Универзитет у Београду
Електротехнички факултет

Дражен Д. Драшковић

Софтверски систем за учење и примену алгоритама вештачке интелигенције

Докторска дисертација

Београд, 2018.

UNIVERSITY OF BELGRADE
SCHOOL OF ELECTRICAL ENGINEERING

Dražen D. Drašković

SOFTWARE SYSTEM FOR LEARNING AND APPLICATION OF ARTIFICIAL INTELLIGENCE ALGORITHMS

Doctoral dissertation

Belgrade, 2018.

Ментори:

др Бошко Николић, редовни професор

Универзитет у Београду - Електротехнички факултет

др Милош Цветановић, ванредни професор

Универзитет у Београду - Електротехнички факултет

Чланови комисије:

др Бошко Николић, редовни професор

Универзитет у Београду - Електротехнички факултет

др Милош Цветановић, ванредни професор

Универзитет у Београду - Електротехнички факултет

др Зоран Шеварац, доцент

Универзитет у Београду - Факултет организационих наука

др Предраг Тадић, доцент

Универзитет у Београду - Електротехнички факултет

др Јелица Протић, редовни професор

Универзитет у Београду - Електротехнички факултет

Датум одбране:

_____ године.

„Знање је светлост која осветљава наш пут кроз живот
и води нас у живот будућности пун вечне славе.“

Михајло Пупин

Захвалнице

Ова докторска дисертација представља резултат мог рада од уписивања докторских студија у децембру 2011. године на Електротехничком факултету у Београду, па све до данас. У њој су обухваћени аспекти развоја и тестирања савремених софтверских система, показане примене алгоритама вештачке интелигенције у едукационе и индустријске сврхе и дати нови правци истраживања у области вештачке интелигенције.

Овом приликом бих желео пре свих да се захвалим својим менторима, проф. др Бошку Николићу, на предлогу ове интересантне теме и што ме је увео у област истраживања, као и проф. др Милошу Цветановићу, на многим сугестијама и великој помоћи у завршним фазама истраживања и писања ове дисертације. Њихова запажања су била драгоцена за напредак мог рада на докторским студијама, а коментари су значајно допринели квалитету саме дисертације. Хвала и комисији на прегледу ове дисертације.

Захвалио бих се колегама са којима сам блиско сарађивао, делио први радни простор и са којима сам од запослења на ЕТФ-у успешно радио на многим научним радовима – Немањи Којићу, Марку Мишићу, Жарку Станисављевићу, Игору Анђелковићу. Заједнички смо се борили у настави, науци, на пројектима, у лепим и мање лепим тренуцима. Захваљујем се сарадницима који су помогли у деловима овог истраживачког рада и свим својим студентима који су тестирали рад реализованог софтверског система. Дугујем захвалност и професорима који су ме водили на прве конференције и увели у истраживачки рад, пре свих професори Вељко Милутиновић и Јоца Ђорђевић. Још двоје професора имају значајно место у мом животу, јер су ме задржали да се бавим педагошким радом на ЕТФ-у и убедили ме да останем у Србији – моја драга професорка Јелица Протић и мој драги покојни професор Влада Благојевић Блеки. Хвала и свим наставницима и сарадницима Катедре за рачунарску технику и информатику и ЕТФ-а са којима сам дивно сарађивао током претходних осам година. ЕТФ је моја друга кућа, ако не и прва, како воли да каже мој драги професор Бранко Ковачевић.

Велико хвала и колегама са Факултета за рачунарство и информатику Универзитета у Љубљани, са којима сам успешно сарађивао на истраживачким пројектима претходних година: професору Марку Бајецу, Славку Житнику, Марку Јанковићу и Ловри Шубељу.

Моји Анчи, Сањица, Крле, Коста, Заки, Сале, Марија, Жика, Мићко, Уки, Јоцко, Маја, Фића, Милана, Сања, Роглић, Цвеле, Небор, Павковић, Влада, Пивке, Кети и моји другари из Истека, моји кумови Новаковићи, њихов ведри дух, позитивна енергија и безрезервна подршка давали су ми ентузијазам да дођем до остварења овог циља. Надам се да ћу сва путовања и све заједничке лепе догађаје које сам пропустио са својим пријатељима, због својих оправданих разлога, надокнадити у наредном периоду.

Посебну захвалност дугујем својој породици - брату Драшку, родитељима Милени и Драгану и мојој баки Смиљки, особи од које сам научио да волим књигу, да будем позитиван и од које сам научио да ценим оно што ми је живот дао. Они су били моја највећа подршка кроз живот и школовање и зато ову дисертацију посвећујем њима.

У Београду, Дражен Драшковић
28. јуна 2018. године

Резиме

Наслов: Софтверски систем за учење и примену алгоритама вештачке интелигенције

У овој докторској дисертацији описан је софтверски систем за учење и примену алгоритама вештачке интелигенције назван *SAIL* (енг. *System for Artificial Intelligence Learning*). У систему је реализовано 30 стандардних алгоритама претраживања и алгоритама теорије игара, алгоритама резоновања и репрезентација знања, алгоритама за решавање проблема, алгоритама за рад у неизвесном окружењу и алгоритама машинског учења. Идеја да се реализује овакав софтверски систем проистекла је пре свега из наставе, на предмету Интелигентни системи, где су студенти због разноликости ових алгоритама врло тешко разумели рад тих алгоритама и концепте који су предавани. Значај оваквог система је што он такође осим у настави, може да се примени и у решавању практичних и реалних проблема, па су показане и његове примене код проблема налажења слободног места на паметном паркингу са бежичном сензорском мрежом, код система за одлучивање у медицинске сврхе и код система за обраду велике количине података.

Истраживање током дисертације обухватило је преглед постојећих софтверских система који се користе у овој области на различитим светским универзитетима, примену таквих система у електронском и мобилном учењу, самосталном учењу и учењу на даљину, дефинисање захтева за израду потпуно новог софтверског система заснованог на резултатима анализе постојећих система, реализацију новог вишеплатформског система заједничког корисничког интерфејса за све симулације и његову примену.

Софтверски систем *SAIL* је модуларно пројектован, итеративном методологијом уз делимичну примену агилног приступа и развој тестова током процеса имплементације. Систем је осмишљен тако да буде лак за имплементирање, тестирање и надоградњу. У првој верзији реализована је рачунарска апликација са алгоритмима претраживања, којом је показан значај визуелизације алгоритама, висок ниво интеракције између корисника и система, извршавање симулације алгоритма корак по корак и приказ симулације у реалном времену. Осим тога реализовано је учитавање поставке предефинисаних проблема и задатака из датотеке и њихово извршавање у оквиру система, могућност прављења сопствених примера, примера са променљивим улазним параметрима симулације, снимање текућих примера са свим параметрима и учитавање раније снимљених примера. У другој верзији система су додате симулације алгоритама теорије игара, алгоритми резоновања, продукциони системи, *GPS* и *STRIPS* алгоритам, а затим су у наредним верзијама додати алгоритми рада у неизвесном окружењу и стратегије машинског учења. У дисертацији је дискутовано и за које проблеме и задатке је најбоље употребити коју групу алгоритама.

Разноликост међу реализованим алгоритмима утицала је да се пре почетка реализације организује заједнички графички кориснички интерфејс за све симулације, са интуитивним заједничким командама. Свака симулација додатно има своје специфичне команде у траци са алатима и корисничком менију. Систем подржава и опцију извожења симулираних примера у *PDF* датотеку и вишејезичност, па у систему тренутно постоје српски језик ћириличног писма, српски језик латиничног писма и енглески језик, али је могућа врло лака надоградња и за друге језике.

Један од битних циљева истраживања био је да овај софтверски систем буде вишеплатформски, односно да осим рачунарске апликације буде реализована и мобилна апликација, која би нашла примену у самосталном учењу уз помоћ мобилних уређаја. Имплементиране класе из рачунарске апликације искоришћене су приликом пројектовања мобилне платформе уз генерисање новог изгледа апликације прилагођеног екранима на мобилним уређајима.

Уз софтверски систем *SAIL* који је реализован као рачунарска и мобилна апликација, у оквиру истраживања приказана је и могућност његове надоградње као компјутерске игре и показана је примена игара у едукацији (енг. *gamification*). Игре су направљене тако да корисник игре може у сваком тренутку да види извршавање алгоритма уз граф/стабло

претраживања, неки други графички или текстуални приказ и да има могућност да направи упоредну анализу алгоритама током њиховог извршавања.

На крају истраживања приказане су примене овог софтверског система у настави и у пракси, и дати су резултати кроз евалуацију. Евалуација је показала да примена оваквог система у настави има јако велики значај и за предаваче и за студенте, јер су резултати примене овог система у интервалу од шест година показали да студенти боље разумеју алгоритме обухваћене овим софтверским системом и да је њихов успех, као и знање које су стекли, веће него у временском периоду пре примене овог софтверског система.

Примери из реалних система су показали да такви системи засновани на алгоритмима вештачке интелигенције имају веома широку примену у многим областима и данас не постоји софтверски систем који не користи неки аспект вештачке интелигенције.

Кључне речи: интелигентни системи, пројектовање софтвера, алгоритми претраживања, алгоритми резоновања и репрезентације знања, алгоритми за решавање проблема, алгоритми за рад у неизвесном окружењу, алгоритми машинског учења, визуелне симулације, електронско и мобилно учење, примена игара у едукацији

Научна област: Електротехника и рачунарство

Ужа научна област: Софтверско инжењерство

УДК број: 621.3

ABSTRACT

Title: Software system for learning and application of artificial intelligence algorithms

The aim of this doctoral thesis is to present a software system for learning and application of the artificial intelligence algorithms SAIL (System for Artificial Intelligence Learning). Within the system 30 standard algorithms have been realised: search and game theory algorithms, reasoning and knowledge representation algorithms, problem solving algorithms, uncertain environment algorithms and machine learning algorithms. The idea to realise this system was generated during the teaching of the course Intelligent Systems, where students had difficulties to understand these algorithms and related concepts due to their complexity. The importance of this system is in its applications – in addition to its educational purposes, the system can be applied in solving practical and real problems. Its applications in finding free parking spot within the smart parking with wireless sensor network, within the system for decision-making for medical applications and within the big data processing system have been demonstrated.

This research work included the review of the existing software systems that are used in this area at universities across the world, application of such systems in electronic and mobile learning, individual and distance learning, requirements definition for the development of completely new software system following the analysis of the existing systems, realisation of the new multiplatform system of common user interface for all the simulations and applications.

Software system SAIL was modularly designed, using iterative methodology and partially using agile approach and testing during its implementation. The system was designed to be easy to implement, test and upgrade. In its first version, a computer application with search algorithms was

realised, which demonstrated the importance of algorithm visualisation, high level of interaction between the user and the system, execution of the algorithm simulation step-by-step and real-time simulation. Furthermore, loading of existing predefined problems and exercises from files and their execution within the system have been realised, the possibility to define new examples, examples with changed input parameters, saving of examples and loading of saved examples. The second version of the system added game theory algorithms simulations, reasoning algorithms, production systems, GPS and STRIPS algorithm, and the following versions added the uncertain environment algorithms and machine learning strategies. This research work discusses which problems and exercises are most suitable for certain groups of algorithms.

Diversity among realised algorithms influenced the choice of the common graphical user interface for all the simulations, with intuitive commands. In addition, every simulation has its own specific commands in the toolbar and user menu. The system supports exporting of simulated examples into a PDF file and it has multilingual support, it currently supports Serbian Cyrillic, Serbian Latin and English, with an option to easily add other languages.

One of the important aims of this research work was to create a multiplatform system and in addition to a computer application, a mobile application was realised, which finds use in self-learning using mobile devices. The implemented classes from the computer application were used during the design of the mobile platform and the new application was adjusted to mobile screens.

In addition to the software system SAIL realised as computer and mobile application, this research work presented the possibility of its upgrade as a computer game and the concept of gamification was demonstrated. Games are designed in a way that allows user to see running of the algorithm with a graph/search tree, other graphic or textual representation, with the possibility to compare algorithms during their execution.

The final part of this research presents the applications of this software system in education and in practice, and it includes the evaluation of the results. The evaluation demonstrated that the application of this system in education is of great importance for both instructors and students, as the results of the application of this system during six years of use show that students understand the included algorithms much better and that their success and acquired knowledge are greater when compared with the period before this software system was applied.

Examples from real systems have shown that such systems based on artificial intelligence algorithms have a very wide application in many areas and today there is no software system that does not use some aspect of artificial intelligence.

Keywords: intelligent systems, software design, search algorithms, algorithms for reasoning and representation of knowledge, algorithms for solving problems, algorithms for working in an uncertain environment, machine learning algorithms, visual simulations, electronic and mobile learning, gamification

Scientific field: Electrical Engineering and Computing

Scientific subfield: Software Engineering

UDC number: 621.3

КРАТАК САДРЖАЈ

САДРЖАЈ

1. Увод

Вештачка интелигенција је научна дисциплина која има за циљ да учи и развија интелигентне машине и софтверске системе, који обављају послове који захтевају неку врсту резоновања, односно закључивања, што је својствено људској интелигенцији. Изучавање области вештачке интелигенције обухвата хеуристичке алгоритме за претраживање и планирање, формалне начине за репрезентацију знања и резоновања, технике машинског учења, методе које се примењују на проблеме као што су разумевање говора и језика, компјутерски вид и роботика. Алгоритми из ове области данас се користе у широком спектру других области укључујући медицинске дијагностике, контролу роботских система, управљање индустријским системима, телекомуникације, финансије и трговину акцијама на берзи, индустрију компјутерских игара, музичку индустрију и многе друге [1]. Развојем вештачке интелигенције и свих њених подобласти, данас се све више развијају и интелигентни софтверски системи, који на основу улазних података, коришћењем једног алгоритма или групе алгоритама, дају неке излазне податке као резултат извршавања.

Вештачка интелигенција се данас активно примењује у софтверским системима за учење, експертским системима, при анализи и обради података, у роботским системима у тешкој индустрији, у системима заснованим на неуралним мрежама или на обради природних језика и у индустрији игара [2]. Због комплексности алгоритама из области вештачке интелигенције постоји потреба за системом који би омогућио коришћење како у свакодневном раду, тако и у процесу образовања. Предмет овог рада је евалуација постојећих софтверских система у области вештачке интелигенције и предлог новог система који би корисницима, односно инжењерима омогућио симулацију различитих алгоритама са циљем да стекну увид у могуће ефекте одређених алгоритама и на тај начин лакше одаберу адекватан алгоритам за специфичне потребе, али и да кроз симулацију на оптималан начин

одреде параметре са којима би требало да користе одређене алгоритме. Систем би такође био прилагођен студентима који би систем могли да користе у оквиру наставе са циљем лакшег разумевања суштине вештачке интелигенције.

Технике електронског учења (е-учења, енг. *e-learning*) и мобилног учења (м-учења, енг. *mobile learning / m-learning*), учења на даљину (енг. *distance learning*) и самосталног учења (енг. *selflearning*) примењују се у великој мери у образовању, чиме постижу већу интерактивност између предавача и слушалаца [3] [4] [5]. Системи реализовани на овим техникама све више се користе у системима образовања, како у основним и средњошколским системима, тако и на факултетима [6] [7]. Универзитетски курсеви преселили су се у највећој мери на интернет, нудећи кроз веб системе предавања, електронске материјале и форуме за дискусију студената. Са друге стране, испитивања и електронске провере знања се много спорије пребацују у облачне (енг. *cloud*) технологије. Популарни веб системи за учење (енг. *Learning Management System*, *LMS*) нашли су примену и у курсевима који примењују учење на даљину, али и као наставно средство у традиционалним курсевима високошколског образовања. Постоји много позитивних извештаја када је у питању коришћење *LMS* у образовању, јер помажу наставницима код организације предмета са већим бројем студената. Неколико студија је показало да овакви системи за е-учење и учење на даљину имају позитивне ефекте на учење код студената, уколико се адекватно примењују у оквиру академских курсева [8] [9]. Да би ефекат од коришћења оваквих система био већи, за самостално учење неопходно је развити и специфичне софтверске системе, који би студентима помогли да савладају одређено градиво. Систем који ће бити реализован у овој докторској дисертацији биће заснован на искуствима и принципима који долазе из области е-учења и самосталног учења.

У рачунарским наукама, осим *LMS*, као помоћни софтверски алати у образовању студената најчешће се користе имплементације софтверских система за учење концепата хардверских система, попут симулација за архитектуру и организацију рачунара [10], симулација за рачунарске мреже и основне протоколе у локалним мрежама и на интернету [11, 12], визуелни симулатори за учење основних алгоритама и структура података [13] и алгоритама заштите података [14, 15]. У области вештачке интелигенције, постоји неколико система који се користе у настави на светским универзитетима и који ће бити детаљније описани у овој дисертацији, у оквиру другог поглавља.

Циљ овог рада је да се моделује и реализује софтверски систем за симулацију и визуелизацију алгоритама вештачке интелигенције, почев од основних стратегија претраге - алгоритама претраживања и алгоритама теорије игара, алгоритама закључивања и модела представљања знања, коришћењем предикатске и фази логике, све до напредних техника претраге, стабала одлучивања и индуктивног учења.

Прва фаза истраживања обухватила је преглед и класификацију алгоритама вештачке интелигенције и постојећих софтверских система за учење таквих алгоритама. Закључак ове фазе истраживања је да не постоји систем који обједињује све подржане алгоритме и њихов приказ.

Друга фаза истраживања која је спроведена у оквиру ове докторске дисертације заснована је на дефинисању и анализи функционалних захтева методама случајева коришћења (енг. *use case*) и нешто савременијим начином помоћу приче (енг. *story*).

Трећа фаза обухватила је постављање архитектуре софтверског система, моделовање, имплементацију и тестирање таквог система. У овом делу истраживање је било усмерено и на дизајнирању система за различите платформе, тако да се мањим променама у коду, исти програмски код може искористити за реализацију и рачунарске (десктоп) и мобилне апликације, пошто су такве апликације све популарније. Коришћењем исте логике и истих функционалности овог софтверског система показана је трансформација овог софтверског система у компјутерску игру доступну за више уређаја и платформи.

Четврта фаза истраживања која је спроведена у оквиру ове докторске дисертације представља евалуацију резултата коришћења овог реализованог софтверског система у образовном процесу и примену таквих алгоритама у различитим областима. Анализирани су резултати успеха студената на факултетском предмету Интелигентни системи током шест школских година, пре употребе и током развоја овог софтверског система, који је развијан у неколико фаза. Такође, приказана је примена ових алгоритама на реалним свакодневним проблемима и примена овог софтверског система као помоћног алата за анализу улазних података и закључивање у различитим областима. Неки од сложених инжењерских проблема који су анализирани у оквиру дисертације су налажење слободног паркинг места у оквиру великих паркинга покривених бежичним сензорским мрежама, помоћ у одређивању терапије коришћењем техника одлучивања (енг. *decision making systems*) и примена различитих врста алгоритама машинског учења на великим базама података (енг. *big data*).

Дисертација је подељена у осам поглавља. Структура саме дисертације прати редослед фаза истраживања. У другом поглављу дат је опис свих анализираних алгоритама вештачке интелигенције и анализиран је проблем пројектовања софтвера који ће подржати симулацију великог броја алгоритама у оквиру истог корисничког интерфејса на више платформи. У трећем поглављу представљени су софтверски системи реализовани на универзитетима у свету, који примењују алгоритме вештачке интелигенције, и дефинисани су сви захтеви које овакав софтверски систем мора да има, обухватајући и корисничке захтеве и захтеве за архитектуром система. Четврто поглавље приказује проблеме који су решавани приликом реализације и начин како да се имплементира овакав систем користећи савремене технологије и методологије развоја софтвера у области софтверског инжењерства. У оквиру четвртог поглавља такође је описана модуларност софтверског система и интеграција свих имплементираних алгоритама, као и вишеплатформска реализација. Пето поглавље даје приказ рада реализованог система. У шестом поглављу описана је примена софтверског система у образовном процесу и могућности за примену овог софтвера као помоћног алата за решавање инжењерских проблема у другим областима, заснованих на овој групи алгоритама вештачке интелигенције. Седмо поглавље даје резултате евалуације добијене применом реализованог софтверског система. На крају ове докторске дисертације дат је закључак који приказује кључне доприносе и даље правце истраживања. Списак коришћене литературе, списак скраћеница, листа слика и листа табела налазе се након закључка.

2. АНАЛИЗА ПРОБЛЕМА И ОПИС АЛГОРИТАМА ВЕШТАЧКЕ ИНТЕЛИГЕНЦИЈЕ

Први циљ који је постављен у овом истраживању био је да се реформише наставни програм предмета Интелигентни системи, користећи препоруке струковних организација из области рачунарства, *IEEE Computer Society* и *ACM* и да се предмет прилагоди темама које су препоручене у последњој верзији курикулума за рачунарске науке и софтверско инжењерство. Вештачка интелигенција је веома широка област, а главни системи који се изучавају у оквиру ове области су системи засновани на знању (енг. *knowledge-based systems*) и системи засновани на машинском учењу (енг. *machine learning systems*). Настава из области вештачке интелигенције и интелигентних система се на Електротехничком факултету у Београду изводи од 1987. године, али се развојем нових информационих технологија јавила потреба да се алгоритми и интелигентно понашање оваквих система студентима представи на ефикаснији начин.

Интелигентно понашање састоји се од примене стеченог знања, размишљања, планирања и способности да се науче нове ствари. Вештачка интелигенција је управо наука која се бави проучавањем и дизајнирањем интелигентних агената, односно система за аутономно наменско деловање и резоновање, са циљем да се успешно испуне задаци који су додељени таквом систему.

Многи софтверски инжењери и студенти рачунарских наука уче псеудо кодове ових алгоритама и углавном могу показати како такви алгоритми раде на мањим проблемима. Међутим, њихово разумевање самог алгоритма остаје површно, па када се сусрећу са већим проблемима, не могу применити своје знање. Да би се детаљније разумели алгоритми потребно је интерактивно окружење, које би омогућило извршавање сваког алгоритма корак по корак, посматрања и детаљне анализе понашања рада алгоритма од почетка до краја. Поред тога, такво интерактивно окружење треба да омогући кориснику да види промене у понашању одабраног алгоритма или да упореди два алгоритма увидом у паралелну визуелну репрезентацију и анимацију различитих алгоритама.

Други циљ био је да се реализује такав софтверски систем који ће подржати све битне теме и подтеме, које се препоручују на уводном курсу из вештачке интелигенције, али и да се подрже технике електронског учења (е-учења) и мобилног учења (м-учења), уз примену учења на даљину и самосталног учења. Пре развоја новог софтверског система, анализирани су постојећи системи који се користе у настави из области вештачке интелигенције, на светским универзитетима, њихове карактеристике, предности и недостаци.

Трећи циљ је био да се покаже да такав систем може имати примену и ван наставе, на реалним проблемима са којима се свакодневно сусрећемо и да се покаже предност оваквог система.

У овом поглављу биће анализиране теме и подтеме из области вештачке интелигенције, које су препоручене да се укључе у оквиру универзитетских курсева. Такође, биће приказани описи алгоритама који припадају тим областима и подобластима. Алгоритми су подељени у пет група: алгоритми претраживања, алгоритми резоновања и репрезентације знања, алгоритми за решавање проблема, алгоритми за рад у неизвесном окружењу и алгоритми машинског учења.

2.1. Теме и подтеме из области вештачке интелигенције

Према препорукама струковних организација *IEEE CS* и *ACM*, студент треба да има могућност да област вештачке интелигенције и интелигентих система изучава на најмање једном предмету, а највише три предмета [16]. Исходи учења које студенти треба да стекну у знању ове области су разумевање основних концепата, идентификовање потенцијала и ситуација када треба користити интелигентни систем, затим дизајнирање, имплементација и евалуација интелигентног система [17]. Предвиђено је да изучавање области вештачке

интелигенције укључује хеуристичке алгоритме за претрагу и планирање, формализме за репрезентацију знања и резоновања, технике машинског учења, методе које се примењују на проблеме као што су разумевање говора и језика, компјутерски вид и роботика.

Организација *ACM* је препорукама из 2013. године поделила област интелигентних система на 12 тема од којих су четири обавезне теме, а преосталих осам су изборне теме [18]. Свака тема има одређени број подтема и по неколико исхода учења. За теме које су обавезне дат је и предвиђени фонд часова који је препоручен да се тема толико времена обрађује.

Раније су све теме биле означене као базичне (енг. *Core*), а од 2013. су подељене на базичне нивоа 1 (енг. *Core-Tier 1*) и базичне нивоа 2 (енг. *Core-Tier 2*). Базичне теме нивоа 1 су апсолутно суштинске теме, које је потребно да студент научи за ниво основних академских студија (енг. *undergraduate*, *Bachelor degree*). Базичне теме нивоа 2 су такође важне теме и велика већина њих, према препорукама од 80% до 90% треба да буде укључена у ниво основних академских студија. Све четири групе тема које су наведене као обавезне за област вештачке интелигенције су класификоване као базичне нивоа 2 и то су следеће теме: основни појмови из вештачке интелигенције, стратегије основних претраживања, основни начини представљања знања и резоновања и основи машинског учења [18]. Оне су суштински важне за планове и програме рачунарских наука (енг. *Computer Science*), рачунарског инжењерства (енг. *Computer Engineering*) и у мањој мери софтверског инжењерства (енг. *Software Engineering*) и најчешће се такве теме обрађују на каснијим предметима на основним академским студијама. Ове препоруке се примењују на Електротехничком факултету Универзитета у Београду, у оквиру предмета под називом Интелигентни системи.

Предмет Интелигентни системи се на Електротехничком факултету у Београду изучава на два студијска програма: на програму „Електротехника и рачунарство“ у оквиру Одсека за рачунарску технику и информатику, на четвртој години основних академских студија, као обавезни предмет у седмом семестру, и на програму „Софтверско инжењерство“, на трећој или четвртој години основних академских студија, као изборни предмет у петом или седмом семестру. Недељни фонд часова је 5 часова, од тога 2 часа за предавања, 2 часа за аудиторне вежбе и решавање проблема на табли и 1 час за лабораторијске вежбе у рачунарским лабораторијама. На оба студијска програма овај предмет вреди 6 ЕСПБ (енг. *ECTS*). Часови предавања обухватају теоријске концепте тема

које су приказане у табели 1. Часови аудиторних вежби засновани су на анализи и решавању проблема из области које су предаване, а часови лабораторијских вежби служе за интерактивну демонстрацију помоћу симулација софтверског система и неопходни су за боље разумевање комплексних тема.

У табели 1 приказане су теме, њихов степен покривености и обухваћене подтеме на предмету под називом Интелигентни системи. Остале напредније теме које се налазе у тој табели обрађују се у оквиру других предмета на вишим годинама основних академских студија или на мастер академским студијама, као што су: неуралне мреже (енг. *Neural Networks*), прикупљање и анализа података (енг. *Data Mining*), машинско учење (енг. *Machine Learning*), роботика и аутоматизација (енг. *Robotics and Automation*), теорија роботских система (енг. *Robotics Systems Theory*), препознавање облика (енг. *Pattern Recognition*). У табели 2 приказане су напредније теме и на којим предметима, одсецима и студијским програмима су оне укључене.

ТАБЕЛА 1
ТЕМЕ ОБУХВАЋЕНЕ НА ПРЕДМЕТУ ИНТЕЛИГЕНТНИ СИСТЕМИ

ТЕМЕ [О]БАВЕЗНЕ / [И]ЗБОРНЕ	**НИВО ПОКРИВЕНОСТИ**	**ОБУХВАЋЕНЕ ПОДТЕМЕ**
Основни појмови [О]	⊕	Преглед проблема у вештачкој интелигенцији; Карактеристике главних проблема; Природа интелигентних агената;
Стратегије основних претраживања [О]	⊕	Проблеми простора и решавања проблема претраживањем; Факторска репрезентација; Неинформисана претрага; Хеуристике и информисана претрага; Простор и временска ефикасност претраге; Игре са два играча; Проблеми задовољења ограничења;
Основни начини представљања знања и резоновања [О]	⊕	Предикатска логика; Резолуција и доказивачи теорема; Уланчавање унапред и уланчавање уназад; Преглед пробабилистичког резоновања, Бајесова теорема;
Основи машинског учења [О]	⊕	Индуктивно учење; Једноставна учења заснована на основама статистике, стабла одлучивања;
Напредна претраживања [И]	⊕	Изградња стабала претраживања, динамички простор претраживања; Стохастичко претраживање – симулирано каљење (енг. *simulated annealing*), генетски алгоритми; A* претраживање, претраживање снопом; Претраживање *Minimax*, алфа-бета одсецање;
Напредни начини представљања знања и резоновања [И]	○	Семантичке мреже;
Резоновање у условима неизвесности [И]	●	Репрезентовање знања – Бајесове мреже; Теорија одлучивања;

ТАБЕЛА 2
НАПРЕДНЕ ТЕМЕ ВЕШТАЧКЕ ИНТЕЛИГЕНЦИЈЕ ОБУХВАЋЕНЕ НА ДРУГИМ ПРЕДМЕТИМА

ТЕМЕ [О]БАВЕЗНЕ / [И]ЗБОРНЕ	**НИВО ПОКРИВЕНОСТИ**	**НИВО СТУДИЈА / СТУД.ПРОГРАМ И ОДСЕК / ГОДИНА СТУДИЈА / ПРЕДМЕТ И ТИП (НИВО СТУДИЈА: [ОС] - ОСНОВНЕ, [МС] - МАСТЕР) (ТИП ПРЕДМЕТА: [О]БАВЕЗНИ, [И]ЗБОРНИ)**
Агенти [И]	○	ОС, Сигнали и системи, IV година, Вештачка интелигенција, И
Обрада природног језика [И]	⊕	МС, Софтверско инжењерство, I година, Обрада природних језика, И
Напредно машинско учење [И]	●	МС, Рачунарска техника и информатика, I година, Проналажење скривеног знања, И
Роботика [И]	⊕	ОС, Сигнали и системи, III година, Роботика и аутоматизација, И IV година, Теорија роботских система, И
Перцепција и компјутерска визија [И]	●	МС, Сигнали и системи, I година, Компјутерска визија, О

ЛЕГЕНДА	**НИВО ПОКРИВЕНОСТИ**
○	Делимичан ниво покривености тема
●	Значајан ниво покривености тема
⊕	Потпуни ниво покривености тема

Из табеле 1 може се видети да су све четири обавезне теме препоручене од стране ACM потпуно покривене, а још три изборне су покривене у мањем или већем обиму. Алгоритми и репрезентације знања, који су укључени у градиво предмета Интелигентни системи, тематски су веома разнолики и обухватају:

- алгоритме претраживања

  а) алгоритме засноване на неинформисаном претраживању

  - претраживање по ширини (енг. *breadth-first*)
  - претраживање по дубини (енг. *depth-first*)
  - претраживање по дубини са итеративним продубљивањем (енг. *depth-first with iterative deepening*)

  б) алгоритме претраживања засноване на хеуристичким функцијама и информисаном претраживању

  - алгоритам планинарења (енг. *hill climbing*)
  - алгоритам прво најбољи (енг. *best first*)
  - алгоритам гранања и ограничавања (енг. *branch and bound*)
  - алгоритам A* (енг. *A star*)

в) алгоритме теорије игара

- алгоритам *Minimax*
- алгоритам *Minimax* са алфа-бета одсецањем (енг. *alpha-beta pruning*)

- алгоритме резоновања и репрезентације знања

  а) доказивање теоријом резолуције (енг. *resolution theorem proving*)

  б) продукционе системе (енг. *production system*)

  в) семантичке мреже и оквире (енг. *semantic networks and frames*)

  г) Бајесове мреже (енг. *Bayesian networks*)

- алгоритме за решавање проблема
  - системе планирања (енг. *planning systems*)
- алгоритме за рад у неизвесном окружењу
  - репрезентације за резоновање у неизвесном окружењу (енг. *representations for reasoning under uncertaintly*)
  - расплинуту логику (енг. *fuzzy logic*)
- алгоритме машинског учења
  - индуктивне алгоритме (енг. *inductive systems*)

2.2. Алгоритми претраживања

Алгоритми претраживања представљају начин решавања проблема до кога би се тешко дошло применом метода класичног програмирања [19]. До решења таквих проблема долази се претраживањем простора стања проблема. Циљ ових алгоритама је да се полазећи од почетног стања, низом прелаза између стања, дође до завршног стања, које представља циљ, односно решење проблема.

2.2.1. Стандардни алгоритми претраживања

Код имплементације стандардних алгоритама претраживања, од интереса је било претраживање графова. Графови представљају апстрактан начин представљања скупа објеката, при чему су одређени парови објеката спојени релацијама. Објекти се представљају помоћу чворова, а релације између парова чворова представљају гране графа. Гране могу

бити усмерене или неусмерене, чиме се означава да ли је веза између посматраних чворова асиметрична или симетрична. Ако су све гране истог типа, онда се цео граф означава као усмерен или неусмерен граф. Већина алгоритама гарантује налажење неке путање кроз граф, док неки алгоритми проналазе оптималну путању.

Постоји велики број алгоритама претраживања графова, али су у оквиру ове дисертације разматрани следећи:

- претраживање по ширини,
- претраживање по дубини,
- алгоритам планинарења,
- алгоритам прво најбољи,
- алгоритам гранања и ограничавања,
- алгоритам A*.

Због адекватнијег приказа рада стандардних алгоритама претраживања и лакшег праћења извршавања корака, често се користе стабла претраге. Чворови стабла одговарају чворовима графа, при чему једном чвору графа у општем случају одговара више чворова у стаблу. Стабло се конструише на следећи начин:

1) Почетни чвор графа се приказује као корен стабла претраге. При развијању почетног чвора у стабло се уносе сви следбеници (синови) кореног чвора и одговарајуће гране до њих. Тиме је корен стабла означен као посећен чвор и прелази се на следећи корак конструкције.
2) Бира се један од непосећених чворова у зависности од одабраног стандардног алгоритма. Уколико је реч о циљном чвору, то означава да смо дошли до краја рада алгоритма. У супротном се развија одговарајући чвор, а у стабло се уносе чворови који одговарају сваком од суседа развијеног чвора у графу, уколико се посматрани суседни чвор већ није појавио на путањи од корена до развијеног чвора. Процес се понавља све док се не дође до циља или док сви чворови у стаблу претраживања не буду посећени. Комплетно стабло претраживања представља стабло код кога су сви чворови развијени.

У области алгоритама претраживања често се дефинишу хеуристичке функције. Основни проблем у претраживању је смањивање простора стања. Коришћење хеуристичких функција омогућава елиминисање грана из процеса претраживања, избор опште путање коју треба следити, као и избор следећег чвора који треба да се обиђе. Хеуристичка функција треба да процени појединачна стања проблема и да одреди колико су она повољна. Добро изабрана хеуристичка функција чини процес претраживања ефикаснијим, али у реалним применама цена израчунавања хеуристичке функције често може да буде већа него уштеда у времену претраживања.

Код стандардних алгоритама претраживања може се применити и принцип динамичког програмирања (енг. *dynamic programming*). То је метод оптимизације решавања комплексних проблема који се заснива на њиховом рашчлањивању на мање потпроблеме. При томе се користи најчешће рекурзивни приступ, а за проблеме који се могу рашчланити рекурзијом каже се да имају оптималну структуру. Примена динамичког програмирања значајно доводи до уштеде у времену рада алгоритма.

Коришћење динамичког програмирања код алгоритама претраживања засновано је на одржавању листе већ посећених чворова, чиме се омогућава да се раније посећени чворови не посећују поново. На тај начин из комплексног проблема налажења путање између задатих чворова графа, уклања се потпроблем скупа оних путања за које се зна да не би ништа допринеле решавању главног проблема. Такође, у случају постојања више путања до истог чвора, све путање осим оне са најмањом ценом коштања се одстрањују из разматрања.

2.2.2. Алгоритми теорије игара

Теорија игара најчешће се користи код потезних игара са два играча. Свака потезна игра може се реализовати у виду стабла. Чворови стабла су позиције у игри, а гране представљају могуће потезе који преводе једну позицију игре у другу. Корен стабла представља почетно стање игре, а листови представљају терминална стања, односно победу, нерешен исход или пораз. Број грана који излази из сваког чвора једнак је броју могућих потеза у том стању игре и назива се фактор гранања. Фактор гранања је обично добар индикатор колико је игра компликована.

Најпознатији алгоритми теорије игара су *Minimax* и *Minimax* са алфа-бета одсецањем [19].

2.2.2.1. Алгоритам Minimax

Алгоритам *Minimax* проналази најбољи потез за задату почетну позицију потезне игре са два играча. Алгоритам наизменично бира најбољи потез првог играча, а затим најбољи потез другог противничког играча, и тако докле год има могућих потеза. Корен стабла припада нивоу један, његови следбеници припадају нивоу два, и тако даље. Све чворове на непарним нивоима називамо МАКС чворовима, а све чворове на парним нивоима називамо МИН чворовима. Одлика овог алгоритма је да размишља унапред. Он користи статичку функцију процене која се исказује бројем поена из задатог опсега за дату игру. Код терминалне позиције у стаблу, рачунање је тривијално, јер су унапред задате и минимална и максимална вредност опсега и те вредности се додељују у случају победе или пораза. За случај нерешеног исхода додељује се средња вредност одабраног опсега. Уколико се играч налази у средини партије, потребно је дефинисати процену позиције у игри и тада се користи хеуристичка функција.

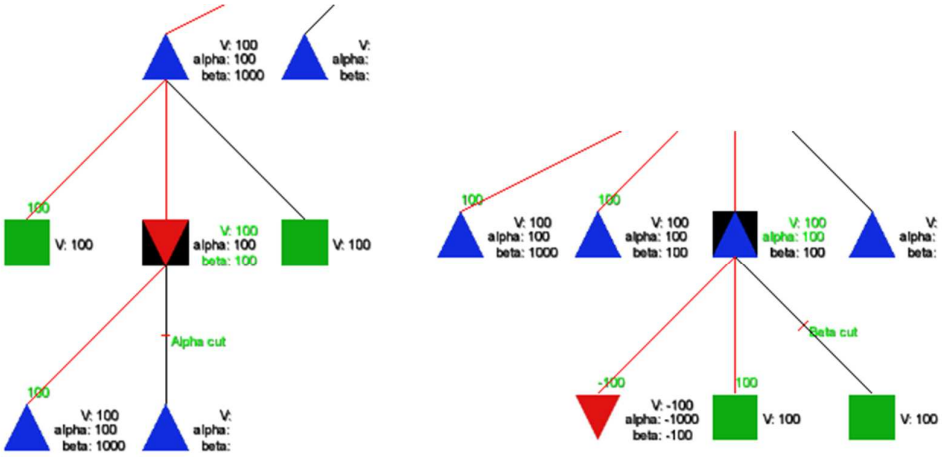
Алгоритам *Minimax* је реализован проласком кроз стабло игре алгоритмом претраживања по дубини. Овај алгоритам се најчешће реализује коришћењем рекурзивне реализације, али је могуће реализовати га и коришћењем итеративног приступа. На пример, итеративна реализација је боља када је потребно да се заузме мање радне меморије и да се постигне боља ефикасност извршавања алгоритма.

Алгоритам има просторну завиност $O(d)$, где је d дубина претраживања. Ако је n максималан број потеза који следују почетни чвор, онда је укупна просторна зависност $O(n*d)$. Временска зависност самог *Minimax* алгоритма је $O(n^d)$ и за сложеније игре то није прихватљиво. Из тог разлога се раде оптимизације, па алфа-бета одсецањем ту зависност можемо да смањимо на $O(n^{d/2})$. Ни то не може да се сматра као прихватљива сложеност, па се најчешће код ових алгоритама уводе максимална дубина претраживања и хеуристика. Са друге стране, да би алгоритам био ефикасан, потребно је да дубина стабла буде што већа, а то доводи до огромног броја чворова.

