УНИВЕРЗИТЕТ У БЕОГРАДУ

ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ

Новак С. Заграђанин

ПЛАНИРАЊЕ ПУТАЊЕ РОБОТА БАЗИРАНО НА D* Lite АЛГОРИТМУ И АУТОНОМНОЈ ПРЕТРАЗИ ОКРУЖЕЊА

докторска дисертација

Београд, 2022

UNIVERSITY OF BELGRADE

SCHOOL OF ELECTRICAL ENGINEERING

Novak S. Zagrađanin

ROBOT PATH PLANNING BASED ON D* Lite ALGORITHM AND AUTONOMOUS EXPLORATION OF ENVIRONMENT

Doctoral Dissertation

Belgrade, 2022

Ментор:

др Коста Јовановић, ванредни професор
Универзитет у Београду - Електротехнички факултет

Чланови комисије:

др Коста Јовановић, ванредни професор
Универзитет у Београду - Електротехнички факултет (ментор)

др Жељко Ђуровић, редовни професор
Универзитет у Београду - Електротехнички факултет

др Бојан Павковић, научни сарадник
Војнотехнички институт

др Драган Памучар, ванредни професор
Универзитет одбране - Војна академија

др Милан Бебић, ванредни професор
Универзитет у Београду – Електротехнички факултет

Датум одбране: _____

ЗАХВАЛНИЦА

Пре свега бих желео да изразим искрену захвалност ментору, ванредном професору др Кости Јовановићу, на инспирацији и мотивацији за даљи рад у критичним фазама докторских студија, као и на константној подршци током истраживања и рада на дисертацији.

Срдачно захваљујем и професору др Жељку Ђуровићу на непроцењивим саветима и подршци током студија.

На конструктивним дискусијама и саветима захваљујем потпуковнику ванредном професору др Драгану Памучару са Војне академије, као и колегама са Електротехничког факултета мс Николи Кнежевићу и мс Завиши Гордићу.

Велику захвалност за разумевање и подршку дугујем и колегама са посла, као и професору др Стевици Граовцу са којим сам сарађивао у почетним фазама истраживања.

Захваљујем и својим родитељима, Србиславу и Миљци, као и сестри Дубравци.

Коначно, неизмерну захвалност за безусловну подршку, разумевање и стрпљење, без којих дисертације не би ни било, дугујем супрузи Гордани и ћерки Анастасији, која је рођена и расла током трајања докторских студија. Ћерки Анастасији уједно и посвећујем ову дисертацију.

Наслов тезе: Планирање путање робота базирано на *D* Lite* алгоритму и аутономној претрази окружења

Сажетак: Предмет истраживања дисертације су *on-line frontier-based* стратегије за претрагу окружења, код којих је примењен *D* Lite* алгоритам за планирање путање робота и које за избор следеће позиције користе методе вишекритеријумског одлучивања (*ВКО*). Као критеријуми за избор следеће позиције из скупа кандидата користе се: дужина путање до кандидата, информативни потенцијал кандидата и удаљеност кандидата од базне станице.

За меру ефикасности претраге примењен је просечан пређени пут робота у односу на више стартних позиција за потребе реализације претраге 90% окружења.

Симулацијама у Матлабу, доказано је да у дефинисаним условима *TOPSIS* обезбеђује ефикаснију претрагу комплексних окружења у односу на друге две анализиране методе *ВКО (SAW* и *COPRAS)*, као и у односу на 4 класичне стратегије из литературе. Ово се објашњава тиме да *TOPSIS* при одабиру алтернативе максимизује Еуклидову удаљеност од хипотетички најлошијег решења, чиме се минимизује ризик у одлучивању који је изражен код комплексних окружења. Кључни резултати су верификовани у *Gazebo 3D* симулатору, користећи *Robot Operating System*.

У дисертацији се, такође, предлажу 2 приступа за унапређење планирања путање робота у мисијама трагања и спасавања.

Први приступ је примена *D* Lite* алгоритма у комбинацији са фази логиком. Фази логика се користи за дефинисање тежина ћелија мапе намењене за планирање путања робота, како оне не би имале бинарне вредности. Тиме је постигнуто да претрага окружења покрива објекте према приоритету, као и да роботи за мапирање при кретању не ометају друге роботе.

Други приступ је примена *D* Lite* алгоритма у комбинацији са *on-line* учењем. Овим је скраћен пређени пут робота за мапирање и минимизована су његова ризична кретања, јер се маневар у циљу избегавања препрека изводи пре потенцијалног сусрета.

Кључне речи: планирање путање робота, *D* Lite* алгоритам, *frontier-based* претрага окружења, вишекритеријумско одлучивање, *TOPSIS*, фази систем, *on-line* учење.

Научна област: Електротехника и рачунарство.

Ужа научна област: Аутономни мобилни роботи.

Doctorial Dissertation Title: Robot path planning based on D* Lite algorithm and autonomous exploration of environment.

Abstract: The subject of the research in the dissertation are *on-line frontier-based* strategies for exploration of environment, which have applied *D* Lite* algorithm for robot path planning and which use the multi-criteria decion making (MCDM) methods for selecting the next position. The criteria used in the process of decion making are: the length of a collision free path, the expected information gain and the distance from the base station.

To measure the efficiency of the exploration, the average travelled robot path was applied in relation to several starting positions for the needs of the 90% environment explored.

By simulations in Matlab, it was proved that in defined conditions the *TOPSIS* provides more efficient exploration of complex environment in relation to other two analzyed MCDM methods *(SAW* and *COPRAS)*, as well as in relation to 4 classical strategies from the literature. This is explained in a way that, in selecting the alternative, the *TOPSIS* maximizes Euclidean distance from the hypothetically worst solution, by which a risk in decision making, emphasized in complex environment, is minimized. Key results were verified in the *Gazebo 3D* simulator, using the *Robot Operating System*.

The dissertation, also, proposes 2 approaches for improving the robot path planning in search and rescue missions.

The first approach is the application of *D* Lite* algorithm in combination with fuzzy logics. The fuzzy logics is used to define the map cell cost used for robot path planning, so they do not have the binary values. This has made the exploration cover the objects according to the priority, as well as that robots for mapping do not interfere with other robots during movement.

The second approach is the the application of *D* Lite* algorithm in combination with *on-line* learning. This has made the mapping robot path shortened and its risky movement has been minimized, since the maneuvre with the aim of avoiding the obstacles is performed before a potential encounter.

Key words: robot path planning, *D* Lite* algorithm, *frontier-based* exploration of environment, multi-criteria decion making, *TOPSIS*, fuzzy system, *on-line* learning.

Scientific area: Electrical Engineering and Computer Science.

Specific scientific area: Autonomous mobile robots.

САДРЖАЈ

Списак слика:

Списак табела:

1. УВОД

Након описа предмета, циља и мотива истраживања, у оквиру Увода изложен је детаљан опис садржаја докторске дисертације. На крају поглавља наводе се остварени научни доприноси и публиковани радови у оквиру истраживања.

1.1. Предмет, циљ и мотиви истраживања

Аутономни мобилни роботи се све више користе за обављање различитих задатака, како у уобичајеним, свакодневним ситуацијама, тако и у ситуацијама и окружењима где из економских или безбедносних разлога није могуће ангажовати људе. Ти задаци су често везани за неку потребу претраге окружења, било да се ради о формирању његове мапе [1,2] или о откривању неких других карактеристика окружења које су од интереса за задату мисију, као што је то у случају трагања и спасавања [3], планетарних мисија [4], визуелне инспекције [5], рударства [6], и сл. Поред тога, у пракси су честе ситуације комбинованог извршења више врста задатака током претраге окружења, на пример када се паралелно или непосредно пре активности основне мисије (рецимо трагања и спасавања) мора формирати мапа окружења како би се мисија могла реализовати [7], итд.

У зависности од расположивих информација о окружењу у ком се планира ангажовање робота, за претрагу окружења примењује се *on-line* или *off-line* приступ [8,9]. У случају *on-line* претраге, која је са аспекта реализације много изазовнија (примењује се када је окружење потпуно непознато или када нема довољно информација за примену *off-line* алгоритама), робот након прикупљања података са актуелне позиције бира следећу позицију у складу са неком (претходно дефинисаном) стратегијом, премешта се на њу, извршава опсервацију окружења са нове позиције и понавља овај процес све док не испуни задатак или заврши мисију [10].

Током последњих деценија у литератури су предложене различите стратегије за претрагу окружења. Већина њих се заснива на концепту избора следеће позиције робота из скупа позиција-кандидата које леже на граници између претражених и непретражених делова окружења и то су тзв. *„frontier-based“* стратегије. Концепт *frontier-based* претраге окружења први пут је званично презентован у раду аутора *Yamauchi* 1997. године [11].

Најједноставнија из групе *frontier-based* стратегија је тзв. „*Dist_Min“* стратегија, која за следећу позицију робота бира ону која је најближа његовој тренутној позицији [12]. Са друге стране, стратегија која се најчешће предлаже у литератури као довољно добар избор је тзв. *GBL* стратегија (названа по ауторима) [13], која евалуацију сваке позиције-кандидата врши узимајући у обзир њену удаљеност од тренутне позиције робота и очекивану количину информација о окружењу коју робот може прикупити на тој позицији. Поред ова два критеријума, укључивање вероватноће успостављања комуникације између робота и базне станице у процес евалуације позиција-кандидата, имајући у виду значај тог критеријума, први пут је предложено у [14], код тзв. *AOJRF* стратегије (названа по тиму *Amsterdam Oxford Joint Rescue Forces* који је ову стратегију са успехом користио на такмичењу *RoboCup Rescue Virtual Robots Competition* 2009. године). Наведене стратегије за претрагу окружења, уз додатак још неких које ће бити објашњене у даљем тексту, могу се сматрати класичним и коришћене су као референтне (једна или више њих) у значајном делу релевантне литературе која проучава предметну област [8,10,12,15-17].

Предмет истраживања у оквиру ове дисертације су *on-line frontier-based* стратегије за претрагу окружења, код којих је примењен *D* Lite* алгоритам за планирање путање робота и које за избор следеће позиције осматрања и прикупљања података (из скупа могућих кандидата) користе методе вишекритеријумског одлучивања, које (према сазнању аутора) до сада нису коришћене за ту намену. Као критеријуми за избор следеће позиције робота из скупа кандидата користе се: дужина путање до кандидата, информативни потенцијал кандидата и удаљеност кандидата од базне станице.

На бази проучене литературе, за меру ефикасности претраге у дисертацији је примењен просечан укупан пређени пут робота у односу на више стартних позиција за потребе реализације претраге 90% окружења. Имајући то у виду, два изабрана критеријума - „дужина путање до позиције-кандидата“ и „информативни потенцијал позиције-кандидата“ непосредно утичу на ефикасност претраге, први обрнуто пропорционално јер кумулативно увећава укупан пређени пут робота, а други директно пропорционално јер кумулативно увећава претражени део окружења. Трећи изабрани критеријум - „удаљеност позиције-кандидата од базне станице“ не утиче непосредно на ефикасност претраге, али је битан са аспекта одржавања комуникације са базном станицом, што посебно долази до изражаја у мисијама трагања и спасавања када за што краће време треба проследити прикупљене податке са терена.

У оквиру наведеног предмета истраживања, у дисертацији се најпре проучавају алгоритми за планирање путање робота из фамилије *A** алгоритма [18]. Ови алгоритми функционишу на бази обраде графа којим се моделира мапа окружења у којем се креће робот. Често се примењују у пракси за планирање путање робота, па тако и у стратегијама за претрагу окружења [19-23]. Типичан представник поменуте фамилије алгоритама је *D* Lite* [24]. *D* Lite* представља поједностављену верзију (у смислу алгоритамске процедуре) стандардног *D** алгоритма [25]. *D** алгоритам је ефикаснији од *A** алгоритма [26], у смислу да брже долази до решења, због чега је нашао широку примену у пракси, укључујући беспосадне верзије возила *HMMWV*, беспосадне платформе које су учествовале на *Defense Advanced Research Projects Agency (DARPA)* такмичењима, *NASA* ровере на Марсу, итд. [26-30]. *D* Lite* захтева око 30% мање линија кода од *D**, при чему је ефикасан као и *D**, а у појединим случајевима је и ефикаснији [26], па је стога *D* Lite* данас један од најчешће примењиваних алгоритама за планирање путање робота. Ради јаснијег увида у специфичности *D* Lite* алгоритма, он се у дисертацији пореди са алгоритмима *Anytime Replanning A** (*ARA**) [31] и *Anytime Dynamic A** (*AD**) [32].

У дисертацији се, даље, истражују и тестирају предности и недостаци коришћења метода вишекритеријумског одлучивања (*ВКО*) *SAW* [33], *COPRAS* [34] и *TOPSIS* [35] за избор следеће позиције робота у стратегијама за претрагу окружења. Мотив за наведено је чињеница да се у последњој деценији, поред претходно поменутих класичних стратегија, све више користе методе *ВКО* у мисијама претраге окружења, имајући пре свега у виду да нису ограничене у броју критеријума које могу разматрати при евалуацији позиција-кандидата. Широким и флексибилним приступом у одабиру критеријума и њихових тежина *ВКО* обезбеђује ефикасно управљање претрагом у зависности од услова у којима се одвија и њених циљева [8,12]. Од наведених метода *ВКО*, у контексту предмета дисертације, посебно је карактеристична *TOPSIS*. Основна специфичност *TOPSIS* методе је да при одлучивању настоји да максимизује критеријуме бенефитног типа и минимизује критеријуме трошковног типа, али тако да истовремено тежи да изабрана алтерантива има што мању Еуклидову удаљеност од идеалног решења и што већу Еуклидову удаљеност од анти-идеалног решења. Идеално решење је хипотетичко решење и код њега све вредности критеријума одговарају најпожељнијим вредностима у односу на све алтернативе. Анти-идеално решење је, такође, хипотетичко и код њега све вредности критеријума одговарају

најнепожељнијим вредностима у односу на све алтернативе. Максималним удаљавањем од хипотетички најлошијег решења минимизује се ризик у одлучивању [36], што посебно долази до изражаја ако је већина критеријума велике важности. Другим речима, смањује се ризик избора неке алтернативе на рачун тога што већина критеријума има релативно повољне вредности, али чија нпр. вредност једног критеријума има у значајнијој мери лошију вредност. Код других анализираних метода *ВКО,* као и код класичних стратегија, може се десити да се на рачун повољних вредности осталих критеријума изабере алтернатива која има у значајнијој мери лошу вредност једног критеријума.

Имајући изнето у виду, а у складу са предметом истраживања, дисертација се додатно фокусира на истраживање аутономне претраге комплексних окружења (окружења са великим бројем препрека), чија је основна карактеристика да се позиције-кандидати (алтернативе при одлучивању) значајно разликују, како у погледу критеријума „информативни потенцијал позиције-кандидата“, тако и у погледу критеријума „дужина путање до позиције-кандидата“ и где је последично изражен ризик од лошег избора следеће позиције робота управо са аспекта ефикасности претраге. Код окружења која нису комплексна разлика алтернатива у погледу вредности поменутих критеријума је драстично мања. Базирајући се на поменутој чињеници да *TOPSIS* минимизује ризик у одлучивању, у дисертацији се полази од хипотезе да ће ова метода, у случају претраге комплексних окружења, уз примену изабраних критеријума, у просеку изабрати бољу алтернативу у односу на друге две анализиране методе *ВКО - SAW* и *COPRAS*, као и у односу на остале класичне стратегије за претрагу окружења анализиране у овој дисертацији и тиме обезбедити да се претрага реализује ефикасније тј. за краћи укупан пређени пут робота.

У циљу тестирања постављене хипотезе реализоване су опсежне симулације у Матлабу. За верификацију кључних резултата истраживања коришћен је *Gazebo* симулатор (специјализовано окружење за *3D* симулацију робота), као веродостојна замена за тестирање анализираних стратегија за претрагу окружења на реалним роботима. За те потребе примењен је виртуелни модел *Turtlebot 3* робота. Уз *Gazebo* су коришћени пакети који раде под *Robot Operating System* (*ROS*).

У дисертацији се, такође, истражују и предлажу два приступа за унапређење планирања путање робота применом *D* Lite* алгоритма у мисијама претраге окружења за потребе трагања и спасавања у урбаном окружењу. Овај део истраживања је такође мотивисан проученом литературом, што ће бити објашњено у наставку.

Cost maps технике или технике за дефинисање тежина ћелија мапе су погодне за примену у комбинацији са алгоритмима који планирање путање робота врше на бази обраде графа, како би се унапредио процес планирања. Ове технике пружају могућност реалистичног осликавања сложености терена са аспекта проходности [37,38], док посебан изазов представља дефинисање тежина ћелија у мапи која се користи за планирање путање робота у окружењу где су присутни људи [39-41]. Поред *cost maps* приступа, последњих година један од главних праваца истраживања у роботици је примена различитих техника учења у комбинацији са алгоритмима за планирање путање робота. Циљ је да се на овај начин у мери која је могућа примени стечено искуство како би се унапредио процес планирања. Модели учења могу бити засновани на формирању базе података где се чувају сегменти научених локалних путања у зависности од уобичајених геометријског облика препрека [42-44], затим коришћењем тзв. „графова базираних на искуству“, којим се увек изнова на исти начин моделира део окружења који има фиксну структуру [45-47], и сл.

Уважавајући претходно наведено, у дисертацији се предлаже примена фази интегралног система за дефинисање тежина ћелија мапе коју користи *D* Lite* алгоритам за планирање путања робота за мапирање, ради управљања претрагом окружења за потребе трагања и спасавања у урбаном окружењу. Фази интегрални систем се састоји од два фази експертска система и фази свича. Оба фази система имају исте улазе и излазе, с тим што један систем има мању осетљивост промене тежине ћелија мапе, а други већу осетљивост, што се постиже подешавањем положаја центроида одговарајућих функција припадности. Фази свич има један улаз и два излаза и он одређује који фази систем у ком тренутку има већи утицај на формирање коначне тежине ћелије. Намена фази интегралног система је да обезбеди да *D* Lite* алгоритам генерише путање најмање цене (у зависности од три параметра) и то тако да се претрага окружења реализује на такав начин и тим редом да „покрива“ објекте према приоритету, од највећег ка најмањем, као и да роботи за мапирање при кретању у што мањој мери ометају друге роботе и људе, који због природе својих задатака обично морају прићи ближе објектима. Са друге стране, у ситуацији када није примењен фази интегрални систем тежине ћелија имају бинарне вредности, тако да алгоритам увек генерише путање најмање дужине, а не путање најмање цене, чиме се губи флексибилност у управљању мисијом.

У другом случају ради се о мисијама претраге окружења са споропроменљивим препрекама детектованим у претраженом делу окружења. При претрази окружења карактеристично је високо понављајуће кретање робота кроз исти простор, што је искоришћено за примену *on-line* учења базираног на „*M-out-of-N*“ детектору. Намена *on-line* учења је да се евентуалне споропроменљиве препреке детектоване у претраженом делу окружења (други роботи, машине или група људи који на некој фиксној позицији одређено време раде своје задатке и сл.), интегришу односно бришу из мапе која се користи за планирање путања робота на начин који унапређује процес планирања, што се регулише подешавањем параметара „*M-out-of-N*“ детектора тј. одговарајућег статистичког теста. Ове ситуације су такође карактеристичне за трагање и спасавање у урбаном окружењу, али нису ограничене само на ту врсту мисија претраге окружења. Циљ је да се на овај начин оптимизује путања робота за мапирање (благовременим извођењем маневра за избегавање препреке), односно скрати његов укупан пређени пут и минимизују његова ризична кретања у простору који дели са другим роботима, машинама или људима.

1.2. Опис садржаја дисертације

Поред увода и закључка, докторска дисертација садржи још три поглавља.

У другом поглављу обрађују се алгоритми за планирање путање робота *ARA*, D* Lite* и *AD*,* базирани *A** алгоритму. Најпре се образлаже концепт *A** алгоритма, затим детаљно описује принцип функционисања *anytime* алгоритама (алгоритама са брзим одзивом), односно типичног представника те класе планера – *ARA** алгоритма. У наставку се описују динамички (инкрементални) алгоритми *D* Lite* и *AD**, чија основна карактеристика је ефикасно процесуирање детектованих промена у окружењу. На крају поглавља даје се детаљан пример функционисања *D* Lite* алгоритма и врши поређење функционисања алгоритама *ARA*, D* Lite* и *AD** у неколико сценарија.

Треће поглавље обрађује аутономну претрагу окружења. После дефиниције претраге окружења и описа различитих приступа за њену реализацију, прелази се на детаљну обраду концепта *frontier-based* претраге окружења, код којег се следећа позиција робота бира од позиција-кандидата на граници између претраженог и непретраженог дела окружења. Поглавље се завршава навођењем кључних *frontier-based* стратегија за

претрагу окружења из литературе и образлагањем „техника“ које те стратегије користе за избор следеће позиције робота.

Четврто поглавље посвећено је унапређењу претраге окружења применом вишекритеријумског одлучивања и *D* Lite* алгоритма, као и унапређењу планирања путање робота у мисијама трагања и спасавања и има три целине. На почетку прве целине уводе се стратегије за претрагу окружења које за избор следеће позиције робота користе методе вишекритеријумског одлучивања *SAW, COPRAS* и *TOPSIS*. После описа модела окружења за претрагу и критеријума за избор следеће позиције робота, врши се опсежно тестирање путем симулација у Матлабу различитих сценарија претраге окружења применом поменутих метода *BKO*, као и класичних стратегија за претрагу окружења из литературе анализираних у дисертацији (*Dist_Min, GBL, AOJRF, WS*). На крају ове целине презентују се резултати и врши дискусија у вези са истим. У другој целини у оквиру овог поглавља предложена су два приступа за унапређење планирања путање робота применом *D* Lite* алгоритма у мисијама претраге окружења за потребе трагања и спасавања у урбаном окружењу. Први приступ представља примену *D* Lite* алгоритма у комбинацији са фази логиком, а други примену *D* Lite* алгоритма у комбинацији са *on-line* учењем. Трећа целина бави се верификацијом кључних резултата истраживања коришћењем *Gazebo* симулатора (специјализованог окружења за *3D* симулацију робота) у комбинацији са пакетима који раде под *Robot Operating System* (*ROS*), као веродостојна замена за тестирање анализираних стратегија за претрагу окружења на реалним роботима. За те потребе примењен је виртуелни модел *Turtlebot 3* робота.

1.3. Научни доприноси и публиковани радови у оквиру истраживања

Резултати истраживања обухваћеног дисертацијом представљају оригиналне научне доприносе. С тим у вези, у наставку ће бити наведени публиковани радови везани за остварене доприносе, након чега ће се сваки допринос укратко образложити уз референцирање на наведене радове и одговарајућа поглавља у дисертацији.

Часописни радови:

(1) Novak Zagradjanin, Dragan Pamucar, Kosta Jovanovic, Nikola Knezevic and Bojan Pavkovic, „Autonomous exploration based on multi-criteria decision-making and using D* Lite algorithm“, Intelligent Automation & Soft Computing, Special Issue: Soft Computing Methods for Intelligent Automation Systems, pp. 1369-1386, 2021 (M23, IF=1.647).

(2) Novak Zagradjanin, Aleksandar Rodic, Dragan Pamucar and Bojan Pavkovic, „Cloud-based multi-robot path planning in complex and crowded environment using fuzzy logic and online learning“, Information Technology and Control, Vol. 50, No. 2, pp. 357-374, 2021 (M23, IF=1.228).

(3) Novak Zagradjanin, Dragan Pamucar and Kosta Jovanovic, „Cloud-based multi-robot path planning in complex and crowded environment with multi-criteria decision making using full consistency method“, Symmetry, Vol. 11, No. 10, pp. 1-15, 2019 (M22, IF=2.645).

Конференцијски радови:

(4) Novak Zagradjanin and Aleksandar Rodic, „Cloud-based distributed intelligence of robot team in complex accident situation“, 4th International Conference on Electrical, Electronic and Computing Engineering – IcETRAN 2017, Kladovo, Serbia, 2017.

(5) Novak Zagradjanin and Stevica Graovac, „Autonomous mobile robot path planning in complex and dynamic environments“, International Scientific Conference on Defensive Technologies - OTEH 2016, Belgrade, Serbia, 2016.

(6) Novak Zagradjanin and Stevica Graovac, „Analysis and comparison of a command guidance and the combination of programmed and homing guidance“, International Scientific Conference on Defensive Technologies - OTEH 2014, Belgrade, Serbia, 2014.

У оквиру докторске дисертације остварени су следећи научни доприноси:

- **Допринос 1.** (*Поглавље 4.1*)**:** Унапређена је претрага комплексних окружења применом вишекритеријумског одлучивања (*ВКО*) за избор следеће позиције робота и *D* Lite* алгоритма за планирање путање робота. Конкретно, доказано је да се применом *TOPSIS* методе *ВКО* за избор следеће позиције и *D* Lite* алгоритма за планирање путање обезбеђује претрага комплексних окружења за краћи укупан пређени пут робота у односу на методе *SAW* и *COPRAS*, као и у односу на остале анализиране класичне стратегије за претрагу окружења из литературе (*Dist_Min, GBL, AOJRF, WS*). За избор следеће позиције робота коришћени су критеријуми: „дужина путање од актуелне позиције робота до позиције-кандидата“, „информативни потенцијал позиције-кандидата“ и „удаљеност позиције-кандидата од базне станице“. Рад у коме је објављен овај научни допринос наведен је на позицији (1), док су у сегменту истраживања *D* Lite* алгоритма и његове примене за овај допринос везани и радови: (2), (3), (4) и (5).

- **Допринос 2.** (*Поглавље 4.2.1*)**:** Унапређен је процес планирања путање робота у мисијама претраге окружења за потребе трагања и спасавања у урбаном окружењу, применом *D* Lite* алгоритма у комбинацији са оригиналним фази интегралним системом. Фази интегрални систем се користи за дефинисање тежина ћелија мапе намењене за планирање путања робота, како оне не би имале бинарне вредности, тако да алгоритам генерише путање најмање цене у зависности од три параметра, а не путање најмање дужине. Тиме је постигнуто да претрага окружења покрива објекте према приоритету, као и да роботи за мапирање при кретању не ометају друге роботе. За овај научни допринос примарно је везан рад на позицији (2), а резултати истраживања везани за овај допринос објављени су и у радовима (3), (4) и (5).

- **Допринос 3.** (*Поглавље 4.2.2*)**:** Унапређен је процес планирања путање робота при претрази динамичких окружења, конкретно окружења са споропроменљивим препрекама детектованим у претраженом делу терена, применом *D* Lite* алгоритма у комбинацији са *on-line* учењем. Поменуте ситуације су карактеристичне за трагање и спасавање у урбаном окружењу, али нису ограничене само на ту врсту мисија претраге окружења. Намена *on-line* учења је да се детектоване препреке интегришу односно бришу из мапе која се користи за планирање путања робота на начин који унапређује процес планирања, што се постиже применом прилагођеног „*M-out-of-N*“ детектора тј. одговарајућег статистичког теста. Овим је скраћен пређени пут робота за мапирање и минимизована су његова ризична кретања, јер се маневар у циљу избегавања препрека изводи пре потенцијалног сусрета. Поменути научни допринос такође се примарно односи на рад наведен на позицији (2), а резултати дела везаних истраживања објављени су и у радовима (3), (4) и (5).

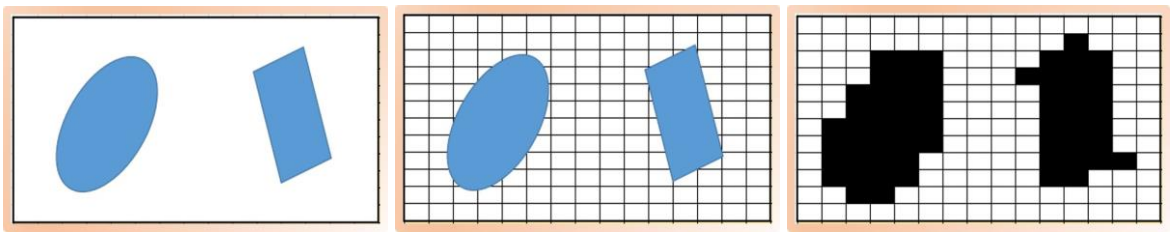
- **Допринос 4.** (*Поглавље 4.1.*)**:** Проширена је база стратегија за претрагу окружења које користе *ВКО* за избор следеће позиције робота имплементирањем стандардних *ВКО* метода које до сада нису коришћене у ту сврху - *SAW*, *COPRAS* и *TOPSIS*. Овај научни допринос такође је обухваћен радом на позицији (1), а резултати истраживања везани за овај допринос објављени су и у раду (3).

2. АЛГОРИТМИ ЗА ПЛАНИРАЊЕ ПУТАЊЕ АУТОНОМНИХ МОБИЛНИХ РОБОТА

Планирање путање робота је један од фундаменталних задатака у роботици и представља процес дефинисања путање, коришћењем неког алгоритма (планера), којом робот, пратећи је, може стићи од стартне до циљне позиције на безбедан начин (без физичког контакта са препрекама, без оштећења робота, итд.) [48].

За потребе примене алгоритама за планирање путање робота који функционишу на бази обраде графа, мапа окружења у којем се креће робот представља се у виду *2D* мреже ћелија униформне резолуције, где је свакој ћелији додељена тежина која према унапред усвојеној логици описује могућност или сложеност проласка робота кроз припадајући део терене [49]. Овако формирана *2D* мрежа може се даље представити у виду графа, тако што се свакој појединачној ћелији мреже додели један чвор лоциран у центру ћелије који је повезан са свих 8 суседних чворова формираних на исти начин. Уведимо и појам цена транзиције или цена преласка између два суседна чвора и нека она зависи од међусобне удаљености чворова и тежина ћелија које представљају [49]. На овај начин створени су предуслови за планирање путање робота применом алгоритама који функционишу на бази обраде графа.

Принцип униформне дискретизације мапе окружења са препрекама и њеног приказа у виду мреже ћелија бинарне класификације (ћелије могу бити слободне или блокиране), ради примене алгоритама за планирање путање робота који функционишу на бази обраде графа приказан је на *Слици 1*, а детаљнији опис се може пронаћи у [50].



а) Мапа окружења са препрекама

б) Дискретизација мапе окружења

ц) Приказ мапе окружења где се свака ћелија делимично или потпуно заузета препреком класификује као блокирана

Слика 1. Пример процеса униформне дискретизације мапе окружења са препрекама (означене плавом бојом) и њеног приказа у виду мреже ћелија бинарне класификације (белом бојом су означене слободне ћелије, а црном бојом блокиране ћелије).

У реалним сценаријима најчешће су ситуације у којима је у тренутку отпочињања кретања мапа окружења у које се робот упућује само делимично позната. У таквим ситуацијама као полазна претпоставка узима се да су сви делови окружења који су у почетном тренутку непознати уједно и слободни за пролаз, а робот у току кретања на основу информација прикупљених својим сензорима врши детаљно упознавање терена. Поред тога, могуће су и динамичке промене претходно познатог окружења у току самог кретања робота. Без обзира о којој конкретно ситуацији се ради, оне подразумевају измену

у графу на основу којег је алгоритам генерисао путању, те је стога неопходно извршити њено репланирање.

Постоји више врста алгоритама који планирање или прорачун путање робота врше на бази обраде графа. Један од најстаријих и свакако најпознатијих из ове класе алгоритама је *Dijkstra* алгоритам, који се и данас често примењује [51]. *A** алгоритам [18] за разлику од *Dijkstra* алгоритма спада у тзв. „хеуристичке“ алгоритме, о чему ће бити речи у даљем тексту, што му омогућава да фокусира обраду графа од стартног према циљном чвору, тако да до решења долази на ефикаснији начин. Алгоритми *D*, Incremental A*, ARA*, D* Lite, AD** и сл. представљају „надоградње“ *A** алгоритма и такође су нашли широку примену у мобилној роботици. Поред роботике, алгоритми за прорачун путање који функционишу на бази обраде графа користе се често и у другим областима примене вештачке интелигенције.

2.1. Уопштено о концепту *A** алгоритма

Међу најпознатијим алгоритмима за планирање путање који раде на бази обраде графа спадају алгоритми засновани на концепту *A** алгоритма. У наставку ће бити изложен принцип функционисања *A** алгоритма.

Претпоставимо да је са S означен скуп свих чворова у једном графу формираном према претходно наведеном поступку. Сваки чвор s ($s \in S$) представља једну ћелију мапе и он је повезан са чворовима који представљају суседне ћелије. За било који пар чворова s, $s' \in S$, дефинишимо тзв. цену преласка од чвора s до чвора s' коју ћемо означити са $c(s, s')$ и за коју важи $0 < c(s, s') \leq \infty$. У општем случају $c(s, s') \neq c(s', s)$. Цену путање од чвора s_0 до чвора s_k, где $s_0, s_k \in S$, одређујемо као збир сукцесивних прелаза између суседних чворова у скупу чворова $\{s_0, s_1, \ldots, s_{k-1}, s_k\} \in S$, тј. као $(c(s_0, s_1) + \ldots + c(s_{i-1}, s_i) + c(s_i, s_{i+1}) + \ldots + c(s_{k-1}, s_k))$, $1 \leq i \leq k$. Најмања цена путање од чвора s_0 до чвора s_k означава се са $c^*(s_0, s_k)$. За $s_0 = s_k$ важи $c^*(s_0, s_k) = 0$ [52].

Уведимо и појмове чвор „претходник“ и чвор „наследник“ произвољног чвора $s \in S$, на следећи начин. У циљу израчунавања путање најмање цене између стартног и циљног чвора, *A** алгоритам током обраде графа формира стабло потенцијалних путања идући од стартног чвора док не дође до циљног чвора. Ако посматрамо конкретну путању у стаблу путања којој припада чвор s у било ком тренутку обраде графа, чвор „претходник“ чвора s је чвор који непосредно физички претходи чвору s, док чвор „наследник“ чвора s представља чвор који непосредно физички следи после чвора s и то идући искључиво у смеру „од стартног чвора“. У стаблу путања чвор s може имати више чворова претходника и више чворова наследника. Скуп чворова претходника чвора s означавамо са $Pred(s)$, док скуп чворова наследника чвора s означавамо са $Succ(s)$.

*A** алгоритам је намењен за прорачун путање од чвора s_{start} до чвора s_{goal} чија цена је минимална тј. једнака $c^*(s_{start}, s_{goal})$. *A** алгоритам у току рада формира и одржава (ажурира) вредност четири функције које описују чвор s, и то [32,50]:

- $g(s)$ - цена путање чвора s, представља актуелну (пронађену до тренутка посматрања) минималну вредност цене путање од чвора s_{start} до чвора s;
- $h(s)$ - хеуристичка функција, представља процену минималне цене путање од чвора s до чвора s_{goal}; хеуристичка функција треба да задовољи услов конзистентности: $h(s) \leq c(s, s') + h(s')$ за свако $s \in S$ и $s \neq s_{goal}$, где $s' \in Succ(s)$ и $h(s) = 0$ за $s = s_{goal}$, где $c(s, s')$ означава цену путање од s до s' и има позитивну вредност; конзистентност хеуристичке функције гарантује да $h(s)$ није никада веће од стварне цене путање од s до s_{goal} [31,52-54];

- $f(s) = g(s) + h(s)$ – функција „кључа“ или само „кључ“, представља актуелну минималну цену путање од чвора s_{start} до чвора s_{goal} која иде преко чвора s [24,31,32];
- „показивач“ чвора s - указује на један од чворова из скупа $Pred(s)$ преко којег је изведено $g(s)$; показивач чвора s се као и $g(s)$ може мењати (ажурирати) више пута у току рада алгоритма; показивачи служе да се након завршетка прорачуна путање њиховим сукцесивним повезивањем од s_{goal} до s_{start} формира израчуната путања коју треба да следи робот током кретања.

Поред горе наведених функција, *A** алгоритам у току рада формира и одржава (ажурира) скуп чворова *OPEN*. Да бисмо објаснили сврху формирања скупа *OPEN* уведимо појам инконзистенције који је битан за функционисање *A** алгоритма, али и за његово проширење на друге алгоритме који користе ову особину чворова у процесу прорачуна путање. Чвор s код *A** алгоритма постаје инконзистентан ако дође до смањења актуелне вредности цене путање $g(s)$ [31]. Претпоставимо да је чвор s „најбољи“ претходник чвора s' тј. да важи $g(s') = g(s) + c(s, s')$. Ако се $g(s)$ смањи биће $g(s') > g(s) + c(s, s')$. Другим речима, смањење вредности $g(s)$ доводи до транзиције инконзистенције између чвора s и неких његових наследника. Скуп *OPEN* садржи све инконзистентне чворове детектоване до посматраног тренутка обраде графа, који су кандидати за даље процесуирање у смислу пропагације инконзистенције. Псеудокод основне верзије *A** алгоритма приказан је на *Слици 2* [55].

ComputeShortestPath()
01. while ($\text{argmin}_{s \in OPEN}(g(s) + h(s, s_{goal})) \neq s_{goal}$)
02. remove state s from the front of *OPEN*;
03. for all $s' \in Succ(s)$
04. if $(g(s') > g(s) + c(s, s'))$
05. $g(s') = g(s) + c(s, s')$;
06. insert s' into *OPEN* with value $(g(s') + h(s', s_{goal}))$;

Main()
07. for all $s \in S$
08. $g(s) = \infty$;
09. $g(s_{start}) = 0$;
10. $OPEN = \emptyset$;
11. insert s_{start} into *OPEN* with value $(g(s_{start}) + h(s_{start}, s_{goal}))$;
12. ComputeShortestPath();

Слика 2. A алгоритам [55].*

При иницијализацији A* алгоритма (линије 7.-11. псеудокода) дефинише се да је $g(s_{start})=0$, док се за почетну вредност цене путање свих осталих чворова усваја ∞. Такође се формира скуп *OPEN* у који се инсертује чвор s_{start}.

Генерални принцип формирања цене путање објашњен је у претходном делу текста. Пошто $g(s)$ представља тренутно важећу минималну цену путање од стартног чвора до чвора s, а $h(s)$ процену минималне цене пута од чвора s до циљног чвора, то

функција $f(s) = g(s) + h(s)$ представља процену минималне цене пута путање од стартног до циљног чвора који иде преко чвора s.

*A** алгоритам у сваком кораку прорачуна путање брише из скупа *OPEN* чвор s са најмањом вредношћу кључа (линија 2.) и проверава и ако за то постоје услови ажурира (смањује) цену пута сваког од његових суседа, потенцијалних чворова наследника (линије 3.-5.). Чвор s' сусед чвора s постаје његов наследник ако се током ове операције испостави да у том тренутку израчуната најкраћа путања од s_{start} до s' иде преко s. При овом процесу неопходно је ажурирати и одговарајуће показиваче. Сваког суседа чвора s чија цена пута је том приликом смањена алгоритам проглашава инконзистентним и категорише га као члана скупа *OPEN* уз истовремено ажурирање његовог кључа (линија 6.). Инконзистенција се на овај начин шири, чиме се обезбеђује да се чворовима који се обрађују увек додељује најмања цена путање пронађена до тог тренутка. Другим речима, алгоритам тако шири стабло путања најмањих цена од стартног чвора до обрађених чворова све док не дође до циљног чвора. Зато се за чворове избрисане из скупа *OPEN* и процесуиране на овај начин (линије 2.-6.) каже да су проширени (*expanded*), а сам поступак зове се ширење чвора [31]. У току рада алгоритма један исти чвор може бити инсертован у скуп *OPEN* и проширен више пута. Број проширених чворова је стандардни поступак за анализу ефикасности алгоритама за планирање путање у роботици базираних на обради графова [32].

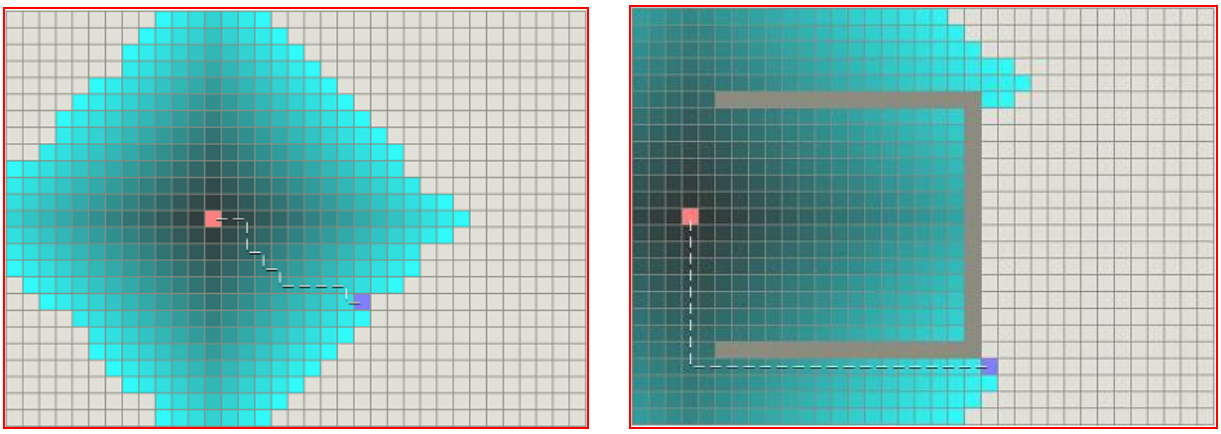
Приоритизација чворова у скупу *OPEN* при процесу ширења врши се на основу најмање вредности функције „кључа“ $f(s)$, како би се у сваком кораку као наредни ширио чвор за који постоји највећи изглед да припада најкраћој путањи од стартног до циљног чвора. Процес ажурирања скупа *OPEN* и ширења чворова из тог скупа траје све док се не испуни услов за терминацију рада алгоритма (линија 1.). Описани процес гарантује добијање путање од s_{start} до s_{goal} са најмањом ценом [52,55]. Када алгоритам заврши рад сукцесивним повезивањем показивача од s_{goal} до s_{start} формира се израчуната путања.

Осврнимо се на крају овог поглавља на улогу хеуристичке функције у планирању путање применом *A** алгоритма. Њена основна намена је да у току рада алгоритма обезбеди додатне информације (процену удаљености чвора који се испитује од циљног чвора) које треба да унапреде процес прорачуна путање у смислу да се скрати време обраде графа. Хеуристичка функција фокусира обраду графа у смислу да се „стабло путања“ усмерава према циљном чвору. Из наведених разлога се за алгоритме који користе хеуристичку функцију каже да припадају класи алгоритама који врше тзв. „информисану“ обраду графа, док се за алгоритме који не користе хеуристичку функцију каже да припадају класи алгоритама који врше тзв. „неинформисану“ обраду графа (типичан представник ове класе алгоритама је *Dijkstra* алгоритам). Обрада графа применом хеуристичке функције (која задовољава услов конзистентности) гарантује добијање оптималне путање тј. путање са најмањом ценом као и у случају неинформисане обраде, при чему алгоритам испитује много мање чворова него у случају неинформисане обраде [31,32,52]. Хеуристичка функција се може пондерисати тежинским фактором $\varepsilon>1$ како би се додатно убрзала обрада графа. Коришћење хеуристичке функције пондерисане тежинским фактором $\varepsilon>1$ убрзава обраду графа, али даје ε-субоптимално решење [31,32,52]. У том случају *A** алгоритам гарантује добијање субоптималне путање чија цена није више од ε пута већа од цене оптималне путање (путање са најмањом ценом) [31,32,52]. Што је веће ε већи је удео хеуристичке функције у вредности кључа у односу на $g(s)$, те се на тај начин „стабло путања“ више фокусира према циљном чвору.

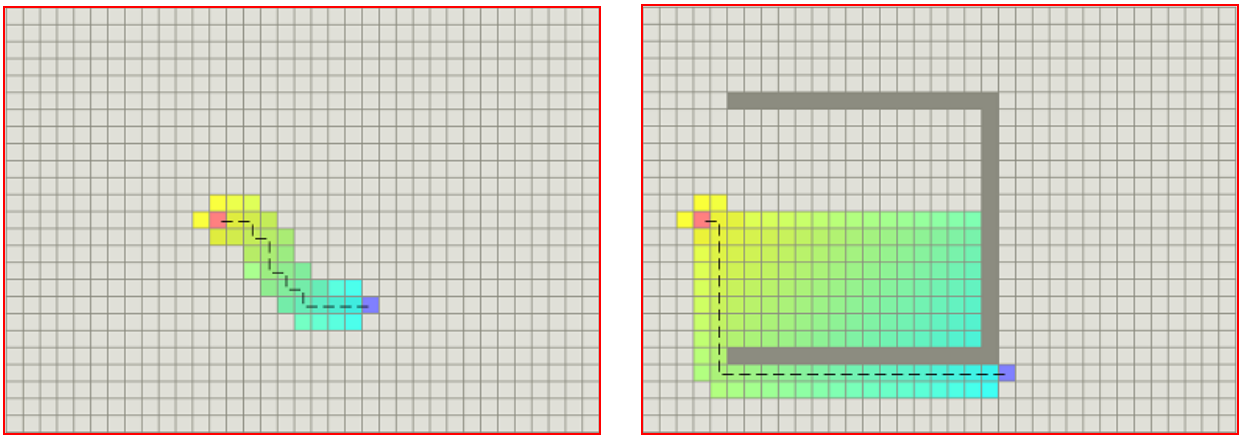
За хеуристичку функцију тј. процену минималне цене путање чвора s од циљног чвора (s_{goal}) може се узети Еуклидова удаљеност та два чвора, затим већа вредност од

следеће две - удаљеност чвора s од s_{goal} по хоризонталној (x) оси и по вертикалној (y) оси [54], итд. У сваком случају, хеуристичка функција због правилности обраде графа и гаранције добијања оптималног решења, мора да задовољи услов конзистентности наведен у опису функционисања *A** алгоритма.

Ефекти увођења хеуристичке функције на процес планирање путање илустративно су приказани на *Слици 3* [56]. Стартне позиције су приказане розе бојом, а циљне позиције тамно плавом бојом. Свака ћелија која није сиве боје је процесуирана у току прорачуна путање. Код *Dijkstra* алгоритма, што је ћелија светлије нијансе плаве боје то значи да је све удаљенија од стартне позиције, визуелизујући на тај начин ширење фронта потенцијалних путања током прорачуна путање док се не дође до циљне позиције. *A** алгоритам у току прорачуна путање узима у обзир удаљеност ћелије и од стартне и од циљне позиције, што је на слици визуелизовано коришћењем две боје. Већи интензитет жуте боје означава већу удаљеност ћелије од циљне позиције (тј. већу вредност хеуристичке функције), а већи интензитет плаве боје означава већу удаљеност ћелије од стартне позиције. *A** у току прорачуна путање идући од стартне ћелије према циљној „трага“ за ћелијама чији је збир ове две удаљености минималан, што је на слици илустративно приказано постепеним смањењем интензитета жуте и јачањем интензитета плаве боје. Као резултат добије се да је број процесуираних ћелија при прорачуну путање значајно мањи код примене *A** алгоритма, него у случају примене *Dijkstra* алгоритма.



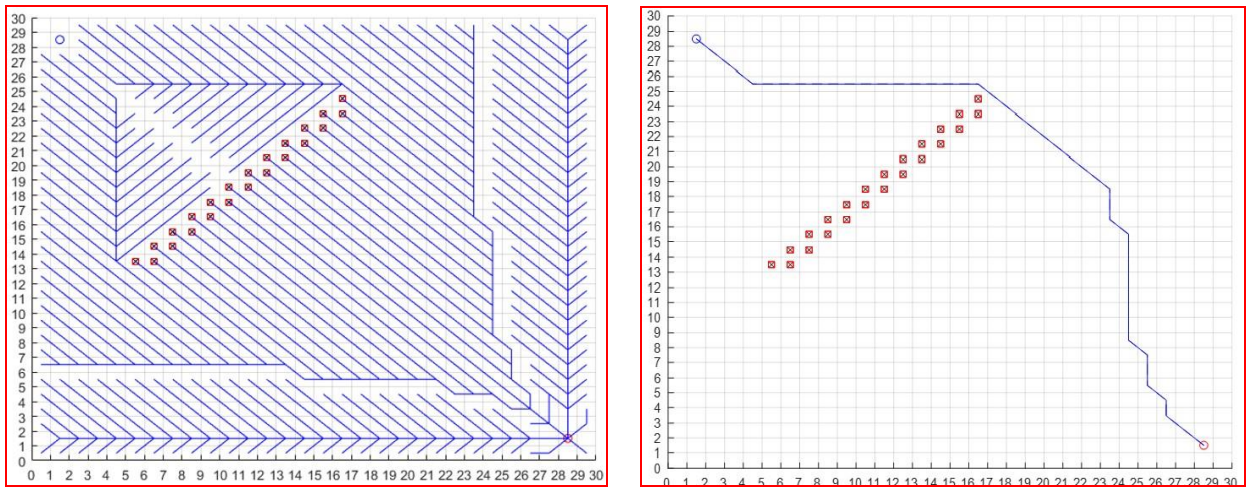
а) Dijkstra алгоритам (лево – сценарио без препреке, десно – сценарио са препреком)



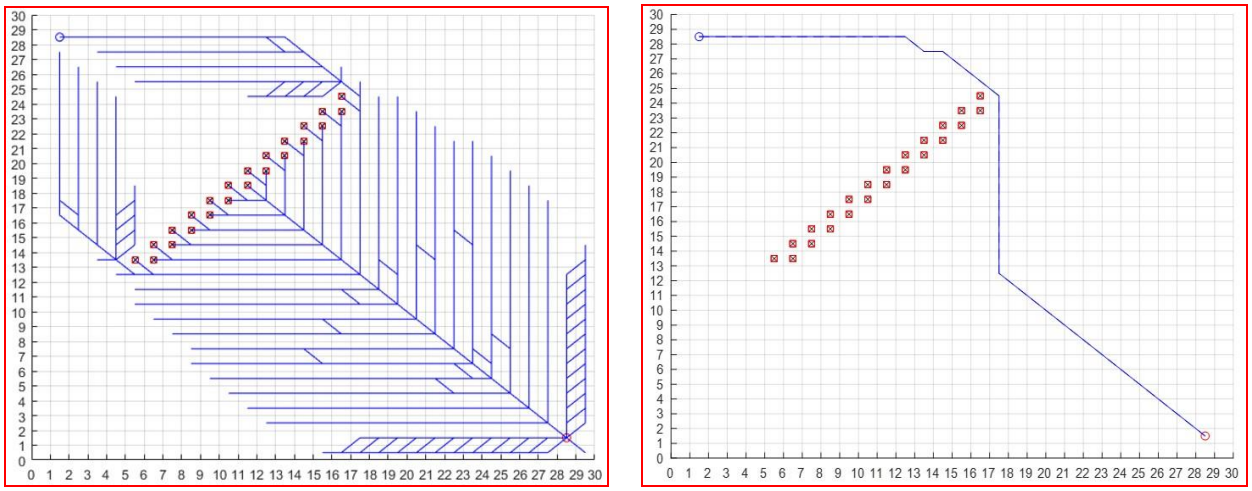
б) A алгоритам (лево – сценарио без препреке, десно – сценарио са препреком)*

Слика 3. Илустративни приказ планирања путање без коришћења хеуристичке функције (Dijkstra алгоритам) и са коришћењем хеуристичке функције (A алгоритам). Стартне позиције су приказане розе бојом, а циљне позиције тамно плавом бојом [56].*

На *Слици 4*. приказан је пример тестирања кодова алгоритама *Dijkstra* и *A**, урађених у склопу истраживања обухваћених дисертацијом, где се такође може видети ефекат увођења хеуристичке функције на процес планирање путање.



