Технически университет – София

Факултет Компютърни Системи и Технологии

ДИСЕРТАЦИОНЕН ТРУД

на тема

Изследване приложението на оптимизационни модели за навигация при група мобилни роботи

за присъждане на образователна и научна степен „**ДОКТОР**"

по професионално направление

5.3 Комуникационна и компютърна техника

специалност „Системи с изкуствен интелект"

Автор: **маг. Антуан Христов Ангелов**

Научни ръководители: **проф. д-р инж. Румен Иванов Трифонов**
проф. д-р инж. Огнян Наков Наков

София, 2023

СЪДЪРЖАНИЕ

Table of Contents

Списък на фигурите

Списък на таблици

Общоприети съкращения

$X \times Y$	Векторно произведение на векторите X и Y
HP	Host Prevention

Използвани английски съкращения

ACK - Acknowledgement
AMS - Agent Management System
API - Application Programming Interface
AS - Agents Society
AP - Agent Platform
CI - Computational intelligence
DR - степента на откриване
FIPA - Foundation for Intelligent Physical Agents
FRCM - fuzzy rough c-means algorithm
FRL - Fuzzy Armor Learning
GNP - Genetic Network Programming
GUI - Graphical User Interface
HTTP - Hypertext Transfer Protocol
IoT - Internet of Things
MA - Управление на множество агенти
MAF - Mobile Agent Facility
MASIF - Mobile Agent System Interoperability Facility
MDP - Markov Decision Processes
OA - общата точност
OSI - Open Systems Interconnection Basic Reference Model
RL - Reinforcement Learning
SWOT анализ - Strengths Weaknesses Opportunities
SI – swarm intelligence
SVM - Support Vector Machines
HNAV („Hopfield Navigation”)
SLAM (Simultaneous Localization and Mapping)

Използвани български съкращения

БД - бази данни
ГА - Генетични алгоритми (ГА)
ЕС - Експертната система
ИИ - Изкуствен интелект
ИИС - Имунна изкуствена система (ИИС)
ИКТ - Информационни и комуникационни технологии
НМ - Невронни мрежи
РЛ - Размита логика
МКР - Мобилни колаборативни роботи
МР – Мобилни роботи
МСКР – Мобилни сервизни и колаборативни роботи
РС - роботизираните системи
УДНМР – Управление на движение и навигацията при мобилни роботи
УНМР – Управление навигацията на мобилни роботи
СУ – Системи управление
СМР - Системата от мобилни роботи
МРС - мултиагентна роботизирана система
ГМР – група мобилни роботи
САНС - софтуерната автономна навигационна система
СНПТГМР - системата за навигацията и планиране на траекторията на група мобилни роботи
НК - Невронна карта
КП – Конструктор на пътя
НМХ – Невронна мрежа „Хопфийлд“,
РЧ – „рояка частици“
СПТ - система за планиране траекторията
МПВ - Машина с поддържащи вектори
МАС - Мулти-агентни системи
МПП - Метода на потенциалните полета
ИПП - изкуствени потенциални полета
ЦИБ - Централен изчислителен блок
ДФМР – Динамична формация мобилни роботи
КГПЦ - Конструктор глобален път до целта
ХМП – хибриден многослоен планировчик
ЛИМ - локалния изчислителен модул
ОС – операционна система

УВОД

В последните години все повече индустриите се радват на растеж във внедряването на мобилни колаборативни роботи. Интереса и необходимостта идва от търсене на МКР, които да заменят част от ежедневните човешки дейности, свързани с работа във вредни среди, работа на труднодостъпни места, извършване на подпомагащи дейности като инспектиране на различни обекти или оптимизиране на транспортни услуги. Нещо повече роботите не винаги оперират самостоятелно и изолирано. А напротив, много често се налага те да си сътрудничат и кооперират. Такава системата от мобилни колобаративни роботи (МКР) може да бъде напълно самостоятелна, без да зависи от наличие на конкретна подреденост в работната среда, служеща им за ориентация или външна намеса за контрол. И този дисертационен труд се занимава с изследване на методи от изкуствения интелект, където чрез сътрудничество между екип мобилни роботи се оптимизира движението им в пространства и непозната среда.
Движението в популация от индивидуални агенти взаимодействащи си локално един с друг и със своята среда, и имаща за цел намиране на оптимума с алгоритъм за търсене в работното пространство. Така системата може да бъде напълно самостоятелна, без да зависи от наличие на конкретна подреденост в работната среда, служеща й за ориентация. Подобни решения са търсени и актуални при дълбоко-подводни изследвания, изучаване на други планети, или др. ситуации които са общия случай на работа в затворена среда (indoor).
Ето защо в дисертацията са засегнати някои фундаментални проблеми в управление на движението и навигацията при група от мобилни роботи функциониращи като мулти-агентна система. Под полезрението също попадат техники за комуникация и обмен на данни и информация нужни както за ориентирането на агентите в група така и за функциониране на системата като цяло.
Един от съществените приноси на дисертацията е задълбоченият анализ на видовете алгоритми за колективна интелигентност като оптимизационни модели нужни за навигация позициониране, координация с цел избягване на препятствия и предотвратяване на инциденти по време на движението на група роботи. Друг съществен принос от дисертацията е предлагане използване на децентрализирана софтуерна система за навигация с модифицирана „невронна карта“ „Хопфийлд“ за намиране на глобален път. Разработения контролер „HNAV“ на основата автоасоциатор и модел на „Хопфийлд“ успешно може да се комбинира с други методи, като например интелигентните потенциални полета [31] за избягване на локални препятствия при ситуации с колективно придвижване в динамични формации МР, и превръщайки процеса една в тривиална задача.

БЛАГОДАРНОСТИ

Изказвам благодарност на:
Двамата ми научни ръководител проф. д-р инж. Румен Трифонов и проф. д-р инж. Огнян Наков за полезните съвети и насоки, които ми даваха при разработването на настоящия дисертационен труд както и за указаното им доверие към мен да бъда докторант във факултета КСТ в ТУ и то точно в тази нова и набираща популярност научна област.
Моята съпруга и деца, които ме подкрепяха през целия период на докторантурата и проявиха търпение и разбиране по отношение на отделеното ми време.

ПЪРВА ГЛАВА. ЛИТЕРАТУРЕН ОБЗОР. ВЪВЕДЕНИЕ И ПОСТАНОВКА НА ЗАДАЧАТА

В тази глава е направен кратък преглед на методи и средства за осъществяването на една навигационна система. Ще бъде уточнен обхвата и задачите на дисертацията.

1.1. Въведение

1.1.1. Основни понятия и определения

Терминът "робот" произхожда от чешката дума "robota", която означава непосилен и тежък труд. През 1920 г. чешкият писател Карел Чапек в пиесата си „Росумски универсални роботи" въвежда в употреба за първи път в историята на човечеството думата „робот" като механичен работник, помагащи на човека[23].

В съвременният свят робота (роботизираната система) представлява сложна махано-електрическа система, състояща се както от електроника, пневматика, специализирани механични задвижвания, така и от високотехнологична конструкция с множество предавки, различни манипулационни приспособления, програмно осигуряване, средства за наблюдение и комуникация и т.н. В този смисъл роботите са типични мехатронни системи с пълен набор от съставящите ги елементи [24, 25]:

- Конструкция - тип механизъм за придвижване, задвижване с отворена/затворена кинематична структура за изпълняването на специфични цели. При мобилните роботи имаме съоръжение с основна цел придвижване и структура, която поддържа компоненти като вериги, колела, крака, и др. тип механизми за придвижване. Ето защо роботизираните системи може да бъдат *класифицирани* на мобилни и фиксирани в зависимост от това дали могат да се придвижват от една в друга точка на работното пространство. Дисертацията се фокусира върху мобилни роботи и разглежда основно взаимодействието между мобилните роботи за изпълнение на поставената задача *с диференциално колесно задвижване;*
- Изпълнителен механизъм – по определен закон отделните звена на механизма се задвижва, като преместват работния орган в зададена точка. Има различни и разнообразни изпълнителни механизми - от хващачи, специфични инструменти, технологично оборудване, сензори като камери и др.;
- Управляваща (навигационна и др.) и комуникационна система. При управлението на робота непрекъснато постъпват данни за състоянието на роботизираната система от заобикалящата го среда (работно пространство). На база на заданието от човека или контролни център се изработват управляващи въздействия за двигателните и изпълнителни механизми. Управляващите системи на МР могат да функционират както автономно и самостоятелно така и чрез периодични команди от наблюдаващ оператор или дистанционно управляеми с контролен център. Ако отчетем измененията както в работната среда така и в манипулираните обекти може да се обособят адаптивни роботи, каквито са и в частност кооперативните мобилни, върху които ще бъде проучването в дисертацията.

1.1.2. Навигацията и управлението на движението при група МР

Кооперативните роботи са такива работещи в условия на екип от n-бр., които изпълняват

цели в едно общ работно пространство и често имат една и съща навигационна цел. Такива роботи са в обхвата на изследването. Ако приемем, че работното пространство на мобилните роботи “С” е средата, в която те работят следва да има задача за планиране на маршрут и планиране на движение на група МР. Маршрутът е представен като непрекъсната крива в работното пространство променяно във времето оформя траектория, с помощта, на която е възможно да се изчислят скоростите и ускоренията на МР, а намирането му се нарича траектория или планиране на движение.

Днес, дори при наличие на глобална навигационна система, навигацията в затворени пространства остава нетривиален проблем. При условия на висока точност и бързодействие, са нужни нови и алтернативни навигационни и локализациони решения. Такива действия като автономно управление, разпознаване основни предмети както и манипулиране им трябва да бъдат рутина за МКР. А в извънредни ситуации трябва да се осигурят условия на непрекъснатост и децентрализация в групата за изпълнение на конкретните поставени задачи за по-висока вероятност за успешно завършване на задачата, постигната поради възможността за преразпределение на целите между роботите от групата в случай на проблем в някои от тях.

Списък с въпроси, на които един робот ще трябва да отговори за да добие навигационни способности може на кратко да се определи като:

1. Трябва да може да запомня пътя? Примерно с когнитивно картографиране.
2. Къде се намира? Локализация.
3. Къде трябва да отиде? Планиране на пътя.
4. Как мога да отида? Контрол на движенията

В дисертацията ще се прави изследване в обхвата на точка 3 без да навлизаме в детайли за останалите. Приемаме, че МР има останалите системи и както знае да картографира така знае и къде се намира във всеки един момент.

