametre GP-a instancama u pojedinim klasterima. Nakon određivanja parametara, rezultati dobiveni GP-om s parametrima određenim automatizirano u svakom klasteru su uspoređeni s GP-om koji koristi ručno određene parametre za cijeli skup instanci prije grupiranja. Rezultati pokazuju da se korištenjem parametara prilagođenih instancama u pojedinom klasteru ostvaruju bolji rezultati nego korištenjem GP-a s ručno određenim parametrima. Dodatno, što je bolje grupiranje, to su veće razlike u vrijednostima funkcije cilja u korist parametara određenih automatizirano.

U svim prethodno spomenutim eksperimentima grupiranja su rađena na temelju samo jedne značajke krajolika dobrote. Dodatno je uporabljena metoda automatiziranog odabira značajki krajolika dobrote na temelju kojih će se grupirati instance problema. Cilj ove metode je automatizirano pronaći kombinaciju značajki koja će dati dobro grupiranje i vidjeti mogu li se u tako dobivenim klasterima vrijednosti funkcije cilja dobivene GP-om sa parametrima prilagođenim klasteru usporediti sa vrijednostima funkcije cilja dobivenih GP-om sa ručno određenim parametrima. I u ovom slučaju su parametri određeni za GP automatizirano u svakom dobivenom klasteru različiti, što znači da su u dobivenim klasterima uistinu smještene instance koje se razlikuju. Usporedbom vrijednosti funkcije cilja u slučaju parametara prilagođenih pojedinom klasteru i parametara određenih ručno za cijeli skup, bolja vrijednost za parametre prilagođene klasteru dobija se u većini klastera. U klasterima gdje je vrijednost lošija za parametre prilagođene klasteru, pogoršanje u vrijednostima funkcije cilja je znatno manje nego poboljšanje u

vrijednosti funkcije cilja pa se sveukupno postupkom grupiranja i prilagođavanja parametara klasterima dobivaju bolji rezultati nego korištenjem ručno određenih parametara. Dodatno poboljšanje bi se moglo postići korištenjem nekog drugog oblika automatiziranog odabira značajki ili korištenjem nekog drugog algoritma grupiranja.

8.1.3 Odabir GP parametara klasifikacijom

U prethodno opisanim eksperimentima, parametri za GP su se određivali automatizirano korištenjem okruženja *irace*. U ovom dijelu se pokušalo vidjeti mogu li se parametri za pojedine instance problema odabrati na temelju značajki krajolika dobrote iz nekog skupa prethodno definiranih parametara. Definirana je shema koja za probleme raspoređivanja s ograničenim sredstvima od cijelog skupa instanci stvara skup za učenje klasifikatora. Zatim je na tom skupu naučeno nekoliko klasifikatora i na temelju onih koji daju najbolje rezultate su dotad neviđene instance problema raspoređene u odgovarajuće klase. U svakoj klasi je na instance primijenjen GP s odgovarajućim parametrima za tu klasu i rezultati su uspoređeni s rezultatima dobivenim GP-om sa parametrima koji su se pokazali najboljima na cijelom skupu za učenje.

Rezultati su pokazali da se ovim postupkom dobivaju vrijednosti funkcije cilja približno jednake vrijednostima funkcije cilja dobivenima uporabom GP-a s parametrima koji su najbolji na cijelom skupu za učenje. Prednost ovog postupka je što se raspoređivanjem instanci problema u klase dobivaju manji skupovi problema pa sam GP puno brže dolazi do rješenja nego prilikom uporabe na cijelom skupu za učenje. U provedenim su eksperimentima vrijednosti za skup GP parametara koji se može dodijeliti nekoj od klasa odabrane na slučajan način. Određeno poboljšanje bi se možda moglo postići ukoliko su skupovi parametara iz kojih se biraju parametri za pojedinu klasu pomnije odabrani kako bi bili što više prilagođeni promatranom problemu.

8.2 Buduća istraživanja

Eksperimenti provedeni u ovoj disertaciji samo su početna točka razmatranja kako se analiza krajolika dobrote može uporabiti za unaprijeđenje rješavanja problema raspoređivanja. Ostaju još mnoga otvorena pitanja u smjeru kojih mogu ići daljnja istraživanja.

Prva nadogradnja koja se može učiniti je istraživanje većeg broja značajki krajolika dobrote ili stvaranje novih značajki koje bi bile prilagođene prikazu jedinki stablima. Prilagođene značajke krajolika dobrote bi mogle dati još bolji uvid u strukturu problema i omogućiti bolje razlikovanje instanci problema.

U eksperimentima provedenim u okolini jednog stroja i okolini nesrodnih strojeva, promatran je samo jedan kriterij optimizacije i to ukupno težinsko zaostajanje, dok je u problemima raspoređivanja s ograničenim sredstvima promatran samo prilagođeni kriterij temeljen na vre-

menu završetka projekta. U daljnjim istraživanjima bi se moglo proučiti postoje li neke razlike u krajolicima dobrote promatranih problema ukoliko se koriste drugi kriteriji optimizacije te hoće li se najbolja grupiranja u tom slučaju temeljiti na značajkama krajolika dobrote različitima od značajki koje su dale najbolje grupiranje s ovdje promatranim kriterijima.

Nadalje, u provedenim eksperimentima su se za GP korištenjem okruženja *irace* određivali samo veličina populacije, vjerojatnost mutacije, maksimalna dubina stabla i veličina turnira. U daljnjim istraživanjima bi se moglo proučiti kakvi se rezultati postižu ukoliko dozvolimo i odabir operatora mutacije i križanja. Postoji mogućnost da će automatizirani odabir parametara pronaći dobre kombinacije tih operatora koje daju bolje rezultate.

Također, moguće je ispitati kakvi rezultati se postižu ukoliko se grupiranje instanci u klastere provodi na temelju više značajki krajolika dobrote. Potrebno je pronaći način kako odrediti dobru kombinaciju značajki. Algoritam odabira skupa značajki korišten u ovom radu je pohlepna metoda pa bi bilo dobro istražiti postoje li neke bolje metode koje će iskoristiti više značajki i na taj način postići bolje grupiranje.

Na poslijetku, u eksperimentima provedenim u ovoj disertaciji koristi se EM algoritam koji daje i vjerojatnosti da pojedina instanca problema pripada pojedinom klasteru. Za instance iz testnog skupa bi se na temelju značajki krajolika dobrote mogla odrediti vjerojatnost da promatrana instanca pripada nekom od dobivenih klastera. Na taj način bi se za dobivene instance moglo, umjesto fiksno određenih parametara za pojedini klaster, koristiti linearnu kombinaciju parametara iz pojedinog klastera gdje bi težine odgovarale vjerojatnostima pripadnosti određenom klasteru.