а) Dijkstra алгоритам (лево – стабло путања при обради графа, десно – израчуната путања)



б) A алгоритам (лево – стабло путања при обради графа, десно – израчуната путања)*

Слика 4. Пример тестирања кодова алгоритама Dijkstra и A урађених у склопу истраживања обухваћених дисертацијом. Стартне позиције су означене црвеним кружићима, а циљне позиције плавим кружићима.*

Са *Слике 4*. се може приметити да се током прорачуна путање применом *A** алгоритма формира знатно уже стабло путања, него у случају примене *Dijkstra* алгоритма. Такође, иако се форме коначних путања које су израчунали *Dijkstra* и *A** алгоритам визуелно разликују, обе путање су исте дужине и уједно представљају најкраће могуће путање у условима ограничења проистеклих из дискретизације мапе терена.

2.2. *Anytime* алгоритми

У пракси је често потребно генерисати путању робота за што краће време, како би он започео кретање уз минималан губитак времена планираног за извршење задатка. За те потребе се у литератури предлажу тзв. *anytime* алгоритми тј. алгоритми који имају брз

одзив на рачун тога што генеришу субоптималну иницијалну путању чиме скраћују време планирања [31]. У тим ситуацијама роботи започињу своје кретање дуж иницијалне путање, а алгоритми настављају свој итеративни рад у циљу побољшања добијеног решења, све док не генеришу путању која је оптимална (у односу на тренутну позицију робота) или док не истекне расположиво време.

2.2.1 Принцип функционисања *anytime* алгоритама

Као што је наведено у *Поглављу 2.1*. *A** алгоритам са хеуристичком функцијом пондерисаном тежинским фактором $\varepsilon>1$ убрзава генерисање иницијалног решења, али је оно субоптимално, при чему цена добијене путање није више од ε пута већа од цене оптималне путање (путање са најмањом ценом). Како би се креирао један једноставан *anytime* алгоритам са особином субоптималности може се сукцесивно понављати *A** алгоритам са опадајућим тежинским фактором ε којим се пондерише хеуристичка функција. Овакав приступ има контролу над границом субоптималности добијеног решења, али и велики недостатак који се огледа у томе да се свака нова итерација *A** алгоритма у суштини реализује независно у смислу да не користи резултате обраде графа из претходних итерација.

2.2.2 *Anytime Repairing A** (*ARA**) алгоритам

*Anytime Repairing A** (*ARA**) алгоритам [31] подразумева сукцесивно итеративно понављање *A** алгоритма уз опадајући тежински фактор за пондерисање хеуристичке функције, при чему у свакој итерацији користи резултате прорачуна из претходних итерација. На тај начин *ARA** за релативно кратко време генерише иницијално решење које је ε_0-субоптимално (где је ε_0 почетна вредност тежинског фактора за пондерисање хеуристичке функције), како би робот што пре започео кретање, након чега наставља свој итеративни рад у циљу побољшања добијеног решења тј. путање (у односу на тренутну позицију робота), при чему контролише задату границу субоптималности. Да би приступили детаљнијем појашњењу функционисања *ARA** алгоритма, објаснићемо најпре разлику између „*forward*“ и „*backward*“ верзија алгоритама који прорачун путање врше на бази обраде графа. Наиме, за разлику од основне верзије *A** алгоритма који обраду графа и прорачун путање врши у смеру од стартног ка циљном чвору (*forward* верзија) тј. $g(s)$ се рачуна у односу на s_{start} (онако како је то објашњено у *Поглављу 2.1*), обрада графа и прорачун путање могу се вршити и у смеру од циљног ка стартном чвору (*backward* верзија), тако да се $g(s)$ тада рачуна у односу на s_{goal}. Разлог што се у пракси углавном користи *backward* верзија сложених алгоритама итеративне концепције рада, као што је *ARA**, је што у том случају стабло путања остаје валидно приликом преласка из једне у другу итерацију алгоритма, имајући у виду да је у овом случају оно везано за позицију s_{goal}, која се не мења у току времена тј. остаје везана за један исти чвор, док се s_{start} које представља тренутну позицију робота мења од чвора до чвора дуж израчунате путање којом се креће робот [31,52]. Наравно, недостатак је то што се код *backward* верзије алгоритама хеуристичка функција рачуна у односу на позицију s_{start}, па је потребно њено ажурирање. Пошто се на почетку сваке итерације *ARA** алгоритма врши ажурирање приоритета чворова у скупу *OPEN*, то се истом приликом ажурира и хеуристичка функција која у начелу представља једноставну операцију, што умањује утицај овог недостатка. Уважавајући претходно наведено, у овој дисертацији се обрађује *backward* верзија алгоритма *ARA**, а из истог разлога се обрађују и *backward* верзије алгоритама *D* Lite* и *AD**, који ће бити објашњени у даљем току текста.

Битно је, такође, напоменути да се у случају примене *backward* верзија алгоритама појмови чвор „претходник“ и чвор „наследник“ произвољног чвора *s* дефинишу у смеру „ка циљном чвору“, из простог разлога што се стабло путања најмање цене у овом случају формира идући од циљног чвора док се не дође до стартног чвора. Ако посматрамо конкретну путању у стаблу путања којој припада чвор *s* у било ком тренутку обраде графа, чвор „претходник“ чвора *s* је чвор који непосредно физички претходи чвору *s*, док чвор „наследник“ чвора *s* представља чвор који непосредно физички следи после чвора *s* и то идући искључиво у смеру „ка циљном чвору“. Такође, као код *A** алгоритма, и овде у стаблу путања чвор *s* може имати више чворова претходника и више чворова наследника, где скуп претходника чвора *s* означавамо са *Pred*(*s*), док скуп чворова наследника чвора *s* означавамо са *Succ*(*s*). Код *backward* верзије алгоритама се и услов конзистентности хеуристичке функције дефинише на мало другачији, генералнији начин: $h(s, s') \leq c^*(s, s')$ и $h(s, s'') \leq h(s, s') + c^*(s', s'')$, за све чворове $s, s', s'' \in S$, где $c^*(s, s')$ означава најмању цену путање од чвора *s* до чвора *s'* [24,32,55].

Цена путање *g*(*s*), хеуристичка функција *h*(*s*) и кључ *f*(*s*) код *ARA** алгоритма израчунавају се по истој логици као и код *backward* верзије *A** алгоритма [31]. *ARA** алгоритам свој рад започиње тако што, такође на сличан начин као *backward* верзија *A**, генерише иницијално решење (путању) за неку задату вредност тежинског фактора $\varepsilon=\varepsilon_0$. Међутим, у даљем току функционисања постоје значајне разлике *ARA** алгоритма у односу на *A**, које ће бити описане у наставку. *A** алгоритам са хеуристичком функцијом која задовољава услов конзистентности дефинисан у *Поглављу 2.1*. гарантује да сваки проширени чвор неће бити проширен више од једном, али ако дефинишемо да фактор пондерисања ε има вредност већу од један то може довести до тога да *A** алгоритам може вршити ширење једног истог чвора више пута у току исте итерације [31]. Код *ARA** алгоритма је имплементирано ограничење да се један чвор не може проширити више од једном у току исте итерације, што значајно убрзава рад алгоритма и то на начин да је граница субоптималности и даље очувана [31,32]. Конкретно, код *ARA** алгоритма поред скупа *OPEN* уведен је скуп *CLOSED*, који садржи све чворове који су претходно били у скупу *OPEN* и који су у међувремену проширени. Међутим, са овим ограничењем скуп *OPEN* не може више садржати све инконзистентне чворове (сваки проширени чвор може поново постати инконзистентан), тј. садржи само инконзистентне чворове који претходно нису били проширени. Важно је, међутим, да се сачувају информације о свим инконзистентним чворовима на крају текуће итерације алгоритма, како би били узети у обзир при пропагацији инконзистенције у наредној итерацији. На тај начин се искоришћавају резултати обраде графа из претходне фазе (итерације). Да би се то постигло, код *ARA** алгоритма формира се додатни скуп *INCONS* у који се инсертују сви инконзистентни чворови који нису испунили поменути услов за инсертовање у скуп *OPEN*. Стога, код *ARA** алгоритма инконзистентне чворове чини у ствари унија скупова *OPEN* и *INCONS*, тако да чворове из оба наведена скупа треба узети у обзир при обради графа тј. пропагацији инконзистенције у наредној итерацији алгоритма.

Сумирајмо на крају принцип функционисања *ARA** алгоритма, према псеудокоду који је приказан је на *Слици 5* [32]. Након иницијализације, *ARA** у сваком кораку прорачуна путање брише из скупа *OPEN* чвор *s* са најмањом вредношћу кључа (линија 3.), инсертује га у скуп *CLOSED* (линија 4.), затим проверава и ако за то постоје услови ажурира (смањује) цену путање сваког од његових суседа, потенцијалних чворова претходника (линије 5.-9.). Чвор *s'*, сусед чвора *s*, постаје његов претходник ако се током ове операције испостави да до тог тренутка израчуната најкраћа путања од *s'* до s_{goal} иде преко *s*. Сваког суседа чвора *s* чија цена путање је том приликом смањена алгоритам проглашава инконзистентним и категорише га као члана скупа *OPEN* уз истовремено ажурирање његовог кључа и показивача, изузев ако је већ претходно био проширен и у

том случају га инсертује у скуп *INCONS*, да би се та информација искористила у следећој итерацији алгоритма (линије 10.-13.).

Дакле, процес ширења чвора код *ARA** алгоритма подразумева реализацију линија 3.-13. псеудокода [31]. На овај начин се шири и инконзистенција, чиме се обезбеђује да се чворовима који се обрађују увек додељује најмања цена путање пронађена до тог тренутка. Алгоритам тако шири стабло путања најмањих цена, али у овом случају у смеру од циљног чвора до обрађених чворова све док не дође до стартног чвора. Када се текућа итерација алгоритма заврши, сукцесивним повезивањем показивача од s_{start} до s_{goal} формира се израчуната путања коју робот следи током кретања. Пре отпочињања следеће итерације алгоритам декрементира фактор ε, сви чворови из скупа *INCONS* се пребацују у скуп *OPEN* (линије 21. и 22.), који се користи за прорачун путање у наредној итерацији. Скуп *CLOSED* се празни, а свим чворовима из проширеног скупа *OPEN* се ажурира вредност кључа у складу са новом вредношћу фактора ε и промењеном позицијом s_{start} (линије 23. и 24.).

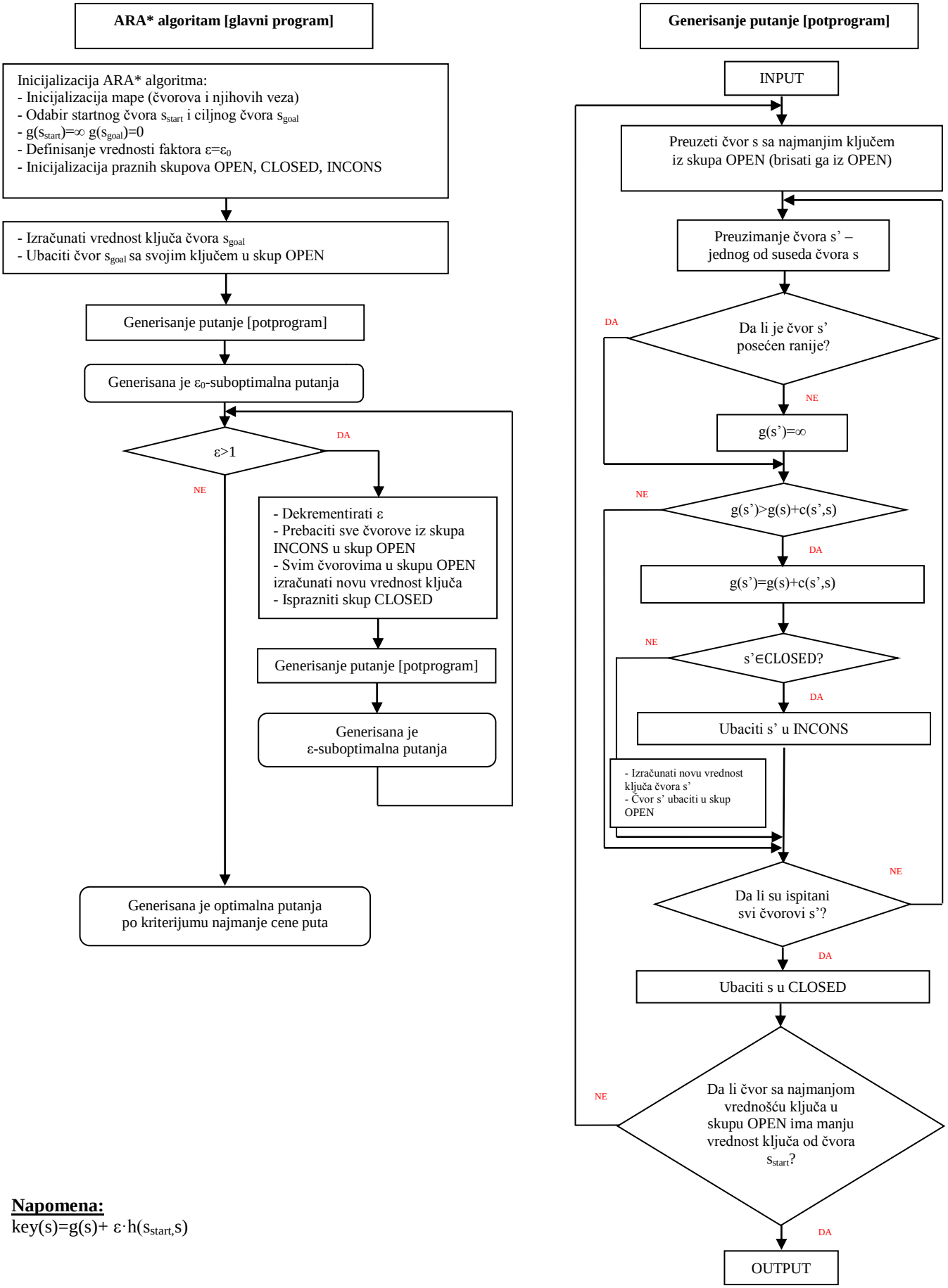
key(s)
01. return $g(s) + \epsilon \cdot h(s_{start}, s)$;

ImprovePath()
02. while ($\min_{s \in OPEN}$(key(s)) < key(s_{start}))
03. remove s with the minimum key from *OPEN*;
04. $CLOSED = CLOSED \cup \{s\}$;
05. for all $s' \in Pred(s)$
06. if s' was not visited before
07. $g(s') = \infty$;
08. if $g(s') > c(s', s) + g(s)$
09. $g(s') = c(s', s) + g(s)$;
10. if $s' \notin CLOSED$
11. insert s' into *OPEN* with key(s');
12. else
13. insert s' into *INCONS*;

Main()
14. $g(s_{start}) = \infty$; $g(s_{goal}) = 0$;
15. $\epsilon = \epsilon_0$;
16. $OPEN = CLOSED = INCONS = \emptyset$;
17. insert s_{goal} into *OPEN* with key(s_{goal});
18. ImprovePath();
19. publish current ϵ-suboptimal solution;
20. while $\epsilon > 1$
21. decrease ϵ;
22. Move states from *INCONS* into *OPEN*;
23. Update the priorities for all $s \in OPEN$ according to key(s);
24. $CLOSED = \emptyset$;
25. ImprovePath();
26. publish current ϵ-suboptimal solution;

Слика 5. ARA алгоритам [32].*

Дијаграм тока *ARA** алгоритма, урађен у склопу истраживања обухваћеног овом дисертацијом, приказан је на *Слици 6*.



Napomena:
$key(s)=g(s)+ \varepsilon \cdot h(s_{start},s)$

Слика 6. Дијаграм тока ARA алгоритма, урађен у склопу истраживања обухваћених дисертацијом.*

2.3. Динамички (инкрементални) алгоритми

*ARA** алгоритам реализује се у више итерација за различите вредности тежинског фактора ε, али увек над истим графом и са непромењеним ценама прелазака између чворова. У реалним применама су, ипак, најчешће ситуације у којима је у тренутку отпочињања мисије окружење у које се робот упућује само делимично познато. У таквим ситуацијама кретање робота обично отпочиње уз полазну претпоставку да су сва подручја која су у почетном тренутку непозната уједно и слободна за пролаз, док робот у току кретања на основу информација прикупљених својим сензорима врши детаљно испитивање терена. Поред тога, могуће су и динамичке промене окружења у току самог кретања робота. Без обзира о којој конкретно ситуацији се ради, ове промене подразумевају измене у графу на основу којег је алгоритам извршио генерисање путање. У таквим околностима може се догодити да решење до ког је дошао *ARA** алгоритам није оптимално или у очекујућим границама суботималности, а можда чак ни валидно што је најгора опција. Због тога *ARA** у овом случају врши прорачун путање из почетка, при чему се не користе резултати из претходних итерација алгоритма. У поменутим сценаријима ефикаснија је класа алгоритама познатих под називом инкрементални или динамички алгоритми.

Динамички алгоритми базирани на концепту *A** алгоритма иницијално решење (путању) генеришу на сличан начин као оригинални *A** алгоритам. Када дође до измене окружења или уопштеније речено до појаве нових информација од утицаја на генерисану путању, иницијално решење се коригује уз максимално коришћење резултата из претходне фазе обраде графа. Као резултат добија се да је планирање путање на овај начин значајно ефикасније него код алгоритама који у сваком појединачном случају измене окружења генерисање решења почињу из почетка.

2.3.1. *D* Lite* алгоритам

D Lite* је ефикасан инкрементални алгоритам за планирање путање робота [24]. Представља поједностављену верзију (у смислу алгоритамске процедуре) стандардног *D** алгоритма [25], при чему су и *D** и *D* Lite* суштински напредне надоградње базичног *A** алгоритма [57]. *D** алгоритам је 1 до 2 реда величине ефикаснији од *A** алгоритма [26], у смислу да брже долази до решења, због чега је нашао широку примену у пракси, укључујући беспосадне верзије возила *HMMWV*, беспосадна возила која су учествовала на *Defense Advanced Research Projects Agency (DARPA)* такмичењима, *NASA* ровере на Марсу итд. [26-30]. *D* Lite* захтева око 30% мање линија кода од *D**, а са друге стране је минимално ефикасан као и *D**, у појединим случајевима је и ефикаснији [26]. Стога је *D* Lite* данас један од најчешће примењиваних алгоритама за планирање путање робота у пракси, а неке од беспосадних платформи на којима је успешно имплементиран приказане су на *Слици 7* [49]. Такође, и *D** и *D* Lite* се могу користити за планирање путања вишероботских система [58-60].

Слика 7. Неке од беспосадних платформи на којима је успешно примењен D Lite алгоритам у комбинацији са интерполацијом (Field D*) ради добијања „природније“ путање [49].*

D Lite* генерише путању између стартног чвора $s_{start} \in S$ и циљног чвора $s_{goal} \in S$, где је S ограничени скуп чворова, при чему је основна верзија алгоритма *backward* тј. обраду графа врши од циљног према стартном чвору.

У оквиру обраде произвољног чвора s у графу ($s \in S$) *D* Lite* алгоритам, као и *ARA**, рачуна најмању цену путање од чвора s до чвора s_{goal}, $g(s)$. Поред тога, *D* Lite* алгоритам рачуна и тзв. параметар $rhs(s)$, који представља збир најмање од цена путања његових наследника $g(s')$ и цене преласка из s у s', тј. [24,61]:

$$rhs(s) = min_{s' \in Succ(s)} \left(c(s,s') + g(s') \right)$$
$$rhs(s) = 0 \text{ ако је } s = s_{goal} \quad (1)$$

где $Succ(s)$ означава скуп наследника чвора s.

Параметар $rhs(s)$ представља информацију о најбољем суседу чвора s у актуелном тренутку и управо та информација се користи код корекције путање у случају појаве/детекције нових препрека или пролаза у окружењу у ком се креће робот. Показивач чвора s је у овом случају увек усмерен према чвору из ког је изведено $rhs(s)$ и, наравно, може се мењати у току рада алгоритма. Показивачи служе, као и код *anytime* алгоритама, да се на основу њих пратећи их у смеру од стартном према циљног чвору након завршетка обраде графа формира путања.

Суштина увођења параметра $rhs(s)$ је да омогућава детаљнију анализу појаве инконзистенције у односу на анализу инконзистенције примењену код алгоритама *A** и *ARA**. Код инкременталних алгоритама чвор је прекоконзистентан ако је $g(s) > rhs(s)$, подконзистентан ако је $g(s) < rhs(s)$, а конзистентан ако је $g(s) = rhs(s)$. Када дође до смањења цене путање $g(s')$ чвора s' који је сусед чвора s, то може довести до смањења $rhs(s)$, што може чвор s учинити прекоконзистентним. У другом случају, повећање цене путање $g(s')$ чвора s' који је сусед чвора s може довести до повећања $rhs(s)$, што може чвор s учинити подкоконзистентним. Инконзистентни чворови тригерују процес ажурирања параметара њихових суседа, те се стога може рећи да прекоконзистентни чворови врше пропагацију смањења цена путања, а подконзистентни чворови пропагацију повећања цена путања у окружењу у ком се креће робот. Као и код претходно анализираних алгоритама, инконзистентни чворови се инсертују у скуп *OPEN*. Поред скупа *OPEN*, *D* Lite* као и алгоритми *A** и *ARA** користи хеуристичку функцију у циљу фокусирања обраде графа, која у овом случају треба да задовољи услов конзистентности за *backward* верзије алгоритама дефинисан у *Поглављу 2.2.2*.

Приоритизација чворова у скупу отворених чворова и овде се врши на основу вредности додељене функције кључа. Међутим, за разлику од алгоритама *A** и *ARA**, код *D* Lite* кључ има форму вектора врсте димензије 1x2:

$$key(s) = [k_1(s), k_2(s)] = [\min(g(s), rhs(s)) + h(s_{start}, s), \min(g(s), rhs(s))] \quad (2)$$

Вредност кључа чвора *s* је мања од вредности кључа чвора *s'* тј. $key(s) < key(s')$, ако је $k_1(s) < k_1(s')$ или ако је $k_1(s) = k_1(s')$ и $k_2(s) < k_2(s')$.

Псеудокод основне верзије *D* Lite* алгоритма приказан је на *Слици 8* [32]. Након иницијализације алгоритма (линије 15.-17.), *D* Lite* израчунава путању најмање цене (линија 19.). При прорачуну путање алгоритам у сваком кораку врши брисање из скупа *OPEN* инконзистентног чвора *s* који у том тренутку има најмању вредност кључа (линија 8.). Ако је чвор *s* прекоконзистентан, ажурира се његова вредност цене путање $g(s)$ како би се учинио конзистентним (линија 10.). Уколико је чвор подконзистентан, алгоритам вредност његове цене путање сетује на ∞ (линија 13.). Без обзира да ли је чвор *s* прекоконзистентан или подконзистентан, његово процесуирање на горе описан начин може утицати на конзистенцију суседних чворова, због чега се ажурира и њихов параметар *rhs*, као и статус. Након ажурирања параметра *rhs*, сваки суседни чвор се проверава на услов инконзистенције и у случају задовољења тог услова инсертује се у скуп *OPEN* (у случају да је већ члан скупа *OPEN* брише се из тог скупа да би нова провера на услов инконзистенције имала смисла) (линије 4.-6.). Тиме се омогућава пропагација инконзистенције. Ширење чворова из скупа *OPEN* у овом случају подразумева реализацију линија 8.-14. псеудокода и врши се све док је задовољен најмање један од следећа два услова: да има чворова чија је вредност кључа мања од вредности кључа стартног чвора или да важи $rhs(s_{start}) \neq g(s_{start})$ (линија 7.). Описани процес гарантује добијање путање од s_{goal} до s_{start} са најмањом ценом [32]. Линија 18. практично значи да алгоритам завршава свој рад у тренутку када се задовољи услов $s_{start} = s_{goal}$ [62].

Ако након генерисања иницијалне путање и отпочињања кретања робот у неком тренутку детектује промене у окружењу (у односу на мапу према којој је израчуната иницијална путања), *D* Lite* најпре врши ажурирање измењених цена прелазака, а затим и ажурирање параметара и статуса свих чворова који су директно погођени насталом променом (линије 21.-23.). На овај начин ажурира се и скуп *OPEN*, тј. у њега се додају чворови кандидати за корекцију путање. Потом се из ажурираног скупа *OPEN* бришу и шире чворови према приоритизацији у складу са додељеном вредношћу кључа на горе описан начин док се не испуни услов за терминацију рада алгоритма. На тај начин *D* Lite* алгоритам врши корекцију постојеће путање (не израчунава нову путању од почетка), што га чини много ефикаснијим од базичног *A** алгоритма [63].

D Lite* алгоритам врши ажурирање параметара само оних чворова који су директно погођени насталом променом цена прелазака. Другим речима, овај алгоритам не понавља обраду над целим графом као на почетку рада, већ користи претходно добијене резултате при израчунавању кориговане путање, тако што ажурира скуп *OPEN* на основу насталих промена у окружењу. Укључивањем $k_2(s)$ у вредност кључа чвора *s*, *D* Lite* обезбеђује да се чворови из скупа *OPEN* шире на најефикаснији начин [32]. Са друге стране, укључивање $k_1(s)$ у вредност кључа чвора *s*, због чињенице да $k_1(s)$ садржи и хеуристичку функцију, обезбеђује да се, када се цене прелазака смање, шире само нови преконзистентни чворови, тј. чворови за које постоји вероватноћа да могу смањити цену актуелне путање од стартног до циљног чвора [32]. У супротном, из истог разлога, укључивање $k_1(s)$ у вредност кључа чвора *s*, обезбеђује да се, када се цене прелазака повећају, процесуирају само нови подконзистентни чворови, тј. чворови за које постоји

вероватноћа да су учинили невалидном актуелну путању од стартног до циљног чвора [32].

```
key(s)
01. return [min(g(s), rhs(s)) + h(s_start, s); min(g(s), rhs(s))];

UpdateState(s)
02. if s was not visited before
03.     g(s) = ∞;
04. if (s ≠ s_goal) rhs(s) = min_{s'∈Succ(s)}(c(s, s') + g(s'));
05. if (s ∈ OPEN) remove s from OPEN;
06. if (g(s) ≠ rhs(s)) insert s into OPEN with key(s);

ComputeShortestPath()
07. while (min_{s∈OPEN}(key(s)) < key(s_start) OR rhs(s_start) ≠ g(s_start))
08.     remove state s with the minimum key from OPEN;
09.     if (g(s) > rhs(s))
10.         g(s) = rhs(s);
11.         for all s' ∈ Pred(s) UpdateState(s');
12.     else
13.         g(s) = ∞;
14.         for all s' ∈ Pred(s) ∪ {s} UpdateState(s');

Main()
15. g(s_start) = rhs(s_start) = ∞; g(s_goal) = ∞;
16. rhs(s_goal) = 0; OPEN = ∅;
17. insert s_goal into OPEN with key(s_goal);
18. forever
19.     ComputeShortestPath();
20.     Wait for changes in edge costs;
21.     for all directed edges (u, v) with changed edge costs
22.         Update the edge cost c(u, v);
23.         UpdateState(u);
```

Слика 8. D Lite алгоритам [32].*

Осврнимо се на начин на који *D* Lite* шири подконзистентне и прекоконзистентне чворове, односно како решава проблем њиховог утицаја на корекцију путање. *D* Lite* брише подконзистентан чвор *s* (*g*(*s*) < *rhs*(*s*)) из скупа *OPEN* (у складу са претходно извршеном приоритизацијом), при чему се његовом параметру *g* додељује вредност ∞. Затим се иницијализује ажурирање параметара *rhs* суседних чворова (као и њихових показивача) и статуса. Такође се ажурира и параметар *rhs* и статус самог чвора *s* и он се враћа у скуп *OPEN* уколико вредност *rhs*(*s*) није ∞ (параметру *g* је претходно додељена вредност ∞). Чвор *s* се на тај начин може поново проширити као прекоконзистентан, када *g*(*s*) добија реалну вредност (једнаку *rhs*), што је битно за комплетирање процеса корекције путање. Доказ да се један исти чвор може проширити највише два пута (највише једном као подконзистентан и највише једном као прекоконзистентан) у току израчунавања путање тј. у току једног извршавања функције *Computepath* дат је у [52,53].

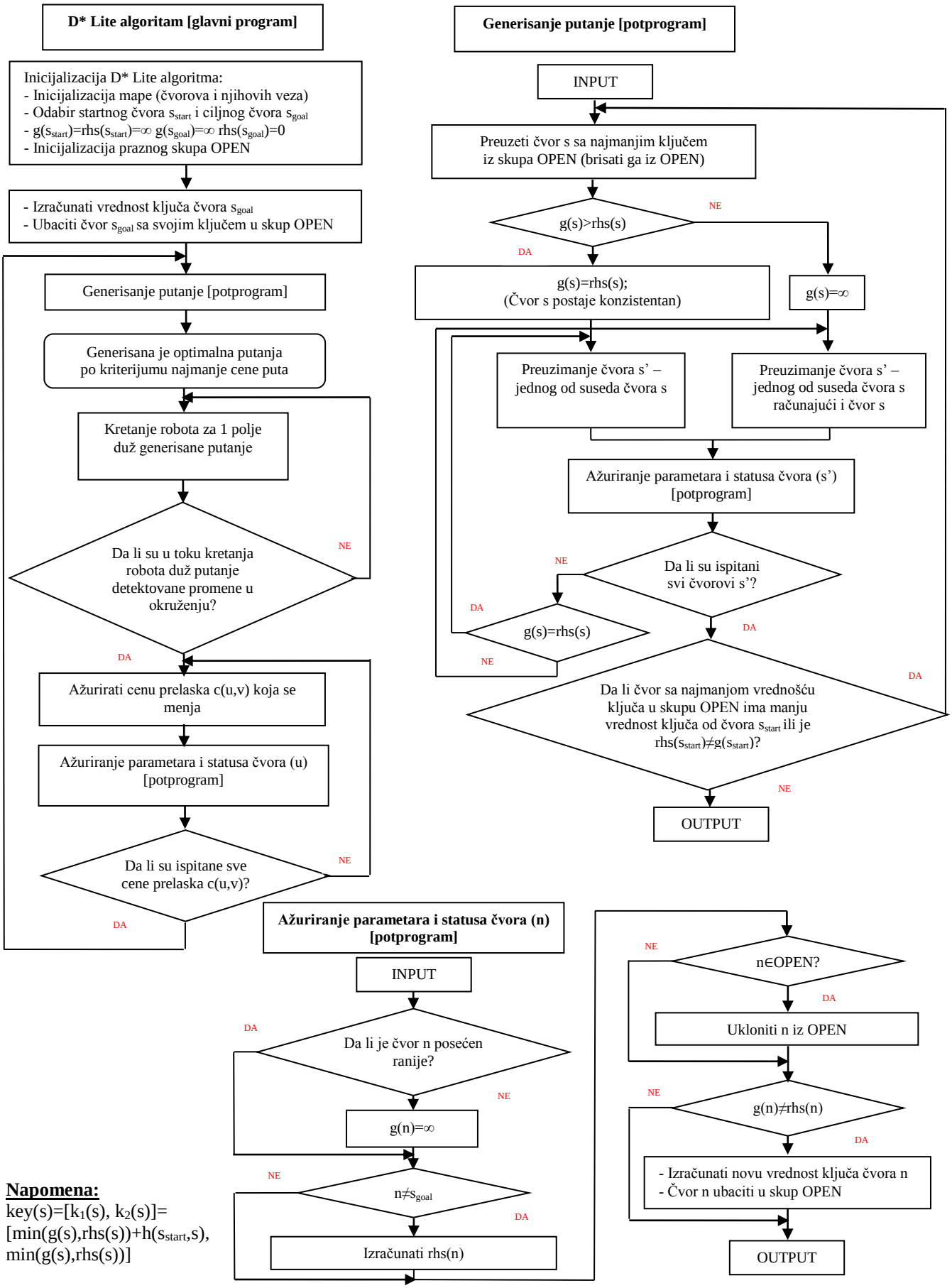
Процесуирање прекоконзистентног чвора *s* је једноставније и подразумева, такође, његово брисање из скупа *OPEN* (у складу са претходно извршеном приоритизацијом), али се у овом случају смањује вредност параметра *g*(*s*) (постаје једнака вредности параметра

rhs), што чини могућим да неки од његових суседа након ажурирања сопственог параметра *rhs* „усмери свој показивач“ према *s* и тиме га укључи у стабло путања.

Важно је напоменути и то да *D* Lite* алгоритам не подразумева никакву полазну претпоставку о томе да ли се промене цена прелазака између чворова јављају услед промена у окружењу у ком се робот креће или је само у питању ажурирање мапе која није била у потпуности позната пре отпочињања кретања. Може се користити за решавање проблема кретања робота и у потпуно непознатом окружењу, под условом да је познат циљни чвор.

На крају, приметимо да *D* Lite* алгоритам, за разлику од *ARA**, због начина на који функционише (није итеративне природе и не користи пондерисану хеуристичку функцију) у току рада не формира скупове *CLOSED* и *INCONS*, већ само скуп *OPEN*. Коришћење резултата из иницијалне обраде графа након појаве нових информација везаних за промене у окружењу обезбеђено је на тај начин што се иницијално формирани скуп *OPEN* ажурира у смислу проширења новопојављеним инконзистентним чворовима. *D* Lite*, ипак, може бити предефинисан у смислу да користи пондерисану хеуристичку функцију и у том случају генерише решење које је субоптимално са одређеном, фиксном границом субоптималности.

Дијаграм тока *D* Lite* алгоритма, урађен у склопу истраживања обухваћеног овом дисертацијом, приказан је на *Слици 9*.



Слика 9. Дијаграм тока D Lite алгоритма, урађен у склопу истраживања обухваћених дисертацијом.*

2.3.2. *Anytime Dynamic A** (*AD**) алгоритам

Као што је наведено у претходним поглављима, постоје алгоритми за ефикасно решавање проблема кретања у динамичком окружењу (*D* Lite* алгоритам), као и алгоритми за планирање кретања када је иницијалну путању потребно генерисати за што краће време (*ARA** алгоритам). Међутим, које је решење за ситуације које подразумевају истовремено присуство оба наведена проблема?

*Anytime Dynamic A** (*AD**) је један од алгоритама који има могућност инкременталног и *anytime* планирања [32]. Функционише на начин да у релативно кратком времену генерише иницијално решење које је ε_0-субоптимално, након чега робот започиње кретање дуж генерисане путање, а алгоритам наставља свој итеративни рад у циљу побољшања добијеног решења (при чему контролише границу субоптималности), све док не генерише путању која је оптимална у односу на тренутну позицију робота (тј. тренутну позицију чвора s_{start}). При томе, на почетку сваке нове итерације поступа се на исти начин као код *ARA** алгоритма - декрементира се ε, инсертују се сви чворови из скупа *INCONS* у скуп *OPEN*, свим чворовима у новоформираном скупу *OPEN* израчунају се нове вредности кључа, скуп *CLOSED* се испразни. Псеудокод *backward* верзије *AD** алгоритма приказан је на *Слици 10* [32].

key(s)
01. if $(g(s) > rhs(s))$
02. return $[rhs(s) + \epsilon \cdot h(s_{start}, s); rhs(s)]$;
03. else
04. return $[g(s) + h(s_{start}, s); g(s)]$;

UpdateState(s)
05. if s was not visited before
06. $g(s) = \infty$;
07. if $(s \neq s_{goal})$ $rhs(s) = \min_{s' \in Succ(s)}(c(s, s') + g(s'))$;
08. if $(s \in OPEN)$ remove s from *OPEN*;
09. if $(g(s) \neq rhs(s))$
10. if $s \notin CLOSED$
11. insert s into *OPEN* with key(s);
12. else
13. insert s into *INCONS*;

ComputeorImprovePath()
14. while $(\min_{s \in OPEN}(\text{key}(s)) \dot{<} \text{key}(s_{start})$ OR $rhs(s_{start}) \neq g(s_{start}))$
15. remove state s with the minimum key from *OPEN*;
16. if $(g(s) > rhs(s))$
17. $g(s) = rhs(s)$;
18. $CLOSED = CLOSED \cup \{s\}$;
19. for all $s' \in Pred(s)$ UpdateState(s');
20. else
21. $g(s) = \infty$;
22. for all $s' \in Pred(s) \cup \{s\}$ UpdateState(s');

Main()
01. $g(s_{start}) = rhs(s_{start}) = \infty; g(s_{goal}) = \infty$;
02. $rhs(s_{goal}) = 0; \epsilon = \epsilon_0$;
03. $OPEN = CLOSED = INCONS = \emptyset$;
04. insert s_{goal} into *OPEN* with key(s_{goal});
05. ComputeorImprovePath();
06. publish current ϵ-suboptimal solution;
07. forever
08. if changes in edge costs are detected
09. for all directed edges (u, v) with changed edge costs
10. Update the edge cost $c(u, v)$;
11. UpdateState(u);
12. if significant edge cost changes were observed
13. increase ϵ or replan from scratch;
14. else if $\epsilon > 1$
15. decrease ϵ;
16. Move states from *INCONS* into *OPEN*;
17. Update the priorities for all $s \in OPEN$ according to key(s);
18. $CLOSED = \emptyset$;
19. ComputeorImprovePath();
20. publish current ϵ-suboptimal solution;
21. if $\epsilon = 1$
22. wait for changes in edge costs;

Слика 10. AD алгоритам [32].*

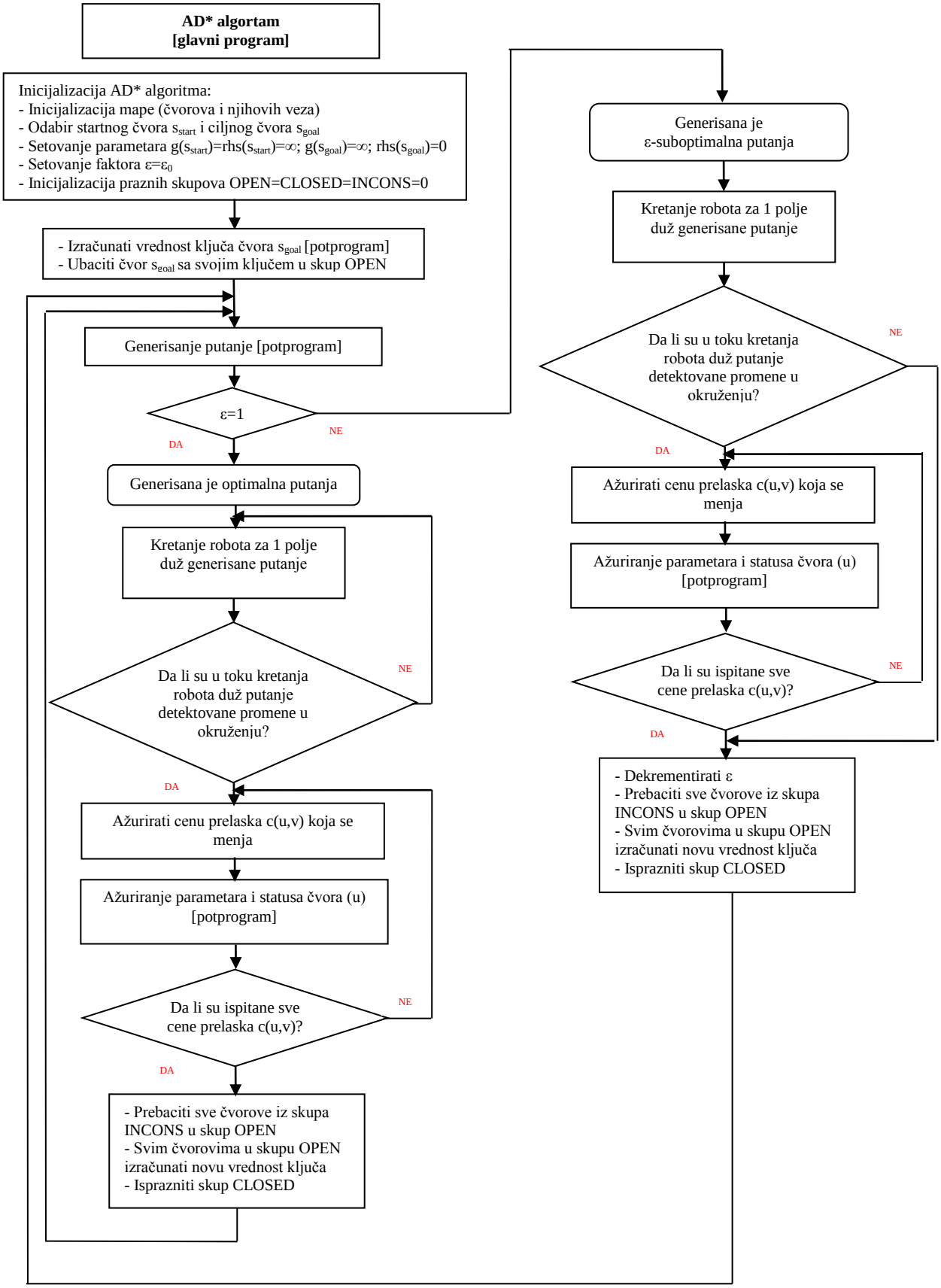
Као што се види, псеудокод *AD** алгоритма тежишно је базиран на псеудокоду *D* Lite* алгоритма, уз измену главне функције како би се имплементирала могућност *anytime* рада, на исти начин као код *ARA**. Пошто је истовремено и инкрементални и *anytime* алгоритам, *AD** вредност функције кључа рачуна на измењен начин у односу и на *D* Lite* и на *ARA**, који је прилагођен у циљу правилне пропагације прекоконзистенције и

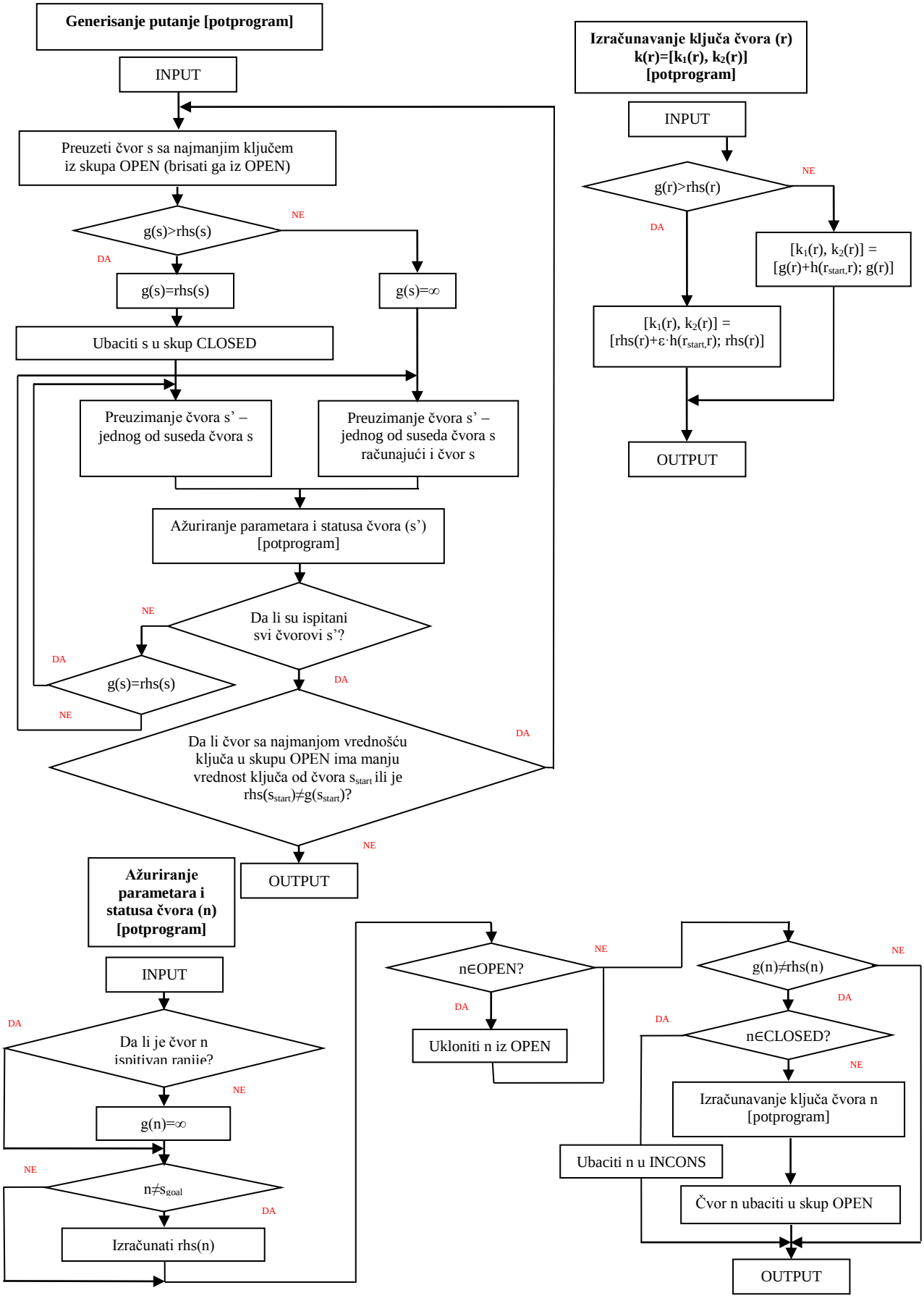
подконзистенције у овим условима. Постоји још једна специфичност код *AD**, а то је да у случају када су промене цена прелазака између чворова бројне, алгоритам инкрементира ε, како би се до решења дошло брже, по цени да је оно мањег квалитета од тренутног или у таквим случајевима прорачун путање просто почиње из почетка.

Када дође до детекције промене цена прелазака између чворова, *AD** алгоритам, као и *D* Lite*, врши најпре ажурирање измењених цена прелазака, а затим и ажурирање параметара и статуса свих чворова који су директно погођени насталом променом. На овај начин ажурира се и скуп *OPEN*, тј. додају се чворови кандидати за корекцију путање. Потом се из ажурираног скупа *OPEN* чворови шире на исти начин као и код *D* Lite* алгоритма. Ширење чворова из скупа *OPEN* врши се све док је задовољен најмање један од следећа два услова: да има чворова чија је вредност кључа мања од вредности кључа стартног чвора или да важи $rhs(s_{start}) \neq g(s_{start})$. Резултат оваквог рада је израчунавање кориговане путање која је валидна и при томе ε-субоптимална.

Дакле, *AD** алгоритам има могућност да истовремено процесуира промену цена прелазака између чворова насталу услед детекције промена у окружењу, као и декрементирање тежинског фактора ε. Стога је ефикасан за *anytime* планирање кретања у динамичком окружењу.

Дијаграм тока *AD** алгоритма, урађен у склопу истраживања обухваћеног овом дисертацијом, приказан је на *Слици 11*.


AD* algortam
[glavni program]
Inicijalizacija AD* algoritma:
- Inicijalizacija mape (čvorova i njihovih veza)
- Odabir startnog čvora s_{start} i ciljnog čvora s_{goal}
- Setovanje parametara $g(s_{start})=rhs(s_{start})=\infty$; $g(s_{goal})=\infty$; $rhs(s_{goal})=0$
- Setovanje faktora $\varepsilon=\varepsilon_0$
- Inicijalizacija praznih skupova OPEN=CLOSED=INCONS=0
- Izračunati vrednost ključa čvora s_{goal} [potprogram]
- Ubaciti čvor s_{goal} sa svojim ključem u skup OPEN
Generisanje putanje [potprogram]
ε=1
NE
DA
Generisana je optimalna putanja
Kretanje robota za 1 polje duž generisane putanje
Da li su u toku kretanja robota duž putanje detektovane promene u okruženju?
NE
DA
Ažurirati cenu prelaska c(u,v) koja se menja
Ažuriranje parametara i statusa čvora (u) [potprogram]
Da li su ispitane sve cene prelaska c(u,v)?
NE
DA
- Prebaciti sve čvorove iz skupa INCONS u skup OPEN
- Svim čvorovima u skupu OPEN izračunati novu vrednost ključa
- Isprazniti skup CLOSED
Generisana je ε-suboptimalna putanja
Kretanje robota za 1 polje duž generisane putanje
Da li su u toku kretanja robota duž putanje detektovane promene u okruženju?
NE
DA
Ažurirati cenu prelaska c(u,v) koja se menja
Ažuriranje parametara i statusa čvora (u) [potprogram]
Da li su ispitane sve cene prelaska c(u,v)?
NE
DA
- Dekrementirati ε
- Prebaciti sve čvorove iz skupa INCONS u skup OPEN
- Svim čvorovima u skupu OPEN izračunati novu vrednost ključa
- Isprazniti skup CLOSED



Слика 11. Дијаграм тока AD алгоритма, урађен у склопу истраживања обухваћених дисертацијом.*

2.4. Примери и поређење функционисања алгоритама *ARA*, D* Lite* и *AD**

У овом поглављу биће презентовани конкретни примери функционисања алгоритама *ARA*, D* Lite* и *AD**, као и њихово поређење. За хеуристичку функцију $h(s)$ за потребе ове анализе усвојена је Еуклидова удаљеност чвора s од чвора s_{start}. Промене у окружењу се детектују посматрајући суседне ћелије у мапи за планирање путање из ћелије у којој се тренутно налази робот, по аналогији са сензорским опажањем околине из перспективе робота.

2.4.1. Процесуирање промене статуса чворова при обради графа

Процесуирање промене статуса блокираних чворова у слободне чворове и обрнуто, при обради графа, може се вршити на различите начина. Оно што је заједничко за све приступе при обради графа јесте да цена преласка између два суседна чвора зависи од међусобне удаљености чворова и тежина ћелија које представљају [49]. Концепт униформне дискретизације мапе терена и њеног приказа у виду мреже ћелија једнаких димензија, као и начин формирања одговарајућег графа којим се моделира овако дискретизована мапа, објашњен је у уводном делу *Поглавља 2*. Ради поједностављења за потребе истраживања алгоритама за планирање путање, у литератури се обично димензија ћелије мапе усваја као јединична величина, како при анализи мапе, тако и при обради одговарајућег графа и тај приступ је примењен у дисертацији.