Самата системата за УДНМР включва: модул за управление на навигацията на мобилния робот включително неговата локализация в пространството, както и модул за координиране на поведението на робота при промяна на топологията на групата и заобикалящата го среда. В тези условия на кооперирane между МР, комуникационна система при управлението на групата от мобилни роботи и днес остава най-критичната част от хардуера. Протокола за комуникация трябва да бъде съобразен с факта, че данни за сензорите трябва да бъде споделена в реално време с всички останалите както и контролния център [AARTON2021]. В реално време трябва да се реагира, променя и адаптира, което налага да се внедри система за контрол и синтез на навигацията, управлявана от събития (event-driven) и комуникация в реално време [AARTON2021].

Задачата за придвижването на един МР в работната среда колкото и да е съвършен, никога не ще бъде тривиален проблем. Задачи от по-високо ниво, като търсене, събиране, картографиране и други подобни дейности са много комплексни за един автономен (самостоятелен) робот. Когато обаче има достатъчно голям брой роботи, работещи съвместно, тези задачи могат да бъдат изпълнени много по-лесно, тъй като данните от сензори включително за абсолютна локализация може бързо да бъде споделена между останалите участници в групата[21][22].

УДМР включва обширни и комплексни дейности като:

- *Локализация*. Един от първичните въпроси остава правилната ориентация с всички възможни похвати като одометрия (енкодери за мотори), релативна/инерционна (и абсолютна локализация или хибридна комбинации. Налични са и други 2 типа – триангулация (подход при който локализирането на обект в пространството става с измерване на ъглите, под които обекта вижда три или повече репери. Най-често това се извършва с лазер) и трилатерация (измерва

разстоянието от антени до обекта и за изчисление на позицията са нужни поне четири антени. Най-често се използва при сензори и МР с безжична свързаност – като GPS системите.);

- *Филтри*. При ползването на релативни (с потенциална опасност от динамично растящи грешки) техники за навигация и координация, винаги се налага и математическа оценка на неопределеността тъй като изчислената позиция не е предварително известна поради. Най-често срещания подход е да се използва филтър за оценяване на неопределеността - „Kalman“. Това е техника за сливане на данните от различни по-тип и вид сензори, за да се коригират грешките при абсолютни измервания в заложения модел и състояние на МР;
- *Контрол на задвижването* – в зависимост от типа задвижване като диференциално с 2/3 или 4/6 колела (2/3/4/6WD), или с всички (AWD). Допълнително типа на колелата също е от значение при използвания алгоритъм за управление. Колела от типа „Mecanum“ може да маневрира в четири посоки без завиване на оста. В този случай всички колела са оборудвани с малки ролки, които са разположени под 45-градусов ъгъл вместо стандартните джанти и се задвижват от индивидуален електромотор за всяко колело;
- *Управление и оптимизация на целия път* на МР от начална точка до целта с използването на различни модели и тактики;
- *Техники и алгоритми за избягване на препятствията* програмирани в софтуерната част;
- *Моделиране и изграждане на пътна карта*;

1.1.3. Подходи при решаване проблема с груповия контрол на МР

Мултиагентната система е съставена от агенти, където в повечето случаи тези агенти са изчислителни процеси (агентите могат да бъдат разпределени на различни компютри), но те също могат да бъдат робот, човек и т.н. В такава хипотеза попада и проектирането на навигационни алгоритми за решаването на проблема с груповия контрол. В дисертацията се разглежда кооперативно решаване на проблеми или кооперативната мултиагентна система, която има за цел да доведе до увеличаване на ефективността на общата система, а не изпълнението на отделните агенти. Всеки мобилен робот обработва само локална информация за сензори и навигация (в рамките на обхвата на собствените си сензори и комуникация) докато информация за групата и околната среда свързана с желаното поведение на групата като цяло трябва да се комуникира. В система с множество агенти, група на автономните агенти действа в среда за постигане на цел, която се постига чрез сътрудничество или конкуренция, споделяне или не споделяне на знания между тях.

Второ, аналитичният начин за получаване на поведенческите свойства, и натрупване на знания и характеристики на движението на група мобилни роботи е труден. От една страна, системата от уравнения на движение при група мобилни роботи е често нелинейна, от друга страна, взаимодействието между роботите се променя с времето поради промени в топологията на формацията и/или промяна в средата, в която те оперират.

И накрая, за изпълнението на алгоритмите за управление на ГМР се налага изискване всеки робот да действа координирано в реално време, което предполага използването на сигурни безжични комуникации [AARTON2019].

При управление на няколко робота може да се наложи да координирате действията на агентите и да разпределите количеството работа между тях. Съществуват различни

подходи за управление на група мобилни роботи в пространството:

- Група роботи (евентуално с изключение на лидера) е хомогенна, тоест членовете ѝ имат еднакви характеристики.
- Движението на група мобилни роботи се извършва на открито пространство или в помещение, чийто план е известен предварително.
- Топологията им остава непроменена през цялото време.

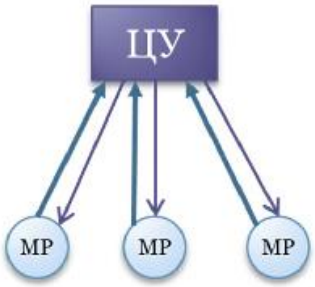
Но на практика характеристиките не винаги може да са така: на роботите често се налага и могат да се движат при липса на априорни познания за външната среда и е необходимо да се промени топологията на групата поради променящата се външна среда или задачата.

В момента за управление на групи роботи се предлага да се използват три основни архитектури за система за управление (СУ)[43]:

- централизиран;
- децентрализиран;
- хибриден.

В момента, повечето случаи използват централизирани подход при изпълнението на СУ в групата, като информацията за средата трябва да е достатъчна, за да могат действията на групата да се планират предварително и в реално време.

Основното предимство на централизирания подход е неговата лекота на използване при групово управление (фигура 1-1).



Фиг. 1-1 Схема на централно управление при МР в група с 2 канала – информационен и контролен

Също така този подход се отличава със специализацията на функциите на членовете на групата (колкото по-голяма е групата, толкова по-тясна е специализацията) и по този начин ви позволява да генерирате най-кратката последователност от групови действия за постигане на крайното решение [15].

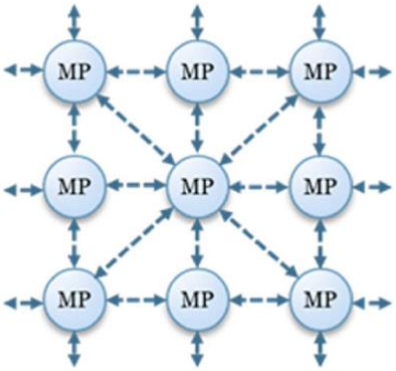
Въпреки това, централизираната архитектура също има редица значителни недостатъци:

- *ограничено мащабиране на системата от мобилни роботи (СМР).* С увеличаване на броя на роботите в групата, съответно и размерът на проблема за решаване с централно управление (ЦУ) ще се увеличи. Това поставя високи изисквания към скоростта на контролния център (КЦ) за изчислителните му способност и съответно мощност на хардуера и енергия на консумация;
- *скоростта и сложността на организиране на обмена на информация между МР и КЦ*. Тъй като планирането на действията на групата се извършва централизирано, всеки робот трябва непрекъснато да предава информация за текущото си състояние и състоянието на околната среда към контролния център. Въз основа на получената информация контролният център генерира план за действие (или го коригира, ако е необходимо) и след това генерира и предава команди за извършване на тези действия към изпълнителните системи на съответните роботи. След това цикълът на системата се повтаря отново, като се отчита нова информация за състоянието на роботите и околната среда, докато се

достигне крайната цел. Такива цикли на обмен на информация също налагат ограничения върху скоростта на вземане на решения и реакцията на МР при наличие на външни смущения [AARTON2019];

- *ограничен обхват.* За МР, работещи на значително разстояние от стационарен контролен център, е необходимо да има мощни приемо-предаватели на борда, което не винаги е възможно поради редица ограничения на мобилните бордови системи по отношение на полезен товар и енергийни ресурси [AARTON2019];
- *нисък процент отказоустойчивост*. Отказът на контролния център води до невъзможност за изпълнение на общия план за действие за постигане на целта от членовете на групата, което всъщност води до отказ на цялата група от МР [AARTON2019].

Един от обещаващите подходи за премахване на тези ограничения е децентрализираният подход. Архитектурата на децентрализирана СУ предполага разпределението на функциите за управление, движение и координация между членове на групата, т.е. всеки МР може да взема решения самостоятелно, като обменя информация с други членове на групата за постигане на обща цел (Фигура 1-2).



Фиг. 1-2 Схема на децентрализирано управление при МР в група с многоканалност, специфичен протокол на комуникация

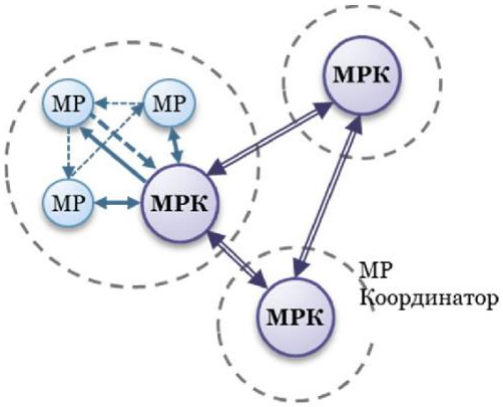
Основната характеристика на децентрализираната система е възможността за мащабируемост на изчислителните ѝ ресурси (по аналогия с клъстерните изчислителни системи). Така децентрализираното управление има следните предимства:

- производителността на всеки отделен изчислителен възел при децентрализирана мрежа е обект на по-малко изисквания за производителност в сравнение с централизирана система, тъй като измерението на проблема, решен на отделен МР (отделен възел), е много по-малко от измерението на проблема разрешаван от ЦУ. Това прави възможно реализирането на изчислителен мощност под формата на компактно микропроцесорно устройство, поставено директно на борда на МР;
- всеки отделен агент в групата може да взаимодейства само с най-близките членове на групата, като същевременно обменя и по-малко количество информация [AARTON2021]. По този начин изискванията за честотната лента и обхвата на информационните канали са значително намалени в сравнение с централизирана система за управление, която позволява използването на по-енергийно ефективни приемо-предаватели и комуникационни протоколи за обмен на данни между МР [AARTON2019], [AARTON2021];
- осигурява се висока надеждност на системата, т.е. провал на отделен робот (изчислителен възел) не води до неработоспособност на цялата група. Също така, при извънредна ситуация с отделен робот задачите може да се преразпределят

към останалите членове на групата по време на следващия итеративен цикъл на обмен на информация и цялостният план за действие да се коригира в реално време[15];
- осигурява високо ниво на адаптация на системата към външни смущения, т.е. промяната на задачата за всеки отделен МР е по-малко и решението/реакцията ще е в реално време [15].