Literatura

[1] Pinedo, M. L., Scheduling - Theory, Algorithms, and Systems. Springer-Verlag New York, 2012.

[2] Ribeiro, C. C., "Sports scheduling: Problems and applications", International Transactions in Operational Research, Vol. 19, No. 1-2, 2012, str. 201-226.

[3] Petrovic, S., Burke, E., "University timetabling", in Handbook of Scheduling, 2004.

[4] Cheang, B., Li, H., Lim, A., Rodrigues, B., "Nurse rostering problems—a bibliographic survey", European Journal of Operational Research, Vol. 151, No. 3, 2003, str. 447 - 460.

[5] Grosche, T., Airline Scheduling Process. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, str. 7–46.

[6] Bai, D., Xue, H., Wang, L., Wu, C.-C., Lin, W.-C., Abdulkadir, D. H., "Effective algorithms for single-machine learning-effect scheduling to minimize completion-time-based criteria with release dates", Expert Systems with Applications, Vol. 156, 2020, str. 113445.

[7] de Weerdt, M., Baart, R., He, L., "Single-machine scheduling with release times, deadlines, setup times, and rejection", European Journal of Operational Research, 2020.

[8] Fanjul-Peyro, L., "Models and an exact method for the unrelated parallel machine scheduling problem with setups and resources", Expert Systems with Applications: X, Vol. 5, 2020, str. 100022.

[9] Saberi-Aliabad, H., Reisi-Nafchi, M., Moslehi, G., "Energy-efficient scheduling in an unrelated parallel-machine environment under time-of-use electricity tariffs", Journal of Cleaner Production, Vol. 249, 2020, str. 119393.

[10] Prata, B., Abreu, L. R., Lima, J. Y. F., "Heuristic methods for the single-machine scheduling problem with periodical resource constraints", TOP, 2020, str. 1-23.

[11] Türkyılmaz, A., Senvar, Ö., İrem Ünal, Bulkan, S., "A research survey: heuristic approaches for solving multi objective flexible job shop problems", Journal of Intelligent Manufacturing, 2020, str. 1-35.

[12] Tsai, C., Huang, W., Chiang, M., Chiang, M., Yang, C., "A hyper-heuristic scheduling algorithm for cloud", IEEE Transactions on Cloud Computing, Vol. 2, No. 2, 2014, str. 236-250.

[13] Koulinas, G., Kotsikas, L., Anagnostopoulos, K., "A particle swarm optimization based hyper-heuristic algorithm for the classic resource constrained project scheduling problem", Information Sciences, Vol. 277, 2014, str. 680 - 693.

[14] Jakobović, D., "Raspoređivanje zasnovano na prilagodljivim pravilima", Doktorski rad, Fakultet elektrotehnike i računarstva, Zagreb, 2006.

[15] Đurasević, M., Jakobović, D., Knežević, K., "Adaptive scheduling on unrelated machines with genetic programming", Applied Soft Computing, Vol. 48, 2016, str. 419 - 430.

[16] Đumić, M., Šišejković, D., Čorić, R., Jakobović, D., "Evolving priority rules for resource constrained project scheduling problem with genetic programming", Future Generation Computer Systems, Vol. 86, 2018, str. 211 - 221.

[17] Poli, R., Langdon, W. B., McPhee, N. F., A field guide to genetic programming. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008, (With contributions by J. R. Koza).

[18] de Lima, E. B., Pappa, G. L., de Almeida, J. M., Gonçalves, M. A., Meira, W., "Tuning genetic programming parameters with factorial designs", in IEEE Congress on Evolutionary Computation, 2010, str. 1-8.

[19] Castelli, M., Manzoni, L., Vanneschi, L., Silva, S., Popovic, A., "Self-tuning geometric semantic genetic programming", Genetic Programming and Evolvable Machines, Vol. 17, 2015, str. 55-74.

[20] Semenkin, E., Semenkina, M., "Self-configuring genetic programming algorithm with modified uniform crossover", in 2012 IEEE Congress on Evolutionary Computation, 2012, str. 1-6.

[21] Wright, S., "The Roles of Mutation, Inbreeding, Crossbreeding and Selection in Evolution", in Proceedings of the Sixth International Congress on Genetics, 1932, str. 365-366.

[22] Kinnear, K. E., “Fitness landscapes and difficulty in genetic programming”, in Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence, 1994, str. 142-147 vol.1.

[23] Pitzer, E., Affenzeller, M., A Comprehensive Survey on Fitness Landscape Analysis, 10 2011, Vol. 378, str. 161-191.

[24] Beham, A., Pitzer, E., Affenzeller, M., “Fitness landscape based parameter estimation for robust taboo search”, in Computer Aided Systems Theory - EUROCAST 2013, Moreno-Díaz, R., Pichler, F., Quesada-Arencibia, A., (ur.). Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, str. 292–299.

[25] Picek, S., Jakobovic, D., “From fitness landscape to crossover operator choice”, in GECCO ’14, 2014.

[26] Liefooghe, A., Derbel, B., Verel, S., Aguirre, H., Tanaka, K., “Towards landscape-aware automatic algorithm configuration: Preliminary experiments on neutral and rugged landscapes”, in Evolutionary Computation in Combinatorial Optimization, Hu, B., López-Ibáñez, M., (ur.). Cham: Springer International Publishing, 2017, str. 215–232.

[27] Gil-Gala, F. J., Mencía, C., Sierra, M. R., Varela, R., “Evolving priority rules for on-line scheduling of jobs on a single machine with variable capacity over time”, Applied Soft Computing, Vol. 85, 2019, str. 105782.

[28] Dimopoulos, C., Zalzala, A., “Investigating the use of genetic programming for a classic one-machine scheduling problem”, Advances in Engineering Software, Vol. 32, No. 6, 2001, str. 489 - 498.

[29] Đurasević, M., Jakobović, D., “Comparison of schedule generation schemes for designing dispatching rules with genetic programming in the unrelated machines environment”, Applied Soft Computing, Vol. 96, 2020, str. 106637.

[30] Lin, J., Zhu, L., Gao, K., “A genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem”, Expert Systems with Applications, Vol. 140, 2020, str. 112915.