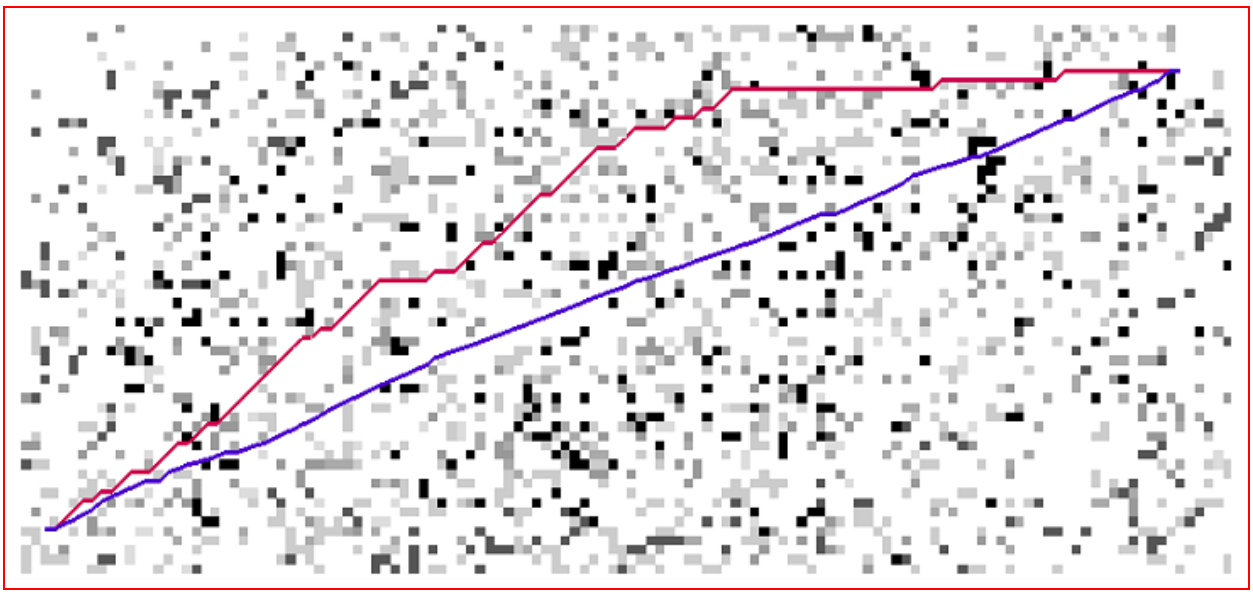
У [50] се предлаже приступ по коме је цена преласка између два суседна слободна чвора једнака тежини ћелије у коју се врши померај помноженој са удаљеношћу између чворова. Пошто је у конкретном случају за тежине слободних ћелија дефинисана вредност 1, цене прелазака које подразумевају ортогонални померај између слободних чворова у графу имају вредност 1, док оне које подразумевају дијагонални померај имају вредност $\sqrt{2}$. Специфичност приступа изложеног у [50] је да, у циљу процесуирања мањег броја чворова при израчунавању путање, алгоритми у току рада „игноришу“ (не процесуирају) блокиране чворове, пошто није могућа путања која их садржи, а ови чворови постају „видљиви“ (за алгоритме) тек када се деблокирају и, наравно, када се та промена детектује сензором робота. Да би у овом случају алгоритми *D* Lite* и *AD** правилно функционисали врше процесуирање параметара и статуса и чвора који је деблокиран и околних чворова. У генералном случају се ажурирају параметри и статус само околних чворова (линије 22. и 23. у псеудокоду *D* Lite* и линије 10. и 11. у псеудокоду *AD** на *Слици 8.* и *Слици 10*, респективно). У случају када се уместо слободног чвора појави блокирани чвор и ту промену детектује робот, њега алгоритам у даљем раду „игнорише“ као и све блокиране чворове. Претходно се брише из скупа *OPEN* уколико је био члан тог скупа, а такође се поништавају и његови параметри *g* и *rhs*, као и показивач. Описани приступ се сугерише и у [32].

У [54] се такође предлаже приступ по коме је цена преласка између два суседна слободна чвора једнака тежини ћелије у коју се врши померај помноженој са удаљеношћу између чворова, али се у овом случају „не игноришу“ блокирани чворови, већ се чворови процесуирају по реду који дефинише логика алгоритама независно да ли су блокирани или не, а „механизми“ прорачуна путање онемогућавају ширење оних путања које садрже блокирани чвор. У конкретном случају је дефинисано да тежине слободних ћелија имају вредност 1, а тежине блокираних ћелија имају вредност ∞. Сходно наведеном, цена

преласка из блокираног у слободан чвор има вредност 1 или √2, у зависности од тога да ли се врши ортогонални или дијагонални померај, док цена преласка из слободног у блокирани чвор у свакој варијанти помераја има вредност ∞. Наравно, цене прелазака које подразумевају ортогонални померај између слободних чворова у графу имају вредност 1, док оне које подразумевају дијагонални померај имају вредност √2.

Такође треба истаћи да у оба наведена случаја алгоритми *A** и *D* Lite* (као и *ARA** и *AD** у ситуацији када је ε=1) генеришу путању најмање цене која је уједно и најкраћа путања у условима ограничења проистеклих из дискретизације мапе терена по претходно објашњеној логици (чија последица је минимални инкремент промене правца путање од π/4).

У [49] је изложен сличан приступ као у [54], с тим да се предлаже шири распон вредности тежина ћелија, па самим тим и одговарајућих цена прелазака између чворова, који као такав реалније осликава сложеност окружења са аспекта проходности тј. могућности кретања робота. У овом случају алгоритам генерише путању најмање цене која у општем случају није најкраћа путања. Поред тога, у [49] се предлаже примена интерполације како би се у одређеној мери превазишла поменута ограничења проистекла из дискретизације мапе тј. минимални инкремент промене правца путање од π/4. На *Слици 12.* је приказан пример планирања путање робота коришћењем *D* Lite* алгоритам без примењене интерполације и са примењеном интерполацијом (*Field D**) у случају мапе чијим ћелијама је додељен шири распон вредности у зависности од проходности терена [49]. При имплементацији у Матлабу приступа из [49] и [54] за тежину блокираних чворова се дефинише неки произвољно велики број, већи од тежине сваког појединачног слободног чвора (чија тежина у генералном случају као што је описано не мора имати вредност 1), и тај исти број се доследно мора користи као „замена“ за ∞ у свим операцијама алгоритма (нпр. код иницијализације). Уважавајући претходно наведено, може се рећи да генерисање путање најмање дужине представља у ствари специјални случај генерисања путање најмање цене.



Слика 12. Пример планирања путање робота коришћењем D Lite алгоритма без примењене интерполације (црвена путања) и са примењеном интерполацијом (плава путања). Стартна позиција је у доњем левом углу, а циљна у горњем десном. Тамнија боја ћелије илуструје њену већу тежину са аспекта проходности [49].*

Имплементација алгоритама *ARA**, *D* Lite* и *AD** у Матлабу ради истраживања у оквиру ове дисертације урађена је према псеудокодовима приказаним на *Слици 5*, *Слици 8*. и *Слици 10*, који су као такви дати у оригиналном раду у ком је први пут презентован алгоритам *AD** [32], као и према одговарајућим дијаграмима токова (*Слика 6*, *Слика 9*. и *Слика 11*), урађеним у склопу истраживања обухваћеног овом дисертацијом. При томе су за сваки алгоритам урађене две варијанте кода које се разликују у томе што за процесуирање промене статуса блокираних чворова у слободне чворове и обрнуто при обради графа користе различите, претходно објашњене приступе. Независно од примењене варијанте алгоритми дају исте резултате, с тим да је код у Матлабу компликованији и има више линија у случају примене првог приступа, али због уведених ограничења при прорачуну путање обрађује мањи број чворова.

За потребе визуелног објашњења функционисања алгоритама *ARA**, *D* Lite* и *AD** у овом делу текста применићемо приступ за процесуирање промене статуса блокираних чворова у слободне чворове и обрнуто при обради графа предложен у [50], имајући у виду да је визуелна прегледност корака алгоритма у овом случају боља (процесуира се мањи број чворова) и лакше је фокусирати се на специфичне моменте у раду алгоритма. За тежине слободних ћелија дефинисана је вредност 1, тако да цене прелазака које подразумевају ортогонални померај између слободних чворова у графу имају вредност 1, док оне које подразумевају дијагонални померај имају вредност √2.

2.4.2. Пример функционисања *D* Lite* алгоритма

Пошто се у овој дисертацији *D* Lite* користи као примарни алгоритам за планирање путање робота, изложићемо детаљан пример његове примене, урађен у склопу истраживања обухваћених дисертацијом (*Слика 13*).

Стартна позиција робота је у ћелији мапе означеној са (3,1), а циљна у ћелији (1,5), односно у одговарајућим чворовима замишљеног графа лоцираним у центрима наведених ћелија. Прорачун путање врши се применом *D* Lite* алгоритма, на начин објашњен у *Поглављу 2.3.1.* и према псеудокоду приказаном на *Слици 8*. У оквиру иницијализације (корак 2.) параметрима *g* и *rhs* стартног чвора се додељује вредност ∞. Иста вредност се додељује и параметру *g* циљног чвора, док његов параметар *rhs* добија вредност нула. Такође се у скуп *OPEN* инсертује циљни чвор.

Након иницијализације врши се наизменично ширење чворова из скупа *OPEN* (у складу са извршеном приоритизацијом) и ажурирање тог скупа. У склопу ширења сваког чвора приказано је ажурирање његовог параметра *g* у зависности од типа инконзистенције (линије 10. и 13. псеудокода), као засебан корак, а као пратећи корак или пратећи кораци приказани су процеси ажурирања параметара и статуса околних чворова (уколико је чвор подконзистентан врши се и ажурирање параметра и статуса и тог чвора) (линије 11. и 14. псеудокода). Наравно, проширени чвор се брише из скупа *OPEN* што неће бити посебно наглашавано. Такође, неће бити посебно наглашавано (сем у специфичним случајевима) да ажурирање параметра *rhs* подразумева и ажурирање показивача. Први се шири циљни чвор што је приказано у корацима 3. и 4., имајући у виду да је он у склопу иницијализације једини инсертован у скуп *OPEN*. У кораку 3. приказано је ажурирање параметра *g* циљног чвора (изједначава се са параметром *rhs* и добија вредност 0 пошто циљни чвор испуњава услов прекоконзистентности), док је процес ажурирања параметара и статуса околних чворова приказан у кораку 4. У овом кораку, према псеудокоду *D* Lite* алгоритма и конкретној ситуацији, параметри *g* и *rhs* чворова (1,4) и (2,5) добијају вредност ∞, односно 1, респективно, и инсертују се у скуп *OPEN*, пошто задовољавају

услов инконзистентности. Као што је претходно наведено, примењен је приступ према којем алгоритам у току рада „игнорише“ (не процесуира) блокиране чворове, пошто није могућа путања која их садржи, тако да ови чворови постају „видљиви“ тек када се деблокирају и, наравно, када се та промена детектује сензором робота. У кораку 5. започиње процес ширења чвора (2,5), када се његов параметар *g* изједначава са параметром *rhs* (добија вредност 1). Ажурирања параметара и статуса околних чворова код којих је дошло до промене приказано је у кораку 6., где параметри *g* чворова (3,4) и (3,5) добијају вредност ∞, а параметри *rhs* вредности 2.4, односно 2, респективно, и оба чвора се инсертују у скуп *OPEN*. Корак 7. приказује ширење чвора (1,4), тј. његов параметар *g* се изједначава са параметром *rhs* (добија вредност 1). При провери параметара и статуса његових суседа нема промена (тј. нема потребе за ажурирањем) тако да ти кораци нису ни приказани.

У кораку 8. започиње процес ширења чвора (3,5), када се његов параметар *g* изједначава са параметром *rhs* и добија вредност 2. Ажурирања параметара и статуса околних чворова код којих је дошло до промене приказано је у кораку 9., где параметри *g* чворова (4,4) и (4,5) добијају вредност ∞, а параметри *rhs* вредности 3.4, односно 3, респективно, и оба чвора се инсертују у скуп *OPEN*. Код осталих суседа није било промена. Процес ширења чвора (3,4), када се његов параметар *g* изједначава са параметром *rhs* и добија вредност 2.4 започиње у кораку 10. У кораку 11. приказани су суседи чвора (3,4) код којих је при провери параметара и статуса дошло до ажурирања, а то су чворови (3,3) и (4,3), при чему њихови параметри *g* добијају вредност ∞, а параметри *rhs* вредности 3.4, односно 3.8, респективно, и оба чвора се инсертују у скуп *OPEN*. У корацима 12. и 13. приказано је ширење чворова (4,5) и (4,4) када се њихови параметри *g* изједначавају са параметрима *rhs* и добијају вредности 3, односно 3.4 респективно. У оба случаја промена код суседних чворова није било. У кораку 14. започиње процес ширења чвора (3,3), када се његов параметар *g* изједначава са параметром *rhs* и добија вредност 3.4. У кораку 15. приказано је ажурирање параметара и статуса околних чворова код којих је дошло до промене, када параметри *g* чворова (2,2), (3,2) и (4,2) добијају вредности ∞, а параметри *rhs* вредности 4.8, 4.4 и 4.8, респективно, и сва три чвора се инсертују у скуп *OPEN*. Корак 16. представља ширење чвора (4,3), при чему се његов параметар *g* изједначава са параметром *rhs* и добија вредност 3.8. При провери параметара и статуса његових суседа нема промена, па нема ни потребе за ажурирањем, тако да ти кораци нису ни приказани. Процес ширења чвора (3,2), при којем се његов параметар *g* изједначава са параметром *rhs* и добија вредност 4.4 започиње у кораку 17. У кораку 18. приказани су суседи чвора (3,2) код којих је при провери параметара и статуса дошло до ажурирања, а то су чворови (2,1), (3,1) и (4,1), при чему њихови параметри *g* добијају вредност ∞, а параметри *rhs* вредности 5.8, 5.4 и 5.8, респективно, и сва три чвора се инсертују у скуп *OPEN*. У кораку 19. приказано је ширење чвора (4,2), при чему се његов параметар *g* изједначава са параметром *rhs* и добија вредност 4.8. Промена код суседних чворова није било. Корак 20. представља започињање процеса ширења чвора (2,2), када се његов параметар *g* изједначава са параметром *rhs* и добија вредност 4.8. Ажурирање параметара и статуса околних чворова код којих је дошло до промене приказано је у кораку 21., када параметри *g* чворова (1,1) и (1,2) добијају вредности ∞, а параметри *rhs* вредности 6.2 и 5.8, респективно, и оба чвора се инсертују у скуп *OPEN*.

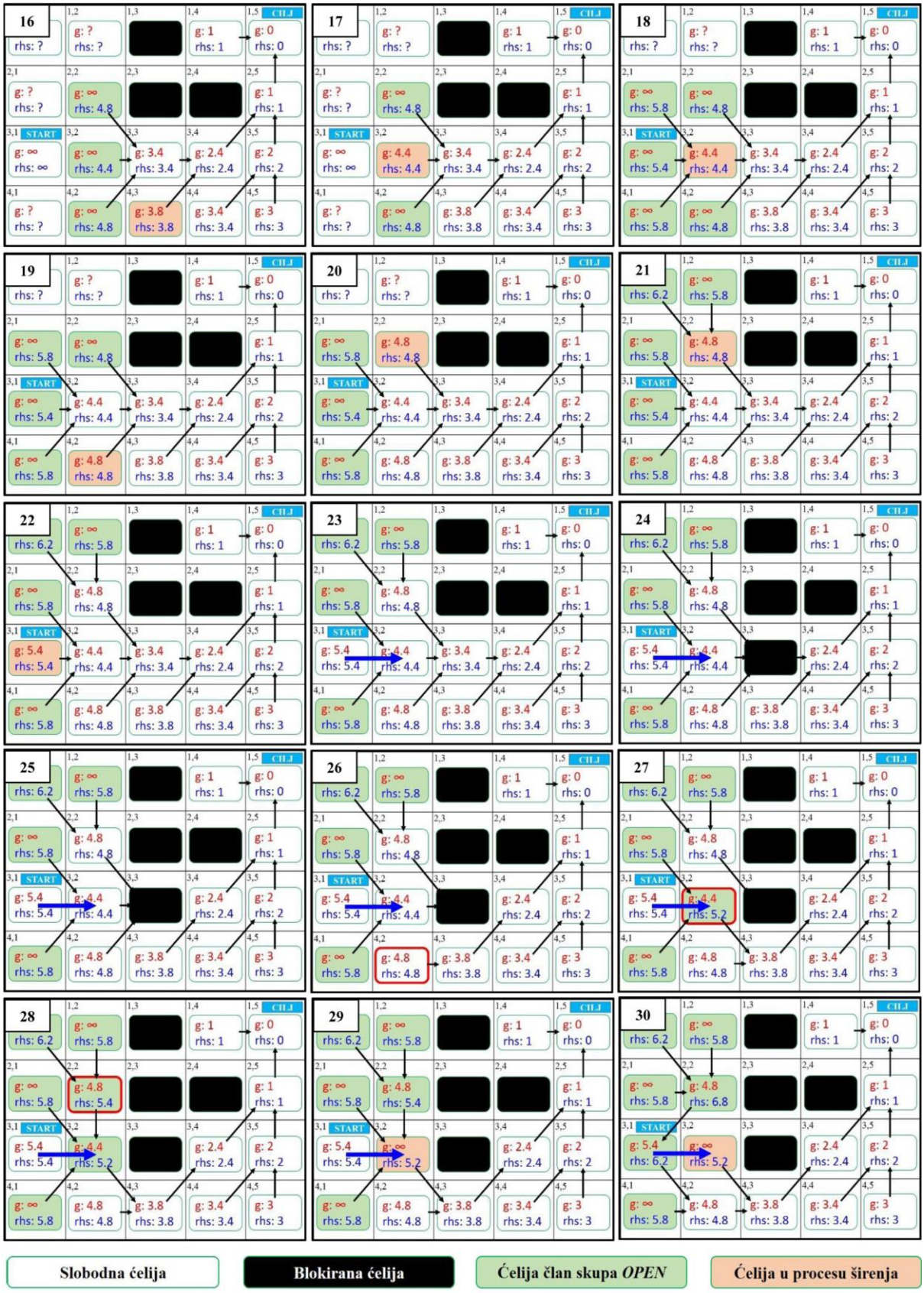
У кораку 22. се завршава процес израчунавања иницијалне путање тј. проширен је стартни чвор, при чему се његов параметар *g* изједначава са параметром *rhs* и добија вредност 5.4 (промена код суседних чворова није било), након чега робот започиње кретање. Робот се помера из ћелије (3,1) у ћелију (3,2) (корак 23.). Из нове позиције робот врши осматрање окружења (корак 24.) и тада детектује прву промену, која се огледа у томе да је ћелија (3,3) која је претходно била слободна сада блокирана. У складу са

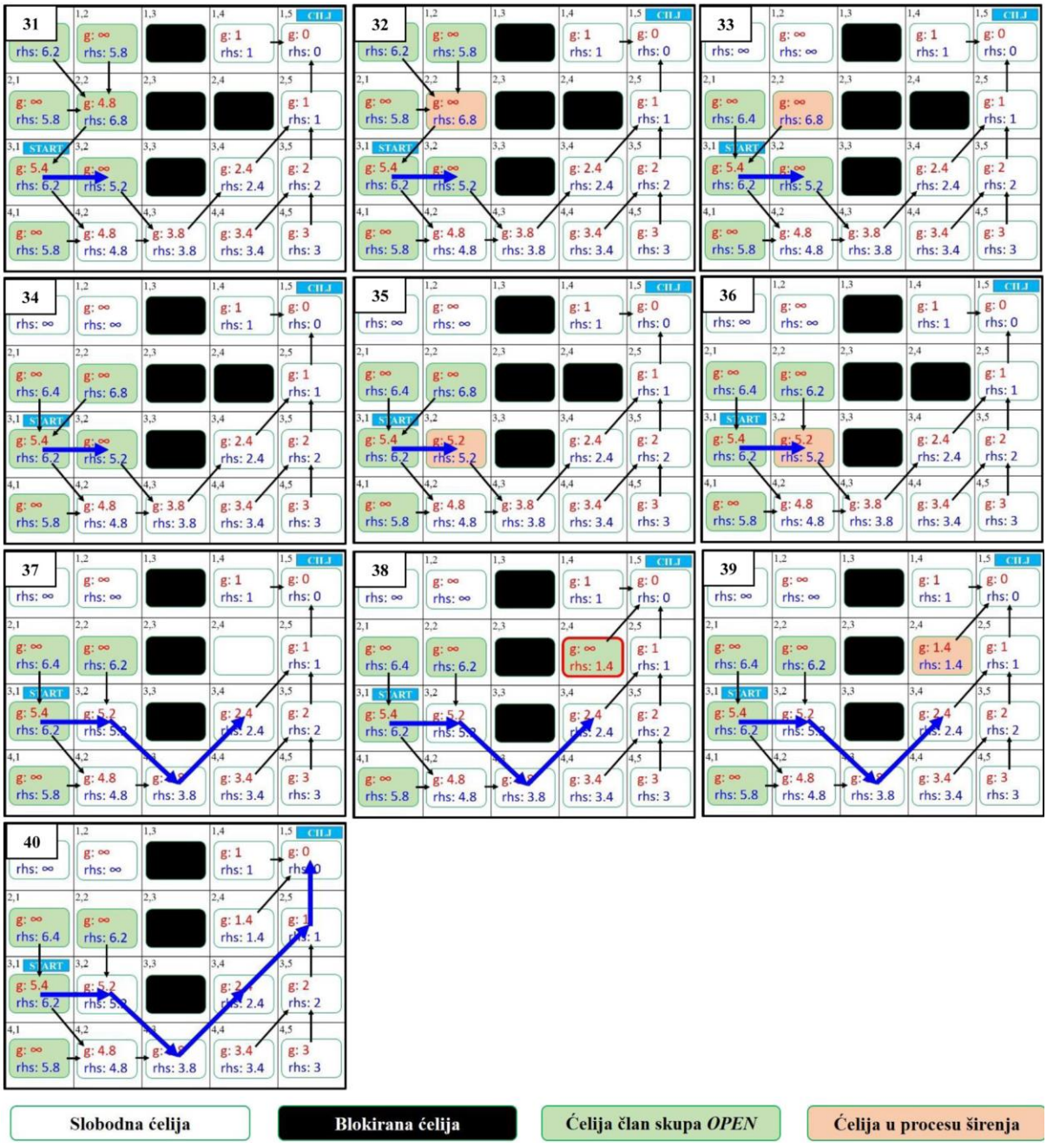
претходно објашњеном процедуром алгоритам проверава да ли је одговарајући чвор члан скупа *OPEN* и уколико јесте брише га из тог скупа (у овом случају није), а такође се поништавају и његови параметри *g* и *rhs*, као и показивач (корак 25.). Даље се, ради корекције путање, а у складу са псеудокодом *D* Lite* алгоритма (линије 22. и 23.), врши ажурирање параметара и статуса околних чворова, при чему су на *Слици 13*. приказани само кораци у којима је дошло до неке промене (кораци 26. - 28.). Пошто се ради о специфичним моментима који представљају битну карактеристику *D* Lite* алгоритма, црвеним оквиром су наглашени одговарајући чворови, тачније ћелије (код којих је дошло до промене). Приметимо да је при ажурирању параметра *rhs* чвора (4,2) добијена иста вредност (корак 26.), тако да није дошло до нарушавања конзистенције тог чвора, али је промењен његов показивач, јер је вредност параметра *rhs* сада изведена у односу на чвор (4,3), уместо у односу на чвор (3,3), како је било у претходном случају. Са друге стране, у току овог процеса чворови (3,2) и (2,2) су постали инконзистентни, јер им се повећава вредност параметара *rhs* на 5.2, односно 5.4, респективно, (вредност *g* се не мења), због чега се инсертују у скуп *OPEN* (кораци 27. и 28.). У наставку долази до ширења чворова из ажурираног скупа *OPEN* како би се комплетирао процес корекције путање.

Приметимо да у корацима 29. – 31. имамо први случај ширења подконзистентног чвора тј. чвора код кога је вредност параметра *g* мања од вредности параметра *rhs* (претходно су сви проширени чворови били прекоконзистентни). У питању је чвор (3,2) и у процесу ширења најпре се његовом параметру *g* додељује вредност ∞ (корак 29.). У кораку 30. врши се ажурирање параметара и статуса околних чворова, када се чвору (2,2) мења вредност параметра *rhs* са 5.4 на 6.8, али остаје инконзистентан (истовремено се ажурира и показивач), чвору (2,1) се мења само показивач пошто је при ажурирању његовог параметра *rhs* добијена иста вредност, при чему је сада она изведена у односу на чвор (2,2), уместо у односу на чвор (3,2), како је било у претходној ситуацији. Слична промена као код чвора (2,1) дешава се код чвор (4,1), код којег се такође при ажурирању не мења вредност параметра *rhs* чвора, али је сада она изведена у односу на чвор (4,2), уместо у односу на чвор (3,2). У овом кораку се такође ажурира параметар *rhs* чвора (3,1), који овом променом постаје инконзистентан (претходно је био конзистентан) и инсертује се у скуп *OPEN*. У кораку 31. врши се ажурирање параметара и статуса и самог чвора (3,2), имајући у виду да то подразумева процедура ширења подконзистентног чвора, при чему није дошло до промене вредности његових параметара *g* и *rhs*, а такође је остао члан скупа *OPEN*. Други случај ширења подконзистентног чвора имамо у корацима 32. – 34., сада је у питању чвор (2,2). Најпре се његовом параметру *g* додељује вредност ∞ (корак 32.). Корак 33. приказује ажурирање параметара и статуса околних чворова, када се параметрима *rhs* чворова (1,1) и (1,2) додељује вредност ∞, јер су окружени блокираним чворовима и чворовима чији параметар *g* такође има вредност ∞. Овим чворови (1,1) и (1,2) постају конзистентни и бришу се из скупа *OPEN*. Промене има још код чвора (2,1), чији параметар *rhs* се ажурира и добија вредност 6.4 (ову промену прати и ажурирање показивача који је сада усмерен према чвору (3,1)), али он остаје инконзистентан тј. члан скупа *OPEN*. Корак 34. приказује проверу параметара и статуса и самог чвора (2,2), при чему није дошло до промене вредности његових параметара *g* и *rhs*, а такође је остао члан скупа *OPEN*. У корацима 35. и 36. шири се последњи чвор из скупа *OPEN* који има услов за ширење чиме се завршава процес корекције путање. Ради се о чвору (3,2) који се сада шири као прекоконзистентан, због чега се његов параметар *g* изједначава са параметром *rhs* и добија вредност 5.2 (корак 35.). Ажурирање параметара и статуса околних чворова код којих је дошло до промене приказано је у кораку 36., када се мења вредност параметра *rhs* чвора (2,2) са 6.8 на 6.2 (ову промену прати и ажурирање показивача који је сада усмерен према чвору (3,2)), при чему је остао члан скупа *OPEN*.

Пратећи кориговану путању робот прелази из ћелије (3,2) у ћелију (4,3), па затим из ћелије (4,3) у ћелију (3,4) што је приказано у кораку 37. Из позиције (3,4) детектује другу промену у окружењу, која се огледа у томе да је ћелија (2,4) која је претходно била блокирана сада слободна. Сада се ажурирају параметри и статуси чвора (2,4), као и околних чворова, при чему је на *Слици 13*. приказан само корак у којем је дошло до неке промене (корак 38.). У овом кораку параметрима *g* и *rhs* чвора (2,4) се додељују вредности ∞, односно 1.4, респективно и инсертује се у скуп *OPEN*. У кораку 39. приказано је ширење чвора (2,4), када се његов параметар *g* изједначава са параметром *rhs* и добија вредност 1.4, након чега долази до терминације рада алгоритма, јер ниједан други чвор из скупа *OPEN* нема услов за ширење. Приметимо да ширење чвора (2,4) није довело до корекције путање, тако да је претходно израчуната путања остала валидна. Настављајући кретање робот прелази из ћелије (3,4) у ћелију (2,5), затим из ћелије (2,5) у ћелију (1,5), која је уједно и циљна позиција.

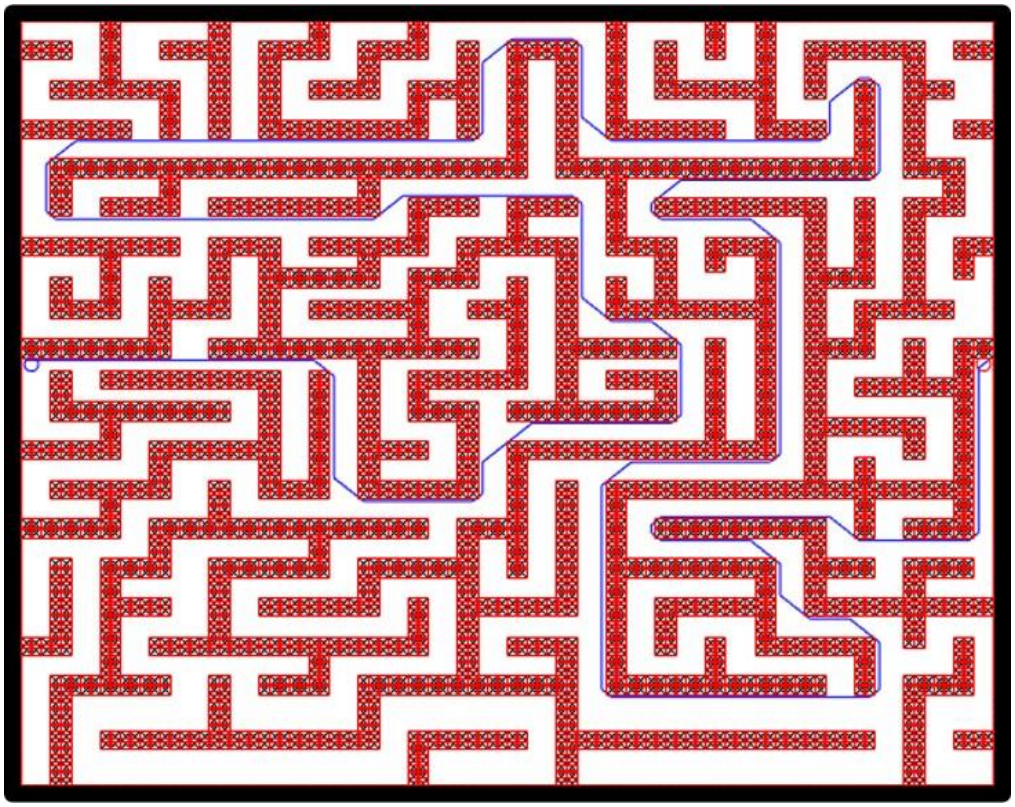

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
START
CILJ
Slobodna ćelija
Blokirana ćelija
Ćelija član skupa OPEN
Ćelija u procesu širenja


16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
Slobodna ćelija
Blokirana ćelija
Ćelija član skupa OPEN
Ćelija u procesu širenja



Слика 13. Пример планирања путање робота применом D Lite алгоритма са детаљним описом корака функционисања, урађен у склопу истраживања обухваћених дисертацијом.*

На *Слици 14*. приказано је тестирање кода *D* Lite* алгоритма, урађеног у склопу истраживања обухваћеног овом дисертацијом, на примеру решавања лавиринта.



Слика 14. Провера функционисања кода D Lite алгоритма на примеру решавања лавиринта, урађен у склопу истраживања обухваћених дисертацијом. Старт је означен црвеним кружићем, а циљ плавим кружићем.*

2.4.3. Поређење функционисања алгоритама *ARA*, D* Lite* и *AD**

У овом поглављу је презентовано поређење функционисања алгоритама *ARA*, D* Lite* и *AD**, на основу резултата добијених у Матлабу, коришћењем кодова за наведене алгоритме (урађених у склопу истраживања обухваћених овом дисертацијом) за решавање симулираних сценарија. На сликама које визуелизују добијене резултате коришћена је симбологија из *Табеле 1*.

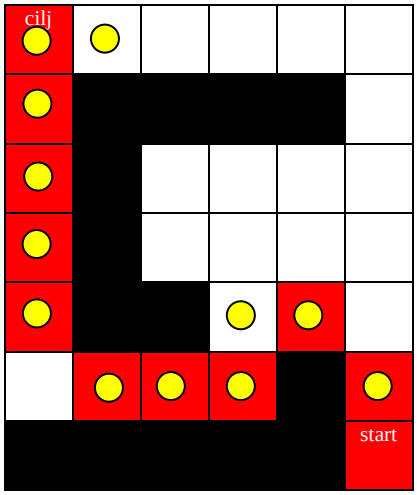
Табела 1. Значење симбола у опису сценарија поређења функционисања алгоритама ARA, D* Lite и AD*.*

Симбол	Значење
	Чвор са препреком (блокиран чвор)
	Слободан чвор
	Чвор преко којег иде путања
	Проширени чвор

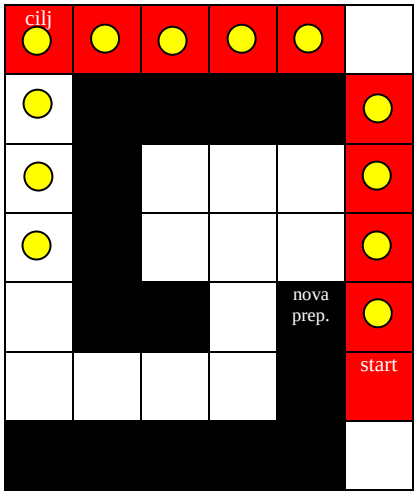
На сликама нису приказани показивачи, имајући у виду да је њихова функција презентована у претходном поглављу. Ћелијама мапе се додељује нумерација где први број означава број реда „одозго на доле“, а други број означава број колоне „слева на десно“.

Први сценарио се односи на приказ функционисања и поређење *ARA** и *AD**, који користе хеуристичку функцију пондерисану тежинским фактором $\varepsilon>1$, и то у свим итерацијама почев од $\varepsilon=\varepsilon_0=3$ до $\varepsilon=1$, укључујући истовремену појаву нове блокиране ћелије током кретања робота. Дакле, код оба алгоритма ε се декрементира из итерације у итерацију у циљу побољшања решења све док се не дође до $\varepsilon=1$. Декрементирање ε се ради и у случају појаве промена у окружењу. Стартна позиција робота је у ћелији мапе (7,6), а циљна у ћелији (1,1). У датом примеру робот, крећући се по иницијално генерисаној путањи, долази у позицију (6,6) где детектује нову препреку (5,5). У тој ситуацији *ARA** алгоритам врши израчунавање нове путање из почетка (за актуелну позицију робота), док *AD** алгоритам врши корекцију актуелне путање. Пошто се нове препреке детектују на почетку друге итерације алгоритма, дошло је истовремено и до декрементирања фактора ε са вредности 3 на вредност 2 код оба алгоритма. Такође је презентовано како овај сценарио решава *D* Lite* алгоритам, који не користи пондерисану хеуристичку функцију. На *Слици 15*. визуелизовани су резултати симулације у Матлабу.

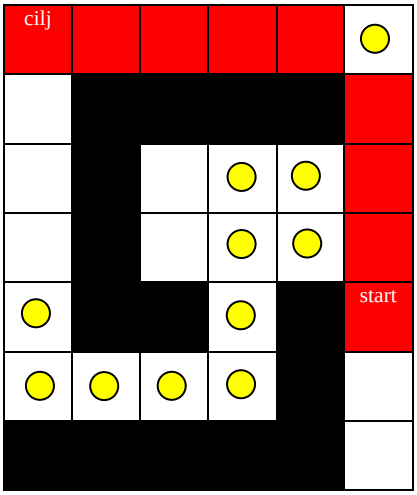
*ARA**



(I iteracija, *eps*=3)

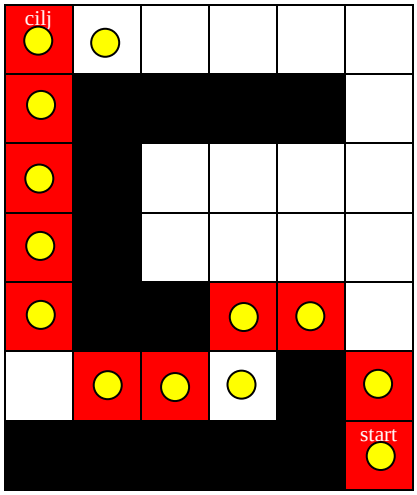


(II iteracija, *eps*=2)

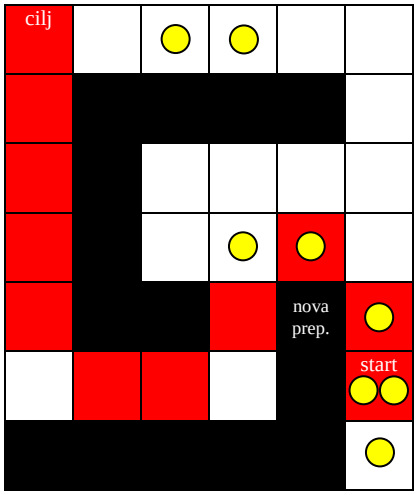


(III iteracija, *eps*=1)

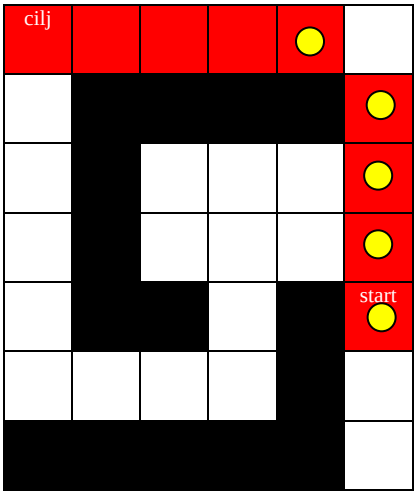
*AD**



(I iteracija, *eps*=3)

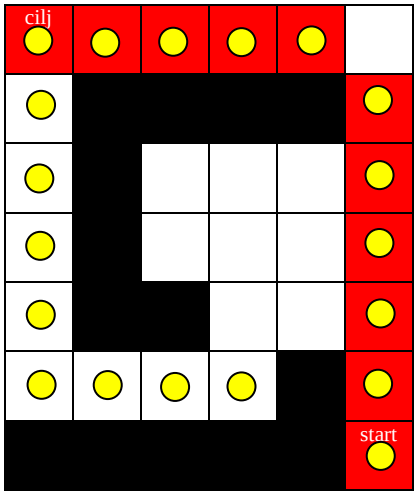


(II iteracija, *eps*=2)

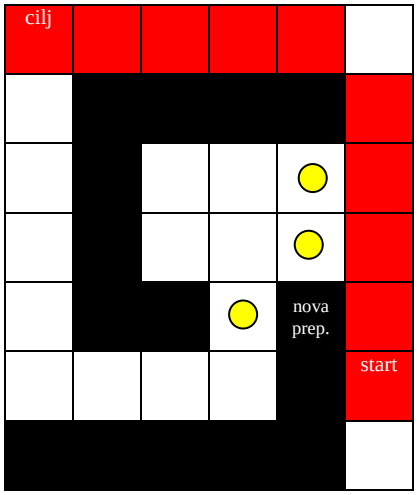


(III iteracija, *eps*=1)

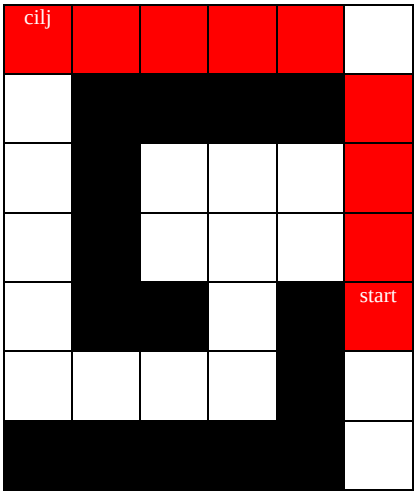
D Lite*



(I iteracija, eps_0=*eps*=1)



(II iteracija, eps_0=*eps*=1)



(III iteracija, eps_0=*eps*=1)

Слика 15. Први сценарио поређења функционисања алгоритама за планирање путање ARA, D* Lite и AD* (урађен у склопу истраживања обухваћених дисертацијом), којег карактерише детекција нове блокиране ћелије.*

Резултати симулације у смислу броја проширених чворова за сваки алгоритма и пратећи коментари дати су у *Табели 2*.

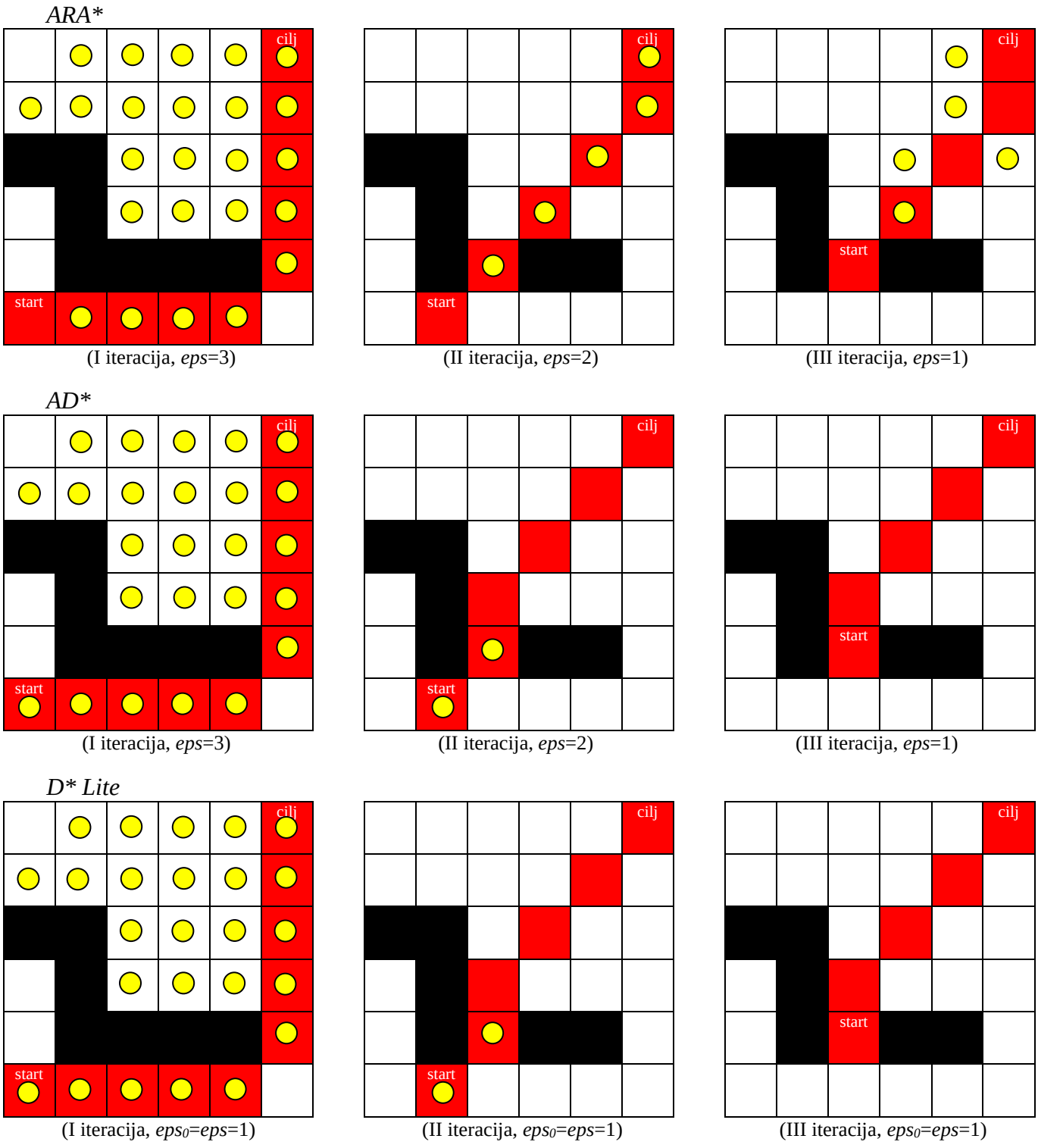
Табела 2. Резултати поређења функционисања алгоритама ARA, D* Lite и AD* у првом сценарију.*

Iteracija	Broj proširenih čvorova			Komentar
	ARA*	***AD****	***D* Lite***	
I iteracija eps_0=3 kod *ARA** i *AD**, kod *D* Lite* eps_0=*eps*=1	12	13	19	Poredeći u ovoj iteraciji rezultate *ARA** i *AD** (eps_0=3) sa jedne strane i *D* Lite* (eps_0=1) sa druge strane može se primetiti da je za eps_0>1 potrebno značajno manje proširenih čvorova da bi se došlo do inicijalnog rešenja.
II iteracija *eps*=2 kod *ARA** i *AD**, kod *D* Lite eps*=1 (detektuje se jedna promena u okruženju - novi čvor sa preprekom)	12	8	3	U ovoj iteraciji *ARA** algoritam vrši izračunavanje nove putanje iz početka (za aktuelnu poziciju robota), dok *D* Lite* i *AD** algoritam vrše korekciju aktuelne putanje uz korišćenje rezultata prethodno izvršenog proračuna. Istovremeno se kod *ARA** i *AD** dekrementira *eps*.
III iteracija *eps*=1	11	5	-	Velika razlika broja proširenih čvorova u ovoj iteraciji između *ARA** i *AD**, iako nije detektovana nova prepreka, je zbog toga što *AD** algoritam koristi rezultate iz obe prethodne iteracije, dok je *ARA** algoritam resetovan na početku druge iteracije zbog detekcije nove prepreke, tako da se gube rezultati proračuna iz prve iteracije. Razlika broja proširenih čvorova između *AD** i *D* Lite* je zbog toga što AD* popravlja rešenje, imajući u vidu da je došlo do dekrementiranja *eps*.
UKUPNO:	**35**	**26**	**22**	

Poredeći *ARA** i *AD** vidimo da su oba algoritma uspešno rešila postavljeni test scenario, kao i da su došla do istih rešenja, ali za različito vreme tj. broj proširenih čvorova. *ARA** algoritmu je bilo potrebno da proširi ukupno 35 čvorova, a *AD** algoritmu 26 čvorova tj. 9 čvorova manje. Ovde je bitno istaći jedno karakteristično zapažanje, a to je da je u okviru druge iteracije kod *AD** algoritma čvor (6,6) proširen dva puta, prvi put kao podkonzistentan, a drugi put kao prekokonzistentan. Naime, krećući se po inicijalno generisanoj putanji robot dolazi u poziciju (6,6) gde detektuje novu prepreku (5,5). S obzirom da je inicijalna putanja upravo išla preko čvora (5,5), *AD** algoritam nastali problem rešava tako što najpre vrši ažuriranje „pogođenih" cena prelazaka, parametara *rhs* odgovarajućih čvorova, kao i skupa *OPEN* u koji se insertuju novonastali inkonzistentni čvorove. Imajući u vidu da je parametar *rhs* čvora (6,6) upravo izračunat u odnosu na čvor (5,5), procesuiranje nastale promene dovešće do promene vrednosti *rhs* čvora (6,6) jer se sada ta vrednost računa u odnosu na čvor (5,6). Ova promena u konkretnom slučaju dovodi do povećanja vrednosti *rhs* čvora (6,6) što reflektuje povećanja cene putanje izazvano zaobilaženjem čvora (5,5) (to je jednostavno zaključiti na osnovu *Slike 15*). Ne treba zaboravit da se ažurira i pokazivač čvora (6,6)). Ova promena čvor (6,6) čini podkonzistentnim (g(6,6) < *rhs*(6,6)) i insertuje ga u skup *OPEN*. На овај начин се у ствари на чвор (6,6) шири повећање цене путање настало услед појаве нове препреке. Пошто је сада чвор (6,6) у скупу *OPEN* долази до његовог првог ширења када се параметар g(6,6) сетује на ∞, како би се подконзистенција даље пренела на суседе чвора (6,6). У даљем току алгоритма долази до другог ширења чвора (6,6) када се он квалификује као прекоконзистентан и као такав процесуира како би се процес корекције путање комплетирао. Доказ да се један исти чвор може проширити највише два пута (највише једном као подконзистентан и највише

једном као прекоконзистентан) у току израчунавања путање тј. извршавања функције *Computepath* дат је у [52,53].

Други сценарио подразумева детекцију нове слободне ћелије у току кретања робота на позицији (5,3), која је претходно била блокирана. Резултати симулације у Матлабу визуелизовани су на *Слици 16*.



Слика 16. Други сценарио поређења функционисања алгоритама за планирање путање ARA, D* Lite и AD* (урађен у склопу истраживања обухваћених дисертацијом), којег карактерише детекција нове слободне ћелије.*

Резултати симулације у смислу броја проширених чворова за сваки алгоритма и пратећи коментари дати су у *Табели 3.*

Табела 3. Резултати поређења функционисања алгоритама ARA, D* Lite и AD* у другом сценарију.*

Iteracija	**Broj proširenih čvorova**			**Komentar**
	ARA*	***AD****	***D* Lite***	
I iteracija eps_0=3 kod *ARA** i *AD**, kod *D* Lite* eps_0=*eps*=1	24	25	25	Ovde je scenario takav da su i *AD** za eps_0=3 i *D* Lite* za eps_0=*eps*=1 došli do istog inicijalnog rešenja, kao i da su pri tome proširili isti broj čvorova.
II iteracija *eps*=2 kod *ARA** i *AD**, kod *D* Lite eps*=1 (detektuje se jedna promena u okruženju – novi slobodan čvor)	5	2	2	U ovoj iteraciji *ARA** algoritam vrši izračunavanje nove putanje iz početka (za aktuelnu poziciju robota), dok *D* Lite* i *AD** algoritam vrše korekciju aktuelne putanje uz korišćenje rezultata prethodno izvršenog proračuna. Istovremeno se kod *ARA** i *AD** dekrementira *eps*.
III iteracija *eps*=1	5	0	0	Razlika broja proširenih čvorova u ovoj iteraciji između *ARA** sa jedne i *D* Lite* i *AD** sa druge strane, (iako nije detektovana nova prepreka), predstavlja „produženu posledicu“ činjenice da je *ARA** algoritam resetovan na početku druge iteracije zbog detekcije nove prepreke, tako da se gube rezultati proračuna iz prve iteracije.
УКУПНО:	**34**	**27**	**27**	

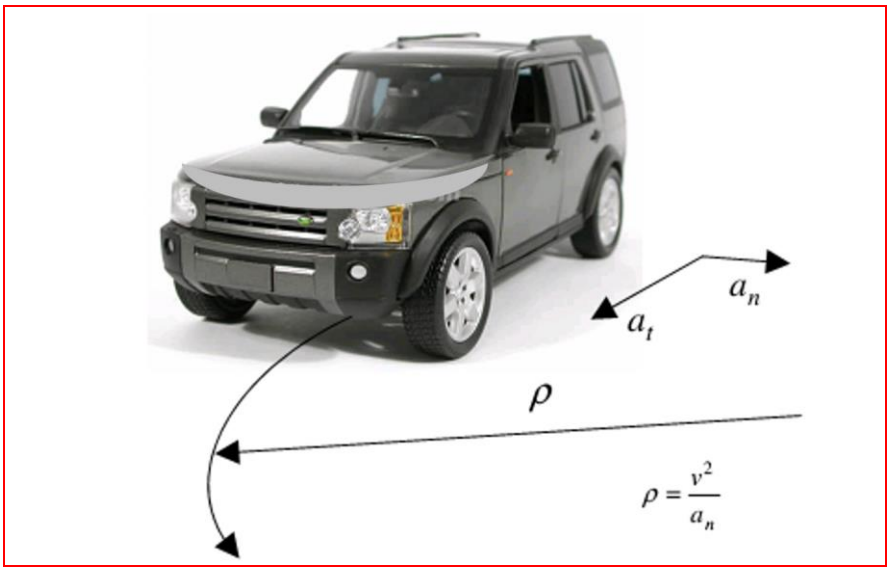
Сва три алгоритма су успешно решила постављени тест сценарио, али за различито време тј. број проширених чворова. *ARA** алгоритму је било потребно да прошири укупно 34 чвора, а алгоритмима *AD** и *D* Lite* 27 чворова тј. 7 чворова мање.