Децентрализираният подход обаче има и своите недостатъци, които са свързани с факта, че в рамките на поведенческия модел, когато всеки член на екипа може самостоятелно да взема решения, се изисква също така да се осигури високо „интелектуално ниво“ на ниво всеки агент. Това е доста трудна задача [15] защото отново поставя изисквания за по-специален хардуер (примерно за обработка на техники за ИИ).

За преодоляване на редица трудности, които възникват при изграждането на децентрализирана система на управление, е възможно да се използва хибриден мулти-агентен подход за управление на група роботи[90]. Същността на този подход се състои във факта, че синтезът на плана за действие за постигане на обща цел, както и координацията на взаимодействието на агентите в група, се извършва с помощта на специални „координиращи агенти“ (Фигура 1-3). Координиращия агент може да бъде отделно звено обособено извън МР – като централен блок или система за управление.



Фиг. 1-3 Схема на хибридно управление и комуникация при МР в група с „координиращи агенти“ [AARTON2021].

По добрия вариант е един или няколко агента в една мултиагентна система да е МР, способен да действа в интерес на постигането на поставените от потребителя общи цели в рамките на групова или индивидуална задача за даден робот. Агентите имат редица характерни свойства [91]:

- адаптивност – агентът е в състояние да се адаптира към промените в средата, т.е. реагира на външни смущения;
- автономност - агентът е в състояние да декомпозира задачата, както и да взема оперативни решения като част от нейното изпълнение;
- сътрудничество – агентът може да взаимодейства с други агенти, като предоставя каквато и да е информация на агентите от групата (при поискване), изисква или препраща (пренасочва) информация от други агенти и въз основа на тази информация извършва съвместни действия.

Всеки робот в мултиагентна роботизирана система (МРС) е интелигентен мехатронен обект със собствена база данни и знания. За попълване на базите данни и знания се използват специални методи за обучение и адаптация. Тези методи се прилагат както на тактическо (локално) ниво (т.е. за отделни роботизирани агенти), така и на стратегическо (надзорно) ниво на контрол [91].

Тези методи и техните модификации позволяват решаването на следните проблеми с

интелигентно управление [92]:

- планиране на траекторията на движение на робот-агент в среда с препятствия с помощта на локална (сензорна) или глобална (надзорна) информация;
- моделиране във виртуалното пространство на агент на околната среда и поведението на други роботи;
- разпознаване във виртуалното пространство на ситуации и вземане на оптимални решения;
- адаптивен контрол на движението на агентите по планирани траектории.

Принципът на действие на СУ за мулти-агенти се основава на разлагането на общата задача, която трябва да се изпълнява от МРС, върху множество взаимосвързани локални подзадачи, чието решение е възложено на интелигентната СУ за отделните агенти, и решаването на общата задача е възможна само колективно, т.е. ресурсите на отделните агенти са достатъчни само за решаване на техния локален проблем. За изпълнение на всички задачи на МРС е необходимо да се синтезира координиран график на дейностите на агентите, като се вземат предвид ограниченията за броя и видовете агенти и тяхната взаимозаменяемост при изпълнение на конкретни задачи. По този начин архитектурата на системата за планиране и управление на взаимодействието на агентите в МРС има разпределен характер и йерархична организация: на горното (надзорно) ниво има „координиращ агент“, който има директни и обратни връзки с всички местни МР агенти. „Координаторът“ декомпозира общата задача на локални подзадачи и ги разпределя между агентите а агентите на тактическо ниво осигуряват изпълнението на своите локални подзадачи за най-кратко време, което води до изпълнение на общата задача също за най-кратко време [91].

Когато групата МР се придвижи към дадена цел, задачата на координиращия агент и останалите в близост е бързо да преминат планираните маршрути без сблъсъци, за да изпълнят общата задача. Предотвратяването на сблъсъци (конфликти) между агенти в процеса на движение може да се извършва както на надзорно ниво (от координиращия агент), така и на тактическо ниво на оперативните агенти-МР. В този случай разрешаването на конфликти ще се основава на използването на мулти-агентни експертни правила за „трафик“ и алгоритми за разпознаване на конфликтни (аварийни) ситуации във виртуалното пространство [92].

1.2. Навигационни модели използвани при УДМР

Планирането на най-кратките маршрути за агентите се извършва с помощта на специализирани алгоритми за търсене на траектории, базирани както на класически изчислителни методи, така и на усъвършенствани интелектуални подходи и техники от ИИ, които се представят по долу.

Преди да се разгледа в детайли областта на планиране движението на роботите [11] може да обобщи, че изследванията са разработени както на основа на методите за прилагане на класически т.н. *базови алгоритми* за търсене (алгоритъмът на „Дейкстра“, търсене в дълбочина, „A*“ или др. класически подходи), така и на новите тенденции с интелигентни алгоритми, базирани на невронни мрежи (НМ), размити множества, стохастични и генетични алгоритми.

Когато в реално време се обработват сензорите на МР за да се планира движението му може да се говори за *локално навигация*. При *глобална навигация* се изгражда пътна карта на цялото работно пространство без да се знае ситуацията „на дадено конкретно място“ по време когато МР ще го достигне.

Към днешна дата изследванията в областта на колективното движение на роботите могат

да бъдат обособени *в следните области*:

- Типично математически и статистически подходи и решения за намиране на най-кратък път. Статистическите системи за разлика от адаптивните нямат каквито и да е способности за коригиране и приспособяване в динамична среда;
- Технологии от мулти-агентни системи (МАС) [42], където МР и препятствията се възприемат като отделни агенти. Такава система може да се определи като набор от система за управление, роботи, работещи в едно и също работно пространство при определени условия с поставена обща роботизирана задача. Множество роботи може да се възползва от поведението на сътрудничество, ако с някакъв критерий за оценка на сътрудничество или координация може да се предложи увеличение на общата полезност на системата като цяло[58];
- Симулационно моделиране, т.е. внедряване на модели на взаимодействие между субекти, като за основа се вземат биологични обекти. Тук може да се отдаде на изследвания в областта в изкуствения интелект като невронни мрежи;
- Специфични изкуствени адаптивни системи и алгоритми като интелигентност на рояка. Физически системи, които имат способността да се адаптират на промените в основната среда. Адаптивната система може да реагира на нови ситуации. Това са методи, които изследват външните, чисто феноменологични аспекти на поведението на живите организми. В имената на повечето референтни публикации се появява думата „рояк“ – и „ято“. Наистина, много изследвания черпят вдъхновение от света на насекомите и повечето от тях правят аналогии с колонии от мравки, пчели или други организми;
- Стохастични модели;
- Еволюционни методи. Основната задача е внедряването по еволюционен начин и механизъм на вътрешно взаимодействие – тук попадат генетичните алгоритми;
- Хибридни – с комбинация от посочените горни методи.

Алгоритмите използвани при навигацията на МР, формират голямо звено в кооперативната и роева роботика. Напоследък с развитието на автономното движение при автомобилите, тази област се развива с главоломна скорост. Главната идея на планирането е избягването на препятствия включително сблъсъци между самите тях, докато роботът се придвижва от точка А до точка Б при минимум (минимално) [26][27][28] усилие и разстояние. Без подобни алгоритми и техники, МКР не биха могли да функционират коректно в работната си среда и движенията им не биха били оптимални по отношение на изразходвано време и енергия. Оптимизацията е наука с много богато минало и затова е изучавана е още от 18 век с появата на теорията на графите и проблема „Piano Mover“ [29][30]. Принципно, в зависимост от наличната сензорна информация, можем да разделим техниките за планиране на глобални и локални. Глобалните методи са по-оптимални по отношение на точност, но изискват пълно познаване на работната среда и възможните препятствия предварително. За това и са чувствителни към грешки породени от непознаването на цялото работно поле. Подобни грешки са попадане в локален минимум при ползването на алгоритми базирани на потенциални полета [31][32]. Затова комбинирането на глобални и локални методики е най-добрия избор за построяване на стратегия за движение в работната среда [33][34][35].

По-долу се предоставя кратък преглед на *основните подходи за навигация* на роботи. Повечето от подходите се занимават основно с проблема на планирането на пътя и избягването на появилите се препятствия. В общия случай се предпочита да се разглеждат като подходи за навигация, тъй като това е общият проблем, който се опитват да разрешават.

Условно тези подходите за намиране на път могат да бъдат разделени на 3 групи:

- алгоритми за просто избягване на препятствия;
- методи за намиране на път по протежение на графиката;
- интелигентни алгоритми.

Прилагането на конкретен тип алгоритъм се обуславя от съответните критерии, които определят ефективността на прилагането на този алгоритъм за решаване на зададения контролен проблем.

1.3. Класически подходи за намиране на най-кратък път

Дисциплината стартира в средата на 60-те, но едва с революционния принос на Лозано-Перес [12] в пространственото планиране на движението (ПД) привлече вниманието на повечето изследователи. Доказано е, че проблемът за планиране на пътя е NP-пълен [13]. Познатите класически методи са всъщност варианти на няколко общи подхода като: „пътна карта“, „клетъчна декомпозиция“, „потенциални полета“ [98] и „математическо програмиране“. Повечето класове проблеми при ПД могат да бъдат решени с помощта на тези подходи. Те не са непременно взаимно изключващи се и комбинация от тях често се използва при разработване плана за движение [16]. При подхода на пътната карта, свободното C-пространство от набор на възможни движения, се налага върху мрежа от едномерни линии. Този подход се нарича още подход на „прибиране“, „скелет“ или „магистрала“. Търсеното решение е ограничено до мрежата и ПД се преобразува в решение на проблеми за търсене в граф. Добре познатите ни „пътни карти“ са „графики на видимост“, „диаграми на Вороной“, „силует“ и „мрежа от подцели. Графиката на видимостта (ГВ) е колекция от линии в свободното пространство, която свързва характеристика на даден обект с тази на друг. В основната си форма тези характеристики са върхове на многоъгълни препятствия и има вида O(n^2) ръбове в графиката на видимостта, които могат да бъдат конструирани в O(n^2) време и пространство в 2D, където n е броят на характеристиките. Идеята за използване на “графика на видимостта” за планиране движението на робот е използвана в [17].