[31] Morikawa, K., Nagasawa, K., Takahashi, K., “Job shop scheduling by branch and bound using genetic programming”, Procedia Manufacturing, Vol. 39, 2019, str. 1112 - 1118, 25th International Conference on Production Research Manufacturing Innovation: Cyber Physical Manufacturing August 9-14, 2019 | Chicago, Illinois (USA).

[32] Ochoa, G., Rodríguez, J. A. V., Petrovic, S., Burke, E., "Dispatching rules for production scheduling: A hyper-heuristic landscape analysis", 2009 IEEE Congress on Evolutionary Computation, 2009, str. 1873-1880.

[33] Ochoa, G., Qu, R., Burke, E., "Analyzing the landscape of a graph based hyper-heuristic for timetabling problems", in GECCO '09, 2009.

[34] Bille, P., "A survey on tree edit distance and related problems", Theor. Comput. Sci., Vol. 337, 2005, str. 217-239.

[35] Gustafson, S., Vanneschi, L., "Crossover-based tree distance in genetic programming", IEEE Transactions on Evolutionary Computation, Vol. 12, No. 4, 2008, str. 506-524.

[36] Pawlik, M., Augsten, N., "Tree edit distance: Robust and memory-efficient", Inf. Syst., Vol. 56, 2016, str. 157-173.

[37] Smith-Miles, K., Lopes, L., "Measuring instance difficulty for combinatorial optimization problems", Computers & Operations Research, Vol. 39, No. 5, 2012, str. 875 - 889.

[38] Merz, P., Freisleben, B., "Memetic algorithms and the fitness landscape of the graph bi-partitioning problem", in Parallel Problem Solving from Nature — PPSN V, Eiben, A. E., Bäck, T., Schoenauer, M., Schwefel, H.-P., (ur.). Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, str. 765–774.

[39] Himmelblau, D. M., Applied Nonlinear Programming. McGraw-Hill, 1972.

[40] Rastrigin, L., Systems of Extreme Control. Nauka, Moscow, 1974.

[41] Katada, Y., Estimating the Degree of Neutrality and Ruggedness of Fitness Landscapes. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, str. 207–231.

[42] Richter, H., Fitness Landscapes: From Evolutionary Biology to Evolutionary Computation. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, str. 3–31.

[43] Kauffman, S., Levin, S., "Towards a general theory of adaptive walks on rugged landscapes", Journal of Theoretical Biology, Vol. 128, No. 1, 1987, str. 11 - 45.

[44] Kauffman, S. A., Weinberger, E. D., "The nk model of rugged fitness landscapes and its application to maturation of the immune response", Journal of Theoretical Biology, Vol. 141, No. 2, 1989, str. 211 - 245.

[45] Weinberger, E., "Correlated and uncorrelated fitness landscapes and how to tell the difference", Biological Cybernetics, Vol. 63, 09 1990, str. 325-336.

[46] Weinberger, E. D., “Local properties of kauffman’s n-k model: A tunably rugged energy landscape”, Phys. Rev. A, Vol. 44, Nov 1991, str. 6399–6413.

[47] Derrida, B., Peliti, L., “Evolution in a flat fitness landscape”, Bulletin of Mathematical Biology, Vol. 53, No. 3, 1991, str. 355 - 382.

[48] Forrest, S., Holland, J., “The royal road for genetic algorithms: Fitness landscapes and ga performance”, in Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life, 1991, str. 245–254.

[49] Spears, W. M., De Jong, K. A., Bäck, T., Fogel, D. B., de Garis, H., “An overview of evolutionary computation”, in Machine Learning: ECML-93, Brazdil, P. B., (ur.). Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, str. 442–459.

[50] Horn, J., Goldberg, D. E., “Genetic algorithm difficulty and the modality of fitness landscapes”, ser. Foundations of Genetic Algorithms, WHITLEY, L. D., VOSE, M. D., (ur.). Elsevier, 1995, Vol. 3, str. 243 - 269.

[51] Pitzer, E., Affenzeller, M., Beham, A., “A closer look down the basins of attraction”, in 2010 UK Workshop on Computational Intelligence (UKCI), 2010, str. 1-6.

[52] Malan, K. M., Engelbrecht, A. P., “Ruggedness, funnels and gradients in fitness landscapes and the effect on pso performance”, in 2013 IEEE Congress on Evolutionary Computation. IEEE, 2013, str. 963–970.

[53] Barnett, L., “Tangled webs – evolutionary dynamics on fitness landscapes with neutrality”, 1997.

[54] Barnett, L., “Ruggedness and neutrality—the nkp family of fitness landscapes”, in Proceedings of the Sixth International Conference on Artificial Life, ser. ALIFE. Cambridge, MA, USA: MIT Press, 1998, str. 18–27.

[55] Reidys, C. M., Stadler, P. F., “Neutrality in fitness landscapes”, Applied Mathematics and Computation, Vol. 117, No. 2, 2001, str. 321 - 350.

[56] Bogon, T., Poursanidis, G., Lattner, A. D., Timm, I. J., “Setting up particle swarm optimization by decision tree learning out of function features”, in Agents and Artificial Intelligence, Filipe, J., Fred, A., (ur.). Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, str. 72–85.

[57] Vassilev, V. K., Fogarty, T. C., Miller, J. F., “Information characteristics and the structure of landscapes”, Evolutionary Computation, Vol. 8, 2000, str. 31-60.

[58] Morgan, R., Gallagher, M., “Analysing and characterising optimization problems using length scale”, Soft Computing, Vol. 21, 2017, str. 1735 - 1752.

[59] Morgan, R., Gallagher, M., “Length scale for characterising continuous optimization problems”, in Parallel Problem Solving from Nature - PPSN XII, Coello, C. A. C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M., (ur.). Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, str. 407–416.

[60] Prügel-Bennett, A., Tayarani-Najaran, M.-H., “Maximum satisfiability: Anatomy of the fitness landscape for a hard combinatorial optimization problem”, IEEE Transactions on Evolutionary Computation, Vol. 16, 2012, str. 319-338.

[61] Ochoa, G., Chicano, F., Tomassini, M., “Global landscape structure and the random max-sat phase transition”, in PPSN, 2020.

[62] Chicano, F., Luque, G., Alba, E., “Elementary landscape decomposition of the quadratic assignment problem”, in GECCO ’10, 2010.

[63] Ochoa, G., Herrmann, S., “Perturbation strength and the global structure of qap fitness landscapes”, in PPSN, 2018.