2.4.4. Осврт на неке од варијација алгоритама *D* Lite* и *AD**

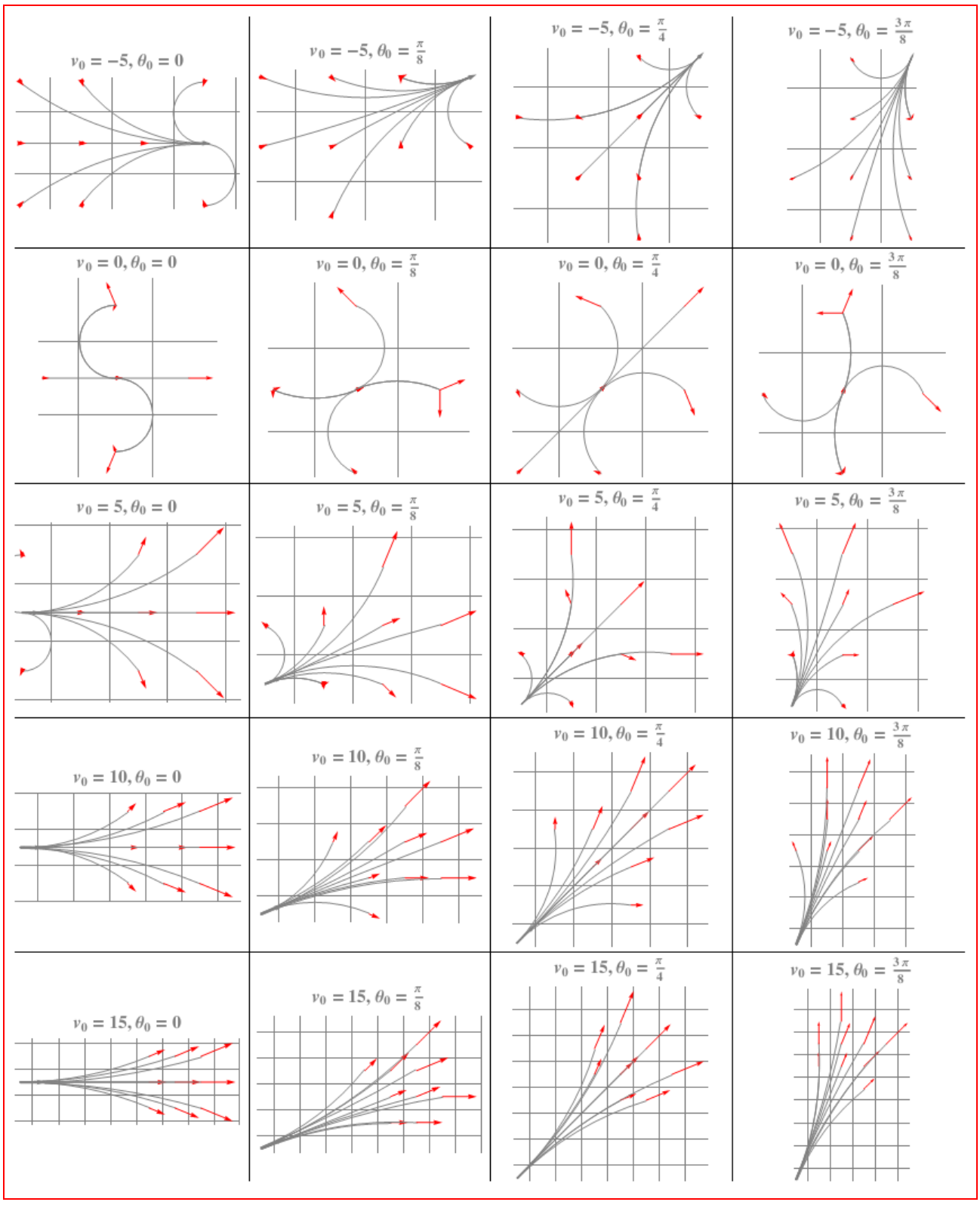
Већина алгоритама за планирање путање у роботици који су базирани на обради *2D* графа подразумева дискретизован инкремент заокрета возила (ако је чвор повезан са свих 8 суседних чворова онда је минимални инкремент заокрета у односу на правац кретања $\pi/4$), тако да иако са математичког аспекта оптимални ови алгоритми генеришу путању која је неприродна и субоптимална са аспекта кретања робота. Због наведеног се у литератури као варијанте алгоритама *D* Lite* и *AD** предлажу решења која инкорпорирају у поступак прорачуна путање и динамику возила. На пример, у [32] се предлаже решење које се базира на, условно речено, обради *4D* графа, где је сваки чвор одређен следећим параметрима: позиција (x,y), оријентација возила (θ) и брзина возила (υ). Задатак алгоритма у овом случају је да генерише оптималну путању узимајући у обзир динамичка ограничења робота. За разлику од обраде *2D* графа која генерише путању која може имати оштре заокрете (нпр. 90°) и где робот који се креће оваквом путањом мора успоравати и свести брзину на $\upsilon \approx 0$ како би направио заокрет, обрада *4D* графа генерише путању са „ширим" и природнијим заокретима тако да се просечна брзина робота у овом случају може држати на значајно вишем нивоу него у случају обраде *2D* графа. За сваки случај у обе варијанте обично се дефинише бафер-зона око препрека са високом ценом уласка како би се осигурало безбедно кретање. Као што се може претпоставити, повећање димензионалности планирања путање подразумева значајно више времена потребног за прорачун, што посебно долази до изражаја на простору веће површине. Једна од могућности за имплементацију *4D* планера је тзв. планер са 2 нивоа где се комбинује обрада *4D* графа коју извршава *AD** алгоритам и обрада *2D* графа исте основе коју извршава *A** алгоритам и чији резултат се користи као хеуристичка функција код обраде *4D* графа. Обрада *4D* графа се у овом случају врши од циљног према стартном чвору (стартни чвор одговара почетној позицији робота у посматраној итерацији), док се обрада *2D* графа врши од стартног према циљном чвору. На овај начин *4D* планер не мора да врши прорачун путање од почетка сваки пут кад се робот помери. *2D* планер са друге стране је доста бржи и може генерисати решење обрађујући граф од стартног према циљном чвору сваки пут када се робот помери без уношења кашњења. Дакле, *4D* планери представљају добру основу за оптимизацију планирања путање, али ипак имају више недостатака који се посебно рефлектују код примене у случају окружења већих димензија. У наставку ће бити описан у кратким цртама један сценарио који илуструје обраду *4D* графа [54].

Претпоставимо да се ради о простору димензија 60m x 60m, као и да су ограничења возила таква да се може кретати максималном брзином од 15m/s напред и 5m/s назад. Како би дефинисали стратегију управљања возилом, претпоставимо да на сваком сегменту путање тангенцијално и латерално (бочно) убрзање могу бити нула или имати максималну вредност (позитивну или негативну). Ако уведемо ограничење оба убрзања на 5 m/s^2 , то значи да постоји 9 могућих комбинација убрзања за прелазак у следећу ћелију мапе {–5, 0, 5} x {–5, 0, 5}. Пошто возило има максималну брзину кретања напред или назад неке од ових комбинација се одбацују, редукујући на тај начин грањање при ширењу стабла путања. Инкремент прираста брзине одређује се на бази одговарајућег производа *временски интервал x тангенцијално убрзање*. У *4D* графу, сваки чвор је, као што је претходно наведено, одређен следећим параметрима: позиција (x,y), оријентација возила (θ) и брзина возила (υ). Цене преласка дефинишу се за сегменте путање који одговарају истом временском интервалу, тако да се овде не ради о преласку аутономног возила из тренутне у суседну ћелију (као код *2D* графа), већ у ћелију у зависности од сегмента путање најмање цене. У овом случају посматрамо сегменте путање који одговарају временском интервалу од 1 секунде у мрежи (x,y) резолуције 2.5m, промене угла

оријентације возила резолуције $\pi/8$ и промене брзине возила резолуције 5m/s, тако да анализирани *4D* граф има укупно око 46000 чворова (одговарајући *2D* граф има 576 чворова). Максимални могући угао заокрета рачуна се на бази средње брзине на посматраном сегменту путање (имамо податак о брзини на почетку и на крају сегмента) и узимајући у обзир динамичка ограничења возила. Тачније, минимални радијус заокрета тј. кривине зависи од брзине возила и максималног дозвољеног латералног (бочног) убрзања које зависи од динамичких ограничења возила и израчунава се као $\rho_{min}=v^2/a_{nmax}$ (*Слика 17*). Ово значи да већим средњим брзинама одговара већи минимални радијус заокрета односно мањи максимални угао скретања у односу на тренутну оријентацију возила. Што је већи угао скретања већа је цена преласка тј. цена одговарајућег сегмента путање, на тај начин се фаворизују комбинације са већом средњом брзином. На *Слици 18.* приказане су различите комбинације параметара који описују *4D* граф, где лукови сиве боје представљају на бази претходно описаног прорачуна изведене, а у складу са динамичким ограничењима возила изводљиве, сегменте путање за прелазак из тренутне у следећу ћелију. Поља са препреком су блокирана (теоретски им се додељује бесконачна цена уласка). Обрадом овако дефинисаног *4D* графа долази се до путање за дати сценарио која узима у обзир динамичка ограничења возила.



Слика 17. Упрошћен модел динамичких ограничења возила. Минимални радијус заокрета тј. кривине (максимални угао скретања у односу на тренутну оријентацију возила) зависи од брзине возила и максималног дозвољеног латералног (бочног) убрзања. Што је већи угао скретања већа је цена одговарајућег сегмента путање, на тај начин алгоритам за планирање фаворизује комбинације са већом средњом брзином [54].



Слика 18. Различите комбинације параметара који описују 4D граф, где лукови сиве боје представљају аутономним возилом изводљиве (у складу са његовим динамичким ограничењима) сегменте путање за прелазак из тренутне у следећу ћелију мапе. Једној истој колони одговара исти угао оријентације возила (θ), при чему се мења његова брзина (v_0). Што је веће v_0 мањи је максимални угао скретања возила у односу на његову тренутну оријентацију (θ) тако да је ужа „лепеза“ могућих наредних сегмената путање [54].

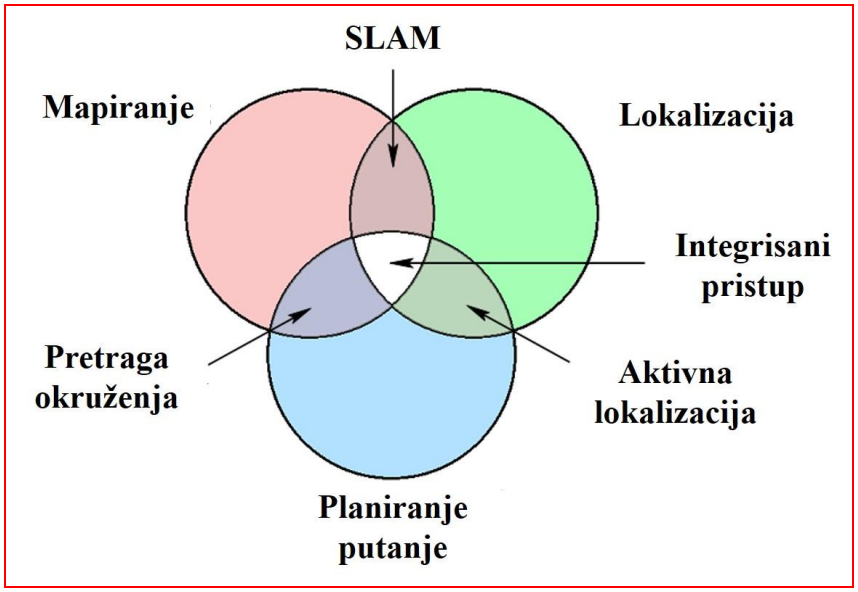
Field D* је још један у класи алгоритама дизајниран са намером да се превазиђу недостаци алгоритама базираних на обради 2D графа. Поменути алгоритам користећи линеарну интерполацију прорачунава одговарајућу путању која елиминише сувишне, велике и нагле заокрете возила. Може се стога условно рећи да Field D* до решења долази обрадом 3D графа чије димензије су: позиција возила (x,y) и оријентација возила (θ). Путања добијена овим поступком обраде је генерално глаткија него у случају обраде 2D графа. Иако веома погодан за практичну имплементацију алгоритам Field D* има неколико недостатака, од којих су кључни: 1) Field D* врши планирање и репланирање путање много спорије у односу на D* алгоритам; 2) Путања коју генерише Field D* није увек оптимална. Такође, Field D* алгоритам и поред тога што генерише природнију путању не узима у обзир динамичка ограничења возила. Оштри заокрети јесу минимизовани, али нису у потпуности елиминисани те стога ово решење није погодно за веће брзине. Ово решење се показало погодним за окружења са релативно ретким препрекама, док са друге стране има исте недостатке као и класични алгоритми за планирање путање базирани на обради графа када су у питању урбана окружења са већом густином препрека, као и када су у питању возила са израженијим динамичким ограничењима.

Уважавајући претходно наведено, један он најчешће примењиваних концепата за планирање путање робота подразумева комбинацију глобалног и локалног планера. Глобални планер прорачунава путању на бази обраде *2D* графа (*x, y*), при чему се игноришу динамичка ограничења возила/платформе. Након тога локални планер узимајући у обзир динамичка ограничења возила генерише одговарајућу локалну трајекторију. За глобални планер се веома често користи обичан *A** алгоритам или *D* Lite*. Такође, пошто се возило у реалности не може посматрати као тачкасти објекат, мора се применити адекватна стратегија за избегавање препрека. Најједноставнији начин је да се возило апроксимира са кругом одговарајућег пречника.

Имајући у виду да су капацитети данашњих рачунара далеко већи у односу на рачунаре из времена када је почео да се употребљава *A** алгоритам, па чак и у односу на рачунаре од пре нешто више од петнаестак година када су први пут презентовани алгоритми *ARA**, *D* Lite* и *AD**, то се сасвим сигурно може рећи да они данас представљају све мање ограничење за практичну примену ових алгоритама са становишта потребне рачунарске снаге и меморијских ресурса. Стога је сасвим оправдано изабрати *D* Lite*, као добру опцију алгоритма за планирање путање робота.

3. АУТОНОМНА ПРЕТРАГА ОКРУЖЕЊА МОБИЛНИМ РОБОТОМ

Аутономна претрага окружења је важна област изучавања у роботици. У литератури се најчешће дефинише као комбинација задатака мапирања простора тј. окружења у којем се креће робот и планирања путање робота [15], *Слика 19*.



Слика 19. Претрага окружења као комбинација задатака мапирања простора и планирања путање робота, модификована слика из [15].

Под мапирањем се овде подразумева интеграција података прикупљених сензорима робота у једну целину која представља верну репрезентацију одговарајућег дела окружења у виду мапе. Да би се формирала мапа целог окружења неопходно је да робот иде од једне до друге позиције (путањом коју је дефинисао неки планер) и да са тих позиција врши сукцесивно опсервацију окружења, прикупљање података и њихову интеграцију, све док се задатак не заврши. Како би се овај задатак обавио на рационалан начин неопходно је дефинисати одговарајућу стратегију претраге окружења. Пример који илуструје претрагу окружења у циљу формирања његове мапе приказан је на *Слици 20* [64].

Слика 20. Пример претраге окружења у циљу формирања његове мапе [64].

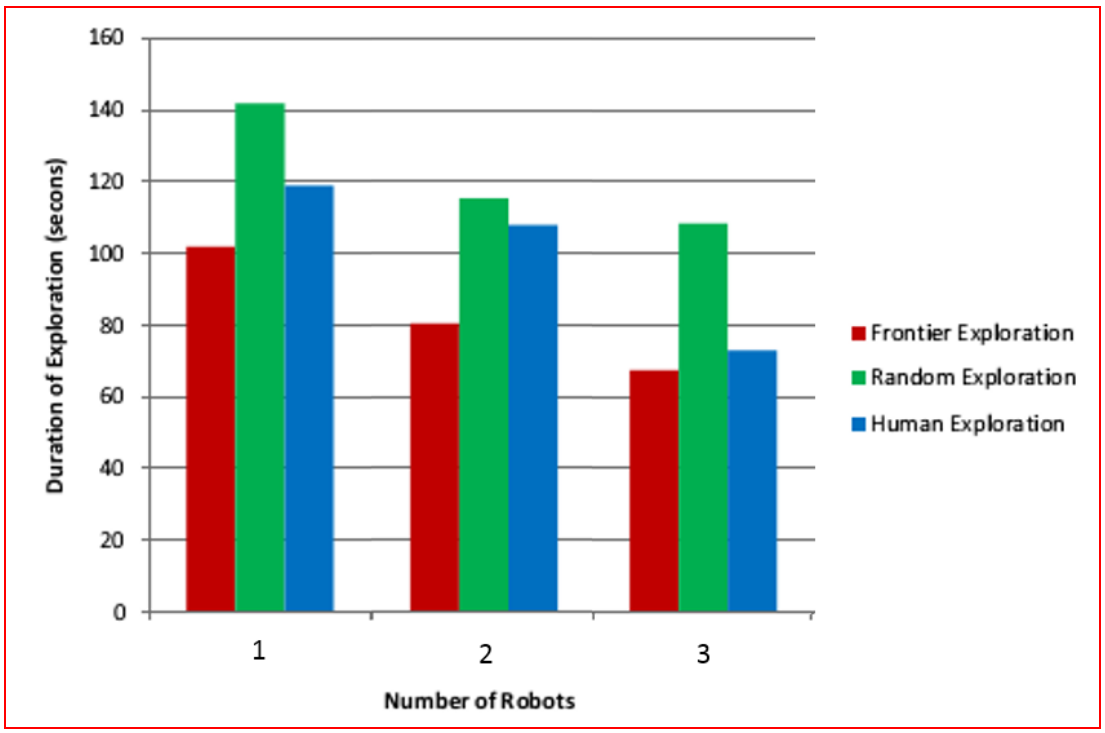
У ширем смислу, претрага окружења се не ограничава на формирање мапе окружења, већ се дефинише као скуп акција које предузима аутономни мобилни робот у складу са претходно дефинисаном стратегијом, са крајњим циљем откривања карактеристика окружења које су од интереса за задату мисију. Претрага окружења, стога, представља основу бројних примена роботике у пракси, као што су, поред формирања мапе окружења [1,2], трагање и спасавање [3], планетарне мисије [4], визуелне инспекције [5], рударство [6], роботски усисивачи [17], итд. Примера ради, код трагања и спасавања циљ мисије је лоцирање повређених лица или жртава природних непогода, пожара или неких других акцидената [12]. У пракси су честе ситуације да се паралелно или непосредно пре основних активности мисије мора урадити формирање мапе окружења [7], како би мисија уопште могла да се реализује. Слични сценарији претраге окружења са комбинованом реализацијом више врста задатака могу се срести и у другим применама роботике у пракси.

3.1. Приступи за реализацију претраге окружења

Претрага окружења може се у начелу класификовати у две групе у зависности од примењеног приступа реализације [8,9]. Први приступ подразумева претходно потпуно или делимично познавање окружења, на основу чега се користе *off-line* алгоритми за стратегију претраге. Код овог приступа путања робота се обично одређује унапред тј. пре саме мисије. Други приступ се примењује када је окружење потпуно непознато или када нема довољно информација за ефикасну примену *off-line* алгоритама. Претрага окружења је тада много изазовнија и подразумева *on-line* инкрементални принцип претраге. У овом случају робот након прикупљања података са актуелне позиције бира следећу позицију, премешта се на њу, извршава опсервацију окружења са нове позиције и понавља овај процес све док не испуни задатак или заврши мисију [10]. Главни проблем оваквог концепта претраге окружења је избор следеће позиције и планирање путање робота. Највећи број стратегија за претрагу окружења базира се на избору следеће позиције робота из скупа „позиција-кандидата“ које леже на граници између претражених и непретражених делова окружења, примењујући неке критеријуме за њихову евалуацију [10,12] и то су тзв. *frontier-based* стратегије.

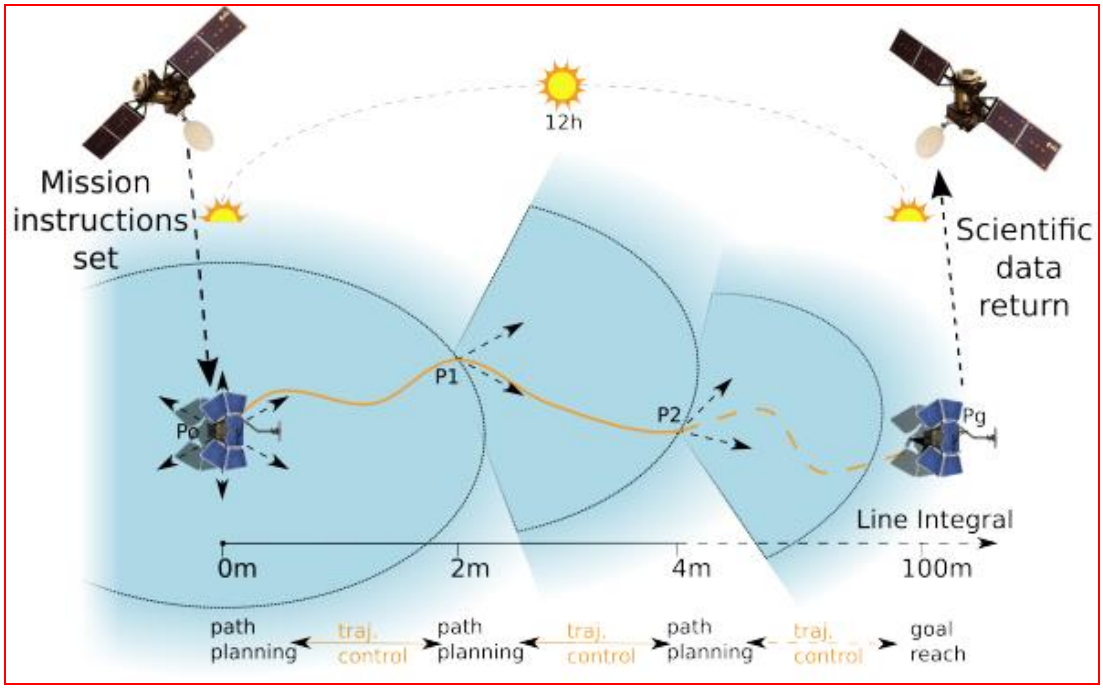
За претрагу окружења примењују се, осим *frontier-based*, и друге врсте стратегија. Неке стратегије, на пример, подразумевају учешће човека у петљи избора следеће позиције робота, затим примену анализе вероватноће разних бенефита у процесу селекције између позиција-кандидата или се једноставно следећа позиција бира случајним избором, итд. Оно што је заједничко за све класичне стратегије за претрагу окружења јесте да имају за циљ да се претрага реализује за што краће време или за што краћи укупан пређени пут робота.

На *Слици 21.* је презентовано истраживање *Carnegie Mellon* универзитета које показује да *frontier-based* стратегије имају боље перформансе у односу на остале две анализиране врсте стратегија за претрагу окружења [64].



Слика 21. Различити концепти претраге окружења, модификована слика из [64].

На *Слици 22.* је приказан генерални принцип функционисања претраге окружења у планетарним мисијама [65].



Слика 22. Сценарио претраге окружења у планетарним мисијама [65].

3.2. Концепт *frontier-based* претраге окружења

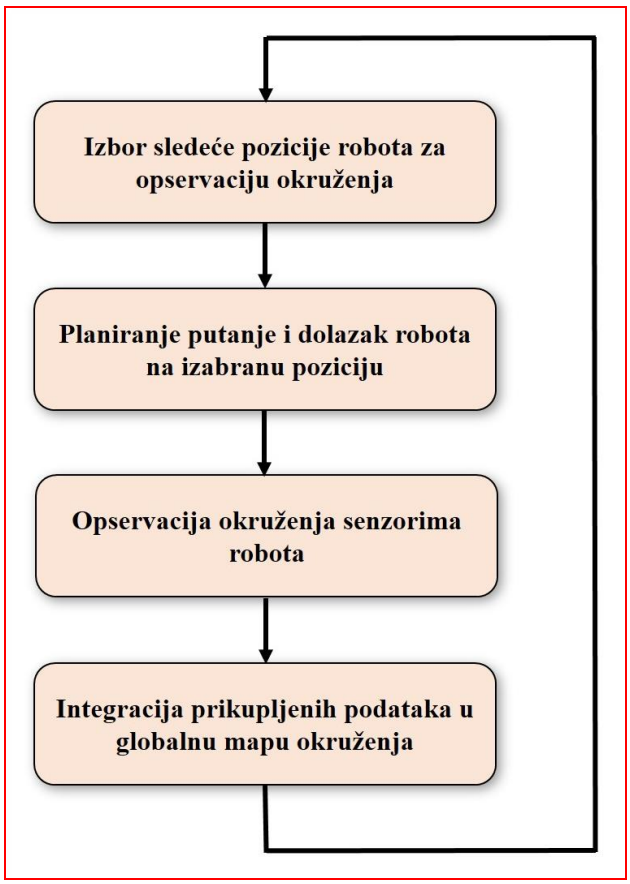
Као што је већ наведено у *Поглављу 3.1*, највећи број стратегија за претрагу окружења припада групи *frontier-based* стратегија и базира се на избору следеће позиције робота из скупа позиција-кандидата које леже на граници између претражених и непретражених делова окружења. Концепт *frontier-based* претраге окружења први пут је званично презентован у раду аутора *Yamauchi* 1997. године [11], а и данас представља један од основних праваца истраживања у области аутономне претраге окружења.

У [66] за претрагу окружења великих димензија предлаже се комбинација локалне (са више детаља) и глобалне (са мање детаља) *frontier-based* претраге, покушавајући да се на тај начин искористе предности и умање недостаци ове две варијанте претраживања како би се као коначан резултат добило повећање ефикасности претраге. Интересантан алгоритам у мисијама *frontier-based* претраге окружења за потребе трагања и спасавања коју реализују дронови презентован је у [19]. Ту се за опажање окружења као сензори користе *RGB-D* камере, а у процесу формирања делова граница између претраженог и непретраженог дела окружења, где се лоцирају кандидати за следећу позицију робота, примењује се техника кластеризације. За избор следеће позиције робота примењује се анализа потенцијалне добити података о окружењу сваке позиције-кандидата, док је за планирање путање робота имплементиран *A** алгоритам. У [67] је предложен и тестиран приступ *frontier-based* претраге окружења где се следећа позиција робота бира анализирајући потребан број ротација робота да дође до сваке позиције-кандидата на граници између претраженог и непретраженог дела окружења, и то тако да се тежи да се што мање одступа од задатог правца претраге.

Концепт *frontier-based* претраге примењује се често и код претраге окружења са вишероботским системима. С тим у вези, у литератури су предложене различите технике истраживања непознатог окружења које примењују аутономни тимови са више робота. На пример, у [68] је презентована *frontier-based* претрага са више робота која користи доступне информације о окружењу како би се оптимизовало планирање путања за сваки робот. У [69] је представљен метод за координацију тима робота током претраге непознатог окружења које може укључивати покретне препреке. Овај метод се заснива на подели мапе на зоне, тако да сваки робот претражује додељену зону. У [20] је примењено пет стратегија за претрагу окружења са више робота у циљу тестирања како ефикасност стратегије за претрагу зависи од врсте окружења.

Основни кораци *frontier-based* претраге окружења, као комбинације задатака мапирања окружења у којем се креће робот и планирања путање робота, приказани су на *Слици 23*. и могу се дефинисати на следећи начин [70]:

1. Избор следеће позиције робота за опсервацију окружења на основу примењене стратегије претраге.
2. Долазак робота на позицију изабрану у кораку 1. Овај корак захтева претходно планирање путање робота од актуелне до следеће позиције применом изабраног алгоритма (планера).
3. Опсервација окружења прикупљањем података сензорима робота са нове позиције.
4. Интеграција података прикупљених у кораку 3. у глобалну мапу окружења.



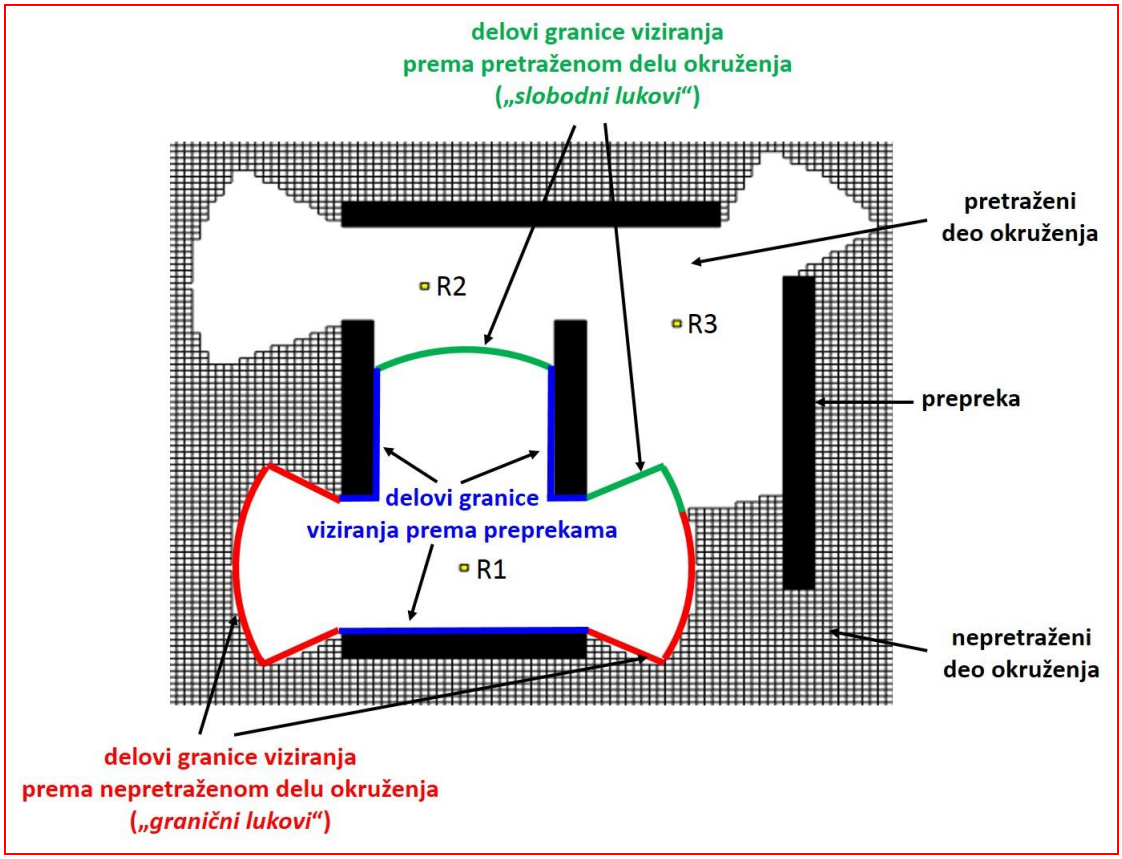
Слика 23. Основни кораци frontier-based претраге окружења.

Да би се примениле *frontier-based* стратегије за претрагу окружења, мапа окружења у којем се креће робот обично се представља у виду мреже ћелија униформне резолуције, на исти начин као што се то ради у случају примене алгоритама за планирање путање робота који функционишу на бази обраде графа [11]. Свака ћелија која припада претраженом делу окружења, а чија било која суседна ћелија припада непретраженом делу окружења, назива се гранична ћелија. Груписањем суседних граничних ћелија формирају се већи ентитети које ћемо назвати условно гранични лукови („*frontier arcs*"). Услов који треба група суседних граничних ћелија да задовољи да би се класификовала као гранични лук може се дефинисати на различите начине. У овој дисертацији усвојен је приступ према којем се гранични лук састоји од непрекинутог низа суседних граничних ћелија које припадају границама визирања робота (дефинисаним максималним дометом сензора посматрано из његове тренутне позиције) између две суседне граничне линије визирања робота и одговарајућих граничних ћелија које припадају тим линијама визирања [71]. Други начин дефинисања граничних лукова може бити према унапред задатом фиксном броју суседних граничних ћелија и слично.

Осим граничних лукова (који се дакле одређују према непретраженом делу окружења), граница визирања робота дефинисана максималним дометом његових сензора посматрано из његове тренутне позиције састоји се и од границе према остатку претраженог или познатог окружења, која може садржавати делове оријентисане према слободном простору (тзв. слободни лукови – „*free arcs*") и/или делове који одговарају контурним линијама препрека, такође ограничене суседним граничним линијама визирања робота [71].

Примери конкретне примене претходно описаног концепта *frontier-based* претраге окружења могу се пронаћи у радовима [8,11,12,71,72].

На *Слици 24.* су приказани гранични лукови, слободни лукови и делови границе визирања према препрекама у оквиру *frontier-based* претраге окружења по претходно описаном концепту, а на примеру једног сценарија симулираног у Матлабу у склопу истраживања обухваћених дисертацијом.



Слика 24. Гранични лукови, слободни лукови и делови границе визирања робота према препрекама у оквиру frontier-based претраге окружења (пример урађен у склопу истраживања обухваћених дисертацијом).

Усвојимо и да централна тачка сваког граничног лука представља једног кандидата за следећу позицију робота у мисији претраге окружења. Такав приступ је, такође, чест у литератури која проучава ову област [8,12,73].

За евалуацију позиције-кандидата p у процесу избора следеће позиције робота у *frontier-based* стратегијама за претрагу окружења, у литератури се најчешће предлажу следећи критеријуми:

1. $L(p)$, минимална дужина или цена (генералнији случај) путање од актуелне позиције робота до позиције-кандидата p;
2. $A(p)$, информативни потенцијал позиције-кандидата p, који у ствари представља унапред процењену добит података о окружењу коју робот може остварити својим сензорима уколико дође на ту позицију;

3. $P(p)$, вероватноћа да ће робот, у случају да дође на позицију p, бити у могућности да успостави комуникацију са базном станицом и проследи истој прикупљене податке.

За израчунавање вредности критеријума *L(p)* најчешће се користи неки од алгоритма за планирање путање робота (планера). Обично се исти алгоритам користи и за планирање путање робота од тренутне до изабране позиције тако да је он од изузетне важности за претрагу окружења. Алгоритми који прорачун путање робота врше на бази обраде графа се као пouздани планери често користе у мисијама претраге окружења. Из те фамилије најчешће је коришћен базични алгоритам тј. *A** [19-22], затим *Dijkstra* алгоритам [66], као и *D** [23] и *RRT** [74]. *D* Lite* као варијанта *D** и напредна верзија алгоритма који функционише на бази обраде графа са пуно практичних имплементација је такође погодан за примену у мисијама претраге окружења. Могу се применити и различити модели анализе вероватноће ради повећања поузданости процеса планирања путања, тако да је и тај приступ могућ у мисијама претраге окружења [75].

Израчунавање вредности критеријума *A(p)* у генералном случају подразумева највише неодређености у процесу управљања претрагом окружења. Са друге стране, једино он доприноси директно циљу претраге – увећава кумулативно претражени део окружења. Класичан начин одређивања вредности овог критеријума базира се на мапи тренутно претраженог дела окружења и познавању максималног домета сензора робота, на основу чега се процењује добит података о окружењу коју робот може остварити својим сензорима уколико дође на одређену позицију. Имајући у виду важну улогу овог критеријума у процесу избора следеће позиције робота, треба поменути да се у литератури предлажу и друге технике за одређивање његове вредности. У том контексту, у [76] се предлаже имплементирање неуронске мреже како би се извршила процена вредности критеријума *A(p)*. У [77] се користе топографске карте или ручно урађени нацрти окружења како би се процена вредности критеријума *A(p)* учинила прецизнијом и тиме унапредила ефикасност претраге. Приступ где се за процену подобности потенцијалних позиција са аспекта вредности критеријума *A(p)* користи анализа вероватноће предложен је у [78]. У [79,80] је предложено увођење хеуристике за додатни опис потенцијалне следеће позиције робота у циљу поузданије процене добити података о окружењу.

Вредност критеријума *P(p)* је у генералном случају обрнуто пропорционална раздаљини између позиције-кандидата p и базне станице са којом треба одржавати комуникацију, па се стога за његову меру често користи реципрочна вредност Еуклидове удаљености p од базне станице [12].

У литератури се, такође, предлажу различити додатни критеријуми за избор следеће позиције робота. На пример, у [81] се као додатни критеријум предлаже степен преклапања ($O(p)$) тренутне мапе окружења и дела окружења видљивог из позиције p. Такође се предлажу критеријуми као што су број унапред дефинисаних оријентира видљивих из позиције p, као и број потребних ротација робота да дође на позицију p [82]. Избор критеријума у суштини зависи од специфичности и циљева конкретне мисије претраге окружења. У том контексту, у раду [10] се предлаже да се у мисијама претраге окружења у циљу трагања и спасавања када год је то могуће уведе критеријум који ће узети у обзир типове објеката који су најближи анализираној позицији-кандидату, како би се нпр. мисија најпре усмерила у стамбени део окружења. Такође се као један од критеријума може користити и преостали капацитет батерије робота [12].

3.3. *Frontier-based* стратегије за претрагу окружења

Приликом „дизајнирања“ стратегије за претрагу окружења, главни изазов је постизање добрих глобалних (дугорочних) перформанси предузимањем локалних (краткорочних) одлука које се доносе на основу ограничене, обично недовољне количине информација у тренутку одлучивања. Највећи број *frontier-based* стратегија за претрагу окружења врши избор следеће позиције робота анализирајући погодност сваке позиције-кандидата према унапред одређеним критеријумима комбинујући их у форми неке функције. Најчешће коришћени критеријуми за ту намену наведени су у *Поглављу 3.2*. Неке од базичних *frontier-based* стратегија за претрагу окружења које се предлажу у литератури су следеће:

- Прва стратегија је тривијална и она за следећу позицију робота бира ону до које је планер у односу на актуелну позицију робота израчунао најмању дужину или цену путање рачунајући све позиције-кандидате. Обично се означава са *Dist_Min*.

- Стратегија презентована први пут у [83] комбинује критеријуме $A(p)$ и $L(p)$ у форми линеарне функције:

$$u(p) = \alpha A(p) - L(p) \tag{3}$$

  Ова стратегија се обрађује и у раду [12] у измењеној форми:

$$u(r) = A(r) - \beta L(r) \tag{4}$$

  где параметер β регулише утицај критеријума $L(p)$ у односу на критеријум $A(p)$ на избор позиције. Код ове стратегије у обе наведене појавне форме вредности $A(p)$ и $L(p)$ се претходно нормализују максималним вредностима тих критеријума, узимајући у обзир све позиције-кандидате чија евалуација се ради у актуелном кораку одлучивања. По ауторима који су је први предложили ова стратегија у литератури се назива као *WS* стратегија [12].

- У [13] се предлаже стратегија која описује позицију-кандидата p у форми експоненцијалне функције критеријума $A(p)$ и $L(p)$:

$$u(r) = A(r) \cdot \exp(-\lambda L(r)) \tag{5}$$

  Параметар λ је већи од 0. Ова стратегија у литератури се назива, такође по ауторима, као *GBL* стратегија [10,12] или као *Latombe* стратегија [8] (по једном од аутора).

- Укључивање вероватноће успостављања комуникације између робота и базне станице у процес евалуације позиција-кандидата, имајући у виду значај тог критеријума, посебно у мисијама трагања и спасавања (када за што краће време треба проследити прикупљене податке са терена), први пут је предложено у [14] у следећој форми:

$$u(r) = \frac{A(r)P(r)}{L(r)} \tag{6}$$

У [12] се ова стратегија назива *AOJRF* стратегија, имајући у виду да је исту са успехом користио тим *Amsterdam Oxford Joint Rescue Forces* на такмичењу *RoboCup Rescue Virtual Robots Competition* 2009. године.

У последњој деценији, поред претходно наведених стратегија, све више се користе класичне методе вишекритеријумског одлучивања (*ВКО*) за избор следеће позиције робота у мисијама претраге окружења. Основни мотив за то је чињеница да *ВКО* пружа широк и флексибилан приступ у одабиру критеријума и њихових тежина који се могу користити за евалуацију позиција-кандидата, чиме се обезбеђује ефикасно управљање претрагом у зависности од услова у којима се одвија и њених циљева [8,12]. У [8,12], на пример, предложена је стратегија која користи тзв. *Choquet fuzzy* интеграл за избор следеће позиције робота. Резултати презентовани у [12] показују да у случају претраге окружења једним роботом стратегија базирана на *Choquet fuzzy* интегралу има упоредиве перформансе са *WS* и *AOJRF* стратегијом, а боље перформансе у односу на *Dist_Min* стратегију. У [10] се за избор следеће позиције робота предлаже стандардна метода *ВКО PROMETHEE II*. Ова метода пружа, поред тежинских коефицијената критеријума, функције преференције као додатни механизам који кроз оцену међусобног односа критеријума утиче на одлучивање. Резултати презентовани у [10] показују да стратегија која користи *PROMETHEE II* има упоредиве перформансе са *GBL* и стратегијом базираном на *Choquet fuzzy* интегралу, а такође боље перформансе у односу на *Dist_Min* стратегију. Стратегија за претрагу окружења базирана на *ВКО* методи *ARAS* (*Additive Ratio Assessment*) сугерише се у [84]. У оквиру те стратегије за избор следеће позиције робота комбинују се, поред критеријума „дужина путање од актуелне позиције до позиције-кандидата“ и „информативни потенцијал позиције-кандидата“ и следећи критеријуми: преостали капацитет батерије робота, вероватноћа судара са другим објектима, број потребних ротација робота, итд. У [9] предлаже се приступ по коме се у оквиру стратегија за претрагу окружења које су базиране на *ВКО* користи нацрт мапе окружења, иако је он у мањој или већој мери непрецизан. Резултати експеримента показују да предложени приступ има боље перформансе у различитим типовима окружења у односу на стратегију без претходних информација. Иако употреба прецизнијих претходних информација доводи до значајног побољшања перформанси, коришћење недовољно поузданих информација може, такође, довести до одређених предности. У [85] су представљене стратегије за аутономну претрагу окружења након нуклеарне катастрофе. Ови приступи су развијени за мобилне роботе који користе мерне инструменте за откривање радиоактивних жаришта. На основу презентованих резултата, закључено је да стратегије базиране на *ВКО* имају добре перформансе у откривању и мапирању извора радиоактивног зрачења. Могуће је и потенцијално укључивање теорије *soft* скупова у *ВКО* [86] за потребе примене у мисијама претраге окружења.

Имајући у виду претходно дат преглед литературе, може се закључити да се као референтне стратегије за претрагу окружења најчешће користе *Dist_Min*, *GBL*, *AOJRF* и *WS* [8,10,12].

У прилог томе да су *Dist_Min*, *GBL*, *AOJRF* и *WS* у литератури примарно разматране као класичне и уједно референтне стратегије за претрагу окружења може се додати и следеће. У [16] се врши поређење више стратегија, са акцентом на *GBL* и унапређену варијанту *Dist_Min*. *GBL* се класификује као *„state-of-the-art decision-theoretic“* стратегија, а циљ истраживања у раду је доказивање да *Dist_Min* стратегија, иако тривијална, може у неким ситуацијама дати упоредиве или боље резултате од *GBL* стратегије. Референцирајући се на [16] у [17] се предлаже примена *Dist_Min* код роботских усисивача. У [15] се предлаже *WS* као стратегија за претрагу окружења којом се постиже добар баланс између укупне пређене дужине пута робота и броја позиција које

робот мора да посети током претраге (што је у директној вези са информативним потенцијалом позиција које се бирају).

На основу наведеног у овом поглављу, стратегије за претрагу окружења *Dist_Min*, *GBL*, *AOJRF* и *WS* су и у дисертацији коришћене као референтне.

3.4. Врсте сензора које робот најчешће користи у мисијама претраге окружења

Више типова сензора који се одликују различитим карактеристикама може се користити у поимању сцене која окружује робота тј. за опажање окружења у којем обавља задатке. Следе примери неких од сензора који се најчешће користе у мисијама претраге окружења [87]:

- Оптички сензори
  - Инфрацрвени (*IC*) сензори: сензори који функционишу по принципу емитовања импулсног зрачења и његове повратне рефлексије. Удаљеност објекта може да се одреди мерењем времена потребног да се светлост (зрачење) емитује и прими као рефлектовани сигнал од објекта;
  - *Light Detection and Ranging* (*LIDAR*): детектор светлости и мерач даљине (често називан и као ласерски скенер) омогућава мерење удаљености до објекта коришћењем импулсног ласерског зрачења;
  - *RGB* камера: камера са приказом слике сцене у боји;
  - Стерео камера: врста камере која користи два или више сочива од којих свако има посебан сензор за генерисање слике;
  - *RGB-D* камера: представља аквизициони систем који обухвата камеру у боји и „дубинску“ камеру у циљу *3D* опажања окружења (нпр. *Microsoft Kinect*);
- *GPS* сензор: модул намењен за одређивање положаја и детекцију кретања;
- Инерцијални мерни сензори (*IMU*): омогућавају мерење специфичних параметара робота, пре свега убрзања и инклинације тј. нагиба (нпр. акцелерометар и жироскоп);
- Контактни сензори: омогућавају проверу да ли је робот дошао у физички контакт са неким објектом;
- Сензори близине: омогућавају детекцију присуства објеката у близини без физичког контакта, нпр. *SONAR* је сензор који проверава близину околних објеката користећи звучни талас.

4. УНАПРЕЂЕЊЕ ПРЕТРАГЕ ОКРУЖЕЊА ПРИМЕНОМ ВИШЕКРИТЕРИЈУМСКОГ ОДЛУЧИВАЊА И *D* Lite* АЛГОРИТМА

Ово поглавље посвећено је унапређењу претраге окружења применом вишекритеријумског одлучивања и *D* Lite* алгоритма, као и унапређењу планирања путање робота у мисијама трагања и спасавања и има три целине. На почетку прве целине уводе се стратегије за претрагу окружења које за избор следеће позиције робота користе методе вишекритеријумског одлучивања *SAW, COPRAS* и *TOPSIS*. После описа модела окружења за претрагу и критеријума за избор следеће позиције робота, врши се опсежно тестирање путем симулација у Матлабу различитих сценарија претраге окружења применом поменутих метода *ВКО*, као и класичних стратегија за претрагу окружења из литературе анализираних у дисертацији (*Dist_Min, GBL, AOJRF, WS*). На крају ове целине презентују се резултати и врши дискусија у вези са истим. У другој целини у оквиру овог поглавља предложена су два приступа за унапређење планирања путање робота применом *D* Lite* алгоритма у мисијама претраге окружења за потребе трагања и спасавања у урбаном окружењу. Први приступ представља примену *D* Lite* алгоритма у комбинацији са фази логиком, а други примену *D* Lite* алгоритма у комбинацији са *on-line* учењем. Трећа целина бави се верификацијом кључних резултата истраживања коришћењем *Gazebo* симулатора (специјализованог окружења за *3D* симулацију робота) у комбинацији са пакетима који раде под *Robot Operating System* (*ROS*), као веродостојна замена за тестирање анализираних стратегија за претрагу окружења на реалним роботима. За те потребе примењен је виртуелни модел *Turtlebot 3* робота.

4.1. Стратегије за вишекритеријумско одлучивање у претрази окружења

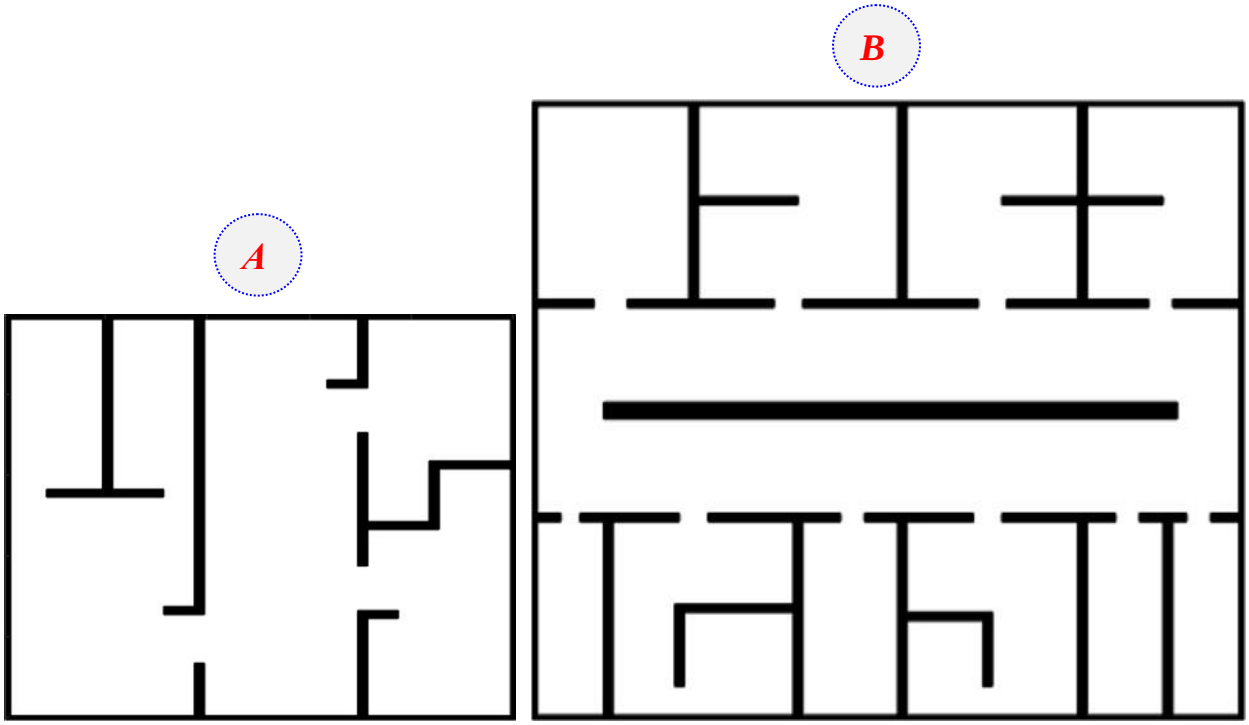
У циљу тестирања путем симулација у Матлабу различитих сценарија претраге окружења применом поменутих метода *ВКО*, као и класичних стратегија за претрагу из литературе која проучава ову област, најпре треба дефинисати моделе окружења за претрагу и изабрати критеријуме за евалуацију позиција-кандидата.