По долу са изброени най-известните класически подходи:

А/ Методи за пътна карта

Методите на пътната карта се опитват да намалят размерността на пространството, в което се извършва планирането на пътя и навигацията. Това става чрез улавяне на структурата на свободното пространство чрез набор от едномерни криви и техните взаимовръзки, пътната карта. Резултатът е структура, подобна на графика, където пътищата могат да се планират лесно. Пълният път се състои от три компонента: начален сегмент на пътя, който свързва текущата конфигурация с пътната карта, главният път, който се намира върху пътната карта и крайният сегмент, който свързва крайната конфигурация с пътната карта.

Добре известни методи в тази категория са диаграмите на Вороной и графиките на видимостта. Свързан проблем в този случай е как може да бъде изградена пътната карта (изчислително тежък проблем, като цяло) и как може да бъде модифицирана в случай, че средата се промени динамично.

Б/ Геометрични подходи

Тези подходи използват геометрични методи за извличане на решения на проблема с планирането на пътя и навигацията. Идеята е да се моделира околната среда и робота с помощта на геометрични структури (например многоъгълници, правоъгълници, кръгове) и след това да се дефинират необходимите ротации и транслации на тези обекти, които ще доведат до желания резултат. Известен проблем в тази категория е

проблемът „преместване на пиано“, който се занимава с „маневрите“, необходими за преместване на дълго пиано от стая през тясна врата.
Геометричните методи рядко се прилагат към реални роботизирани проблеми, поради изчислителните разходи за моделиране и манипулиране на геометрични обекти. Освен това те не могат да бъдат лесно адаптирани към динамично променящи се светове.

В/ Клетъчно разлагане (дискретизация) и графи

Идеята тук е да се разложи свободното пространство на голям брой малки области, наречени клетки. Разлагането е такова, че съседните клетки дефинират конфигурации, при които роботът може тривиално (или поне лесно) да се придвижва из между тях. Резултатът е граф от свързани съседни клетки или възли. Тези алгоритми търсят решение чрез преминаване през възлите на графиката, които съответстват на промяна на състоянието. Размерът на клетките може да бъде променлив, както и тяхното разпределение. Точното разлагане на клетките се отнася до случая, когато обединението на всички клетки е точното свободно пространство, за разлика от приблизителното разлагане на клетките, където обединението на клетките е правилно включено в свободното пространство. В последния случай клетките обикновено имат стандартна предварително дефинирана форма, но може би променлив размер.
Конструирането на пътя до целта тук може да се реализира с всеки алгоритъм за търсене в граф. Детайлите на декомпозицията обаче могат да повлияят значително на пълнотата и изчислителната сложност на алгоритъма, както и на качеството на получения път. По този начин дискретизацията на работното пространство се използва за формализиране на проблема за намиране на пътя и преходът от една дискретна клетка към друга е еквивалентен на прехода между два възела на графиката, т.е. промяна на състоянието.
Има голям брой алгоритми, които използват различни стратегии за придвижване през възлите на графика в процеса на намиране на решение.

- Такъв е „първо обхождане в дълбочина„ или *„дълбоко първо търсене“ (DFS).* В този алгоритъм търсенето се извършва чрез последователно придвижване до най-близкия не посетен възел "дълбоко" в номерираната графика. Ако бъде намерен връх с всички посетени съседни възли, тогава алгоритъмът се рестартира от предишния възел и така нататък, докато целта бъде намерена (или не останат неотворени възли). В класическата си форма алгоритъмът се използва главно за търсене в графи с дървовидна структура.
- При работа с графи, които описват *дискретно работно пространство* (т.е. "мрежова" структура), в алгоритъмът трябва да се въведе отделен параметър за ограничаване на "дълбочината" на търсенето. Тази модификация ще намали броя на итерациите и ще премахне зацикляне.
- Алтернативна стратегия за конструиране на най-краткия път в графика е *търсенето в ширина (или т.н. вълнов алгоритъм* [AARTON2020]*)* . Този тип търсене предполага, че за всеки нов възел (започвайки от началния) първо ще бъдат прегледани всички негови най-близки съседи, след това ще бъдат прегледани съседите на отворените възли и така нататък, докато бъде намерен целевият възел. Но и този алгоритъм има недостатък, че процесът на търсене върви във всички посоки, без да се взема предвид местоположението на целта, което се отразява негативно на времето за решаване на проблема.
- Проблемът с "насочеността" на търсенето може да бъде решен с помощта на алгоритмите Дейкстра и A*. *Те се счита за един от най-добрите за решаване на проблема с намирането на пътища, които са близки до оптималните [37].* Формализацията на дискретното работно пространство и принципът на конструиране

на път в A* (протича подобно на алгоритъма на Дейкстра): формира се граф, чиито възли (n) са свързани с ръбове чрез "тегла", съответстващи на дължината на прехода. Така, в плоско работно пространство с ортогонална дискретизация, всеки възел има 8 най-близки възела, с които е свързан с ръбове в 8 възможни посоки: 4 за директни преходи с „цена“ на теглото например 10 и още 4 за диагонални преходи с „цена“ 14, съответно. За всеки нов възел и неговите съседи се изчислява "цената" на прехода, която е най-малката сума от всички преходи, започвайки от началния. На всяка стъпка алгоритъмът търси необработени възли и изчислява разстоянието до всички свои съседи, докато ако пътят до един от съседните възли вече е изчислен в ранните етапи, тази стойност може да бъде актуализирана, ако получената „цена“ на пътят е по-малък на настоящия етап. Транзакциите продължават, докато се намери минималната дължина на пътя до целевия възел.

Алгоритми за планиране от фамилията на търсене в граф като алгоритъма „A*“ [36][37][38], си остават и най-популярните поради сравнително лесната им имплементация.

Г/ Реактивни подходи

Основната идея е да се съчетае възприятието със задействането, така че конкретни усетени модели директно да активират подходящи двигателни команди. Такива връзки обикновено се наричат поведения и могат да бъдат реализирани като проста структура (напр. двигателни схеми). Въпреки че подобен подход изглежда доста ограничен, реалната сила на реактивните системи се проявява, когато няколко поведения се използват едновременно и цялостното поведение на роботизираната система се появява в резултат на някаква комбинация от тях.

Други начини за внедряване на реактивни контролери е чрез използване на техники на размита логика и невронни мрежи (с интелигентни подходи). Бърз механизъм за размити изводи или добре обучена мрежа могат да дадат на робота бърза реакция на внезапни промени в околната среда, избягване на сблъсъци и дори плавен контрол на движението особено в сценарии с много мобилни роботи, ограничено пространство и различни формирования като препятствия. *Дисертация представя точно такава възможност на хибридна архитектура с невронна мрежа за преодоляване на този проблем. За съжаление обхвата на дисертацията не позволява да навлезе в детайли на хибридния режим на управление на група от мобилни роботи.*

Д/ Потенциални полета

В този случай навигацията се дефинира като процес на следване на максималния градиент на определено количество в околната среда. Произходът на подобни методи трябва да се припише на заобикалящата ни природа, където животните и дори хората използват такива методи за навигация. Например, планера достига източник на храна, като тества водата и се движи винаги към посоката, в която се увеличава химическата стимулация. По подобен начин хората се движат към посоката, където звукът става по-силен, за да локализира източника си, или към посоката, където миризмата става по-силна, за да достигне източника на обонятелния стимул.

Говорейки за роботи, едно просто приложение на тази идея би било да се използва някакъв източник на стимул за идентифициране на целевата позиция и сензори на робота, които могат да измерват градиента на стимула, така че роботът да се насочва към целта, като следва максималното градиент на стимулацията. Въпреки това, тъй като не се добавят допълнителни „смущения“ в околната среда, по-разумният (макар и по-труден) начин е да се изгради негов модел, да симулираме дифузията на стимула в

модела, да определим получения максимален градиент и насочвайте робота съответно в реалната среда.
Потенциалните полеви подходи като цяло решават по-широк проблем в сравнение с други методи. Процесът на дифузия, който споменахме, протича във всички посоки в околната среда и в резултат на това градиентът може да се изчисли във всяка позиция. Очевидно може да се конструира път, започвайки от всяка начална позиция. Това е един вид навигационна карта (или навигационна функция), която е структура, която „описва връзката между среда и конкретно целево местоположение в тази среда“. Навигационната карта предоставя цялата необходима информация за достигане до целта от всяка първоначална позиция.
Потенциалните полета е може би най-популярният подход за навигация на роботи. В известен смисъл тя включва виртуалните силови полета, описани по-горе, в които силите определят посоките на градиента.
Слабостите на метода на изкуственото потенциално поле са, че при определени случаи по време движението на робота дадена сила може да получи стойност нула [72]. Това може да накара робота да спре да се движи или да обикаля около една точка - локален минимум на потенциалната функция [72], като така не би могъл да достигне целта си. Освен това, пътищата формирани чрез потенциално поле на робота може да се сближат на твърде опасно разстояние, когато има твърде много препятствия. [72].
Въпреки слабостите им от алгоритмите на класическия подход най-адаптивни и широко внедрявани се оказват тези, които използват изкуствени потенциални полета и затова ще се спрем в детайли по долу.

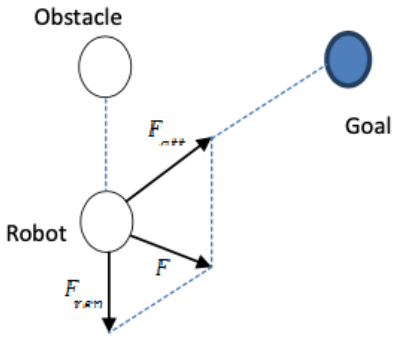
1.3.1. Метод на изкуствени потенциални полета.