[64] Tayarani-Najaran, M.-H., Prügel-Bennett, A., “An analysis of the fitness landscape of travelling salesman problem”, Evolutionary Computation, Vol. 24, 2016, str. 347-384.

[65] Ochoa, G., Veerapen, N., “Mapping the global structure of tsp fitness landscapes”, Journal of Heuristics, Vol. 24, 2018, str. 265-294.

[66] Bouziri, H., Mellouli, K., Talbi, E., “The k-coloring fitness landscape”, Journal of Combinatorial Optimization, Vol. 21, 2011, str. 306-329.

[67] Tayarani-N., M.-H., Prügel-Bennett, A., “Anatomy of the fitness landscape for dense graph-colouring problem”, Swarm and Evolutionary Computation, Vol. 22, 2015, str. 47 - 65.

[68] Tavares, J., Pereira, F., Costa, E., “The role of representation on the multidimensional knapsack problem by means of fitness landscape analysis”, 2006 IEEE International Conference on Evolutionary Computation, 2006, str. 2307-2314.

[69] Tavares, J., Pereira, F., Costa, E., “Multidimensional knapsack problem: A fitness landscape analysis”, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), Vol. 38, 2008, str. 604-616.

[70] Pitzer, E., Vonolfen, S., Beham, A., Affenzeller, M., Boļšakovs, V., Merkurjeva, G., "Structural analysis of vehicle routing problems using general fitness landscape analysis and problem specific measures", 2012.

[71] Kovács, L., Agárdi, A., Bányai, T., "Fitness landscape analysis and edge weighting-based optimization of vehicle routing problems", Processes, Vol. 8, 2020, str. 1363.

[72] Czogalla, J., Fink, A., "Fitness landscape analysis for the no-wait flow-shop scheduling problem", Journal of Heuristics, Vol. 18, 2012, str. 25-51.

[73] Baioletti, M., Santucci, V., "Fitness landscape analysis of the permutation flowshop scheduling problem with total flow time criterion", in ICCSA, 2017.

[74] Czogalla, J., Fink, A., "Fitness landscape analysis for the resource constrained project scheduling problem", in LION, 2009.

[75] Tayarani-Najaran, M.-H., Prügel-Bennett, A., "On the landscape of combinatorial optimization problems", IEEE Transactions on Evolutionary Computation, Vol. 18, 2014, str. 420-434.

[76] Jones, T., Forrest, S., "Fitness distance correlation as a measure of problem difficulty for genetic algorithms", in ICGA, 1995.

[77] Naudts, B., Kallel, L., "A comparison of predictive measures of problem difficulty in evolutionary algorithms", IEEE Trans. Evol. Comput., Vol. 4, 2000, str. 1-15.

[78] He, J., Reeves, C., Witt, C., Yao, X., "A note on problem difficulty measures in black-box optimization: Classification, realizations and predictability", Evolutionary Computation, Vol. 15, 2007, str. 435-443.

[79] Pelikan, M., "Nk landscapes, problem difficulty, and hybrid evolutionary algorithms", in GECCO '10, 2010.

[80] Malan, K., Engelbrecht, A., "Particle swarm optimisation failure prediction based on fitness landscape characteristics", 2014 IEEE Symposium on Swarm Intelligence, 2014, str. 1-9.

[81] Suzuki, H., Iwasa, Y., "Ga performance in a babel-like fitness landscape", Proceedings Ninth IEEE International Conference on Tools with Artificial Intelligence, 1997, str. 357-366.

[82] Pitzer, E., Beham, A., Affenzeller, M., "Automatic algorithm selection for the quadratic assignment problem using fitness landscape analysis", in EvoCOP, 2013.

[83] Graham, R., Lawler, E., Lenstra, J., Kan, A., "Optimization and approximation in deterministic sequencing and scheduling: a survey", in Discrete Optimization II, ser. Annals of Discrete Mathematics, Hammer, P., Johnson, E., Korte, B., (ur.). Elsevier, 1979, Vol. 5, str. 287 - 326.

[84] Wu, C.-C., Bai, D., Chen, J.-H., Lin, W.-C., Xing, L., Lin, J.-C., Cheng, S.-R., "Several variants of simulated annealing hyper-heuristic for a single-machine scheduling with two-scenario-based dependent processing times", Swarm and Evolutionary Computation, Vol. 60, 2021, str. 100765.

[85] Wang, S., Cui, W., Chu, F., Yu, J., Gupta, J. N., "Robust (min–max regret) single machine scheduling with interval processing times and total tardiness criterion", Computers and Industrial Engineering, Vol. 149, 2020, str. 106838.

[86] Bachtler, O., Krumke, S. O., Le, H. M., "Robust single machine makespan scheduling with release date uncertainty", Operations Research Letters, Vol. 48, No. 6, 2020, str. 816 - 819.

[87] Davari, M., Ranjbar, M., De Causmaecker, P., Leus, R., "Minimizing makespan on a single machine with release dates and inventory constraints", European Journal of Operational Research, Vol. 286, No. 1, 2020, str. 115 - 128.

[88] Chen, L., Wang, J., Xu, X., "An energy-efficient single machine scheduling problem with machine reliability constraints", Computers and Industrial Engineering, Vol. 137, 2019, str. 106072.

[89] Aghelinejad, M., Ouazene, Y., Yalaoui, A., "Complexity analysis of energy-efficient single machine scheduling problems", Operations Research Perspectives, Vol. 6, 2019, str. 100105.

[90] Vlašić, I., Đurasević, M., Jakobović, D., "A comparative study of solution representations for the unrelated machines environment", Computers & Operations Research, Vol. 123, 2020, str. 105005.

[91] Wang, X., Li, Z., Chen, Q., Mao, N., "Meta-heuristics for unrelated parallel machines scheduling with random rework to minimize expected total weighted tardiness", Computers & Industrial Engineering, Vol. 145, 2020, str. 106505.

[92] Marinho Diana, R. O., de Souza, S. R., "Analysis of variable neighborhood descent as a local search operator for total weighted tardiness problem on unrelated parallel machines", Computers & Operations Research, Vol. 117, 2020, str. 104886.

[93] Lei, D., Liu, M., “An artificial bee colony with division for distributed unrelated parallel machine scheduling with preventive maintenance”, Computers & Industrial Engineering, Vol. 141, 2020, str. 106320.

[94] Perez-Gonzalez, P., Fernandez-Viagas, V., Zamora García, M., Framinan, J. M., “Constructive heuristics for the unrelated parallel machines scheduling problem with machine eligibility and setup times”, Computers & Industrial Engineering, Vol. 131, 2019, str. 131 - 145.