4.1.1. Модели окружења и поставка проблема

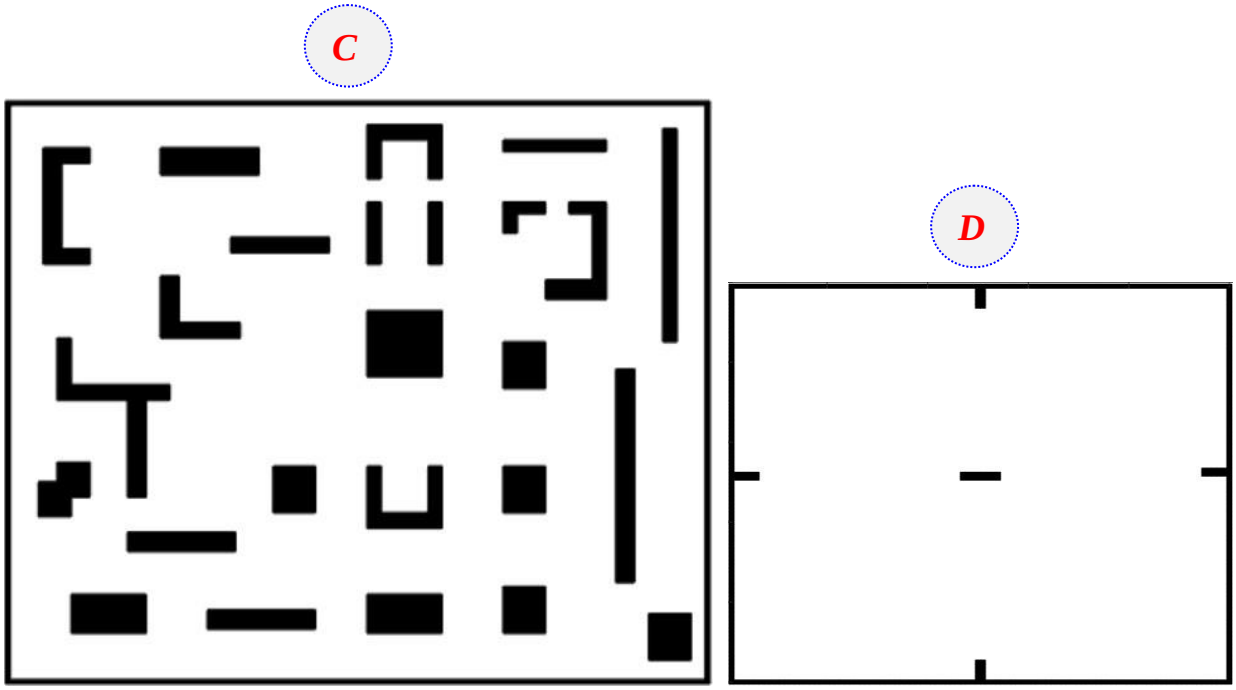
Евалуација стратегија за претрагу извршиће се симулацијом претраживања окружења са различитим степеном комплексности, а у циљу њиховог тестирања у условима што ближим различитим реалистичним сценаријима. Основне мапе окружења су представљене у бинарној форми, тако да је слободан простор означен белом бојом, а препреке (у овом случају зидови) црном боје. Остале врсте препрека су изостављене и не разматрају се. Димензије мапа су приказане у односу на њихову дискретизовану форму тј. према броју ћелија добијених дискретизацијом. Претпоставка је да су све ћелије у свим случајевима квадратног облика и једнаких димензија.

Стратегије су тестиране у три „*indoor*“ окружења. Прво окружење је димензија 100x100 ћелија и симулира пословни или стамбени простор са више одвојених просторија различитих величина. Друго окружење је димензија 150x150 ћелија и симулира нпр. један спрат тржног центра са просторијама сличних величина и дугачким ходницима. У оба случаја просторије су одвојене зидовима који представљају једине препрека са аспекта кретања робота. Треће окружење је димензија 100x100 ћелија и симулира једну већу просторију типа магацина која садржи пет полупреградних зидова. Стратегије су, такође, тестиране и у једном „*outdoor*“ окружењу димензија 150x150 ћелија, које симулира један

стамбени блок. Присутност препрека (у овом случају зидова) и њихове димензије су мера комплексности окружења. Мапе наведених окружења које ће се користити за тестирање стратегија за претрагу приказане су на *Слици 25*.



а) „Indoor“ окружења A (100x100) и B (150x150)



б) „Outdoor“ окружење C (150x150) и „indoor“ окружење D (100x100)

Слика 25. Мапе окружења коришћене за тестирање стратегија за претрагу.

За избор следеће позиције робота у току претраге окружења примењују се методе вишекритеријумског одлучивања *SAW*, *COPRAS* и *TOPSIS*, затим се тестира и пореди њихова ефикасност међусобно, као и са осталим анализираним класичним стратегијама из литературе (*Dist_Min*, *GBL*, *AOJRF*, *WS*).

На бази проучене литературе, за меру ефикасности претраге узет је просечан укупан пређени пут робота у односу на више стартних позиција и то за потребе реализације претраге 90% окружења (преосталих 10% окружења углавном се односи на углове и мање приступачне делове) [10,12]. Више стартних позиција се тестира из разлога што оне утичу на почетне услове који усмеравају даљи ток претраге [8].

4.1.2. Критеријуми за избор следеће позиције робота

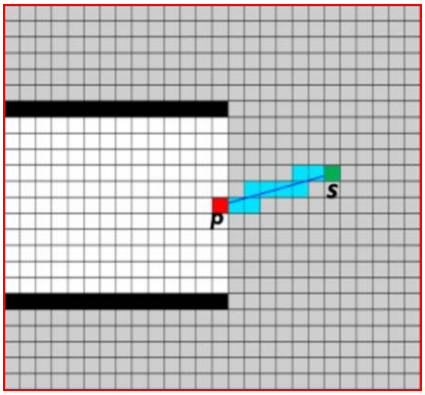
За избор следеће позиције робота (p), у дисертацији су коришћени следећи критеријуми, ближе објашњени у *Поглављу 3.2*:

- $L(p)$, дужина путање од актуелне позиције робота до позиције-кандидата p,
- $A(p)$, информативни потенцијал позиције-кандидата p,
- $P(p)$, удаљеност позиције-кандидата p од базне станице.

За одређивање вредности критеријума $L(p)$ у дисертацији се користи *D* Lite* алгоритам, описан у *Поглављу 2.3.1*.

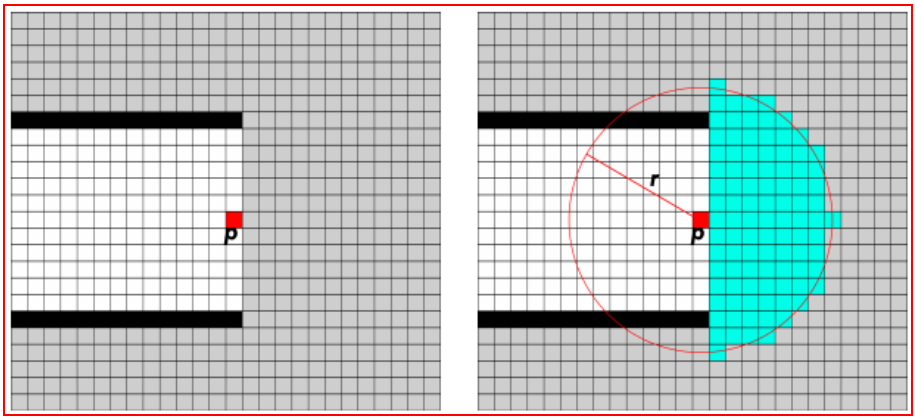
Вредност критеријума $P(p)$ представља Еуклидову удаљеност p од базне станице [12].

Вредност критеријума $A(p)$, који у ствари представља унапред процењену добит података о окружењу коју робот може остварити својим сензорима уколико дође на позицију p, израчунава се на начин описан у наставку [9]. Како се врши дефинисање позиција-кандидата код *frontier-based* претраге окружења изложено је у *Поглављу 3.2*. Да би се проценила видљивост тачке s сензором робота домета r из тачке p (у случају позиционирања робота у тачку p) потребно је проверити да ли тачка s припада одговарајућој линији визирања. Најпре треба проверити да ли је тачка s у домету сензора робота што се ради израчунавањем Еуклидове удаљености између тачака p и s. Затим се проверава има ли препрека између тачака p и s идући од p према s које би онемогућиле видљивост тачке s сензором робота из тачке p. Имајући у виду да се мапа окружења у којем се креће робот представља у виду мреже ћелија униформне резолуције, на *Слици 26.* су плавом бојом означене ћелије које припадају линији визирања између тачака p и s, где је тачка q означена зеленом бојом и у овом случају припада линији визирања. Ћелије које припадају претраженом делу окружења су означене белом бојом, а ћелије које припадају непретраженом делу окружења сивом бојом. Ћелије које припадају препрекама (зидовима) су означене црном бојом.



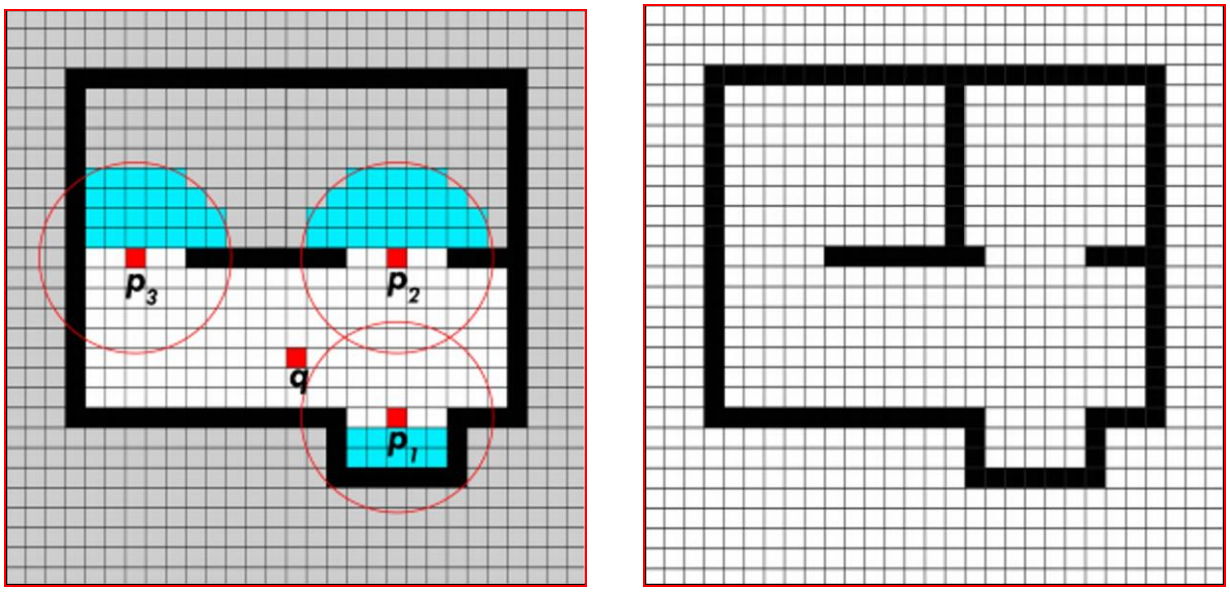
Слика 26. Линија визирања (ћелије означене плавом бојом) између позиција p и s у дискретизованој мапи окружења, модификована слика из [9].

Узимајући у обзир претходно наведено, на *Слици 27.* илустрована је процена вредности информативног потенцијала позиције *p* (скуп ћелија означених плавом бојом).



Слика 27. Процена вредности информативног потенцијала (ћелије означене плавом бојом) позиције-кандидата p, модификована слика из [9].

На *Слици 28.* илустрована је процена информативног потенцијала за три различите позиције-кандидате.



а) мапа претраженог дела окружења *б) стварна мапа окружења*

Слика 28. Процена вредности информативног потенцијала (ћелије означене плавом бојом) позиција-кандидата p_1, p_2 и p_3. Тренутна позиција робота је означена са q. Модификована слика из [9].

Са *Слике 28.* види се да је дошло до грешке у процени информативног потенцијала за позицију-кандидата p_2, јер се у непретраженом делу окружења налази још један преградни зид који ће, уколико робот изабере ту позицију, умањити стварно опажање сензором у односу на процену изведену на бази његове максималне могућности.

4.1.3. Опис метода вишекритеријумског одлучивања *SAW, COPRAS* и *TOPSIS*

- ***TOPSIS* (*Technique for Ordering Preference by Similarity to Ideal Solution*)**

TOPSIS метода [33,35,88-91] омогућава рангирање алтернатива по више критеријума на основу поређења удаљености од идеалног и анти-идеалног решења. Идеално решење минимизира критеријуме трошковног типа, а максимизира критеријуме бенефитног типа, док за негативно идеално решење важи обрнуто. Прост пример је настојање да се у пословном одлучивању доносе (идентификују) одлуке у којима се максимизира профит, а минимизира ризик. Оптимална алтернатива је она која је у геометријском смислу најближа идеалном решењу, односно најдаља од анти-идеалног (идеалног негативног) решења.

Проблем формално приказујемо избором једне од n опција (алтернатива), које оцењујемо и поредимо међу собом на основу m критеријума чије су нам вредности познате. Алтернативе приказујемо векторима r_{ij}, где је r_{ij} вредност i-те алтернативе по j-том критеријуму. Матрица одлучивања R има облик (7):

$$R = \begin{matrix} A_1 \\ A_2 \\ \vdots \\ A_n \end{matrix} \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1m} \\ r_{21} & r_{22} & \dots & r_{2m} \\ \dots & \dots & \dots & \\ r_{n1} & r_{n2} & \dots & r_{nm} \end{bmatrix} \tag{7}$$

У овако формираној матрици одлучивања R, сваки ред матрице одговара једној алтернативи из скупа алтернатива ($A_1, A_2, ..., A_n$), а свака колона једном критеријуму из скупа критеријума ($C_1, C_2, ..., C_m$).

Будући да критеријуми у различитој мери утичу на коначне оцене алтерантива, сваком критеријуму приписујемо тежински коефицијент w_j, j=1,2,…,m (где је $\sum_{j=1}^{m} w_j = 1$) који одражава његов релативни значај у оцењивању алтернатива.

Примена *TOPSIS* метода подразумева следеће кораке:

Корак 1. Нормализација вредности матрице одлучивања. Код већине метода вишекритеријумског одлучивања, први корак представља нормализација елемената матрице одлучивања да би се добила матрица у којој су сви елементи бездимензионалне величине. Код *TOPSIS* методе примењује се векторска нормализација која је представљена следећим изразом:

$$x_{ij} = \frac{r_{ij}}{\sqrt{\sum_{i=1}^{n} r_{ij}^2}} \tag{8}$$

Након спровођења нормализације, добијамо матрицу X у којој су сви елементи нормирани и налазе се у интервалу [0, 1]:

$$X = \begin{matrix} A_1 \\ A_2 \\ \cdot \\ A_3 \end{matrix} \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1m} \\ x_{21} & x_{22} & \ldots & x_{2m} \\ \ldots & \ldots & \ldots & \\ x_{n1} & x_{n2} & \ldots & x_{nm} \end{bmatrix} \tag{9}$$

Корак 2. Множење нормализованих вредности матрице X тежинским коефицијентима критеријума:

$$v_{ij} = x_{ij} \cdot w_j; \;\; j = 1,2,...,m \tag{10}$$

Помоћу релације (10) добијамо елементе отежане нормализоване матрице $V = (v_{ij})$, где је свако v_{ij} производ одговарајуће нормализоване перформансе алтернативе и одговарајућег тежинског коефицијента критеријума:

$$V = \begin{matrix} A_1 \\ A_2 \\ \vdots \\ A_3 \end{matrix} \begin{bmatrix} v_{11} & v_{12} & \ldots & v_{1m} \\ v_{21} & v_{22} & \ldots & v_{2m} \\ \ldots & \ldots & \ldots & \\ v_{n1} & v_{n2} & \ldots & v_{nm} \end{bmatrix} = \begin{matrix} A_1 \\ A_2 \\ \vdots \\ A_3 \end{matrix} \begin{bmatrix} w_1 \cdot x_{11} & w_2 \cdot x_{12} & \ldots & w_m \cdot x_{1m} \\ w_1 \cdot x_{21} & w_2 \cdot x_{22} & \ldots & w_m \cdot x_{2m} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ w_1 \cdot x_{n1} & w_2 \cdot x_{n2} & \ldots & w_m \cdot x_{nm} \end{bmatrix} \tag{11}$$

Корак 3. Одређивање идеалног и анти-идеалног решења. Идеално решење A^* и анти-идеално решење A^- одређују се помоћу релација:

$$A^* = \left\{ (\max v_{ij} \mid j \in G), (\min v_{ij}, j \in G'), i = 1,..,n \right\} = \left\{ v_1^*, v_2^*, ..., v_m^* \right\} \tag{12}$$

$$A^- = \left\{ (\min v_{ij} \mid j \in G), (\max v_{ij}, j \in G'), i = 1,..,n \right\} = \left\{ v_1^-, v_2^-, ..., v_m^- \right\} \tag{13}$$

где је

$$G = \left\{ j = 1,2,...,m \mid j \text{ припада критеријумима који се максимизирају} \right\}$$
$$G' = \left\{ j = 1,2,...,m \mid j \text{ припада критеријумима који се минимизирају} \right\}$$

Најбоље алтернативе су оне које имају највеће v_{ij} у односу на критеријуме који се максимизирају и најмање v_{ij} у односу на критеријуме који се минимизирају. A^* указује на идеално решење, а по истој логици A^- указује на анти-идеално (идеално негативно) решење.

Корак 4: Одређивање растојања алтернатива од идеалних решења. У овом кораку се помоћу следећих релација:

$$S_i^* = \sqrt{\sum_{j=1}^{m} (v_{ij} - v_j^*)^2}, \;\; i = 1,...,n \tag{14}$$

$$S_i^- = \sqrt{\sum_{j=1}^{m} (v_{ij} - v_j^-)^2}, \;\; i = 1,...,n \tag{15}$$

израчунавају n димензиона Еуклидска растојања свих алтернатива од идеалног и анти-идеалног решења.

Корак 5. Одређивање релативне близине алтернатива идеалном решењу. За сваку алтернативу одређује се релативно одстојање:

$$Q_i^* = \frac{S_i^-}{S_i^* + S_i^-}, i = 1,...,n \tag{16}$$

где је $0 \le Q_i^* \le 1$. Алтернатива A_i је ближа идеалном решењу ако је Q_i^* ближе вредности 1, или, што је исто, ако је S_i^* ближе вредности 0.

Корак 6. Рангирање алтернатива. Алтернативе се рангирају по опадајућим вредностима Q_i^*. Најбоља алтернатива је она алтернатива чија је вредност Q_i^* највећа и обрнуто.

- ***COPRAS* (*Compressed Proportional Assessment*)**

COPRAS методу [34,92,93] карактерише у одређеној мери комплекснији поступак агрегације вредности критеријумских функција и поједностављени поступак нормализације података (не разматра се карактер критеријума - бенефитни/трошковни). У наредном делу укратко је представљен математички апарат *COPRAS* методе.

Проблем формално, као и код *TOPSIS* методе, приказујемо матрицом одлучивања $R(7)$. Сваки ред матрице R одговара једној алтернативи из скупа алтернатива ($A_1, A_2, ..., A_n$), а свака колона једном критеријуму из скупа критеријума ($C_1, C_2, ..., C_m$).

Код примене *COPRAS* методе, кораци 1. и 2. су идентични као код *TOPSIS*, с тим да се елементи x_{ij} нормализоване матрице *X* код *COPRAS* добијају применом адитивне нормализације:

$$x_{ij} = \frac{r_{ij}}{\sum_{i=1}^{n} r_{ij}} \tag{17}$$

Корак 3. У трећем кораку сумирају се елементи отежане нормализоване матрице *V* (формиране у кораку 2.) по редовима и то тако да се за сваку алтернативу формирају две вредности - S_i^+ која представља суму отежаних нормализованих вредности v_{ij} за бенефитне критеријуме и S_i^- која представља суму отежаних нормализованих вредности v_{ij} за потрошне критеријуме.

Корак 4. Агрегација вредности критеријумских функција. У четвртом кораку, применом израза (18) одређују се критеријумске функције за сваку од алтернатива из скупа алтернатива које се пореде.

$$Q_i = S_i^+ + \frac{S_{\min}^- \sum_{i=1}^{m} S_i^-}{S_i^- \sum_{i=1}^{m} \left(\frac{S_{\min}^-}{S_i^-} \right)} = S_i^+ + \frac{\sum_{i=1}^{m} S_i^-}{S_i^- \sum_{i=1}^{m} \frac{1}{S_i^-}} \tag{18}$$

Корак 5. Рангирање алтернатива. У последњем, петом, кораку врши се рангирање алтернатива на основу додељене вредности критеријумске функције, али тако што се претходно оне претварају у проценте применом израза (19).

$$N_i = \frac{Q_i}{Q_{\max}} \cdot 100\% \tag{19}$$

Алтернатива која на овај начин добије вредност критеријумске функције 100% бира се као најбоља.

- ***SAW* (*Simple Additive Weighting*)**

SAW метода [33,94] се убраја у једну од најпознатијих и свакако најзаступљенијих метода у области вишекритеријумског одлучивања. За сваку алтернативу рачуна се збирна карактеристика, која представља вредност добијену сумирањем отежаних нормализованих вредности по свим критеријумима. Она алтернатива којој одговара највећа овако израчуната вредност представља „најбоље“ решење. Поред тога што *SAW* метода обезбеђује веома једноставан и практичан поступак рангирања алтернатива, резултати који се добијају њеном применом углавном не одступају од резултата добијених неким тзв. напредним методама.

На основу наведеног, функција агрегације за рангирање алтернатива код *SAW* методе може се изразити на следећи начин [95,96]:

$$Q_i = \sum_{j=1}^{m} x_{ij} \cdot w_j \tag{20}$$

где w_j представља тежински коефицијент критеријума j, m представља укупан број критеријума, док x_{ij} представља нормализовану вредност критеријума j за алтернативу i (i=1,2,…,n).

Један од главних недостатака ове методе је што се може директно применити само ако су сви критеријуми бенефитног типа, док се критеријуми трошковног типа морају прво конвертовати у бенефитне како би се могли максимизирати.

4.1.4. Резултати тестирања

У склопу истраживања која су претходила припреми за израду дисертације, аутор дисертације је публиковао рад у коме је тестирана примена стратегија за претрагу окружења које за избор следеће позиције робота такође користе *ВКО* методе *SAW*, *COPRAS* и *TOPSIS*, као и примена стратегија *Dist_Min* и *GBL* у окружењима која одговарају окружењима у дисертацији *A*, *B* и *D* [97]. У оквиру дисертације је повећан број тестираних стартних позиција са 4 на 12 по окружењу (по три са сваке стране са једнаким међусобним растојањем које износи ¼ стране при чему је једна тачно на средини стране), а такође је симулацијом обухваћен већи број окружења (четири), као и већи број стратегија које нису базиране на *ВКО* (*Dist_Min*, *GBL*, *AOJRF*, *WS*) и све то за више варијанти домета сензора робота у окружењима *A* и *B*. Домет сензора робота изражен је у броју јединичних димензија ћелије мапе.

За вредност коефицијента β у изразу (4) за *WS* стратегију на основу [12] дефинисано је β=1, док је за вредност коефицијента λ у изразу (5) за *GBL* стратегију на основу [10,13] дефинисано λ=0.2.

Резултати тестирања у смислу просечних (средњих) укупних дужина путања робота у циљу претраге 90% окружења *A*, *B*, *C* и *D* (*Слика 25*) за анализиране стратегије приказани су у *Табелама 4*, *5*, *6*. и *7*. Графички приказ резултата тестирања који укључује и стандарде девијације приказани су на *Сликама 29*, *31*, *33*, *36*, *37*, *39*. и *41*.

У оквиру тестирања претраге окружења *A* дати су и графикони зависности прираста претраженог дела окружења од дужине путање робота за по једну изабрану стартну позицију, за све тестиране стратегије и варијанте домета сензора робота (*Слике 30, 32*. и *34*). На крају су дати резултати тестирања претраге окружење *A* са променом тежинских коефицијената критеријума током трајања претраге (*Табела 8*. и *Слика 43*).

Примери путања робота генерисаних од стране *D* Lite* алгоритма током претраге разматраних окружења, као и мапе окружења формиране након претраге применом неке од тестираних стратегија приказани су на *Сликама 35, 38, 40*. и *42*.

Подсећања ради, за избор следеће позиције робота (*p*), коришћени су критеријуми, ближе објашњени у *Поглављу 3.2*:

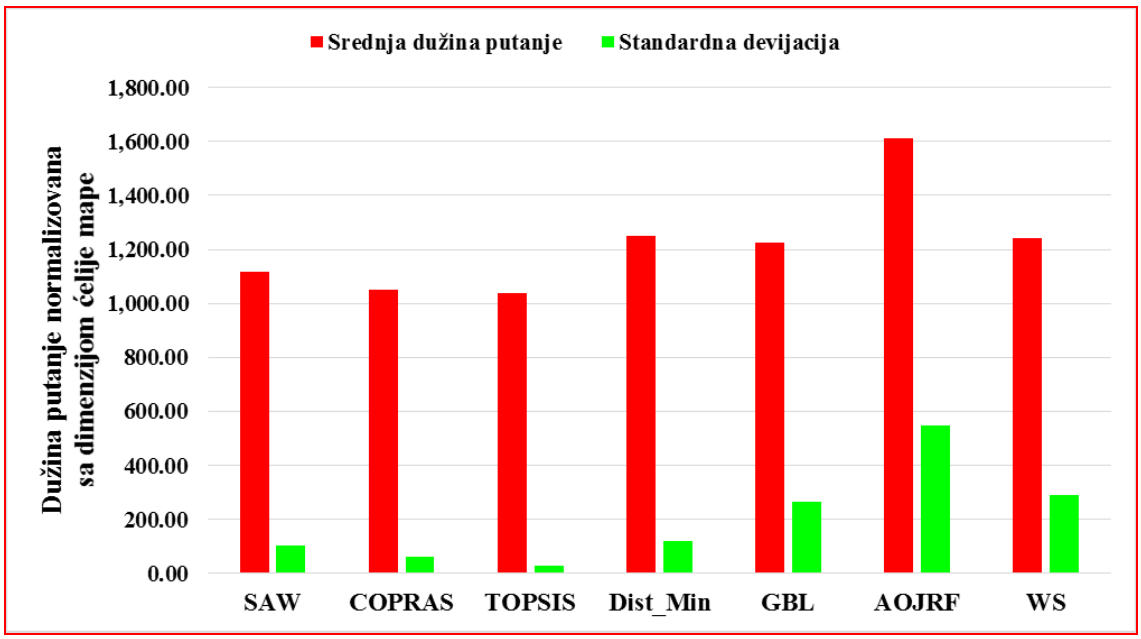
- *L*(*p*), дужина путање од актуелне позиције робота до позиције-кандидата *p*,
- *A*(*p*), информативни потенцијал позиције-кандидата *p*,
- *P*(*p*), удаљеност позиције-кандидата *p* од базне станице.

Координате стартних позиција робота дефинисане су у односу на Декартов координатни систем везан за леви доњи угао мапе. Један подеок на оси координатног система једнак је димензији ћелије мапе.

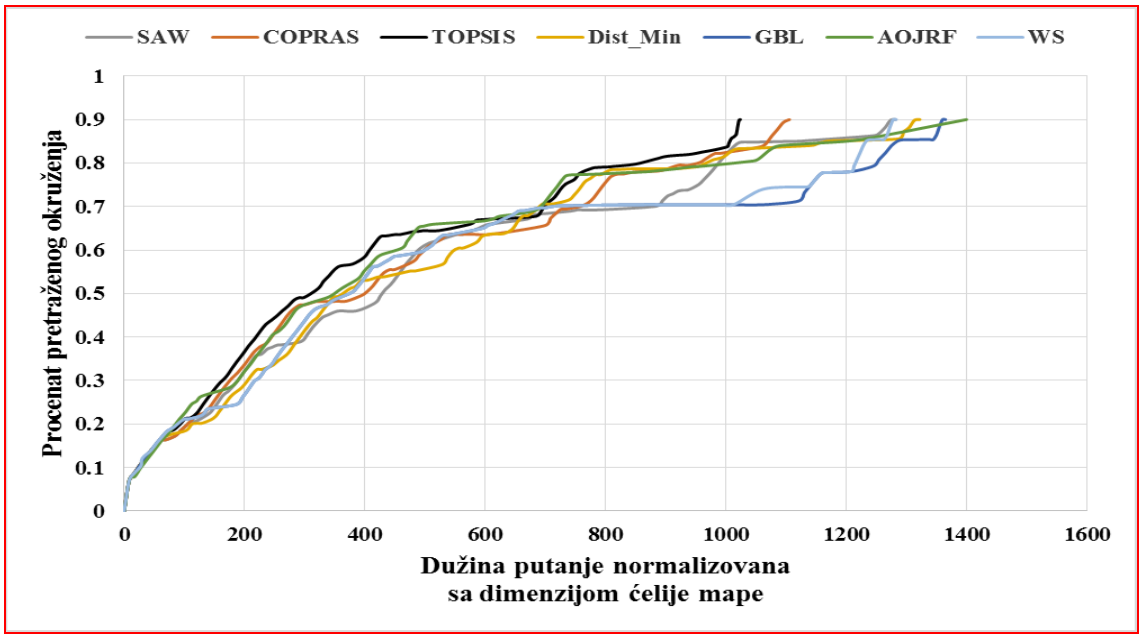
- **Окружење *A* (димензије 100x100)**

Табела 4. Просечне (средње) дужине путања робота за потребе претраге 90% окружења A применом тестираних стратегија за три варијанте домета сензора.

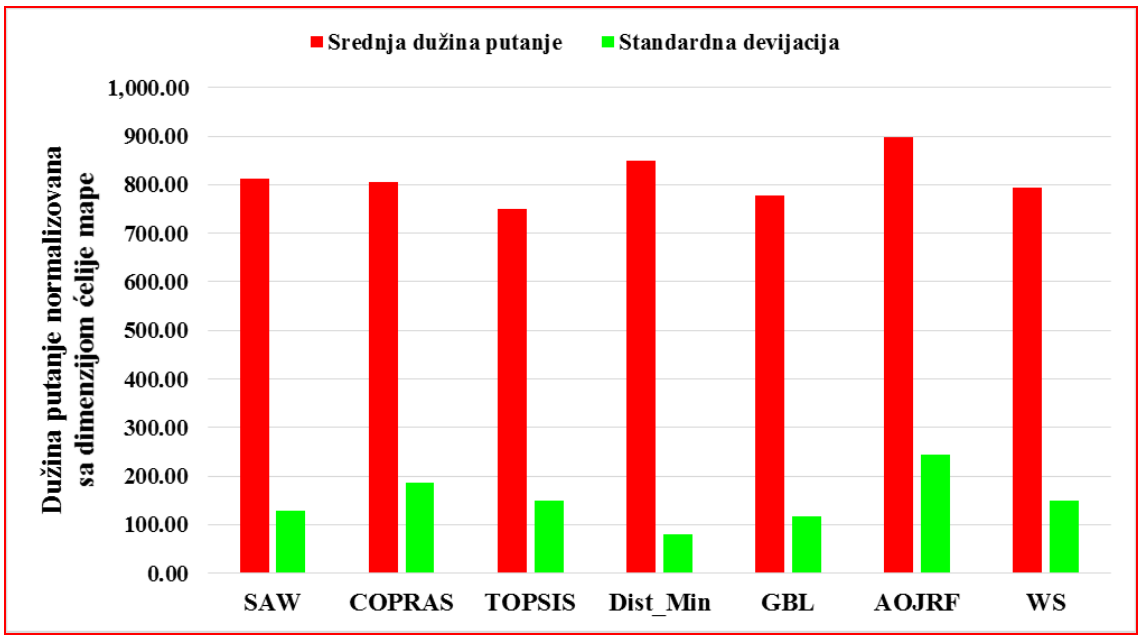
Strategije za pretragu okruženja		Srednje dužine putanja robota u okruženju *A*		
		domet senzora *r*=10	domet senzora *r*=15	domet senzora *r*=20
VKO	*SAW*	1119.05	813.03	697.51
	COPRAS	1049.31	806.06	667.88
	TOPSIS	1038.63	769.82	630.24
Dist_Min		1249.13	848.32	749.96
GBL		1225.11	778.08	736.85
AOJRF		1612.30	897.26	742.89
WS		1243.21	793.76	662.06
Napomena: Kod metoda *VKO* primenjene su kombinacije težinskih koeficijenata *(0.7, 0.2, 0.1)* pri *r*=10, odnosno *(0.6, 0.3, 0.1)* pri *r*=15 i *r*=20, za kriterijume *L*(*p*), *A*(*p*) i *P*(*p*), respektivno.				



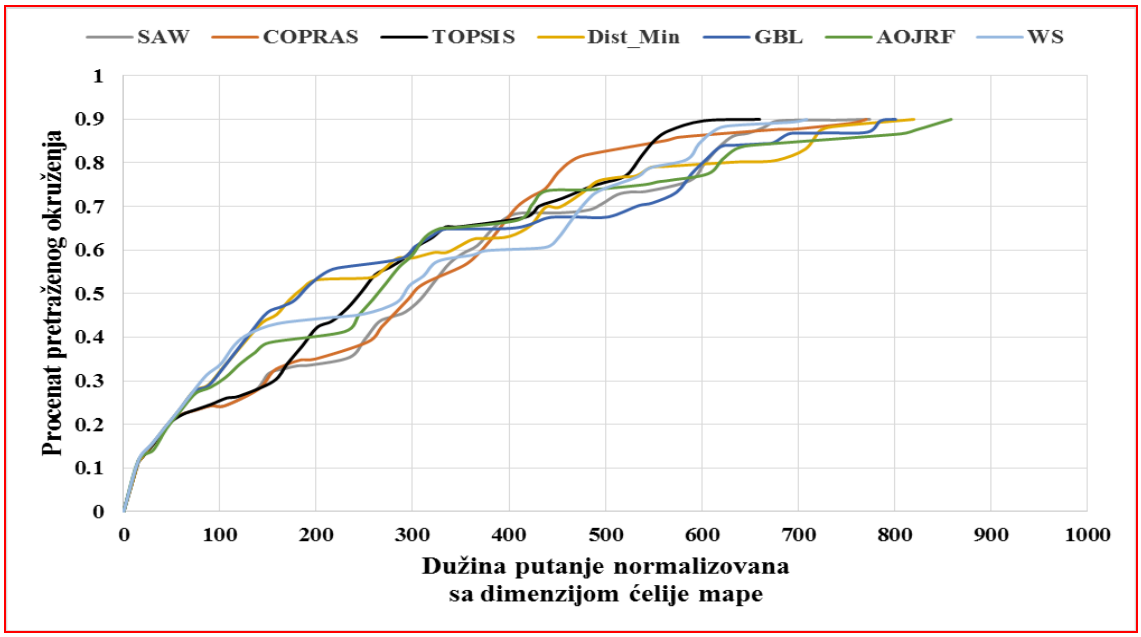
Слика 29. Графички приказ просечних (средњих) дужина путања и стандардних девијација за потребе претраге 90% окружења A применом тестираних стратегија за варијанту домета сензора r=10 и комбинацију тежинских коефицијената код ВКО (0.7, 0.2, 0.1) за критеријуме L(p), A(p) и P(p), респективно.



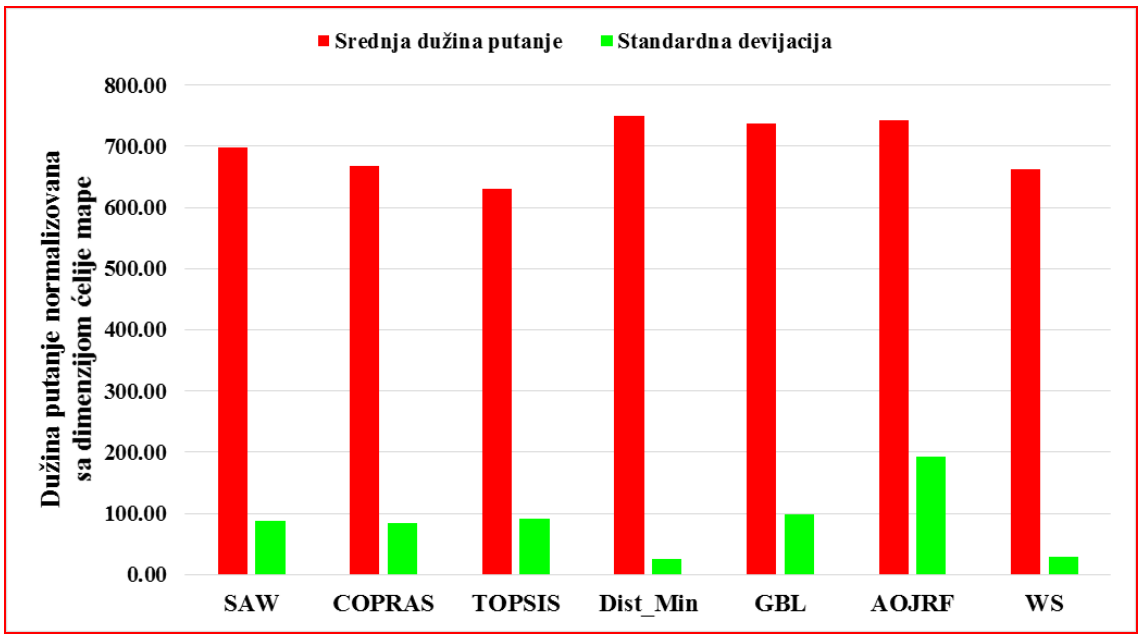
Слика 30. Зависност прираста претраженог дела окружења од дужине путање робота за потребе претраге 90% окружења A применом тестираних стратегија за варијанту домета сензора r=10, комбинацију тежинских коефицијената код ВКО (0.7, 0.2, 0.1) за критеријуме L(p), A(p) и P(p), респективно, и стартну позицију робота (x=2, y=50), у односу на Декартов КС везан за доњи леви угао мапе.



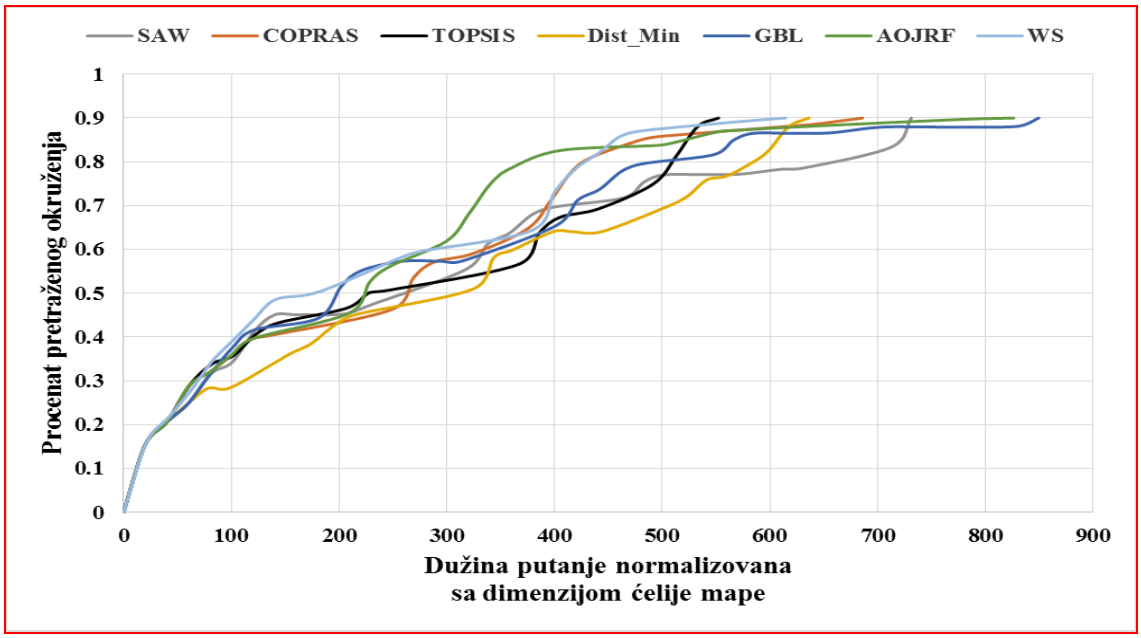
Слика 31. Графички приказ просечних (средњих) дужина путања и стандардних девијација за потребе претраге 90% окружења A применом тестираних стратегија за варијанту домета сензора r=15 и комбинацију тежинских коефицијената код ВКО (0.6, 0.3, 0.1) за критеријуме L(p), A(p) и P(p), респективно.



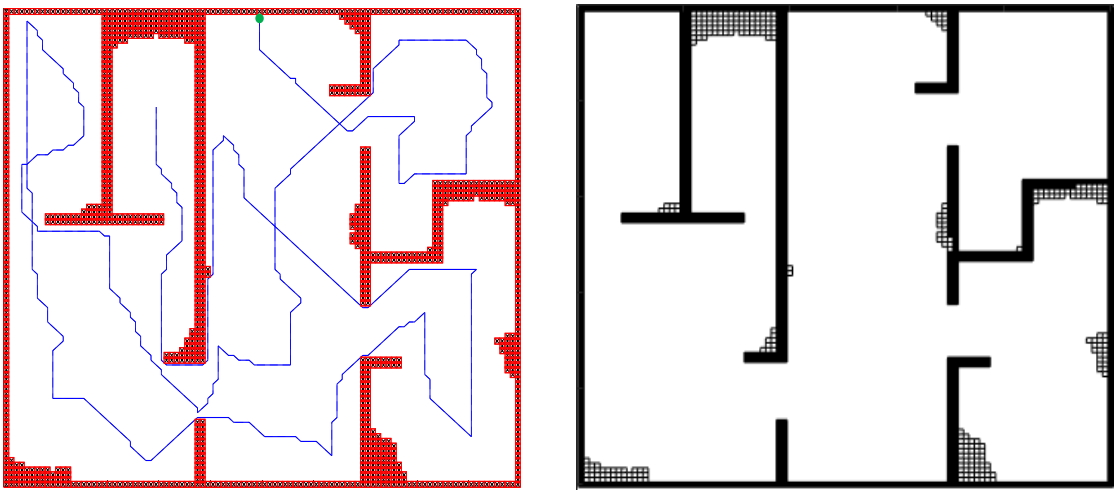
Слика 32. Зависност прираста претраженог дела окружења од дужине путање робота за потребе претраге 90% окружења A применом тестираних стратегија за варијанту домета сензора r=15, комбинацију тежинских коефицијената код ВКО (0.6, 0.3, 0.1) и стартну позицију робота (x=50, y=99), у односу на Декартов КС везан за доњи леви угао мапе.



Слика 33. Графички приказ просечних (средњих) дужина путања и стандардних девијација за потребе претраге 90% окружења A применом тестираних стратегија за варијанту домета сензора r=20 и комбинацију тежинских коефицијената код ВКО (0.6, 0.3, 0.1) за критеријуме L(p), A(p) и P(p), респективно.



Слика 34. Зависност прираста претраженог дела окружења од дужине путање робота за потребе претраге 90% окружења A применом тестираних стратегија за варијанту домета сензора r=20, комбинацију тежинских коефицијената код ВКО (0.6, 0.3, 0.1) за критеријуме L(p), A(p) и P(p), респективно, и стартну позицију робота (x=2, y=50), у односу на Декартов КС везан за доњи леви угао мапе.

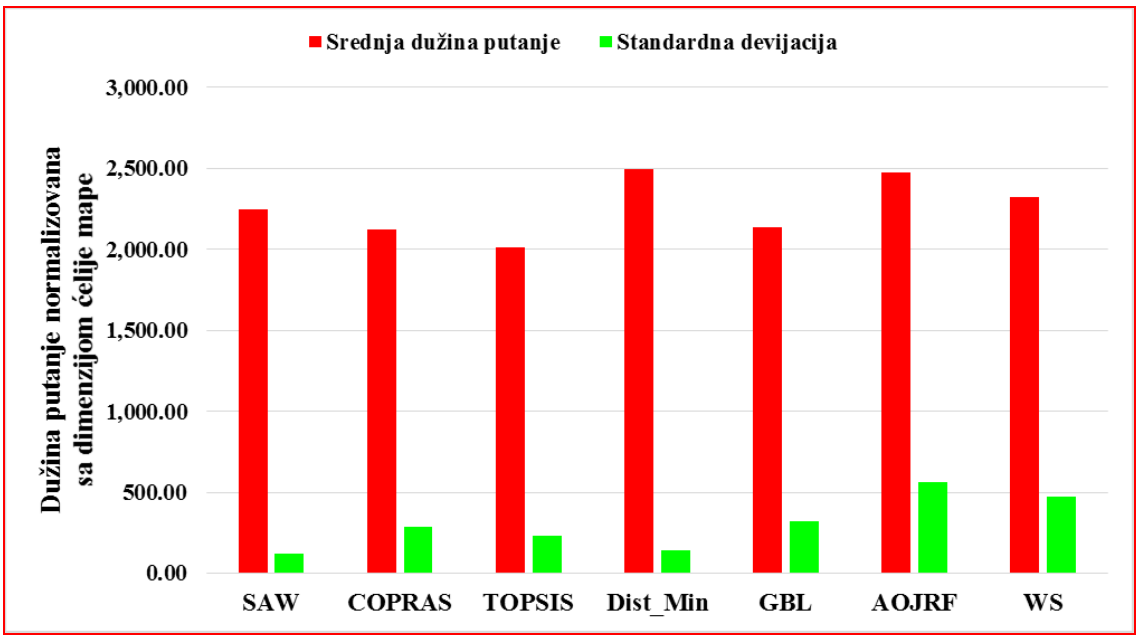


Слика 35. Путања робота генерисана D Lite алгоритмом током претраге 90% окружења A (лево) и мапа окружења након претраге (десно) применом TOPSIS методе за избор следеће позиције робота уз комбинацију тежинских коефицијената (0.6, 0.3, 0.1) за критеријуме L(p), A(p) и P(p), респективно, за варијанту домета сензора r=15 и стартну позицију робота (x=50, y=99), означену зеленим кружићем на слици лево.*

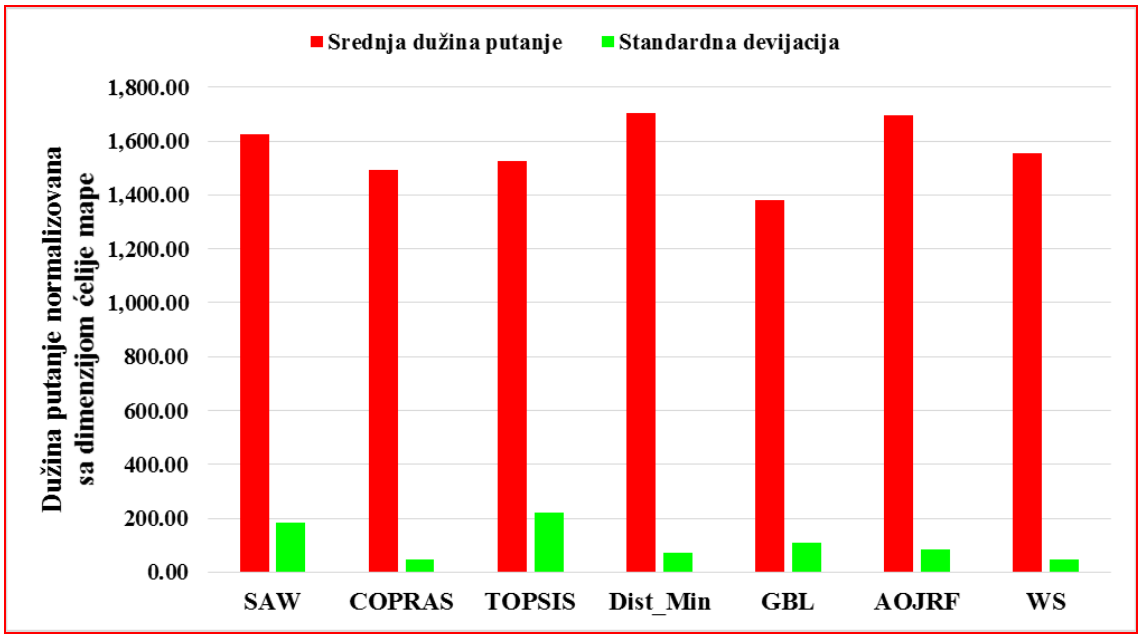
- **Окружење *B* (димензије 150x150)**

Табела 5. Просечне (средње) дужине путања робота за потребе претраге 90% окружења B применом тестираних стратегија за две варијанте домета сензора.

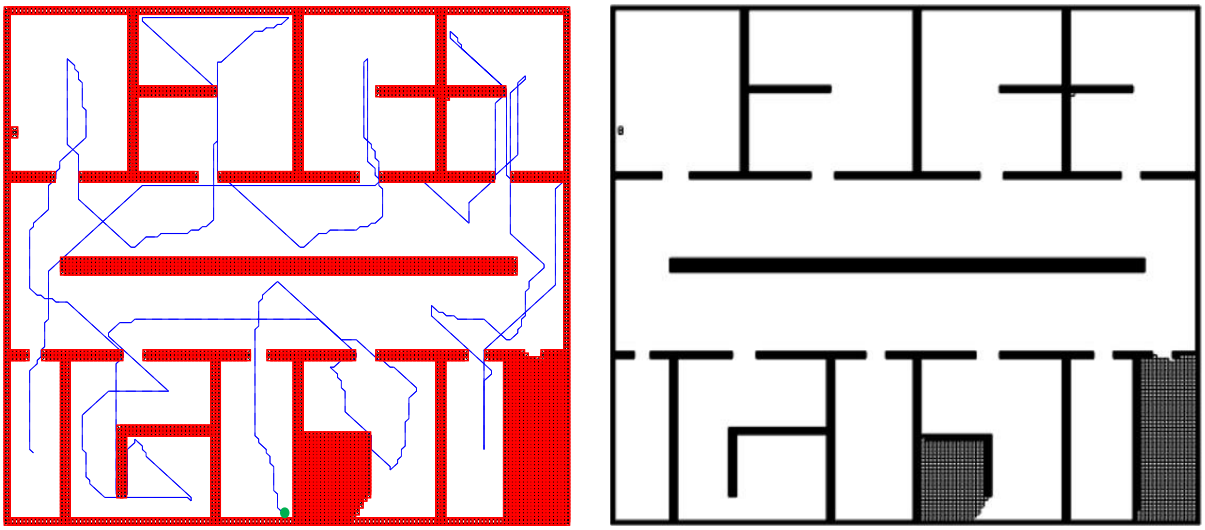
Strategije za pretragu okruženja		Srednje dužine putanja robota u okruženju *B*	
		domet senzora r=15	domet senzora r=25
VKO	*SAW*	2249.14	1625.73
	COPRAS	2121.23	1493.00
	TOPSIS	2011.27	1525.05
Dist_Min		2493.17	1703.50
GBL		2134.32	1378.85
AOJRF		2473.43	1694.00
WS		2325.25	1555.50
Napomena: Kod metoda *VKO* primenjena je kombinacija težinskih koeficijenata (0.7, 0.2, 0.1) za kriterijume *L*(*p*), *A*(*p*) i *P*(*p*), respektivno, u obe varijante dometa senzora.			



Слика 36. Графички приказ просечних (средњих) дужина путања и стандардних девијација за потребе претраге 90% окружења B применом тестираних стратегија за варијанту домета сензора r=15 и комбинацију тежинских коефицијената код ВКО (0.7, 0.2, 0.1) за критеријуме L(p), A(p) и P(p), респективно.