Методите в тази категория дефинират виртуални сили, които действат върху робота и определят неговото движение. Като цяло се приема, че позицията на целта прилага сила на привличане върху робота, докато препятствията прилагат отблъскващи сили. Движението е резултат от комбинацията от всички сили, която обикновено е сумиране на векторите на силите. Резултатът е, че роботът ще се движи към целта поради привличането на целта, като в същото време избягва препятствията, които „отблъскват“ робота от тях. Недостатъкът на тези методи е, че роботът може лесно да бъде хванат в „локални минимуми“, които са ситуации, при които привлекателният и отблъскващ сили са отменени [115]. ***Този подход може успешно да се комбинира с предложения в дисертацията алгоритъм на основата на модифицирана НМХ.***
Методът на *изкуствените потенциални полета* дава възможност за създаване на разпределени системи за групово управление на роботи [54], [55] и затова си заслужва да се разгледат по обстойно. В този случай всички елементи на околната среда: роботи, препятствия, граници и т.н. има присвоени виртуални полета, чийто физически аналог са потенциални полета. Тези обекти, които трябва да се движат един към друг или близо един до друг, са надарени със "заряди" от противоположни знаци, а тези, които не трябва да се приближават, за да избегнат сблъсък, са надарени със "заряди" от същия знак. Понятието "заряд" тук има общоприето значение, отразяващо свойството на електрическите заряди с противоположни знаци да се привличат един към друг и свойството на електрическите заряди от един и същи знак да се отблъскват. Движението на роботите се определя от влиянието на произтичащите естествено образувани „сили“ от системите за управление, в зависимост от видовете „заряди“ и разстоянията между посочените по-горе елементи на средата. Получената сила се определя от всеки робот чрез сумиране на векторите на всички действащи върху него сили на привличане и отблъскване. Привличащи и отблъскващи потенциални полета често се реализират с

помощта на функциите на Ляпунов. На тяхна основа се изгражда законът за груповото управление и се доказва неговата валидност [56].
Прилагането на метода на потенциалните полета дава възможност за реализиране на различни видове групов контрол [55], [56] и придвижване във формация МР:

- „надпревара за лидер“ – водещият робот е привлечен от целта, а останалите роботи от групата са привлечени от лидера;
- "верига" - водещият робот се привлича към целта, а всеки следващ е привлечен от предишния;
- "дивергенция" - всички роботи в групата са подложени на отблъскваща сила от страна на лидера, което води до разпръскване на групата;
- "конвергенция" - всички роботи от групата се влияят от силата на привличане към водещия робот. Този подход се използва по късно в дисертация в Глава 3, където ще бъде обследван в детайли;
- "безплатно търсене" - използва се за наблюдение на затворени помещения.

Роботите се влияят само от силите на отблъскване от препятствия и от други роботи в групата в случай на прекомерна близост.



Фиг. 1-4 Резултатна сила на потенциалната функция.

Роботите избират техните траектории в съответствие с получената цел. Отблъскването от препятствия и/или други роботи предотвратява възможни сблъсъци.
Въвеждането ИПП (Фигура 1-4) при движението на МР може да бъде наложително в някои случаи, с цел оптимизиране ма калкулациите при управление на движението на много голям брой агенти в динамично в сложно работното пространство - много препятствия (примерно на локално ниво). ИПП *има добавената стойност при подобряване фалшивите положителни резултати, ускоряване на сходимостта на процеса на създаване на навигационния път и по този начин занижаване на изискванията към изчислителна достатъчност необходима при ситуации с много препятствия и МР-ти*.
Изкуственото потенциално поле като подход за намиране на път е предложено за първи път от Хатиб [98] и е широко изследвано поради своята простота и интересен математически анализ [99]. Основната идея на този подход е да се запълни работното пространство на робота с привличащо потенциално поле от целта, и отблъскващо потенциално поле, причинено от препятствие. Силата на потенциалното поле във всяка дадена точка в работното поле е функция както на относителното разстояние между робота и обекта, така и на конкретния профил на обекта. Метода на изкуствени потенциални полета (ИПП) (на английски „artificial potential fields“ - AFP) [31][32] е възможно да се комбинира с редица др. подходи за достигане до целта е често предпочитан в комплексни задачи за планиране на пътя. При подхода на изкуствен потенциал, препятствието в работното пространство създава отблъскваща сила, която

избутва робота далеч от препятствията, целевата точка създава атрактивна сила, която привлича робота към целевата точка. Отблъскващата сила и атрактивната сила са описани като функция на отблъскващ потенциал и атрактивна потенциална функция. По принцип тези два вида потенциални функции се използват заедно в практически приложения, за да се постигне сходимост, без сблъсък и без препятствия [51-52].
Векторното поле в ИПП е разделено на два компонента: *целта на движението е представена от атрактивно векторно поле, докато препятствията са представени от отблъскващо векторно поле.* Добавянето на две векторни полета дава възможност за решаване на проблемите с придвижването до дадена целева точка с избягването на препятствия.
МР-ти притежават актуална карта на околната среда и са оборудвани всеки по отделно с навигационна система, която общо за сформираната група позволява образуването на 360 сензорна система в кръг. Всеки един МР има ясна представа за точните му координатите. На ниво отделни роботи – има модул за изчисляване на потенциално поле *U(q),* с алгоритмите за управление за изчисляване на виртуални сили, които осигуряват привличането на робота към целта (в текущото изследване – лидера на формацията) или отблъскването му от предварително известни препятствия (в посочения случай това са останалите роботи във формираната група). Когато по пътя на движението на робота се появят нови препятствия (примерно извън очакваните МР в групата), МР мина в режим „инцидентно избягване на препятствие), при който полето на виртуалните сили незабавно (в реално време) се актуализира заедно с „невронната карта“ активирана от КП. Целта е да се образува обща навигационен път, за да се избегне инцидент – сблъсък с новото препятствие или с др. робот в групата.
Ограничава се разглеждането на равнинната задача $C \subseteq R2$, а МР като материална точка и следователно МР.
Потенциалното поле U(q) се формира от сумата на две полета - *привлекателно*, което се създава от целта (в това проучване лидера, и *отблъскващо*, създадено от препятствието (останалите МР в групата) и се представя чрез формулата:

$$U(q) = U_{attr}(q) + U_{rep}(q),$$

Потенциалната сила в позиция q = (x,y) се намира като антиградиент на функцията U(q):

$$F(q) = -\nabla U(q),$$

В съответствие с горното, виртуалната сила, създадена от полето, също може да бъде разложена на два компонента - привлекателна и отблъскваща сила:

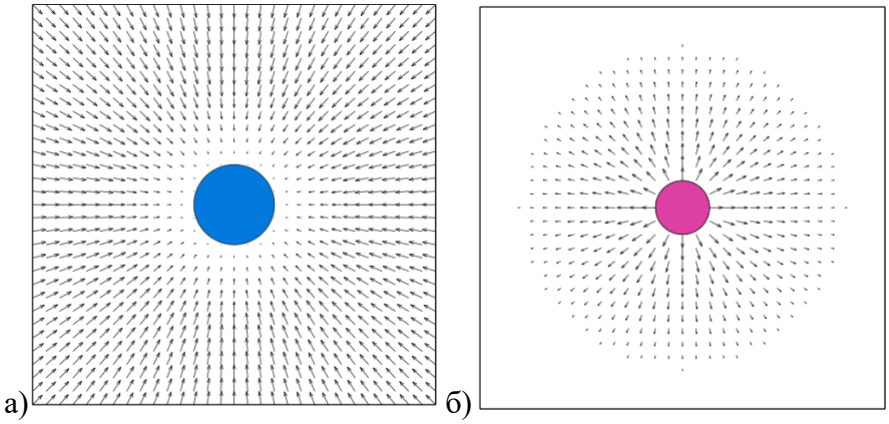
$$F(q) = F_{attr}(q) + F_{rep}(q),$$

където

$$F_{att}(q) = -\nabla U_{att}(q), \qquad F_{rep}(q) = -\nabla U_{rep}(q).$$

Според тип потенциалната функция може да разграничим различни виртуални силови полета, обикновено обвързани с текущия режима, работа и поведение на МР. Виртуалните силови полета обикновено имат вида на параболична или частично линейна.
Основният градивен елемент на потенциалните полета е векторът на действие, който съответства на скоростта и ориентацията на движещ се робот. Всеки тип поведение извежда и желания изходен вектор. Например, при поведение на „следвай целта“, на което е възложена задачата да накара представения робот да се насочи към

идентифицирана цел. Резултатът от поведението в този режим е вектор, който насочва робота към целта. Ако предприемам, че робота преминава през всяка точка в двуизмерно пространство и запишем изходния вектор във всяка точка, колекцията от тези вектори ще изглежда нещо като диаграмата, илюстрирана на фигура 1-5 (а).:



Фиг. 1-5. (а) атрактивно/привлекателно потенциално поле (графиката в ляво); (б) отблъскващо потенциално поле (графиката в дясно)

Тази колекция от вектори образува потенциално поле и представлява синтетични енергийни потенциали, които роботът ще следва. Потенциалното поле, свързано с това поведение е пример за атрактивен потенциал, тъй като полето кара робота да бъде привлечен към целта (т.е. всички вектори сочат към целта).

Един от подходите за представяне на потенциално поле е да се представи като преобразуване от един вектор в друг. При двумерна навигация, това е преобразуването от вектора $v = [x, y]^T$ в градиентния вектор $\Delta = [\Delta x, \Delta y]^T$ където "T" представлява индекс „транспониране".