[95] Đurasević, M., Jakobović, D., “A survey of dispatching rules for the dynamic unrelated machines environment”, Expert Systems with Applications, Vol. 113, 2018, str. 555 - 569.

[96] Kelley, J. E., The Critical Path Method: Resource Planning and Scheduling. Englewood Cliffs, NJ, SAD: Prentice-Hall, 1963, str. 347-365.

[97] Artigues, C., Demassey, S., Néon, E., Sourd, F., “Resource-constrained project scheduling: Models, algorithms, extensions and applications”, Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications, 01 2010, str. 1-308.

[98] Chen, H., Ding, G., Qin, S., Zhang, J., “A hyper-heuristic based ensemble genetic programming approach for stochastic resource constrained project scheduling problem”, Expert Systems with Applications, 2020, str. 114174.

[99] Elloumi, S., Loukil, T., Fortemps, P., “Reactive heuristics for disrupted multi-mode resource-constrained project scheduling problem”, Expert Systems with Applications, 2020, str. 114132.

[100] Guo, W., Vanhoucke, M., Coelho, J., Luo, J., “Automatic detection of the best performing priority rule for the resource-constrained project scheduling problem”, Expert Systems with Applications, 2020, str. 114116.

[101] Sallam, K. M., Chakrabortty, R. K., Ryan, M. J., “A two-stage multi-operator differential evolution algorithm for solving resource constrained project scheduling problems”, Future Generation Computer Systems, Vol. 108, 2020, str. 432 - 444, dostupno na: http://www.sciencedirect.com/science/article/pii/S0167739X19317480

[102] Rahman, H. F., Chakrabortty, R. K., Ryan, M. J., “Memetic algorithm for solving resource constrained project scheduling problems”, Automation in Construction, Vol. 111, 2020, str. 103052.

[103] Chen, H., Ding, G., Zhang, J., Qin, S., "Research on priority rules for the stochastic resource constrained multi-project scheduling problem with new project arrival", Computers & Industrial Engineering, Vol. 137, 2019, str. 106060.

[104] Chu, Z., Xu, Z., Li, H., "New heuristics for the rcpsp with multiple overlapping modes", Computers & Industrial Engineering, Vol. 131, 2019, str. 146 - 156.

[105] Koza, J. R., "Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems", Stanford, CA, USA, Tech. Rep., 1990.

[106] Koza, J. R., Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge, MA, USA: MIT Press, 1992.

[107] Koza, J. R., Genetic Programming II: Automatic Discovery of Reusable Programs. Cambridge Massachusetts: MIT Press, May 1994.

[108] Koza, J. R., Andre, D., Bennett III, F. H., Keane, M., Genetic Programming III: Darwinian Invention and Problem Solving. Morgan Kaufman, Apr. 1999.

[109] Koza, J. R., Keane, M. A., Streeter, M. J., Mydlowec, W., Yu, J., Lanza, G., Genetic Programming IV: Routine Human-Competitive Machine Intelligence. Kluwer Academic Publishers, 2003.

[110] Mousavi Astarabadi, S. S., Ebadzadeh, M. M., "Genetic programming performance prediction and its application for symbolic regression problems", Information Sciences, Vol. 502, 2019, str. 418 - 433.

[111] Amir Haeri, M., Ebadzadeh, M. M., Folino, G., "Statistical genetic programming for symbolic regression", Applied Soft Computing, Vol. 60, 2017, str. 447 - 469.

[112] Gomes, F. M., Pereira, F. M., Silva, A. F., Silva, M. B., "Multiple response optimization: Analysis of genetic programming for symbolic regression and assessment of desirability functions", Knowledge-Based Systems, Vol. 179, 2019, str. 21 - 33.

[113] dal Piccol Sotto, L. F., de Melo, V. V., "Studying bloat control and maintenance of effective code in linear genetic programming for symbolic regression", Neurocomputing, Vol. 180, 2016, str. 79 - 93, progress in Intelligent Systems Design.

[114] Spector, L., Barnum, H., Bernstein, H. J., "Genetic programming for quantum computers", in Genetic Programming 1998: Proceedings of the Third Annual Conference, Koza, J. R., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M. H., Goldberg, D. E., Iba, H., Riolo, R., (ur.). University of Wisconsin, Madison, Wisconsin, USA: Morgan Kaufmann, 22-25 Jul. 1998, str. 365–373.

[115] Andre, D., Scholar, V., Bennett III, F., Koza, J., "Discovery by genetic programming of a cellular automata rule that is better than any known rule for the majority classification problem", 09 1996.

[116] Lohn, J., Hornby, G., Linden, D., "An evolved antenna for deployment on nasa's space technology 5 mission", in Genetic Programming Theory and Practice II, O'Reilly, U.-M., Yu, T., Riolo, R. L., Worzel, B., (ur.). Ann Arbor: Springer, may 2004, ch. 18, str. 301–315.

[117] Meza-Sánchez, M., Clemente, E., Rodríguez-Liñán, M., Olague, G., "Synthetic-analytic behavior-based control framework: Constraining velocity in tracking for nonholonomic wheeled mobile robots", Information Sciences, Vol. 501, 2019, str. 436 - 459.

[118] Mariot, L., Picek, S., Jakobovic, D., Leporati, A., "An evolutionary view on reversible shift-invariant transformations", in Genetic Programming, Hu, T., Lourenço, N., Medvet, E., Divina, F., (ur.). Cham: Springer International Publishing, 2020, str. 118–134.

[119] Moore, J. H., Olson, R. S., Chen, Y., Sipper, M., "Automated discovery of test statistics using genetic programming", Genet Program Evolvable Mach, Vol. 20, 2019, str. 127 - 137.

[120] Hernandez-Beltran, J. E., Diaz-Ramirez, V. H., Trujillo, L., Legrand, P., "Design of estimators for restoration of images degraded by haze using genetic programming", Swarm and Evolutionary Computation, Vol. 44, 2019, str. 49 - 63.

[121] Acharya, D., Goel, S., Asthana, R., Bhardwaj, A., "A novel fitness function in genetic programming to handle unbalanced emotion recognition data", Pattern Recognition Letters, Vol. 133, 2020, str. 272 - 279.