Слика 37. Графички приказ просечних (средњих) дужина путања и стандардних девијација за потребе претраге 90% окружења B применом тестираних стратегија за варијанту домета сензора r=25 и комбинацију тежинских коефицијената код ВКО (0.7, 0.2, 0.1) за критеријуме L(p), A(p) и P(p), респективно.

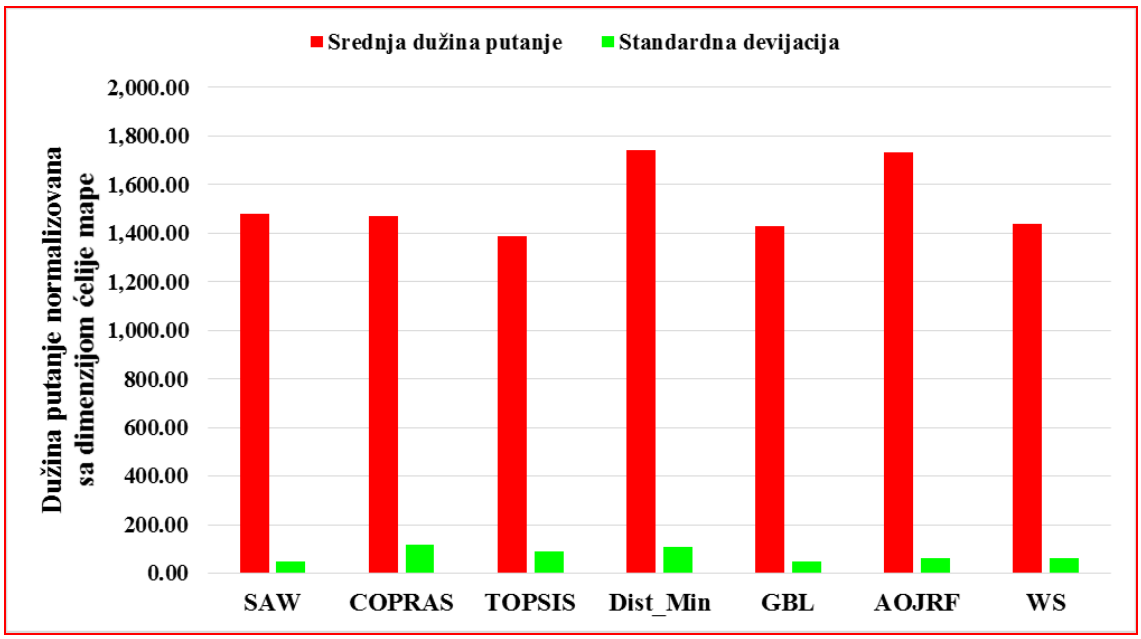


Слика 38. Путања робота генерисана D Lite алгоритмом током претраге 90% окружења B (лево) и мапа окружења након претраге (десно) применом COPRAS методе за избор следеће позиције робота уз комбинацију тежинских коефицијената (0.7, 0.2, 0.1) за критеријуме L(p), A(p) и P(p), респективно, за варијанту домета сензора r=25 и стартну позицију робота (x=75, y=2), означену зеленим кружићем на слици лево.*

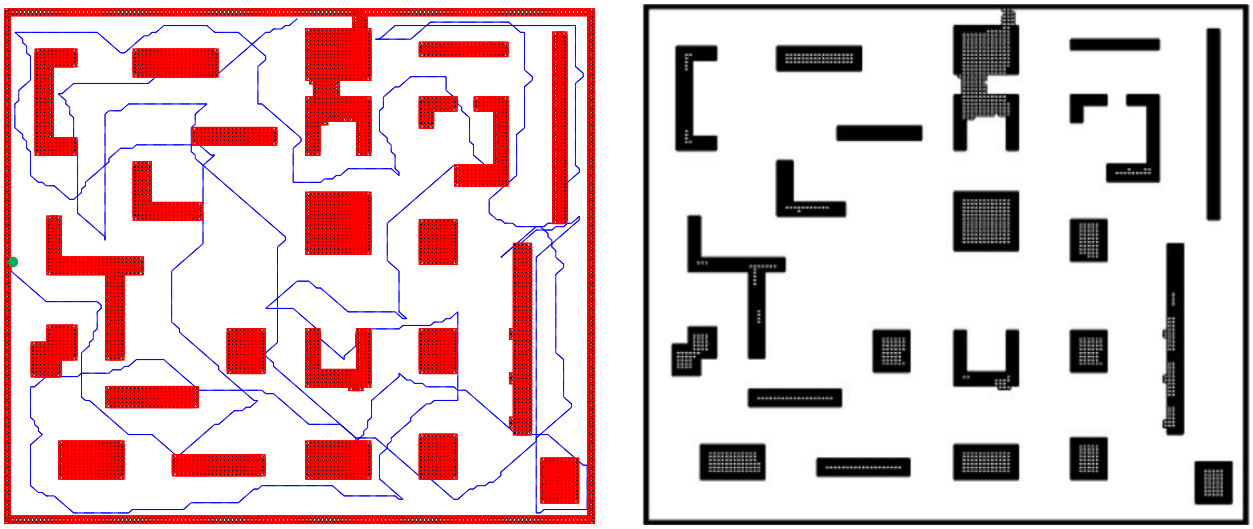
- **Окружење *C* (димензије 150x150)**

Табела 6. Просечне (средње) дужине путања робота за потребе претраге 90% окружења C применом тестираних стратегија за једну варијанту домета сензора.

Strategije za pretragu okruženja		Srednje dužine putanja robota u okruženju *C*
		domet senzora r=15
VKO	*SAW*	1480.25
	COPRAS	1470.03
	TOPSIS	1386.08
Dist_Min		1740.79
GBL		1429.30
AOJRF		1729.88
WS		1436.55
Napomena: Kod metoda *VKO* primenjena je kombinacija težinskih koeficijenata (0.7, 0.2, 0.1) za kriterijume *L*(*p*), *A*(*p*) i *P*(*p*), respektivno.		



Слика 39. Графички приказ просечних (средњих) дужина путања и стандардних девијација за потребе претраге 90% окружења C применом тестираних стратегија за варијанту домета сензора r=15 и комбинацију тежинских коефицијената код ВКО (0.7, 0.2, 0.1) за критеријуме L(p), A(p) и P(p), респективно.

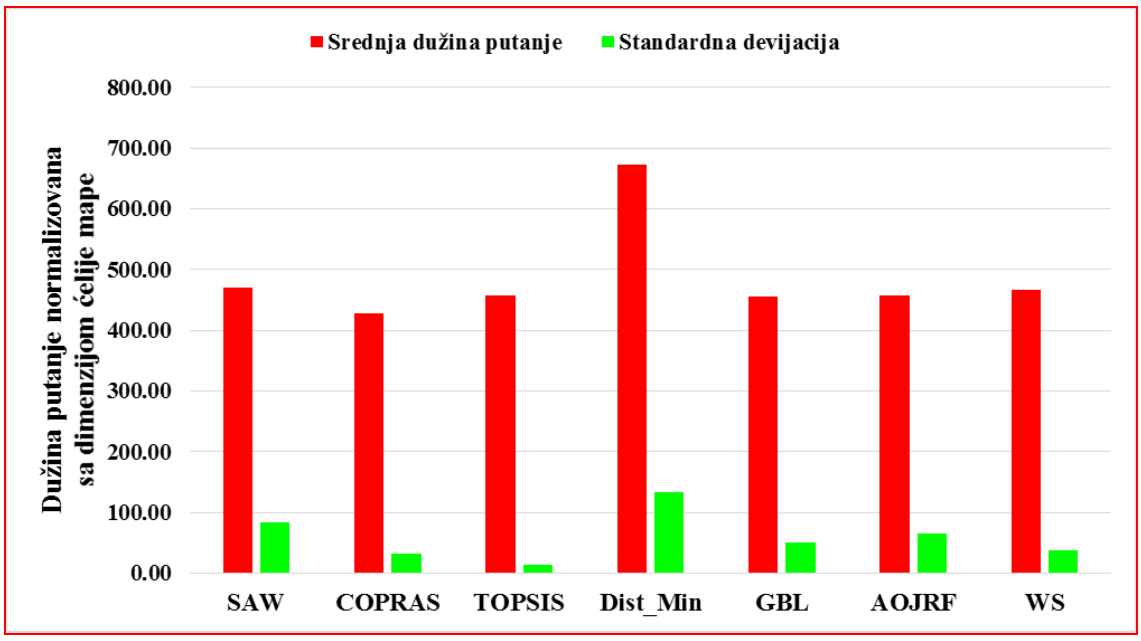


Слика 40. Путања робота генерисана D Lite алгоритмом током претраге 90% окружења C (лево) и мапа окружења након претраге (десно) применом GBL стратегије за избор следеће позиције робота, за варијанту домета сензора r=15 и стартну позицију робота (x=2, y=75), означену зеленим кружићем на слици лево.*

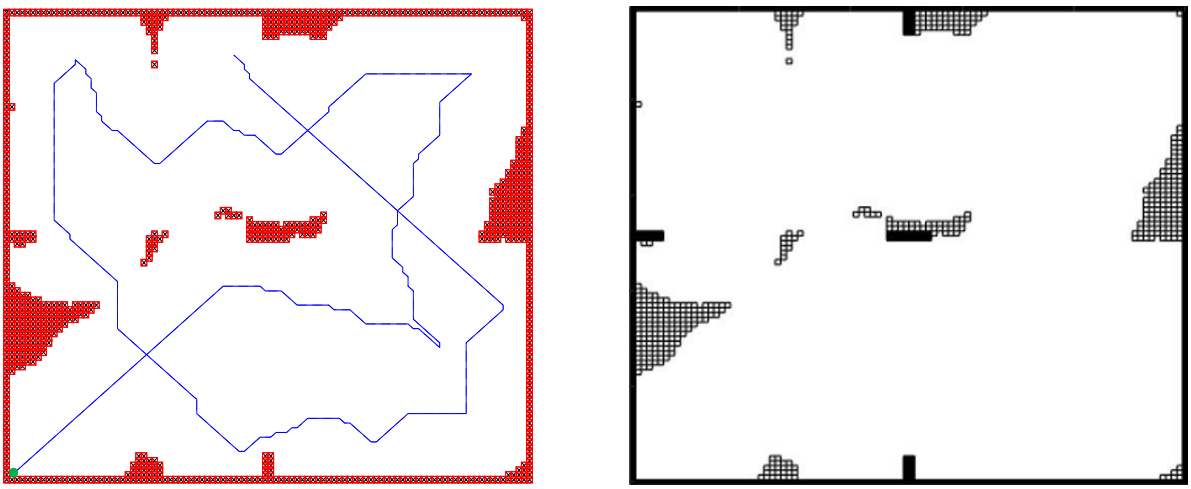
- **Окружење *D* (димензије 100x100)**

Табела 7. Просечне (средње) дужине путања робота за потребе претраге 90% окружења D применом тестираних стратегија за једну варијанту домета сензора.

Strategije za pretragu okruženja		Srednje dužine putanja robota u okruženju *D*
		domet senzora r=15
VKO	*SAW*	469.62
	COPRAS	427.87
	TOPSIS	456.82
Dist_Min		673.10
GBL		455.41
AOJRF		457.72
WS		465.98
Napomena: Kod metoda *VKO* primenjena je kombinacija težinskih koeficijenata (0.6, 0.3, 0.1) za kriterijume *L*(*p*), *A*(*p*) i *P*(*p*), respektivno.		

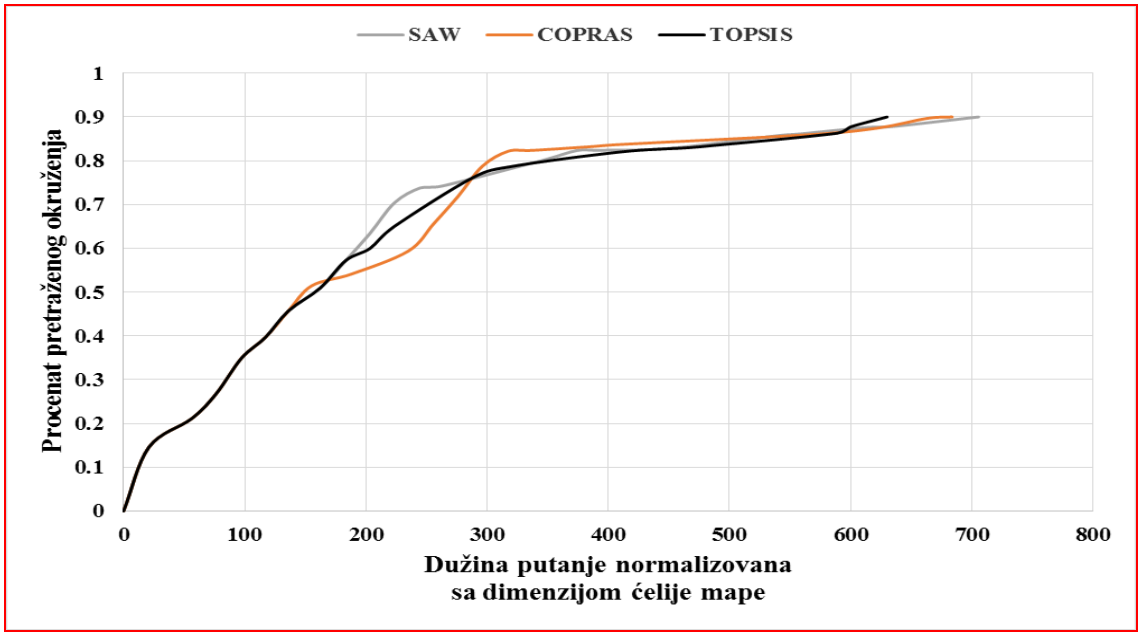


Слика 41. Графички приказ просечних (средњих) дужина путања и стандардних девијација за потребе претраге 90% окружења D применом тестираних стратегија за варијанту домета сензора r=15 и комбинацију тежинских коефицијената код ВКО (0.6, 0.3, 0.1) за критеријуме L(p), A(p) и P(p), респективно.



Слика 42. Путања робота генерисана D Lite алгоритмом током претраге 90% окружења D (лево) и мапа окружења након претраге (десно) применом WS стратегије за избор следеће позиције робота, за варијанту домета сензора r=15 и стартну позицију робота (x=2, y=2), означену зеленим кружићем на слици лево.*

- **Окружење *A* (димензије 100x100), са променом тежинских коефицијената критеријума током трајања претраге**



Слика 43. Зависност прираста претраженог дела окружења од дужине путање робота за потребе претраге 90% окружења A применом тестираних ВКО метода за избор следеће позиције робота уз промену комбинација тежинских коефицијената током трајања претраге, за варијанту домета сензора r=20 и стартну позицију робота (x=2, y=50), у односу на Декартов КС везан за доњи леви угао мапе.

Табела 8. Просечне (средње) дужине путања робота за потребе претраге 90% окружења А применом тестираних ВКО метода за избор следеће позиције робота уз промену комбинација тежинских коефицијената током трајања претраге.

Strategije za pretragu okruženja		Srednje dužine putanja robota u okruženju *A*
		domet senzora r=20
VKO	*SAW*	800.35
	COPRAS	783.45
	TOPSIS	664.38
Napomena: U prvom delu pretrage primenjena je kombinacija težinskih koeficijenata (0.2, 0.7, 0.1), a u drugom delu kombinacija (0.6, 0.3, 0.1) za kriterijume $L(p)$, $A(p)$ i $P(p)$, respektivno. Promena vrednosti težinskih koeficijenata vrši se u trenutku kada se pretraži 80% okruženja.		

4.1.5. Дискусија

Ради адекватног увода у дискусију, на почетку ће бити додатно образложене специфичности методе *ВКО TOPSIS* и изабраних критеријума за евалуацију (вредновање) позиција-кандидата робота током претраге окружења, као и опште и посебне специфичности комплексних окружења изабраних за тестирање стратегија за претрагу у дисертацији.

Специфичност *TOPSIS* методе је да све алтернативе представља у *n*-димензионалном простору (где је *n* број критеријума) тако да при одлучивању бира ону алтернативу која има најмању Еуклидову удаљеност од идеалног решења и истовремено највећу могућу Еуклидову удаљеност од анти-идеалног решења. Идеално решење је хипотетичко решење и за њега све вредности критеријума одговарају најпожељнијим вредностима критеријума у односу на све алтернативе. Анти-идеално решење је такође хипотетичко и код њега све вредности критеријума одговарају најнепожељнијим вредностима критеријума у односу на све алтернативе. Максималним удаљавањем од анти-идеалног решења минимизује се ризик у одлучивању [36]. Због наведених особина *TOPSIS* метода се често користи у сферама људске делатности када се при одлучивању тежи максималном бенефиту, уз истовремено максимално могуће избегавање ризика (нпр. у економији).

Другим речима, *TOPSIS* настоји да при одлучивању максимизује критеријуме бенефитног типа и минимизује критеријуме трошковног типа, али тако да истовремено тежи да изабрана алтернатива има уједно што мању Еуклидову удаљеност од идеалног решења и што већу Еуклидову удаљеност од анти-идеалног решења. На тај начин смањује се ризик избора неке алтернативе на рачун тога што већина критеријума има релативно повољне вредности, али чија нпр. вредност једног критеријума има у значајнијој мери лошију вредност. Код других анализираних метода *ВКО,* као и код класичних стратегија, може се десити да се на рачун повољних вредности осталих критеријума изабере алтернатива која има у значајнијој мери лошу вредност једног критеријума, а пошто у примењеном моделу претраге окружења два од три критеријума непосредно утичу на

ефикасност претраге, онда постоји тенденција да је донета одлука у овом случају лошијег квалитета него када је у питању *TOPSIS*.

Од изабраних критеријума за евалуацију (вредновање) позиција-кандидата робота током претраге окружења - „дужина путање до позиције-кандидата“ ($L(p)$), „информативни потенцијал позиције-кандидата“ ($A(p)$) и „удаљеност позиције-кандидата од базне станице“ – ($P(p)$), посебно су карактеристична прва два ($L(p)$ *и* $A(p)$), пошто директно утичу на ефикасност претраге (што се не може рећи за трећи критеријум изабран у дисертацији, али ни за друге критеријуме који се предлажу у литератури, као што је нпр. преостали капацитет батерије робота, тип објеката у посматраном делу окружења, итд, при чему треба нагласити да је сваки од њих битан са неког аспекта и користи се у зависности од конкретне ситуације).

Истовремено, кључна карактеристика комплексних окружења (окружења са великим бројем препрека) је да се позиције-кандидати (алтернативе при одлучивању) значајно разликују управо у погледу вредности критеријума „информативни потенцијал позиције-кандидата“ и „дужина путање до позиције-кандидата“, због чега је у овом случају изражен ризик од лошег избора следеће позиције робота. Те разлике су узроковане присуством великог броја препрека које на различите начине ограничавају опажање окружења сензором робота (*Слика 28*), при чему то ограничење зависи и од тога колико је робот близу препрекама [12]. Присуство препрека компликује и планирање путање робота од једне до друге позиције захтевајући доста заобилажења и маневара промене праваца.

На бази претходно наведеног, као и презентованих резултата тестирања, може се закључити да *TOPSIS* при претрази комплексних окружења, уз изабране критеријуме за одлучивање $L(p)$ *и* $A(p)$), минимизује ризик лошег избора и у просеку бира бољу алтернативу у односу на друге две анализиране методе *ВКО - SAW* и *COPRAS*, као и у односу на остале класичне стратегије за претрагу окружења анализиране у овој дисертацији (*Dist_Min*, *GBL*, *AOJRF*, *WS*), и тиме обезбеђује да се претрага реализује за краћи укупан пређени пут робота. Овоме доприноси и чињеница да комплексна окружења захтевају велики број сукцесивних одлучивања (корака претраге) са великим бројем алтернатива у сваком кораку, због чега долази до акумулирања бенефита у просеку бољег одлучивања.

Од значаја за детаљнију дискусију резултата је и чињеница да „*indoor*“ окружења *A*, *B* и *D* представљају средине са препрекама (зидовима) које ограничавају кретање робота, дакле ради се о комплексним окружењима, али су различитог нивоа комплексности. Окружење *D* карактерише само 5 полупреградних зидова те је из тог разлога оно значајно мањег нивоа комплексности у односу на окружења *A* и *B*. Ако поредимо окружења *A* и *B*, може се приметити специфичност окружење *A* која се огледа у томе да садржи две просторије које су значајно веће у односу на остале.

Као што се може констатовати из презентованих резултата, током тестирања вршена је и промена домета сензора (r). Домет сензора директно пропорционално утиче на вредност критеријума „информативни потенцијал позиције-кандидата“, тако да је ова промена рађена ради следећих анализа:

- Да ли је промена r, у циљу његовог повољнијег односа у поређењу са димензијама окружења, довољан услов за примену агресивније стратегије, независно од структуре окружења;
- Да ли промена r утиче на ранг тестираних стратегија по ефикасности претраге у дефинисаним окружењима.

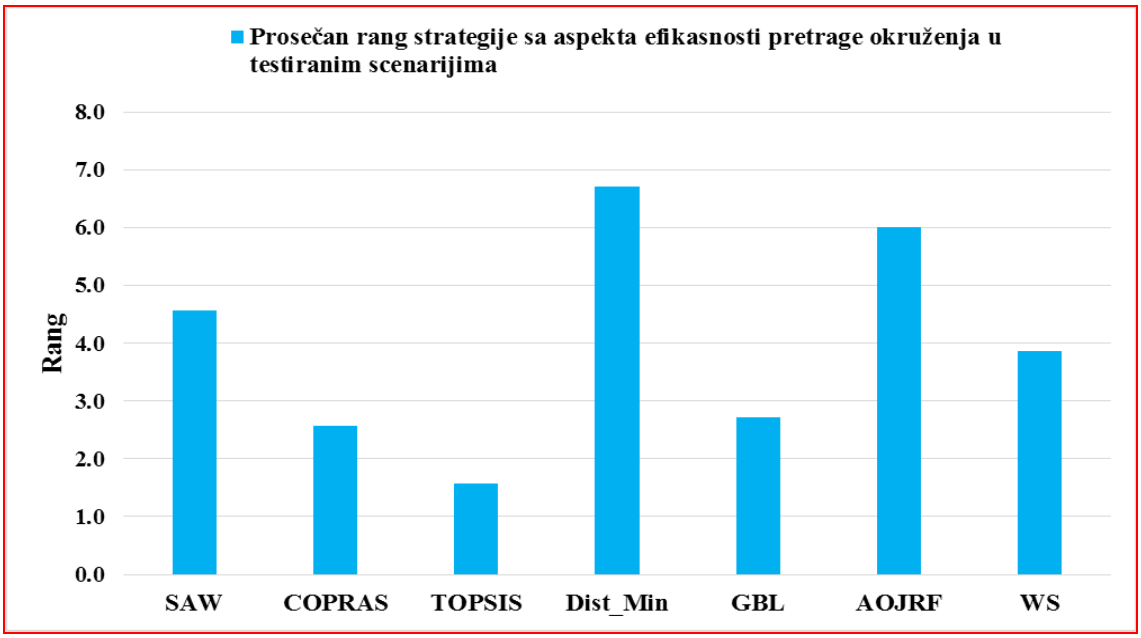
Изабрано је да се промена *r* врши тако да је домет сензора максимално 10 пута мањи од димензија окружења.

Ранг стратегија за претрагу окружења према презентованим резултатима у *Поглављу 4.1.4*, тј. са аспекта ефикасности претраге за чију меру је дефинисан просечан укупан пређени пут робота у односу на више стартних позиција за потребе реализације претраге 90% окружења, у свим тестираним сценаријима, приказан је у *Табели 9*.

Табела 9. Ранг стратегија за претрагу окружења према ефикасности у свим тестираним сценаријима.

Rang	Okruženje *A*			Okruženje *B*		Okruženje *C*	Okruženje *D*
	r=10	*r*=15	*r*=20	*r*=15	*r*=25	*r*=15	*r*=15
1	*TOPSIS*	*TOPSIS*	*TOPSIS*	*TOPSIS*	*GBL*	*TOPSIS*	*COPRAS*
2	*COPRAS*	*GBL*	*WS*	*COPRAS*	*COPRAS*	*GBL*	*GBL*
3	*SAW*	*WS*	*COPRAS*	*GBL*	*TOPSIS*	*WS*	*TOPSIS*
4	*GBL*	*COPRAS*	*SAW*	*SAW*	*WS*	*COPRAS*	*AOJRF*
5	*WS*	*SAW*	*GBL*	*WS*	*SAW*	*SAW*	*WS*
6	*Dist_Min*	*Dist_Min*	*AOJRF*	*AOJRF*	*AOJRF*	*AOJRF*	*SAW*
7	*AOJRF*	*AOJRF*	*Dist_Min*	*Dist_Min*	*Dist_Min*	*Dist_Min*	*Dist_Min*

Из *Табеле 9.* се може закључити да је *TOPSIS* постигла најбоље резултате у односу на све конкурентне стратегије у 5 од 7 тестираних сценарија. У једном сценарију најбоље резултате је имала *GBL* стратегија, а у једном *COPRAS*. Просечан ранг стратегија са аспекта ефикасности претраге окружења у тестираним сценаријима, графички је приказан на *Слици 44*.



Слика 44. Просечан ранг стратегија у тестираним сценаријима. TOPSIS метода има најбоље резултате у 5 од 7 сценарија, затим следе COPRAS метода и GBL. Најлошије су рангиране стратегије AOJRF и Dist_Min.

Анализирајући *Слику 44.* може се закључити да после *TOPSIS* по ефикасности следе стратегије *COPRAS* и *GBL*, затим *WS* и *SAW*, док су најлошије резултате у тестираним сценаријима имале стратегије *AOJRF* и *Dist_Min*.

У окружењу *D*, *TOPSIS* је трећа по рангу остварених резултата. Ово се образлаже тиме да *D* представља окружење мале комплексности, тако да не долазе до изражаја бенефити бољег одлучивања, који су као што је претходно наведено карактеристични за примену *TOPSIS* код претраге комплексних окружења у комбинацији са изабраним критеријумима за евалуацију позиција-кандидата. С тим у вези, анализирајући *Слику 41.* и *Табелу 7*, може се приметити да у случају претраге окружења *D* све тестиране стратегије постижу упоредиве резултате, изузев *Dist_Min*.

Образложење зашто *TOPSIS* није дала најбоље резултате при претрази окружења *B* за варијанту домета сензора r=25 је везано за другу битну карактеристику комплексних окружења, а то је изражен ризик од грешке у процени информативног потенцијала позиција-кандидата. Уколико је домет сензора већи, већа је и вредност критеријума $A(p)$), а самим тим је и већи ризик од грешке у његовој процени (услед присуства препрека у комплексним окружењима), што је објашњено детаљније у *Поглављу 4.1.2.* и приказано на *Слици 28.* Овај ризик је посебно изражен на почетку претраге. Са повећањем истраженог дела окружења грешка у процени информативног потенцијала позиција-кандидата се смањује.

Даље, резултати презентовани у *Табелама 4, 5, 6.* и *7.* у *Поглављу 4.1.4.* показују да се оптимални баланс између критеријума код сваке тестиране методе *ВКО* током претраге постиже са комбинацијом тежинских коефицијената (0.6, 0.3, 0.1) или (0.7, 0.2, 0.1), за критеријуме „дужина путање до позиције-кандидата“, „информативни потенцијал позиције-кандидата“ и „удаљеност позиције-кандидата од базне станице“, респективно.

При томе, агресивнија стратегија (0.6, 0.3, 0.1) препоручује се само ако су испуњена два услова: да је однос димензија окружења и домета сензора робота мањи од 10 и ако се процени да је структура окружења таква да може донети значајно већи бенефит по питању критеријума „информативни потенцијал позиције-кандидата“. На овакав закључак упућују резултати тестирања према којима је током претраге окружења *B* и поред повећања домета сензора на r=25 (однос димензија окружења и домета сензора је 6) најбоље резултате свака тестирана *ВКО* метода имала са комбинацијом тежинских коефицијената (0.7, 0.2, 0.1). За окружење *A* се веже специфичност да има такву структуру (садржи две просторије које су значајно веће у односу на остале) да може донети значајно већи бенефит по питању информативног потенцијала [12], тако да за варијанте домета сензора r=15 (однос димензија окружења и домета сензора је 6,7) и r=20 (однос димензија окружења и домета сензора је 5) агресивнија стратегија даје боље резултате.

Што се тиче генералног утицаја промене домета сензора робота на ранг стратегија по ефикасности претраге, што је тестирано у окружењима *A* и *B*, из *Табела 4.* и *5*, као и са *Слика 29, 31, 33, 36.* и *37*, видимо да је и по том параметру *TOPSIS* најбољи у смислу да је најимунији на промену вредности r. Другим речима, описана предност коју *TOPSIS* у односу на друге стратегије испољава у претрази комплексних окружења у комбинацији са изабраним критеријумима не зависи од промене домета сензора робота.

Дакле, у тестираним сценаријима комбинације са повећањем тежинског коефицијента критеријума „информативни потенцијал позиције-кандидата“ изнад вредности 0.3 и истовремено смањењем тежинског коефицијента критеријума „дужина путање до позиције-кандидата“ испод 0.6 имају за анализиране *ВКО* стратегије на почетку

претраге постижу добре резултате, али касније, форсирајући стално робота да бира позиције са већим информативним потенцијалом, овај приступ доводи до тога да укупан пређени пут почиње убрзано да расте, имајући у виду да се претражени део простора константно увећава па су следеће позиције све удаљеније од тренутне.

Већ је наведено да је једна од основних предности стратегија базираних на *ВКО* то што пружају широк и флексибилан приступ у одабиру критеријума и њихових тежина који се могу користити за евалуацију позиција-кандидата у циљу избора следеће позиције робота, чиме се обезбеђује ефикасно управљање претрагом у зависности од услова у којима се одвија и њених циљева. С тим у вези, у неким сценаријима постоји потреба да се што већи део окружења претражи за што краће време. На пример, ако је ограничено време за претрагу, можемо на почетку да дозволимо роботу привилегију да бира удаљеније позиције ако процени да оне имају већи информативни потенцијал. Како истиче време претраге битније је да то време робот не троши на прелазак већих растојања, већ да претражује у локалу, па се стога повећава значај минимизације критеријума „дужина путање до позиције-кандидата“. У тим ситуацијама пожељно је применити у старту екстремно агресивну стратегију претраге (вредност тежинског коефицијента критеријума $A(p)$ је значајно већа од вредности тежинског коефицијента критеријума $L(p)$), која ће форсирати робота да бира позиције са већим информативним потенцијалом, мање водећи рачуна о њиховој удаљености од тренутне позиције. Како време истиче ова стратегија даје све лошије резултате, јер се претражени део окружења увећава, тако да робот прелази све већа растојања (тежећи да и даље обилази позиције са већим информативним потенцијалом). Стога је пожељно у неком тренутку променити вредност тежинских коефицијената критеријума $A(p)$ и $L(p)$, како би се у даљем току претраге применила мање агресивна стратегија.

Овај концепт је тестиран на примеру претраге окружења *A*. У првом делу претраге примењена је комбинација тежинских коефицијената (0.2, 0.7, 0.1), а у другом делу комбинација (0.6, 0.3, 0.1). Промена вредности тежинских коефицијената врши се у тренутку када се претражи 80% окружења.

Са *Слика 34.* и *43.* може се приметити да је роботу у овом случају потребно око 40% краћа путања да претражи 80% окружења, него када је током целе претраге примењивана стратегија са комбинацијом тежинских коефицијената (0.6, 0.3, 0.1). Међутим, средња укупна дужина путања (израчуната у односу на исте стартне позиције) је већа у случају промене тежинских коефицијената критеријума током трајања претраге него у случају када су вредности коефицијената константне, што се може констатовати поредећи резултате из *Табела 4.* и *8.*

Кључно ограничење предложеног приступа са применом *TOPSIS* методе за избор следеће позиције робота при претрази окружења проистиче из претходно дела текста у овом поглављу и огледа се у следећем:

- У случају избора других критеријума за евалуацију позиција-кандидата и/или претраге окружења мале комплексности (са малим бројем препрека) не може се очекивати да ће *TOPSIS* дати боље резултате у односу на друге стратегије за претрагу окружења анализиране у овој дисертацији.

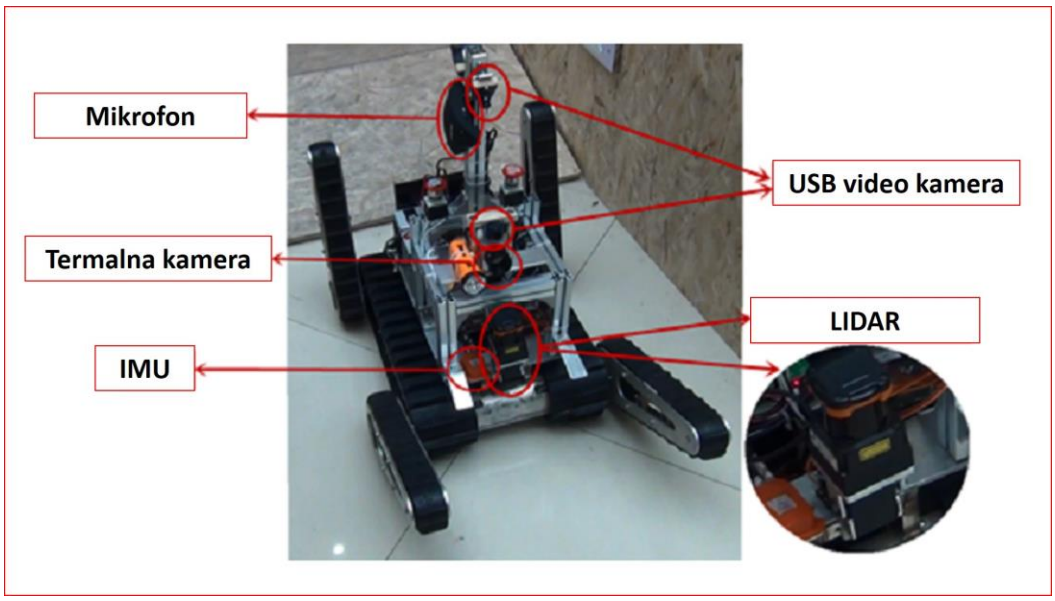
4.2. Приступи за унапређење планирања путање робота у мисијама претраге окружења за потребе трагања и спасавања у урбаном окружењу

Последњих година све већа пажња посвећује се роботима за трагање и спасавање, како од стране истраживача у области роботике, тако и од потенцијалних корисника ових робота [7,98]. Роботи могу помоћи људима у задацима мисија трагања и спасавања на неколико начина. Једна од основних улога мобилних робота у овим мисијама је да раде као уређаји за даљинско детектовање којим се прикупљају корисни подаци са опасних места до којих људи не могу лако и/или безбедно доћи. Посебан изазов данас представља ангажовање у мисијама трагања и спасавања мобилних робота са одређеним степеном аутономије. Аутономни и полуаутономни роботи могу да обрађују прикупљене податке самостално на одређеном нивоу. Полуаутономни роботи, за разлику од аутономних, функционишу кроз ограничену интеракцију са човеком оператором. Овај концепт омогућава човеку да контролише више робота тако што даје команде високог нивоа, на пример, „истражи ову област“, „достигни ову тачку“ [7], итд. У случају привременог квара у комуникацији, мобилне роботске платформе обично наставе са извршавањем текућег задатка, након чега се врате на унапред дефинисану основну позицију [99,100].

У мисијама трагања и спасавања могуће је ангажовање једног робота, који тада мора користити различите сензорске уређаје, како би извршио мисију [7]. На пример, мора имати сензоре за опажање окружења у циљу формирања његове мапе. Са друге стране, откривање повређених лица захтева сензоре друге врсте, који су обично краћег домета од претходно поменутих. Због наведених специфичности, уместо једног робота у мисијама трагања и спасавања обично се ангажују вишероботски системи хетерогеног састава, где сваки робот има свој задатак.

Аутономни роботи који обављају задатак претраге окружења за потребе трагања и спасавања морају бити у стању да формирају мапу окружења и да истовремено одређују свој положај у односу на ту мапу (*Simultaneous Localization and Mapping, SLAM*), али и да одлучују у складу са примењеном стратегијом за претрагу окружења коју ће следећу позицију да посете, као и да планирају путању до изабране позиције [7].

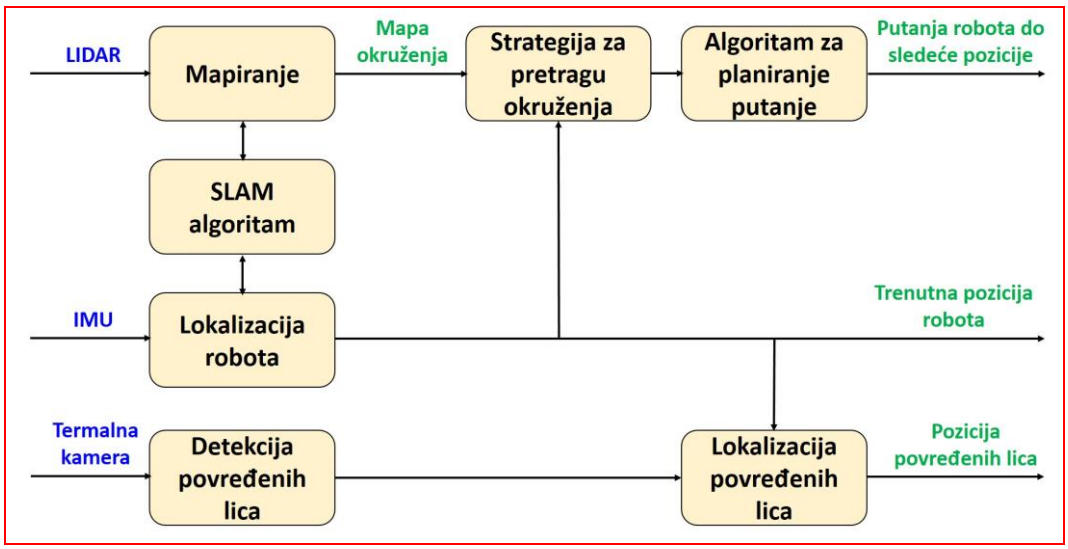
На *Слици 45.* је приказан пример аутономног робота за мисије трагања и спасавања. Ради се о роботу под називом *NuBot*, којег је креирао тим из састава *National University of Defense Technology* из Кине [98]. Поменути тим је са овим роботом освојио 1. место у класи малих робота на такмичењу *RoboCup Rescue Robot League* (*RRL*) 2016. године у *Лајпцигу* (*Немачка*), као и прво место на такмичењу *RoboCup China Open RRL* 2017. године.



Слика 45. Опремљеност сензорима NuBot-а, аутономног робота за трагање и спасавање, модификована слика из [98].

NuBot је опремљен ласерским скенером (*LIDAR*), инерцијалним мерним сензором (*IMU*), микрофоном, *USB* камерама и термалном камером.

Блок-шема софтверске архитектуре *NuBot*-а приказана је на *Слици 46*.



Слика 46. Блок-шема софтверске архитектуре NuBot-а, аутономног робота за трагање и спасавање, модификована слика из [98].

NuBot је израђен на гусеничној платформи, имајући у виду да ове платформе у односу на точкашке омогућавају већу проходност по тешком терену, који је посебно карактеристичан за мисије трагања и спасавања у урбаном окружењу (*urban search and rescue, USAR*).

4.2.1. Примена *D* Lite* алгоритма у комбинацији са фази логиком

Алгоритми који прорачун путање робота врше на бази обраде графа, као што је *D* Lite*, погодни су за примену различитих метода за дефинисање тежина ћелија одговарајуће мапе (*cost maps*), како би се унапредио процес планирања. *Cost maps* технике посебно пружају могућност реалистичног осликавања сложености терена са аспекта проходности тј. могућности кретања робота, што обезбеђује већу поузданост у генерисању путање у смислу да неће захтевати озбиљније накнадне корекције.

У [37] је представљен концепт дефинисање тежина ћелија на бази вероватноће, која узима у обзир непотпуно познавање карактеристика терена. Робот у току кретања упознаје боље терен и по потреби коригује претходно израчунату путању. У циљу унапређења планирања путање робота, у [38] је описан приступ који укључује дефинисање тежина ћелија на бази познатих статичких препрека, као и на бази структуре путева у окружењу и других доступних информација о терену које могу утицати на кретање робота. Динамичке препреке су представљене као региони одређених димензија и облика са ћелијама великих тежина како би се осигурала безбедна удаљеност робота.

Посебан изазов представља дефинисање тежина ћелија у мапи која се користи за планирање путање робота у окружењу где су присутни људи. С тим у вези, у [39] је презентован приступ по коме се у фази предпланирања на основу иницијалне мапе генеришу путање робота, истовремено се симулирају путање пешака (на основу претходних посматрања) и на основу тога се дефинишу коначне тежине ћелија. У тако ажурираној мапи се планирају коначне путање робота. У [40] и [41] су представљена четири алгоритма базирана на *A** алгоритму за планирање путање робота у окружењу где су присутни људи (*CSA**, *Flow-A**, *Risk-A** и *CUSUM-A**). На почетку, када робот нема информацију о присуству људи у окружењу, сваки од ових алгоритама генерише путање које су идентичне решењима оригиналног *A** алгоритма. Међутим, када робот започне кретање и сукцесивно упознаје окружење, врши се ажурирање тежина ћелија у мапи. Када се прикупи довољно информација и ажурира се мапа, *CSA** обезбеђује планирање путања које једноставно избегавају делове окружења са великим бројем људи. Са друге стране, *Flow-A** у прорачун укључује и правце кретања група људи како би генерисао путању која избегава те правце. Међутим, људи обично мењају своје кретање у присуству робота. У таквим сценаријима, на бази евентуалних ризичних контакта са људима *Risk-A** користи моделе за планирање безбеднијих путања. *CUSUM-A** прати промене у обрасцима кретања већих група људи и према њима прилагођава путање робота.

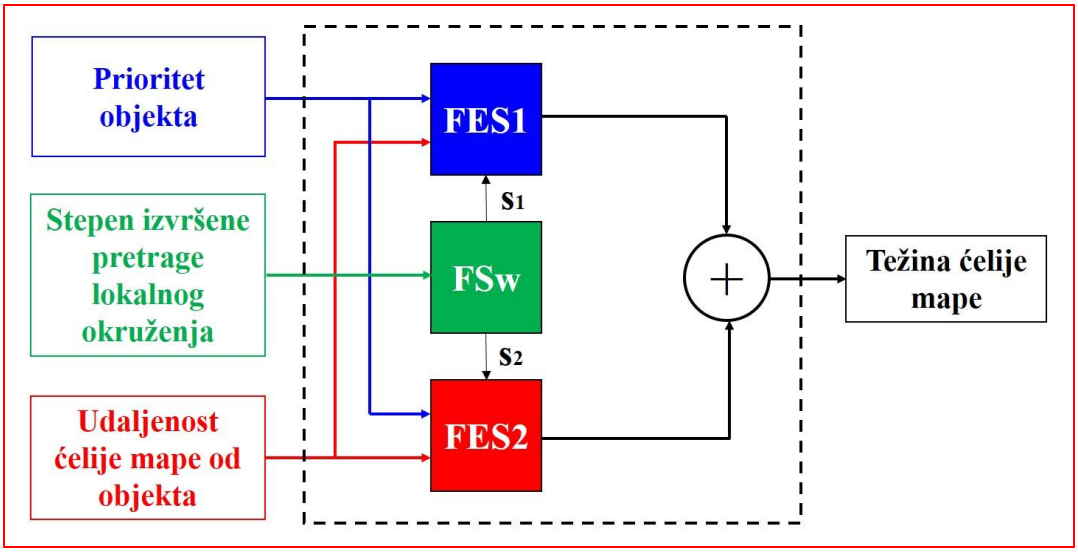
У мисијама претраге окружења за потребе трагања и спасавања честе су ситуације да је на почетку само делимично познато окружење. На пример, иако постоји детаљна мапа неког терена може доћи до акцидента при којем је дошло до делимичног урушавања објеката, што је посебно карактеристично за урбано окружење. У таквим, комплексним сценаријима, обично се у подручја погођена акцидентом упућују истовремено и роботи за мапирање, али и роботи и људи за реализацију других задатака, нпр. за евакуацију повређених, уклањање препрека, неутралисање опасних материја [7], итд. Истовремено, могући су сценарији да се у зони од интереса за мисију трагања и спасавања налазе објекти различите намене (насељени објекти, складишта опасних материја, напуштени

објекти и сл.), који су услед тога и различитог приоритета за претрагу. У циљу управљања претрагом окружења у оваквим условима, а за потребе ефикасне реализације мисије трагања и спасавања, у дисертацији се предлаже примена фази логике за дефинисање тежина ћелија мапе која се користи за планирање путања робота. У склопу истраживања која су претходила припреми за израду дисертације, аутор дисертације је публиковао рад у коме је комбинована употреба *D* Lite* алгоритма и фази логике примењена за унапређење планирања кретања робота у сценарију продајног центра [61].

У оквиру дисертације, главна идеја примене фази логике је да се обезбеди претрага окружења на такав начин и тим редом да „покрива“ објекте према приоритету, од највећег ка најмањем, и то тако да роботи за мапирање у што мањој мери ометају друге роботе или људе, који због природе својих задатака обично морају прићи ближе објектима (такође „покривајући“ објекте према истом редоследу приоритета). Стога се уводи додатни параметар – степен реализације локалне претраге, који ће са једне стране да онемогући удаљавање робота за мапирање пре извршавања задатка из зоне једног објекта у зону другог објекта (према њиховом приоритету), а са друге стране ће да регулише ниво пенализације његових путања које пролазе близу објекта на директно пропорционалан начин (ниво пенализације се повећава са порастом степена реализације локалне претраге).

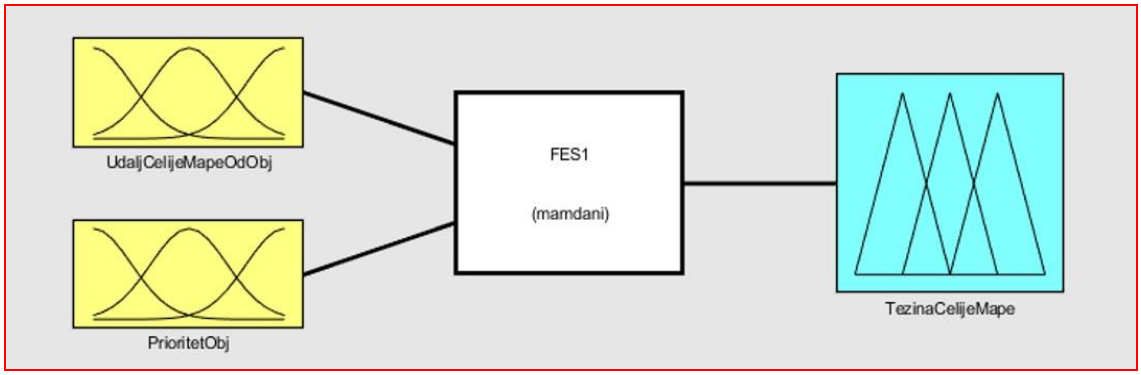
Сходно наведеном, фази систем треба да регулише дефинисање тежина ћелија мапе која се користи за планирање кретања, тако да тежина ћелија зависи од три фактора: врсте објекта односно његовог приоритета за претрагу, удаљености ћелије од објекта и степена реализације претраге локалног окружења око тог објекта. Пошто је једноставније подесити фази систем са два него са три улаза, у конкретном случају предлаже се употреба тзв. фази интегралног система [101]. Он се састоји од два фази експертска система и фази свича (*Слика 47*).

Оба фази система имају исте улазе и излазе, с тим што један систем има мању осетљивост промене тежине ћелија мапе, а други већу осетљивост. Фази свич има један улаз и два излаза и он одређује који фази систем у ком тренутку има већи утицај на формирање коначне тежине ћелије.

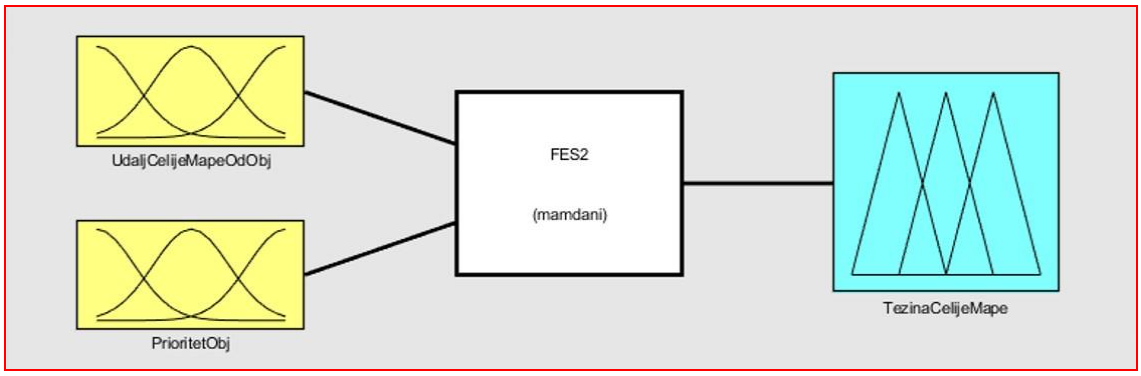


Слика 47. Блок шема фази интегралног система.