2.2.2.2. Алфа-бета одсецање

Метод алфа-бета одсецања представља оптимизацију која ограничава алгоритам са доњом и горњом границом вредности процене. Алфа вредност представља вредност најповољнијег потеза који је до сада пронађен, односно означава доњу границу процене потеза која је прихватљива. Све потезе, који су лошији од најбољег до тада пронађеног, нећемо одиграти, и самим тим неће бити разматрани сви потези на које нас противник

натера, а лошији су од нашег до тада пронађеног потеза, а тај део стабла коме припадају ти потези ће бити одсечен. Бета вредност представља вредност која можда може да се достигне, али не може да се достигне више од те горње границе, јер то противник неће дозволити. Део стабла где постоје вредности могућег потеза, које су веће од бета вредности, такође се одсеца. Примери алфа и бета одсецања дати су на слици 1.



Слика 1 – Примери алфа и бета одсецања

Са слике 1 се види да су сваком чвору стања придружене вредности *V*, алфа и бета. МАКС чворови су представљени плавим троуглом усмереним на горе, а МИН чворови су представљени црвеним троуглом усмереним на доле. Рад алгоритма формира корени чвор стабла, што је почетно стање игре и додељује том чвору вредности $V = 1000$, ако је у питању МИН чвор (односно -1000 ако је у питању МАКС чвор), вредност алфа = -1000 и вредност бета = 1000. Ове вредности су замена за вредности $+\infty$ и $-\infty$, које се користе у поставци овог алгоритма. У софтверској имплементацији је могуће да евентуално поставимо уместо ових вредности највећи, односно најмањи цео број. Процена вредности позиције не сме бити мања, односно не сме бити већа од тих додељених вредности.

У сваком кораку алгоритма ради се експанзија текућег стања, осим када је текуће стање терминални чвор или су сви чворови који следе из датог чвора већ обиђени. Тада се врши повратак на чвор изнад, грана се обележава завршеном, а вредности *V*, алфа и бета пропагирају:

- ако је МАКС чвор изнад:
    - *V = MAX (stara vrednost V, propagirana vrednost V)*

- *alfa = max (stara vrednost Alfa, propagirana vrednost Alfa)*

- ако је МИН чвор изнад:
  - *V = MIN (stara vrednost V, propagirana vrednost V)*
  - *beta = MIN (stara vrednost Beta, propagirana vrednost Beta)*

2.3. Алгоритми резоновања и репрезентације знања

Алгоритми резоновања и репрезентације знања приказани су алгоритмима формалне логике, продукционим системима, семантичким мрежама, оквирима и Бајесовим мрежама.

2.3.1. Формална логика

Четири кључна алгоритма, који се користе у закључивању формалне логике првог реда су:

- Налажење конјуктивне нормалне форме (КНФ, енг. *CNF*)
- Закључивање алгоритмом резолуције коришћењем стратегије скупа подршке (енг. *set of support*)
- Закључивање алгоритмом резолуције коришћењем стратегије првенства јединице (енг. *unit preference*)
- Закључивање алгоритмом резолуције коришћењем стратегије избора по ширини (енг. *breadth-first strategy*)

2.3.1.1. Алгоритам конјуктивне нормалне форме

Помоћу еквивалентних трансформација, свака формула може бити трансформисана у КНФ. Алгоритам конјуктивне нормалне форме изгледа овако [19]:

Улаз: Исказна формула F

Излаз: КНФ формуле F

Кораци КНФ алгоритма:

1. Елиминисање импликације:

   $E1 \Rightarrow E2$ трансформише се у $\neg E1 \vee E2$

2. Спуштање негација до атомских формула:

$\neg(\mathrm{E1} \wedge \mathrm{E2})$ трансформише се у $\neg\mathrm{E1} \vee \neg\mathrm{E2}$

$\neg(\mathrm{E1} \vee \mathrm{E2})$ трансформише се у $\neg\mathrm{E1} \wedge \neg\mathrm{E2}$

$\neg\neg\mathrm{E1}$ трансформише се у $\mathrm{E1}$

$\neg\forall x(\mathrm{E1}(x))$ трансформише се у $\exists x(\neg\mathrm{E1}(x))$

$\neg\exists(\mathrm{E1}(x))$ трансформише се у $\forall x(\neg\mathrm{E1}(x))$

3. Уклањање егзистенцијалних квантификатора:

   Ако се посматра израз $\forall x\ \exists y\ (\mathrm{E1}(x,y) \wedge \mathrm{E2}(y))$, односно да за сваку вредност *x*, увек може да се нађе нека вредност *y*, тако да постоји пресликавање F, тада можемо посматрани израз заменити следећим изразом: $\mathrm{E1}(x, F(x)) \wedge \mathrm{E2}(F(x))$. Функције уведене ради замене егзистенцијалних квантификатора зову се Сколемове функције.

4. Преименовање променљивих тако да сваком квантификатору одговара посебна променљива.

5. Премештање свих универзалних квантификатора на леву страну без промене њиховог редоследа.

6. Спуштање дисјункција до најнижег нивоа, према закону дистрибуције $\vee$ у односу на $\wedge$: $(\mathrm{E1} \wedge \mathrm{E2}) \vee \mathrm{E3}$ трансформише се у $(\mathrm{E1} \vee \mathrm{E3}) \wedge (\mathrm{E2} \vee \mathrm{E3})$

7. Елиминација конјункције, тако да сваки члан пишемо као засебну формулу.

8. Преименовање променљивих тако да не постоји иста променљива у различитим формулама.

9. Уклањање квантификатора.

У зависности од сложености формуле, неки кораци трансформације одређене формуле у КНФ се могу прескочити или објединити.

2.3.1.2. Алгоритам резолуције

Поступак доказивања теореме користећи резолуцију гласи [19]:

- Негирати теорему коју треба доказати и додати је на листу аксиома.
- Довести листу аксиома у КНФ.

- Док се не добије празна клаузула, или док се не добије пар клаузула које се не могу разложити, извршити разлагање клаузула и додавати резултат на листу клаузула.
- Ако се добије празна клаузула, обавестити да је теорема истинита, а у супротном да је неистинита.

Када треба одредити да ли два литерала у клаузулама које се разлажу могу да се учине идентичним, односно да се пониште, користи се процес унификације. Процедура унификације гласи:

- Представити оба литерала која се унифицирају листама у којима је предикатски симбол први елемент и иза којег следе аргументи тачно по редоследу.
- Напустити процедуру ако две листе нису исте дужине.
- Поредити елементе који се налазе на истим позицијама у обе листе:
  - Предикатски и функцијски симболи, као и константе морају бити идентични.
  - За промењљиве треба да се изведе упаривање путем замене; када се наиђе на промењљиву, замењује се, осим уколико замењени израз не садржи исту промењљиву, да не би дошло до бесконачне петље замена.
- Два предиката су унифицирана ако се сви елементи поклапају после замене промењљивих.

У сваком кораку алгоритма резолуције може се применити одређена стратегија за избор два става која се упарују. Код стратегије скупа подршке за спајање се увек прво бирају они ставови који представљају или негацију тврђења или ставове изведене из негације тврђења. Код стратегије првенства јединице приоритет при избору даје се ставовима са најмањим бројем чланова. Стратегија избора по ширини разматра све могуће комбинације постојећих ставова, пре него што се пређе на разматрање новодобијених ставова.

2.3.2. Продукциони системи

Продукциони системи представљају најчешћи начин представљања знања у експертским системима. Продукционе системе чине: радна меморија, продукциона меморија и интерпретер, који спроводи одговарајући алгоритам закључивања [19].

Радна меморија се састоји од низа меморијских елемената, а сваки елемент представља стање неког објекта од интереса. Меморијски елементи се називају другачије чињенице. У општем случају чињеница је уређена *n*-торка симбола, где први симбол представља име или идентификатор објекта, док преостали симболи представљају парове атрибут-вредност који описују неко својство тог објекта. У овом систему, чињенице се састоје од три симбола: идентификатора, атрибута и вредности.

Продукциона меморија се састоји од низа појединачних правила. Свако правило се састоји од услова и последице. Услов чине један или више условних елемената који представљају образац симбола (енг. *pattern*). Условни елемент може да се посматра као уређена *n*-торка у којој први члан представља идентификатор неког појма, док остали чланови представљају његове атрибуте и вредности. Код условних елемената сваки од *n* чланова може бити нека константа или променљива, без обзира да ли тај елемент представља идентификатор, атрибут или вредност. У реализованом софтверском систему условни елементи садрже три елемента.

Интерпретер је део система задужен за упаривање условних елемената правила са постојећим чињеницама, давање приоритета окинутих правила, као и додавање и уклањање чињеница из радне меморије и извршавање осталих акција. Интерпретер садржи више мањих функционалних целина задужених за различите делове имплементације алгоритма закључивања као што су упаривање (енг. *Pattern matcher*) и резолуција конфликта.

Постоји више различитих алгоритама закључивања, који се могу сврстати у следеће групе [19]:

- алгоритми директног уланчавања или закључивање вођено подацима (енг. *Data driven*);
- алгоритми повратног уланчавања или циљно-усмерено закључивање (енг. *Goal driven*);
- алгоритми хибридног уланчавања.

Основна разлика између алгоритама директног и повратног уланчавања је у начину добијања нових чињеница. Код закључивања вођеног подацима са сваким додавањем чињеница у радну меморију, проверава се да ли постоје окинута правила. У случају да постоје, након фазе резолуције конфликта, правила се извршавају услед чега најчешће следи

убацивање нових чињеница у радну меморију. Након тога се поступак итеративно понавља, све док систем не уђе у стационарно стање, односно докле год се додају нове чињенице. Код алгоритма повратног уланчавања, поставља се упит који алгоритам даље рекурзивно дели на подупите, док се не добије истинита вредност почетног упита.

Хибридно уланчавање представља комбинацију повратног и директног уланчавања. Код овог алгоритма се циклично проверава свако правило, као упит о чињеницама, док се у случају задовољености услова неког правила додају нове чињенице. Упити се врше све док се добијају нове чињенице, слично као код директно уланчаних алгоритама.

Рете алгоритам је алгоритам директног уланчавања, који за разлику од класичног алгоритма директног уланчавања има константно време извршавања. Основна предност овог алгоритма је што приликом учитавања правила у продукциону меморију формира одговарајућу структуру података која смањује број поређења.

2.3.3. Семантичке мреже и оквири

Семантичке мреже и оквири представљају графичке презентације знања. Семантичке мреже описују елемент и везе између њих. Објекти се означавају помоћу правоугаоника и кругова, а везе као усмерене линије са стрелицама. Постоји доста различитих веза између елемената код семантичких мрежа, а највише се користе: *IS_A*, *A_PART_OF*, *INSTANCE_OF* и *PROTOTYPE_OF* [20].

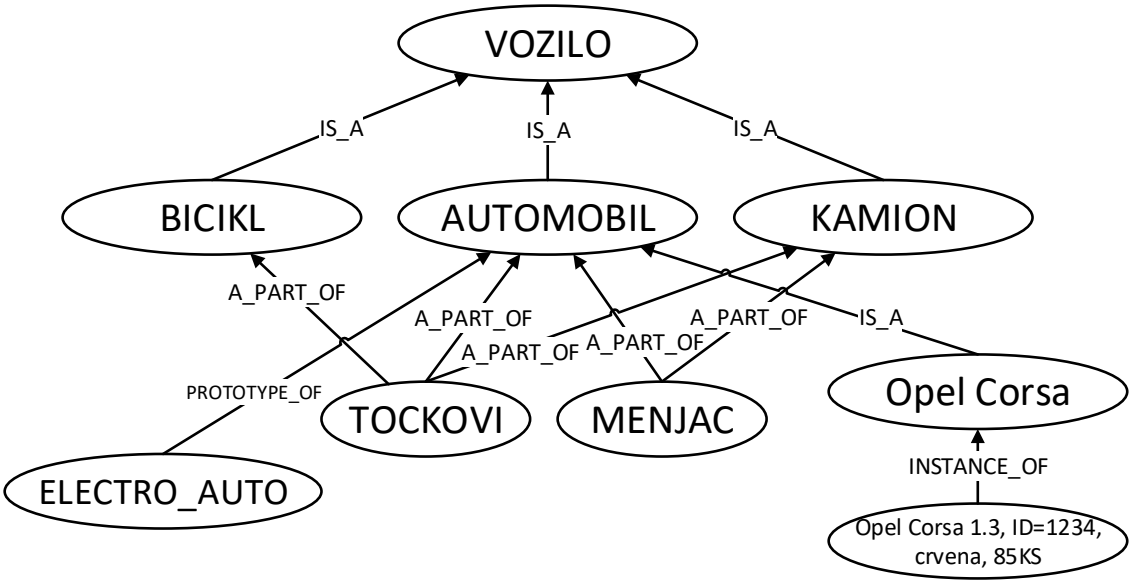
Веза *IS_A* се користи да означи припадност неког елемента класи елемената. Чвор који представља специфичан пример класе назива се примерак, односно инстанца. Пример ове везе је: делфин је (*IS_A*) водени сисар.

Веза *A_PART_OF* се користи да означи да је неки елемент физички део неког другог елемента. Преградак чвора чине везе различитих имена. Пример ове везе је: Волан је део (*A_PART_OF*) аутомобила.

Веза *INSTANCE_OF* приказује да један објекат може представљати примерак другог објекта. Пример ове везе је: Голф је инстанца (*INSTANCE_OF*) путничког аутомобила.

Веза *PROTOTYPE_OF* означава да један објекат представља специјални случај или модел другог објекта. Пример ове везе је: Паметни телефон је прототип (*PROTOTYPE_OF*) мобилног телефона.

Концепт класе и веза *IS_A* могу се искористити за представљање неке ситуације и догађаја. Процес наслеђивање омогућава померање описа са класе на инстанце, односно дефинише начин да један објекат наследи особине другог објекта. Пример семантичких мрежа дат је на слици 2.



Слика 2 – Пример семантичких мрежа

Преградак не садржи увек вредности, већ се често те вредности могу израчунати користећи постојеће информације. То се постиже процедурама које се називају *if-needed* и оне се користе само онда када је то потребно, а не онда када се оне позову по имену.

Предности употребе семантичких мрежа у области представљања знања су:

- једноставан и прецизан графички приказ;
- укључивање већег броја различитих типова објеката у оквиру мреже;
- могу да се користе као средство комуникације између инжењера знања и експерата током фазе прикупљања знања.

Недостаци употребе семантичких мрежа су:

- велика сложеност и мала поузданост презентације знања, јер наслеђивање постаје процес претраживања;
- могућност комбинаторне експлозије због великог броја могућих врста веза и начина како се могу комбиновати везе помоћу индиректних форми повезивања;

- комплексна презентација која може тражити захтевније операције претраживања да би се дошло до одређеног закључка.

Оквири су начин представљања знања којим се организује представљање истинитих ствари за неку општу класу елемената [21]. Оквири су скуп семантичких чворова и преградака који описују заједно стереотипске објекте, догађаје и акције. Они су једноставнији од семантичких мрежа, јер се код њих примењују само *IS_A* везе. Код оквира је нагласак на имплицитном знању. Семантичке мреже је могуће трансформисати у структуру оквира [22]. Пример оквира дат је на слици 3.

```
Frame AUTOMOBIL
IS_A MotornoVozilo
PROIZVODJAC
SNAGA_MOTORA
PUTUJE_DRUMOM true
BROJ_TOCKOVA Default = 4
TIP opseg: limuzina, karavan, kupe, dzip
KAPACITET_REZERVOARA_GORIVA
TRENUTNO_GORIVA
MOGUCA_KILOMETRAZA if_needed: racunati na osnovu snage motora
i trenutne kolicine goriva
```

Слика 3 – Пример оквира

Постоје три различита типа оквира:

1) Оквири у којима именовани преградак има попуњене стандардне вредности одређених података. На пример уколико оквир који представља аутомобил има у прегратку за број точкова подразумевану вредност 4. Ова вредност се може променити, уколико се код неког специфичног возила не поклапа та подразумевана вредност, већ аутомобил има 3 точка. Такође, могуће је задати и опсег вредности. На пример, преградак који означава величину има вредности мала, средња и велика.

2) Оквири у којима преградак представља везу *IS_A*. На пример, ако је аутомобил моторно возило и има преградак *IS_A*, оквир аутомобил може бити повезан са оквиром који описује основне особине моторног возила.

3) Оквири у којима преградак садржи само процедурални програмски код. На пример број километара које тренутно аутомобил може да пређе, на основу количине бензина у резервоару и на основу снаге мотора. Преградак који представља овај

резултат у неком опсегу може садржати процедурални код базиран на прегратку за количину бензина и прегратку за снагу мотора.

Оквири који представљају класу објеката на датом нивоу апстракције могу наслеђивати прегратке и вредности које се налазе на вишем нивоу. Процедуралне информације које се могу уградити су: *if-needed* и *if-addes*. Процедура *if-needed* извршава се када треба да се израчуна вредност прегратка, а процедура *if-addes* се извршава када је вредност прегратка већ додељена.

Предности употребе оквира у области представљања знања су:

- Принцип наслеђивања, када неки оквири могу да наследе атрибуте од неког другог родитељског оквира.
- Информације могу једноставно да се читају и користе, јер се оквири најчешће представљају табеларно.
- Могућност чувања подразумеваних вредности, када један оквир има преградак са *IS_A*, односно када је повезан са другим оквиром који садржи основне информације за одређену групу оквира.
- Употреба подразумеваних вредности у процесу резоновања. Систем може да преправи подразумевану вредност и понови поступак резоновања, уколико се утврди да је подразумевана вредност некоректна.
- Хијерархијска структура и редукована сложеност.
- Коришћење стандардних формата и синтаксе, чиме се реализује чврста структура документа.
- Јединствена презентација знања комбиновањем процедуралног и декларативног знања.
- Коришћење дозвољених вредности или дозвољен унос вредности у одређеном предефинисаном опсегу.

Недостаци употребе оквира су:

- Могу да буду неефикасни у време извршавања, јер не омогућавају најефикаснији метод чувања података.

- Могу довести до превелике употребе процедура, када се тежи већој реализацији нових процедура, него што се проверава тренутна структура и садржај тренутних оквира.

2.3.4. Бајесове мреже

Бајесове мреже засноване на Бајесовом учењу такође спадају у групу алгоритама одлучивања и репрезентација знања које имају широку примену у различитим научним областима. Оне користе узрочно-последичне везе између догађаја и дају детаљни начин решења проблема. Најчешће се користе у медицинској дијагностици, јер различите врсте болести прати одређени број могућих симптома. Бајесове мреже и Бајесов начин размишљања се такође користе за класификацију научних радова. Циљ је да се постигне класификација докумената према темама којима припадају или према кључним речима који су везане за одређене теме. Сваки систем заснован на Бајесовом начину резоновања неопходно је да има прецизно имплементиран рачун вероватноће.

Вероватноћом представљамо неодређеност и неизвесност догађаја. Вероватноћа догађаја *A*, у ознаци *P(A)*, представља степен веровања да ће се тај догађај десити. Условна вероватноћа представља вероватноћу да ће се неки догађај десити под условом да се неки други догађај десио. Условна вероватноћа означена као $P(A|B)$ означава да се догађај *A* десио под условом да се догађај *B* десио.

Бајесова теорема омогућава нам да израчунамо апостериорне вероватноће и ажурирамо веровања у догађаје. Она показује да ако имамо узајамно искључиве хипотезе H_1, H_2,...H_n у односу на неки догађај *E* и ако је вероватноћа $P(E) \neq 0$, веза између вероватноће хипотезе пре него што добије доказе *P(H)* и вероватноће хипотезе након што добије доказе $P(H|E)$ је:

$$P(H_i|E) = \frac{P(E|H_i)}{P(E)} \cdot P(H_i)$$

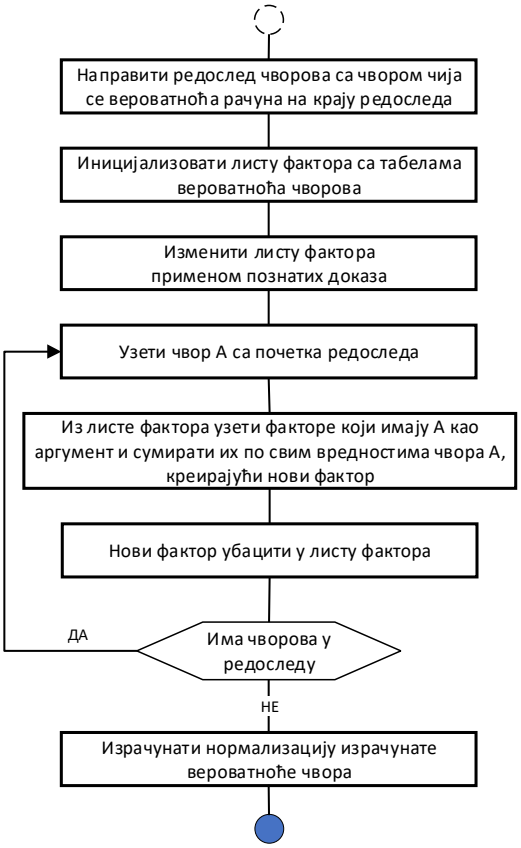
за свако $i = 1, ..., n$.

Бајесове мреже се представљају усмереним ацикличним графовима, јер је њихова структура погодна за комбиновање почетног знања о вероватноћама догађаја и каснијих доказа о томе да ли се догађај десио. Чворови графа се представљају скупом случајно променљивих. Свака случајна променљива репрезентује неки неизвесни догађај који се посматра, а скуп стања случајно променљивих представља њен домен. Случајно променљиве

могу бити дискретне или континуалне. Највећи број алгоритама за рад са Бајесовим мрежама претежно је оријентисан на руковање са дискретним стањима променљивих. Разлог је што континуалну случајну променљиву могуће дискретизовати.

Закључивање у Бајесовим мрежама тежи да израчуна вероватноће чворова, уз дате доказе за вредности неких чворова. Брзина закључивања може бити врло компликована уколико је мрежа комплексна, тако да сваки пар чворова у графу буде повезан највише једним путем. У том случају се посматра цела мрежа и користи се алгоритам размене порука, тако да до сваког чвора графа долазе параметри за ажурирање од његових чворова родитеља и чворова деце. Код мреже чија структура представља једноставан ланац чворова, закључивање се своди на једноставну примену Бајесове теореме.

Алгоритам елиминације чворова и спајања више табела вероватноћа чворова у једну је бољи, јер прави листу чворова које треба да елиминишемо. У току извршавања овог алгоритма чува се листа фактора која зависи од табела вероватноћа чворова и њиховог редоследа елиминације. Алгоритам елиминације чворова дат је дијаграмом на слици 4.



Слика 4 - Дијаграм алгоритма елиминације чворова

2.4. Алгоритми за решавање проблема

GPS алгоритам представља један од првих система за универзално решавање проблема и имплементиран је у програмском језику трећег реда под називом *IPL* [23]. Овај алгоритам је у области вештачке интелигенције многим алгоритмима који су касније развијани и који се данас примењују, послужио као основа и инспирација. Иако је овај алгоритам успешно решио једноставне проблеме, попут Ханојских кула, он није могао решити никакав проблем у реалном свету, јер његова претрага лако долази до комбинаторне експлозије.

На Универзитету Стенфорд је седамдесетих година прошлог века развијен *STRIPS* алгоритам, од стране истраживача Ричарда Фикеса и Нилса Нилсона [24]. Модификације овог алгоритма данас се примењују у многим системима. На пример, у развоју компјутерске игре *F.E.A.R.* где је примењена вештачка интелигенција, коришћена је једна модификација *STRIPS* алгоритма, обухватајући и алгоритам *A** и машину са коначним бројем стања *FSM* [25]. У неким системима који користе *STRIPS* алгоритам, планирање је могуће (полиномијално) због увођења рестрикција помоћу формула, предуслова и постуслова, а у неким системима планирање није могуће и представља *NP*-тежак проблем [26].

2.4.1. GPS алгоритам

Имплементација *GPS* алгоритма изгледа овако:

1. Формира се листа која ће иницијално да садржи само текуће стање.
2. Стање на почетку листе назива се текућим стањем. Док се листа не испразни или док се не достигне циљно стање, извршавају се следећи кораци:
   2.1. Ако се прегледом табеле разлика закључи да не постоји оператор који није употребљен за смањивање разлике између текућег и циљног стања, уклонити текуће стање из листе.

   У супротном:

   2.2. Изабрати оператор за смањивање разлике из табеле разлика.

      2.2.1. Ако предуслови за примену оператора нису задовољени, покушати њихово задовољавање формирањем новог циљног стања од тих предуслова и рекурзивним позивом *GPS* алгоритма, ради достизања новог циљног стања из текућег.

2.2.2. Ако су предуслови задовољени применити оператор и новодобијено стање ставити на почетак листе стања.

3. Ако се достигне циљно стање, исписати поруку о успеху, у супротном, исписати поруку о неуспеху.

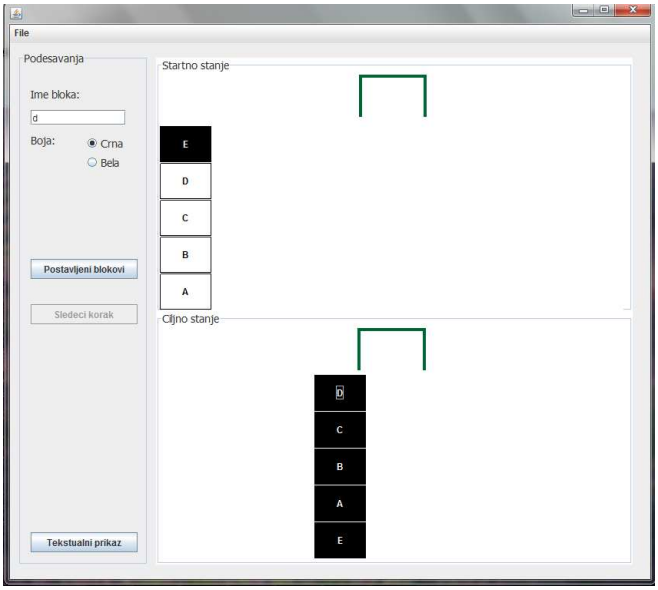
2.4.2. STRIPS алгоритам

Научници са Универзитета Стенфорд су седамдесетих година прошлог века добили задатак да развију алгоритам који би управљао роботом и који би радио на врло ограниченим хардверским ресурсима тог времена [24]. Циљ је био да се робот креће у окружењу са више просторија и да премешта кутије са једног места на друго, уз истовремено укључивање или искључивање сијалица у просторијама у којима се налази. Такође, омогућено је да се робот попне на кутију уколико је прекидач за светло високо на зиду. Окружење је било идеализовано, са једноставним сензорима који су тада постојали.

Алгоритам *STRIPS* припада класи система планирања унутар стратегија за решавање проблема. Планирање представља процес одлучивања о акцијама и редоследу акција пре њиховог извршавања. Након симулације сваке акције посматра се њен ефекат. Акције које се не могу кориговати у оквиру реалног окружења се одбацују из симулационог модела.

Процес планирања повећава ефикасност у доменима у којима се грешке могу занемарити или кориговати, али не мења ефикасност тамо где су грешке непроменљиве. Системи планирања су засновани на декомпозицији, па уколико је проблем сложен, решава се један проблем, а затим други. Уколико постоји велика интеракција између проблема, тада се јављају проблеми, па се планери ослањају најчешће на динамичко решавање интеракција између потпроблема.

Алгоритам *STRIPS* је базиран на предикатској логици и одржава модел света, односно скуп израза који заједно показују тренутну ситуацију. За сваки модел се претпоставља да за њега постоји скуп употребљивих оператора такав да применом сваког од њих може да се трансформише тренутни модел у неки други модел света. Задатак овог алгоритма није искључиво доказивање да је неки став могућ, већ и да уз помоћ неке секвенце оператора од почетног стања и иницијалног модела света може да се пређе у циљно стање, односно тражени модел света. Овај алгоритам се најчешће приказује као пример „Свет блокова“ приказан на слици 5.



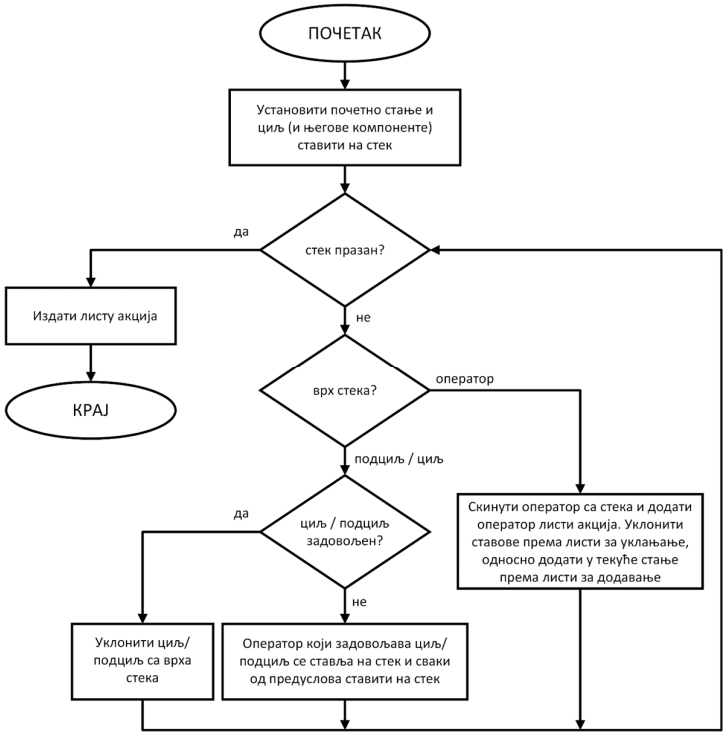
Слика 5 – *STRIPS* алгоритам на примеру анимације „Свет блокова“

Основни ставови промене стања задају се уз помоћ три листе:

1) предуслови, односно листа ставова који морају бити испуњени у текућем стању да би оператор могао да се примени;
2) листа за уклањање, односно листа свих ставова који се уклањају из текућег стања у тренутку примене оператора;
3) листа за додавање, односно листа свих ставова који се додају текућем стању у тренутку примене оператора.

У сваком кораку алгоритам *STRIPS* користи и памти текуће стање проблема, циљни стек и скуп оператора који се користе за промене стања. Иницијално је текуће стање почетно стање проблема, а ажурира се применом неког оператора. Циљни стек садржи ставове који одговарају тренутном циљу или подциљу у току рада алгоритма, као и операторе који се примењују.

Дијаграм тока алгоритма *STRIPS* приказан је на слици 6.



Слика 6 – Дијаграм извршавања алгоритма *STRIPS*

Осим примене у роботици, овај алгоритам се може користити у теорији игара, за решавање различитих математичких проблема, код доказивања теорема, затим у индустријским постројењима и фабрикама са аутоматском контролом.

2.4.3. Метод задовољења ограничења

Проблеми са задовољењем ограничења (енг. *Constraint Satisfaction Problems*, *CSP*) су математички проблеми који се састоје од променљивих, домена вредности које те променљиве могу добити и ограничења која се морају испунити [27]. Ови проблеми често имају велику сложеност и захтевају комбинацију хеуристика и комбинаторних метода претраживања да би проблем решили у разумном времену. Проблем задовољења услова (енг. *Boolean satisfiability problem*, *SAT*) [28], задовољење теорије модула (енг. *satisfiability modulo theories*, *SMT*) [29] и програмирање одговора (енг. *answer set programming*, *ASP*) [30], могу се посматрати као одређени облици проблема са задовољењем ограничења.

Проблеми са задовољењем ограничења се најчешће представљају коришћењем графа ограничења и метода претраживања. Многи важни проблеми у реалном свету могу се описати као проблеми задовољења ограничења, као што су: проблем 8 или *N* краљица (енг. *eight / N queens puzzle*), проблем бојења мапа (енг. *map coloring problem*), Судоку, проблеми

са укрштеним речима (енг. *crosswords*), Футошики (енг. *more or less*), Какуро (енг. *cross sums*), Нумбрикс, Хидато (Хидоку) и многи други логички проблеми. Неки проблеми су тежи, па се не могу приказати као једноставни системи. Такви сложенији системи су системи за аутоматско планирање, разумевање реченица и многи други.

Приликом решавања *CSP* проблема примењујемо најчешће претраживање уназад (енг. *backtracking search*), односно један начин алгоритма претраживања по дубини. Претраживање уназад је рекурзивни алгоритам. Извршавањем овог претраживања значи да се треба вратити путањом уназад до прве променљиве која је имала више легалних вредности, затим да се одабере нова вредност те променљиве и онда кренути новом путањом ка могућем решењу [19]. Конзистентност се постиже као задовољење свих ограничења која су дефинисана у променљивама.

Један од начина побољшања овог метода претраживања зове се провера унапред (енг. *forward checking*). Када се некој променљивој *X* додели вредност, треба проверити све променљиве којима није додељена вредност, а која је повезана ограничењима са променљивом *X*. Затим се бришу све вредности из домена променљиве *Y* које нису конзистентне са вредношћу коју смо додели променљивој *X*.

Пропагације ограничења (енг. *constraint propagation*) су методе које се користе за модификовање проблема задовољења ограничења. То су методе које спроводе облик локалне конзистентности са циљем да неки проблем претворе у еквивалентан проблем, који је обично једноставнији за решавање. Најпознатији облици локалне конзистентности су: конзистентност лука (енг. *arc consistency*), конзистентност хипер-лука (енг. *hyper-arc consistency*) и конзистентност путање (енг. *path consistency*).

Конзистентност лука обезбеђује брзи метод пропагације ограничења који је знатно бољи од провере унапред. Конзистентност лука је провера која може бити извршена као претпроцес пре почетка претраге или као пропагацијски корак, након сваке доделе вредности током претраживања. У оба случаја процес мора бити поновљен више пута све док не буде постојала ниједна неконзистентност. Кад год избришемо неку вредност из домена неке променљиве да би отклонили неку неконзистентност, нова неконзистентност се може појавити у гранама које иду ка тој променљивој. Процес се мора понављати све док се не отклоне све неконзистентности. Најпознатији алгоритам који примењује конзистентност лука је *AC-3* алгоритам.

Осим претраживања уназад и пропагације ограничења, постоје и методе локалне претраге који су непотпуни алгоритми. Такви алгоритми могу да нађу решење проблема, али могу и да га не нађу, чак и у случају да је проблем решив.

2.5. Алгоритми за рад у неизвесном окружењу

Постоје ситуације када систем мора да ради у неизвесном окружењу. Могућа решења таквих проблема су резоновање на основу фактора извесности, расплинута логика и системи за одржавање истинитости.

2.5.1. Резоновање на основу фактора извесности

Резоновање на основу модела фактора извесности први пут је било примењено у оквиру експертског система *MYCIN*. То је експертски систем који је кренуо да се развија 1972. на Универзитету Стенфорд, са циљем да покуша да изврши дијагнозу пацијента на основу пријављених симптома и резултата медицинских испитивања [31].

Фактор извесности (енг. *certainity factor*, *CF*) представља рационалан број који припада интервалу $-1 \leq CF \leq 1$. Помоћу овог фактора се квантификује степен поверења у неки закључак на основу датог скупа догађаја, појава, чињеница, опажања. Субјективне вероватноће представљене су као $MB(h,e)$ и $MD(h,e)$ [32]. *MB* представља меру поверења, односно нотацијом $MB(h,e)$ се означава мера увећаног поверења у хипотезу h, засновану на чињеници e која је тачна. Слично томе, *MD* представља меру неповерења. *CF* означава број који спаја мере *MB* и *MD* и израчунава се по формули: $CF(h,e) = MB(h,e) - MD(h,e)$.

Сваком правилу у систему се додељује фактор извесности заснован на експертској процени и компонтента фактора извесности. Вредности *MB* и *MD* се иницијализују на нулу, а затим се инкрементално укључују ефекти сваког од правила. Сваки пут када се разматра додатно правило израчунавају се нове вредности за мере *MB* и *MD*. Опсези вредности за *MB*, *MD* и *CF* су:

$$0 \leq MB(h,e) \leq 1$$

$$0 \leq MD(h,e) \leq 1$$

$$-1 \leq CF(h,e) \leq 1$$

Уколико се ради о узајамно искључивим хипотезама и уколико је:

- хипотеза h извесна, односно $P(h \mid e) = 1$, тада је:

$$MB(h,e) = \frac{1 - P(h)}{1 - P(h)} = 1$$

$$MD(h,e) = 0$$

$$CF(h,e) = 1$$

- негација хипотезе *h* извесна, односно $P(\sim h \mid e) = 1$, тада је:

$$MB(h,e) = 0$$

$$MD(h,e) = \frac{0 - P(h)}{0 - P(h)} = 1$$

$$CF(h,e) = -1$$

Према дефиницијама за *MB* и *MD*, вредност $MB(\sim h,e) = 1$, само уколико је $MD(h,e) = 1$, односно вредност 1 представља апсолутно поверење (или неповерење) за мере *MB* (или *MD*). На пример, ако су h_1 и h_2 узајамно искључиве хипотезе и ако је $MB(h_1,e) = 1$, тада је $MD(h_2,e) = 1$.

Одсуство опажања се изражава помоћу следећих вредности:

- $MB(h,e) = 0$, ако опажање *e* не потврђује хипотезу *h*,
- $MD(h,e) = 0$, ако опажање *e* не оспорава хипотезу *h*,
- $CF(h,e) = 0$, ако опажање *e* ни не потврђује ни не оспорава хипотезу *h*.

Када има више правила која доприносе за или против закључка, тада се рачунају кумулативне вредности за *MB* и *MD*, као и кумулативни фактор извесности. Кумулативни фактор извесности за хипотезу *h* која се посматра, добија се као:

$CF(h,eh) = MB(h,ez) - MD(h,ep)$

где вредност *eh* представља сва опажања везана за хипотезу *h*, која су узета у обзир до посматраног тренутка, вредност *ez* представља опажања која подржавају хипотезу *h*, а вредност *ep* су опажања која оспоравају хипотезу *h*. Кумулативне мере поверења $MB(h,ez)$, односно неповерења $MD(h,ep)$ у хипотезу *h*, добијају се на основу вредности *ez*, односно *ep* респективно.

Непрецизност фактора извесности у моделу *MYCIN* је био повод за многе студије случаја последњих година, које су приказале алгоритме корекције ове стратегије и алтернативне функције израчунавања [33, 34].

2.5.2. Расплинута логика

Расплинута логика је вишевредносна логика проистекла из теорије расплинутих скупова. Теорија расплинутих скупова представља проширење теорије скупова. У класичној теорији скупова један елемент или припада или не припада скупу. Код теорије расплинутих скупова, један елемент има степен припадности скупу (енг. *degree of membership*), а степен припадности исказан је функцијом чланства (енг. *membership function*). Термин „*fuzzy*“ означава да логика описује непрецизне појмове, попут људског знања.

Расплинути скуп је пар (A, m), где је A скуп, а m:$A\rightarrow[0,1]$ функција чланства која врши пресликавање скупа A у реалан број из опсега [0,1]. За елемент чији је степен припадности 0 каже се да уопште не припада скупу, док за елемент са степеном припадности 1 каже се да скроз припада скупу. Расплинути скуп (A, m) се дефинише над класичним скупом S. На пример, скуп S може да представља температуру, док расплинути скуп A представља високу температуру. Функцијом чланства m се одређује да ли нека конкретна температура припада скупу високе температуре.

2.5.3. Систем за одржавање истинитости

Систем за одржавање истинистости (*TMS* систем) је помоћни алат за решавање проблема чији је циљ одржавање конзистентности базе знања за системе који раде у неизвесном окружењу [35]. Овај алат нема активну улогу и примењује ревизију поверења да би се одржала конзистентност базе знања.