[122] Asim, K. M., Idris, A., Iqbal, T., Martínez-Álvarez, F., "Seismic indicators based earthquake predictor system using genetic programming and adaboost classification", Soil Dynamics and Earthquake Engineering, Vol. 111, 2018, str. 1 - 7.

[123] Álvarez Díaz, M., Caballero Miguez, G., "The quality of institutions: A genetic programming approach", Economic Modelling, Vol. 25, No. 1, 2008, str. 161 - 169.

[124] Hajek, P., Henriques, R., Castelli, M., Vanneschi, L., "Forecasting performance of regional innovation systems using semantic-based genetic programming with local search optimizer", Computera and Operations Research, Vol. 106, 2019, str. 179 - 190.

[125] Ritchie, M. D., Motsinger, A. A., Bush, W. S., Coffey, C. S., Moore, J. H., "Genetic programming neural networks: A powerful bioinformatics tool for human genetics", Applied Soft Computing, Vol. 7, No. 1, 2007, str. 471 - 479.

[126] Santorelli, M., Lam, C., Morsut, L., "Synthetic development: building mammalian multicellular structures with artificial genetic programs", Current Opinion in Biotechnology, Vol. 59, 2019, str. 130 - 140, tissue, Cell and Pathway Engineering.

[127] Fernández-Ares, A., Mora, A., García-Sánchez, P., Castillo, P., Merelo, J., "Analysing the influence of the fitness function on genetically programmed bots for a real-time strategy game", Entertainment Computing, Vol. 18, 2017, str. 15 - 29.

[128] Azaria, Y., Sipper, M., "Gp-gammon: Genetically programming backgammon players", Genetic Programming and Evolvable Machines, Vol. 6, No. 3, Sep. 2005, str. 283–300, published online: 12 August 2005.

[129] Ghazouani, H., Barhoumi, W., "Genetic programming-based learning of texture classification descriptors from local edge signature", Expert Systems with Applications, Vol. 161, 2020, str. 113667.

[130] Campobello, G., Dell'Aquila, D., Russo, M., Segreto, A., "Neuro-genetic programming for multigenre classification of music content", Applied Soft Computing, Vol. 94, 2020, str. 106488.

[131] Bader-El-Den, M. B., Poli, R., "A gp-based hyper-heuristic framework for evolving 3-sat heuristics", in GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation, Thierens, D., Beyer, H.-G., Bongard, J., Branke, J., Clark, J. A., Cliff, D., Congdon, C. B., Deb, K., Doerr, B., Kovacs, T., Kumar, S., Miller, J. F., Moore, J., Neumann, F., Pelikan, M., Poli, R., Sastry, K., Stanley, K. O., Stutzle, T., Watson, R. A., Wegener, I., (ur.), Vol. 2. London: ACM Press, jul 2007, str. 1749–1749.

[132] Miranda, P. B., Prudêncio, R. B., "Generation of particle swarm optimization algorithms: An experimental study using grammar-guided genetic programming", Applied Soft Computing, Vol. 60, 2017, str. 281 - 296.

[133] Poli, R., Di Chio, C., Langdon, W. B., "Exploring extended particle swarms: a genetic programming approach", in GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation, Beyer, H.-G., O'Reilly, U.-M., Arnold, D. V., Banzhaf, W., Blum, C., Bonabeau, E. W., Cantu-Paz, E., Dasgupta, D., Deb, K., Foster, J. A., de Jong, E. D., Lipson, H., Llora, X., Mancoridis, S., Pelikan, M., Raidl, G. R., Soule, T., Tyrrell, A. M., Watson, J.-P., Zitzler, E., (ur.), Vol. 1. Washington DC, USA: ACM Press, jun 2005, str. 169–176.

[134] Oltean, M., Dumitrescu, D., "Evolving TSP heuristics using multi expression programming", in Computational Science - ICCS 2004: 4th International Conference, Part II,

ser. Lecture Notes in Computer Science, Bubak, M., van Albada, G. D., Sloot, P. M. A., Dongarra, J., (ur.), Vol. 3037. Krakow, Poland: Springer-Verlag, jun 2004, str. 670–673.

[135] Chand, S., Huynh, Q., Singh, H., Ray, T., Wagner, M., "On the use of genetic programming to evolve priority rules for resource constrained project scheduling problems", Information Sciences, Vol. 432, 2018, str. 146 - 163.

[136] Langdon, W. B., Soule, T., Poli, R., Foster, J. A., "The evolution of size and shape", in Advances in Genetic Programming 3, Spector, L., Langdon, W. B., O'Reilly, U.-M., Angeline, P. J., (ur.). Cambridge, MA, USA: MIT Press, Jun. 1999, ch. 8, str. 163–190.

[137] Rosca, J., "A probabilistic model of size drift", in Genetic Programming Theory and Practice, Riolo, R. L., Worzel, B., (ur.). Kluwer, 2003, ch. 8, str. 119–135.

[138] Nguyen, Q. U., Chu, T. H., "Semantic approximation for reducing code bloat in genetic programming", Swarm and Evolutionary Computation, Vol. 58, 2020, str. 100729.

[139] Banzhaf, W., "Genetic programming for pedestrians", in Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93, Forrest, S., (ur.). University of Illinois at Urbana-Champaign: Morgan Kaufmann, 17-21 Jul. 1993, str. 628.

[140] Eklund, S. E., "A massively parallel GP engine in VLSI", in Proceedings of the 2002 Congress on Evolutionary Computation CEC2002, Fogel, D. B., El-Sharkawi, M. A., Yao, X., Greenwood, G., Iba, H., Marrow, P., Shackleton, M., (ur.). IEEE Press, 12-17 May 2002, str. 629–633.

[141] Poli, R., "Parallel distributed genetic programming", in New Ideas in Optimization, ser. Advanced Topics in Computer Science, Corne, D., Dorigo, M., Glover, F., (ur.). Maidenhead, Berkshire, England: McGraw-Hill, 1999, ch. 27, str. 403–431.

[142] Shan, Y., McKay, R. I., Baxter, R., Abbass, H., Essam, D., Hoai, N. X., "Grammar model-based program evolution", in Proceedings of the 2004 IEEE Congress on Evolutionary Computation. Portland, Oregon: IEEE Press, 20-23 Jun. 2004, str. 478–485.

[143] Bartoli, A., De Lorenzo, A., Medvet, E., Squillero, G., "Multi-level diversity promotion strategies for grammar-guided genetic programming", Applied Soft Computing, Vol. 83, 2019, str. 105599.