За да генерираме атрактивното полета (фигура 1-5а), дефинираме Δx и Δy по отношение на „v", както следва:

- нека с (x_G, y_G) се обозначи позицията на целта и нека „r" означава радиуса на целта. Нека $v = [x, y]^T$ означава позицията (x, y) на агента.
- намираме евклидовото разстоянието между целта и агента:

$$d = \sqrt{(x_G - x)^2 + (y - y_G)^2}, \tag{4.7}$$

- намираме ъгъла между агента и целта (ъгъл към квадранта):

$$\theta = tan^{-1}\left(\frac{y - y_G}{x_G - x}\right), \tag{4.8}$$

- определят се стойностите за Δx и Δy според следните ако-тогава (if-then) правила:

ако $d < r$, *тогава:* $\Delta x = \Delta y = 0$
ако $r \leq d \leq s+r$, *тогава:* $\Delta x = \alpha.(d - r)\cos(\theta)$ *и* $\Delta y = \alpha(d - r)\sin(\theta)$
ако $d > s+r$, *тогава:* $\Delta x = \alpha.s.\cos(\theta)$ *и* $\Delta y = \alpha.s.\sin(\theta)$;

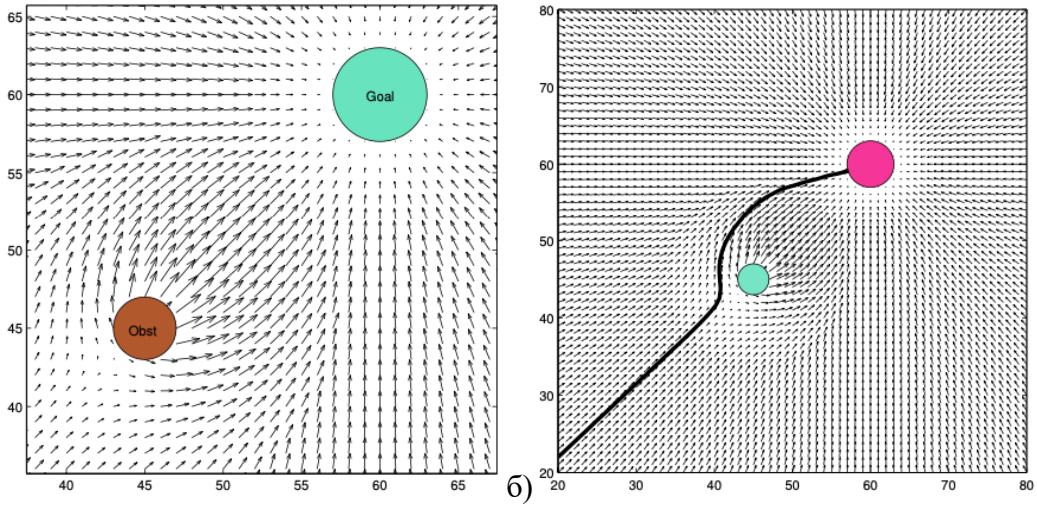
Така „целта" – (в лидера на групата) се представя като окръжност с радиус „r". Когато агентът достигне целта, върху него спират да действат сили на привличане от целта,

защото d < r и Δx, и Δy са нула. Полето има „покритие“ или зона на влияние „s“ и агентът попада под влияние на това поле ако d = s + r. Извън този кръг на обхвата, величината на вектора се задава на максималната възможна стойност. В рамките на този кръг на обхват, но извън радиуса на целта, векторната величина се задава пропорционална на разстоянието между агента и целта. Константата „α“ e > 0, така че силата на полето да може лесно да се мащабира с този коефициент.
И обратно, в ИПП има и поле, което отблъсква (създава се от препятствие) илюстрирано на фигура (фигура 1-5б), и което отговаря на примерно поведение „избягване на сблъсък“.

- нека с (x_o, y_o) се обозначи позицията на препятствието. Нека „“r означава радиуса на препятствието и нека $v = [x, y]^T$ означава позицията (x, y) на агента;
- отново намираме разстоянието „d“ между препятствието и агента като в 3.7 и ъгъла “θ”по 4.8;
- приемаме стойностите за Δx и Δy като следва:

ако d < r, тогава: $\Delta x = -sign(\cos(\theta))\infty$ *и* $\Delta y = -sign(\sin(\theta))\infty$
ако $r \leq d \leq s+r$, *тогава:* $\Delta x = -\beta(s + r - d)\cos(\theta)$ *и* $\Delta y = -\beta(s + r - d).\sin(\theta)$
ако d > s+r, тогава: $\Delta x = \Delta y = 0;$

В рамките на отблъскващото потенциалното поле, силата на отблъскване е безкрайна и *сочи към центъра на препятствието*. Извън кръга на влияние (s + r) отблъскващото потенциално поле е нула. В рамките на кръга на влияние, но извън радиуса на препятствието, величината на вектора нараства от нула (когато $\beta(s + r - d) = 0$, като при $d = s + r$ - агента е на ръба на кръга на влияние) до β(s), като $\beta(s + r - d) = \beta(s)$, като при $d = r$ – агента е на ръба на препятствието). Константата $\beta > 0$ се ползва, за да позволи на агента да мащабира силата на отблъскващото поле. Забележете, че векторът сочи далеч от препятствието - това става чрез въвеждане на отрицателния знак в дефинициите на Δx и Δy.
Множество потенциални полета се комбинират (сумират) за да се получи реалното им влияние и съответно движение на робота в пространството като е важно е да отбележим, че полетата никога няма да има нужда да се изчисляват за цялото работно пространство. В реални условия МР ще „вижда“ само тази част от това полето, което пряко среща. На фигура 1-5 графически са представени силите, които роботът ще изпита във всяка точка от двумерното пространство.
С две поведения - режима „заобикаляне на препятствие“ и „достигане на цел“, новото изчислено потенциално поле е показано на фигура 1-6:



Фиг. 1-6. (а) потенциалното поле, генерирано от робот с две поведения, когато в работното пространство присъстват цел и препятствие (графиката в ляво);
(б) траектория калкулирана от софтуера за робот с две поведения и когато има цел и препятствие. (графиката в дясно)

Потенциалното поле (фигура 1-6а) се изчислява като първо се намира вектора генериран веднъж от атрактивната сила за "x" и "y" - Δ_{Gx} и Δ_{Gy}, и втори път от отблъскващата сила генерирана от препятствието - Δ_{Ox} и Δ_{Oy}. Накрая обобщаване чрез обединяване на векторите в едно:

$$\Delta x = \Delta_{Ox} + \Delta_{Gx}, \quad (4.9)$$
$$\Delta y = \Delta_{Oy} + \Delta_{Gy}. \quad (4.10)$$

Поставя се робота близо до началото на работното пространство, показан на фигура 1-6б. За да калкулираме траекторията МР (агента) определя се Δx с помощта на уравнение (4.9) на потенциалното поле, генерирано от двете му поведения, а Δy с помощта на уравнение (34.10). Скоростта на МР тогава ще е пропорционална на големината на двата вектора изчислени както следва:

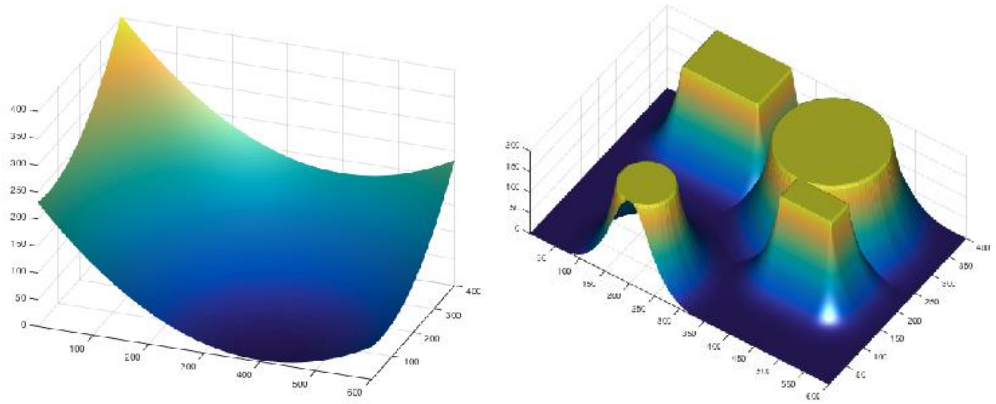
$$v = \sqrt{(\Delta x)^2 + (\Delta y)^2},$$

а ъгъла на посоката:

$$\theta = tan^{-1}\left(\frac{\Delta y}{\Delta x}\right),$$

Така докато се движи през работното пространство, агента прави наблюдения на околната среда (полети), идентифицира нови вектори на действие и избира нови посоки и скорости. Получената траектория на робота е визуализирана на фигура 1-6б.
Планировчика (КП) в процеса се стреми да изгради такава плавна функция върху обхвата на работното пространство така, че да има високи стойности, когато роботът е близо до препятствие и по-ниски стойности, когато е по-далеч. В последствие може да се използва *градиента такава функция*, за да се насочи робота към желаната конфигурация – центъра на определената формация екип.
А/ *Атрактивна потенциална функция* $f_{attr}(x)$ се представя като разстояние между текущата позиция на робота $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ и местоположение на целта $x_g = \begin{pmatrix} x^g{}_1 \\ x^g{}_2 \end{pmatrix}$ както следва: $f_{attr}(x) = \varepsilon\,(||x - x_g||^2)$,
където ε е константен параметър за мащабиране на силата. Функцията е визуализирана

във фигура 1-7а



*Фиг. 1-7. (а - ляво) Визуализация на функцията на **атрактивното**/притегателно потенциално поле* $f_{attr}(x) = \varepsilon(||x - x_g||^2)$*; (б -дясно) функцията на **отблъскващо** потенциално поле*

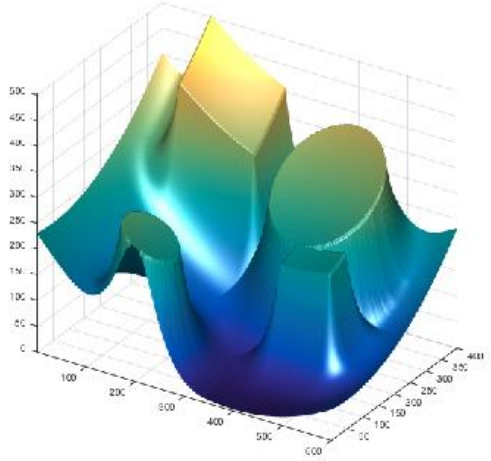
Б/ Конструиране на *отблъскващо потенциално поле (фиг. 1-7б)*
В равнината функция на отблъскващия потенциал $f_{rep}(x)$ може да бъде конструирана въз основа на функция $p(x)$, която връща разстоянието до най-близкото препятствие от дадена точка в конфигурационното пространство "x"., като се определя стойностите й според следните „ако – тогава" (***if - then***) правила:

ако $p(x) \leq d_0$, *то тогава:* $\boldsymbol{f_{rep}(x) = \eta \left(\frac{1}{\rho_x} - \frac{1}{d_0}\right)^2}$,