База знања овог система се састоји од правила, која се називају чворовима. Правило је један исказ који може да зависи од других правила и који има своје стање. Стање правила, односно чвора је *IN*, ако је правило истинито, односно *OUT*, ако није истинито. Уз сваки чвор придружује се листа претпоставки или листа оправдања. Претпоставке су друга правила, односно други чворови у бази знања. *TMS* води рачуна о међузависности појединих чворова и на основу информација о стањима чворова одржава конзистентност базе [35].

Листе оправдања су:

- листа подршке (енг. *support list*, *SL*):

SL (*IN-lista*) (*OUT-lista*)

- листа условних ограничења (енг. *conditional-proof justifications*, *CP*):

  CP <*posledica*> (*IN-lista*) (*OUT-lista*)

IN-листа је листа свих чворова који морају бити у *IN* стању да би чвор коме је ова листа придружена био истинит. *OUT*-листа је листа свих чворова који морају бити у *OUT* стању да би чвор коме је ова листа придружена био истинит. Чвор је истинит тек онда када су сви чворови из *IN*-листе у *IN* стању и када су сви из *OUT* листе у *OUT* стању.

TMS системи активно се користе у семантичком вебу код система за управљање и резоновање онтологија [36].

2.6. Стратегије машинског учења

2.6.1. Стабла одлучивања

Индуктивно учење представља процес размишљања који изводи опште закључке из скупа коначних примера. Претпоставке, односно премисе индуктивног логичког аргумента указују на неку могућност за закључак и сугеришу га, али не гарантују његову истинитост. Примена индуктивног закључивања и учења огледа се у томе да систем који учи буде у стању да донесе закључке ако је потребно на основу једног примера или да додатним новим примерима промени своје знање.

Циљ машинског учења је да конструишемо алгоритме који су способни да науче да предвиђају одређене циљане излазе. Да би се ово постигло, алгоритму учења се задају одређени примери за тренирање, који демонстрирају циљају релацију улазних и излазних вредности. Алгоритам треба да генерише приближно тачан излаз, укључујући и оне примере које није имао током тренирања. Овакав задатак не може да буде решен тачно без додатних претпоставки, јер ситуације за које алгоритам није обучаван могу да имају и друге излазне вредности. Врсте неопходних претпоставки о природи циљане функције називају се термином индуктивни биас.

Учење преко стабла одлучивања (енг. *decision trees*) представља учење које се највише користи у практичним методама индукционог закључивања. Стабла одлучивања се најчешће користе за класификацију [37]. Најпознатији алгоритми учења засновани на принципу рада стабала одлучивања су *CLS (Concept Learning System),* Хантов алгоритам (енг. *Hunt's algorithm*), *CART*, *ID3 (Iterative dichotomiser 3)*, *AQ/AQR* (*Algorithm Quasi-Optimal*), *ILA*

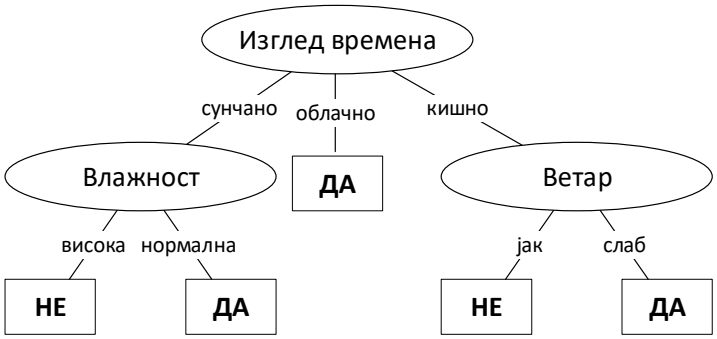
(*Inductive Learning Algorithm*), *ASSISTANT*, *SLIQ* (*Superviserd Learning In Quest*), *C4.5* и *Random Forest* [38]. У основи ови алгоритми обилазе стабло приступом од врха ка дну (енг. *top-down*).

Класификатори стабала одлучивања почињу са тренинг скупом који има придружене лабеле класа и корени чвор који представља главну карактеристику. Сваки унутрашњи чвор представља атрибуте теста, свака грана представља исход теста, а сваки чвор лист представља лабелу класе. Да бисмо идентификовали лабелу класе за непознат пример, класификатор стабла одлучивања прати пут од кореног чвора до чвора листа, који садржи лабелу класе за тај пример.

Алгоритми учења код стабла одлучивања претражују простор хипотеза и на тај начин избегавају проблеме који се јављају код ограничених простора хипотеза. Индуктивни биас код ових алгоритама представља предност малих стабала над великим стаблима одлучивања. Стабла одлучивања групишу инстанце тако што их сортирају крећући се кроз стабло на доле од корена до листа, који одређује класу инстанце.

Инстанца је уређен пар атрибут-вредност. На слици 7 дат је пример стабла одлучивања помоћу кога разматрамо да ли је одређено јутро погодно за играње тениса у односу на временске услове. Дати су атрибути: изглед дана који има вредности из скупа {сунчано, облачно, кишно}, влажност која има вредности из скупа {висока, нормална} и ветар који има вредност из скупа {јак, слаб}. Инстанца се разврстава од корена стабла тестирањем атрибута. Лист стабла је чвор који нема чворове потомке. Овај процес се рекурзивно понавља за свако подстабло чији корен представља нови унутрашњи чвор.

Сваки чвор који је лист мора да има излазну лабелу, у случају класификације та лабела садржи име класе, а у случају регресије нумеричку вредност. Стабла одлучивања представљају дисјункције конјункција на основу вредности атрибута инстанце. Сваки пут од корена ка листу стабла одговара конјункцији тестова атрибута, а целокупно стабло одговара дисјункцији свих конјукција.



Слика 7 – Пример стабла одлучивања

Током изградње стабла одлучивања могуће је управљати подацима за које неки атрибути имају непознату вредност проценом добити. Коришћењем стабла одлучивања могуће је класификовати записе који имају непознате вредности тако што се израчуна процена вероватноћи за различите исходе.

Учење уз помоћ стабала одлучивања одговара проблемима са следећим карактеристикама:

- Стабла одлучивања представљају дисјунктне изразе;
- Инстанце су представљене као скуп атрибута и њихових вредности, а сваки атрибут има релативно мали број дисјунктних могућих вредности;
- Циљна функција има дискретне излазне вредности (најчешће стабла имају Булову класификацију: тачно / нетачно);
- Тренинг подаци могу да садрже грешке - грешке при класификацији тренинг примера и грешке у вредностима атрибута које описује ове примере;
- Тренинг подаци могу да садрже непотпуне вредности атрибута.

Многи практични проблеми се уклапају у наведене карактеристике, као што су класификација медицинских података пацијената према болестима и симптомима које осећају, класификација кварова опреме према узроку настанка, и слично. Овакви проблеми који имају за циљ да се улазни подаци разврстају у један одређени скуп могућих категорија називају се класификациони проблеми [39].

2.6.2. ID3 алгоритам

Основна идеја *ID3* алгоритма је да конструише стабло одлучивања примењујућу претрагу одозго на доле кроз дате скупове, тако што сваки атрибут тестира на сваком чвору.

Како би се одабрао најкориснији атрибут за класификацију датог скупа, користи се информациона добит као критеријум најбоље поделе. Алгоритам *ID3* претпоставља да сваки атрибут може имати вредност из скупа неких предефинисанх вредности, односно садржи само дискретне податке, насупрот редним подацима као што су године, висина, итд. Стабло се конструише од врха ка дну, рекурзивним путем. На почетку се одређује корени чвор тако што се сваки атрибут тестира и одређује који је најбољи атрибут за поделу, а потом се врши подела података на подскупове. Затим се рекурзивно позива *ID3* алгоритам за сваки подскуп. Овај алгоритам има најчешће примену на проблемима са следећим карактеристикама:

1. Инстанце су представљене паровима атрибут-вредност.
2. Циљна функција има дискретне излазне вредности.
3. Вредности атрибута су номиналне.

Алгоритам *ID3* претражује простор хипотеза како би нашао једну која најбоље одговара тренинг примерима [19].

3. ПРЕГЛЕД ПОСТОЈЕЋИХ СОФТВЕРСКИХ СИСТЕМА И ЗАХТЕВИ ЗА РЕАЛИЗИЈУ НОВОГ СИСТЕМА

У овом поглављу биће приказани описи постојећих софтверских система за учење и примену алгоритама вештачке интелигенције, дати захтеви реализације новог софтверског система и представљена методологија која је коришћена у пројектовању таквог система.

3.1. Постојећи софтверски системи за учење алгоритама вештачке интелигенције

У истраживању су анализирани софтверски системи који се као софтверски системи и помоћни софтверски алати користе у држању наставе на предметима из области вештачке интелигенције, који су веома заступљени на универзитетима широм света [40]. У наставку овог поглавља ће бити анализирани најважнији софтверски системи описани у литератури, који се могу пронаћи у овој области, уз навођење уочених предности и недостатака.

Симулатор „*Chinese room*“ реализован је на Државном универзитету Илиноји, Сједињене америчке државе, у оквиру курса из вештачке интелигенције [41]. Овај проблем се обично приказује у уводном делу курса, када се разматра појам рачунарског модела људског размишљања. Недостатак овог симулатора је што не постоји могућност никакве интеракције са корисником, већ је све приказано кроз анимацију.

Симулатор „*MIT Open Course Ware*“ представља алат у настави курса вештачке интелигенције на Универзитету технологије у Масачусетсу и на Универзитету у Чикагу [42]. Састављен је од шест целина: планирање, претраживање, метод задовољења ограничења, биолошки миметици, други видови учења и Бајесове мреже. Планирање је представљено употребом алгоритма *STRIPS*, а од алгоритама претраживања подржани су алгоритми претраживања по дубини и ширини, алгоритми планинарења, прво најбољи, гранања и ограничавања и *A**. Од осталих алгоритама и стратегија овај систем је подржао алгоритам *Minimax* и алгоритам *Minimax* са алфа-бета одсецањем, метод задовољења ограничења реализован као проблем бојења мапа (енг. *map coloring problem*), генетске алгоритме, самоорганизујуће неуралне мреже и Бајесове мреже. У овој симулацији главни недостатак је што корисници нису у могућности да уносе своје проблеме за претраживање и не могу да прате извршавање алгоритама по корацима.

„*AI Search*“ симулатор развијен на Универзитету РМИТ у Аустралији демонстрира рад различитих алгоритама претраживања на примеру игре „*8-puzzle*“ и користи се на лабораторијским вежбама из вештачке интелигенције [43]. Евалуација је показала да су студенти који су били део експерименталне групе и користили овај симулатор, много боље разумели градиво тих алгоритама и показали су боље резултате у односу на групу која није користила овај симулатор [44]. Симулатор је врло интерактиван и са детаљним описом извршавања одабраног алгоритма. Главни недостатак овог симулатора је што су кроз ову игру приказани само алгоритми претраживања.

Софтверски систем „*CIspace*“ / „*AIspace*“ развијен је на Универзитету британске Колумбије [45]. Он представља колекцију интерактивних алата и покрива осам области: претраживање графа, решавање проблема задовољења ограничења засновано на доследности и засновано на стохастичком локалном претраживању, закључивање, Бајесове мреже, стабла одлучивања, неуралне мреже и конверзија *STRIPS* алгоритма у алгоритме задовољења ограничења, односно *CSP*. Систем подржава интерактивне симулације, што омогућава корисницима делимичну контролу над примерима [46]. Овај софтверски систем је добар као подршка у учењу ове области, али нажалост њиме нису покривене све обавезне теме из области вештачке интелигенције.

„*AIMA3e*“ симулатор базиран је на свим алгоритмима из најпознатије књиге из ове области [47]. Овај систем садржи седам симулација и 13 текстуалних демоа, који приказују

рад појединих алгоритама вештачке интелигенције. Детаљно су приказани алгоритми претраживања, популарне игре из области теорије игара и игре базиране на алгоритму задовољења ограничења. Као и претходни систем, ни овај не покрива све препоручене теме за област вештачке интелигенције, а такође један од главних недостатака је што корисник система не може да утиче на примере који су дати као демо анимације.

Симулатор „*Searcher*“ представља помоћ едукаторима и студентима у учењу и подучавању алгоритама претраживања [48]. Користи се на Универзитету у Ливерпулу. Веома је погодан за рад, јер добар графички приказ омогућава лако праћење анимација, што је и највећа предност. Недостатак овог симулатора је што је базиран само на алгоритмима претраживања.

„*Simple Expert System*“ представља приказ једноставног експертског система који доноси закључке на основу базе знања [49]. У овом симулатору, корисник има могућност да изабере једну од понуђених база знања, или да креира своју, поштујући правила формата продукција и чињеница. Корисник одговара на питања са тачно или нетачно и на основу корисникових одговора систем треба да изведе закључак, користећи продукције и чињенице. Недостатак ове симулације је што нема могућност чувања базе знања, ни учитавања базе знања из фајла, а уз то базиран је само на системима закључивања.

Симулатор „*Sudoku Solver*“ представља алат који користи метод задовољења ограничења за демонстрацију решавања Судоку проблема [50]. Корисник може да учита пример из библиотеке са примерима, самостално креира нови Судоку пример, упише вредности у поља табеле и решава Судоку игру без помоћи. Када корисник унесе у поље вредност која нарушава неко од ограничења, то поље са погрешном вредношћу ће бити јасно уоквирено. Недостатак овог симулатора је што кроз игру Судоку приказује искључиво алгоритме из области метода задовољења ограничења.

Систем „*AI Exploratorium*“ демонстрира рад стабала одлучивања (енг. *decision tree learning*) и алгоритма претраге *IDA** [51]. Његови недостаци су што кроз реализоване алгоритме приказује само основе машинског учења, без могућности да корисник система сам креира сопстевене примере и извршава их.

Врло често се софтверски системи засновани на вештачкој интелигенцији примењују и у симулацијама спортских игара, симулацијама ратних игара или у играма на срећу [52] [53]

[54]. Такви системи, иако у серверском делу реализују већи број различитих алгоритама, тежи су за објашњавање рада алгоритма, и нису погодни за самостално учење студената.

Фудбалски симулатор „*Multi-Agent Soccer*“ реализовао је поједностављени модел фудбала и може се користити у главним темама на уводном курсу из вештачке интелигенције [53]. Предност овог симулатора је што покрива већи број алгоритама и што корисници система сами могу учитавати податке, али као главни недостаци се издавајају немогућност прегледа појединачног рада одређеног алгоритма и његовог извршавања корак по корак.

Сличан систем, назван „*RobotSoccer*“, имплементирао је алгоритме претраживања и генетски алгоритам [54]. Систем је визуелно врло прегледан, али слаба интеракција са корисником и немогућност прегледа појединих корака алгоритма представљају главне недостатке овог система.

Једнако значајан пројекат је „*MLeXAI*“, развијен за учење вештачке интелигенције, са акцентом на примени машинског учења [55]. Циљ пројекта је да развије прилагодљив систем за презентовање тема из области вештачке интелигенције на рачунару, чиме показује велику повезаност рачунарских наука и вештачке интелигенције [56]. Овај комплексни систем замишљен је да се користи у виду шест практичних лабораторијских пројеката: 1) класификација веб докумената, помоћу које се уче основе откривања података и основни алгоритми машинског учења; 2) профилисање веб корисничких профила; 3) препознавање знакова помоћу неуралне мреже; 4) решавање проблема „*N-puzzle*“ помоћу алгоритама претраживања; 5) решавање проблема „*Dice Game Pig*“ са основним концептима *reinforcement* учења; 6) проблем „*Clue Deduction*“ за репрезентације знања и резоновања. Добра страна овог пројекта је разноврсност тема и лака интеграција ових софтверских алата у уводни курс из вештачке интелигенције, а лоша страна је непостојање униформности и разноликост тих алата, који су саставни део пројекта. У неким лабораторијским задацима „*MLeXAI*“ се ослања на резултате добијене из других система, као што је на пример „*WEKA*“.

Систем „*WEKA*“ представља колекцију алгоритама из области машинског учења [57] [58]. Постоји могућност директне примене тих алгоритама из кода писаног у програмском језику Јава. Систем „*WEKA*“ пружа могућност обраде података што подразумева: претпроцесирање, класификацију, регресију, кластеризацију и визуелизацију. Неки од подржаних алгоритама за конструкцију стабала одлучивања су: *ADTree*, *BFTree* и *ID3*. Овај систем је врло комплексан, даје добру визуелизацију и објашњења кориснику, али је главни

недостатак овог система што је акценат његовог рада искључиво на подобластима проналажења и обраде података и машинског учења, а не темама које су дефинисане као обавезне у курикулумима *IEEE* и *ACM*.

У табели 3 дат је преглед главних тема и степен којим су те теме обухваћене у анализираним системима за учење вештачке интелигенције. Анализирани софтверски системи који су реализовани за потребе одређених курсева на различитим интернационалним универзитетима, са успехом се користе у настави, али ниједан не обухвата све потребне елементе који су препоручени од стране струкованих организација *IEEE* и *ACM* за курс из области вештачке интелигенције. Из анализе можемо видети да постоје две групе оваквих симулација: системи који обухватају само једну област са имплементираним једним или неколико алгоритама и системи који обухватају неколико области вештачке интелигенције и имплементирају већи број алгоритама који су прилично разнолики.

ТАБЕЛА 3
ПОКРИВЕНОСТ ГЛАВНИХ ТЕМА ИЗ ОБЛАСТИ ВЕШТАЧКЕ ИНТЕЛИГЕНЦИЈЕ У АНАЛИЗИРАНИМ СИСТЕМИМА

ЛЕГЕНДА	НИВО ПОКРИВЕНОСТИ
○	Делимичан ниво покривености тема
●	Значајан ниво покривености тема
⊕	Потпуни ниво покривености тема
x	Теме које нису покривене

Систем	Основни концепти	Основне стратегије претраживања	Резоновање засновано на знању	Напредне стратегије претраживања	Напредне технике резоновања	Машинско учење	Планирање
Chinese Room	⊕	x	x	x	x	x	x
MIT OCW	●	⊕	⊕	⊕	○	●	●
AI Search	●	⊕	x	x	x	x	x
CIspace / AIspace	●	⊕	⊕	⊕	○	●	●
AIMA3e	●	⊕	⊕	⊕	○	●	x
Searchr	●	⊕	x	●	x	x	x
Simple ES	●	x	●	x	x	x	x
Sudoku Solver	●	○	x	x	x	x	x
Minesweeper	●	○	x	x	x	x	x
AI Explor.	●	○	x	x	x	○	x
Multi-Agent Soccer	○	⊕	●	○	○	x	x
RobotSoccer	●	○	x	⊕	x	x	x
MLeXAI	●	●	●	●	○	⊕	x
WEKA	●	x	○	x	○	⊕	x

У табели 4 дате су основне информације о анализираним софтверским системима: универзитет на коме је систем реализован и програмски језик у коме је систем имплементиран, као и евалуација ових софтверских система гледајући главне карактеристике

(к1-к9). Визуелна репрезентација је графички приказ алгоритма и изражава се у три нивоа: одлична, добра и лоша. Контролисање брзине извршавања симулације, односно алгоритма, означена је као КК, ако се користи режим корак по корак и са Р, ако симулација ради константно, док не достигне коначно решење. У другим карактеристикама, присуство одређених особина означено је са ДА, одсуство са НЕ, а делимично са ДЕЛ.

ТАБЕЛА 4
ОПШТЕ ИНФОРМАЦИЈЕ О АНАЛИЗИРАНИМ СОФТВЕРСКИМ СИСТЕМИМА И ЕВАЛУАЦИЈА ЗАСНОВАНА НА ГЛАВНИМ КАРАКТЕРИСТИКАМА

Назив система	Универзитет	Програмски језик	к1	к2	к3	к4	к5	к6	к7	к8	к9
Chinese Room	State Illinois University, US	Flash	одлична	не	Р	не	да	не	не	да	не
MIT Open Course Ware	Massachusetts Institute of Technology, US	Java	одлична	да	Р	не	да	не	да	да	не
AI Search	RMIT University, Melbourne, Australia	Java	одлична	да	КК+Р	дел	да	да	не	да	не
CIspace / AIspace	The University of British Columbia, Canada	Java	одлична	да	КК+Р	дел	да	дел	да	да	не
AIMA3e	by several universities and researchers [10]	Java	одлична	да	КК+Р	не	да	не	да	не	не
Searchr	University of Liverpool, UK	Coffee-Script	одлична	да	КК+Р	не	да	да	не	да	не
Simple Expert System	University of Kent, UK	Java	добра	да	Р	не	да	да	не	не	не
Sudoku Solver	University of Nebraska, US	Java	одлична	да	КК+Р	не	да	да	не	да	не
Minesweeper	University of Nebraska, US	Java	одлична	да	КК+Р	не	да	да	не	да	не
AI Exploratorium	University of Alberta, Canada	Java	одлична	да	КК+Р	не	да	не	не	да	не
Multi-Agent Soccer	Delft University of Technology, Netherlands	Java	одлична	да	Р	не	да	да	да	да	не
RobotSoccer	Adelaide University, Australia	Java	добра	не	Р	не	не	да	не	не	не
MLeXAI	University of Hartford, US	Java/Lisp /Prolog/ Matlab	-	не	Р	не	да	не	не	не	не
WEKA	University of Waikato, New Zealand	Java	одлична	да	Р (са детаљима)	дел	да	да	не	да	не

ЛЕГЕНДА:	**к1 – ВИЗУЕЛНА РЕПРЕЗЕНТАЦИЈА**	**к6 – РЕАЛИЗАЦИЈА НОВИХ ПРОБЛЕМА/ЗАДАТАКА**
	к2 – ИНТЕРАКТИВАН ВИД СИМУЛАЦИЈЕ	**к7 - КОНЗИСТЕНТНОСТ**
	к3 – КОНТРОЛА ИЗВРШАВАЊА АЛГОРИТМА	**к8 – ПОМОЋ**
	к4 – УПОРЕДНИ ПРИКАЗ ИЗВРШАВАЊА	**к9 – МОБИЛНА АПЛИКАЦИЈА РЕАЛИЗОВАНА**
	к5 – ПРЕДЕФИНИСАНИ ПРОБЛЕМИ	**КК/Р – КОРАК ПО КОРАК ТЕМПО / РАД ДО КРАЈА СИМУЛАЦИЈЕ**

Закључак анализе описаних софтверских система који се користе у настави из вештачке интелигенције је да су најкомплетнији системи „*MIT Open Course Ware*“, „*AIspace*“ и „*AIMA3e*“. Они имају врло добар кориснички интерфејс и покривају већи број главних тема из вештачке интелигенције, што их сврстава у најпопуларније софтверске системе у настави из ове области. Анализом је такође закључено да ниједан систем нема све карактеристике, што је био довољан разлог да се реализује нови софтверски систем *SAIL - System for Artificial Intelligence Learning* [59]. Карактеристике к1-к9 приказане у табели 4 мапиране су у захтеве за реализацију новог софверског система, који су описане у поглављу 3.3.

3.2. Избор модела развоја софтверског система

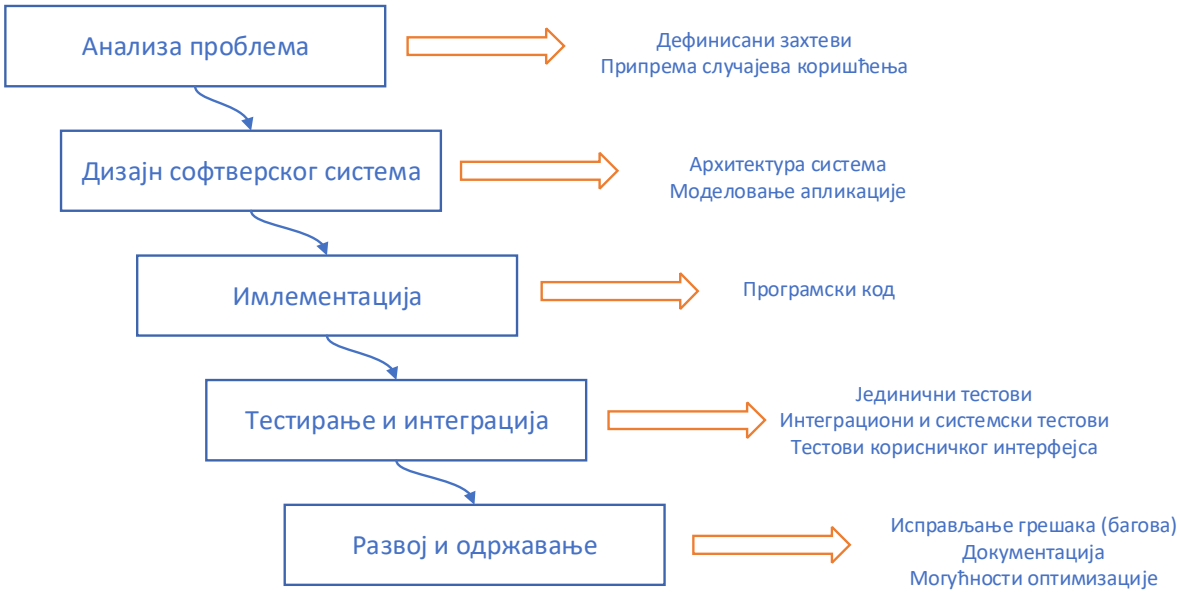
Животни циклус развоја софтвера или модел развоја софтвера садржи одређени број фаза које се извршавају у одређеном редоследу. Врло је важно да софтверски инжењери који развијају софтверске системе имају довољно знања о томе који модел одабрати, према захтевима и контексту софтверског пројекта који треба да реализују. Неки од најважнијих фактора који утичу на избор модела су:

- колико су јасни захтеви за израду софтверског система,
- колико је позната технологија,
- каква је комплексност система,
- реупотребљивост компоненти (делова система),
- фактори управљања софтверским пројектом и контрола/мониторинг (дужина трајања пројекта, ограничења у буџету, учесници пројекта),
- потреба за документацијом.

Главне типови модела развоја софтвера су:

- модел водопада (енг. *waterfall*),
- В модел (енг. *V model*),
- еволутивни прототипски модел (енг. *evolutionary prototyping model*),
- спирални модел (енг. *spiral*),
- итеративни и инкрементални модел (енг. *iterative and incremental model*),
- агилни развој софтвера (енг. *agile development*).

Модел водопада је најранији и најпознатији приступ у развоју софтверских система. Модел водопада садржи линеарни секвенцијални низ фаза развоја софтвера [60]. Број фаза је најчешће пет, али их може бити и више. Свака фаза у развојном циклусу почиње само уколико је претходна фаза завршена. Модификовани модел водопада даје могућност да се циклус врати у претходну фазу, уколико има неких промена у захтевима, дизајну система, имплементацији и другим фазама. Модел водопада са главним фазама које овај модел укључује, приказан је на слици 8.



Слика 8 – Модел водопада са могућим излазима сваке фазе

В модел је проширење модела водопада. Главна разлика између модела водопада и В модела је што се тестирање може извршавати раније код В модела и што се након фазе имплементације ради неколико фаза тестирања - јединично тестирање, интеграционо тестирање и системско тестирање, након чега се систем ставља у продукцију [61].

Еволутивни - прототипски модел заснован је на развоју прототипске апликације, односно непотпуне верзије софтверског система [61]. Циљ овог модела је да када се развије коначни прототип нема даљих измена у захтевима. Овај модел не може да се имплементира са моделом водопада, због нефлексибилног приступа, али прототип као фаза у развоју софтверских пројеката се често јавља код многих модела, јер се тиме постижу ранији коментари и измене захтева од стране будућих корисника система.

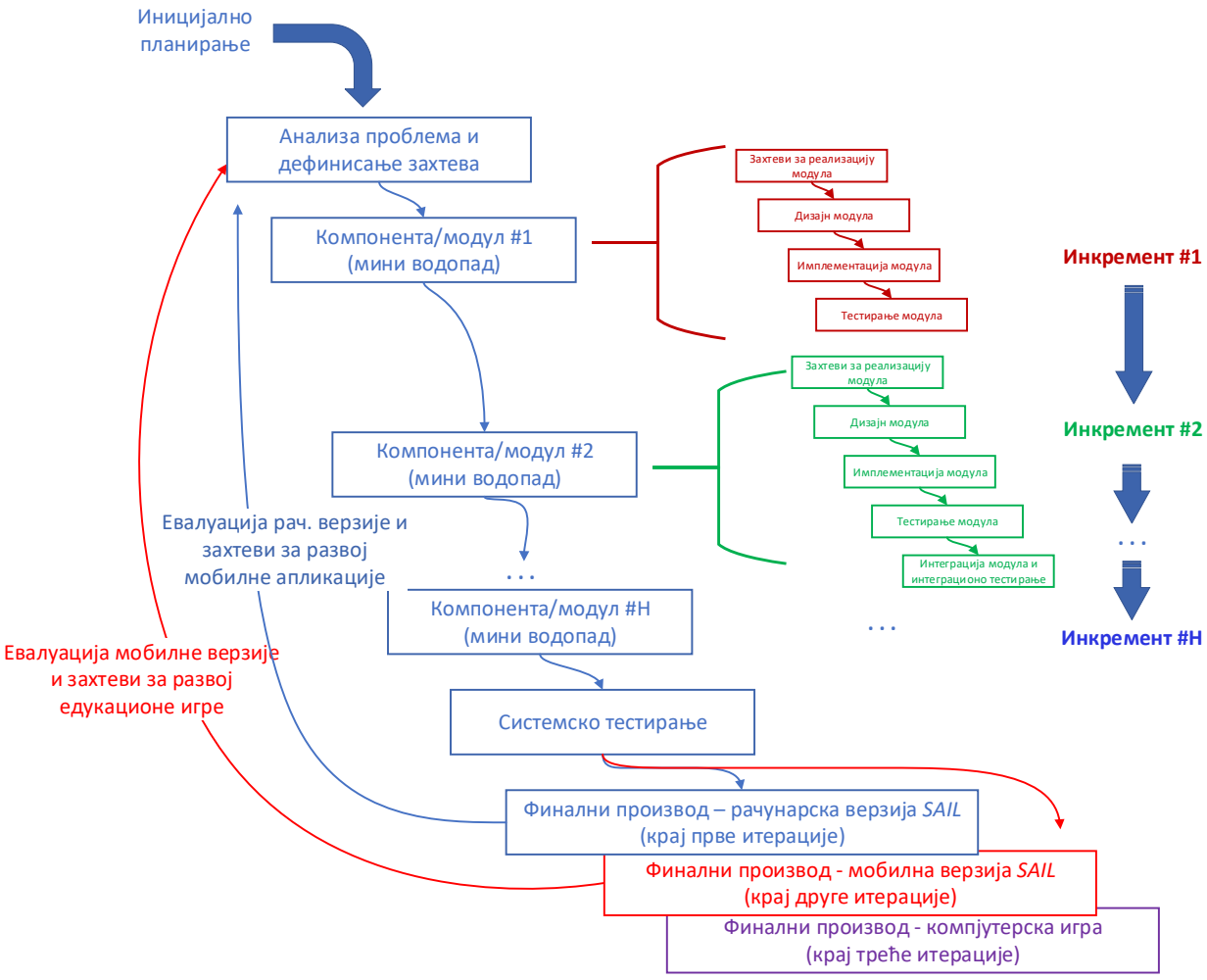
Спирални модел комбинује одређене особине модела водопада и прототипског модела и најчешће се користи код великих, скупих и сложених софтверских пројеката [61]. Фазе овог модела су врло сличне моделу водопада, односно извршавају се истим редоследом, али свака фаза има своје четири потфазе прогреса – одређивање циљева, идентификација и процена ризика, имплементација и тестирање и планирање следеће фазе. Фазе из модела водопада у овом моделу праве спиралу по чему је модел добио назив.

Итеративни и инкрементални модел базиран је на развоју система са поновљеним циклусима (итерацијама) и испоруком мањих делова у времену (инкрементима) [60].

Предност овог приступа је да софтверски инжењери могу раније да примене одређене измене као коментар клијената, а клијенти да могу раније да виде делове система или неке верзије подсистема/система који се развија.

Агилни развој софтвера представља данас најсавременији приступ који користи одређене процесе из већ описаних модела и серију инкремената, тако да сваки инкремент означава једну нову функционалност унутар система који се развија [60] [61]. Најпознатија агилна методологија је екстремно програмирање (енг. *extreme programming*, *XP*) [62], а друге значајне и често коришћене су: Скрам (енг. *Scrum*) [63], Кристал (енг. *Crystal*) [64], адаптивни развој (енг. *Adaptive Software Development*) [65], *DSDM – Dynamic Systems Development Method* [66] и развој заснован на својствима (енг. *Feature Driven Development*) [67].

Приликом развоја софтверског система *SAIL* коришћен је хибридни приступ, комбиновањем више различитих модела развоја софтвера. У основи то је итеративно-инкрементални модел, који је одабран због велике комплексности система. Свака итерација у себи је имала примену неколико мањих модела водопада за сваки модул појединачно, а надоградњом једног модула на други формирани су нови инкременти система. Итерацијама је претходила фаза анализе захтева, заснована на детаљном прегледу постојећих сличних система и дефинисању свих захтева. Ова фаза укључила је и развој дела прототипског решења у виду предложених корисничких екрана, прво за рачунарску, а затим и за мобилну верзију система. Хибридни модел развоја софтверског система примењен приликом развоја *SAIL* приказан је на слици 9. Више различитих модела често се примењује заједно и они нису међусобно искључиви, посебно код развоја великих софтверских система, што је у овом истраживању при развоју *SAIL* био случај.



Слика 9 - Модел развоја софтверског система *SAIL*

3.3. Захтеви за реализацију новог софтверског система

Након анализе сродних софтверских система и истраживања, анализе курикулума и стручних препорука, искустава у развоју софтверских система који се користе у образовању и за учење алгоритама, закључено је да такав софтверски систем треба да има следеће захтеве:

- Захтев 1 (з1): заједнички графички кориснички интерфејс (енг. *graphic user interface*, *GUI*) за све симулације, интуитивне команде и заједничке команде у траци са алатима (енг. *toolbar*);
- Захтев 2 (з2): висок ниво интеракције између корисника и система;
- Захтев 3 (з3): симулација корак по корак за сваки алгоритам и приказ симулације/алгоритма у реалном времену;
- Захтев 4 (з4): могућност упоредне анализе алгоритама током њиховог извршавања;
- Захтев 5 (з5): учитавање већ предефинисаних проблема и задатака и њихово извршавање у оквиру система;
- Захтев 6 (з6): могућност прављења сопствених примера, промене улазних параметара симулације, снимања текућег примера са свим параметрима и учитавање раније снимљених примера;
- Захтев 7 (з7): конзистентност у представљањима и описима решења имплементираних алгоритама и конзистентност у корисничком интерфејсу;
- Захтев 8 (з8): могућност коришћења менија за помоћ у сваком кораку извршавања алгоритма;
- Захтев 9 (з9): вишеплатформски (мултиплатформски) систем који укључује рачунарску и мобилну апликацију уз могућност самосталног учења помоћу мобилних уређаја.

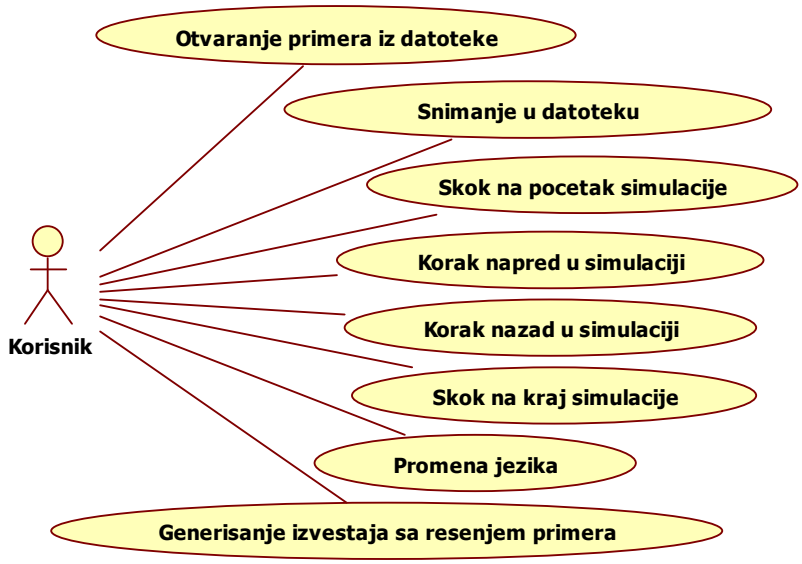
Још један од кључних захтева био је да реализовани систем буде објектно-оријентисан и модуларан, где сваки модул представља једну симулацију одређене групе алгоритама. Модуларност је уведена да би систем могао да се имплементира, тестира и развија кроз различите верзије и да би у новијим верзијама на лак начин могле да се додају симулације

неких других група алгоритама. Такође, приликом постављања иницијалних захтева о архитектури система, одлучено је да систем буде независан од интернет конекције, па се одустало од веб система и клијент-сервер архитектуре.

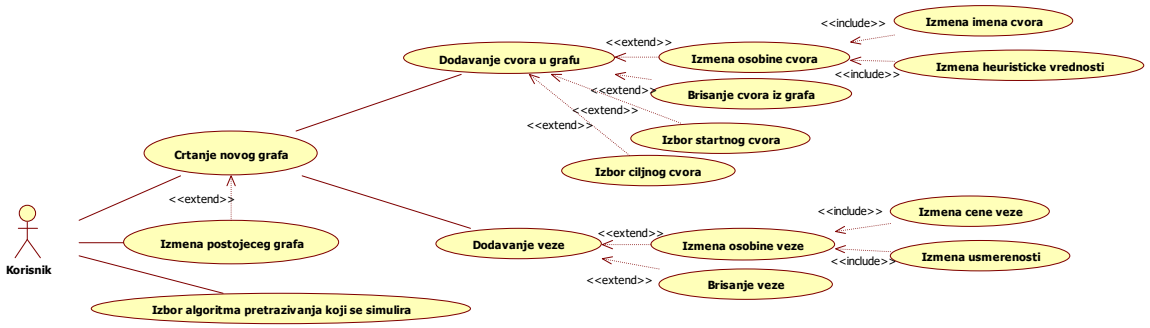
3.3.1. Припрема случајева коришћења и прототипа

Након дефинисања општих захтева које систем треба да подржи, било је потребно дефинисати функционалности система, односно специфичне захтеве. Сваки модул има своје специфичне захтеве, тако да избором модела базираног на итеративно-инкременталном приступу, те захтеве одређујемо непосредно пре започињања рада на том модулу.

Функционалности система у овој фази развојног циклуса биле су дефинисане преко дијаграма случајева коришћења (енг. *use case diagram*) и коришћењем спецификације сценарија употребе (ССУ). На слици 10 дат је дијаграм случајева коришћења који обухвата заједничке опције свих симулација алгоритама и те опције су укључене у заједнички графички корисничких интерфејс. Сваки модул има своје додатне опције и представља се са по једним дијаграмом случајева коришћења само тих функционалности. На слици 11 дат је дијаграм случајева коришћења за модул који реализује стандардне алгоритме претраживања.



Слика 10 - Дијаграм случајева коришћења заједничког интерфејса свих симулација



Слика 11 - Дијаграм случајева коришћења за модул стандардних алгоритама претраживања

Сваки случај коришћења има свој сценарио употребе. Спецификацију сценарија чине назив сценарија, актер(и), предуслов(и) и постуслов(и), главни ток извршавања сценарија и алтернативни токови, на које актер може да оде уколико у одређеном кораку сценарија не буде извршен корак главног тока, па се тада прелази на алтернативну грану. Пример једног сценарија употребе за отварање већ дефинисаног примера из датотеке, дат је у наставку:

- Назив сценарија употребе: Отварање примера из датотеке
- Актери: корисник система, систем
- Предуслови: постоји датотека коректног типа (екстензије) и коректног садржаја (формата)
- Постуслови: датотека је успешно отворена унутар система
- Ток извршавања:

1. Корисник одабере симулацију, односно модул који жели да користи.
2. Систем учитава модул који је корисник одабрао, иницијално без података у симулацији.
3. Корисник бира опцију за отварање датотеке и отвара му се дијалог за избор датотеке.
4. Корисник потврђује избор датотеке коју жели да отвори.
5. Систем успешно учитава датотеку и параметре симулације, након чега је корисник спреман да покрене симулацију алгоритма.