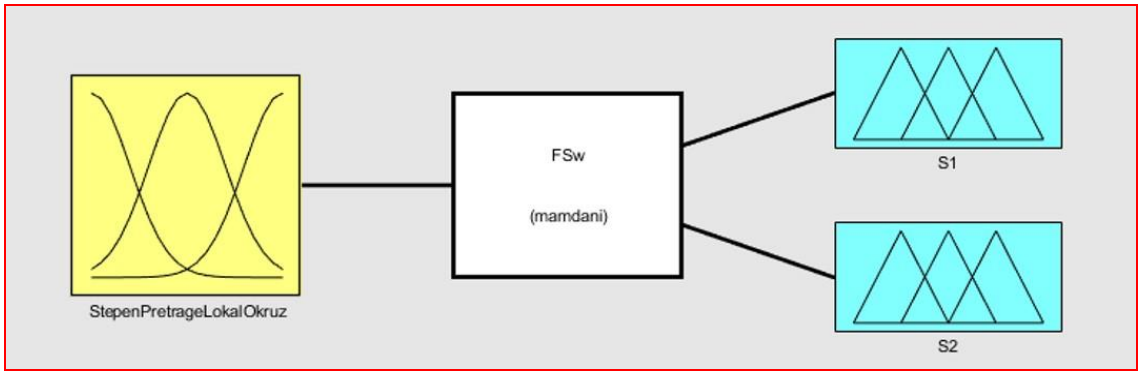
Одговарајуће блок шеме фази експертских система и фази свича имплементиране у Матлабу приказане су на *Слици 48.*



а) Фази експертски систем 1



б) Фази експертски систем 2

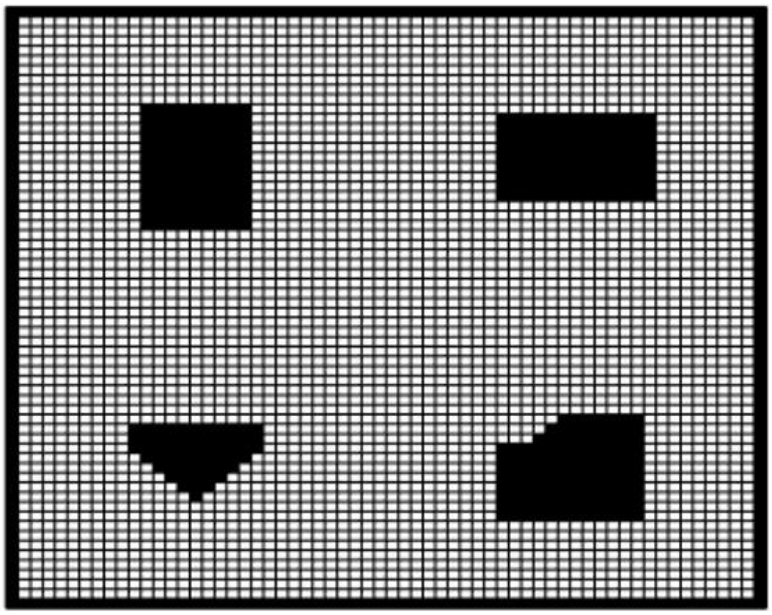


ц) Фази свич

Слика 48. Блок шеме фази експертских система и фази свича имплементиране у Матлабу у оквиру истраживања обухваћених дисертацијом

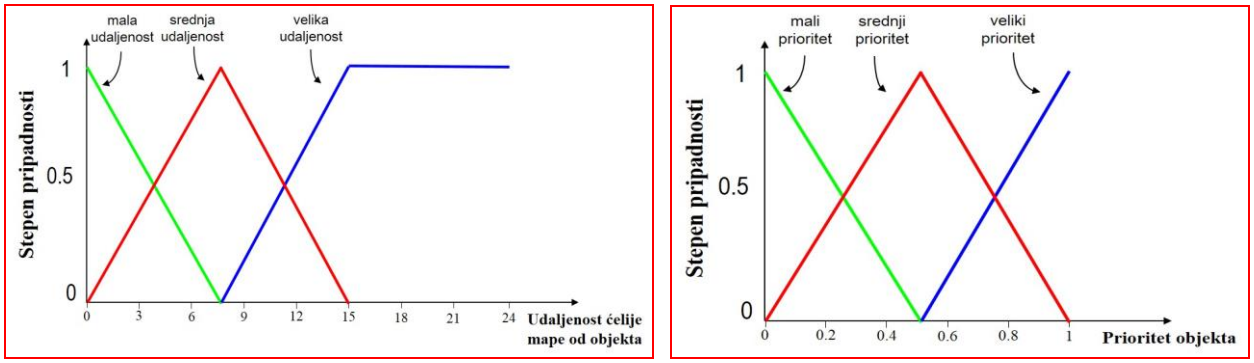
Предложени фази интегрални систем је тестиран на симулираном сценарију. Полазна претпоставка је да се мисија трагања и спасавања спроводи у окружењу са 4 објекта чија иницијална мапа је позната (*Слика 49*), али није познато у ком степену и на који начин је акцидент урушио објекте и изменио иницијалну мапу (због чега је претходно слободан простор на слици приказан шрафирано), те је из тог разлога неопходно да се претражи окружење како би се формирала његова мапа након акцидента.

Даље, претпоставимо да бројчано изражен ниво приоритета објеката, идући од објекта горе лево, преко објекта горе десно и објекта доле десно до објекта доле лево, износи 0.9, 0.6, 0.1 и 0.05, респективно.



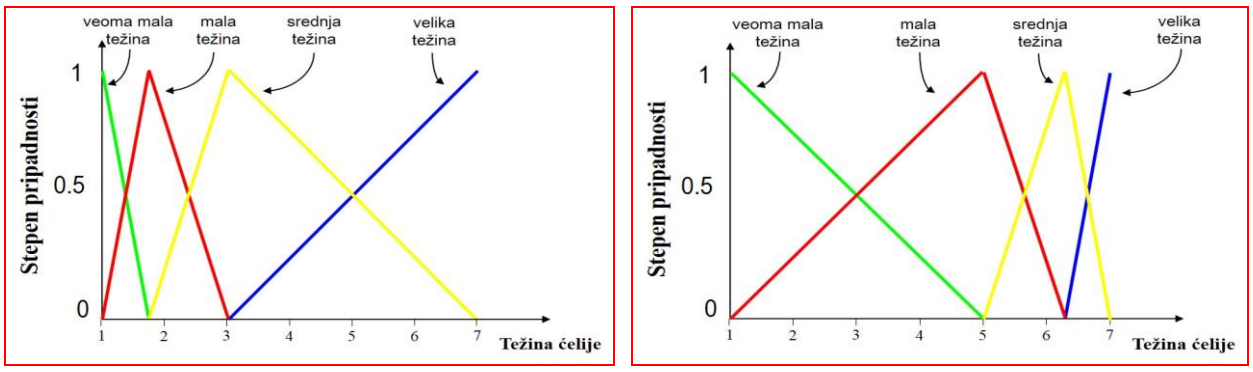
Слика 49. Иницијална мапа окружења у тестираном сценарију мисије трагања и спасавања.

Функције припадности улазних варијабли и излазне варијабле фази експертских система приказане су на *Сликама 50.* и *51.*



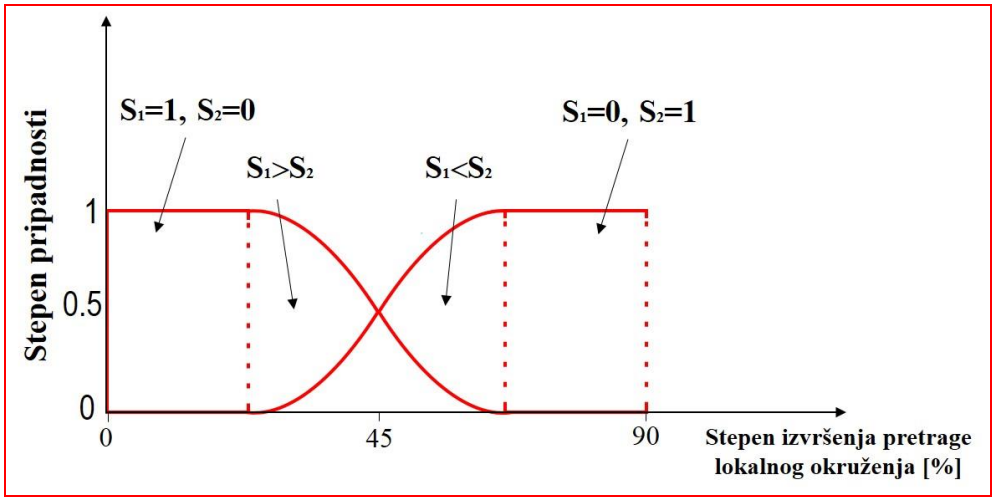
Слика 50. Функције припадности улазних варијабли фази експертских система.

Како би се постигло да један фази систем (*FES1*) има мању осетљивост промене тежине ћелија у односу на улазне варијабле (приоритет објекта и удаљеност ћелије мапе од објекта), а други фази систем (*FES2*) већу осетљивост, центроид функција припадности је у првом случају померен лево ближе нули, а у другом случају је транслиран више десно (*Слика 51*) [101].



Слика 51. Функције припадности излазне варијабле фази експертских система (лево – FES1, десно – FES2).

Функција припадности улазне варијабле фази свича приказана је на *Слици 52*.



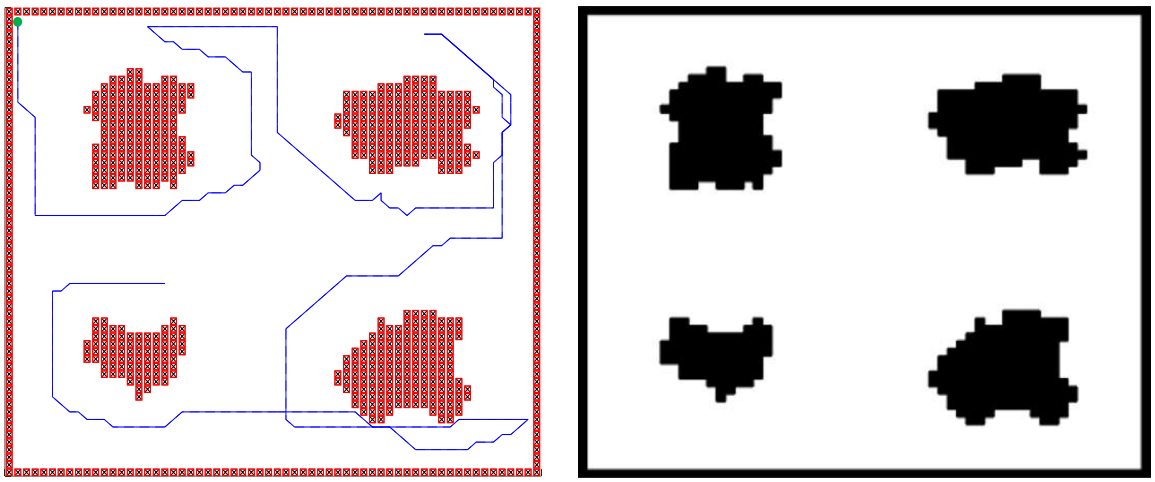
Слика 52. Функција припадности улазне варијабле фази свича.

Фази правила за одређивање тежина ћелија мапе која се користи за планирање путање робота дата су у *Табели 10*.

Табела 10. Фази правила за одређивање тежина ћелија мапе.

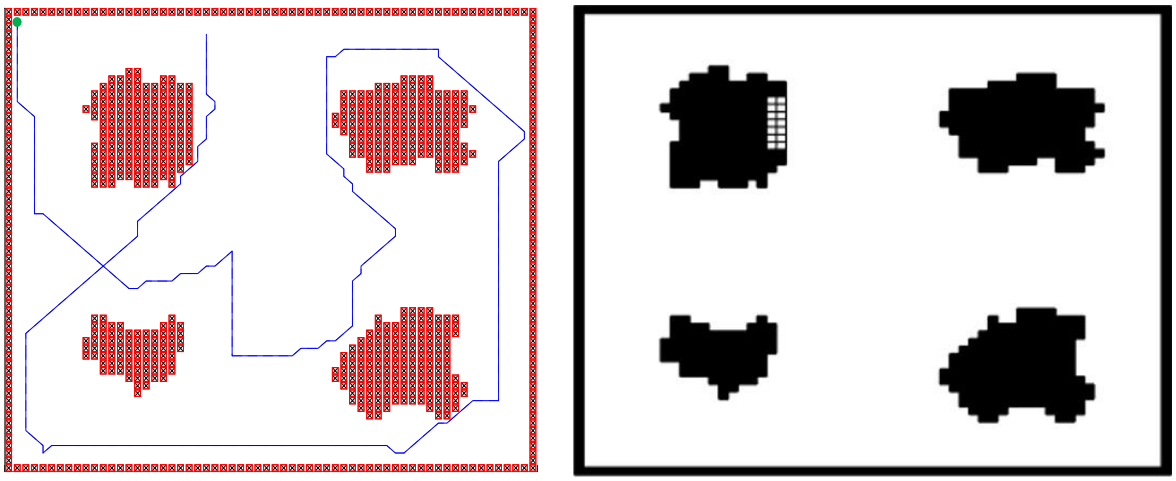
Težina ćelije mape (fazi izlaz)		Prioritet objekta (fazi ulaz 1)		
		mali	*srednji*	*veliki*
Udaljenost ćelije mape od objekta (fazi ulaz 2)	*blizu*	mala	srednja	velika
	srednje	mala	mala	srednja
	daleko	veoma mala	mala	mala
Tip fazi pravila: „∧“				

Резултати симулације приказани су на *Сликама 53.* и *54.* На *Слици 53.* је приказан случај када је примењен предложени фази интегрални систем за управљање претрагом окружења у мисији трагања и спасавања. Може се приметити да је у овом случају при реализацији претраге окружења испоштован приоритет објеката, и то на начин да робот за мапирање већим делом путање не прилази превише близу објектима, посебно не објектима великог приоритета.



Слика 53. Путања робота током претраге (лево) и мапа окружења формирана након претраге (десно) уз примену фази интегралног система за дефинисање тежина ћелија мапе. У овом случају тежине ћелија имају шири распон вредности, тако да D Lite алгоритам генерише путању најмање цене. Стартна позиција робота је означена зеленим кружићем на слици лево.*

У другом случају, када није примењен фази интегрални систем (*Слика 54*), при реализацији претраге окружења није испоштован приоритет објеката.



Слика 54. Путања робота током претраге (лево) и мапа окружења формирана након претраге (десно) без примене фази интегралног система. У овој опцији тежине ћелија имају бинарне вредности, тако да D Lite алгоритам увек генерише путању најмање дужине чиме се губи флексибилност у управљању мисијом. Стартна позиција робота је означена зеленим кружићем на слици лево.*

Додатни проблем представља то што робот за мапирање током претраге прилази превише близу објектима, рачунајући и објекте великог приоритета, тако да је ризик од непожељних контакта са другим роботима значајно изражен.

4.2.2. Примена *D* Lite* алгоритма у комбинацији са *on-line* учењем

Поред наведеног у *Поглављу 4.2.1*, последњих година један од главних праваца истраживања у роботици је примена различитих техника учења у комбинацији са алгоритмима за планирање путање робота. Циљ је да се на овај начин примени стечено искуство како би се унапредио процес прорачуна путање.

Концепт предложен у [42-44] користи формирану базу података која чува сегменте научених локалних путања у зависности од уобичајених геометријског облика препрека у окружењу, како би се на основу њих формирала глобална путања у зависности од будуће конкретне ситуације. Планирање путање коришћењем учења такође је предложено и у [45-47], где су уведени такозвани „графови базирани на искуству“, који у ствари представљају мрежу путања из претходних итерација. Овај приступ је веома користан у окружењу које поседује у одређеном делу фиксну структуру, а у другом делу су могуће промене. Интересантан приступ коришћења научених маневара из претходних ситуација који су захтевали избегавање судара са препрекама како би се побољшало планирање у наредним итерацијама разматра се у [102].

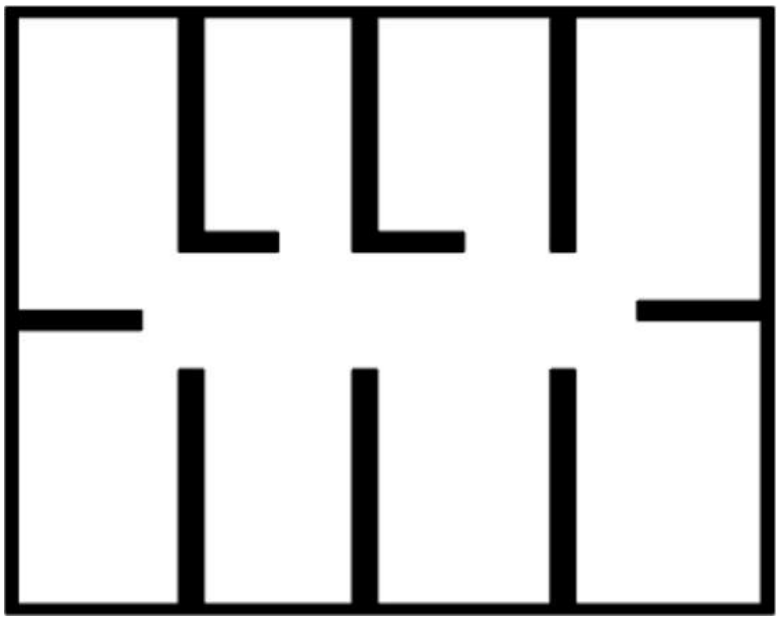
У сценаријима претраге окружења карактеристично је високо понављајуће кретање робота кроз исти простор, јер се путање робота у мањој или већој мери укрштају идући од једне до друге позиције. Наведено се може искористити за примену *on-line* учења базираног на *„M-out-of-N“* детектору како би се евентуалне споропроменљиве препреке детектоване у претраженом делу окружења (други роботи, машине или група људи који на некој фиксној позицији одређено време раде своје задатке и сл.), интегрисале односно брисале из мапе која се користи за планирање путања робота на начин који може да унапреди процес планирања. Ове ситуације су такође карактеристичне за трагање и спасавање у урбаном окружењу, али нису ограничене само на ту врсту мисија претраге окружења. Овде се ради о споропроменљивим препрекама детектованим сензорима које робот користи за безбедно кретање пратећи генерисану путању, који су у генералном случају значајно осетљивији, али и значајно мањег домета од сензора које користи за опсервацију окружења са сукцесивних позиција које посећује у циљу прикупљања података значајних за саму претрагу [87]. Истовремено треба нагласити да се овде не говори о избегавању покретних препрека током кретања робота, за шта се користе посебни алгоритми и технике. У склопу истраживања која су претходила припреми за израду дисертације, аутор дисертације је публиковао рад у коме је комбинована употреба *D* Lite* алгоритма и *on-line* учења базираног на *„M-out-of-N“* детектору примењена за унапређење планирања кретања робота у сценарију продајног центра [61].

У изворној форми, *„M-out-of-N“* детектор се користи у радарској техници за детекцију циљева и изражава се за *k*-ту опсервацију на следећи начин [103]:

$$W_{kn} = \sum_{i=k-n+1}^{k} y_i \tag{21}$$

У (21) важи да је $k \geq n$, где је $n = N$ дужина прозора детектора, док y_i представља i-ту опсервацију и у идеалном случају има вредност „1“ када је добијен одраз од циља (у супротном има вредност „0“). Присуство циља се проглашава након поређења вредности W_{kn} са прагом M, тј. ако се задовољи услов $W_{kn} \geq M$.

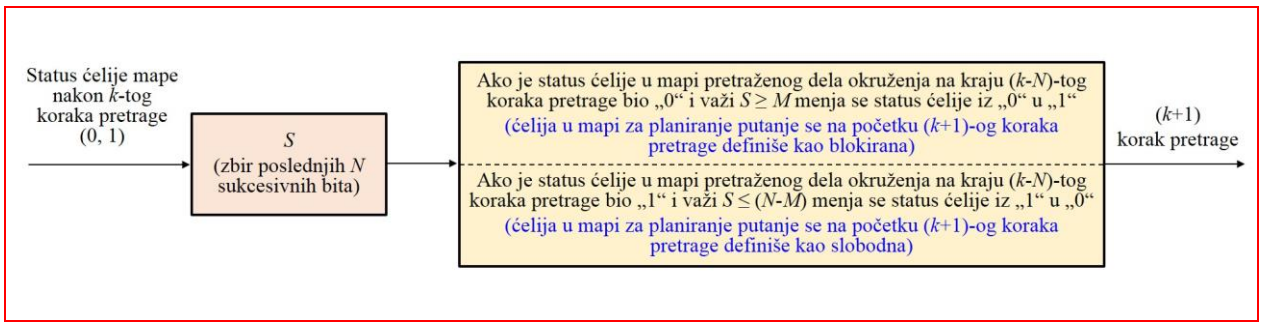
У оквиру дисертације, комбинована употреба *D* Lite* алгоритма и *on-line* учења базираног на „*M-out-of-N*“ детектору за унапређење планирања кретања робота у мисијама претрага окружења биће тестирана на примеру симулације претраге окружења приказаног на *Слици 55*. Циљ примене алгоритма учења је да се подаци прикупљени сензорима робота за безбедно кретање из претходних корака претраге окружења, искористе на почетку следећег корака претраге у фази планирања путање робота. С тим у вези, у оквиру процеса претраге свакој ћелији претраженог дела окружења за потребе примене алгоритма учења додељује се статус према бинарној класификацији, тј. „0“ (слободна ћелија) или „1“ (блокирана ћелија). Статус сваке ћелије мапе претраженог дела окружења се прати и памти у сукцесивним корацима претраге тј. представља се у форми бинарне временске серије.



Слика 55. Мапа окружења за тестирање примене D Lite алгоритма у комбинацији са on-line учењем базираним на „M-out-of-N“ детектору.*

Ако посматрамо конкретну ћелију мапе, одговарајућу бинарну временску серију промене њеног статуса обрађује *„M-out-of-N“* детектор и декларише на почетку ($k+1$)-ог корака претраге промену статуса ћелије у односу на статус који је имала на крају (k-N)-тог корака (а пре отпочињања (k-$N+1$)-ог корака) претраге, у случају да се на крају k-тог корака претраге задовољи следећи статистички тест: постоји најмање M од последњих N битова супротне вредности у односу на вредност бита на крају (k-N)-тог корака претраге, где је N дужина „прозора“ детектора.

Одговарајућа блок шема алгоритма учења базираног на *„M-out-of-N“* детектору, модификованом за примену у конкретном случају, приказана је на *Слици 56*.

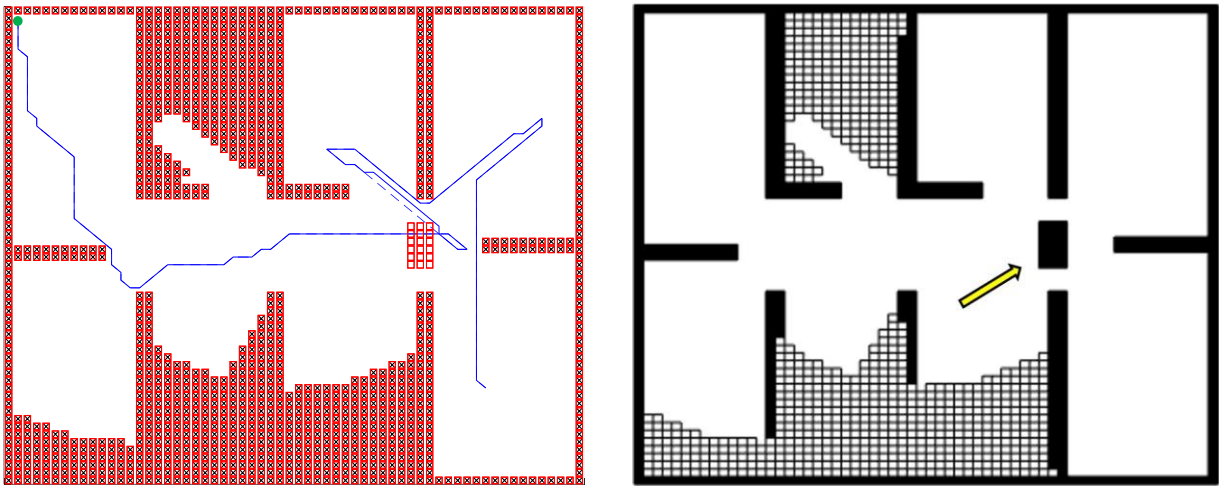


Слика 56. Блок шема алгоритма учења.

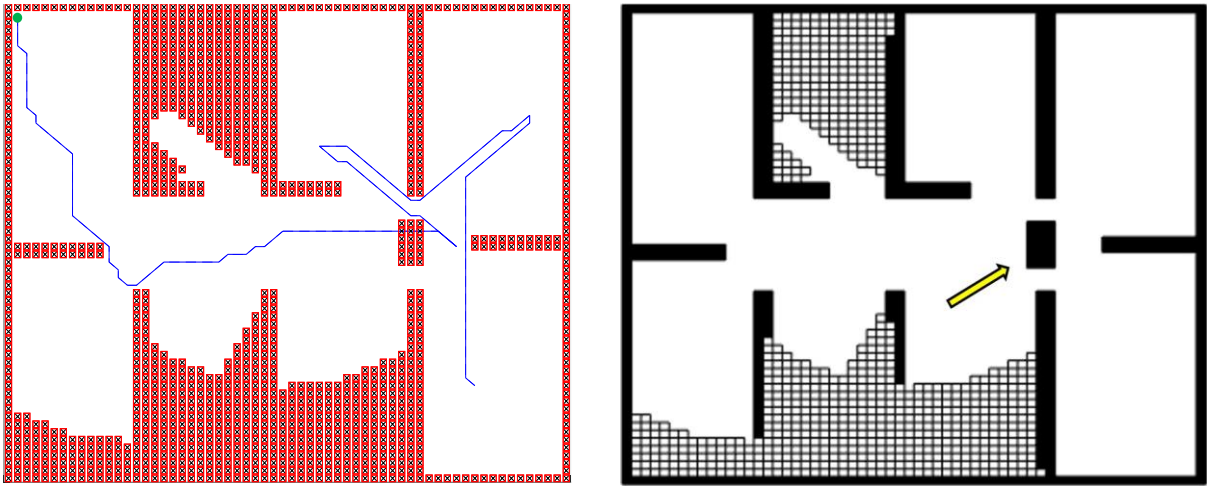
Примењено на конкретном примеру, ово значи да ако се у N узастопних корака претраге M пута детектује препрека у ћелији која је на крају (k-N)-тог корака претраге била слободна, онда се током процеса генерисања путање на почетку ($k+1$)-ог корака претраге ова ћелија декларише као блокирана. Генерално, иста логика се примењује за ћелије које су на крају (k-N)-тог корака претраге биле блокиране. Наравно, у сваком кораку претраге када нека ћелија није опажена одговарајућим сензором робота, њој се додељује статус који одговара основној мапи претраженог дела окружења.

Овако специфициран детектор памти статусе ћелија мапе само у одређеном броју последњих корака претраге, што је регулисано дужином прозора N, чиме се обезбеђује праћење промена у мапи у складу са њиховом динамиком. Параметар M се подешава према специфичности конкретног окружења, у зависности од очекиваних промена.

Сценарио у ком је тестиран предложени приступ је тако конципиран да се препрека појавила током претраге и то у претраженом делу окружења након извршене претраге 60% окружења, а нестала је након извршене претраге 85% окружења. На *Сликама 57. и 58.* приказани су путања робота и мапа окружења након извршене претраге 80% окружења, тј. у тренутку када је присутна препрека која је и приказана на сликама (означена стрелицом).



Слика 57. Путања робота (лево) и мапа окружења након извршене претраге 80% окружења (десно) без примене алгоритма учења. Динамичка препрека (означена стрелицом) је присутна, али је робот детектује тек када приђе релативно близу, након чега D Lite алгоритам врши корекцију путање. Стартна позиција робота је означена зеленим кружићем на слици лево.*

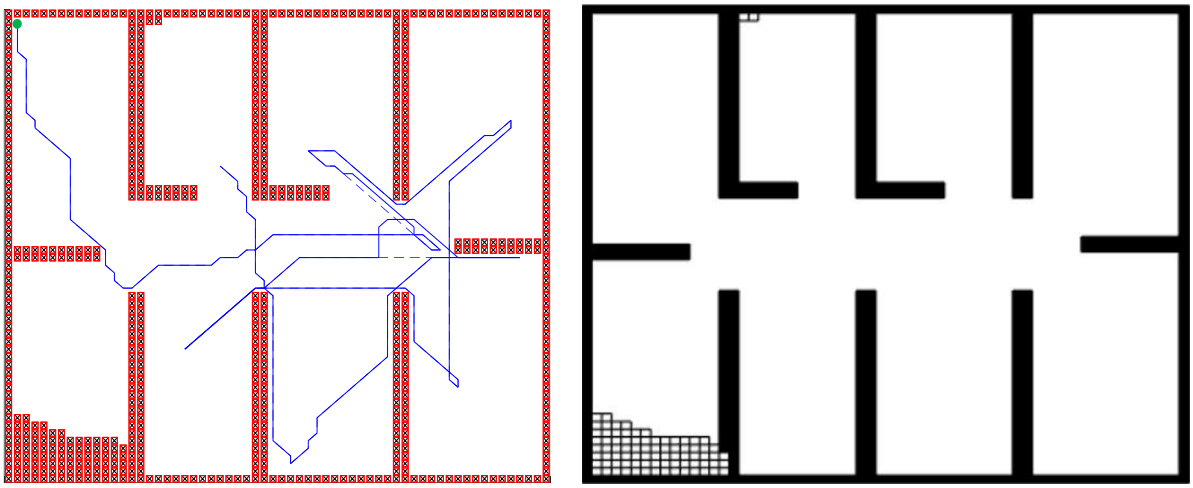


Слика 58. Путања робота (лево) и мапа окружења након извршене претраге 80% окружења (десно) са применом алгоритма учења. Динамичка препрека (означена стрелицом) је присутна. D Lite алгоритам у комбинацији са алгоритмом учења обезбеђује благовремено извршење маневра робота у циљу избегавање препреке (нема корекције путање). Стартна позиција робота је означена зеленим кружићем на слици лево.*

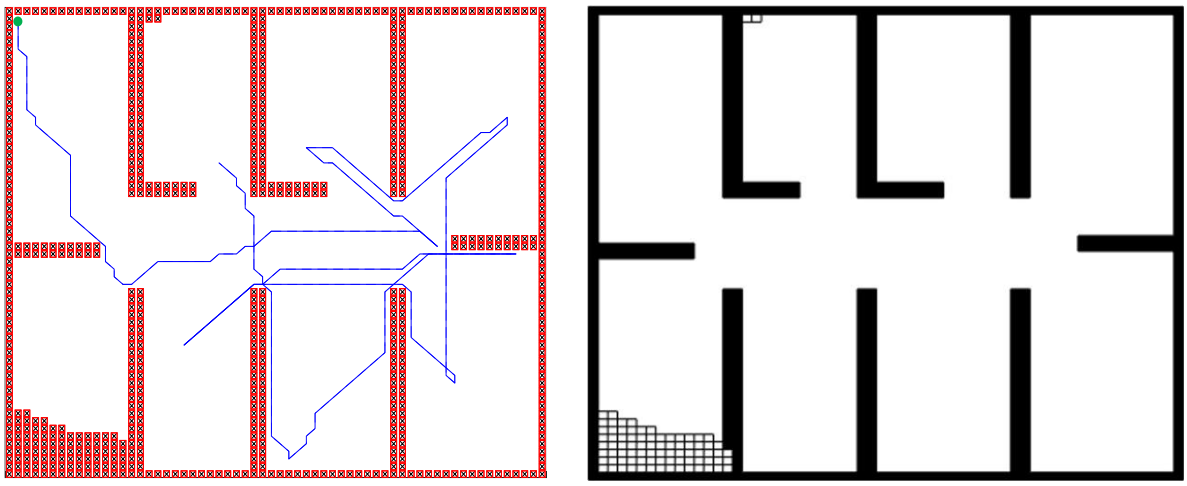
Анализирајући *Слику 57.* може се приметити да је након појаве препреке алгоритам за планирање путање (*D* Lite*), без примене *on-line* учења, био у ситуацији да иницијално генерисану путању од актуелне до следеће позиције робота која је водила кроз препреку (јер иста није била интегрисана у мапу за планирање) коригује у току кретања робота, што се ради у тренутку када робот дође близу препреке и детектује је својим сензорима за безбедно кретање. Део путање који је репланирањем промењен приказан је испрекиданом линијом. Поред додатног времена које је неопходно утрошити на репланирање путање, која је у том случају и дужа (због компликованијег маневра који робот треба да уради како би избегао препреку), робот у овим ситуацијама прилази превише близу препреке тако да је ризик од непожељних контакта значајно изражен.

Уз примену алгоритма учења, *D* Lite* располаже информацијама о препреци благовремено тј. у фази планирања иницијалне путање за посматрани корак претраге, тако да обезбеђује роботу да маневар избегавања препреке изведе много раније (*Слика 58*). Такође, може се приметити да кроз локацију на којој се налази препрека пролази и једна пуна (неиспрекидана) линија. Ради се о валидној путањи робота насталој у фази претраге окружења пре појаве препреке.

На *Сликама 59. и 60.* приказани су путања робота и мапа окружења након извршене претраге 90% окружења (у тренутку када препрека више није присутна у окружењу).



Слика 59. Путања робота током претраге (лево) и мапа окружења након извршене претраге 90% окружења (десно) без примене алгоритма учења. Динамичка препрека више није присутна, али се може уочити да је у току претраге више пута вршена корекција путање. Стартна позиција робота је означена зеленим кружићем на слици лево.



Слика 60. Путања робота током претраге (лево) и мапа окружења након извршене претраге 90% окружења (десно) са применом алгоритма учења. Динамичка препрека више није физички присутна и може се приметити да током претраге није било корекција путање. Стартна позиција робота је означена зеленим кружићем на слици лево.

Као што се види са *Слике 59*, алгоритам за планирање путање је више пута (у више корака претраге) био у ситуацији да коригује иницијално израчунату путању, што је применом алгоритма учења избегнуто (*Слика 60*). На основу *Слика 59.* и *60.* може се закључити да „*M-out-of-N*“ детектор обезбеђује и брисање препреке (након њеног физичког нестанка) из мапе која се користи за планирање путање робота, такође у складу са дефинисаним статистичким тестом.

Укупан пређени пут робота је у ситуацији без примене алгоритма учења износио 348.15, а у ситуацији са применом алгоритма учења 341.10. Ово смањење укупног пређеног пута робота у ситуацији са применом алгоритма учења је такође директна последица оптимизације путање у фази иницијалног планирања.

4.3. Верификација кључних резултата истраживања у *Gazebo 3D* симулатору

Gazebo је специјализовано окружење за *3D* симулацију робота. Омогућава веродостојно тестирање робота и њихових активности у симулираним *3D* сценаријима. У дисертацији је коришћен за верификацију кључних резултата истраживања добијених симулацијом претраге окружења у Матлабу, тј. као веродостојна замена за тестирање анализираних стратегија за претрагу на реалним роботима.

У конкретном случају, примењен је виртуелни модел *Turtlebot 3* робота. Наведени модел робота за мапирање окружења као сензор користи– *LIDAR LDS-01*, домета 3.5m и угаоне резолуције 1°.

За контролу робота, одометрију и *SLAM* функцију тј. за интеграцију података прикупљених сензором робота, формирање мапе окружења и истовремено одређивање положаја (локације) робота у односу на ту мапу, коришћени су пакети који раде под *Robot Operating System* (*ROS*). *ROS* обезбеђује колекцију алата и библиотека за развој и оптимизацију роботских система, за шта је посебно погодан у комбинацији са другим специјализованим софтверским пакетима намењеним за роботику као што је *Gazebo* симулатор, али и у комбинацији са Матлабом, итд.

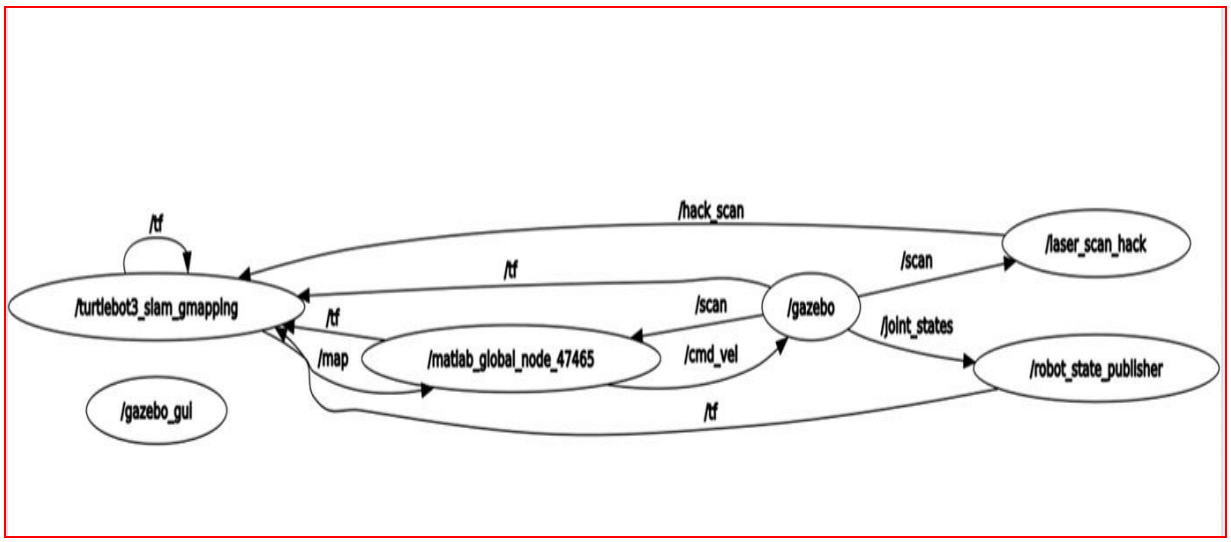
Конкретно, у дисертацији су из *ROS* коришћени следећи пакети: *TurtleBot3_Gazebo* и *TurtleBot3_Gmapping*. Наведени пакети користе следеће типове порука: */odom* (за информације о позицији робота добијене одометријом), */scan* (за информације прикупљене са *LIDAR* сензора), */map* (подаци за формирање мапе окружења), */cmd_vel* (информације о линеарној и ротационој брзини робота), */tf* (описује однос између координатног система мапа и координатног система робота), */joint_states* (информација о стањима роботских актуатора) и */hack_scan* (модификовани */scan* подаци). Коришћена архитектура *ROS* са приказаним чворовима, називима пакета и токовима података приказана је на *Слици 61*.

Gazebo као симулатор комуницира са *ROS*, а *ROS* шаље податке о формираној мапи ка Матлабу где се извршавају стратегије за избор следеће позиције за претрагу и планирање путање робота до те позиције. Након тога се у Матлаба извршава алгоритам који регулише контролу кретања робота како би пратио израчунату путању. Специфичне изазове представљало је следеће:

- Прилагођење приказа мапе из *ROS* у Матлаб у циљу њиховог усаглашавања;
- Прилагођење мапе из Матлаба за избор следеће позиције робота и планирање путање робота. Било је неопходно формирање бафер зона (неколико редова вештачки блокираних поља) око препрека како би се спречио физички контакт робота са истим;
- Адаптација података са *LIDAR*-а како би алгоритам *Gmapping* радио мапирање отвореног простора по принципу *Line-of-Sight*. Физички је сензор тако конципиран да када је препрека много даља од његовог домета повратни сигнал се не региструје. За такве ситуације дефинисано је да у домету сензора нема препрека.

За избор следеће позиције робота (p), такође су коришћени критеријуми (ближе објашњени у *Поглављу 3.2*):

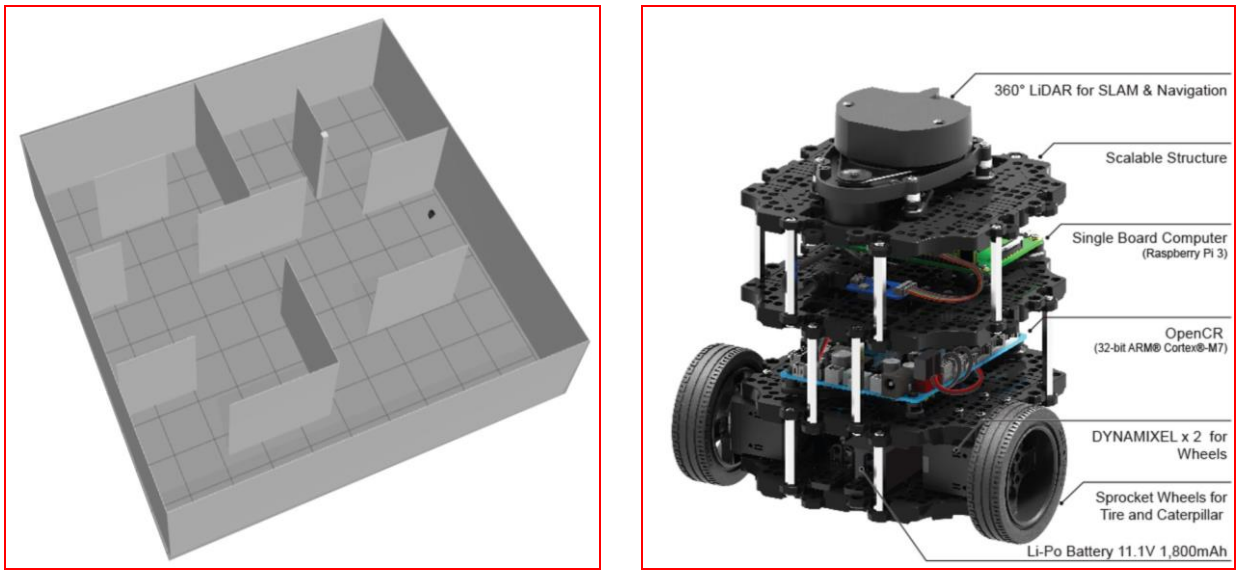
- $L(p)$, дужина путање од актуелне позиције робота до позиције-кандидата p,
- $A(p)$, информативни потенцијал позиције-кандидата p,
- $P(p)$, удаљеност позиције-кандидата p од базне станице.



Слика 61. Коришћена архитектура ROS са приказаним чворовима, називима пакета и токовима података.

За тестирање је коришћена мапа величине 10mx10m, дискретизована на ћелије димензије 0.2mx0.2m, тако да је димензија мапе изражена у ћелијама 50x50 (*Слика 62*).

Тестиране су све стратегије које користе вишекритеријумско одлучивање, анализиране у дисертација, тј. методе *TOPSIS*, *COPRAS* и *SAW*, као и класична стратегија из литературе која је показала најбоље резултате у симулираним сценаријима у Матлабу – *GBL*.



Слика 62. 3D окружење креирано у Gazebo симулатору за потребе тестирања стратегија за претрагу окружења, црна тачка представља робот (лево). Turtlebot 3 робот и његове основне компоненте [104] (десно).

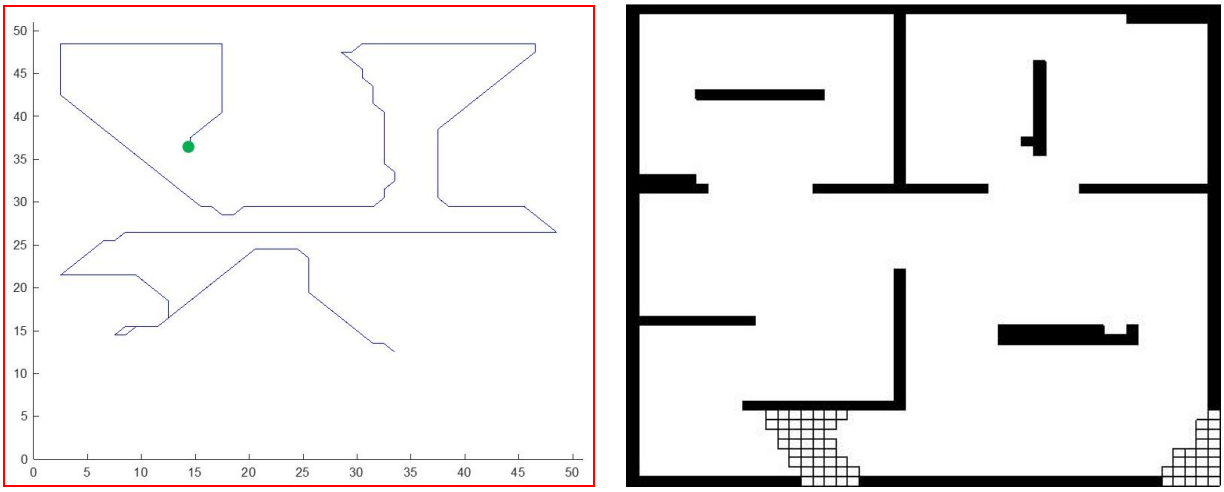
Резултати тестирања у смислу просечних (средњих) укупних дужина путања робота за две стартне позиције (означене на сликама у даљем тексту) у циљу претраге 90% окружења у *Gazebo* симулатору са *Слике 62*, за анализиране стратегије, приказани су у

Табели 11. Координате стартних позиција робота дефинисане су у односу на Декартов координатни систем везан за леви доњи угао мапе. Један подеок на оси координатног система једнак је димензији ћелије мапе.

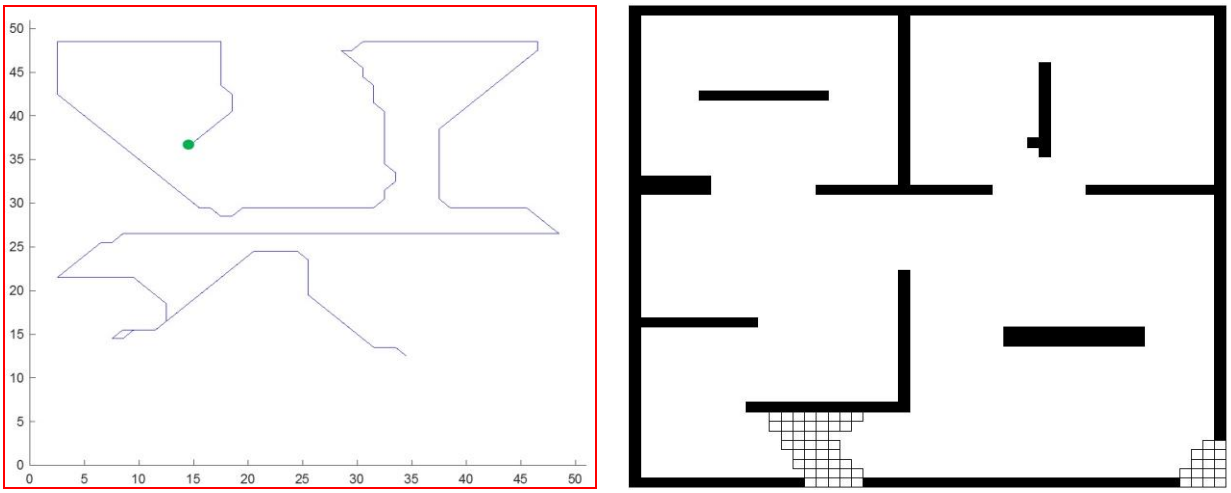
Табела 11. Просечне (средње) дужине путања робота за потребе претраге 90% окружења у Gazebo симулатору применом тестираних стратегија.

Strategije za pretragu okruženja		Srednje dužine putanja robota
VKO	*SAW*	240.59
	COPRAS	205,91
	TOPSIS	201.56
GBL		212.24
Napomena: Kod metoda *VKO* primenjene su kombinacije težinskih koeficijenata *(0.7, 0.2, 0.1)*, za kriterijume *L*(*p*), *A*(*p*) i *P*(*p*), respektivno.		

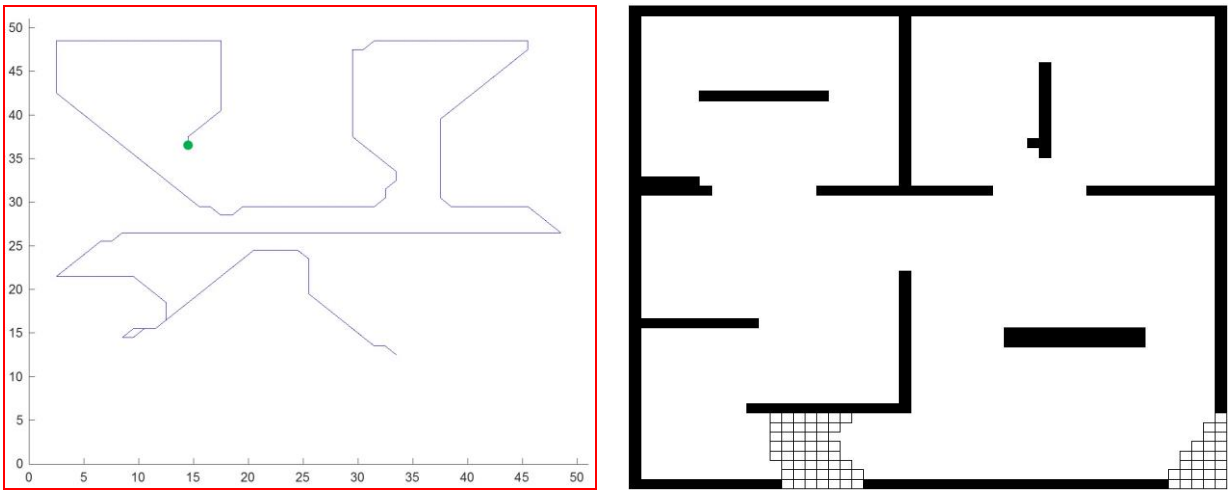
Путање робота генерисане од стране *D* Lite* алгоритма током претраге окружења у *Gazebo* симулатору, као и мапе окружења формиране након претраге применом тестираних стратегија за прву стартну позицију робота, приказани су на *Сликама 63, 64, 65.* и *66.*



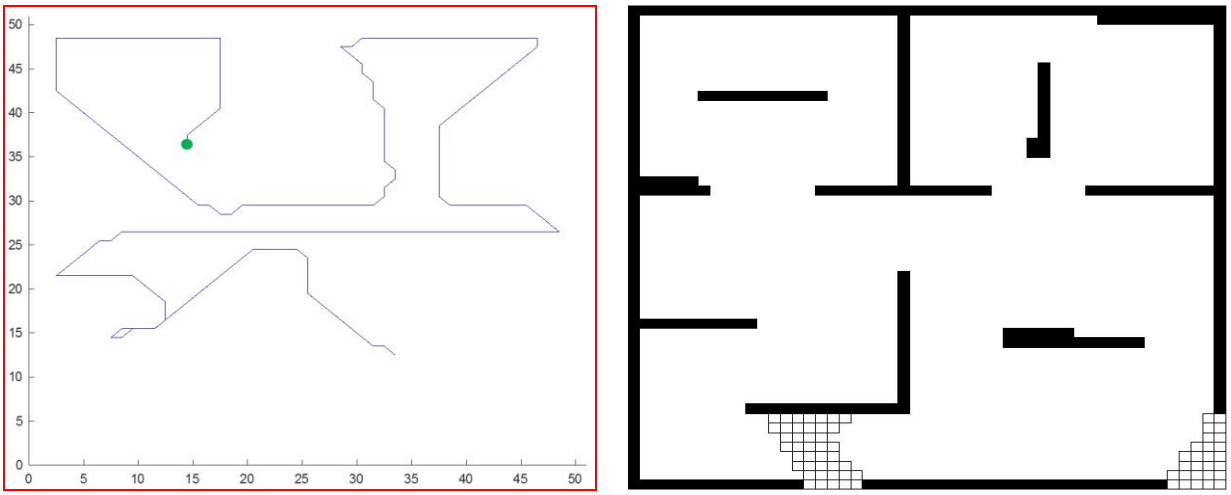
Слика 63. Путања робота генерисана D Lite алгоритмом током претраге 90% окружења у Gazebo симулатору (лево) и мапа окружења након претраге (десно) применом SAW методе за избор следеће позиције робота уз комбинацију тежинских коефицијената (0.7, 0.2, 0.1) за критеријуме L(p), A(p) и P(p), респективно, и за стартну позицију робота (x=15, y=37), означену зеленим кружићем на слици лево.*



Слика 64. Путања робота генерисана D Lite алгоритмом током претраге 90% окружења у Gazebo симулатору (лево) и мапа окружења након претраге (десно) применом COPRAS методе за избор следеће позиције робота уз комбинацију тежинских коефицијената (0.7, 0.2, 0.1) за критеријуме L(p), A(p) и P(p), респективно, и за стартну позицију робота (x=15, y=37), означену зеленим кружићем на слици лево.*

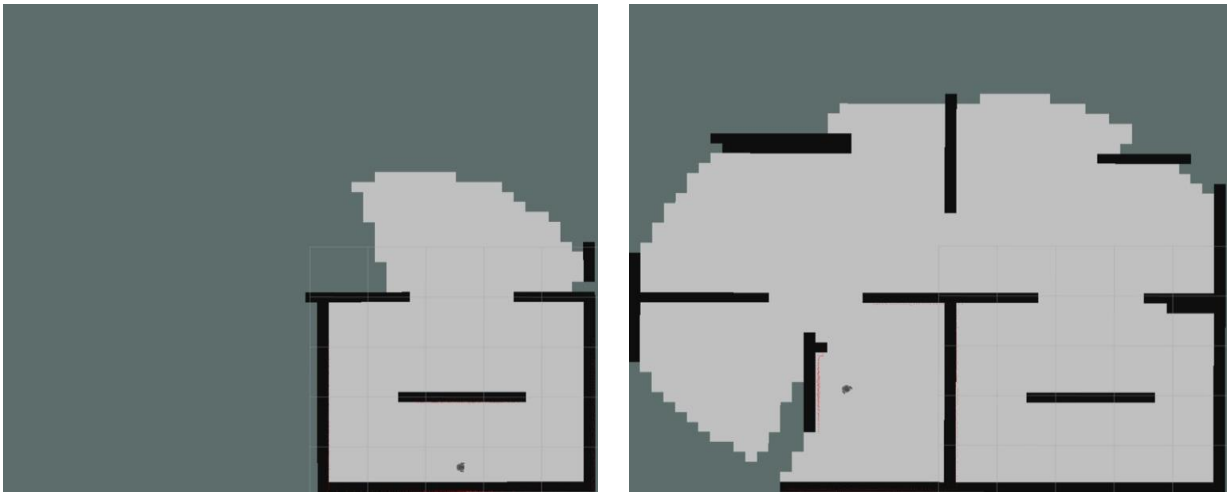


Слика 65. Путања робота генерисана D Lite алгоритмом током претраге 90% окружења у Gazebo симулатору (лево) и мапа окружења након претраге (десно) применом TOPSIS методе за избор следеће позиције робота уз комбинацију тежинских коефицијената (0.7, 0.2, 0.1) за критеријуме L(p), A(p) и P(p), респективно, и за стартну позицију робота (x=15, y=37), означену зеленим кружићем на слици лево.*

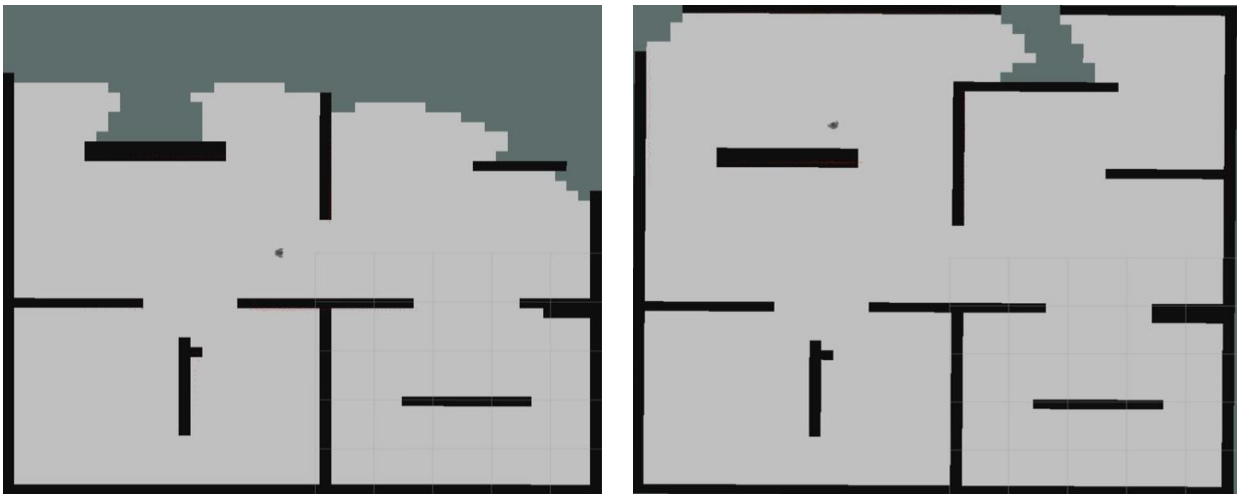


Слика 66. Путања робота генерисана D Lite алгоритмом током претраге 90% окружења у Gazebo симулатору (лево) и мапа окружења након претраге (десно) применом GBL стратегије за избор следеће позиције робота и за стартну позицију робота (x=15, y=37), означену зеленим кружићем на слици лево.*

На *Сликама 67.* и *68.* дате су секвенце претраге у *Gazebo*. Црна тачка приказује робот.