[144] Miller, J. F., "An empirical study of the efficiency of learning Boolean functions using a cartesian genetic programming approach", in Proceedings of the Genetic and Evolutionary Computation Conference, Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., Smith, R. E., (ur.), Vol. 2. Orlando, Florida, USA: Morgan Kaufmann, 13-17 Jul. 1999, str. 1135–1142.

[145] Miller, J. F., Smith, S. L., "Redundancy and computational efficiency in cartesian genetic programming", IEEE Transactions on Evolutionary Computation, Vol. 10, No. 2, Apr. 2006, str. 167–174.

[146] Blickle, T., Thiele, L., "A comparison of selection schemes used in evolutionary algorithms", Evolutionary Computation, Vol. 4, No. 4, 1996, str. 361-394.

[147] Goldberg, D. E., Deb, K., "A comparative analysis of selection schemes used in genetic algorithms", ser. Foundations of Genetic Algorithms, RAWLINS, G. J., (ur.). Elsevier, 1991, Vol. 1, str. 69 - 93.

[148] D'haeseleer, P., "Context preserving crossover in genetic programming", in Proceedings of the 1994 IEEE World Congress on Computational Intelligence, Vol. 1. Orlando, Florida, USA: IEEE Press, 27-29 Jun. 1994, str. 256–261.

[149] Langdon, W. B., "Size fair and homologous tree genetic programming crossovers", in Proceedings of the Genetic and Evolutionary Computation Conference, Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., Smith, R. E., (ur.), Vol. 2. Orlando, Florida, USA: Morgan Kaufmann, 13-17 Jul. 1999, str. 1092–1097.

[150] Luke, S., Spector, L., "A comparison of crossover and mutation in genetic programming", in Genetic Programming 1997: Proceedings of the Second Annual Conference, Koza, J. R., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M., Iba, H., Riolo, R. L., (ur.). Stanford University, CA, USA: Morgan Kaufmann, 13-16 Jul. 1997, str. 240–248.

[151] Jones, T., "Evolutionary Algorithms, Fitness Landscapes and Search", Doktorski rad, University of Waterloo, 06 1995.

[152] Reeves, C. R., "Fitness landscapes and evolutionary algorithms", in Artificial Evolution, Fonlupt, C., Hao, J.-K., Lutton, E., Schoenauer, M., Ronald, E., (ur.). Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, str. 3–20.

[153] Malan, K. M., Engelbrecht, A. P., "A survey of techniques for characterising fitness landscapes and some possible ways forward", Information Sciences, Vol. 241, 2013, str. 148 - 163.

[154] Malan, K. M., Engelbrecht, A. P., Fitness Landscape Analysis for Metaheuristic Performance Prediction. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, str. 103–132.

[155] Vanneschi, L., Tomassini, M., Collard, P., Vérel, S., "Negative slope coefficient: A measure to characterize genetic programming fitness landscapes", in Genetic Programming, Collet, P., Tomassini, M., Ebner, M., Gustafson, S., Ekárt, A., (ur.). Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, str. 178–189.

[156] Vanneschi, L., Gustafson, S., Mauri, G., "Using subtree crossover distance to investigate genetic programming dynamics", in Genetic Programming, Collet, P., Tomassini, M., Ebner, M., Gustafson, S., Ekárt, A., (ur.). Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, str. 238–249.

[157] Vanneschi, L., Pirola, Y., Collard, P., "A quantitative study of neutrality in gp boolean landscapes", in Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, ser. GECCO '06. New York, NY, USA: Association for Computing Machinery, 2006, str. 895–902.

[158] Vanneschi, L., Pirola, Y., Mauri, G., Tomassini, M., Collard, P., Verel, S., "A study of the neutrality of boolean function landscapes in genetic programming", Theoretical Computer Science, Vol. 425, 2012, str. 34 - 57, theoretical Foundations of Evolutionary Computation.

[159] Vanneschi, L., Tomassini, M., Collard, P., Vérel, S., Pirola, Y., Mauri, G., "A comprehensive view of fitness landscapes with neutrality and fitness clouds", in Genetic Programming, Ebner, M., O'Neill, M., Ekárt, A., Vanneschi, L., Esparcia-Alcázar, A. I., (ur.). Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, str. 241–250.

[160] Moraglio, A., Krawiec, K., Johnson, C. G., "Geometric semantic genetic programming", in Parallel Problem Solving from Nature - PPSN XII, Coello, C. A. C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M., (ur.). Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, str. 21–31.

[161] Pétrowski, A., "An efficient hierarchical clustering technique for speciation", Evolution. Technical report, Institute National des Telecommunications, Evry, France, Technique Report, 1997.

[162] Jie Yao, Kharma, N., Yu Qing Zhu, "On clustering in evolutionary computation", in 2006 IEEE International Conference on Evolutionary Computation, 2006, str. 1752-1759.

[163] Maree, S. C., Alderliesten, T., Thierens, D., Bosman, P. A. N., "Real-valued evolutionary multi-modal optimization driven by hill-valley clustering", in Proceedings of the Genetic and Evolutionary Computation Conference, ser. GECCO '18. New York, NY, USA: Association for Computing Machinery, 2018, str. 857–864.

[164] Harrison, K. R., Ombuki-Berman, B. M., Engelbrecht, A. P., "The parameter configuration landscape: A case study on particle swarm optimization", in 2019 IEEE Congress on Evolutionary Computation (CEC), 2019, str. 808-814.

[165] Pavelski, L. M., Delgado, M. R., Kessaci, M.-E., "Meta-learning on flowshop using fitness landscape analysis", in Proceedings of the Genetic and Evolutionary Computation Conference, ser. GECCO '19. New York, NY, USA: Association for Computing Machinery, 2019, str. 925–933.

[166] Hutter, F., Hoos, H. H., Leyton-Brown, K., Stützle, T., "Paramils: An automatic algorithm configuration framework", J. Artif. Int. Res., Vol. 36, No. 1, Sep. 2009, str. 267–306.

[167] Kadioglu, S., Malitsky, Y., Sellmann, M., Tierney, K., "Isac –instance-specific algorithm configuration", in Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence. NLD: IOS Press, 2010, str. 751–756.

[168] López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T., "The irace package: Iterated racing for automatic algorithm configuration", Operations Research Perspectives, Vol. 3, 2016, str. 43 - 58.

[169] Đurasević, M., "Automated design of dispatching rules in unrelated machines environment", Doktorski rad, Fakultet elektrotehnike i računarstva, Zagreb, 2018.