5.а.1. Систем неуспешно учитава датотеку, јер тип датотеке (екстензија) не одговара симулацији одабраног алгоритма.

5.а.2. Систем исписује грешку да је некоректан тип датотеке и враћа корисника у корак 2.

5.б.1. Систем неуспешно учитава датотеку, јер садржај датотеке не може да се коректно учита у симулацију одабраног алгоритма.

5.б.2. Систем исписује грешку да садржај датотеке није добар и враћа корисника у корак 2.

У овом сценарију корак 5 главног тока има две алтернативне гране: 5а и 5б. Алтернативна грана 5а је предвиђена у оквиру овог сценарија, али на крају није реализована у имплементацији, зато што је унутар програмског кода забрањено да се у одређеној симулацији учитавају датотете типа који није подржан (нпр. симулација алгоритама претраживања може да учита само датотеке које имају тип *.grf*). Алтернативна грана 5б је имплементирана у систему.

Код агилних методологија развоја софтверских система уместо ССУ користе се структуре приче. Приче су уведе у приступу развоја софтвера вођеним понашањем софтвера (енг. *Behavior Driven Development, BDD*) да би сваку функционалност представили кроз причу. То је приступ који има за циљ да програмери разумеју детаљно проблем, да софтверски тестери могу лако да дизајнирају тестове, менаџери пројекта да разумеју све као једну целину, а клијенти да кроз кораке приче могу ближе да појасне корисничке захтеве. *BDD* приступ је настао из *TDD* (*Test Driven Development*) приступа. *TDD* је приступ развоја софтвера у коме се тестови пишу током имплементације уз јако кратке циклусе током развоја. Уз помоћ *BDD* лакше је разумети функционалности и како оне међусобно утичу једна на другу. Овим приступом такође брже реагујемо на измене, било у дизајну софтверског система, програмском коду или у тестовима, јер те фазе сада чине једну спрегу, лакше се могу уочити функционалности које нису потребне и приоритетније функционалности се прво реализују у систему. Ово је врло важно зато што се у великим системима дешава да се некада реализују функционалности које нису потребне у коначној верзији софтверског система и које се беспотребно реализују. Структура приче у *BDD* изгледа овако:

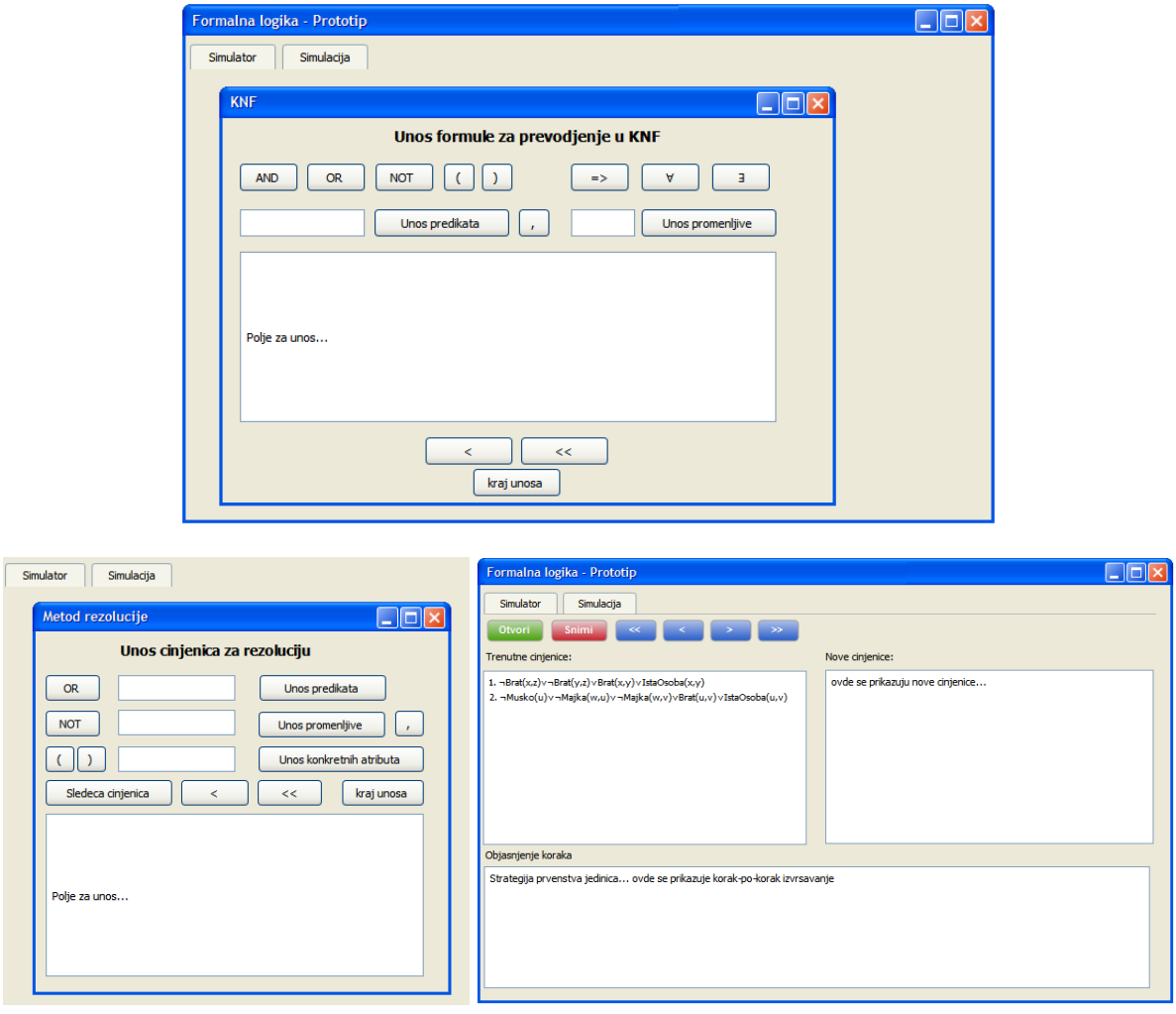
```
Title (one line describing the story)
Narrative:
As a [role]
I want [feature]
So that [benefit]

Acceptance Criteria: (presented as Scenarios)

Scenario 1: Title
Given [context]
And [some more context]...
When [event]
Then [outcome]
And [another outcome]...

Scenario 2: ...
```

Мобилна верзија система *SAIL* реализована је коришћењем *BDD* и *TDD* приступа, да би се реализовале само најприоритетније функционалности и да би се код рефакторисао од самог почетка реализације. Пример приче коришћене у реализацији симулације алгоритама претраживања у мобилној верзији система *SAIL* дат је у наставку:

```
Title: Simulacija algoritma pretraživanja po dubini

As a korisnik
I want simuliram izvršavanje algoritma na primeru
So that naučim rad algoritma pretrage po dubini

Scenario 1: Uspešan scenario (iscrtavanjem grafa)
Given nacrtan graf sa bar 2 čvora i jednom vezom
And označen jedan čvor kao početni
And označen jedan čvor kao ciljni
And korisnik odabrao algoritam pretrage po dubini
When korisnik pritisne dugme "Start simulation"
Then sistem icrtava koren stabla pretraživanja
And korisnik može da izvršava simulaciju algoritma

Scenario 2: Uspešan scenario (iz datoteke)
Given učitan ispravan graf iz datoteke
And označen jedan čvor kao početni
And označen jedan čvor kao ciljni
And korisnik odabrao algoritam pretrage po dubini
When korisnik pritisne dugme "Start simulation"
Then sistem icrtava koren stabla pretraživanja
And korisnik može da izvršava simulaciju algoritma
```

```
Scenario 3: Neuspešan scenario
Given nacrtan graf sa bar 2 čvora i jednom vezom
And nije označen početni/ciljni čvor
And korisnik odabrao algoritam pretrage po dubini
When korisnik pritisne dugme "Start simulation"
Then sistem ispisuje grešku "Morate odabrati početni i ciljni čvor".
```

Прототип је реализован и за рачунарску и за мобилну верзију система, да би се графички кориснички интерфејс што боље искористио. Ово је посебно важно код симулација у којима треба да се прикаже велики број корака и излазних параметара. За мобилну верзију додатно се кроз прототипско решење водило рачуна о величини екрана, да би се све информације симулације коректно приказале на мањим екранима мобилних уређаја. За реализацију прототипа коришћен је алат *Pencil*. Кориснички екрани прототипског решења приказани су на слици 12.



Слика 12 – Прототип једне симулације алгоритама у формалној логици

4. ПРОБЛЕМИ КОЈИ СУ РЕШАВАНИ ПРИЛИКОМ РЕАЛИЗАЦИЈЕ

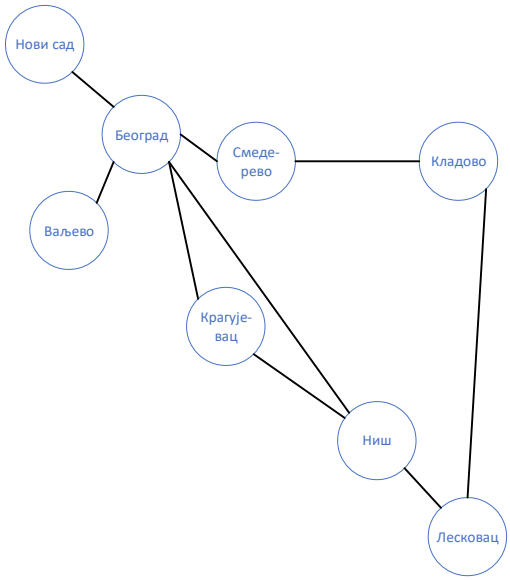
Главна особина архитектуре овог система је модуларност, па се надоградња система може реализовати на веома ефикасан начин додавањем нових алгоритама. У овом поглављу биће описана архитектура рачунарске апликације, реализација мобилне верзије апликације и како је систем могуће надоградити у игру, која би се користила у едукативне сврхе.

4.1. Архитектура рачунарске апликације

Систем је подељен у 14 модула распоређених у неколико пакета. Заједнички кориснички интерфејс и главне команде које имају све симулације алгоритама издвојене су у засебан пакет *glavni*. Стандардни алгоритми претраживања налазе се у пакету *pretrazivanje*, алгоритми формалне логике у пакету *flogika*, продукциони системи се налазе у пакету *produkcioni*, и тако даље. Код неких симулација коришћене су већ реализоване Јавине библиотеке, у изворној верзији или су уз неопходне измене у програмском коду, прилагођене овом систему. Неки проблеми морали су да се реализују од почетка, јер не постоје библиотеке отвореног кода које би се интегрисале у систем *SAIL* или постоје, али имају много већи број недостатака и проблема, који нису могли да се разреше током реализације.

Један од важних проблема које је требало решити приликом реализације система је да сваки модул има свој формат датотеке, такве да чува све релевантне податке о симулацији. При томе дефинисани формати требали су да испуне два циља: да буду једноставни за парсирање и да буду довољно једноставни за унос од стране корисника.

Други важан проблем код симулација које раде са графовима био је уношење графова и формирање стабла претраживања користећи граф. Графови се код алгоритама најчешће представљају као мрежа путева, где чворове представљају градови, а путеви су везе између чворова, као што је приказано на слици 13.



Слика 13 – Пример неусмереног графа са градовима као чворовима

Парсирање улазних датотека реализовано је помоћу *ANTLR* (*ANother Tool for Language Recognition*) парсер генератора. За цртање графова коришћена је Јавина библиотека *JUNG* (*Java Universal Network-Graph Framework*), са неким имплементационим променама које су морале да се модификују. Код расплинуте логике коришћен је формат језика који се користи у контроли система – *FCL* (*Fuzzy Control Language*).

4.1.1. Опис библиотеке JUNG

JUNG представља библиотеку отвореног кода за моделовање, анализу и визуелизацију података у виду графова или мрежа. Библиотеку чине различити алгоритми за анализу структуре графова, својства графова, алгоритми распоређивања чворова у графу, алгоритми анализе социјалних мрежа и многи други алгоритми из сродних области. У реализованом систему *JUNG* је коришћен искључиво за део око графике и визуелизације графова. Библиотека садржи пакет за визуелизацију, који омогућава интерактивно обилажење исцртаних мрежа и обухвата већи број алгоритама обиласка чворова. Такође, могуће је имплементирати и сопствени алгоритам распоређивања.

Архитектура ове библиотеке омогућава реализацију усмерених и неусмерених графова, вишемодалних графова, графова са паралелним гранама и хиперграфова. Библиотека подржава механизам за анотацију графова, ентитета и релација помоћу метаподатака. Помоћу интерфејса *Graph<V,E>* може се додати и уклонити чвор или грана у графу, дохватити колекција чворова и грана или информација о чворовима који припадају означеној грани. За специфичне типове графова постоје још и следећи интерфејси: *DirectedGraph<V,E>*, *UndirectedGraph<V,E>*, *MultiGraph<V,E>*, *Tree<V,E>* и многи други. Чворови и гране у библиотеци *JUNG* могу бити објекти било ког типа, па чак и произвољне класе реализоване од стране корисника. Тако реализовани чворови и гране се могу користити у више од једног графа, што је добро за различите топологије које садрже исте чворове.

Библиотека *JUNG* обухвата два важна пројектна узорка – апстрактну фабрику (енг. *abstract factory pattern*) и трансформер (енг. *transformer pattern*). Први пројектни узорак користи се за креирање нових чворова и грана графа, ажурирања приликом визуелних промена у графу, затим код примене алгоритама распоређивања и код још неких операција. Други пројектни узорак омогућава превођење једног или више својстава једног објекта у неко својство или скуп својстава другог објекта, чији се тип разликује од иницијалног објекта.

Код визуелизација графова, *JUNG* користи концепте распореда чворова и грана графа и визуелизационе компоненте. Основна класа за приказивање графова је *BasicVisualizationServer<V,E>* која имплементира интерфејс *VisualizationServer<V,E>* и наслеђује класу *JPanel* из Јавине библиотеке *Swing*. Начин да се одабере одређени распоред чворова графа по визуелизационом простору остварује се помоћу интерфејса *Layout* или њему сродних класа. Овај интерфејс пружа механизме иницијализације позиција свих чворова у графу или везивања задатог чвора за неку одређену локацију. Важан интерфејс је и *Rendered*, који се користи за исцртавање грана, лабела грана, чворова и лабела чворова, док се додатни параметри као што су дебљина линија гране, боја чворова, позиције лабела и други параметри чувају и задају помоћу интерфејса *RenderContext*.

Библиотека *JUNG* помоћу графичког корисничког интерфејса интерагује са корисницима који одређене операције задају помоћу миша. Најважнији режими у којима миш може да ради са графовима ове библиотеке су: скалирање (енг. *zoom*), трансформација

(енг. *transformation*/*rotation*), транслација (енг. *translation*), анотирање (енг. *anotation*), ажурирање (енг. *edit*) и други режими из интерфејса *GraphMouse* и *GraphMousePlugin*.

4.1.2. FCL језик

FCL језик служи да опише улаз, излаз и рад једног система који користи расплинуту (енг. *fuzzy*) контролу. Типична *FCL* датотека садржи више функцијских блокова, који имају своје име. Систем може паралелно подржати рад са више функцијских блокова, али у сваком тренутку користи само један функцијски блок:

Приказ функцијских блокова код *FCL* датотеке

```
FUNCTION_BLOCK FunBlok1
...
END_FUNCTION_BLOCK

FUNCTION_BLOCK FunBlok2
...
END_FUNCTION_BLOCK
```

Унутар једног функцијског блока налази се неколико угнежђених блокова:

- Блок који садржи улазне променљиве или чињенице и који служи за декларацију свих улазних променљивих и њихових типова (целобројни или реални); Приказ блока са улазним променљивама:

```
VAR_INPUT
     temp: REAL;
     ...
END_VAR
```

- Блок који садржи излазне променљиве; Приказ блока за излазним променљивама:

```
VAR_OUTPUT
     brzina: REAL;
     ...
END_VAR
```

- Блок *Fuzzify* који служи за дефинисање расплинутих скупова над улазним променљивама и опционо опсега променљиве:

```
FUZZIFY
     TERM hladno := funkcija_clanstva;
     ...
END_FUZZIFY
```

- Блок *Defuzzify* који служи за дефинисање расплинутих скупова над излазним променљивама, као и подразумеване вредности, дефинисање метода дефазификације и опционо опсега променљиве:

```
DEFUZZIFY
     TERM spora := funkcija_clanstva;
     ...
     METHOD := metod_defuzzification;
     DEFAULT := 0;
     [RANGE  := (donja ... gornja);]
END_DEFUZZIFY
```

- Блок правила, који служи за дефинисање веће групе расплинутих правила, који служе за контролу система, као и метода акумулације и агрегације. Уколико постоје излазне променљиве, које нису дискретне, неопходно је дефинисати и метод активације. Систем треба да подржи постојање више блокова правила:

```
RULEBLOCK BlokPravila1
     [ACT : metod_aktivacije;]
     ACCU : metod_akumulacije;
     (AND | OR) : metod_agregacije;
     RULE 1: IF ... THEN ... [WITH ...];
     ...
END_RULEBLOCK

RULEBLOCK BlokPravila2
...
END_RULEBLOCK
```

4.1.2.1. Функције чланства

Функције чланства које су реализоване су следеће:

- Функција синглтон је функција која има вредност 1 у дефинисаној тачки x на x-оси и вредност 0 у свим осталим тачкама. Ова функција може да се користи само за излазне променљиве и најчешће постоји да би поједноставила процес дефазификације. Дефиниција ове функције изгледа овако:

```
TERM ime_Funkcije := x;
```

- Троугаона функција је функција која има вредност 0 на сегментима где је $x < x_1$ и $x > x_3$, на сегменту $x_1 \leq x < x_2$ линеарно расте, а на сегменту $x_2 < x \leq x_3$ линеарно опада. Функција има максимум у тачки $x = x_2$. Дефинише се са:

```
TERM ime_Funkcije := (x1, y1), (x2, y2), (x3, y3);
```

- Трапезоидна функција је функција која има вредност 0 на сегментима где је $x < x_1$ и $x > x_4$, на сегменту $x_1 \leq x < x_2$ линеарно расте, а на сегменту $x_3 < x \leq x_4$ линеарно опада. Функција има максимум на сегменту $x_2 \leq x \leq x_3$. Дефиниција ове функције изгледа овако:

```
TERM ime_Funkcije := (x1, y1), (x2, y2), (x3, y3), (x4, y4);
```

- Рамена функција (енг. *shoulder*) је функција која има вредност y_1 за $x \leq x_1$, вредност y_2 за $x \geq x_2$ и линеарно расте на сегменту $x_1 < x < x_2$. Дефиниција ове функције изгледа овако:

```
TERM ime_Funkcije := (x1, y1), (x2, y2);
```

4.1.2.2. Метод дефазификације

Дефазификација или концентрација представља просец добијања конкретне вредности. Постоји више метода дефазификације:

- *COG* – центар гравитације;
- *COGS* – центар гравитације, уколико су излазне промењљиве дискретне;
- *COA* – центар површине;
- *RM* – десни максимум;
- *LM* – леви максимум.

4.1.2.3. Метод агрегације

Агрегација је поступак евалуације подуслова, односно рачунање чланства за конкретну улазну промењљиву у подусловима и агрегирање тих чланстава у једну вредност, која изражава истинитост целог услова. Спецификација језика *FCL* даје три методе за агрегацију *AND* и *OR* веза подуслова, да би се задовољио Де Морганов закон:

1) метода за *OR* максимум (*MAX*) и за *AND* минимум (*MIN*);

2) метода за *OR* суму (*ASUM*) и за *AND* производ;

3) метода за *OR* ограничену суму (*BSUM*) и за *AND* ограничену разлику (*BDIF*).

4.1.2.4. Метод активације

Активација је процес којим се формира функција чланства за излазну променљиву на основу вредности истинитости услова. Спецификација језика *FCL* има две методе за активацију:

1) метода минимума (*MIN*), што означава да функција чланства активираног скупа добија као минимум вредности истинитости услова и вредности функције чланства конкретног скупа над излазном променљивом;

2) метода производа (*PROD*), што означава да функција чланства активираног скупа добија као производ вредности истинитости услова и вредности функције чланства конкретног скупа над излазном променљивом.

4.1.2.5. Метод акумулације

Акумулација је процес којим се од активираних скупова над излазним променљивама добијају акумулиране функције чланства. Спецификација језика *FCL* има три методе за акумулацију:

1) максимум (*MAX*), која се рачуна као максимум свих активираних функција чланства над једним скупом над излазном променљивом;

2) ограничена сума (*BSUM*), која се рачуна као максимум вредности 1 и суме свих активираних функција чланства над једним скупом над излазном променљивом;

3) нормализована сума (*NSUM*), која се рачуна као разломак суме свих активираних функција чланства над једним скупом над излазном променљивом и максимума вредности 1 и суме свих активираних функција чланства над једним скупом над излазном променљивом.

4.1.2.6. Опис блока правила и правила

Правила унутар датотеке су подељена по блоковима, а сваки блок има своје јединствено име. Свако правило унутар блока има свој јединствени број. Правило се састоји из услова и последице. Уколико је услов сачињен од више подуслова користе се операције *AND* и *OR*. Подуслов има облик *Lingvisticka_Promenljiva IS Lingvisticki_Opis*. Уколико је подуслов негација, тада се *IS* мења са *IS NOT*. Други начин дефинисања подуслова је

користећи прилоге: врло (енг. *very*), нешто (енг. *somewhat*), заиста (енг. *indeed*) и екстремно (енг. *extremly*). Прилогом врло постиже се концентрација и одређен је формулом:

$$\mu_{CON}(x) = \mu^2(x)$$

Прилогом нешто постиже се дилатација и одређен је формулом:

$$\mu_{DIL}(x) = \mu^{\frac{1}{2}}(x)$$

Прилогом заиста постиже се интензификација и одређен је формулом:

$$\mu_I(x) = \begin{cases} 2\mu^2(x), 0 \leq \mu(x) \leq 0.5 \\ 1 - 2(1 - \mu(x))^2, 0.5 \leq \mu(x) \leq 1 \end{cases}$$

Прилог екстремно је сличан прилогу врло, само се вредност чланства ставља на трећи степен:

$$\mu_E(x) = \mu^3(x)$$

4.2. Архитектура мобилне апликације

Као саставни део система *SAIL*, осим рачунарске верзије, развијена је и апликација за мобилне уређаје. Ова мобилна апликација омогућава корисницима система да самостално уче на својим мобилним уређајима. Мобилна верзија *SAIL* имплементирана је коришћењем програмског језика Андроид, који користи Јава програмски језик и *XML* технологију.

За разлику од рачунарске верзије, код мобилне верзије није било задовољавајуће библиотеке за исцртавање графова и стабала, па је приликом развоја било потребно реализовати сопствени метод исцртавања. Пре реализације новог начина исцртавања графова на Андроид платформи, анализиране су већ описана библиотека *JUNG*, која је коришћена за рачунарску верзију апликације, и библиотека *Graphiz* [68], која такође служи за генерисање графова. Анализом је утврђено да ове библиотеке користе пакет за графички интерфејс *AWT* (*Abstract Window Toolkit*) и многобројне *JavaFX* пакете, који нису могли да се примене у програмирању мобилне верзије симулације графова на Андроид платформи. Једини начин да се ово разреши био би да постоји серверски део система који би генерисао графове, а затим у виду слике слао клијентском делу система, односно апликацији на мобилном уређају. Како је почетни циљ био да целокупан софтверски систем буде независан од интернет конекције, одустало се од овог решења да се прави клијент-сервер архитектура.

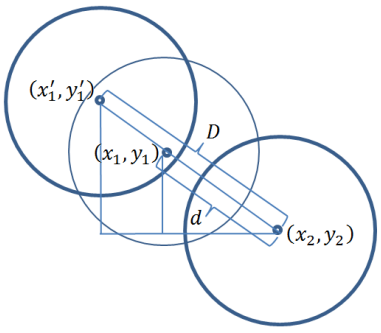
Код реализованог новог начина исцртавања, стабло је скалирано тако да цело може да стане на величину екрана уређаја, али дозвољена је могућност да се делови стабла увећају покретом руке преко корисничког екрана (енг. *pinch to zoom*). Од могућих операција са компонентама подржане су следеће: додавање новог чвора у граф, повлачење линије од чвора који је корисник тренутно одабрао, померање чвора који је корисник одабрао и измена својстава чвора. С обзиром да није било могуће разликовати операцију повлачења линије између чворова, од изворног до одредишног чвора и операцију премештања чвора, уведена су два стања: стање уноса графа и стање промене над графом.

Додиром на дугме унапред, посећује се чвор стабла, односно почетни чвор из унетог графа. Када се обиђе чвор, у том кораку се у стабло претраге уносе новооткривени чворови. Тренутно обрађени чвор је обележен црвеном бојом, посећени чворови су зелене боје, док су непосећени чворови сиви.

Приликом додавања чвора у граф, било је потребно проверити да ли се чворови преклапају, односно да ли је растојање између центара чворова мање од *2R*. Одабрано је растојање између постојећих чворова и новог додатог чвора буде *2R + R/8*, односно да постоји размак између чворова. Додавањем новог чвора постојала је могућност да се померањем преклопе неки други чворови, што доводи до уланчане пропагације преклапања када на радној површини имамо већи број чворова. Начин на који је решено преклапање чворова је илустровано на слици 14, тако што се из сличности троуглова рачунају нове координате центра чвора. Нове координате центра биле би:

$$x_1' = x_2 + \frac{D}{d}(x_1 - x_2)$$

$$y_1' = y_2 + \frac{D}{d}(y_1 - y_2)$$



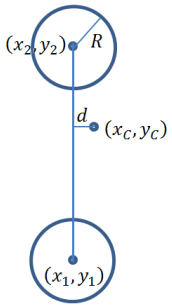
Слика 14 – Сличност троуглова за одређивање фактора помераја чвора

Други велики проблем код мобилне верзије било је детектовање везе која повезује два чвора. Постоје два случаја која се јављају: када су чворови на истој вертикалној линији ($x_1=x_2$) и када нису ($x_1\neq x_2$). У првом специјалном случају било је потребно проверити да ли је растојање од праве мање од *R/2* (слика 15):

$$d = |x_c - x_1| < \frac{R}{2}$$

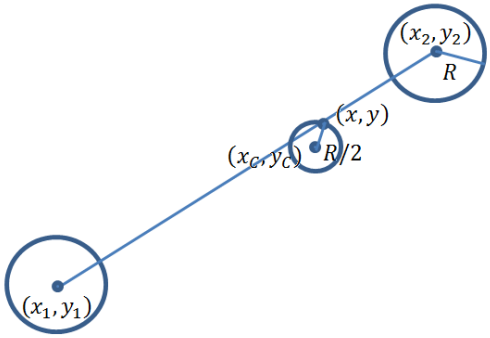
уз услов да је задовољено: $f = \frac{y_c - y_1}{y_2 - y_1}$ и да је $0 < f < 1$.

Фактор *f* означава да се тачка у односу на коју меримо растојање налази на дужи која спаја центре чворова. Фактор је одабран због своје инваријантности у односу на позиције чворова и означава само да ли је тачка између центара два чвора. Пошто се чворови не преклапају, не постоји могућност да имамо операцију дељења са нулом.



Слика 15 – Специјални случај два чвора на вертикалној линији

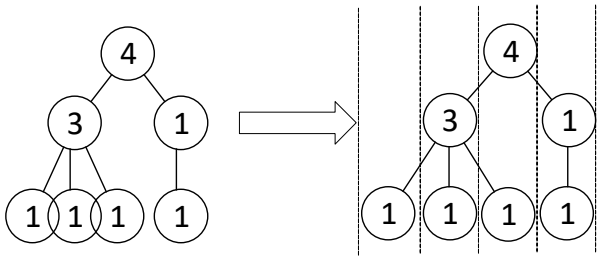
У општем случају, $x_1\neq x_2$, па је проблем решен увођењем круга полупречника *R/2* око тачке додира и провером да ли тај круг сече линију која спаја чворове, као што је приказано на слици 16.



Слика 16 – Начин дохватања везе између чворова на екрану мобилног уређаја

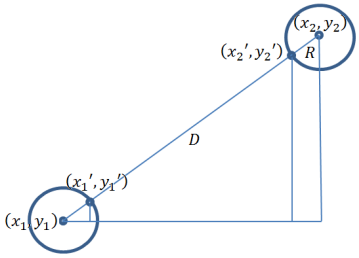
Трећи велики проблем код мобилне верзије представљала је ограничена висина и ширина екрана уређаја, па је унос неких параметара морао да се реализује у корацима. Такође, исписивање резултата корака алгоритма је у неким симулацијама приказивано у више екрана или тек након што корисник притисне одређено дугме за приказ коментара тог корака.

Исцртавање стабла претраживања није једноставно, пошто је за сваки чвор било неопходно израчунати његову позицију у стаблу, али и позиционирати његове чворове наследнике, односно децу, који су морали да се налазе на подједнаким удаљеностима од родитељског чвора. Осим тога код многих алгоритама постоји могућност да се током симулације вратимо на неке непосећене чворове у стаблу или да развијамо нове чворове, што би довело да стабло буде деформисано и неоптимизовано. Из тог разлога одлучено је да се за комплетно стабло израчунају све позиције чворова унапред и да се утврди максимална ширина стабла. Максимална ширина стабла представља збир свих могућих листова стабла, а сваки чвор лист би да се не би сударали добио своју траку приликом исцртавања, као што је приказано на слици 17.



Слика 17 – Начин позиционирања чворова у стаблу са максималном ширином

Симулација се приказује тако што се прво исцртавају чворови, а након тога везе. Тачке центара транслиране су на ивицу кружнице. Израчунавање крајњих тачака на таквој дужи између два чвора приказана је на слици 18.



Слика 18 – Начин исцртавања везе између ивица два чвора

Ако је полупречник чвора R, растојање између чворова $D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, вредност за апсцисе и ординате крајњих тачака везе између чворова утврдиће се на основу сличних троуглова, из којих ће се извести следеће релације:

$$\frac{x_1' - x_1}{x_2 - x_1} = \frac{R}{D} \quad => \quad x_1' = x_1 + \frac{R}{D}(x_2 - x_1)$$

$$\frac{x_2' - x_1}{x_2 - x_1} = \frac{D - R}{D} \quad => \quad x_2' = x_1 + (1 - \frac{R}{D})(x_2 - x_1)$$

$$y_1' = y_1 + \frac{R}{D}(y_2 - y_1)$$

$$y_2' = y_1 + (1 - \frac{R}{D})(y_2 - y_1)$$

4.3. Примена игара у едукацији (гејмификација)

Приликом прављења софтверских система врло је важно водити рачуна о дизајну корисничког интерфејса. Све већи број софтверских система реализује се као дводимензионално или тродимензионално окружење са визуелизацијом одређеног проблема [69], као компјутерске игре [70] [71] или као симулације применом виртуелне стварности и специфичних хардверских уређаја [72].

Гејмификација у едукацији је процес који помаже ученицима и студентима да уче различита знања кроз игре [73]. Због тога све већи број наставника разматра увођење игара у образовање, коришћењем нових техника и алата [74] [75]. Изазов савременог образовања на факултетима подразумева држање пажње студентима током учења, стимулисање њихових интересовања, све то реализовано кроз богато корисничко окружење, које подстиче повратну информацију и јачање интеракције како између наставника и групе студената, тако и између самих студената [76].

Истраживање показује да се у циљној популацији мобилни телефон често користи као средство за играње визуелних игара [77], па су у оквиру овог истраживања развијене и две анимације алгоритама имплементираних помоћу *Unity 3D*, вишеплатформског алата за развој видео игара и анимација за *PC* рачунаре и веб сајтове, конзоле и мобилне уређаје. *Unity* користи следеће графичке *API*-је: *Direct3D* на платформама *Windows* и *Xbox360*, *OpenGL* на платформама *Windows* и *Mac*, *OpenGL ES* на платформама *Android* и *iOS* и *proprietary API* на конзолама за видео игре. *Unity* подржава мапирање испупчења, мапирање рефлексије,

мапирање паралаксе, амбијенталну оклузију *SSAO*, динамичке сенке, рендеровање на текстурама. У *Unity* се могу писати и специфични програми за прављење сенки (енг. *shader*). Од формата датотека овај алат подржава формате следећих апликација за графику: *3D Studio Max*, *Maya*, *Softimage*, *Blender*, *Adobe Photoshop*, *Adobe Fireworks*, *Cinema 4D*, *Zbrush*, *Modo* и многих других апликација. Датотеке ових апликација могу се додати у пројекат и њима се може управљати директно кроз графички интерфејс алата *Unity*. За писање пројеката могу се користити програмски језици *JavaScript*, *C#* или *Boo*, који је веома сличан синтакси програмског језика *Python*.

Прва реализована анимација у оквиру предмета Интелигентни системи била је „Мисионари и људождери“, за представљање простора стања. Анимација омогућава решавање овог интересантног проблема, приказује стабло одлучивања и решавање проблема режимом корак по корак. Анимација је направљена коришћењем програмског језика *C#* и *Unity 3D*, а тестирана је на три платформе: *Android*, *iOS* и *Web Player* за веб прегледаче. Од библиотека коришћене су системске колекције *Unity*-ја и библиотека *Linq* за лакши рад са колекцијама.

Проблем „Мисионари и људождери“ гласи:

Три мисионара и три људождера се налазе на једној обали реке коју треба да пређу. На располагању имају чамац у који стаје највише две особе. Ако у неком тренутку број људождера надмаши број мисионара на једној или на другој обали, људождери ће појести мисионаре. Циљ је да сви безбедно пређу са једне на другу страну реке.

На слици 19 приказано је иницијално стање анимације, када су сва три мисионара и сва три људождера на левој обали реке. Мисионар и људождер се убацују у чамац притиском левог дугмета миша и на исти начин се избацују из чамца. Притиском на дугме за кретање чамца „*MOVE*“ (или алтернативно на тастатури помоћу тастера *M* или тастера за размак), мисионар и људождер се пребацују са једне обале реке на другу. Притиском на дугме „*RESET ALL*“ анимација се враћа на иницијално стање. Опцијом „*STEP BY STEP*“ приказује се корак по корак анимације, а сваки следећи корак се може видети притиском на дугме „*NEXT STEP*“ (или алтернативно на тастатури помоћу тастера *S* или стрелица на горе или у десно).

Слика 19 – Иницијално стање анимације „Мисионари и људождери“

На слици 20 приказано је стање анимације након два корака и графа који се претражује. Стање се може дефинисати као уређена тројка (М, Љ, Ч), где:

- М представља број мисионара на левој обали реке и може имати вредност: $М \in \{0,1,2,3\}$,
- Љ представља број људождера на левој обали реке и може имати вредност: $Љ \in \{0,1,2,3\}$,
- Ч представља стање у коме се налази чамац и може имати вредности: 1, уколико је чамац на десној обали реке, или 0, уколико је чамац на левој обали реке.

Потребно је дефинисати и операторе претраге, како би се представио комплетан граф претраге, пронашла достижна стања из почетног стања и пронашла најоптималнија решења. Оператор 1М представља прелазак једног мисионара у чамцу са једне обале на другу, оператор 1М1Љ представља прелазак једног мисионара и једног људождера, оператор 2Љ представља прелазак два људождера, итд.

Коришћењем ове анимације студент може да у сваком кораку види граф и потенцијалне чворове које може да обиђе у следећем кораку, применом одређеног оператора.

Из почетног стања (чвора) може се приметити да постоје три чвора следбеника. У почетном кораку могућ је избор једног од два оператора, који би играча одвели до оптималне путање.

Слика 20 – Приказ анимације са графом који се претражује

Комплетан граф претраге дат је на слици 21. Све гране овог графа могу да се сматрају двосмерно оријентисаним, пошто применом истог оператора корисник се може кретати кроз граф у оба смера промене стања. Решење проблема је да се из почетног чвора (3,3,0) пређе у циљни чвор (0,0,1). При налажењу решења све затворене путање у графу су елиминисане. Постоје 4 оптималне путање у графу, а свака путања захтева 11 прелазака чамца преко реке.

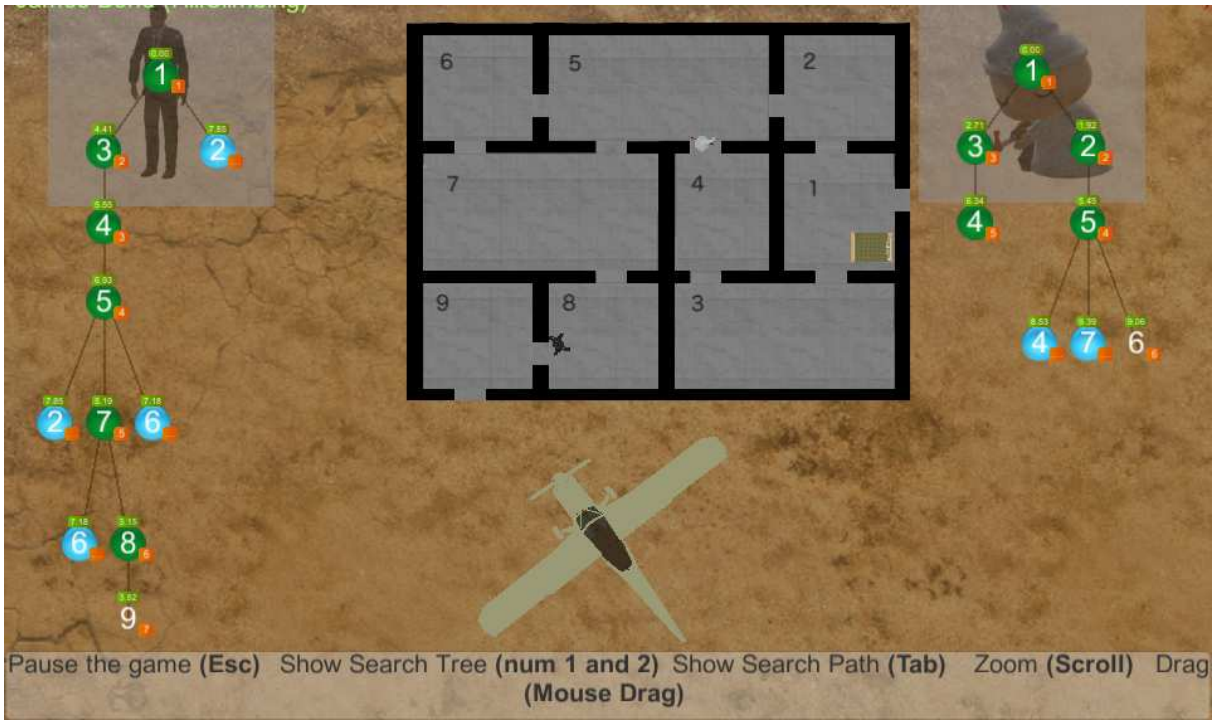
Слика 21 – Комплетан граф претраге у игри „Мисионари и људождери“

Друга реализована игра названа је „Џејмс херој“ и заснована је на већ описаним алгоритмима претраживања, који могу да се извршавају у паралели, са циљем компаративне анализе од стране студената. Једним алгоритмом управља Џејмс, чији је циљ да пронађе излаз из лавиринта, а другим његов пратилац стрелац, који жели да га улови, као што је приказано на слици 22. Постоји шест реализованих различитих тлоцрта лавиринта, односно шест нивоа игре. Игра нуди два режима играња: аутоматски режим извршавања алгоритама, са упоредним приказом, и играчки режим, у коме једним карактером Џејмсом може да управља корисник игре, док га други играч као стрелац може спречавати да изађе из лавиринта, покушавајући да му нанесе штету пуцањем. За студенте је битнији овај први режим, јер се у сваком кораку у коме је неки карактер променио позицију, та промена одмах приказана и са стране, у стаблима претраживања.