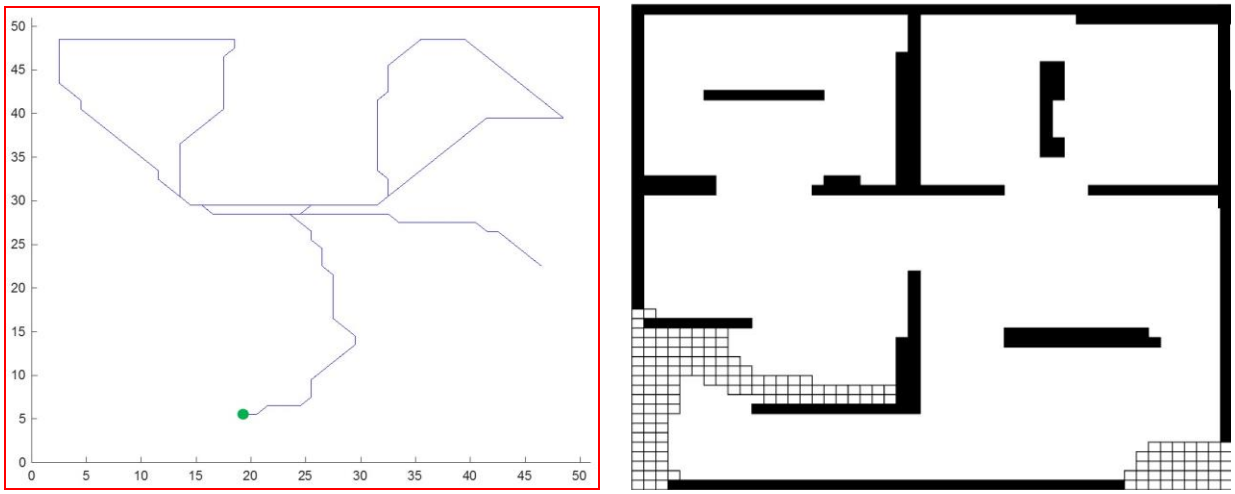


Слика 67. Секвенце 1. и 2. претраге окружења у Gazebo симулатору.

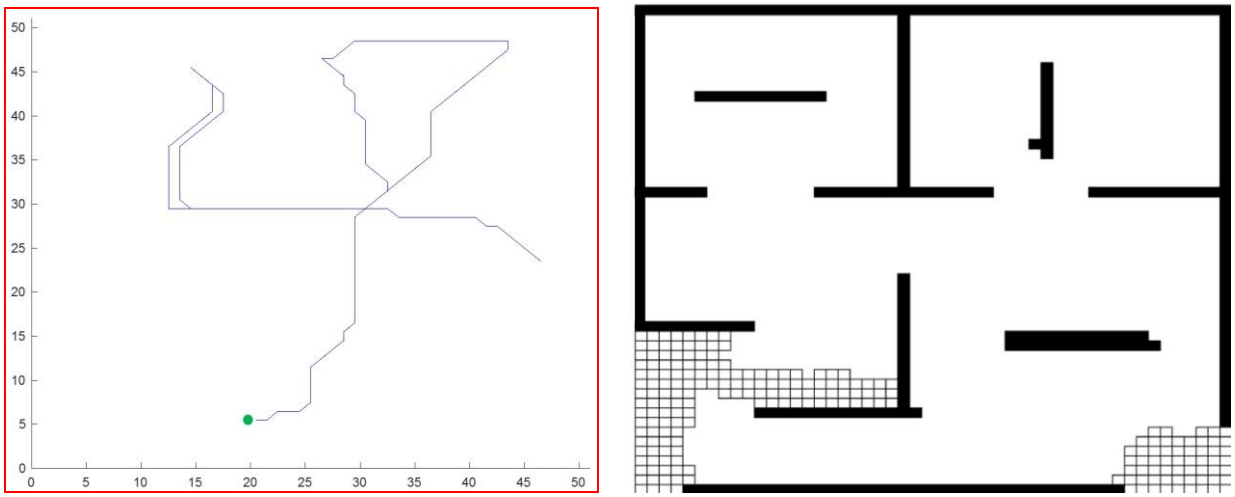


Слика 68. Секвенце 3. и 4. претраге окружења у Gazebo симулатору.

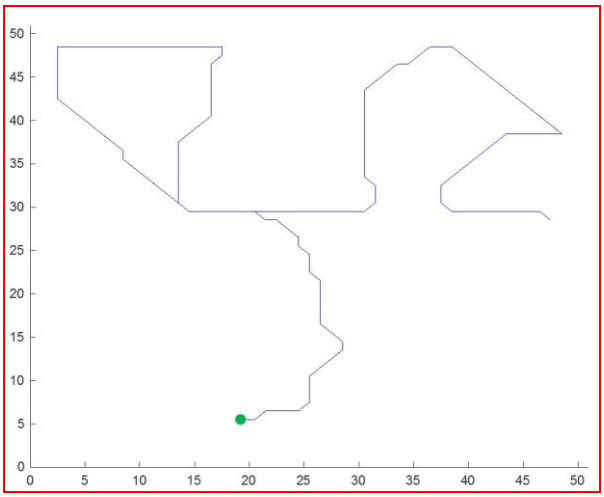
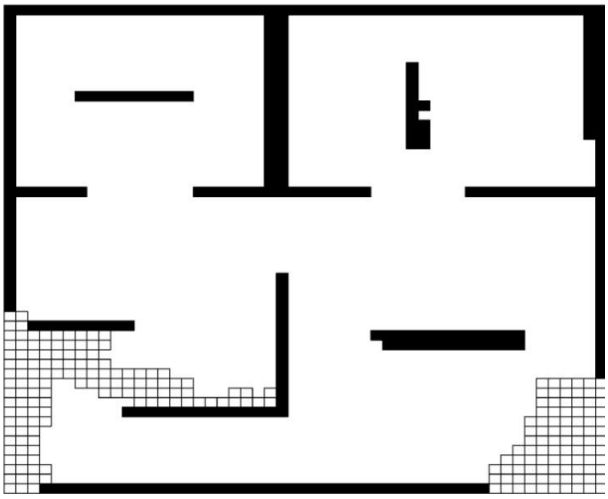
Путање робота генерисане од стране *D* Lite* алгоритма током претраге окружења у *Gazebo* симулатору, као и мапе окружења формиране након претраге применом тестираних стратегија за другу стартну позицију робота, приказани су на *Сликама 69, 70, 71.* и *72.*



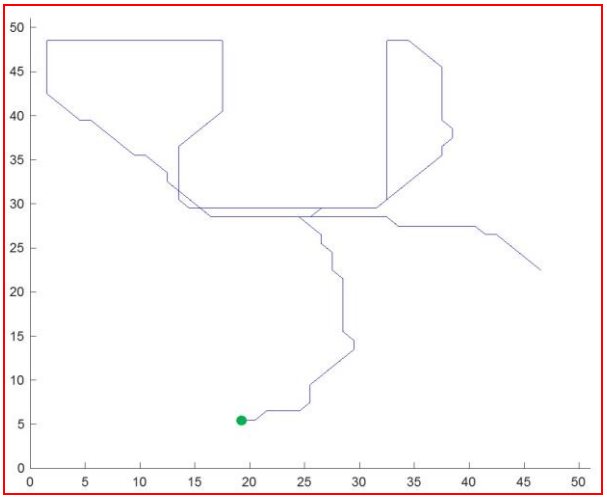
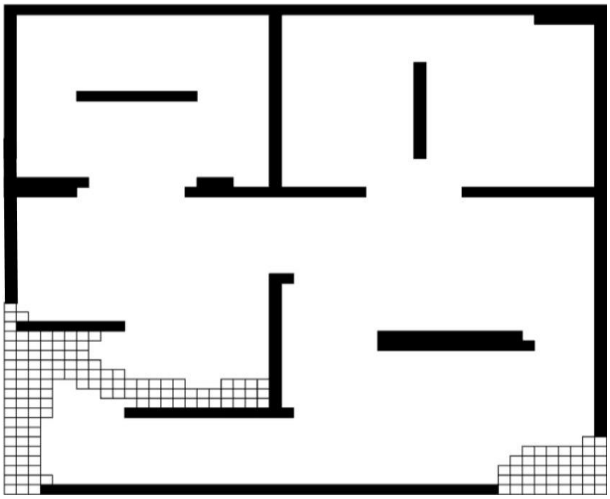
Слика 69. Путања робота генерисана D Lite алгоритмом током претраге 90% окружења у Gazebo симулатору (лево) и мапа окружења након претраге (десно) применом SAW методе за избор следеће позиције робота уз комбинацију тежинских коефицијената (0.7, 0.2, 0.1) за критеријуме L(p), A(p) и P(p), респективно, и за стартну позицију робота (x=19, y=6), означену зеленим кружићем на слици лево.*



Слика 70. Путања робота генерисана D Lite алгоритмом током претраге 90% окружења у Gazebo симулатору (лево) и мапа окружења након претраге (десно) применом COPRAS методе за избор следеће позиције робота уз комбинацију тежинских коефицијената (0.7, 0.2, 0.1) за критеријуме L(p), A(p) и P(p), респективно, и за стартну позицију робота (x=19, y=6), означену зеленим кружићем на слици лево.*

Слика 71. Путања робота генерисана D Lite алгоритмом током претраге 90% окружења у Gazebo симулатору (лево) и мапа окружења након претраге (десно) применом TOPSIS методе за избор следеће позиције робота уз комбинацију тежинских коефицијената (0.7, 0.2, 0.1) за критеријуме L(p), A(p) и P(p), респективно, и за стартну позицију робота (x=19, y=6), означену зеленим кружићем на слици лево.*

Слика 72. Путања робота генерисана D Lite алгоритмом током претраге 90% окружења у Gazebo симулатору (лево) и мапа окружења након претраге (десно) применом GBL стратегије за избор следеће позиције робота и за стартну позицију робота (x=19, y=6), означену зеленим кружићем на слици лево.*

Анализирајући резултате из *Табеле 11.* може се закључити да су претрагом окружења у *Gazebo* симулатору уз његову интеграцију са *ROS* верификовани кључни резултати истраживања добијени симулацијом претраге окружења у Матлабу. Најбоље је рангирана метода *TOPSIS*, затим следе *COPRAS* и *GBL*, а најлошије резултате има *SAW*.

5. ЗАКЉУЧАК

Два главна правца истраживања у докторској дисертацији су планирање путање робота и претрага окружења, оба са високим нивоом аутономности у извршавању. И један и други правац спадају у фундаменталне области изучававања у роботици, имајући у виду да чине основу њених многобројних примена у пракси. Ова констатација произилази из чињенице да се о окружењима у која се робот упућује обично веома мало зна, тако да поред оспособљености за обављање дефинисаних задатака, робот треба да буде у стању да планира своје кретање и упозна своје окружење.

Суштински, и претрага окружења и планирање путање имају своје засебне, сложене изазове и у обе поменуте области изучавања постоји још увек велики број отворених питања. Докторска дисертација у том смислу прати најновије резултате истраживања у поменутим областима и на томе се заснива њена савременост.

У оквиру планирања путање робота, у дисертацији су изложени детаљно алгоритми из фамилије *A** алгоритма - *ARA**, *D** Lite и *AD**. Ови алгоритми се често примењују у пракси за планирање путање робота, па тако и у стратегијама за претрагу окружења. Функционишу на бази обраде графа којим се моделира мапа окружења у којем се креће робот. Основне карактеристике *D* Lite* алгоритма су те да увек генерише путању најмање цене (у специјалном случају најкраћу могућу путању у условима ограничења проистеклих из дискретизације мапе терена), као и ефикано процесуирање детектованих промена у окружењу у циљу корекције путање. Ради јаснијег увида у специфичности *D* Lite* алгоритма, он се у дисертацији пореди са алгоритмима *ARA** и *AD**. *ARA** је *Anytime Replanning* верзија *A** алгоритма, која при процесуирању чворова користи пондерисану хеуристичку функцију тј. процену удаљености од актуелног до циљног чвора како би убрзао прорачун и генерисао иницијалну путању за што краће време, али тада даје субоптимално решење које накнадно поправља у току кретања робота. Мана овог алгоритма је што у случају настанка промена у окружењу ресетује резултате прорачуна и почиње планирање путање из почетка. *AD** је *Anytime Dynamic* верзија *A** алгоритма. Код њега су имплементиране предности и *ARA** и *D* Lite*. Дакле има особину *anytime* алгоритама, што значи да има могућност коришћења пондерисане хеуристичке функције како би иницијалну путању генерисао за што краће време, при чему је та путања такође субоптимална и иста се накнадно поправља у току кретања робота као код *ARA** алгоритма. Уједно има особину ефикасног процесуирања промена у окружењу као и *D* Lite*, тј. када се детектује промена не ресетује прорачун путање већ у максималној мери користи резултате из претходних итерација.

У дисертацији су, даље, тестиране могућности примене *TOPSIS* методе *BKO* у комбинацији са изабраним критеријумима за избор следеће позиције робота, како би се унапредила ефикасност претраге комплексних окружења, при чему је за планирање путање робота примењен *D* Lite* алгоритам. Специфичност *TOPSIS* методе је да све алтернативе представља у *n*-димензионалном простору (где је *n* број критеријума) тако да при одлучивању бира ону алтернативу која има најмању Еуклидову удаљеност од идеалног решења и истовремено највећу могућу Еуклидову удаљеност од анти-идеалног решења. Максималним удаљавањем од анти-идеалног (хипотетички најлошијег) решења минимизује се ризик у одлучивању. Кључна карактеристика комплексних окружења (окружења са великим бројем препрека) је да се позиције-кандидати (алтернативе при одлучивању) значајно разликују у погледу вредности изабраних критеријума, који непосредно утичу на ефикасност претраге, због чега је управо изражен ризик од лошег избора следеће позиције робота. Имајући наведено у виду, у оквиру истраживања је доказана хипотеза да се применом *TOPSIS* методе за избор следеће позиције робота и *D**

Lite алгоритма за планирање његове путање обезбеђује претрага комплексних окружења за краћи укупан пређени пут робота у односу на методе *SAW* и *COPRAS*, као и у односу на остале анализиране класичне стратегије за претрагу окружења из литературе (*Dist_Min, GBL, AOJRF, WS*), што представља први научни допринос дисертације.

На основу реализованих симулација у Матлабу, *TOPSIS* метода је постигла најбоље резултате у односу на све конкурентне стратегије у 5 од 7 тестираних сценарија. У једном сценарију најбоље резултате је имала *GBL* стратегија, а у једном *COPRAS*.

TOPSIS метода није постигла најбољи резултат при претрази *indoor* окружења *B* у варијанти када је домет сензора највећи, што је везано за још једну битну карактеристику комплексних окружења, а то је велики ризик од грешке у процени информативног потенцијала позиција-кандидата. Наведена грешка доводи до девијација коначних резултата претраге. Уколико је домет сензора већи, већи је и ризик од грешке у процени поменутог критеријума, а додатну тежину томе даје и велики број препрека у окружењу *B* које је најкомплексније.

Такође, *TOPSIS* метода није постигла најбољи резултат при претрази *indoor* окружења *D*, што се образлаже тиме да је наведено окружење мале комплексности, тако да не долазе до изражаја бенефити бољег одлучивања, који су као што је претходно наведено карактеристични за примену *TOPSIS* код претраге комплексних окружења у комбинацији са изабраним критеријумима за евалуацију позиција-кандидата. У овом случају су скоро све тестиране стратегије постигле упоредиве резултате.

Анализирајући све симулиране сценарије у Матлабу и ранг које су стратегије постигле у просеку, може се закључити да после *TOPSIS* по ефикасности следе *COPRAS* и *GBL*, које такође представљају добар избор стратегија за претрагу комплексних окружења, затим *WS* и *SAW*, док су најлошије резултате у тестираним сценаријима имале стратегије *AOJRF* и *Dist_Min*.

Кључни резултати истраживања су на крају верификовани у *Gazebo* симулатору (специјализованом окружењу за *3D* симулацију робота), уз коришћење пакета који раде под *Robot Operating System* (*ROS*), као веродостојна замена за тестирање анализираних стратегија за претрагу на реалним роботима. За те потребе примењен је виртуелни модел *Turtlebot 3* робота. И у овом случају најбоље је била рангирана метода *TOPSIS*, затим следе *COPRAS* и *GBL*, а најлошије резултате је имала метода *SAW*.

У дисертацији су, такође, предложена два приступа за унапређење планирања путање робота применом *D* Lite* алгоритма у мисијама претраге окружења за потребе трагања и спасавања у урбаном окружењу.

У првом случају ради се о мисијама претраге за потребе трагања и спасавања када је услед неког акцидента измењена иницијално позната мапа окружења у коме се налазе објекти различите намене и када је поред робота за мапирање неопходно ангажовање и робота и/или људи за обављање других задатака, нпр. за евакуацију повређених, уклањање препрека, неутралисање опасних материја. У циљу управљања претрагом окружења у оваквим условима, а за потребе ефикасне реализације мисије трагања и спасавања, предлаже се примена фази интегралног система за дефинисање тежина ћелија мапе која се користи за планирање путања робота. Намена фази интегралног система је да обезбеди да *D* Lite* алгоритам генерише путање најмање цене (у зависности од три параметра) и то тако да се претрага окружења реализује на такав начин и тим редом да „покрива“ објекте према приоритету, од највећег ка најмањем, као и да роботи за мапирање при кретању у

што мањој мери ометају друге роботе и људе, који због природе својих задатака обично морају прићи ближе објектима. Резултати истраживања и тестирања су показали да је остварена горе описана намена фази интегралног система, што представља други допринос дисертације. Са друге стране, у ситуацији када није примењен фази интегрални систем, при реализацији претраге окружења није испоштован приоритет објеката, а додатни проблем представља то што робот за мапирање током претраге прилази превише близу објектима и то објектима великог приоритета, тако да је ризик од непожељних контакта са другим роботима и људима значајно изражен.

У другом случају ради се о мисијама претраге окружења са споропроменљивим препрекама детектованим у претраженом делу окружења (други роботи, машине или група људи који на некој фиксној позицији одређено време обављају своје задатке и сл.). За оптимизацију планирања путање робота за мапирање у оваквим ситуацијама у дисертацији се предлаже примена *D* Lite* алгоритма у комбинацији са *on-line* учењем базираним на „*M-out-of-N*“ детектору. Циљ је да се поменуте препреке интегришу, односно бришу из мапе која се користи за планирање путања робота на начин који унапређује процес планирања. Ове ситуације су такође карактеристичне за трагање и спасавање у урбаном окружењу, али нису ограничене само на ту врсту мисија претраге окружења. Резултати истраживања су показали да је предложеним приступом обезбеђена оптимизација путања робота за мапирање, чиме је скраћен његов укупан пређени пут и уједно су минимизована његова ризична кретања у простору који дели са другим роботима, машинама или људима, имајући у виду да се маневар робота у циљу избегавања препреке изводи много раније пре потенцијалног сусрета са препреком и то уједно представља трећи допринос дисертације.

Наставак истраживања у области којом се бави дисертација одвијаће се у правцу даљег унапређења примене алгоритама за планирање путање робота који прорачун путање врше на бази обраде графа, као и стратегија за аутономну претрагу окружења. Укључиће пре свега имплементацију техника машинског учења, као и практичну примену предложених решења. Такође ће се истраживати могућност унапређења метода за дефинисање тежина ћелија мапе (која се користи за планирање путање робота) на бази података о окружењу прикупљених сензорима робота, а у циљу што реалнијег моделирања проходности терена.

Други правац истраживања односиће се на проширење и унапређење примене предложених решења на вишероботске системе хетерогеног састава у мисијама претраге окружења. Тежиште ће бити испољено на мисије претраге окружења за потребе трагања и спасавања у урбаном окружењу.

Литература

[1] C. Gomez, A. C. Hernandez and R. Barber, “Topological frontier-based exploration and map-building using semantic information,” Sensors, vol. 19, no. 20, pp. 4595, 2019.

[2] F. Amigoni, V. Caglioti and U. Galtarossa, “A mobile robot mapping system with an information-based exploration strategy,” in Proceedings of the Int. Conf. on Informatics in Control, Automation and Robotics, Setubal, Portugal, pp. 71–78, 2004.

[3] S. Kohlbrecher, J. Meyer, T. Graber, K. Petersen, O. von Stryk et al., “Hector open source modules for autonomous mapping and navigation with rescue robots,” in RoboCup: Robot World Cup XVII, ser. Lecture Notes in Artificial Intelligence (LNAI). Berlin: Springer, pp. 624–631, 2013.

[4] K. Otsu, A.-A. Agha-Mohammadi and M. Paton, “Where to look? Predictive perception with applications to planetary exploration,” IEEE Robotics and Automation Letters, vol. 3, no. 2, pp. 635–642, 2018.

[5] A. Ahrary, A. A. Nassiraei and M. Ishikawa, “A study of an autonomous mobile robot for a sewer inspection system,” Artificial Life and Robotics, vol. 11, no. 1, pp. 23–27, 2007.

[6] T. Neumann, A. Ferrein, S. Kallweit and I. Scholl, “Towards a mobile mapping robot for underground mines,” in Proceedings of the RobMech, Cape Town, RSA, pp. 279–284, 2014.

[7] D. Calisi, A. Farinelli, L. Iocchi and D. Nardi, “Multi-objective exploration and search for autonomous rescue robots,” Journal of Field Robotics, vol. 24, no. 8-9, pp. 763-777, 2007.

[8] N. Basilico and F. Amigoni, “Exploration strategies based on multi-criteria decision making for an autonomous mobile robot,” in Proceedings of the 4th European Conf. on Mobile Robots, Mlini/Dubrovnik, Croatia, pp. 259–264, 2009.

[9] M. Luperto, D. Fusi, N. Alberto Borghese and F. Amigoni, “Exploiting in accurate a priori knowledge in robot exploration,” in Proceedings of the 18th Int. Conf. on Autonomous Agents and MultiAgent Systems, Montreal, Quebec, Canada, pp. 2102–2104, 2019.

[10] P. Taillandier and S. Stinckwich, “Using the PROMETHEE multi-criteria decision making method to define new exploration strategies for rescue robots,” in Proceedings of the IEEE Int. Sym. on Safety, Security, and Rescue Robotics, Kyoto, Japan, pp. 321–326, 2011.

[11] B. Yamauchi, “A frontier-based approach for autonomous exploration,” in Proceedings of the IEEE Int. Sym. On Computational Intelligence in Robotics and Automation, Monterey, California, USA, pp. 146–151, 1997.

[12] N. Basilico and F. Amigoni, “Exploration strategies based on multi-criteria decision making for search and rescue autonomous robots,” in Proceedings of the 10th Int. Conf. on Autonomous Agents and Multiagent Systems, Taipei, Taiwan, pp. 99–106, 2011.

[13] H. H. Gonzales-Banos and J.-C. Latombe, “Navigation strategies for exploring indoor environments,” International Journal of Robotics Research, vol. 21, no. 10, pp. 829–848, 2002.

[14] A. Visser and B. A. Slamet, “Including communication success in the estimation of information gain for multi-robot exploration,” in Proceedings of the 6th Int. Sym. on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops, Berlin, Germany, pp. 680–687, 2008.

[15] C. Stachniss, “Robotic mapping and exploration,” Springer, 2009.

[16] D. Holz, N. Basilico, F. Amigoni and S. Behnk, “Evaluating the efficiency of frontier-based exploration strategies,” Conference: ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics), Munich, Germany, 2010.

[17] S. N. Heredia, “Exploration strategies for robotic vacuum cleaners,” M.S. thesis, KTH Royal Institute of Technology, School of Industrial Engineering and Management, Stockholm, Sweden, 2018.

[18] P. E. Hart, N. J. Nilsson and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” IEEE Transactions on Systems, Science, and Cybernetics, vol. 4, no. 2, pp. 100-107, 1968.

[19] L. Lu, C. Redondo and P. Campoy, “Optimal frontier-based autonomous exploration in unconstructed environment using RGB-D sensor,” Sensors, vol. 20, no. 22, pp. 1–16, 2020.

[20] A. Vellucci, “Multi-robot frontier-based exploration strategies for mapping unknown environments,” M.S. thesis, Politecnico di Torino, Torino, Italy, 2019.

[21] C. Wang, W. Chi, Y. Sun and M. Q.-H. Meng, “Autonomous robotic exploration by incremental road map construction,” IEEE Transactions on Automation Science and Engineering, vol. 16, no. 4, pp. 1720–1731, 2019.

[22] J. Williams, S. Jiang, M. O’Brien, G. Wagner, E. Hernandezet et al., “Online 3D frontier-based UGV and UAV exploration using direct point cloud visibility,” in IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems (MFI), Karlsruhe, Germany, pp. 263–270, 2020.

[23] S. Liu, S. Li, L. Pang, J. Hu, H. Chenet et al., “Autonomous exploration and map construction of a mobile robot based on the TGHM algorithm,” Sensors, vol. 20, no. 22, pp. 490, 2020.

[24] S. Koenig and M. Likhachev, “D* Lite,” in Proceedings of the Eighteenth National Conf. on Artificial Intelligence, Edmonton, Canada, pp. 476–483, 2002.

[25] S. Koenig and M. Likhachev, “Fast replanning for navigation in unknown terrain,” IEEE Transactions on Robotics, vol. 21, no. 3, pp. 354–363, 2005.

[26] S. Koenig and M. Likhachev, “Improved fast replanning for robot navigation in unknown terrain,” in Proceedings of the IEEE International Conference on Robotics and Automation, Washington, DC, USA, 2002.

[27] S. Singh, R. Simmons, T. Smith, A. Stentz, V. Verma, A. Yahja and K. Schwehr, “Recent progress in local and global traversability for planetary rovers,” in Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 2000.

[28] D. Wettergreen, B. Dias, B. Shamah, J. Teza, P. Tompkins, C. Urmson, M. Wagner and W. Whittaker, "First experiments in sun-synchronous exploration," in Proceedings of the IEEE International Conference on Robotics and Automation, Washington, DC, USA, 2002.

[29] A. Kelly, O. Amidi, M. Happold, H. Herman, T. Pilarsky, P. Rander, A. Stentz, N. Vallidis and R. Warner, "Toward reliable autonomous vehicles operating in challenging environments," in Proceedings of the International Symposium on Experimental Robotics, Singapore, pp. 599–608, 2004.

[30] S. Goldberg, M. Maimone and L. Matthies, "Stereo vision and rover navigation software for planetary exploration," in Proceedings of the IEEE Aerospace Conference, Big Sky, MT, USA, 2002.

[31] M. Likhachev, G. Gordon and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," in Proceedings of the International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, pp. 767–774, 2003.

[32] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz and S. Thrun, "Anytime dynamic A*: An anytime, replanning algorithm," in Proceedings of the Int. Conf. on Automated Planning and Scheduling, Monterey, USA, pp. 262–271, 2005.

[33] C. L. Hwang and K. Yoon, "Methods for multiple attribute decision making," in Lecture Notes in Economics and Mathematical Systems-Multilple Attribute Decision Making. Vol. 186. Berlin, Germany: Springer-Verlag, pp. 58–191, 1981.

[34] D. Pamucar and L. Savin, "Multiple-criteria model for optimal off road vehicle selection for passenger transportation: BWM-COPRAS model," Military Technical Courier, vol. 68, no. 1, pp. 28–64, 2020.

[35] S. H. Zolfani, M. Yazdani, D. Pamucar and P. Zarate, "AVIKOR and TOPSIS focused reanalysis of the MADM methods based on logarithmic normalization," Facta Universitatis Series: Mechanical Engineering, vol. 18, no. 3, pp. 341–355, 2020.

[36] D. Lj. Petković, "Izbor biomaterijala - višekriterijumska analiza i razvoj sistema za podršku odlučivanju," Doktorska disertacija, Mašinski fakultet u Nišu, Univerzitet u Nišu, Niš, R. Srbija, 2016.

[37] E. Murphy, "Planning and exploring under uncertainty," Ph.D. thesis, Robotics Research Group, University of Oxford, Oxford, England, 2010.

[38] D. Ferguson and M. Likhachev, "Efficiently using cost maps for planning complex maneuvers," in Proceedings of International Conference on Robotics and Automation - Workshop on Planning with Cost Maps, 2008.

[39] M. Vilela, G. Oluyemi and A. Petrovski, "A holistic approach to assessment of value of information (VOI) with fuzzy data and decision criteria," Decision Making: Applications in Management and Engineer-ing, vol. 3, no. 2, pp. 97-118, 2020.

[40] A. Aroor and S. L. Epstein, "Toward crowd-sensitive path planning," in Proceedings of the AAAI Symposium on Human-Agent Groups, Arlington, VA, USA, pp. 238–245, 2017.

[41] A. Aroor, S. L. Epstein and R. Korpan, “Online learning for crowd-sensitive path planning,” in Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, Stockholm, Sweden, pp. 1702–1710, 2018.

[42] J.-M. Lien and Y. Lu, “Planning motion in environments with similar obstacles,” in 2009 Conference Robotics: Science and Systems V, 2009.

[43] S. R. Martin, S. E. Wright and J. W. Sheppard, “Offline and online evolutionary bidirectional RRT algorithms for efficient replanning in environments with moving obstacles,” IEEE International Conference on Automation Science and Engineering, Scottsdale, AZ, USA, 2007.

[44] M. Zucker, J. Kuffner and M. Branicky, “Multipartite RRTs for rapid replanning in dynamic environments,” in Proceedings of the IEEE International Conference on Robotics and Automation, Rome, Italy, pp. 1603-1609, 2007.

[45] V. Hwang, M. Phillips, S. Srinivasa and M. Likhachev, “Lazy validation of experience graphs,” IEEE International Conference on Robotics and Automation, Seattle, WA, USA, pp. 912-919, 2015.

[46] M. Phillips and M. Likhachev, “Speeding up heuristic computation in planning with experience graphs,” IEEE International Conference on Robotics and Automation, Seattle, WA, USA, pp. 893-899, 2015.

[47] M. Phillips, B. Cohen, S. Chitta and M. Likhachev, “E-graphs: Bootstrapping Planning with Experience Graphs,” Conference Robotics: Science and Systems VIII, 2012.

[48] S. M. LaValle, “Planning algorithms,” Cambridge University Press, 2006.

[49] D. Ferguson and A. Stentz, “Field D*: An interpolation-based path planner and replanner,” Robotics Research - Results of the 12th International Symposium of Robotics Research, San Francisco, California, USA, pp. 239-253, 2005.

[50] X. Sun, “Incremental search-based path planning for moving target search,” Ph.D. thesis, Faculty of the USC Graduate School, University of Southern California, USA, 2013.

[51] E. W. Dijkstra, “A note on two problems in connexion with graphs,” Numerische Mathematik 1, pp. 269-271, 1959.

[52] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz and S. Thrun, “Anytime search in dynamic graphs,” Artificial Intelligence, vol. 172, no. 14, pp. 1613-1643, 2008.

[53] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz and S. Thrun, “Anytime Dynamic A*: The proofs,” Technical Report, CMU-RI-TR-05-12, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA, 2005.

[54] T. Coffee, “Fast optimal kinodynamic path planning for autonomous vehicles in dynamic environments,” Robotics Science & Systems II: Final Project Report, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 2006. Dostupno (na dan 23. oktobar 2016.) na: *tcoffee@mit.edu*.

[55] D. Ferguson, M. Likhachev and A. Stentz, “A guide to heuristic-based path planning,” in Proceedings of the Workshop on Planning under Uncertainty for Autonomous Systems at The International Conference on Automated Planning and Scheduling, 2005.

[56] S.A.M. Coenen, “Motion planning for mobile robots - a guide,” M.S. thesis, Eindhoven University of Technology, Eindhoven, Netherlands, 2012.

[57] D. H. Kim, N. T. Hai and S. K. Jeong, “A guide to selecting path planning algorithm for automated guided vehicle (AGV),” in AETA 2017-Recent Advances in Electrical Engineering and Related Sciences: Theory and Application, Ho Chi Minh City, Vietnam, pp. 587–596, 2017.

[58] B. Brumitt and A. Stentz, “GRAMMPS: A generalized mission planner for multiple mobile robots in unstructured environments,” in Proceedings of the IEEE International Conference on Robotics and Automation, Leuven, Belgium, 1998.

[59] K. Al-Mutib, M. AlSulaiman, M. Emaduddin, H. Ramdane and E. Mattar, “D* Lite based real-time multi-agent path planning in dynamic environments,” in Proceedings of the Third International Conference on Computational Intelligence, Modelling & Simulation, Langkawi, Malaysia, 2011.

[60] J.-H. Peng, I.-H. Li, Y.-H. Chien, C.-C. Hsu and W.-Y. Wang, “Multi-robot path planning based on improved D* Lite algorithm,” in Proceedings of the IEEE 12th International Conference on Networking, Sensing and Control, Taipei, Taiwan, 2015.

[61] N. Zagradjanin, A. Rodic, D. Pamucar and B. Pavkovic, “Cloud-based multi-robot path planning in complex and crowded environment using fuzzy logic and online learning,” Information Technology and Control, vol. 50, no. 2, pp. 357–374, 2021.

[62] N. Zagradjanin, D. Pamucar and K. Jovanovic, “Cloud-based multi-robot path planning in complex and crowded environment withmulti-criteria decision making using full consistency method,” Symmetry, vol. 11, no. 10, pp. 1–15, 2019.

[63] M. Jann, S. Anavatti and S. Biswas, “Path planning for multi-vehicle autonomous swarms in dynamic environment,” in Ninth Int. Conf. on Advanced Computational Intelligence (ICACI), Doha, Qatar, pp. 48–53, 2017.

[64] V. M. Mármol, “Autonomous multi-robot exploration & coverage,” Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA. Dostupno (na dan 27. februar 2022.) na: *http://www.cs.cmu.edu/~humanrobotteams/multimedia/presentations/Auto_Exploration_Coverage.pdf.*

[65] A. Rusu, “Path planning and autonomous navigation for a planetary exploration rover,” Ph.D. thesis, University of Toulouse, Toulouse, France, 2014.

[66] M. Selin, M. Tiger, D. Duberg, F. Heintz and P. Jensfelt, “Efficient autonomous exploration planning of large scale 3D-environments,” IEEE Robotics and Automation Letters, vol. 4, no. 2, pp. 1699–1706, 2019.

[67] W. Gao, M. Booker, A. Adiwahono, M. Yuan, J. Wang *et al.*, “An improved frontier-based approach for autonomous exploration,” in 15th IEEE Int. Conf. on Control, Automation, Robotics and Vision (ICARCV), Singapore, pp. 292–297, 2018.

[68] G. Li, W. Chou and F. Yin, “Multi-robot coordinated exploration of indoor environments using semantic information,” Science China Information Sciences, vol. 61, no. 7, pp. 1–8, 2017.

[69] J. J. Lopez-Perez, U. H. Hernandez-Belmonte, J. P. Ramirez-Paredes, M. A. Contreras-Cruz and V. Ayala-Ramirez, “Distributed multirobot exploration based on scene partitioning and frontier selection,” Mathematical Problems in Engineering, vol. 2018, pp. 1–17, 2018.

[70] M. Kulich, T. Juchelka and L. Preucil, “Comparison of exploration strategies for multi-robot search,” Acta Polytechnica, vol. 55, no. 3, pp. 162–168, 2015.

[71] A. Franchi, L. Freda, G. Oriolo andM. Vendittelli, “A decentralized strategy for cooperative robot exploration,” in Proceedings of the 1st Int. Conf. on Robot Communication and Coordination, Athens, Greece, pp. 1–8, 2007.

[72] B. Yamauchi, A. Schultz, W. Adams and K. Graves, “Integrating map learning, localization and planning in a mobile robot,” in Proceedings of the IEEE Int. Sym. on Intelligent Control, Gaithersburg, Maryland, USA, pp. 331–336, 1998.

[73] W. Burgard, M. Moors, C. Stachniss and F. Schneider, “Coordinated multi-robot exploration,” IEEE Transactions on Robotics, vol. 21, no. 3, pp. 376–386, 2005.

[74] T. Cieslewski, E. Kaufmann and D. Scaramuzza, “Rapid exploration with multi-rotors: A frontier selection method for high speedflight,” in IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Vancouver, British Columbia, Canada, pp. 2135–2142, 2017.

[75] H. Gao, W. Huang and X. Yang, “Applying probabilistic model checking to path planning in an intelligent transportation system using mobility trajectories and their statistical data,” Intelligent Automation & Soft Computing, vol. 25, no. 3, pp. 547–559, 2019.

[76] R. Shrestha, F.-P. Tian, W. Feng, P. Tan and R. Vaughan, “Learned map prediction for enhanced mobile robot exploration,” in Proceedings of the Int. Conf. on Robotics and Automation, Montreal, Quebec, Canada, pp. 1197–1204, 2019.

[77] S. Oßwald, M. Bennewitz, W. Burgard and C. Stachniss, “Speeding-up robot exploration by exploiting background information,” IEEE Robotics and Automation Letters, vol. 1, no. 2, pp. 716–723, 2016.

[78] A. Viseras Ruiz and C. Olariu, “A general algorithm for exploration with gaussian processes in complex, unknown environments,” in Proceedings of the IEEE Int. Conf. on Robotics and Automation, Seattle, Washington, USA, pp. 3388–3393, 2015.

[79] D. Holz, N. Basilico, F. Amigoni and S. Behnke, “A comparative evaluation of exploration strategies and heuristics to improve them,” in Proceedings of the European Conf. on Mobile Robotics, Oerebro, Sweden, 2011.

[80] R. Graves and S. Chakraborty, “A linear objective function-based heuristic for robotic exploration of unknown polygonal environments,” Frontiers in Robotics and AI, vol. 5, pp. 325, 2018.

[81] F. Amigoni and A. Gallo, “A multi-objective exploration strategy for mobile robots,” in Proceedings of the IEEE Int. Conf. on Robotics and Automation, Barcelona, Spain, pp. 3850–3855, 2005.

[82] B. Tovar, L. Munoz-Gomez, R. Murrieta-Cid, M. Alencastre-Miranda, R. Monroyet *et al.*, “Planning exploration strategies for simultaneous localization and mapping,” Robotics and Autonomous Systems, vol. 54, no. 4, pp. 314–331, 2006.

[83] C. Stachniss and W. Burgard, “Exploring unknown environments with mobile robots using coverage maps,” in Proceedings of the 18th Int. Joint Conf. on Artificial Intelligence, Acapulco, Mexico, pp. 1127–1132, 2003.

[84] R. Semenas and R. Bausys, “Autonomous navigation in the robots’ local space by multi criteria decision making,” Open Conf. of Electrical, Electronic and Information Sciences, Vilnius, Lithuania, pp. 1–6, 2018.

[85] H. Ardiny, S. Witwicki and F. Mondada, “Autonomous exploration for radioactive hotspots localization taking account of sensor limitations,” Sensors, vol. 19, no. 20, pp. 292, 2019.

[86] N. Ç. Polat, G. Yaylali and B. Tanay, “A method for decision making problems by using graph representation of soft set relations,” Intelligent Automation & Soft Computing, vol. 25, no. 2, pp. 305–311, 2019.

[87] S. Thrun, W. Burgard and D. Fox, “Probabilistic Robotics (Intelligent Robotics and Autonomous Agents),” The MIT Press, 2005.

[88] I. Petrovic and M. Kankaras, “A hybridized IT2FS-DEMATEL-AHP-TOPSIS multicriteria decision making approach: Case study of selection and evaluation of criteria for determination of air traffic control radar position,” Decision Making: Applications in Management and Engineering, vol. 3, no. 1, pp. 146–164, 2020.

[89] K. R. Ramakrishnan and S. Chakraborty, “A cloud TOPSIS model for green supplier selection,” Facta Universitatis Series: Mechanical Engineering, vol. 18, no. 3, pp. 375–397, 2020.

[90] T. Cakar and B. Çavuş, “Supplier selection process in dairy industry using fuzzy TOPSIS method,” Operational Research in Engineering Sciences: Theory and Applications, vol. 4, no. 1, pp. 82–98, 2021.

[91] M. Zizovic, D. Pamucar, B. Miljkovic and A. Karan, “Multiple-criteria evaluation model for medical professionals assigned to temporary SARS-CoV-2 hospitals,” Decision Making: Applications in Management and Engineering, vol. 4, no. 1, pp. 153–173, 2021.

[92] E. K. Zavadskas, A. Kaklauskas, A. Banaitis and N. Kvederyte, “Housing credit access model: The case for Lithuania,” European Journal of Operational Research, vol. 155, no. 2, pp. 335–352, 2004.

[93] T. Milosevic, D. Pamucar and P. Chatterjee, “A model for selection of a route for the transport of hazardous materials using fuzzy logic system,” Military Technical Courier, vol. 69, no. 2, pp. 355–390, 2021.

[94] Z. Ali, T. Mahmood, K. Ullah and Q. Khan, "Einstein geometric aggregation operators using a novel complex interval-valued pythagorean fuzzy setting with application in green supplier chain management," Reports in Mechanical Engineering, vol. 2, no. 1, pp. 105–134, 2021.

[95] M. R. Gharib, "Comparison of robust optimal QFTcontroller with TFC andMFC controller in amulti-input multi-output system," Reports in Mechanical Engineering, vol. 1, no. 1, pp. 151–161, 2020.

[96] S. Kayapinar Kaya, "Evaluation of the effect of COVID-19 on countries' sustainable development level: A comparative MCDM framework," Operational Research in Engineering Sciences: Theory and Applications, vol. 3, no. 3, pp. 101–122, 2020.

[97] N. Zagradjanin, D. Pamucar, K. Jovanovic, N. Knezevic and B. Pavkovic, "Autonomous exploration based on multi-criteria decision-making and using D* Lite algorithm," Intelligent Automation & Soft Computing, Special Issue: Soft Computing Methods for Intelligent Automation Systems, pp. 1369-1386, 2021.

[98] R. Yan, H. Zhang, H. Lu, J. Xiao, X. Chen, Y. Li, Q. Qiu, S. Zhu and C. Shi, "RoboCup Rescue 2018 team description paper NuBot," National University of Defense Technology, Changsha, Hunan, China. Dostupno (na dan 27. februar 2022.) na: *https://robocup-rescue.github.io/team_description_papers/2018/Champ2018_China_NuBot_TDP.pdf*.

[99] K. Alexis, "Resilient autonomous exploration and mapping of underground mines using aerial robots," in Proceedings of the 19th Int. Conf. on Advanced Robotics (ICAR), Belo Horizonte, Brazil, pp. 1–8, 2019.

[100] R. Mata, "Persistent autonomous exploration, mapping and localization," M.S. thesis, Massachusetts Institute of Technology, Massachusetts, USA, 2017.

[101] L. Hassan, S.H. Sadati and J. Karimi, "Integrated fuzzy guidance law for high maneuvering targets based on proportional navigation guidance," Iranian Journal of Electrical and Electronic Engineering, vol. 9, no. 4, pp. 204-214, 2013.

[102] O. Saha and P. Dasgupta, "Fast path planning using experience learning from obstacle patterns," Conference AAAI Spring Symposium Series, Palo Alto, California, USA, pp. 60-67, 2016.

[103] G. Dillard, "A moving window detector for binary integration," IEEE Transactions on Information Theory, vol.13, no. 1, pp. 2-6, 1967.

[104] Robotis. Dostupno (na dan 27. mart 2022.) na: https://www.robotis.us/turtlebot-3/.

Биографија аутора

Новак Заграђанин, дипломирани инжењер електротехнике, рођен је 08.02.1979. године у Сјеници - Република Србија. Основну школу је завршио у Сјеници 1994. године са одличним успехом као носилац Вукове дипломе, након чега је уписао Војну гимназију, коју је такође завршио са одличним успехом 1998. године, као други у класи. Војнотехничку академију, смер ваздухопловнотехничке службе, специјалност електроника ракетних система, завршио је 2003. године, са просечном оценом 8.81, као први у класи на поменутом смеру. Члан је удружења *MENSA*.

У периоду од 2003. до 2011. године службовао је у 250. рбр за ПВД, где је успешно обављао већи број дужности. Године 2011. је постављен је на дужност вишег истраживача у Одељењу за вођење и управљање ракета Сектора за ракетно наоружање Војнотехничког института.

У Управу за одбрамбене технологије Сектора за материјалне ресурсе Министарства одбране постављен је 2013. године, где је обављао дужност референта за ресурсе средстава ПВО у Групи за ресурсе НВО (наоружање и војна опрема) РВиПВО Одељења за ресурсе НВО. Године 2018. је постављен на садашњу дужност координатора за развој средстава НВО у Одељењу за заједничке послове Сектора за материјалне ресурсе Министарства одбране.

У оквиру садашње дужности прописане су му обавезе да прати рад, врши координацију и усмерава активности на пројектима везаним за развој наоружања и војне опреме у ВТИ, привредним друштвима из групације Одбрамбене индустрије Србије и осталим организацијама које се баве развојем НВО, као и активности везане за проширење и јачање технолошке базе од интереса за систем одбране.

У оквиру даљег војностручног усавршавања завршио је Основни командно-штабни курс (2013. године) и Командно-штабно усавршавање (2019. године) у организацији Универзитета одбране. У току досадашње службе оцењиван је одличним службеним оценама. Завршио је већи број курсева у земљи, као и један курс у иностранству (Оберамергау, Немачка), на тему развоја и опремања средствима наоружања и војне опреме. Био је члан више тимова Министарства одбране и Војске Србије за опремање ВС различитим врстама наоружања и војне опреме.

Докторске студије на Електротехничком факултету у Београду уписао је школске 2012./2013. године на модулу „Управљање системима и обрада сигнала”. Током докторских студија испунио је све обавезе и положио све испите предвиђене наставним планом и програмом, са просечном оценом 9,70.

Први је аутор три рада публикована у еминентним међународним научним часописима са импакт фактором, који су до тренутка писања овог текста цитирани више десетина пута, као и три рада презентована на научним конференцијама међународног карактера.

Изјава о ауторству

Име и презиме аутора: Новак Загрђанин
Број индекса: 5063/2012

Изјављујем

да је докторска дисертација под насловом

"Планирање путање робота базирано на D* Lite алгоритму и аутономној претрази окружења"

- 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, 28.03.2022. године

НЗагрђанин

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора:	Новак Загрђанин
Број индекса:	5063/2012
Студијски програм:	Електротехника и рачунарство (модул Управљање системима и обрада сигнала)
Наслов рада:	"Планирање путање робота базирано на D* Lite алгоритму и аутономној претрази окружења"
Ментор:	др Коста Јовановић, ванредни професор

Изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла ради похрањивања 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.

У Београду, 28.03.2022. године

Потпис аутора

Н.Загрђанин

Изјава о коришћењу

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

“Планирање путање робота базирано на D* Lite алгоритму и аутономној претрази окружења”

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, 28.03.2022. године

НЗаграђанин

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.