ако $p(x) > d_0$, *то тогава:* $\boldsymbol{f_{rep}(x) = 0;}$

Където η е константен мащабиращ параметър а d_0 е параметър, който контролира влиянието на потенциала на отблъскване. Тогава комбинираното потенциално поле (фигура 1-7-2):

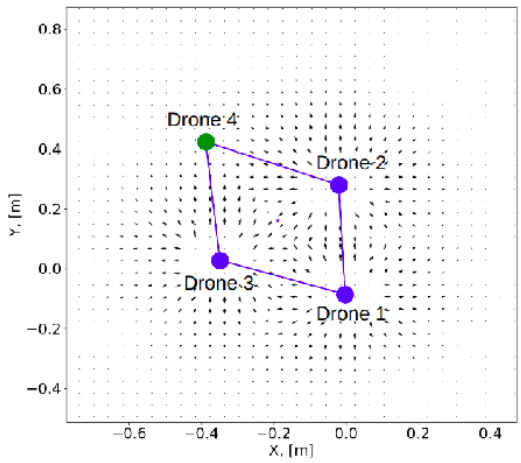
$$\boldsymbol{f(x) = f_{attr}(x) + f_{rep}(x),}$$



Фигура 1-7-2. Визуализация на комбинираното изкуствено потенциално поле

В процеса на създаване на процеса на навигация съществуват още два типа ИПП:
А/ тангенциалното потенциално поле. Това поле се получава чрез намиране на величината и посоката по същия начин, както при отблъскващото поле. Въпреки това, θ се модифицира преди Δx и Δy да бъдат дефинирани чрез задаване на $\theta \leftarrow \theta \pm 90^\circ$, което

кара вектора да се измести от сочещ далеч от центъра на препятствието към *сочещ в посоката на допирателната към окръжността*. Знакът на смяната контролира дали тангенциалното поле причинява завъртане по посока на часовниковата стрелка. При нас този тип поле ще се използва при *непланирано и извънредно избягване* на препятствия нахлуващи в локалното пространство на МР, които се придвижват в екип.
Б/ случайно потенциално поле. Това поле спомага на агента да отскача и да избягва засядането в локални минимуми, които могат да възникнат, когато няколко полета се добавят заедно в близко разстояние. Получава се чрез произволен избор на „d“ при равномерно разпределение (т.е. избиране на произволна точка с еднаква вероятност) през интервала *[0 до γ]* и „θ“ от равномерно разпределение през интервала *[0 до 2 ∗ π]*.
Този метод може също да се използва за подобряване за навигация на рояка от МР, като се определят силите на взаимодействие (които са пропорционални на скоростите в разглеждания пример) между всички агенти. Алгоритъмът може да проследява статични, както и динамични препятствия. Локален плановик за движение, базиран на изкуствени потенциални полета, позволява корекция на позициите на агентите на формацията, предотвратявайки сблъсъци. Разстоянието до лидера „d“ ще се дефинира спрямо позицията му при предварително определена геометрична форма на образуване. Всеки робот и препятствие на познатата карта притежава собствен локален потенциал, който допринася за сумарното глобално поле. Тези изкуствени потенциали определят силите на взаимодействие между съседни роботи и препятствия. Фигура 4.8 представя тези сили във формирането на четири МР, изобразени като свързани кръгове. Този градиентен график се визуализира за робота, изобразен със зеления кръг. Той разпознава други МР в рояка като препятствия, докато желаната му позиция (един от върховете на ромбовидната формация) е привлекателна точка за себе си – фигура 1.8



Фиг. 1-8. Стратегия за избягване на сблъсък между агенти, базирана на метода на потенциалното поле. Графиката представя как зеленият робот разпознава другите три МР (изобразени в син цвят) в градиентната карта.

По този начин роякът МР е в състояние да адаптира формата си според локалните препятствия и да отреагира бързо. Поради интерактивния характер на метода за избягване на потенциално полеви препятствия, е възможно да се предотвратят сблъсъци между МР и други движещи се обекти в реално време. Локален плановик за движение, базиран на изкуствени потенциални полета (ИПП), позволява коригиране на позициите на формираните групи агенти, предотвратявайки сблъсъци показан на фигура 1-8. В допълнение, този алгоритъм може да се приложи към всякакъв вид формиране на роботи, с реални или виртуални лидери.

Стратегия за определяне посоката на МР се основава на най-лесния подход да се стигне до глобалния минимум на потенциала - градиентно спускане докато позицията на робота не е достатъчно близо до целта с функция от вида:

$$\propto -\nabla U(x,y),$$

Тогава ъгъла към глобалния минимума в радиани ще бъде изчислен със следната формула:

$$\theta = tan^{-1}\left(\frac{\Delta y}{\Delta x}\right),$$

А избора на подходяща скорост на робота $|v|$ ще се определи в зависимост от режима на навигация при робота.

Предимството на метод ИПП остава неговата простотата на изчисляване на резултантните сили, които лесно се прилагат дори на бордови компютри с ниска мощност, типични за малки роботи. Като правило методът на потенциалните полета позволява да се реализира формирането на линии с равно отдалечени роботи и не позволява получаването на по-сложни пространствени конструкции на роботи. В допълнение, този метод осигурява ефективно формиране на траектории на роботи само в случаите, когато контурите на елементите на околната среда са описани с изпъкнали многоъгълници или кръгове. В противен случай могат да възникнат т. нар. „локални минимуми“, за чийто изход са необходими допълнителни евристични правила [57].

1.4. Методи за планиране на път, базирани на интелигентни алгоритми

Напоследък започнаха да се появяват алгоритми, които решават проблема с планирането на пътя на базата на интелигентни модели и подходи от ИИ.

Най-обещаващите алгоритми, въз основа на които е възможно да се изградят интелигентни системи за управление, могат да бъдат разделени в 3 основни направления [44, 45]:

- еволюционни (генетични) алгоритми;
- алгоритми с размита логика;
- алгоритми на невронни мрежи.

В хода на анализа беше установено, че по-старите и някои вече известни съществуващи подходи не отчитат спецификата на системите при МР (изчислителни възможности, скорост на реакция на външни смущения поради по-бавни и сложни изчисления), което налага редица ограничения върху приложението от тези методи при наличие на динамични и бързо променящи се препятствия в работната зона и опасност от сблъсък с други агенти от групата. Следователно разработването на система за планиране, отчитаща спецификата на груповото управление и хардуерните възможности на МР, наложи нов прочит по темата. Това са интелигентни алгоритми и имат редица ключови предимства пред класическите подходи, а именно:

- опростяване на формализирането на задачата за планиране като се използва един и същ метод за навигация и координация;
- ниско потребление на хардуерни ресурси;
- висока производителност и скорост.

Тези методи напълно включват забележките, направени по-горе във връзка с класическите методи за планиране на пътя. Допълнително ограничение при

използването им в автономни мобилни системи е необходимостта от големи изчислителни ресурси.

1.4.1. Невронна мрежа на Хопфийлд (НМХ)

Изкуствените невронни мрежи са системи за обработка на информация, която е вдъхновена от биологични нервна система. Изградени от множество обработващи елементи (невронни), всеки от които се изпълняват прости числови операции и споделят резултатите с техните съседи чрез претеглени връзки. Напоследък все по-често се срещат и решения за УДМР на база на „невронни карти“ Хопфийлд [46, 64]. Разработен през 1982 г. от физика Джон Хопфилд, представя модел като еднослоен, цикличен автоасоциатор. *Този модел показва, че физически системи съставени от голям брой взаимодействащи си частици проявяват при определени условия спонтанни изчислителни свойства.* Авторът използва теорията на спиновите стъкла за описание на цикличен модел на НМ съставена от бинарни неврони. Те са познати още като „повтарящи се невронни мрежи“ (на англ. RNN). В процеса на сходимост мрежата се доближава до едно от възможните равновесни позиции, които са локалните минимуми на функцията, наречена „*енергия на мрежата“*.

Така впоследствие беше предложен модел на изкуствен НМ с обратна връзка - повтаряща се НМ, която позволи да се разшири обхватът на задачите, които трябва да бъдат решени. В общия случай, повтаряща се невронна мрежа (RNN) може да се разглежда като невронна мрежа, състояща се от неврони с множество обратни връзки, чрез които както собствения изходен сигнал на неврона, така и сигналите на всички други неврони в мрежата могат да бъдат подавани към входа. Предаденото от обратна връзка възбуждане се връща към този неврон и той изпълнява отново своята функция, като по този начин невродинамиката в този НМ модел става итеративна. Отличителна черта на първите модели НМ е, че всички сигнали в мрежата се разпространяват само в една посока: от вход към изход и всеки неврон в слоя е свързан с всички неврони на следващия слой (наслоен напълно свързан НМ). Според естеството на разпространение на сигнала, НМ данните се класифицират като мрежи за директно разпространение [62]. НМ с пренасочване се използват успешно при проблеми с разпознаване, класификация и прогнозиране, но има сериозни ограничения при моделиране на динамични процеси с помощта на този тип мрежи. Основният проблем е, че изходните стойности на невроните зависят само от входния вектор, т.е. мрежовият изход ще остане непроменен, когато входният сигнал е постоянен. В този случай моделирането на динамични процеси е възможно само чрез прогнозиране на малки промени в изследваните параметри на всяка итерация [46].

Наблюденията на разпространението на електрически сигнал в биологични невронни мрежови структури показват наличието на множество обратни връзки.