Слика 22 – Надоградња система *SAIL* у игру „Џејмс херој“

На слици 23 приказан је један ниво са упоредном анализом два стабла претраживања, играч Џејмс примењује алгоритам планинарења, а други играч као стрелац, који га јури, примењује алгоритам гранања и ограничавања. Могуће је изабрати одређену врсту реализованих алгоритама претраживања, подешавати брзине оба играча, као и предност у

броју корака за одређеног играча, што игру чини прилагодљивом за играње и учење алгоритама претраживања.



Слика 23 – Имплементација алгоритама претраживања коришћењем *Unity 3D* окружења и упоредна анализа два алгоритма

Предност алата *Unity 3D* је што се апликација може прилагодити било којој савременој платформи, па на врло једноставан начин од апликације прилагођене за мобилни уређај са оперативним системом *Android* можемо добити и апликацију за веб прегледаче на оперативном систему *Windows*. Прозори и објекти се прилагођавају величини екрана уређаја и приказују се коректно. Све је већи број игара који користе овакве технологије и развојне алате за вишеплатформски развој, али се за развој игара у образовању ове технологије не користе много. Постоји неколико истраживача који су имплементирали по један или неколико алгоритама, помоћу оваквих развојних алата, и применили их у едукативне сврхе, али анализиране анимације алгоритама нису интерактивне и нису предвиђене за учење [78] [79]. У едукацији се осим игара на рачунарима и на мобилним уређајима све више развијају игре и окружења која користе виртуелну стварност (енг. *virtual reality*, *VR*) и проширену стварност (енг. *augmented reality*, *AR*) [80] и истраживања показују да ће то бити потпуно нови правац примене вештачке интелигенције и паметних система у образовању.

5. Опис рада система

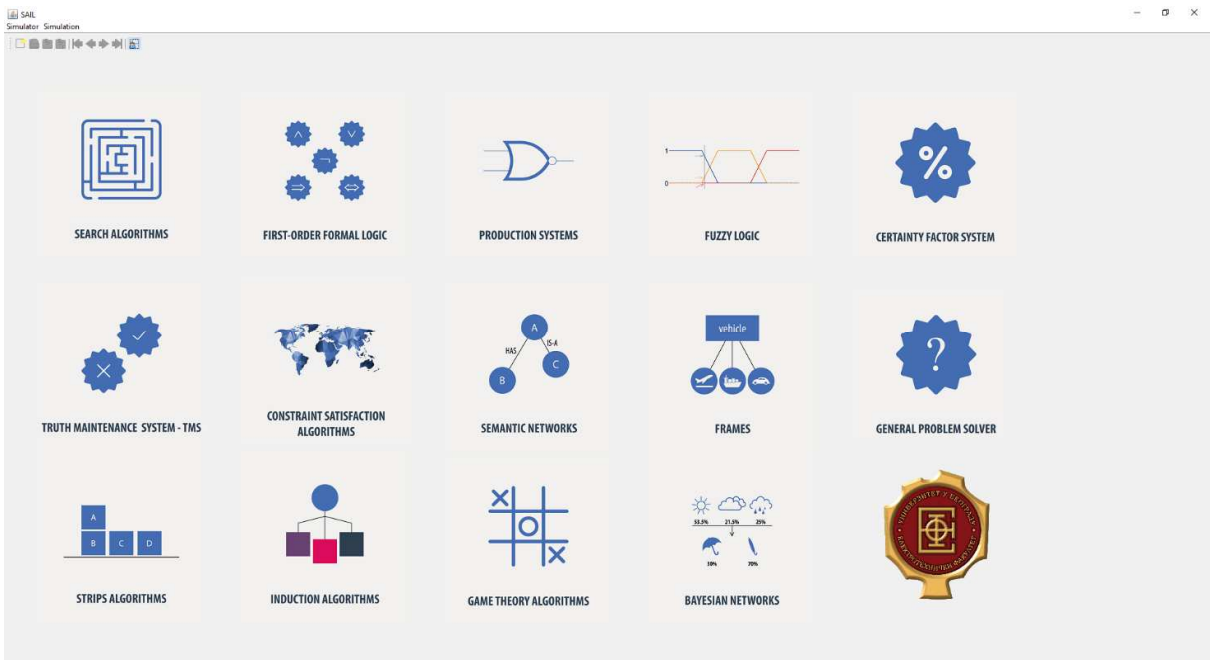
Реализовани софтверски систем *SAIL* обухватио је 30 алгоритама, који су по својој природи веома разнолики. Због постављених захтева било је потребно реализовати заједнички кориснички интерфејс, како корисници система не би морали да се привикавају на различита окружења.

Главне карактеристике система *SAIL* су следеће:

- унос података и примера од стране корисника система, кроз интуитивни кориснички интерфејс;
- провера унетих података од стране система;
- симулирање одређеног алгоритма и праћење симулације методом корак по корак;
- симулирање одређеног алгоритма извршавањем симулације до краја;
- учитавање примера из предефинисаних текстуалних датотека;
- снимање примера у текстуалне датотеке;
- визуелизација и анимација извршеног алгоритма, уз преглед свих структура података, које су неопходне за извршавање таквог алгоритма, као и увид у стање тих структура у реалном времену;
- сваки корак симулације дат уз детаљан визуелни и/или текстуални опис, а свако следеће стање у систему наглашено (енг. *highlighted*);

- могућност генерисања извештаја о примеру, извршеном алгоритму и комплетном поступку решавања примера, у формату *PDF* фајла.

Први корак који студент треба да уради у систему је да одабере алгоритам који жели да изврши. *SAIL* обухвата групу стандардних алгоритама претраживања, групу алгоритама из области теорије игара, алгоритме формалне логике првог реда, алгоритме продукционих система, семантичке мреже, оквире, алгоритме за решавање проблема – *GPS* и *STRIPS*, Бајесове мреже, три врсте алгоритама за рад у неизвесном окружењу и индукционе алгоритме. На слици 24 приказан је уводни кориснички екран система *SAIL*.



Слика 24 – Уводни екран софтверског система за избор алгоритма

SAIL садржи два менија: Симулатор и Симулација. Мени Симулатор садржи опције везане за избор одређеног алгоритма и за рад са датотекама:

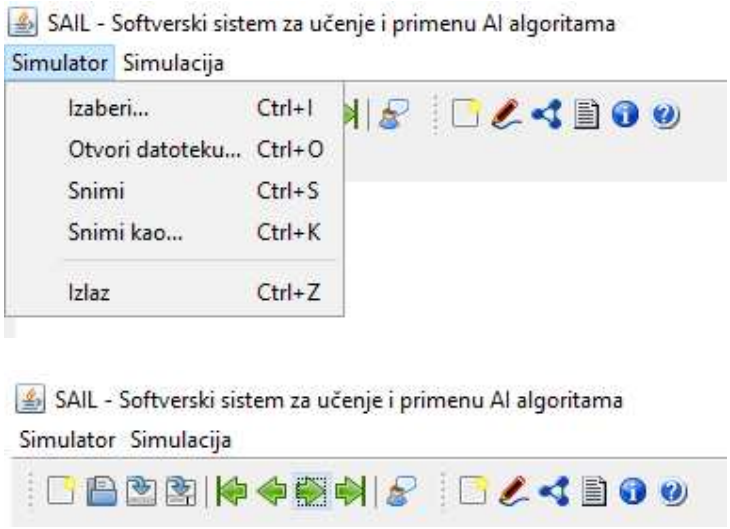
- Изабери – опција којом се отвара дијалог за избор алгоритма;
- Отвори датотеку – опција којом се отвара стандардни дијалог за избор фајла који треба учитати;
- Сними – опција којом се извршава снимање тренутне симулације у учитани фајл;

- Сними као – опција којом се отвара стандардни дијалог за снимање тренутне симулације у потпуно нови фајл или преко садржаја већ постојећег фајла;
- Затвори – опција којом се затвара главни прозор и излази из система *SAIL*.

Мени Симулација садржи опције везане за контролу тока симулације:

- На почетак – опција којом се симулација враћа на почетак рада алгоритма;
- На крај – опција којом се симулација поставља на крајње стање;
- Корак напред – опција која помера симулацију за један корак унапред;
- Корак назад – опција која помера симулацију за један корак уназад.

Поред менија, све наведене опције се могу пронаћи и на помоћној траци са алатима, која се налази одмах испод менија. Опције и помоћна трака са алатима приказане су на слици 25.



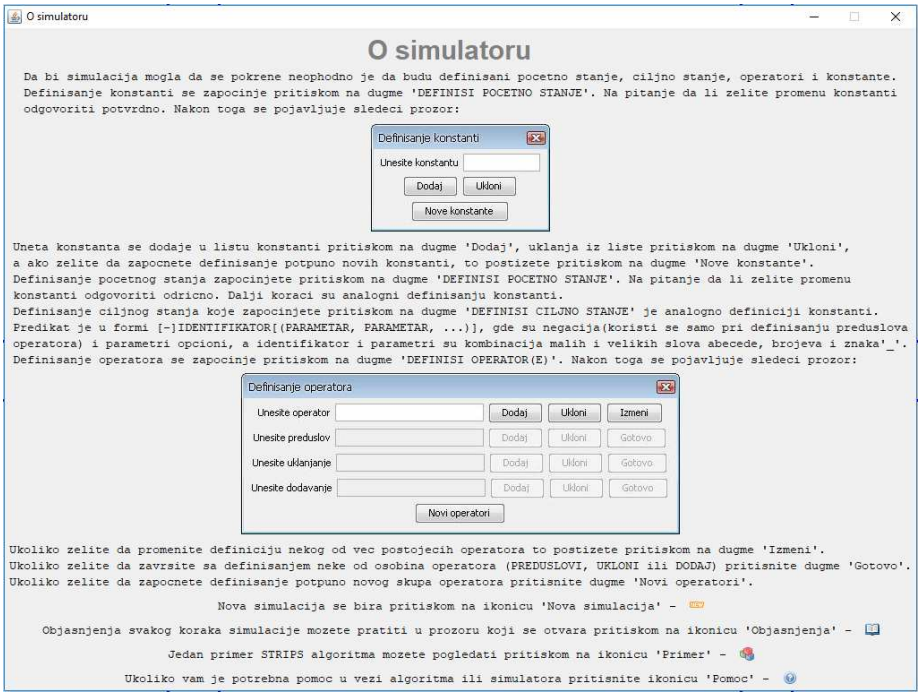
Слика 25 – Приказ падајућег менија и траке са алатима, уз додатне команде симулације алгоритама претраживања

Све алгоритме које решавамо можемо поделити у три групе: (1) алгоритме у чијим решењима приказујемо само текстуални опис и неки вид једноставне графике, (2) алгоритме чија решења су заснована на приказивању графа и/или стабла, уз текстуални опис, и (3) алгоритме који имају специфични визуелни приказ, као што су игре X-O, померање блокова, бојење мапа, укрштене речи, Судоку и сличне игре.

Када се одређени алгоритам одабере, могу се учитати претходно снимљени примери или се у симулацију могу унети параметри у виду новог примера. Параметри се чувају у текстуалним датотекама, са специфичним типовима (екстензијама) у зависности која врста алгоритма или група алгоритама је у питању.

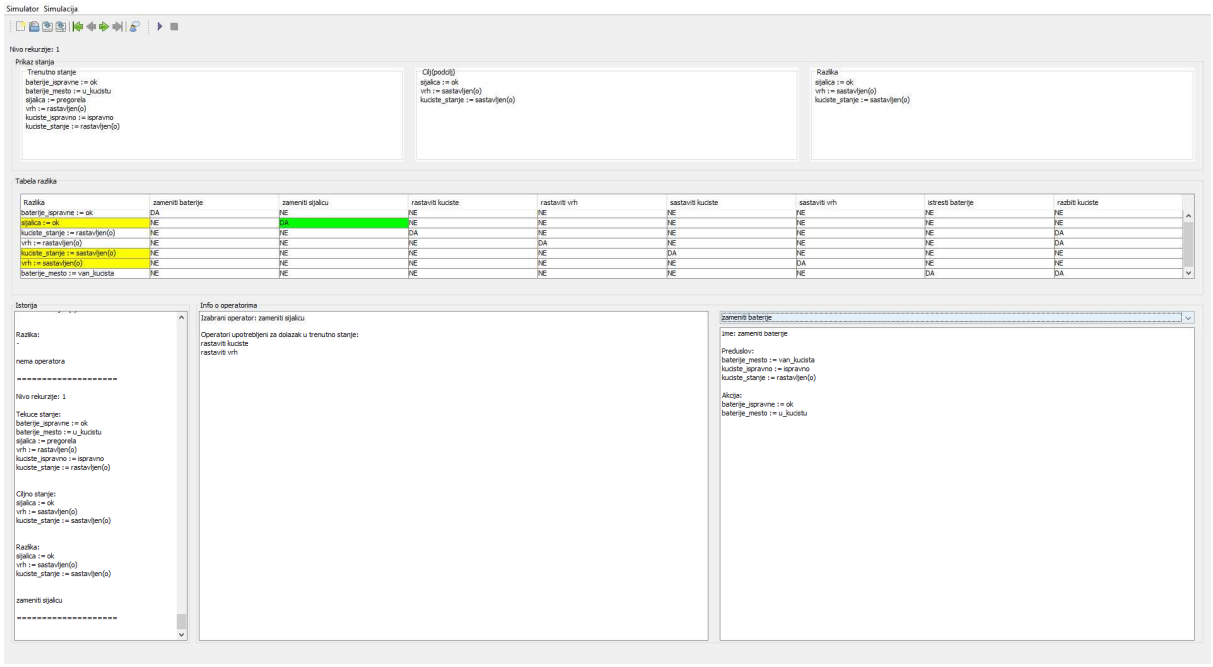
Унос улазних параметра симулације најчешће се врши у посебном прозору, на почетку рада одређеног алгоритма. Пошто рад алгоритма зависи од секвенце корака која се извршава, код већине алгоритама промена параметара током симулације није могућа.

Главни прозор подељен је на неколико целина и служи за праћење релевантних стања симулације и исписивање излазних података, са могућношћу да корисник испрати сваки корак алгоритма који учи. У сваком кораку алгоритма корисник визуелно и текстуално види како се стање променило, како би на крају симулације могао да разуме шта је циљ траженог алгоритма који се извршио и како он ради. Уколико не зна теоријске концепте о самом алгоритму и који се све подаци прате кроз његово извршавање, корисник може да их научи коришћењем менија за помоћ. Помоћни мени садржи једну ставку о самом алгоритму или групи алгоритама и једну ставку о самој симулацији, како се извршава. Пример менија за помоћ, који се увек отвара у новом прозору, дат је на слици 26.



Слика 26 – Пример помоћног менија са описом једне од симулација

На текстуалном приказу решавања засновани су алгоритми *GPS* и *STRIPS*. Њихов циљ је да покажу начин да се креира систем који би био способан за решавање проблема, па се ови алгоритми приказују на уводним часовима из предмета Интелигентни системи. *STRIPS* алгоритам, осим текстуалног приказа извршавања алгоритма може приказати и анимацију кроз пример са светом блокова. Циљ анимације је да се покаже рад алгоритма тако што треба блокове из почетног стања, довести у циљно стање коришћењем дозвољених оператора. Корисник у сваком тренутку симулације види промену која се извршила и који је следећи корак алгоритма, или сви могући кораци, односно стања у која симулација може да пређе. Пример текстуалног приказа *SAIL* симулације дат је на слици 27. Такође, текстуални приказ са мањим графичким приказом карактерише све симулације за рад у неизвесном окружењу - расплинуту логику, фактор извесности и *TMS*. Формална логика првог реда реализована је искључиво текстуалним приказом.

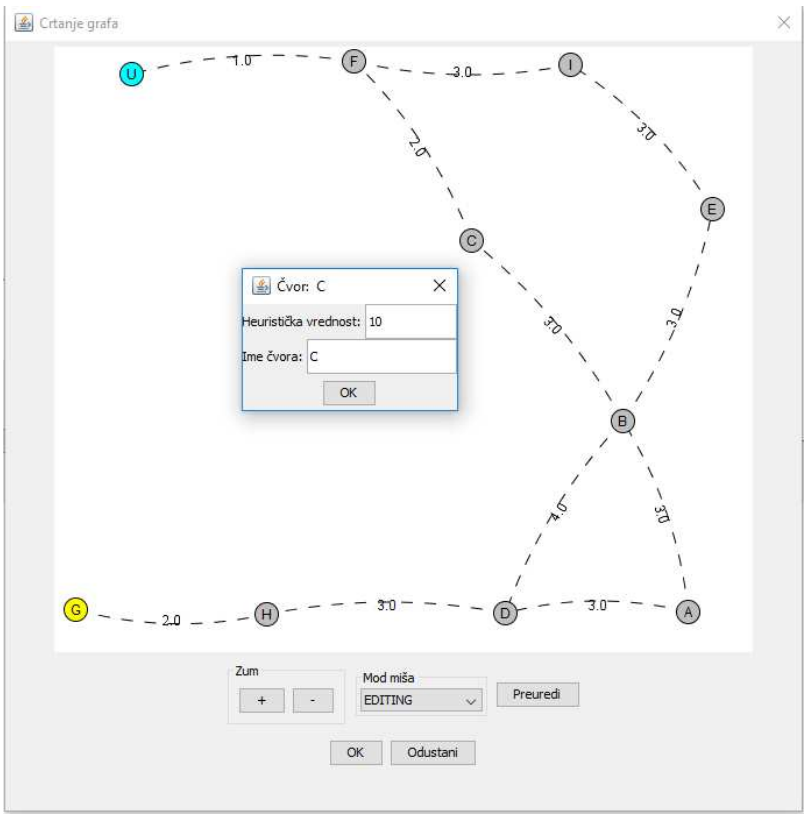


Слика 27 – Извршавање примера коришћењем *GPS* алгоритма

Највећи број реализованих алгоритама користи графове. Алгоритми претраживања који су представљени у систему *SAIL* имају за циљ да нађу путању од почетног до циљног чвора у графу. Ради бољег праћења симулације систем приказује стабло претраживања. Одређени алгоритми претраживања користе хеуристичку функцију, а у *SAIL* је реализован

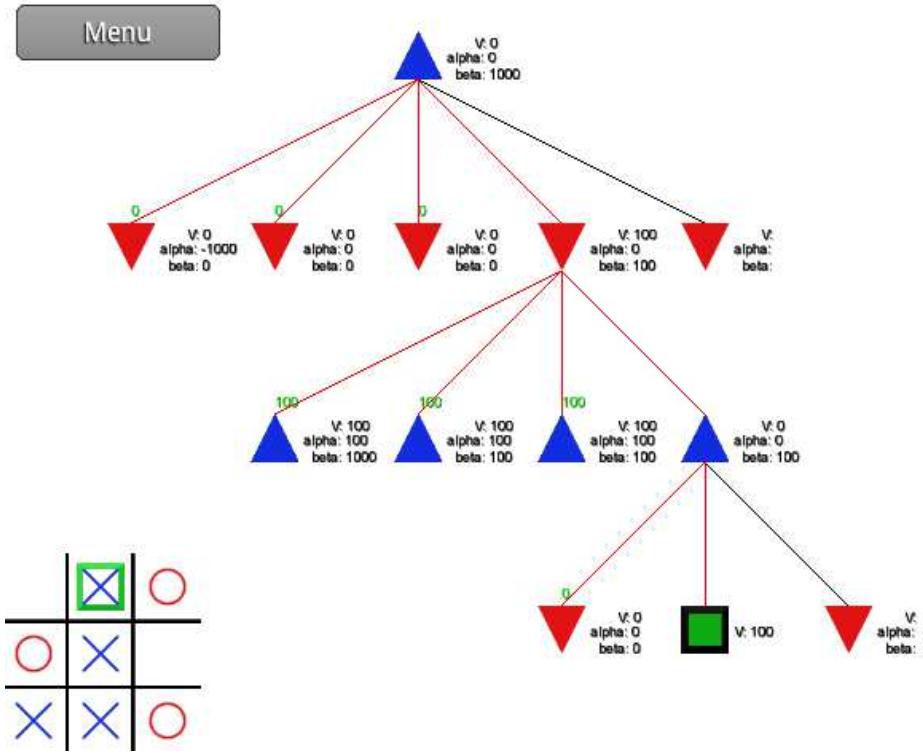
принцип динамичког програмирања, који у комбинацији са реализованим алгоритмима најчешће извршава алгоритам у мање корака и даје боље перформансе претраге.

Цртање графа је врло једноставно и интуитивно. Потребно је нацртати чворове, затим чворове повезати везама и означити који је почетни и који је циљни чвор. Постоје три режима рада за цртање графова: ажурирање графа, дохватање одређеног чвора и трансформација графа. Када се граф ажурира, кликом миша на чвор можемо променити име чвора и доделити хеуристичку вредност том чвору. Уклањањем чвора бришу се и све везе између тог чвора и других чворова графа. Веза се ствара тако што се након клика миша курсор миша превуче од једног чвора према другом чвору. За сваку везу између два чвора може се доделити цена (енг. *path cost*) и опционо се може изабрати усмереност (енг. *directionality*). Други режим рада је одабир чвора. У овом режиму када се чвор одабере може да се помера у циљу добијања бољег визуелног изгледа графа. Опција визуелног приказа графа може и аутоматски да се реаранжира у оквиру прозора у коме се приказује граф. Приказ цртања графа од стране корисника и ажурирање вредности чвора, односно његовог имена и хеуристичке вредности, дато је на слици 28.



Слика 28 – Кориснички екран за цртање графа

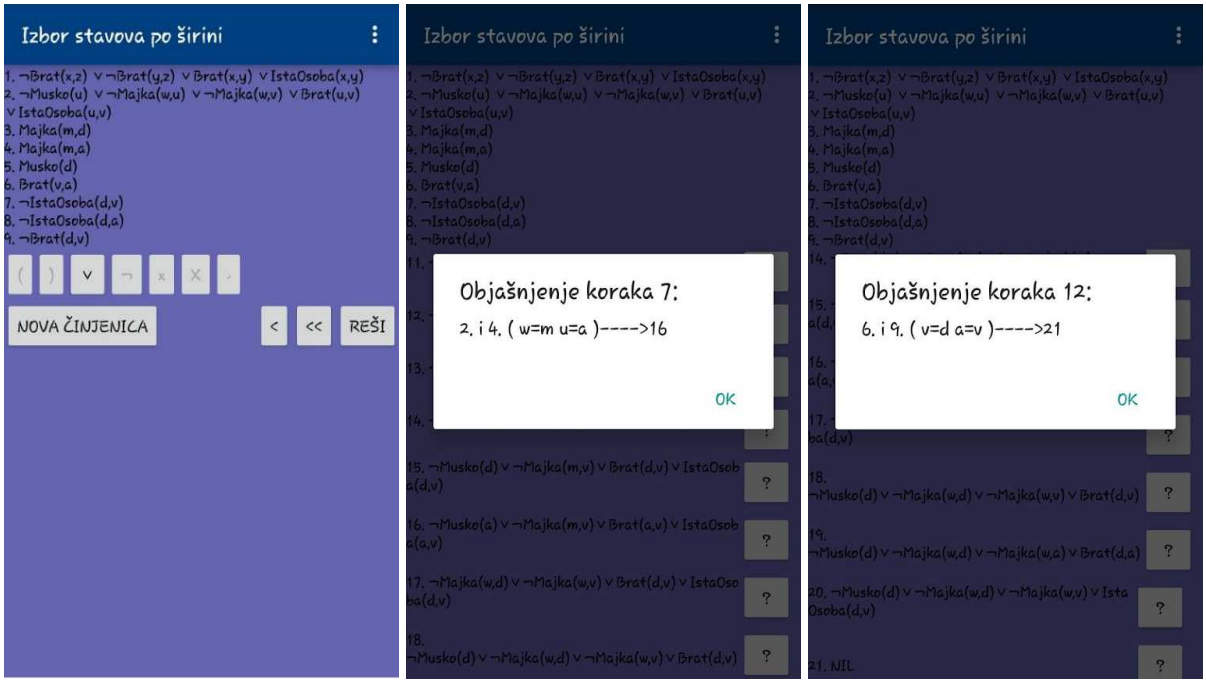
Алгоритми теорије игара најчешће се представљају потезним играма, које се састоје од простора проблема, почетног стања и циљног стања. Чворови стабла представљају позиције у игри, а гране представљају могуће потезе који преводе игру из једног стања игре у друго. Корен стабла увек означава почетно стање, а листови означавају терминална стања – победу, нерешен исход или пораз. У систему *SAIL* реализоване су игре мање комплексности, игра *X-O* (икс-окс) и игра *Nine man morris*, да би корисници система могли лакше да разумеју алгоритам *Minimax* и алгоритам *Minimax* са алфа-бета одсецањем. На слици 29 дат је приказ игре *X-O* и једног дела стабла игре у мобилној верзији система *SAIL*.



Слика 29 – Стабло игре *X-O* у оквиру мобилне апликације *SAIL*

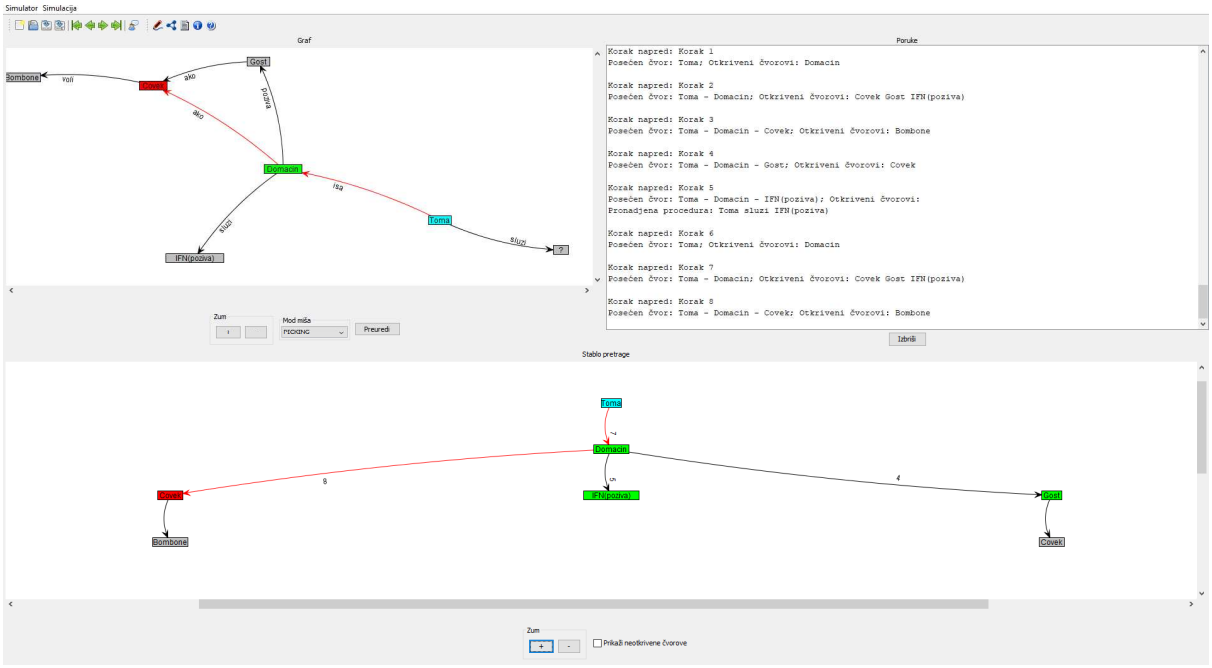
У систему *SAIL* имплементирана су четири алгоритма која се користе при закључивању у логици првог реда: *CNF* алгоритам и три алгоритма резолуције. Ови алгоритми припадају групи алгоритама у којима је важан текстуални приказ, посебно због специфичних симбола математичке логике. Први корак у овој симулацији је да формула буде трансформисана у *CNF* облик. Подржани су следећи симболи: $AND, OR, NOT, (,), \Rightarrow, \forall, \exists$. Када се достигне клаузална форма, добијене чињенице постају улазни подаци за одређени алгоритам резолуције који се може извршити у симулацији: алгоритам са стратегијом скупа

подршке, алгоритам са стратегијом првенства јединица и алгоритам са стратегијом избора ставова по ширини. Пример алгоритма резолуције, са стратегијом избора ставова по ширини, који се приказује у мобилној верзији система *SAIL* приказан је на слици 30.



Слика 30 – Пример алгоритма резолуције

Неформално знање је у систему *SAIL* реализовано семантичким мрежама и оквирима и користи графичку презентацију сличну као код алгоритама претраживања. Корисник када дефинише семантичку мрежу уноси објекте и релације између њих, што се може видети на слици 31. Објекти су приказани помоћу чворова у облику круга или правоугаоника, а релације као усмерене линије са стрелицама. Чвор који представља специфичан пример класе назива се инстанца. Од релација реализоване су *IS_A*, која се користи да означи припадност неког елемента класи елемената, и *PART_OF*, која се користи да означи да је неки елемент део неког другог елемента. За разлику од семантичких мрежа, код оквира сваки чвор има своје атрибуте и вредности атрибута, па су приликом избора одређеног чвора приказани сви атрибути и вредности датог чвора.



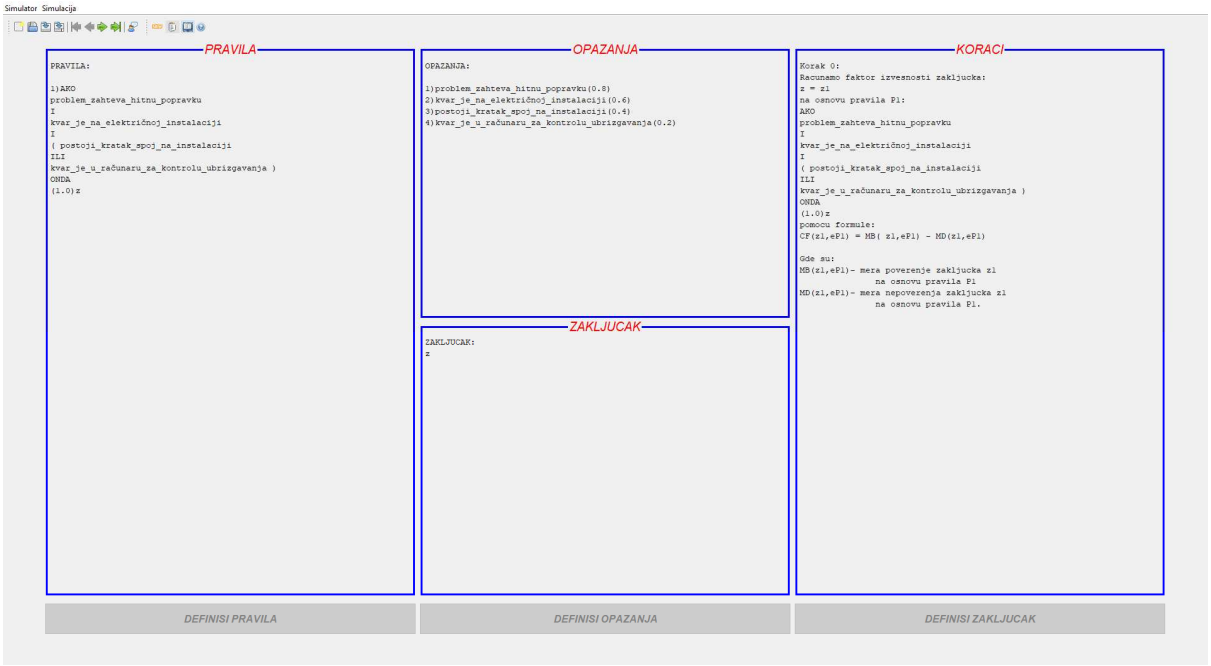
Слика 31 – Кориснички интерфејс симулације семантичких мрежа

Продукциони системи су због своје узрочно-последичне структуре најчешћи вид представљања знања код интелигентних система. Приликом дефинисања продукционог система, корисник треба да дефинише садржај радне меморије и продукционе меморије, а затим да одабере интерпретер који спроводи одговарајући алгоритам закључивања. У овом систему чињенице су представљене као уређена тројка која се састоји од идентификатора, атрибута и вредности, који описују неко својство. Када корисник унесе почетни садржај симулације, он треба да одабере један од алгоритама закључивања: директним уланчавањем, повратним уланчавањем, хибридним уланчавањем или Рете алгоритам. Током симулације може се десити ситуација да су сви условни елементи правила задовољени за неко правило, па се у том случају користе алгоритми приоритизирања, како би се решила резолуција конфликта над датим конфликтним скупом. Доступни алгоритми су: резолуција по приоритету, резолуција по комплексности, резолуција по специфичности, резолуција по редоследу учитавања и резолуција по времену окидања. Сваки од ових алгоритама даје различиту секвенцу током симулације и утиче на коначни резултат.

Рад у неизвесном окружењу представљен је у систему *SAIL* коришћењем три симулације: фактором извесности, системом за одржавање истинитости и расплинутом логиком.

У симулацији алгоритма са фактором извесности постоји велики број формула и прорачуна које би корисник система врло тешко савладао и израчунао без коришћења софтвера. Корисник да би добио резултат и видео кораке извршавања овог алгоритма, мора унутар система да дефинише правила, опажања и закључке, преко текстуалних поља. Помоћу фактора извесности се квантификује степен поверења у неки закључак. Правила имају своју предефинисану граматику и у њиховом дефинисању се користе резервисане речи: *IF*, *AND*, *OR*, *THEN*.

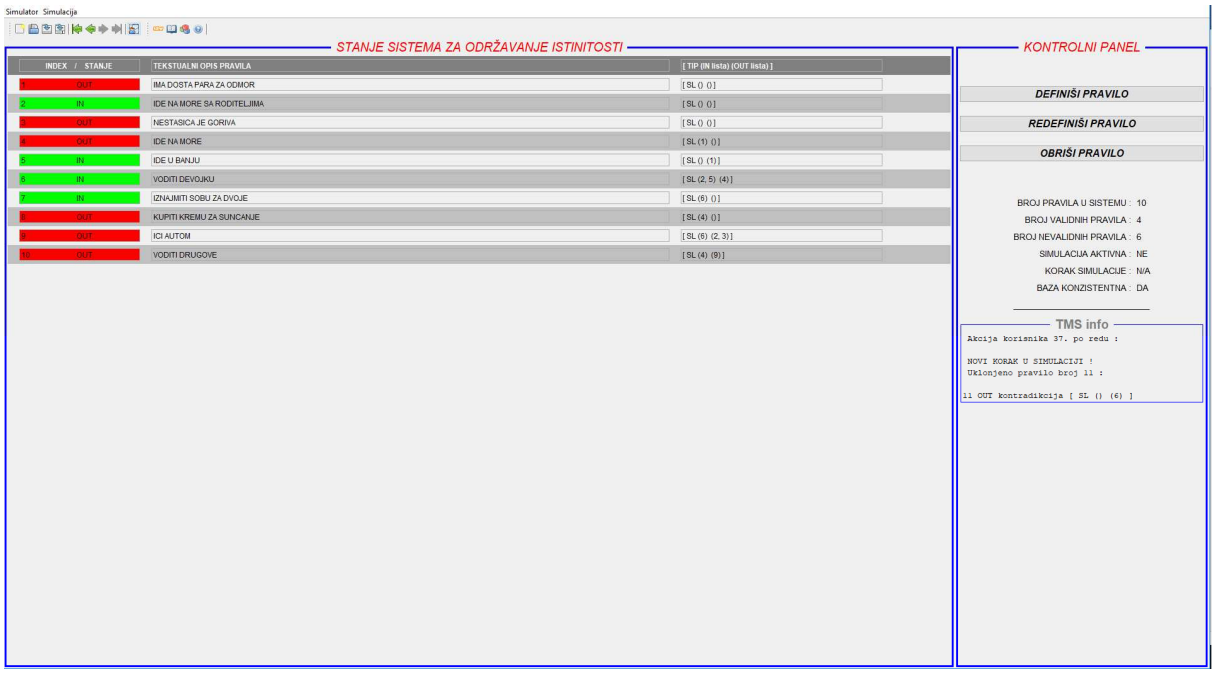
Свако опажање има своју меру поверења. Приликом покретања симулације, уколико има више закључака, корисник система може одабрати онај закључак који жели да разматра. Симулација се извршава корак по корак, а корисник система у сваком тренутку види формуле које се користе и прорачун који је добијен. На слици 32 приказан је један пример симулације са која примењује алгоритам са рачунањем фактора извесности.



Слика 32 – Кориснички интерфејс симулације са фактором извесности

TMS представља механизам чувања информација о зависностима и о препознавању неконзистентности. Пре почетка симулације, неопходно је дефинисати базу знања, чију конзистентност ће *TMS* одржавати. Када корисник дефинише сва правила у систему, може прећи на иницирање промене стања неког правила. Када дође до промене правила, долази до активирања *TMS* алгоритма, који врши одржавање конзистентности базе. Сваки корак тог

алгоритма приказује се кориснику. По завршетку симулације, корисник може да редефинише или обрише постојеће правило и поново покрене извршавање алгоритма (слика 33).



Слика 33 – Систем за одржавање истинитости са контролним панелом за унос параметара симулације

Алгоритми задовољења ограничења реализовани су у оквиру симулације *CSP* помоћу проблема бојења мапа, Н-краљица и Судоку, у рачунарској верзији система, и помоћу проблема укрштенице, у мобилној верзији система. *CSP* се састоји од променљивих, домена вредности које те променљиве могу добити и ограничења која се морају испунити. Приликом решавања ових проблема у нашем систему је примењен алгоритам претраге уназад, односно један од начина претраживања по дубини. Два начина побољшања претраге који су такође имплементирани у оквиру *SAIL* су провера унапред (енг. *forward checking*) и конзистентност лука (енг. *arc consistency*).

Од индукционих алгоритама реализован је *ID3* алгоритам. Рад алгоритма се приказује графички у виду формираног стабла одлучивања и текстуалног приказа свих израчунатих вредности. Корисницима система су омогућена два начина уноса улазних података: креирањем табеле са улазним подацима, директно из система путем контролних дугмади, и уносом података из текстуалне датотеке. Овај део система је накнадно надограђен и другим напреднијим техникама машинског учења, односно техникама машинског учења.

SAIL омогућава формирање произвољне Бајесове мреже и демонстрира процес закључивања и одлучивања у оквиру те мреже. За сваки од чворова графа морамо одредити његов домен, односно скуп могућих стања у којима се случајно променљива коју чвор представља може наћи. Могућа стања морају бити међусобно искључива и морају да обухвате све могуће вредности у којима се случајно променљива може наћи. Након увођења чворова, треба формирати и везе, уколико један чвор узрокује други, где стрелица представља смер зависности. Када је мрежа формирана, потребно је да корисник система квантификује везе између повезаних чворова, специфицирањем условних вероватноћа за сваку везу између пара чворова. Овим се формирају табеле условних вероватноћа чворова. Након што је мрежа формирана и табеле вероватноћа за сваки од чворова дефинисане, прелази се на расуђивање.

6. ПРИМЕНА СОФТВЕРСКОГ СИСТЕМА

У овом поглављу описана је примена софтверског система *SAIL* у настави, кроз реализоване лабораторијске вежбе, примере за самостално учење и могућност увођења електронског испита, и примена система у другим областима.