[170] Đumić, M., "Oblikovanje prioritetnih pravila za problem raspoređivanja s ograničenim sredstvima", Doktorski rad, Fakultet elektrotehnike i računarstva, Zagreb, 2020.

[171] Moon, T., "The expectation-maximization algorithm", IEEE Signal Processing Magazine, Vol. 13, 1996, str. 47-60.

[172] Schwarz, G., "Estimating the dimension of a model", Annals of Statistics, Vol. 6, 1978, str. 461-464.

[173] Grove, D., Sakamoto, Y., Ishiguro, M., Kitagawa, G., "Akaike information criterion statistics", The Statistician, Vol. 37, 1988, str. 477-478.

[174] Rousseeuw, P. J., "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis", Journal of Computational and Applied Mathematics, Vol. 20, 1987, str. 53 - 65.

[175] Davies, D. L., Bouldin, D. W., "A cluster separation measure", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-1, No. 2, 1979, str. 224-227.

[176] Caliński, T., Harabasz, J., "A dendrite method for cluster analysis", Communications in Statistics, Vol. 3, No. 1, 1974, str. 1-27.

[177] Hancer, E., Xue, B., Zhang, M., "A survey on feature selection approaches for clustering", Artificial Intelligence Review, 2020, str. 1-27.

[178] Dy, J. G., Brodley, C., “Feature selection for unsupervised learning”, J. Mach. Learn. Res., Vol. 5, 2004, str. 845-889.

[179] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., “Scikit-learn: Machine learning in Python”, Journal of Machine Learning Research, Vol. 12, 2011, str. 2825–2830.

[180] Fix, E., Hodges, J. L., “Discriminatory analysis - nonparametric discrimination: Small sample performance”, 1952.

[181] Alpaydin, E., “Introduction to machine learning”, in Adaptive computation and machine learning, 2004.

[182] Cox, D. R., “The regression analysis of binary sequences”, Journal of the Royal Statistical Society. Series B (Methodological), Vol. 20, No. 2, 1958, str. 215–242.

[183] Cortes, C., Vapnik, V., “Support-vector networks”, Machine Learning, Vol. 20, 2004, str. 273-297.

[184] Quinlan, J., “C4.5: Programs for machine learning”, 1992.

[185] Rumelhart, D., Hinton, G. E., Williams, R. J., “Learning representations by back-propagating errors”, Nature, Vol. 323, 1986, str. 533-536.

Životopis

Rebeka Čorić rođena je 28. prosinca 1989. godine u Našicama, Hrvatska. Nakon završetka prirodoslovno matematičke gimnazije u Našicama, 2008. godine upisuje preddiplomski studij matematike na Odjelu za matematiku Sveučilišta J. J. Strossmayera u Osijeku koji završava 2011. godine. Diplomski studij matematike, smjer: financijska i poslovna matematika na Odjelu za matematiku završava 2014. godine s temom diplomskog rada: "Linearno programiranje i primjene", a 2017. završava i sveučilišni nastavnički studij matematike i informatike.

Od studenog 2013. godine do svibnja 2014. godine radi kao programer pripravnik u tvrtki Alpha Score d.o.o. Od svibnja 2014. godine zaposlena je kao asistent na katedri za računarstvo Odjela za matematiku Sveučilišta J.J. Strossmayera u Osijeku. Sudjeluje u izvođenju kolegija iz matematike i računarstva na preddiplomskim i diplomskim studijima, kao i na drugim sastavnicama sveučilišta. Njezini istraživački interesi obuhvaćaju evolucijsko računarstvo, analizu krajolika dobrote i probleme raspoređivanja. Objavila je 3 rada u časopisima, od kojih su dva indeksirana u bazama CC i SCI Expanded, a jedan u bazi SCOPUS, te 4 rada na međunarodnim znanstvenim skupovima.

Popis objavljenih djela

Rad u časopisima

1. Čorić, R., Đumić, M., Jakobović, D., Genetic programming hyperheuristic parameter configuration using fitness landscape analysis, Applied Intelligence, prihvaćen za objavljivanje
2. Đumić, M., Šišejković, D., Čorić, R., Jakobović, D., Evolving priority rules for resource constrained project scheduling problem with genetic programming, Future Generation Computer Systems 86, rujan 2018., str. 211-221.
3. Čerkez, N., Čorić, R., Đumić, M., Matijević, D., Finding an optimal seating arrangement for employees traveling to an event, Croatian Operational Research Review 6/2, listopad 2015., str. 419-427.

Radovi na međunarodnim znanstvenim skupovima

1. Čorić, R., Đumić, M., Jelić, S., A clustering model for time-series forecasting, 42nd International Convention - MIPRO 2019, Opatija, 2019, str. 1295-1299.
2. Čorić, R., Đumić, M., Jelić, S., A Genetic Algorithm for Group Steiner Tree Problem, 41st International Convention - MIPRO 2018, Opatija, Hrvatska, 2018, str. 1113-1118.
3. Čorić, R., Đumić, M., Jakobović, D., Complexity Comparison of Integer Programming and Genetic Algorithms for Resource Constrained Scheduling Problems, 40th International ICT Convention - MIPRO 2017, Opatija, 2017, str. 1394-1400.
4. Čorić, R., Picek, S., Jakobović, D., Coello Coello, C.A., On the mutual information as a fitness landscape measure, GECCO '17 Proceedings of the Genetic and Evolutionary Computation Conference Companion, Berlin, Germany, 2017, str. 165-166.

Biography

Rebeka Čorić was born on December 28th, 1989 in Našice, Croatia. After finishing highschool in Našice in 2008, she enrolled the undergraduate study of mathematics at the Department of Mathematics of the J. J. Strossmayer University of Osijek which she completed in 2011. In 2014 she graduated from the Department of mathematics in the field of Financial and Business Mathematics with the topic: "Linear programming and applications", and in 2017 she completed integrated undergraduate and graduate university study program in mathematics and computer science at the same Department.

From November 2013 to May 2014, she worked as a programmer at Alpha Score d.o.o. Since May 2014, she has been working as a teaching and research assistant in Computer Science Research Group at the Department of Mathematics of the J. J. Strossmayer University of Osijek. She is involved in teaching courses in field of mathematics and informatics for undergraduate and graduate study as well as in other university departments. Her research interests include the fields of evolutionary computing, fitness landscape analysis and scheduling problems. She published 3 journal papers, two of which are in the journals indexed in Current Contents and SCI Expanded, and one is in the journal indexed in SCOPUS, and 4 conference papers.