6.1. Примена у настави

Настава на предмету Интелигентни системи изводи се кроз 10 лекција у 15 наставних недеља, од којих се 14 недеља изводи настава, а једна недеља је предвиђена за колоквијум на половини семестра. У току семестра постоји и шест лабораторијских вежби, које се држе у току семестра, након завршетка одређених тема на часовима предавања. За сваку тему реализована је одговарајућа лабораторијска вежба у систему *SAIL*. Рад у лабораторији је показног карактера и не оцењује се, већ има за циљ да студенти боље савладају градиво. Распоред тема по недељама и терминима у којима студенти раде у лабораторији, приказан је у табели 5.

ТАБЕЛА 5

ПРОГРАМ ПРЕДМЕТА ИНТЕЛИГЕНТНИ СИСТЕМИ

Радна недеља	Лекција	Наслов лекције	Број алгоритама	Има лаб.вежбу
Н1	Л1	Увод у вештачку интелигенцију. Неинформисано претраживање.	1	
Н2		Хеуристичке функције и информисано претраживање.	7	+
Н3	Л2	Увод у теорију игара.	2	
Н4		Алгоритми теорије игара.		+
Н5	Л3	Формална логика.	4	
Н6		Формална логика.		+
Н7	Л4	Семантичке мреже и оквири.	2	
Н8	КОЛ.	Колоквијум на средини семестра.	/	
Н9	Л5	Продукциони и аналитички системи.	4	+
Н10	Л6	Рад у неизвесном окружењу.	2	
Н11	Л7	Фази логика.	1	
Н12	Л8	Стратегије решавања проблема.	5	+
Н13	Л9	Бајесове мреже.	1	
Н14	Л10	Основи машинског учења. Индукциони алгоритми. Стабла одлучивања.	1	
Н15				+

Легенда: Н - недеља, Л - лекција, КОЛ - колоквијум

6.2. Реализоване лабораторијске вежбе и примери за самостално учење

Рачунарска верзија *SAIL* реализована је за приказивање основних алгоритама на аудиторним вежбама и одржавање лабораторијских вежби. Ова верзија софтвера може се користити и за самостално учење. Самостално учење се може применити и на мобилним уређајима и тај вид учења називамо још и мобилним учењем. У овом поглављу биће приказана по једна лабораторијска вежба, предвиђена за рачунарску верзију софтвера, један пример за самостално учење на мобилној верзији софтвера и могућност да се *SAIL* примени код електронских испита, трансформацијом досадашњих традиционалних испита на папиру у испит који се ради на рачунару.

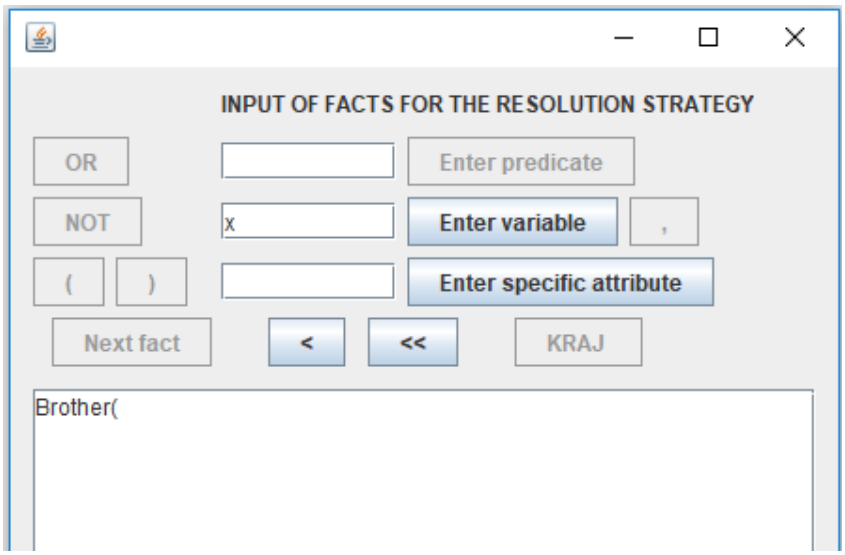
6.2.1. Лабораторијска вежба из формалне логике

У оквиру лабораторијске вежбе из формалне логике студенти треба да савладају алгоритме резолуције. Дат је задатак „Породични односи“, са следећим тврдњама:

1) Ако је особа А брат особе В и особа Б брат особе В, онда је особа А брат и особе Б или су А и Б иста особа.
2) Ако је особа А мушког пола и има исту мајку као и особа Б, онда је А брат особе Б и са А и Б иста особа.
3) Марија је мајка Милана и Ане.
4) Милан је мушко.
5) Јован је Анин брат.
6) Милан и Јован нису иста особа.
7) Милан и Ана нису иста особа.

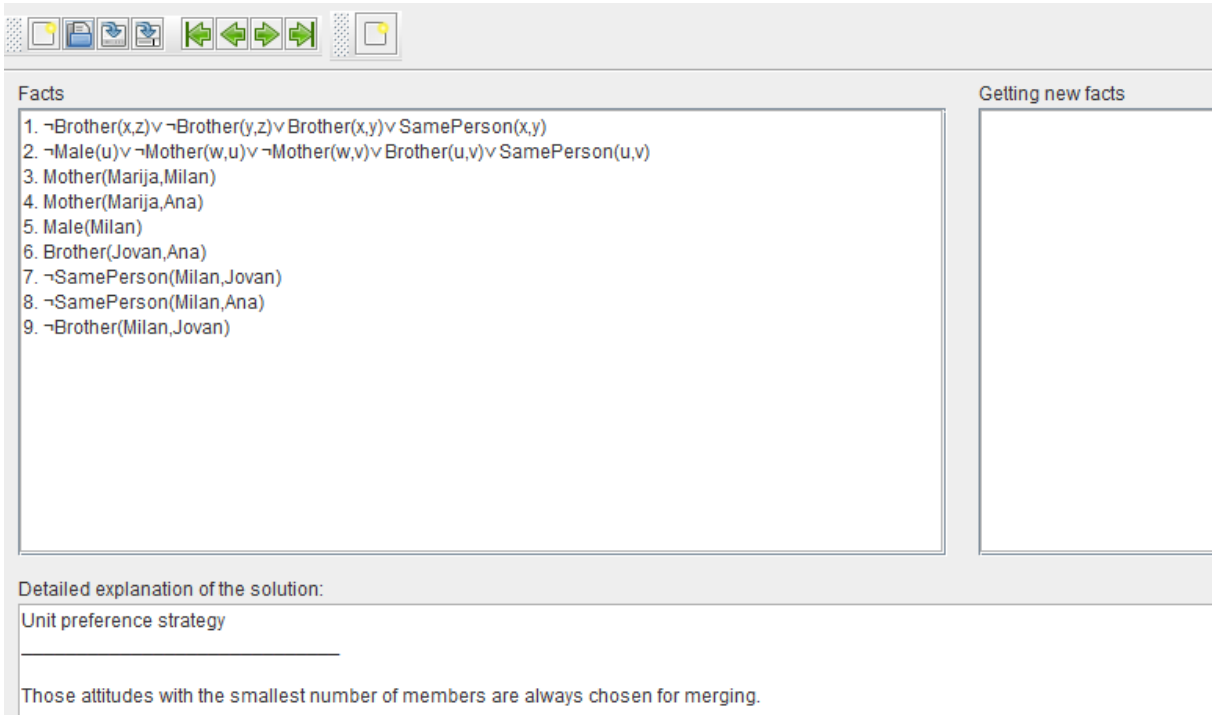
Стратегијом резолуције потребно је доказати или побити тврдњу да је Милан Јованов брат.

Након што се покрене неки алгоритам резолуције, студент кроз систем *SAIL* врши унос чињеница. Могуће је унети ИЛИ, негацију, отворену или затворену заграду, предикат, променљиву или конкретан атрибут, као што је приказано на слици 34.



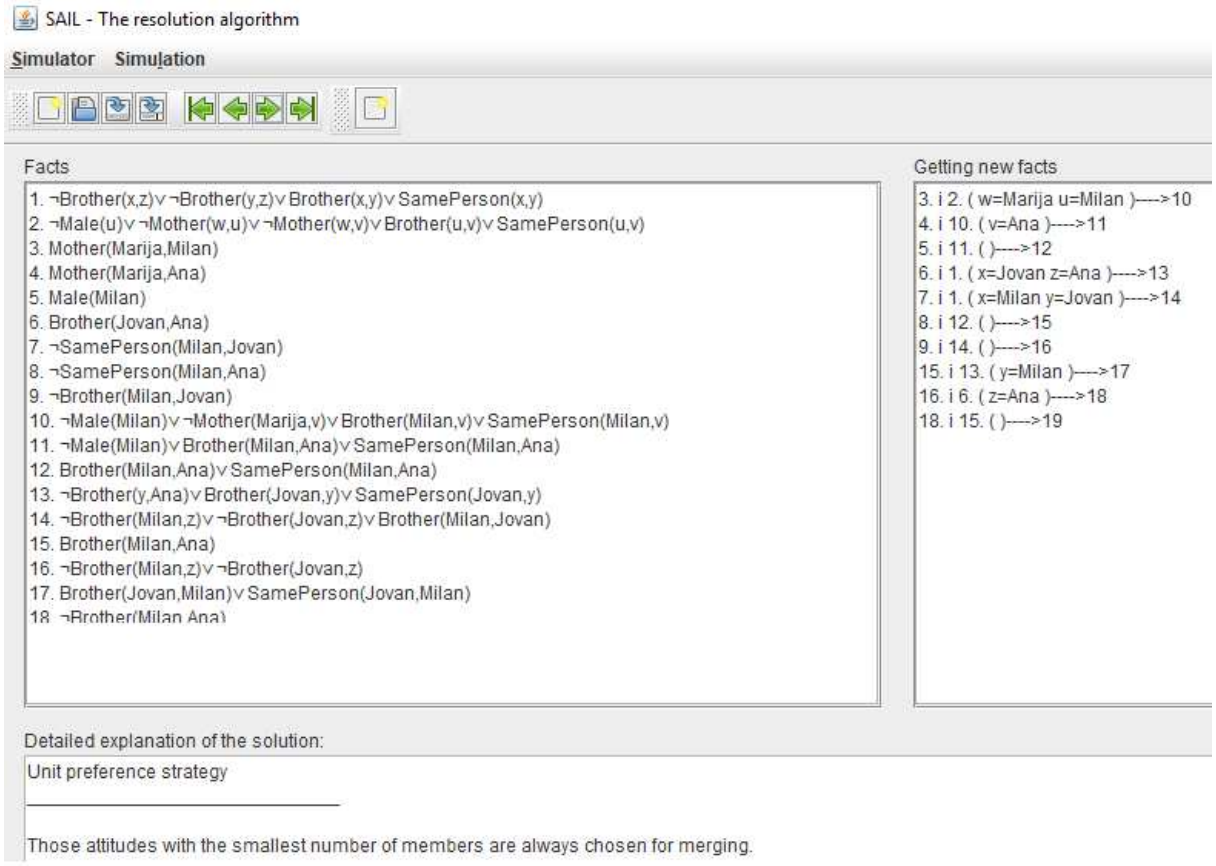
Слика 34 – Пример уноса чињеница у симулацију код алгоритама резолуције

У овом задатку студент прво треба да представи чињенице предикатским формулама на следећи начин: сва тврђења осим тврђења 1) и 2) се већ налазе у клаузалној форми, пошто се ради о литералима; тврђења 1) и 2) доводе се у клаузалну форму уклањањем импликације и применом Де Морганових закона уз преименовање променљивих друге формуле ради једнозначности. Претходним тврдњама додаћемо и негацију претпоставке да је Милан Јованов брат, да бисмо тражили противречност примењујући правило резолуције. Чињенице након што се унесу у систем приказане су на слици 35. У левом делу прозора налазе се све чињенице, а у десном делу новодобијене чињенице, односно ставови који су упарени, уз одговарајуће унификације. У доњем делу прозора приказују се објашњења.



Слика 35 – Кориснички интерфејс након уноса чињеница у симулацији алгоритама резолуције

Од алгоритама резолуције реализоване су имплементације на три начина: стратегијом скупа подршке, стратегијом првенства јединица и стратегијом избора ставова по ширини. На слици 36 приказано је коначно решење које студент добија, након десет корака симулације, уколико је одабрао стратегију избора по ширини. У тој стратегији прво ће се разматрати све могуће комбинације постојећих чињеница, пре него што се упаре новодобијене чињенице.



Слика 36 – Пример корисничког екрана након извршавања алгоритма резолуције коришћењем стратегије избора по ширини

Након завршене симулације одређеног алгоритма, студент има могућност да генерише извештај са детаљним описом решења у виду екстерног *PDF* фајла, који омогућава студенту да анализира решење по завршетку задатка на лабораторијској вежби. Студенти који неураде лабораторијску вежбу, решавају задатак на крају вежбе тако што прате симулацију коју приказује предметни асистент и извршавају те кораке на свом рачунару.

6.2.2. Самостално учење студената на мобилном уређају

Мобилно учење (м-учење) представља са једне стране потпуно нову форму учења на даљину, самосталног учења и ново проширење е-учења. Учење градива из области вештачке интелигенције на паметним телефонима уз помоћ алата *SAIL*, биће демонстрирано на примеру алгоритама претраживања. Један од најпознатијих проблема код ових алгоритама је налажење путање између градова, за које се зна цена путање (дужина пута између градова) и хеуристичка функција. Студент може да исцрта нови граф или да учита постојећи граф из текстуалног фајла. Када се жели учитавање графа из текстуалног фајла, неопходно је да

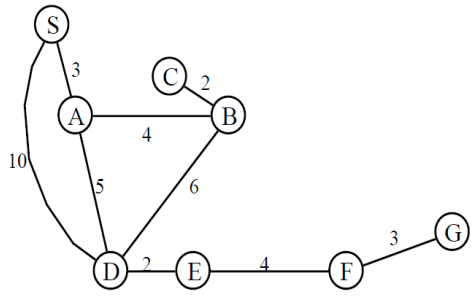
такав фајл буде одређене структуре, да би парсер могао да изврши исправно операцију читања:

- почетни чвор у графу: A 4 start
- чвор у графу: D 5
- циљни чвор у графу: J 0 end
- веза између два чвора: A-D-2

где су *A*, *D* и *J* имена чворова, вредности 4, 5 и 0 су вредности хеуристичке функције, а *A-D-2* представља везу између чворова са вредношћу цене. Смер везе може бити коришћењем стрелица (<- или ->) уместо цртице (-).

Циљ студента који покреће овај задатак за самостално учење на свом мобилном уређају је да разуме кораке алгоритма, односно редослед чворова који се обилазе у стаблу, као и разлог зашто су баш ти чворови одабрани. За разлику од симулације на рачунару, где се исписују и коментари, на мобилном уређају нема довољно места за приказ свих коментара, па се подразумева да студент буде добро упознат са теоријским градивом, када ради задатке за самостално учење.

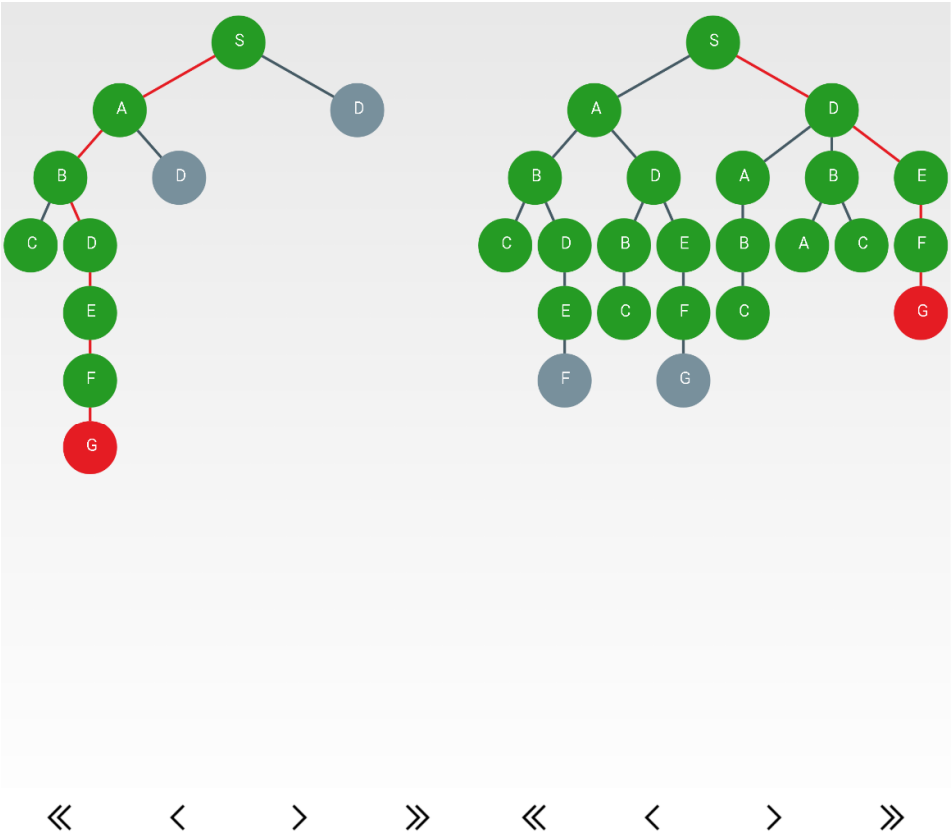
У овом конкретном задатку, студент ће научити како да пронађе пут између градова *S* и *G* коришћењем одређеног алгоритма претраживања. Путна мрежа је дата у километрима, а хеуристичка функција (ХФ) представља ваздушно растојање одређеног града у односу на циљни град *G*, као што је приказано на слици 37.



City	S	A	B	C	D	E	F
Distance to city G	11.5	10.4	6.7	7.0	8.9	6.9	3.0

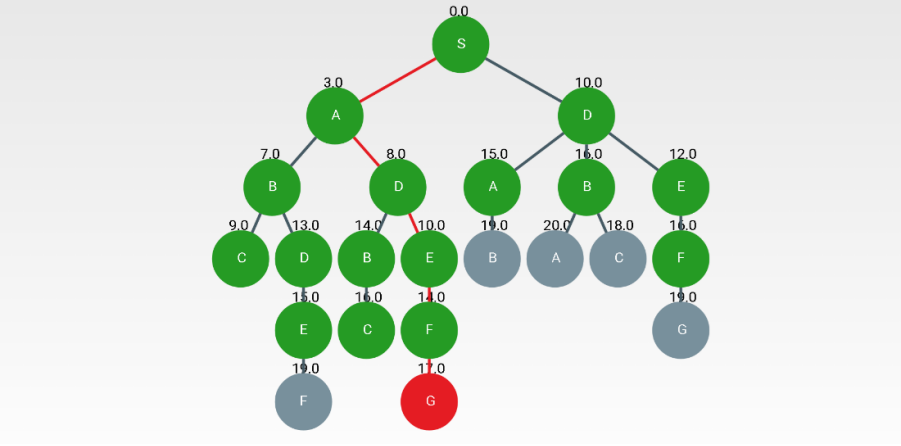
Слика 37 – Пример графа путне мреже са растојањима између градова (у км) и ваздушним растојањима (у км, датим у табели испод графа)

Методе по дубини и по ширини, приказане на слици 38, не дају најкраћу могућу путању између градова. Уколико се користи метода по дубини, редослед обиласка чворова је следећи: *S*, *A*, *B*, *C*, *D*, *E*, *F*, *G*. Пронађена путања је дужине 22 километара и пролази кроз следеће чворове: *S-A-B-D-E-F-G*. Приликом обилажења чворова имплементиран је лексикографски поредак. Методом по ширини путања је: *S-D-E-F-G* и износи 19 километара.



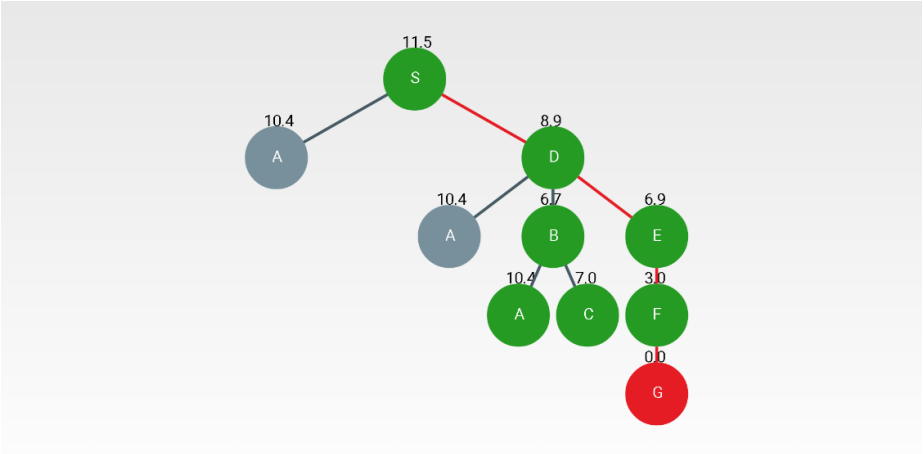
Слика 38 – Стабла претраживања за симулације алгоритма по дубини и алгоритма по ширини

Стабло претраживања коришћењем алгоритма гранања и ограничавања је слично као стабло приликом обиласка по ширини. Овај метод даје најоптималније решење, дужине 17 километара, које пролази кроз следеће чворове: *S-A-D-E-F-G*, као што је приказано на слици 39.



Слика 39 – Стабло претраживања за симулацију алгоритма гранања и ограничавања

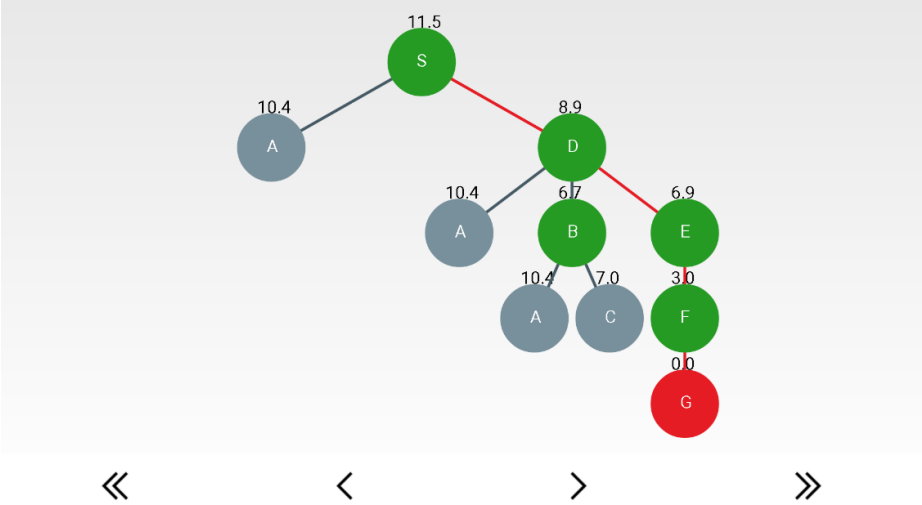
У случају алгоритма претраге методом планинарења користи се хеуристичка функција дата у виду ваздушног растојања између градова. Са слике 40, видимо да је у првом кораку након почетног чвора *S* изабран чвор *D*, пре чвора *A*, зато што је ХФ за чвор *D* 8.9, а за чвор *A* 10.4, па гледамо мању вредност функције процене. Слично је и у наредним корацима обиласка овог стабла, где се локално за сваки чвор гледају ХФ свих следбеника и бира следбеник са најмањом вредношћу ХФ. Коначно добијена путања ове методе је: *S-D-E-F-G* и дужине је 19 километара.



Слика 40 - Стабло претраживања за симулацију алгоритма планинарења

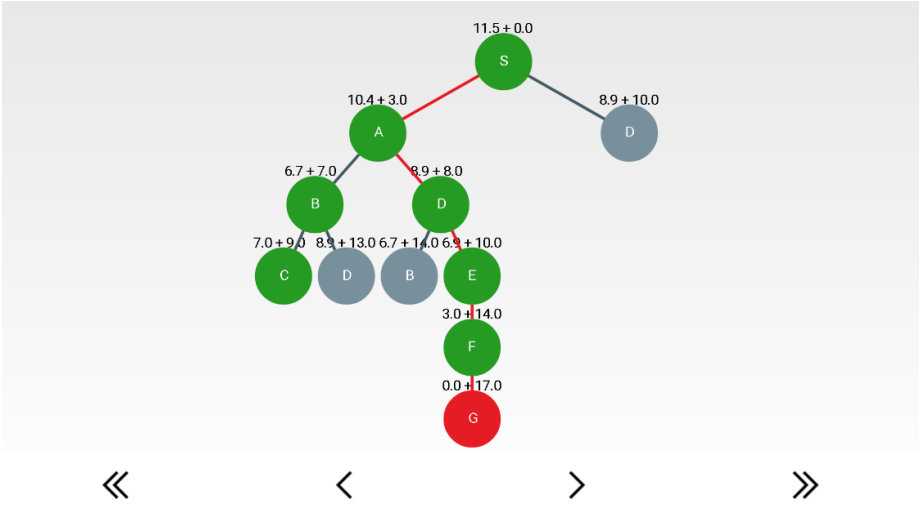
Алгоритам прво најбољи је сличан алгоритму планинарења. Једина разлика је што овај метод проналази најбољи чвор глобалном претрагом свих откривених и још увек

непосећених чворова, па за разлику од алгоритма планинарења овај алгоритам даје исто решење, при томе обилазећи много мањи број чворова. Добијена путања *S-D-E-F-G* је дужине 19 километара уз само шест обиђених чворова, као што је приказано на слици 41.



Слика 41 - Стабло претраживања за симулацију алгоритма прво најбољи

Путања коју добијамо методом *A** даје најкраћу путању дужине 17 километара, као што је приказано на слици 42. Код ове групе алгоритама реализовано је и опционо коришћење динамичког програмирања.



Слика 42 - Стабло претраживања за симулацију алгоритма *A**

6.2.3. Могућност увођења електронског испита

Курсеви на многим светским универзитетима све више користе предности интернета, па се све већи број лекција, материјала за учење и електронских књига данас дистрибуира кроз интернет апликације, које најчешће укључују и форуме за дискутовање. Са друге стране, испитивања и тестирања студената нешто спорије се имплементирају као засебне интернет апликације, јер су наставници пре свега више оријентисани ка традиционалним начинима испитивања у виду тестова на папиру.

У последњој деценији, код курсева на даљину или као додатни алат за универзитетске курсеве, наставници све више почињу да користе системе за електронско учење (*LMS*) у високом образовању. *LMS* нуди много лакше администрирање курса, подсистеме за управљање садржајем и документима, евалуацију корисника и њихових тестова и врло детаљне врсте извештаја. Постоји велики број позитивних извештаја о коришћењу *LMS* у образовању, посебно код курсева са великим бројем предиспитних и испитних обавеза и са великим бројем студената [4] [8]. Најпознатији системи за учење у образовању су: *Blackboard* [81], *Moodle* [82], *Desire2Learn Brightspace* [83] и *Canvas* [84].

Moodle је најпопуларнији *LMS* отвореног кода и користи се у различитим нивоима образовања, од стране ученика, студената, наставника и професора, у више од 230 земаља. У Србији према званичним подацима постоји 298 регистрованих *Moodle* платформи за е-учење [85]. Кључне предности *Moodle* платформе су добра организација курсева, лако управљање и дистрибуирање материјала, једноставна комуникација између професора и студената кроз форуме за дискусију и подршка за електронско тестирање. Многи курсеви на интернету и образовни програми за учење на даљину у потпуности користе платформу *Moodle*. У неколико истраживања представљено је задовољство корисника платформом *Moodle*, у односу на комерцијална алтернативна решења и истакнуте предности у развоју курсева [86] [87]. Курсеви засновани на програмирању и другим предметима из области рачунарства такође могу да имају неки облик тестова на оваквим платформама за е-учење [88] [89] [90]. Новија истраживања приказују и значај друштвених мрежа у образовању и њихово активно коришћење за дистрибуирање материјала за учење, уместо *LMS* и класичних веб сајтова [91]. Групе на друштвеним мрежама нису предвиђене за тестирање и испитивање студената.

Платформа за е-учење *Moodle* укључена је у 18 предмета на Електротехничком факултету Универзитета у Београду, за дистрибуирање наставних материјала, комуникацију

са студентима и испитивање студената кроз тестове за лабораторијске вежбе, колоквијуме и делове или целе електронске испите [92]. Приликом потпуног укључивања овакве платформе у предмет, највећи проблем је како тестове и испите у традиционалној папирној форми трансформисати у електронске испите. На Катедри за рачунарску технику и информатику Електротехничког факултета у Београду реализована је студија и предложено је 12 начина трансформације питања и задатака са традиционалних испита у електронске тестове са кратким питањима [93]. Студија је обухватила четири предмета на основним академским студијама.

Курс Интелигентни системи могао би да се интегрише са платформом за е-учење *Moodle* тако што би се тестови уз лабораторијске вежбе извршавали на њој. Док траје тестирање, студент би у систему *SAIL* пратио извршавање симулације одређеног алгоритма. Студент би за дефинисано време теста имао одређени број питања на која треба да одговори са једним тачним одговором, са више тачних одговора, са допуњавањем одговора, упаривањем или неком другом врстом одговора. Колоквијуми и испити на папиру би се теже интегрисали, због задатака у којима се оцењује и поступак решавања одређеног проблема. Неки задаци би могли да се трансформишу и аутоматски оцењују од стране система, али неки задаци би морали да остану са делимично мануелним оцењивањем. У табели 6 дат је пример трансформације неколико задатака традиционалног испита у електронски испит на платформи за е-учење *Moodle*.

Други начин електронског испитивања био би унапређењем система *SAIL* уз додатни модул за испитивање и тестирање студената или миграцијом система *SAIL* у нови веб-оријентисани систем. Предлог такве веб платформе за полагање тестова знања, оцењивање студената и администрирање тестовима, реализован је као пилот пројекат, али без ослањања на симулације алгоритама и било који део програмског кода из система *SAIL* [94].

ТАБЕЛА 6
ПРИМЕР ТРАНСФОРМАЦИЈЕ ИСПИТНОГ ЗАДАТКА У ПИТАЊА НА ПЛАТФОРМИ ЗА Е-УЧЕЊЕ

	Традиционални тип питања	***Moodle* тип питања**	**Пример из платформе *Moodle***
(1)	Унос вредности	Кратак одговор, питање са једним тачним одговором	Унети колика је дужина путање која се добија алгоритмом планинарења.
(2)	Слика као одговор	Питање са једним или више тачних одговора	Која семантичка мрежа са слике одговара датом опису? Заокружити један тачан одговор.
(3)	Табела као одговор	Тачно/нетачно, упаривање	Поред сваког правила из описаног TMC система написати да ли се налази у IN или OUT стању.
(4)	Кратко теоријско питање	Тачно/нетачно, питање са више тачних одговора	Да ли алгоритам A* гарантује налажење путање у најмањем броју корака?
(5)	Више тачних одговора	Питање са више тачних одговора	Означити сва IN стања код правила описаног TMC система.
(6)	Део програмског кода као одговор	Питање са једним или више тачних одговора	Означити којим од наведених псеудо кодова је коректно описана измена алфа и бета вредности код МАКС играча у алгоритму Минимакс.
(7)	Допуњавање речи која недостаје	Кратак одговор, тачно/нетачно, питање са више тачних одговора	Судоку и проблем Н-краљица се решавају методом _____.
(8)	Нумерички резултат	Нумерички одговор, одговор који се израчунава	Колико износи фактор извесности закључка да је пацијент под грипом?
(9)	Тачно/нетачно	Тачно/нетачно	Алгоритам C4.5 спада у алгоритме индуктивног учења? Да / Не
(10)	Познавање термина/алгоритама	Упаривање	За алгоритме дате у левој листи, означити у десној листи ДА, уколико користе хеуристичку функцију или НЕ, уколико не користе хеуристичку функцију.
(11)	Задатак без објашњења	Кратак одговор, питање са више тачних одговора, упаривање, нумерички одговор	Шта ће бити вредност чвора листа у стаблу игре, када постоје три играча описане игре? Шта треба да буде почетни потез Ане као првог играча, да би Ана победила?
(12)	Задатак са детаљним објашњењем	Кратак одговор, питање са више тачних одговора, есеј (са мануелним оцењивањем)	Задатак: Написати које су путање и израчунати цену путање особе А од старта до циља, ако се примењују: (а) алгоритам прво-најбољи, (б) алгоритам планинарења, (в) алгоритам A* Овај задатак може да се трансформише у 6 питања са кратким одговорима, тачно/нетачно или нумерички одговор. Есеј се може користити за оваква питања: Описати укратко шта се дешава у графу ако се користи динамичко програмирање.

6.3. Примена система у другим областима

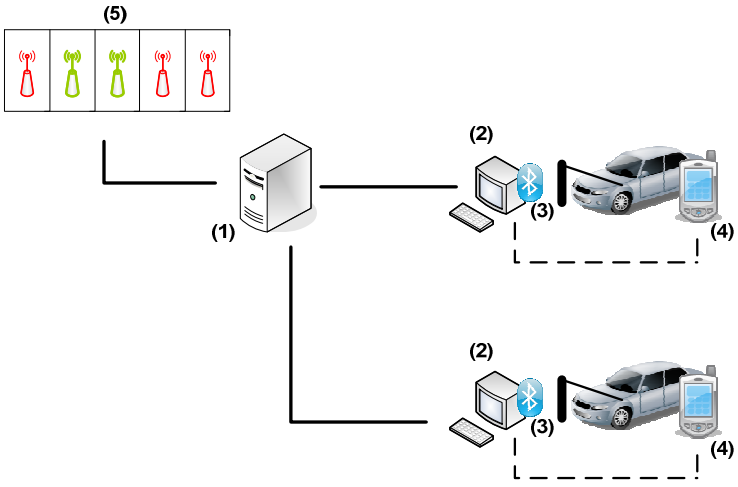
У овом поглављу биће приказана могућност примене софтверског система *SAIL* у другим областима. Први пример показује употребу симулације алгоритама претраживања у оквиру паметног паркинга са бежичном сензорском мрежом. Други пример показује употребу симулације са фактором извесности као помоћ у медицинској дијагностици. Трећи пример показује примену стабала одлучивања и техника машинског учења на великим подацима.

6.3.1. Модел паметног паркинга

Јавна градска паркиралишта и тржни центри данас имају паркинге са великим бројем паркинг места. Такви паркинзи најчешће на улазу имају приказан број слободних места, али корисници нису у могућности да виде где се налази најближе слободно паркинг место. Проблем представља ситуација, где корисник крећући се кроз паркинг може доћи до места

где мора да одлучи да ли да скрене лево или десно или настави право, не знајући на којој страни ће брже и лакше наћи паркинг место. У граничном случају може се десити да корисник бесконачно дуго кружи по паркингу и да се у међувремену нека места ослобађају, али их други корисници, који су били бржи попуњавају. Према студији коју је урадила компанија *IBM* у 20 светских градова, људи дневно проведу у просеку 20 минута да би пронашли слободно паркинг место [95]. Ово време које корисници потроше у проналажењу паркинга повећава потрошњу горива, а ослобађање штетних гасова нарушава еколошку средину.

На слици 43 дат је дијаграм техничких захтева паметног паркинга и корисника са мобилним уређајем, као и њихова међусобна повезаност [96]. Делови овог софтверско-хардверског система су: (1) централни сервер, (2) дистрибуирани терминал, (3) *Bluetooth* адаптер на терминалну, (4) мобилни телефон корисника, (5) сензори статуса паркинг места.



Слика 43 – Софтверско-хардверски систем паметног паркинга [94]

Сензори одређују статус паркинг места и они су повезани једносмерно са централним сервером. Централни сервер и мобилни уређај корисника повезани су преко дистрибуираног терминала и преко адаптера са *Bluetooth* протоколом.

Приликом уласка на паркинг, корисник би на улазу добио шему комплетног паркинга на свом мобилном уређају. Апликација на мобилном уређају на основу добијене шеме, а коришћењем алгоритама претраживања, треба да исцрта пут до најближег паркинг места, чиме би корисник пратећи исцртани пут брзо и једноставно дошао до првог слободног паркинг места. Формирање графа и обилазак стабла претраживања реализовано је

коришћењем класа из пакета *pretrazivanje*, из софтверског система *SAIL*, а за графику паркинг места је реализована засебна класа која генерише кориснички интерфејс.

6.3.1.1. Архитектура хардверског дела паметног паркинга

Сервер води евиденцију о свим паркинг местима и њиховој тренутној расположивости за корисника. Статус паркинг места је слободно или заузето. У сваком тренутку, на серверу постоји ажурно стање свих паркинг места. У случају да паркинг има велики број паркинг места, провера досупности статуса може бити веома временски сложена операција. Терминали приступају серверу путем унапред дефинисаног протокола. Уколико постоји велики број терминала, централни сервер треба да обезбеди паралелно обрађивање захтева како би кашњење на терминалима у случају великих гужви било што мање.

На сваком улазу у паркинг налазе се дистрибуирани терминали са *Bluetooth* адаптером. Потреба за постојањем терминала је што велики паркинзи често имају више улаза. У случају да у архитектури овог система не постоји централизовани сервер, ови терминали би морали међусобно да комуницирају о статусу паркинг места и да обрађују захтеве корисника. Повезивање сервера и терминала мора да буде реализовано брзим везама. Статус паркинг места добија од сензора, који аутоматски детектује заузетост неког паркинг места. Сензор треба да детектује присуство аутомобила. Да би се избегло осцилирање између слободног статуса и заузетог статуса, на свако паркинг место је постављено по два сензора [97]. Сензори статуса повезани су у бежичну сензорску мрежу, која комуницира са централним сервером, слично као што је реализовано у другим системима паметних паркинга [98] [99].

6.3.1.2. Структура шеме паркинга

Шема паркинга се дефинише приликом постављања система, а касније се може ажурирати. Она садржи информације о структури паркинга и распореду паркинг места, бројевима и позицијама улаза и излаза паркинга и променама правца аутомобила приликом преласка из једног дела паркинга у други. Шема паркинга је описана *XML* датотеком, која садржи следеће елементе:

- паркинг блок,
- паркинг место,
- улаз у паркинг,
- прелаз између два паркинг блока.

Паркинг блок је правоугаоног облика и има свој јединствени идентификатор, листу паркинг места која се налазе на њему, листу улаза у тај паркинг блок, као и листу прелаза. Корисник може ући на паркинг у сваком блоку који поседује улаз. Након тога, корисник може да се вози унутар тог блока или да пређе у други суседни паркинг блок, користећи прелаз између два блока. Прелази између два паркинг блока налазе се само по ободу блокова. Правоугаони паркинг блок има своју висину, ширину, родитељски блок, место спајања у односу на родитељски блок и страну спајања у односу на родитељски блок. Уколико родитељски блок не постоји, подаци о месту спајања са родитељским блоком и страном спајања у односу на родитељски блок остају без задате вредности.

Паркинг блокови чине структуру стабла. Корен стабла представља паркинг блок који садржи улаз на коме се корисник налази. Најједноставнији начин да се прикаже овакав модел је у формату *XML* фајла. Фајл садржи главни елемент под називом *<ParkingScheme>* унутар кога се налазе други елементи. Пример једног паркинг блока приказан је *XML* датотеком, приказаном следећим програмским кодом:

XML програмски код паркинг блока

```
<ParkingBlock ID='1'>
     <Type>Rectangular</Type>
     <Width>15</Width>
     <Height>14</Height>
     <OffsetX>0</OffsetX>
     <OffsetY>15</OffsetY>
     <ParentBlock>3</ParentBlock>
     <MergingPoint>10</MergingPoint>
     <MergingSide>right</MergingSide>
     <ParkingSpots>
       <ParkingSpot>...</ParkingSpot>
     </ParkingSpots>
     <Gateways>
       <Gateway>...</Gateway>
     </Gateways>
     <Entries>
       <Entry>...</Entry>
     </Entries>
</ParkingBlock>
```

Из програмског кода може се закључити да је тип паркинг блока правоугаони и да приказани блок у стаблу блокова има родитељски блок са идентификатором ID=3, који се налази уз родитељску десну ивицу, почев од десетог метра посматрано одозго на доле. У случају када би постојала могућност додавања других облика паркинг блокова, променио би се само елемент типа. Апсолутне вредности помераја по одговарајућим осама, иако су

приказане у оквиру овог програмског кода, могле би да буду израчуната и накнадно у самој апликацији.

Паркинг место је простор за једно возило и има своју висину, ширину, позицију у припадајућем паркинг блоку, страну на којој се налази у паркинг блоку и статус, који може бити словодан или заузет. Пример једног паркинг места, описаног у *XML* фајлу дат је следећим програмским кодом:

XML програмски код паркинг места

```
<ParkingSpot>
     <BlockSide>Bottom</BlockSide>
     <StartPoint>0</StartPoint>
     <Width>3</Width>
     <Height>5</Height>
     <Status>Occupied</Status>
</ParkingSpot>
```

Из програмског кода може се закључити да се ово паркинг место налази на доњој страни припадајућег паркинг блока, почетна тачка простирања паркинг места је на координати 0, а блок се простире 3 метра у десно и 5 метара на горе. Тренутни статус овог паркинг места је заузет.

Улаз у паркинг блок представља улаз у паркинг. Улаз има ширину, позицију и страну на којој се налази у припадајућем паркинг блоку. Пример једног улаза на паркинг, описаног у *XML* фајлу дат је следећим програмским кодом:

XML програмски код улаза у паркинг блок

```
<Entry>
     <EntrySide>left</EntrySide>
     <Width>3</Width>
     <StartPoint>5</StartPoint>
</Entry>
```

Овај пример представља улаз паркинг блока који се налази на левој страни, има ширину од 3 метра, а тачка простирања од које почиње се налази на левој страни блока 5 метара од његове горње тачке. Крајња тачка простирања улаза овог блока је на осмом метру од горње стране блока, на његовој левој страни.

Прелаз између два паркинг блока представља место на којем може да се пређе из једног паркинг блока у други. Прелаз има своју ширину, позицију унутар блока и страну на

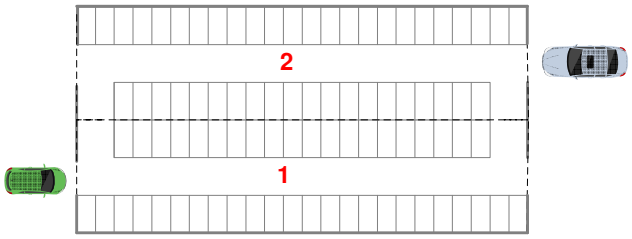
којој се налази у блоку. Сваки прелаз је означен као двосмеран, али би се једном ознаком могао дефинисати смер као једносмеран или двосмеран. Пример једног прелаза између два паркинг блока, дат је следећим програмским кодом:

XML програмски код прелаза

```
<Gateway>
     <GatewaySide>Top</GatewaySide>
     <Width>5</Width>
     <StartPoint>10</StartPoint>
</Gateway>
```

Оваквим предложеним паркингом су подржани сви облици паркинга који би се могли представити низом међусобно нормалних и паралелних правоугаоника, који се међусобно додирују. Ако паркинг има паркинг места само по свом ободу, такав паркинг би био описан само једним блоком унутар кога се налази низ паркинг места. У случају да је паркинг и даље правоугаоног облика, али такав да и унутар блока има паркинг места, а не само по ободу, тада би било потребно реализовати неколико паркинг блокова, који су нанизани један на други.

На слици 44 приказана је структура једног паркинга, који садржи два паркинг блока. Блок 1 је означен као главни блок паркинга и представља корен стабла тог паркинга. Њему је на доњој ивици додат блок 2, пошто се место спајања блокова посматра у односу на родитељски блок. Место спајања је једнако 0 и блокови у овом примеру имају исту ширину. Блок 2 има два прелаза на родитељски блок, оба на његовој горњој ивици, један при почетку те ивице, а други при крају. Прва два реда паркинг места припадају блоку 1, а доња два реда блоку 2. Могуће је било поделити структуру оваквог паркинга и у више блокова, али није било потребе. Сваки блок садржи по два улаза на паркинг, са своје леве, односно десне стране.



Слика 44 – Пример структуре паркинга са два паркинг блока

6.3.1.3. Имплементирани алгоритми у софтверском делу система

Имплементација система паметног паркинга подељена је на следеће класе:

- Класе које чине структуру паркинга – паркинг блок, паркинг место, улаз у паркинг, прелаз;
- Класе које реализују алгоритме претраживања за налажење најближег слободног места, позване из система *SAIL*;
- Андроид класе за кориснички интерфејс саме апликације и комуникацију са паркинг сервером.

На слици 45 дат је пример шеме паркинга Електротехничког факултета у Београду [96]. У графу са слике сваки блок шеме (*A*, *B*, *C*, *D*) представљен је чвором, а сваки прелаз између блокова везом. На местима где се сусрећу два или више блокова биће дефинисани прелази, односно везе ка тим суседним блоковима.

Слика 45 – Блоковска шема на примеру паркинга Електротехничког факултета

За алгоритам претраге из система *SAIL* одабран је алгоритам гранања и ограничавања. На почетку рада алгоритма проналази се блок у коме се возило налази. Затим се покреће алгоритам претраживања у том блоку, али и у свим његовим суседним блоковима. Претраге се врши тако што се тражи најкраће растојање између координата лоцираног возила и координата паркинг места. Алгоритам формира стабло претраге и памти чворове који су обиђени, јер нема потребе кретати се кроз исте блокове више пута.

У правоугаоном блоку потребно је покренути по четири нити алгоритма претраге, за сваку од страна блока по једну. Након завршетка свих нити претраге, свака нит ће вратити по једно слободно паркинг место уколико оно постоји и од највише четири добијена резултата биће одабрано оно паркинг место које је најближе. На исти начин се ради претрага и у суседним блоковима. У суседним блоковима се ради претага истовремено када и у посматраном блоку, јер нека места по дужини путање до њих могу бити ближа, односно

боље је да возило буде паркирано у суседном блоку, него у посматраном. То ће зависити од удаљености возила и слободних паркинг места у односу на прелазе између посматраног блока и суседних блокова. Уколико је најближи прелаз ка суседном блоку даљи од слободног паркинг места пронађеног у посматраном блоку, онда претраживање у суседном блоку није више ни потребно.

Структура паркинга могла би се реализовати и као тежински граф, при чему би се уместо алгоритма гранања и ограничавања користио алгоритам A*. У том случају би се узеле у обзир и тежине путања као вредност хеуристичке функције. Оваква структура би могла бити примењена у случају када би један прелаз између суседних паркинг блокова био проходнијих од другог, иако је према дужини путање даљи.

Овакав систем паметног паркинга омогућава уштеду времена приликом налажења најближег слободног паркинг места. Осим хардверских захтева који су већ дефинисани, потребно је на почетку рада оваквог система дефинисати паркинг шему са поделом на блокове. Ово може бити комплексно у зависности од сложености и величине паркинга, али се ради само једном. Систем је могуће модификовати тако да подржи друге облике паркинга, паркинге са више нивоа, али и једносмерне прелазе између блокова уместо двосмерних.

6.3.2. Примена одлучивања у медицини

Алгоритми вештачке интелигенције имају велику примену у области медицине, где могу помоћи лекару да одреди најбољу терапију или да на основу одређених симптома закључи о којој болести је реч. У овој области највише се користе предиктивне анализе засноване на тренутним и историјским подацима. Предиктивна анализа користи велики број техника и методологија обраде података, укључујући обраду великих података, алгоритме машинског учења, засноване на надгледаном или ненадгледаном учењу и статистичко моделовање.

У овом поглављу биће приказана једна примена у мединици коришћењем модула за рад у неизвесном окружењу помоћу фактора извесности. Пример ће показати како да добијемо фактор извесности одређене медицинске дијагнозе [100]. У систем се прво уносе правила:

Правило 1: АКО пацијент има мање од 8 година или више од 60 година

ОНДА (1.0) пацијент је у критичним годинама

Правило 2: АКО пацијент има високу температуру И

пацијент осећа малаксалост ИЛИ пацијент осећа болове у мишићима

ОНДА (0.7) пацијент има грип

Правило 3: АКО пацијент има натечено грло И

пацијент има кијавицу

ОНДА (0.6) пацијент има грип

Правило 4: АКО пацијент има грип И

пацијент је у критичним годинама

ОНДА (0.9) пацијент треба хитно да се обрати лекару

Након уношења правила, потребно је унети и опажања:

- пацијент има 65 година;
- пацијент има високу температуру;
- пацијент има натечено грло и кијавицу;
- пацијент осећа малаксалост са извесношћу 0.8 и болове у мишићима са извесношћу 0.9.

Систем ће одредити колико су извесни закључци да пацијент има грип (*z1*), односно да пацијент треба да се хитно обрати лекару (*z2*). На слици 46 приказан је први корак симулације након унетих правила и опажања. До закључка да пацијент има грип воде два правила, па због тога морамо да израчунамо збирне мере поверења (*MBcum*) и неповерења (*MDcum*) тог закључка, помоћу следећих формула:

MBcum(z, e1, e2) = 0, ако је MDcum(z, e1, e2) = 1

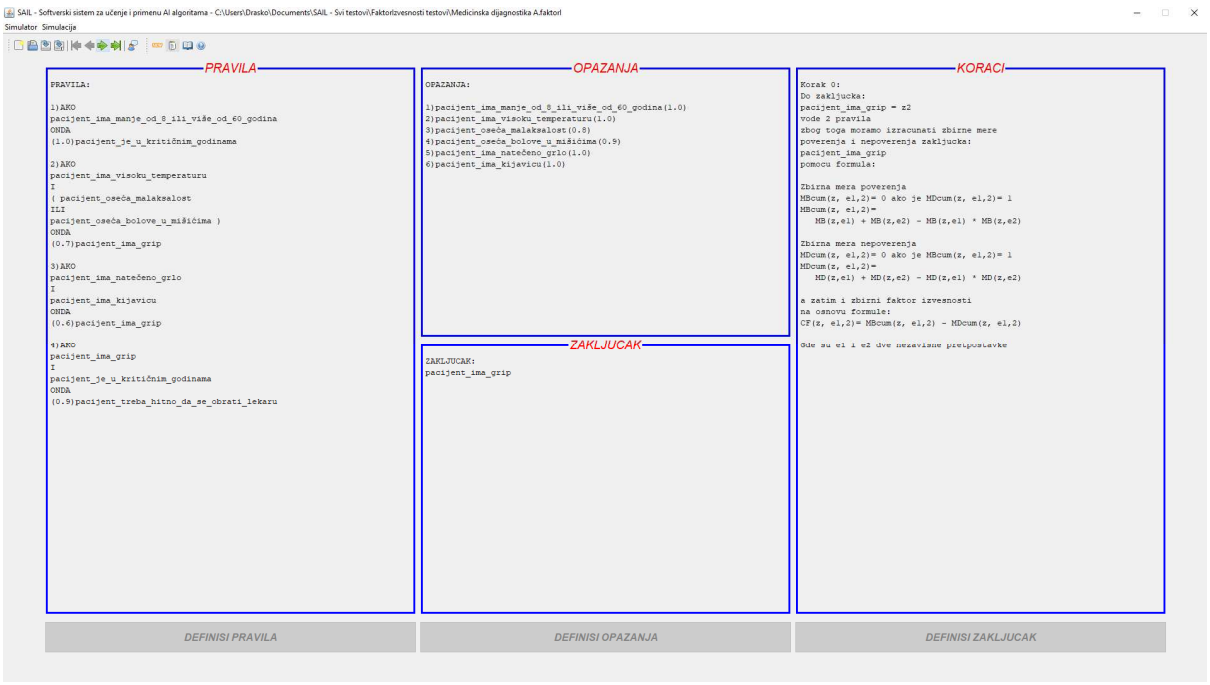
MBcum(z, e1, e2) = MB(z, e1) + MB(z, e2) – MB(z, e1) * MB(z, e2)

MDcum(z, e1, e2) = 0, ако је MBcum(z, e1, e2) = 1

MDcum(z, e1, e2) = MD(z, e1) + MD(z, e2) – MD(z, e1) * MD(z, e2)

Затим треба израчунати и збирни фактор извесности на основу формуле:

CF(z, e1, e2) = MBcum(z, e1, e2) – MDcum(z, e1, e2)



Слика 46 – Први корак симулације модула са фактором извесности на примеру медицинске дијагнозе

Систем израчунава факторе извесности *CF* претпоставки појединих правила и на основу њих ревидира мере поверења у закључке. Прво се ради закључивање на основу правила П2, а затим на основу правила П3.

Закључак да пацијент има грип систем проналази у 25. кораку алгоритма тако што након израчунавања збирних мера поверења и неповерења, израчуна збирни фактор извесности у закључак, који износи:

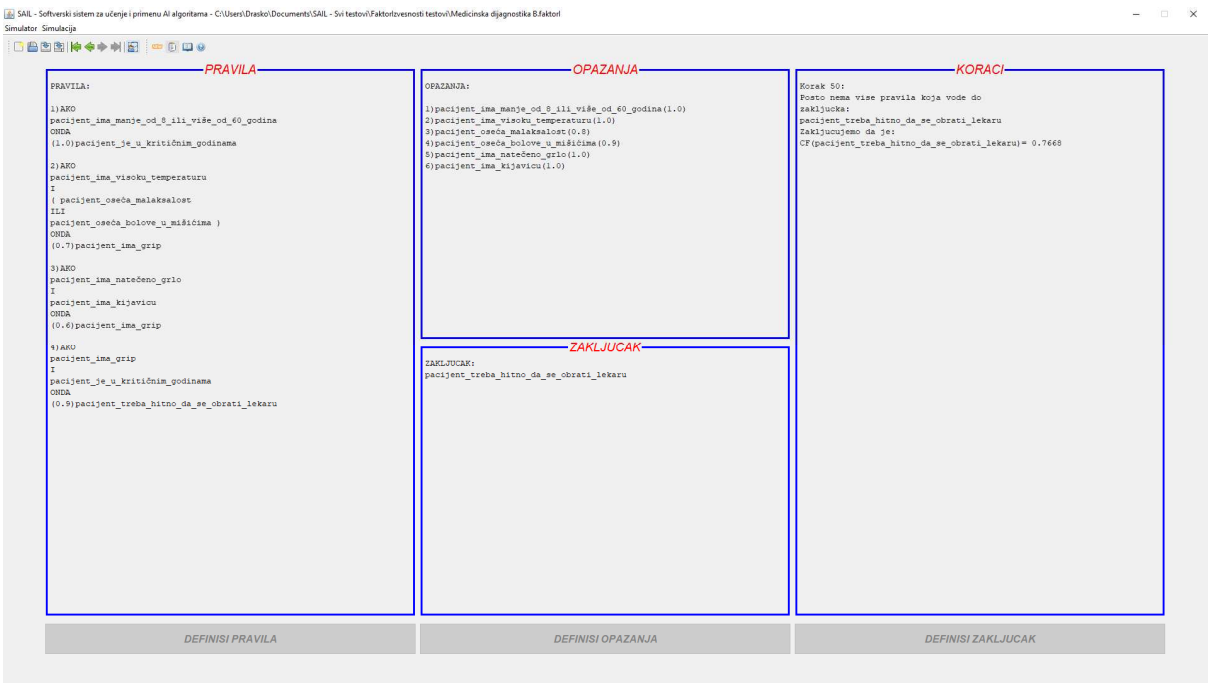
$$MBcum(z1, eP2P3) = MB(z1, eP2) + MB(z1, eP3) - MB(z1, eP2) * MB(z1, eP3) =$$

$$0.63 + 0.6 - 0.63 * 0.6 = 0.852$$

$$MDcum(z1, eP2P3) = MD(z1, eP2) + MD(z1, eP3) - MD(z1, eP2) * MD(z1, eP3) = 0.0$$

$$CF(z1) = MBcum(z1, eP2P3) - \text{MDcum}(z1, eP2P3) = 0.852 - 0.0 = 0.852$$

Закључак да пацијент треба хитно да се обрати лекару систем проналази у 50. кораку алгоритма и он износи: $CF(z2) = 0.7668$. На слици 47 приказан је последњи корак симулације када је израчунат збирни фактор извесности у овај закључак.



Слика 47 – Последњи корак симулације модула са фактором извесности

6.3.3. Примене над великом количином података

Стабла одлучивања и алгоритми машинског учења имају велику примену у анализи и обради података у различитим областима. Резултати таквих техника су много бољи, уколико је улазни скуп података који се обрађује довољно велики. Великом количином података компаније могу да предвиђају своје бизнис моделе и преиспитују бизнис одлуке.

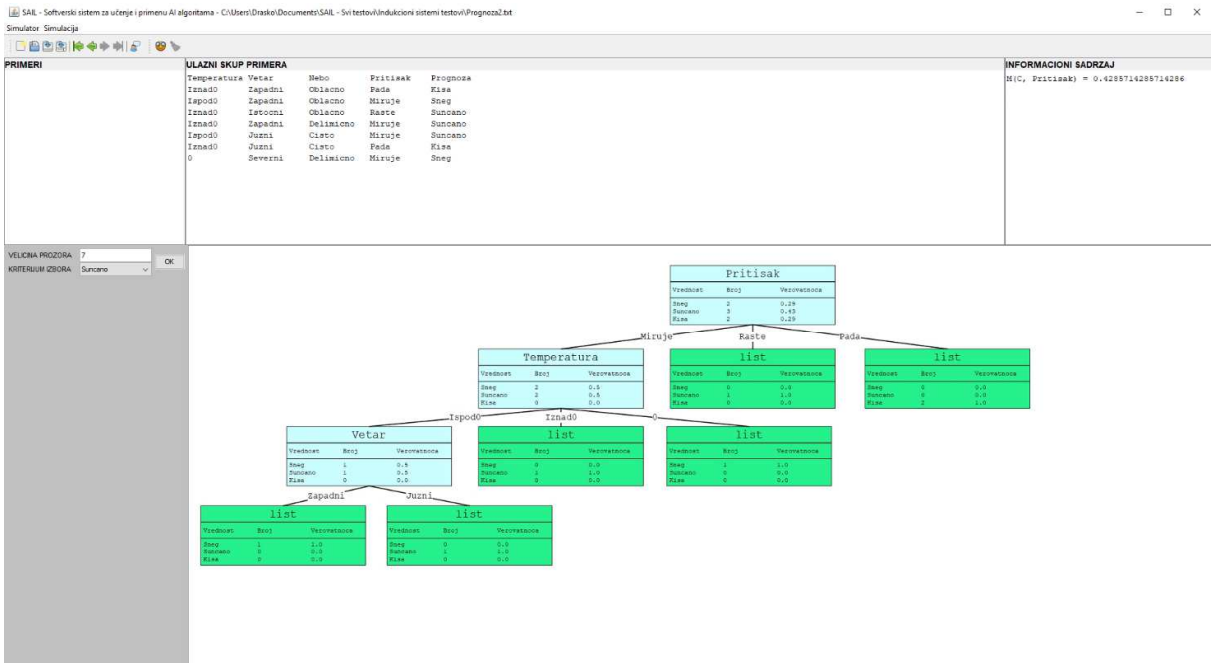
Стабла одлучивања спадају у технике надгледаног учења, које имају за циљ да на основу улазних података генеришу одређену излазну вредност. Друге технике машинског учења су: ненадгледано учење, полунадгледано учење и учење засновано на интеракцији са окружењем. Главни алгоритми техника надгледаног учења, осим стабала одлучивања су: линеарна регресија, логистичка регресија, наивни Бајес, к-најближих суседа, неуралне мреже и метода подржавајућих вектора. Овде ће бити приказана примена стабла одлучивања која раде класификацију на основу дефинисаног скупа података (енг. *dataset*), мада стабла могу бити заснована и на регресији. Из тог разлога се често називају *CART – Classification and Regression Trees*. Типичан проблем који се решава стаблима одлучивања је предвиђање цене некретнине, на основу података о тој некретнини (реон/зона у којој се налази, број просторија, квадратура, година изградње и слично).

У табели 7 дат је мањи скуп података временске прогнозе, који ћемо унети у наш систем и применити стабла одлучивања.

ТАБЕЛА 7
СКУП ПОДАТАКА ПРОГНОЗЕ ВРЕМЕНА НА ОСНОВУ ВРЕМЕНСКИХ ПАРАМЕТАРА

Температура	Ветар	Облачност	Притисак	Прогноза
4	западни	облачно	опада	киша
-5	западни	облачно	мирује	снег
3	источни	облачно	расте	сунчано
7	западни	делимично	мирује	сунчано
-1	јужни	чисто	мирује	сунчано
2	јужни	чисто	опада	киша
0	северни	делимично	мирује	снег

Категорија температуре због различитих нумеричких вредности биће подељена у три класе: изнад 0⁰, испод 0⁰ и температура 0⁰. На основу ових података, унетих у симулацију, добијамо да сунчано време може да буде у три могућа случаја (слика 48).



Слика 48 – Закључивање о прогнози времена на основу стабла одлучивања

У овом стаблу сваки унутрашњи чвор представља тест на неки одабрани атрибут, свака грана је исход тог теста, а сваки чвор лист у стаблу представља класу, односно одлуку која је донета рачунањем свих атрибута, на путањи од кореног чвора до чвора листа. Те путање називају се правилима класификације. Скуп података може бити непотпун, па се

може десити да за одређени улазни скуп примера као чвор лист немамо податке на излазу (на пример: притисак мирује, температура испод 0, а ветар источни или северни). Уколико би улазни скуп података био знатно већи и да постоје рецимо временски параметри за 100 или 300 дана, број таквих недефинисаних излаза би био значајно мањи. Уколико је број атрибута који се анализирају превелики, стабла одлучивања у том случају нису згодна ни за обраду ни за визуелизацију и тада се користе друге технике машинског учења, као што су на пример линеарна регресија или алгоритам к-најближих суседа (кНН). Проблеми које су били анализирани у току писања ове дисертације, а засновани су на техникама машинског учења и великој количини података, користили су управо те две технике.

Први експеримент била је примена над базом са више од 50 000 музичких албума са територије Србије и бивше Југославије. Циљ је био урадити кластеризацију ових улазних података и на основу њих дати препоруку кориснику који тренутно гледа веб страну одређеног музичког албума. Улаз кНН алгоритма имао је 51 хиљаду вектора са 509 димензија, које су обухватиле декаду у којој је музички албум настао (17 декада), жанр коме албум припада (15 жанрова) и стил/стилове којима албум припада (477 стилова). Најбољи резултат предикције добијен је са око 5000 кластера.

Други експеримент спроведен је над тренинг скупом од 5000 филмова, а за 500 тест примера одређене су предикције популарности и зараде тих филмова. За предвиђање популарности достигнут је проценат од 83.7% код кНН алгоритма и 81.6% код линеарне регресије, док је код предвиђања зараде тај проценат био нешто испод 60%.

Прави избор алгоритма машинског учења који треба да применимо у великој мери зависи од података које имамо, њихове количине и структуре тих података. Најбољи начин да се покаже која техника даје најбоље резултате је да се изврши више алгоритама над истим тестним подацима, а затим да се изврши упоређивање добијених резултата.

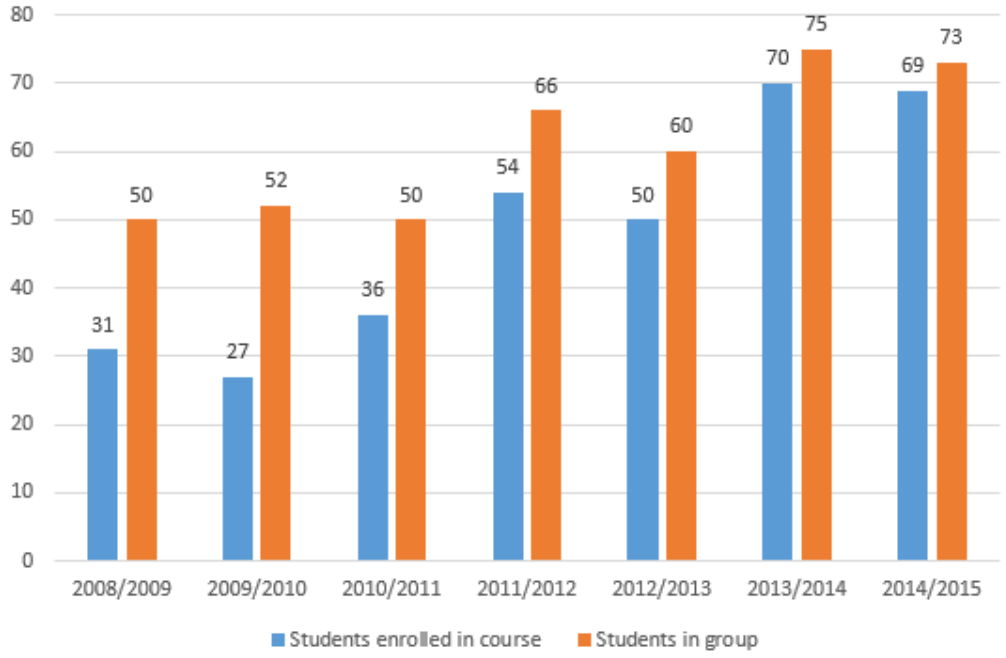
7. ЕВАЛУАЦИЈА

У школској 2009/2010. години пробно је за потребе наставе развијена симулација алгоритама претраживања, како би студенти са много више разумевања могли да прате градиво. Као резултат увођења овакве симулације, број поена који су студенти освајали на задатку из ове области био је те и наредне школске године 16.4 од могућих 20 поена, што је било знатно боље од претходне три школске године, када је просечан број поена остварен на том задатку био у границама од 8.16 поена до 11.81 поена, од могућих 20.

Прва верзија софтверског система *SAIL* обухватила је стандардне алгоритме претраживања. Друга верзија је осим алгоритама претраживања, обухватила алгоритме теорије игара, формалну и фази логику, продукционе системе, *GPS* и *STRIPS* алгоритам, студенти су имали прилику да користе у летњем семестру 2012. године [101]. Већ следеће школске године *SAIL* је укључио и алгоритме рада у неизвесном окружењу – систем са фактором извесности и систем за одржавање истинитости, семантичке мреже и оквире и алгоритме задовољења ограничења, да би школске 2013/2014. систем био комплетиран, додавањем индукционих алгоритама и симулацијом Бајесових мрежа [102]. Прву верзију система *SAIL* на мобилним уређајима студенти су почели да користе од летњег семестра школске 2014/2015. године.

Као резултат увођења овог софтверског система у наставу, на Одсеку за софтверско инжењерство (СИ), где је овај предмет био изборни, број студената који су пратили овај предмет постепено се повећавао кроз школске године, што се може видети са слике 46. На

Одсеку за рачунарску технику и информатику, током овог истраживања, овај предмет је био обавезни, тако да су увек сви студенти тог одсека слушали предмет.



Слика 49 – Број студената који су одабрали изборни курс Интелигентни системи на Одсеку за софтверско инжењерство (у периоду 2009-2015)

Ако се узме у обзир успех приликом полагања испита, у истраживању је праћен успех студената на оба одсека у 2009. години, последњој години када није коришћена никаква симулација у настави, и у периоду од 2012. до 2015. године, када је активно у настави почео да се укључује софтверски систем *SAIL*. Треба напоменути да су у периоду од 2009. године предавања и аудиторне вежбе остале у истом формату, а уведене су лабораторијске вежбе у систему *SAIL*. Као резултат имплементације овог система у табели 8 приказан је успех студената Одсека за РТИ у првом испитном року, а у табели 9 приказан је успех студената Одсека за СИ у првом испитном року. Анализиран је само први испитни рок, као релевантан (јунско-јулски или јануарско-фебруарски), јер статистика показује да између 60% и 75% студената излази на полагање овог предмета управо у првом испитном року.

Табела 8
Остварени поени студената у јуну 2009 и у периоду 2012-2015 на Одсеку за РТИ

Опсег процената остварених поена на испиту	**Испитни рок**				
	Јун 2009	**Јун 2012**	**Јун 2013**	**Јун 2014**	**Јун 2015**
90% до 100%	5.31 %	7.27 %	14.81 %	8.47 %	12.3 %
80% до 89.99%	8.85 %	10 %	8.33 %	14.41 %	11.47 %
70% до 79.99%	16.81 %	10 %	11.11 %	9.32 %	7.38 %
60% до 69.99%	20.35 %	11.82 %	4.63 %	14.41 %	17.21 %
50% до 59.99%	9.74 %	10 %	14.81 %	15.25 %	9.84 %
Испод 50%	9.74 %	12.73 %	7.41 %	9.32 %	7.38 %
Просечан број поена на испиту	63.5	68.8	76.3	72.5	75.25
% од укупног броја студената који су пропустили испит	29.20%	38.18%	38.9%	28.82%	34.42%

Табела 9
Остварени поени студената у јуну 2009 и у периоду 2012-2015 на Одсеку за СИ

Опсег процената остварених поена на испиту	**Испитни рок**				
	Јун 2009	**Јун 2012**	**Јун 2013**	**Јун 2014**	**Јун 2015**
90% до 100%	16.13 %	22.22 %	20 %	12.86 %	15.94 %
80% до 89.99%	3.22 %	7.41 %	12 %	11.43 %	11.59 %
70% до 79.99%	9.68 %	5.56 %	10 %	12.86 %	11.59 %
60% до 69.99%	16.13 %	3.70 %	10 %	11.43 %	10.14 %
50% до 59.99%	9.68 %	1.85 %	6 %	10 %	8.7 %
Испод 50%	12.90 %	3.70 %	4 %	12.86 %	8.7 %
Просечан број поена на испиту	68.2	81.5	79.25	80.33	81.86
% од укупног броја студената који су пропустили испит	32.26 %	55.56 %	38 %	28.57%	33.33%

Анализиран је број студената који су положили предмет Интелигентни системи на оба одсека на нивоу целе школске године, у свих шест испитних рокова: јун, јул, септембар, октобар, јануар, фебруар. Број студената који су добили задовољавајуће оцене пре увођења система на Одсеку за РТИ био је испод 80%, а након увођења система *SAIL* између 83% и 92%. На Одсеку за СИ пре увођења система пролазност у свим испитним роковима била је највише до 80%. Након увођења система *SAIL* пролазност на овом предмету креће се у границама од 88% до 96%, што је приказано у табели 10. Слично томе, гледајући просечан број поена на завршном испиту, укупан успех студената је знатно бољи након увођења и коришћења визуелних симулација у настави, али и припреме колоквијума и завршног испита уз помоћ система *SAIL*, код куће.

ТАБЕЛА 10
ПРОЛАЗНОСТ СТУДЕНАТА НА ИСПИТУ У ШКОЛСКИМ ГОДИНАМА ПРЕ И НАКОН УВОЂЕЊА СИСТЕМА *SAIL*

Одсек	Школска година				
	2008/09	**2011/12**	**2012/13**	**2013/14**	**2014/15**
Рачунарска техника и информатика	86/113	101/110	90/108	99/118	109/122
	76.22%	91.82%	83.33%	83.89%	89.34%
Софтверско инжењерство	23/31	51/54	48/50	62/70	65/69
	74.19%	94.44%	96.0%	88.57%	94.2%

Легенда: Однос м/н означава број студената који су положили испит наспрам броја студената који су могли да слушају предмет те школске године.

Од школске 2009/2010. на предмету Интелигентни системи уведен је домаћи задатак, који носи 20% од укупног броја поена на предмету. До те године је 20% поена било резервисано за теоријска питања, али почев од те школске године, студенти су опционо питања из теорије на завршном испиту могли да замене са домаћим задатком.

Домаћи задатак представља имплементацију неког алгоритма уз већ дате оквире графичког корисничког интерфејса, реализованог у програмском језику Јава. Неки од проблема који су студенти добијали за домаћи задатак су: *Block World*, *Nine Men's Morris*, *Freedom*, *Reversi* (*Othello*), *Wumpus world*, *Backgammon*. Њихов циљ је био да напишу алгоритам који ће најбоље решити проблем, уз што боље перформансе извршавања задатог алгоритма. На крају одбрана домаћих задатака, организује се турнир у коме међусобно играју сви студенти са својим решењима, а по троје најбољих студената на турниру добијали су бонус поене. Домаћи задатак је врло значајан за студенте, јер њиме уче и имплементирају алгоритме, спознају њихове предности и недостатке, повезују свој програмски код са класама доступним у поставци задатка и реализују једну комплетну игру засновану на алгоритмима које су учили током семестра, уз обавезан процес тестирања алгоритма и саме игре.

Уколико се погледа табела 11, може се увидети да је број студената који раде домаћи задатак почео да се повећава од школске 2011/2012. године што се временски подудара са увођењем првих реализованих симулација из система *SAIL*.

ТАБЕЛА 11
БРОЈ СТУДЕНАТА КОЈИ СУ УРАДИЛИ ДОМАЋИ ЗАДАТАК У ОДНОСУ НА УКУПАН БРОЈ СТУДЕНАТА КОЈИ СУ СЛУШАЛИ ПРЕДМЕТ

Одсек	Период одбране домаћег задатка					
	Јун 2010	**Јун 2011**	**Јун 2012**	**Јун 2013**	**Јун 2014**	**Јун 2015**
Рачунарска техника и информатика	21/113	29/107	36/110	40/108	45/118	57/122
Софтверско инжењерство	8/27	11/36	30/54	32/50	36/70	37/69

У склопу анализе извршена је студентска анкета за студенте који су похађали предмет у школској 2013/2014. години и у школској 2014/2015. години на оба одсека. Анкета је била анонимна и у њој је узело учешће око 80% студената који су слушали предмет и користили софтверски систем *SAIL*. Анкета је обухватила шест питања о систему *SAIL*:

1. Да ли Вам је кориснички интерфејс система *SAIL* интуитиван и лак за коришћење?
2. Оцените степен интеракције корисника и система, могућност прављења сопствених примера и учитавање постојећих примера из збирке задатака.
3. Оцените могућност покретања симулације корак по корак за сваки алгоритам и праћење рада симулације у систему *SAIL*.
4. У којој мери су Вам лабораторијске вежбе помогле у припреми колоквијума/испита?
5. Да ли Вам је систем *SAIL* помогао при самосталном учењу и припреми колоквијума/испита?
6. Општи утисак о систему *SAIL*.

На ова питања студенти су одговорили на скали од 1 до 5: 5 – веома задовољан, 4 – задовољан, 3 – просечно, 2 – незадовољан, 1 – врло незадовољан. Просечна оцена по питањима приказана је у табели 9. Анализа је приказала веома високе оцене на питању 4, о утицају лабораторијских вежби на знање које су студенти стекли, као и на питању 6, где су студенти оцењивали општи утисак о систему *SAIL*. Задовољавајуће одговоре студенти су дали на питањима о корисничком интерфејсу и лакоћи употребе система, степену интеракције са системом и праћењу симулације методом корак по корак. Нешто лошију оцену студенти су дали на питању 5, које се односило на самостално учење и припрему испита коришћењем *SAIL*, јер систем не може потпуно да замени традиционалне материјале и учење из уџбеника и са слајдова. Ту се може приметити разлика код анкета из 2014. и 2015. године, где је након увођења система *SAIL* на мобилним уређајима та оцена нешто већа.

Анализирано је време које су студенти утрошили за припремање испита пре увођења система *SAIL* и након увођења система *SAIL*. Постављена питања била су:

7. Колико је време које сте уложили у рад током године на овом предмету (праћење наставе, лабораторијске вежбе, домаћи задатак)?

8. Колико времена сте уложили у самосталну припрему овог испита?

Резултати су приказани у табели 12. Приметно је да студенти уз помоћ овог софтверског система утроше мање времена за припремање испита и мање времена улажу у самосталну припрему испита. Коришћењем студентских анкета током акредитационог процеса, анализирано је и оптерећење курса изражено кроз ЕСПБ. Пре увођења система *SAIL* и практичног домаћег задатка, процењени број ЕСПБ за предмет Интелигентни системи био је 6.55 (РТИ), односно 7.02 (СИ) у 2008. години, а након увођења система *SAIL* у последњој анкети из јуна 2015. године процењени број ЕСПБ за овај предмет био је 5.45 (РТИ), односно 5.52 (СИ). Ова анализа показала је да број ЕСПБ бодова одговара обиму предмета и уложеном раду студената на предмету Интелигентни системи.

ТАБЕЛА 12

ВРЕМЕ КОЈЕ СТУДЕНТИ УТРОШЕ ЗА УЧЕЊЕ

Одсек и година анкетирања	Питање #7			
	Мање од 40 сати	40-89 сати	90-180 сати	Више од180 сати
РТИ (2009)	43.40 %	45.28 %	9.43 %	1.89 %
РТИ (2015)	54.54 %	37.88 %	7.58 %	0 %
СИ (2009)	36.59 %	41.46 %	14.63 %	7.32 %
СИ (2015)	40.38 %	48.08 %	9.62 %	1.92 %
Одсек и година анкетирања	**Питање #8**			
	Мање од 7 дана	**8-15 дана**	**16-30 дана**	**Више од 30 дана**
РТИ (2009)	49.06 %	45.28 %	1.89 %	3.77%
РТИ (2015)	60.61 %	34.85 %	4.54 %	0 %
СИ (2009)	39.02 %	39.02 %	14.64 %	7.32 %
СИ (2015)	55.77 %	36.54 %	7.69 %	0 %

У табели 13 дате су и оцене којим су студенти оценили рад предметног професора и асистента у оквиру студентске анкете.

ТАБЕЛА 13

ОЦЕНЕ НАСТАВНИКА И АСИСТЕНТА У СТУДЕНТСКОЈ АНКЕТИ

Одсек	Просечна оцена Легенда: П - професор, АС - асистент Оцене на скали од 1 (најниже) до 5 (највише)					
	Јун 2013 [П]	Јун 2013 [АС]	Јун 2014 [П]	Јун 2014 [АС]	Јун 2015 [П]	Јун 2015 [АС]
Рачунарска техника и информатика	4.59	4.36	4.4	4.49	4.61	4.78
Софтверско инжењерство	4.16	3.92	4.41	4.16	4.7	4.45

8. Закључак

У овој дисертацији описан је нови софтверски систем *SAIL* за учење и примену алгоритама вештачке интелигенције. Циљ развоја овог система био је да алгоритме, који су при решавању комплексних проблема тешки за разумевање, детаљно прикаже кориснику система кроз поступак решавања, симулацију корак по корак и њихов визуелни приказ. Систем је првобитно био замишљен као једна симулација стандардних алгоритама претраживања и уведен је на предмету Интелигентни системи на Електротехничком факултету у Београду. Таква симулација се показала веома корисном за студенте, па је систем затим модуларно надограђиван са другим групама алгоритама. Током развоја система, у истраживању је дат предлог новог хибридног модела развоја софтверских система који комбинује итеративни и инкрементални приступ, са имплементацијом традиционалних фаза модификованог модела водопада, уз примену савременог агилног приступа, попут развоја вођеног тестовима (*TDD*) и понашањем (*BDD*).

Анализом 14 постојећих софтверских система који се користе у учењу алгоритама вештачке интелигенције на више од 30 светских универзитета, закључено је да не постоји систем који поседује све дефинисане захтеве: такав софтверски систем треба да има заједнички графички кориснички интерфејс за све симулације, висок ниво интеракције између корисника и система, да има могућност симулације сваког алгоритма корак по корак, упоредне анализе алгоритама током њиховог извршавања, да има могућност прављења сопствених примера са променљивим улазним параметрима симулације или учитавање постојећих предефинисаних проблема. Осим ових захтева за симулацијом и визуелизацијом,

на почетку је дефинисано да систем треба да буде модуларан, како би могао да се надогради новим алгоритмима, и вишеплатформски, тако што би се реализовале рачунарска и мобилна верзија система.

Коришћење овог софтверског система приказано је у настави на Електротехничком факултету Универзитета у Београду. Студенти помоћу овог система могу брзо и темељно да савладају предавано градиво, кроз шест формираних лабораторијских вежби, које прате теме које се предају током семестра. Извршавање сваког примера на коме уче одређени алгоритам, студент може да симулира аутоматским режимом рада, до завршетка рада алгоритма, или мануелним режимом, корак по корак. Добра страна овог софтверског система је и што осим већ осмишљених примера, из збирке задатака, који се учитавају из фајла, студент може унети и сопствене примере преко интуитивних команди и графичког корисничког интерфејса. У истраживању је приказана и могућност да се овакав систем и имплементиране симулације алгоритама надограде и реализују као реална компјутерска игра.

Евалуација у настави је већ прве године увођења оваквог система показала да студенти постижу много већи број поена у оним областима које раније нису биле покривене симулацијама алгоритама. Уз интерактивне лабораторијске вежбе и примере за самостално учење на мобилним уређајима, студенти су трошили много мање времена за самосталну припрему овог испита, него студенти који нису користили софтверски систем *SAIL*. Овакав систем би осим студената за учење могли користити и инжењери, који треба да савладају неки алгоритам и разумеју његов рад и како би након тога новостечено знање могли применити у реализацију неког новог система или подсистема.

Као примена алгоритама вештачке интелигенције у другим областима и могућност примене овог софтверског система у решавању практичних проблема, у овој дисертацији су приказане три различите примене система *SAIL*. Прва примена показује употребу симулације алгоритама претраживања у оквиру паметног паркинга са бежичном сензорском мрежом. Друга примена овог система приказала је помоћ у медицинској дијагностици и рад система у неизвесном окружењу. Трећа примена са стаблима одлучивања и обрадом великих количина података је данас најчешћа употреба реализованих алгоритама, заснованих на машинском учењу, и такви алгоритми се срећу често у другим областима где је потребно анализирати и обрађивати податке.

У оквиру докторске дисертације постигнути су следећи научни доприноси:

- Детаљан преглед и класификација постојећих софтверских система који се користе за учење алгоритама вештачке интелигенције у настави на различитим светским универзитетима, али неких и ван наставе;
- Модернизација факултетског предмета активном применом електронског учења, мобилног учења, самосталног учења студената и укључивањем више практичне наставе на предмету са до тада традиционалним извођењем наставе;
- Реализована функционална спецификација новог софтверског система за подршку учењу и примену алгоритама вештачке интелигенције. Моделовање новог софтверског система модуларног карактера, са могућношћу лаке проширивости додатним модулима и алгоритмима;
- Примена мултиплатформске архитектуре софтверског система, приближно исте логике, уз мање измене у програмском коду приликом трансформације рачунарске верзије система у мобилну верзију;
- Нови начин исцртавања графова и стабала код мобилне апликације на Андроид платформи, јер постојеће Јавине библиотеке нису оптимизоване за Андроид платформу;
- Примена игара и анимације у едукацији и разумевање алгоритама уз помоћ класичне компјутерске игре засноване на алгоритмима вештачке интелигенције;
- Анализа резултата коришћења софтверског система за учење вештачке интелигенције у оквиру предмета на основним академским студијама и позитивна оцена од стране студената који су били укључени у истраживање;
- Унапређење пројектовања система за учење и примену алгоритама ВИ, и предлог хибридног модела развоја таквих софтверских система.

Истраживање у оквиру ове дисертације отворило је неколико могућих даљих праваца унапређења овог софтверског система. Прва идеја је да се систем повеже са базом питања, реализованом у оквиру истог система, или са неком од већ описаних платформи за е-учење, тако да студент или инжењер, док извршава симулацију, може да одговара на питања и на тај

начин проверава своја знања. На тај начин би се интерактивност корисника и софтверског система подигла на највиши ниво. Друга идеја унапређења овог система је да се примери које корисник покрене чувају на неком серверу или у облаку и да сами корисници могу да реализују своје подсистеме са својим имплементацијама алгоритама. На таквој платформи корисници би могли међусобно да се такмиче, реализују тест примере другим корисницима платформе или праве интелигентне агенте (рачунарске ботове). Трећи правац развоја могла би бити аутоматска класификација улазних примера и одлучивање од стране система у реалном времену. Систем би самостално одлучивао који алгоритам треба да примени на основу онога што је корисник поставио као проблем систему. Ово је врло значајно у области алгоритама машинског учења и обраде података, где би систем могао самостално да одлучи да ли да примењује технику надгледаног учења или ненадгледаног учења. Како ове врсте учења имају већи број различитих алгоритама и стратегија које би систем могао да одабере, упоредним приказом би систем могао да предложи различите начине да се постављени проблем реши и да одлучи који од предложених начина је најбољи.

Реализовани систем *SAIL* и предложена унапређења овог система показују да је значај вештачке интелигенције чак и код решавања мањих проблема јако велики. Велике компаније данас улажу доста новца, времена и људских ресурса у развој система заснованих на вештачкој интелигенцији. Системи који укључују неуралне мреже, машинско учење, дубоко структурно учење, постају паметнији и свеснији, спремни су да самостално уче без унапред дефинисаног понашања алгоритама и такви системи биће сасвим сигурно стубови нове дигиталне технолошке револуције.

Литература

Сви извори које је аутор користио у току израде овог рада наведени су редом којим су референцирани у раду. За сваки наведени извор је приложена интернет адреса, ако је била доступна у тренутку стварања овог списка. Формат је усклађен са *IEEE* упутством за навођење референци, датим у [103].

[1] H. Jabbar and R. Khan, "Survey on Development of Expert System from 2010 to 2015," in *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies*, Udaipur, India, 2016.

[2] J. Borana, "Applications of Artificial Intelligence and Associated Technologies," in *Proceedings of International Conference on Emerging Technologies in Engineering, Biomedical, Management and Science*, Jodhpur, India, 2016.

[3] R. Moreno and R. Mayer, "Interactive multimodal learning environments," *Educ Psychol Rev,* vol. 19, no. 3, pp. 309-326, 2007.

[4] T. Martin-Blas and A. Serrano-Fernandez, "The role of new technologies in the learning process: Moodle as a teaching tool in Psysics," *Computers & Education,* vol. 52, no. 1, pp. 35-44, 2009.

[5] G. Pankaj Jain, V. Gurupur, J. Schroeder and E. Faulkenberry, "Artificial Intelligence - Based Student Learning Evaluation: A Concept Map-Based Approach for Analyzing a Student's Understanding of a Topic," *IEEE Transactions on Learning Technologies,* vol. 7, no. 3, pp. 267-279, 2014.

[6] J. DeNero and D. Klein, "Teaching Introductory Artificial Intelligence with Pac-Man," in *Proceedings of the 1st Symposium on Educational Advances in Artificial Intelligence*, Atanta, USA, 2010.

[7] L. Hernandez, C. Serrano and J. Builes, "Artificial Intelligence techniques in an e-learning environment for IT engineering projects," in *Technologies Applied to Electrionics Teaching*, Seville, Spain, 2016.

[8] H. Grob, F. Bensberg and B. Dewant, "Developing, deploying, using and evaluating an open source learning management system," in *26th International Conference on Information Technology Interfaces*, Cavtat, Croatia, 2004.

[9] M. Limniou and M. Smith, "Teachers' and students' perspectives on teaching and learning through virtuall learning environments," *European Journal of Engineering Education,* vol. 35, no. 6, pp. 645-653, 2010.

[10] B. Nikolic, Z. Radivojevic, J. Djordjevic and V. Milutinovic, "A Survey and Evaluation of Simulators Suitable for Teaching Courses in Computer Architecture and Organization," *IEEE Transactions on Education,* vol. 52, no. 4, pp. 449-458, 2009.

[11] P. Gil, G. Garcia, A. Delgado, R. Medina, A. Calderon and P. Marti, "Computer networks virtualization with GNS3: Evaluating a solution to optimize resources and achieve a distance learning," in *Proceeding of IEEE Frontiers in Education Conference*, Madrid, Spain, 2014.

[12] Y. Zhang, R. Liang and H. Ma, "Teaching Innovation in Computer Network Course for Undergraduate Students with Packet Tracer," *IERI Procedia,* vol. 2, pp. 504-510, 2012.

[13] B. Koldehofe, M. Papatriantafilou and P. Tsigas, "LYDIAN: An Extensible Educational Animation Environment for Distributed Algorithms," *ACM Journal on Educational Resource in Computing,* vol. 6, no. 2, pp. 1-21, 2006.

[14] J. Thakur and N. Kumar, "DES, AES and Blowfish: Symetric Key Cryptography Algorithms Simulation Based Performance Analysis," *International Journal of Emerging Technology and Advanced Engineering,* vol. 1, no. 2, pp. 6-12, 2011.

[15] Z. Stanisavljevic, J. Stanisavljevic, P. Vuletic and Z. Jovanovic, "COALA - System for Visual Representation of Cryptography Algorithms," *IEEE Transactions on Learning Technologies,* vol. 7, no. 2, pp. 178-190, 2014.

[16] "Computing Curricula 2005 - The Overview Report covering undergradute degree programs in CE, CS, IS, IT and SE," [Online]. Available: http://www.acm.org/education/education/curric_vols/CC2005-March06Final.pdf. [Accessed 18 09 2015].

[17] "Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering," [Online]. Available: http://www.acm.org/education/curric_vols/CE-Final-Report.pdf. [Accessed 16 09 2015].

[18] "Computer Science Curricula 2013 - Curriculum Guidelines for Undergraduate Degree Programs in Computer Sceince," [Online]. Available: https://www.acm.org/education/CS2013-final-report.pdf. [Accessed 11 07 2015].

[19] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, Pearson, 2009.

[20] M. Alfonseca, "Frames, semantic networks, and object-oriented programming in APL2," *IBM Journal of Research and Development,* vol. 33, no. 5, pp. 502-510, 1989.

[21] M. Minsky, "A framework for representing knowledge," in *Computation & intelligence*, Menlo Park, CA, USA, American Association for Artificial Intelligence , 1995, pp. 163-189.

[22] S. U. Khan, M. Khan and M. Nauman, "Semi-automatic knowledge transformation of semantic network ontologies into Frames structures," in *Sixth International Conference on Innovative Computing Technology 2016*, Dublin, Ireland, 2016.

[23] A. Newell, J. Shaw and H. Simon, "Report on a general problem-solving program," in *Proceedings of the International Conference on Information Processing*, 1959.

[24] R. Fikes and N. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence,* vol. 69, no. 1-2, pp. 189-208, 1971.

[25] J. Orkin, "Three States and a Plan: The A.I. of F.E.A.R.," in *Game Developers Conference 2006*, San Jose, CA, USA, 2006.

[26] T. Bylander, "The computational complexity of propositional STRIPS planning," *Artificial Intelligence,* vol. 69, no. 1-2, pp. 165-204, 1994.

[27] V. Kumar, "Algorithms for Constraint-Satisfaction Problems: A Survey," *ACM AI Magazine,* vol. 13, no. 1, pp. 32-44, 1992.

[28] A. Pulka, "On strategies for solving boolean satisfiability problems," in *International Conference on Signals and Electronic Systems 2012*, Wroclaw, Poland, 2012.

[29] L. De Moura and N. Bjorner, "Satisfiability modulo theories: introduction and applications," *Communications of the ACM,* vol. 54, no. 9, pp. 69-77, 2011.

[30] G. Brewka, T. Eiter and M. Truszczynski, "Answer set programming at a glance," *Communications of the ACM,* vol. 54, no. 12, pp. 92-103, 2011.

[31] E. Shortliffe, R. Davis, S. Axline, B. Buchanan, C. Green and S. Cohen, "Computer-based consultations in clinical therapeutics: Explanation and rule acquisition capabilities of the MYCIN system," *Computers and Biomedical Research,* vol. 8, no. 4, pp. 303-320, 1975.

[32] E. H. Shortliffe, Computer-Based Medical Consultations: MYCIN, New York, USA: Elsevier, 1976.

[33] C. WenBin, L. XiaoLing, L. YiJun and F. Yu, "A machine learning algorithm for expert system based on MYCIN model," in *Proc. on 2nd International Conference on Computer Engineering and Technology*, Chengdu, China, 2010.

[34] B. De Baets and J. Fodor, "Van Melle's combining function in MYCIN is a representable uninorm: An alternative proof," *Fuzzy Sets and Systems,* vol. 104, no. 1, pp. 133-136, 1999.

[35] J. Doyle, "A truth maintenance system," *Artificial Intelligence,* vol. 12, no. 3, pp. 231-272, 1979.

[36] Y. Ren and J. Pan, "Optimising ontology stream reasoning with truth maintenance system," in *Proceedings of the 20th ACM international conference on Information and knowledge management*, Glasgow, UK, 2011.

[37] L. Rokach and O. Maimon, "Decision trees," in *Data Mining and Knowledge Discovery Handbook vol. 6*, Heidelberg, Springer, 2005, pp. 165-192.

[38] S. Sathyadevan and R. Nair, "Comparative Analysis of Decision Tree Algorithms: ID3, C4.5 and Random Forest," in *Computational Intelligence in Data Mining - Volume 1*, Springer, 2015, pp. 549-562.

[39] B. Hssina, A. Merbouha, H. Ezzikouri and M. Erritali, "A comparative study of decision tree ID3 and C4.5," *International Jounral of Advanced Computer Science and Applications,* no. spec. issue on Advances in Vehicular Ad Hoc Networking and Applications, 2014.

[40] D. Draskovic and B. Nikolic, "Analiza edukacionih simulatora u nastavi iz veštačke inteligencije," in *Zbornik radova 57. konferencije ETRAN 2013*, Zlatibor, Srbija, 2013.

[41] "The Chinese Room simulation," The Mind Project, Illinois State University, US [online], 1 3 2016. [Online]. Available: http://www.mind.ilstu.edu.

[42] "MIT Open Course Ware software system (course: Artificial Intelligence, Massachusetts Intitute of Technology, US, Prof. P.H. Winston)," [Online]. Available: http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-034-artificial-intelligence-fall-2010/demonstrations/. [Accessed 1 3 2016].

[43] "AI search simulation," [Online]. Available: http://www.cs.rmit.edu.au/AI-Search/. [Accessed

1 3 2016].

[44] P. McDonald and V. Ciesielski, "Design and Evaluation of an Algorithm Animation of State Space Search Methods," *J. Comp. Sci. Educ.,* vol. 12, no. 4, pp. 301-324, 8 2010.

[45] "AIspace software system," [Online]. Available: http://www.aispace.org/. [Accessed 15 3 2016].

[46] S. Amershi, N. Arksey, G. Carenini, C. Conati, A. Mackworth, H. Maclaren and D. Poole, "Designing CIspace: Pedagogy and Usability in a Learning Environment for AI," in *Proc. 10th Annu. SIGCSE conf. on Innovation and technology in Comp. Sci. Educ.*, Monte de Caparica, Portugal, 2005.

[47] "AIMA3e software system," [Online]. Available: http://aima-java.googlecode.com/svn/trunk/aima-all/release/aima3ejavademos.html. [Accessed 28 03 2015].

[48] "Searcher software system," [Online]. Available: http://www.csc.liv.ac.uk/~cs8js/code/coffeescript/project-testing/. [Accessed 5 5 2013].

[49] "Simple expert system simulation," [Online]. Available: http://lucy.ukc.ac.uk/ExpertSys/ExpertSys.html. [Accessed 5 5 2013].

[50] "Simulator Sudoku Solver," [Online]. Available: http://sudoku.unl.edu/SudokuSet/SudokuSetV2/index2.html. [Accessed 5 5 2013].

[51] "AI Exploratorium software system," [Online]. Available: http://webdocs.cs.ualberta.ca/~aixplore/. [Accessed 5 5 2013].

[52] M. Pantic, R. Zwitserloot and R. Grootjans, "Teaching Introductory Artificial Intelligence Using a Simple Agent Framework," *IEEE Trans. on Educ.,* vol. 48, no. 3, pp. 382-390, 8 2005.

[53] I. Borm and L. Rothkrantz, "Teaching Artificial Intelligence Techniques Using Multi-Agent Soccer," in *Proc. 3rd E-learning conf.*, Coimbra, Portugal, 2006.

[54] R. Hill and A. Van den Hengel, "Experiences with simulated robot soccer as a teaching tool," in *Proc. 3rd Int. Conf. on Information Technol. and Applicat.*, Sydney, NSW, Australia, 2005.

[55] I. Russell, Z. Markov, T. Neller and S. Coleman, "MLeXAI: A Project-Based Application-Oriented Model," *J. ACM Trans. Comput. Educ.,* vol. 10, no. 3, pp. 1-36, 2010.

[56] Z. Markov, I. Russell, T. Neller and T. Coleman, "Enhancing undergraduate AI courses through machine learning projects," in *Proc. 35th Annu. Conf. Frontiers in Educ.*, Indianopolis, IN, USA, 2005.

[57] I. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes and S. Cunningham, "Weka: Practical machine learning tools and techniques with Java implementations," University of Waikato, Hamilton, New Zealand, 1999.

[58] M. Hall, E. Frank, G. Holmes, B. Pfahringer and P. W. I. Reutemann, "The WEKA data mining software: an update," *ACM SIGKDD Explorations Newslett.,* vol. 11, no. 1, pp. 10-18, 2009.

[59] D. Draskovic, M. Cvetanovic and B. Nikolic, "SAIL - Software system for learning AI algorithms," *Computer Applications in Engineering Education,* vol. 1, no. SE, pp. 1-22, 2018.

[60] I. Sommerville, Software Engineering, Pearson, 2009.

[61] R. Pressman, Software Engineering - A Practitioner's Approach, McGraw-Hill Education, 2014.

[62] K. Beck, "Embracing change with extreme programming," *Computer,* vol. 32, no. 10, pp. 70-

77, 1999.

[63] L. Rising and N. Janoff, "The Scrum software development process for small teams," *IEEE Software,* vol. 17, no. 4, pp. 26-32, 2000.

[64] J. Highsmith and A. Cockburn, "Agile software development: the business of innovation," *Computer,* vol. 34, no. 9, pp. 120-127, 2001.

[65] J. Highsmith, Adaptive Software Development: A Collaborative Approach to Managing Complex Systems, New York, US: Dorset House Publishing Co., 2000.

[66] J. Stapleton, DSDM, Dynamic Systems Development Method: The Method in Practice, Cambridge University Press, 1997.

[67] S. Palmer and J. Felsing, A Practical Guide to Feature-driven Development, Prentice Hall, 2002.

[68] "Graphviz - Graph Visualization Software," [Online]. Available: http://graphviz.org/. [Accessed 18 07 2015].

[69] D. Radivojevic, D. Draskovic, Z. Radivojevic and M. Cvetanovic, "Java based tool for fault detection processing and result visualization," *Serbian journal of electrical engineering,* vol. 10, no. 1, pp. 185-198, 2013.

[70] S. Tubic and D. Draskovic, "Realizacija algoritma A* u kompjuterskoj igri," in *Zbornik radova 59. konferencije ETRAN 2015*, Srebrno jezero, Srbija, 2015.

[71] M. Prodanov and D. Draskovic, "Softverska simulacija inteligentnog sistema baziranog na algoritmima pretraživanja," in *Zbornik radova 22. konferencije YU INFO 2016*, Kopaonik, Srbija, 2016.

[72] M. Vukasović, M. Mićović, U. Radenković, V. Jocović and D. Drašković, "Primena virtuelne stvarnosti u okviru medicinskih tretmana," in *Zbornik radova 23. konferencije YU INFO 2017*, Kopaonik, Srbija, 2017.

[73] S. Deterding, D. Dixon, R. Khaled and L. Nacke, "From game design elements to gamefulness: defining 'gamification'," in *Proc. 15th ACM Int. Academic MindTrek Conf.*, Tampere, Finland, 2011.

[74] A. McGovern, Z. Tidwell and D. Rushing, "Teaching Introductory Artificial Intelligence through Java-Based Games," in *Proc. 2nd Symp. on Edu. Advances in Artificial Intelligence*, San Francisco, California, USA, 2011.

[75] S. Borges, V. Durelli, H. Reis and S. Isotani, "A systematic mapping on gamification applied to education," in *Proc. 29th Annu. ACM Symp. on Applied Computing*, Gyeongju, Republic of Korea, 2014.

[76] M. Ibanez, A. Di-Serio and C. Delgado-Kloos, "Gamification for Engaging Computer Science Students in Learning Activities: A Case Study," *IEEE Trans. Learning Technologies,* vol. 7, no. 3, pp. 291-301, 2014.

[77] Q. Liu, L. Diao and G. Tu, "The Application of Artificial Intelligence in Mobile Learning," in *Proc. Int. Conf. on System Science, Engineering Design and Manufacturing Informatization*, Yichang, China, 2010.

[78] N. Harshfield, D. Chang and R. Ragade, "A Unity 3D Framework for Algorithm Animation," in *Proc. 20th Int. Conf. on Computer Games*, Louisville, KY, USA, 2015.

[79] Z. He, M. Shi and C. Li, "Research and Application of Path-finding Algorithm Based on Unity 3D," in *Proc. IEEE 15th Int. Conf. Computer and Information Science*, Okayama, Japan, 2016.

[80] D. Liu, C. Dede, R. Huang and J. Richards, Virtual, Augmented, and Mixed Realities in Education, Cambridge, USA: Springer, 2017.

[81] P. Bradford, M. Porciello, N. Balkon and D. Backus, "The Blackboard Learning System: The Be All and End All in Educational Instruction?," *Journal of Educational Technology Systems,* vol. 35, no. 3, pp. 301-314, 2007.

[82] M. Dougiamas and P. Taylor, "Improving the effectiveness of tools for Internet based education," in *Proceedings of the 9th Annual Teaching Learning Forum*, Perth, Australia, 2000.

[83] "Desire2Learn Brightspace Learning Management System," D2L, [Online]. Available: https://www.d2l.com/. [Accessed 16 9 2017].

[84] "Canvas Learning Management System," Instructure Global Ltd. London, UK, [Online]. Available: https://www.canvaslms.com/. [Accessed 16 9 2017].

[85] "Moodle in Serbia," [Online]. Available: https://moodle.net/sites/index.php?country=RS. [Accessed 5 5 2018].

[86] P. Subramanian, N. Zainuddin and S. Alatawi, "A Study of Comparison between Moodle and Blackboard based on Case Studies for Better LMS," *Journal of Information System Research and Innovation,* vol. 6, pp. 26-33, 2014.

[87] A. Carvalho, N. Areal and J. Silva, "Students' perceptions of Blackboard and Moodle in a Portuguese university," *British Journal of Educational Technology,* vol. 42, no. 5, pp. 824-841, 2011.

[88] N. Kalogeropoulos, I. Tzigounakis, E. A. Pavlatou and A. Boudouvis, "Computer-based assessment of student performance in programing courses," *Computer Applications in Engineering Education,* vol. 21, no. 4, pp. 671-683, 2013.

[89] M. Blanco, M. Rosa Estela, M. Ginovart and J. Saà, "Computer Assisted Assessment through Moodle Quizzes for Calculus in an Engineering Undergraduate Course," *Quaderni di Ricerca in Didattica,* vol. 19, no. 2, pp. 78-83, 2009.

[90] J. Ramos, M. Trenas, E. Gutierrez and S. Romero, "E-assessment of Matlab assignments in Moodle: Application to an introductory programming course for engineers," *Computer Applications in Engineering Education,* vol. 21, no. 4, pp. 728-736, 2013.

[91] H. Meishar-Tai and E. Pieterse, "Facebook groups as LMS: A case study," *The international review of research in open and distributed learning,* vol. 13, no. 4, pp. 33-48, 2012.

[92] "E-learning Moodle platforma, Univerzitet u Beogradu," [Online]. Available: http://elearning.rcub.bg.ac.rs. [Accessed 1 3 2018].

[93] D. Draskovic, M. Misic and Z. Stanisavljevic, "Transition From Traditional to LMS Supported Examining: A Case Study in Computer Engineering," *Computer Applicatoins in Engineering Education,* vol. 24, no. 5, pp. 775-786, 2016.

[94] S. Delcev, M. Vukasovic, D. Draskovic, D. Radojevic, M. Jankovic, M. Bajec and B. Nikolic, "Testing artificial intelligence knowledge with interactive web system," in *Proceedings of the 24th Telecommunications Forum TELFOR*, Belgrade, Serbia, 2016.

[95] "IBM Global Parking Survey: Drivers Share Worldwide Parking Woes," 2011. [Online]. Available: https://www-03.ibm.com/press/us/en/pressrelease/35515.wss. [Accessed 20 11 2017].

[96] D. Draskovic and S. Vukicevic, "A model of software system for parking using search algorithms," in *Proceedings of the 35th International Conference MIPRO 2012*, Opatija,

Croatia, 2012.

[97] J. Chinrungrueng, U. Sunantachaikul and S. Triamlumlerd, "Smart Parking: An Application of Optical Wireless Sensor Network," in *International Symposium on Applications and the Internet Workshops*, Hiroshima, Japan, 2007.

[98] M. Chen and T. Chang, "A Parking Guidance and Information System based on Wireless Sensor Network," in *IEEE International Conference on Information and Automation*, Shenzhen, China, 2011.

[99] S. Srikanth, P. Pramod, K. Dileep, S. Tapas, M. Patil and N. Sarat Chandra Babu, "Design and Implementation of a Prototype Smart PARKing (SPARK) System Using Wireless Sensor Networks," in *International Conference on Advanced Information Networking and Applications Workshops*, Bradford, United Kingdom, 2009.

[100] D. Bojić, M. Gligorić and B. Nikolić, Zbirka zadataka iz ekspertskih sistema, Beograd: Elektrotehnički fakultet u Beogradu, 2009.

[101] D. Draskovic and B. Nikolic, "Software system for expert systems learning," in *Proc. IEEE Int. Conf. AFRICON 2013*, Pointe-Aux-Piments Mauritius, 2013.

[102] K. Milenkovic, D. Draskovic and B. Nikolic, "Educational software system for reasoning and decision making using Bayesian networks," in *Proc. IEEE Global Eng. Educ. Conf. EDUCON 2014*, Istanbul, Turkey, 2014.

[103] "The IEEE Editorial Style Manual," 2017. [Online]. Available: http://ieeeauthorcenter.ieee.org/wp-content/uploads/IEEE_Style_Manual.pdf. [Accessed 20 3 2018].

СПИСАК СКРАЋЕНИЦА

Скраћенице на српском језику:

ВИ - Вештачка интелигенција

ЕСПБ - Европски систем преноса и акумулације бодова

КНФ - Конјуктивна нормална форма

РТИ - Рачунарска техника и информатика

СИ - Софтверско инжењерство

ХФ - Хеуристичка функција

- - - - - - - - - - - - - - -

Скраћенице на енглеском језику:

AC-3 - *Arc Consistency Algorithm #3*

ACM - *Association for Computing Machinery*

ADTree - *Alternating Decision Tree*

ANTLR - *ANother Tool for Language Recognition*

API - *Application Programming Interface*

AR - *Augmented Reality*

AWT – *Abstract Window Toolkit*

BDD - *Behavior Driven Development*

BFTree - *Best-First decision Tree classifier*

CNF - *Conjunctive Normal Form*

CF - *Certainity Factor*

CP - *Conditional-proof justifications*

CS - *Constraint Satisfaction*

CSP - *Constraint Satisfaction Problem*

ECTS - *European Credit Transfer System*

FSM - *Finite State Machine*

GPS - *General Problem Solver*

Graphiz - *Graph Visualization Software*

GUI - *Graphic User Interface*

IEEE - *Institute of Electrical and Electronics Engineers*

IDA* - *Iterative deepening A* search algorithm*

ID3 - *Iterative Dichotomiser 3*

JUNG – *Java Universal Network-Graph Framework*

LMS - *Learning Management System*

MB - *Measure of increased Belief*

MD - *Measure of increased Disbelief*

MLeXAI - *Machine Learning Experiences in AI*

NP - *Non-deterministic Polynomial-time*

PDF - *Portable Document Format*

SAIL - *System for Artificial Intelligence Learning*

SL - *Support list*

STRIPS - *STanford Research Institute Problem Solver*

SSAO - *Screen Space Ambient Occlusion*

TDD - *Test Driven Development*

TMS - *Truth Maintenance System*

VR - *Virtual Reality*

WEKA - *Waikato Environment for Knowledge Analysis*

СПИСАК СЛИКА

СПИСАК ТАБЕЛА

БИОГРАФИЈА АУТОРА

Дражен Драшковић рођен је 16. децембра 1985. године у Београду, Република Србија. Гимназију у Смедереву, природно-математичког смера, завршио је са одличним успехом 2004. године. Током школовања освајао је награде на регионалним и републичким такмичењима из математике, физике и информатике, био је учесник Школе младих математичара „Архимедес“ и учесник летњих научних школа из физике и информатике, у истраживачкој станици Петница.

Након завршене средње школе, уписао је Одсек за софтверско инжењерство, Електротехничког факултета Универзитета у Београду. Као први студент Одсека за софтверско инжењерство, дипломирао је 2009. године, са просечном оценом 8.76 и оценом 10 на дипломском раду. Дипломски рад, под називом „Систем за управљање и уређивање веб садржаја“ активно се користио у Министарству просвете, науке и технолошког развоја.

Дипломске академске студије - мастер, уписао је 2009. године на Електротехничком факултету у Београду, модул Софтверско инжењерство. Мастер студије је завршио 2011. године са просечном оценом 9.5 и оценом 10 на завршном мастер раду на истом факултету. У току 2010. године боравио је у истраживачко-развојном центру компаније IBM у Штутгарту (Немачка), на пројекту „*Smarter Planet*“, у склопу летње праксе најбољих мастер студената електротехнике и рачунарства из Европе, Средњег Истока и Африке. Докторске студије уписао је у децембру 2011. године на Електротехничком факултету Универзитета у Београду на модулу Софтверско инжењерство. У току својих студија, био је ангажован од марта 2008. до децембра 2009. године на позицији програмера у Рачунском центру

Електротехничког факултета, на већем броју софтверских пројеката, које је радио и самостално и у тиму.

Од јануара 2010. године запослен је као сарадник у настави на Катедри за рачунарску технику и информатику Електротехничког факултета у Београду, а од октобра 2012. налази се на позицији асистента. Тренутно је ангажован као асистент и изводи аудиторне и лабораторијске вежбе на 8 предмета на основним и мастер академским студијама, а учествовао је у настави на још 6 предмета. Радио је хонорарно као стручни сарадник, софтверски инжењер и интегратор софтверских система на више од 20 софтверских пројеката у државним и међународним институцијама, као што су софтверски системи за Министарство просвете, науке и технолошког развоја, Министарство телекомуникација, *United Nations Office for Project Services* (*UNOPS*), *World Health Organization* (*WHO*) и друге институције.

Члан је интернационалних струковних организација *IEEE* (*Institute of Electrical and Electronics Engineers*) од 2013. године и *ACM* (*Association for Computing Machinery*) од 2016. године. У организацији IEEE од јануара 2017. године налази се на позицији секретара и благајника за секцију *Serbia and Montenegro*, одељак *Computational Intelligence*. Био је активан члан радне групе Факултета у процедури самовредновања и провере квалитета и процесу акредитације Факултета и свих студијских програма (од 2012.), члан Радне групе за сарадњу са привредом и компанијама (од 2016.) и члан Савета Факултета (од 2015.).

Област истраживања Дражена Драшковића обухвата вештачку интелигенцију и интелигентне системе, анализу и обраду података, методологије пројектовања и тестирања софтверских система и примену различитих алгоритама из области вештачке интелигенције у софтверским системима базираним на интернет и мобилним технологијама.

Досадашњи резултати кандидата приказани су у виду 36 научних радоваа, од чега су три рада у међународним часописима са импакт-фактором (*SCI* листа), 12 радова у зборницима међународних скупова, два рада у домаћим часописима и 19 радова у зборницима скупова националног значаја. Дражен Драшковић је коаутор два техничка решења и три уџбеника на Електротехничком факултету Универзитета у Београду.

СПИСАК РАДОВА КАНДИДАТА

Научни радови у научним часописима међународног значаја у вези теме доктората:

1.1 **Draskovic, D.,** Cvetanovic, M., Nikolic, B., **"SAIL – Software system for learning AI algorithms"**, *Computer Applications in Engineering Education*, spec. issue, Wiley, June 2018. **[M22, IF = 1.153]**, DOI: 10.1002/cae.21988, ISSN: 1099-0542

1.2 **Draskovic, D.,** Misic, M., Stanisavljevic, Z., **"Transition from traditional to LMS supported examining: A case study in computer engineering"**, *Computer Applications in Engineering Education*, Vol. 24, No. 5, pp. 775-786, Wiley, Sept. 2016. **[M23, IF = 0.694]**, DOI: 10.1002/cae.21750, ISSN: 1099-0542

Научни радови у зборницима међународних научних скупова у вези теме доктората:

2.1 Delčev, S., Vukasović, M., **Drašković, D.**, Radojević, D., Janković, M., Bajec, M., Nikolić, B., **"Testing artificial intelligence knowledge with interactive web system"**, *Proceedings of 24th Telecommunications Forum TELFOR 2016*, IEEE Serbia & Montenegro, Belgrade, Serbia, Nov. 2016 **[M33]**, DOI: 10.1109/TELFOR.2016.7818922

2.2 Milenkovic, K., **Draskovic, D.,** Nikolic, B.**, "Educational software system for reasoning and decision making using Bayesian networks"**, *Global Engineering Education Conference (EDUCON)*, 2014 IEEE, pp. 1189-1194, IEEE, Istanbul, Apr, 2014 **[M33]**, DOI: 10.1109/EDUCON.2014.7130489

2.3 **Drašković, D.,** Nikolić, B., **"Software System for Expert Systems learning"**, *IEEE Africon 2013*, pp.870-875, IEEE, Mauritius, Sep, 2013 **[M33]**, DOI: 10.1109/AFRCON.2013.6757804

2.4 **Drašković, D.,** Vukićević, S., **"A Model of Software System for Parking Using Search Algorithms"**, *MIPRO 2012, 35th International Convention on Information and Communication Technology, Electronics and Microelectronics,* pp. 1098-1103, IEEE and MIPRO Croatian Society, Opatija, Croatia, May, 2012 **[M33]**, *Electronic ISBN*: 978-953-233-068-7, *Print ISBN*: 978-1-4673-2577-6

2.5 **Drašković, D.,** Milutinović, V., **"Hybrid Approaches to Mutation in Genetic Search Algorithms"**, *6th IEEE International Conference on Intelligent Systems*, pp. 336-340, IEEE, Sofia, Bulgaria, Sept. 2012 **[M33]**, DOI: 10.1109/IS.2012.6335157

2.6 Vukićević, S., **Drašković, D., "Process of moving from Waterfall to Agile Project Management Model"**, *XIII International Symposium „SymOrg 2012",* pp. 1581-1586, University of Belgrade, Faculty of Organizational Sciences, Zlatibor, Serbia, Jun 2012 **[M33]**, ISBN: 978-86-7680-255-5

Научни радови у зборницима скупова националног значаја у вези теме доктората:

3.1 **Drašković, D.**, Kojić, N., Mićović, M., Radenković, U., **„Implementacija sistema za prikupljanje podataka, generisanje klastera i preporuka pomoću mašinskog učenja"**, *24. konferencija "YU INFO" - Zbornik radova*, Društvo za informacione sisteme i računarske mreže, Kopaonik, Srbija, Mart 2018 **[M63]**

3.2 Prodanov, M., **Drašković, D.**, **„Softverska simulacija inteligentnog sistema baziranog na algoritmima pretraživanja"**, *22. konferencija "YU INFO" - Zbornik radova*, pp. 379-383, Društvo za informacione sisteme i računarske mreže, Kopaonik, Srbija, Mar, 2016 **[M63]**, ISBN: 978-86-85525-17-9

3.3 Tubić, S., **Drašković, D.**, **„Realizacija algoritma A* u kompjuterskoj igri",** *59. konferencija "ETRAN 2015" - Zbornik radova*, Društvo za elektroniku, telekomunikacije, računarstvo, automatiku i nuklearnu tehniku, Srebrno jezero, Srbija, Jun, 2015 **[M63]**, ISBN: 978-86-80509-71-6

3.4 Stanisavljević, Ž., **Drašković, D.,** Mišić, M., **„Primena Moodle platforme u nastavi računarske tehnike i informatike"**, *22nd Telecommunications Forum TELFOR 2014 - Zbornik radova*, Beograd, pp. 1039-1042, Nov, 2014 **[M63]**, DOI: 10.1109/TELFOR.2014.7034584

3.5 **Drašković, D.,** Nikolić, B., **„Analiza edukacionih simulatora u nastavi iz veštačke inteligencije"**, *57. konferencija "ETRAN 2013" - Zbornik radova*, Društvo za elektroniku, telekomunikacije, računarstvo, automatiku i nuklearnu tehniku, Zlatibor, Srbija, Jun, 2013 **[M63]**, ISBN: 978-86-80509-68-6

A. Изјава о ауторству

Име и презиме аутора: **Дражен Драшковић**
Број индекса: **2011/5001**

Изјављујем

да је докторска дисертација под насловом

Софтверски систем за учење и примену алгоритама вештачке интелигенције

- резултат сопственог истраживачког рада;
- да дисертација у целини ни у деловима није била предложена за стицање друге дипломе према студијским програмима других високошколских установа;
- да су резултати коректно наведени и
- да нисам кршио/ла ауторска права и користио/ла интелектуалну својину других лица.

Потпис аутора

У Београду, 28.06.2018. год. _____

B. Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора: **Дражен Драшковић**

Број индекса: **2011/5001**

Студијски програм: **Електротехника и рачунарство**

Наслов рада: **Софтверски систем за учење и примену алгоритама вештачке интелигенције**

Ментори: **др Бошко Николић, редовни проф. и др Милош Цветановић, ванредни проф.**

Изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла ради похрањена у **Дигиталном репозиторијуму Универзитета у Београду.** Дозвољавам да се објаве моји лични подаци везани за добијање академског назива доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

Потпис аутора

У Београду, 28.06.2018. год. _____

C. ИЗЈАВА О КОРИШЋЕЊУ

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

Софтверски систем за учење и примену алгоритама вештачке интелигенције

која је моје ауторско дело.

Дисертацију са свим прилозима предао/ла сам у електронском формату погодном за трајно архивирање.

Моју докторску дисертацију похрањену у Дигиталном репозиторијуму Универзитета у Београду и доступну у отвореном приступу могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио/ла.

1. Ауторство (CC BY)

2. Ауторство – некомерцијално (CC BY-NC)

3. Ауторство – некомерцијално – без прерада (CC BY-NC-ND)

4. Ауторство – некомерцијално – делити под истим условима (CC BY-NC-SA)

5. Ауторство – без прерада (CC BY-ND)

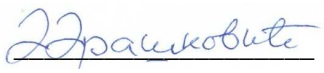
6. Ауторство – делити под истим условима (CC BY-SA)

(Молимо да заокружите само једну од шест понуђених лиценци.

Кратак опис лиценци је саставни део ове изјаве).

Потпис аутора

У Београду, 28.06.2018. год.



1. **Ауторство**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце, чак и у комерцијалне сврхе. Ово је најслободнија од свих лиценци.

2. **Ауторство – некомерцијално**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела.

3. **Ауторство – некомерцијално – без прерада**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела. У односу на све остале лиценце, овом лиценцом се ограничава највећи обим права коришћења дела.

4. **Ауторство – некомерцијално – делити под истим условима**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца не дозвољава комерцијалну употребу дела и прерада.

5. **Ауторство – без прерада**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца дозвољава комерцијалну употребу дела.

6. **Ауторство – делити под истим условима**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца дозвољава комерцијалну употребу дела и прерада. Слична је софтверским лиценцама, односно лиценцама отвореног кода.