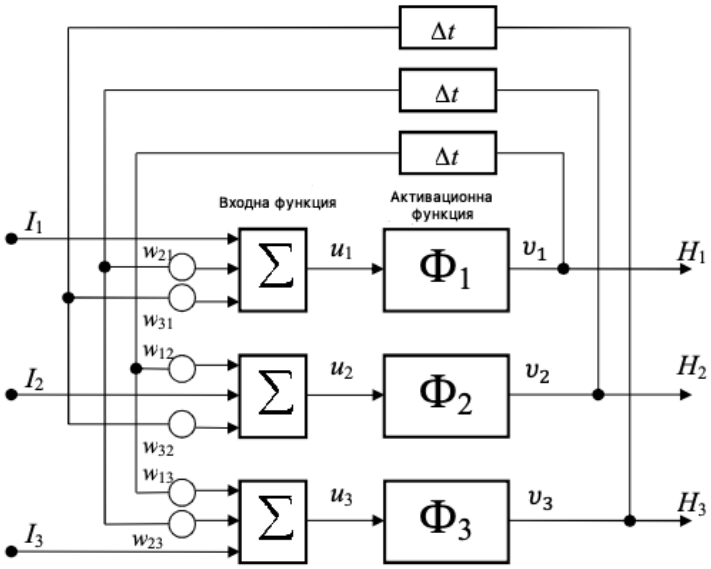
Структурата на дискретната невронна мрежа на Хопфийлд (Фигура 1-9) представлява един слой неврони, чиито входни сигнали са претеглена сума от външни сигнали (I) и сигнали за обратна връзка на други неврони в мрежата. Входните сигнали на невроните се подлагат на нелинейна обработка с помощта на дадена функция (*активационна функция “Ф”*) и след това (с определено време закъснение Δt) отново постъпват на входовете на невроните, в резултат на което изходният сигнал (H) се формира само след като стойностите на изходите на всички неврони останат непроменени (състояние на "баланс"). По този начин този тип НМ моделира определен стохастичен процес, чието крайно състояние се определя от входния вектор (Y), чиито елементи са съответните сигнали на мрежовите неврони [62, 63].

Нека сега се представи напълно свързана НМХ невронна мрежа, където състоянието на

всеки i-ти неврон се определя от неговия очакван изходен сигнал, а енергийната функция за активиране Φ_i може да приеме стойностите $\{0, 1\}$. Както можете да видите от Фигура 1-9, изходът на всеки H_i неврон е претеглената сума от изходите на другите неврони, v_j, плюс стойността на входния сигнал I_i:

$$\mathrm{H_i} = \sum_{j=1}^{j} w_{ij} v_j + I_i, \qquad (1.1)$$

където w_{ij}очаквано е *теглото* на синаптичната връзка, свързваща j-тия неврон с



Фиг. 1-9 Повтаряща (рекурентна) невронна мрежова архитектура на Хопфийлд с три неврона

i-тия неврон. За този тип повтаряща се невронна мрежа на Хопфийлд [73], стойностите на w_{ij}са фиксирани за всички i и j, симетрично, т.е. $w_{ij} = w_{ji}$ [62, 63]. Или двете връзки между двойка възли имат еднакво тегло.

След подаване на входния сигнал започва процесът на активиране на невроните мрежа, през която състоянието на НМ във всеки момент от време се описва от вектора Y(t) = (v_1(t), v_2(t), ….v_n(t)), този случай състоянието на всеки неврон в момент (t + 1) се променя в зависимост от даденото прагово ниво на активиране S_i, както следва [46]:

$$v_i = \begin{cases} v_i(t+1) = 0, H_i(t) < S_i, \\ v_i(t+1) = 1, H_i(t) > S_i, \\ v_i(t+1) = v_i, H_i(t) = S_i. \end{cases} \qquad (1.2)$$

По този начин системата от уравнения (1.1) и (1.2) описва определен стохастичен процес на промяна състоянието на системата Y(t) в зависимост от външните стойности на I_i. Този процес извежда системата до статично състояние на равновесие, т.е. до режим, при който условието за сближаване за всички неврони е изпълнено [64, 65]:

$$\mathrm{Y(t)} = \mathrm{Y(t+1)} \qquad (1.3)$$

Подобната възможност на повтарящи се връзки в невронната мрежа прави използването й като „асоциативна памет“, тъй като дори частично активиране на мрежата (т.е. само част от невроните получават референтния входен сигнал) близо до едно от стабилните състояния води до същото общо стабилно състояние [46]. Също така си струва да се отбележи, че стабилността на мрежата е гарантирана, ако матрицата с всички тегла W е симетрична и всички диагонални елементи са равни на нула. Доказателството за стабилността може да се получи от анализа на „енергийната“ функция на разглежданата невронна мрежа [63, 66]:

$$E = -\frac{1}{2}\sum_i^I \sum_j^J w_{ij} v_i v_j - \sum_{i=1}^I I_i v_i + \sum_{i=1}^I S_i v_i, \qquad (1.4)$$

"E" е стойността на функцията за състояние E(v) на мрежата на Хопфийлд. Тази стойност може да се увеличи или да запази стойността си в процеса на невродинамиката. Така енергийната функция "E" също е функция на Ляпунов за невросистемата [61]. Въз основа на тези свойства на енергийната функция на разглежданата невронна мрежа можем да се заключи, че произволна промяна в състоянието на неврон в архитектурата води до намаляване на енергийната функция на цялата система [61, 63, 67]. Тази особеност на Хопфийлд невронната мрежа определя и ключовото му свойство – способността за самоорганизиране. Съществуването на енергийната функция на Ляпунов за дефинираната по-горе система гарантира, че мрежата в ще достигне в равновесие по време на еволюцията. И това й свойство прави възможно използването на този тип невронна мрежа – „невронна карта" за решаване на редица сложни проблеми, като например обучаващи подсистеми за разпознаване на ситуации и контрол в реално време на движението на МР [64, 67].

НМХ е и основа за модел на биологични асоциации, генерирани карти от мозъка по време на невронна активност. Те се появяват по време на дейността на сетивните функции на мозъка като тактилна карта, слухова карта и др. Като биологични модели, невронните карти могат да се използват като инструмент за решаване на практически проблеми и се базират на модела на невронна мрежа „Хопфийлд" посочена в по горния точка 1.4.1. За първи път използването на този модел за решаване на проблема с намирането на траектория в дадено пространство беше предложено от група учени Рой Глазиус, Анджей Комода, Стан С. А. М. Гилен [72]. Впоследствие методът е усъвършенстван и на негова основа е разработена система за позициониране на един МР [73]. *Същността на метода е, че повтаряща се невронна мрежа се използва като топологично представяне на дискретно работно пространство, т.е. центърът на всяка отделна клетка съответства на неврон, свързан само с най-близките му съседи, а връзките със съседите имат тегло, съответстващо на дължината на прехода между тези клетки.* След това на НМ се подава входният вектор на първоначалното състояние, при което невроните, съответстващи на клетките - препятствия, имат постоянно нулево състояние, целевата клетка има максимална стойност и е равна на единица. След приключване на процеса на активиране, т.е. условието е достигнато, когато всички неврони, с изключение на нулевите, приемат стойност, различна от нула, формира се матрица на състоянието на всички неврони на мрежата (невронна карта) и размерността на матрицата съответства на измерението на дискретния работно пространство. Използвайки алгоритъма за максимален градиент на матрицата на състоянието, е възможно да се формира път, близо до оптималното, от всяка ненулева клетка до целта [73].

Този подход има редица ключови предимства пред разглеждани алгоритми за търсене:

- дискретното работно пространство е възможно чрез въвеждане само на вектора на първоначалното състояние (всъщност броят на нулевите неврони и целевия неврон), което значително опростява формализирането на задачата за намиране на траектория и ви позволява да работите с по-прости структури от данни;
- архитектурата на НМ остава непроменена за всякакви работни пространства, променят се само броят на невроните и наборът от входни параметри;
- алгоритъмът има и висока производителност поради бързата конвергенция на повтарящата се невронна мрежа и възможността за използване на паралелни изчислителни техники, което го прави особено актуално за използването му в динамични работни пространства с неизвестна досега конфигурация;
- възможност за пълна хардуерна реализация на НМ за получаване на още по-

висока производителност и по-малко потребление на ресурси на алгоритъма;
- работейки със стойностите на крайната матрица на състоянията, е възможно да се формират траектории (близки до оптимални), като се вземат предвид претеглените области, както и да се изградят неконфликтни маршрути за движение и гъвкави правила за взаимодействие за движещи се групи MR в общо работно пространство.

1.4.2. Генетични алгоритми

Генетични алгоритми (ГА) включват концепцията за теорията на Дарвин. Те са вдъхновени от биологичната еволюция (развитие), естествен подбор, и генетична рекомбинация [51]. Генетични алгоритми могат да бъдат използвани, за да се развиват прости правила за мрежовия трафик. ГА генерира набор от правила, които по-късно могат да бъдат използвани, за да се разграничат от нормалния и ненормалния мрежови трафик. Алгоритмите за създаване на тези набори от данни, които използва данни със структура тип хромозома- подобни и се развиват хромозомите използвайки оператори селекция, рекомбинация и мутация.

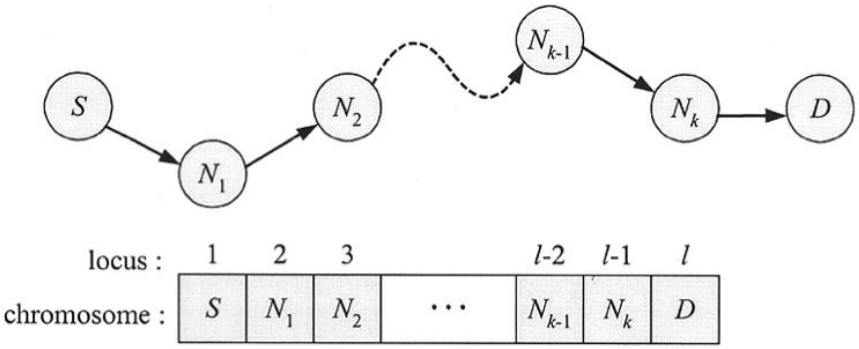
Най-общо казано, ГА е интелигентен алгоритъм за намиране на оптимално решение чрез итеративна произволна извадка (селекция) и последващо комбиниране и вариация на изходните параметри. ГА се отнася до методите на така наречените еволюционни изчисления, които използват формализирани методи на естествена еволюция, като наследяване, кръстосване, мутация, селекция и кросоувър, за намиране на решение на оптимизационен проблем.

През 2002 г. Chang Wook Ahn и R. S. Ramakrishna предложиха един от методите за намиране на най-късите пътища в свързан граф с помощта на ГА [86]. Тази работа е класически пример за прилагане на ГА и за решаване на проблема с намирането на най-краткия път, с единствената разлика, че предлага допълнителни оптимизации и подходи към процедурата за формиране на популацията за повишаване на производителността.

Нека се разгледат накратко основните етапи на алгоритъма. Като изходни данни се представя мрежа с много връзки, която може да бъде представена като насочен граф G = (N, A), където N - означава набор от n възела (върхове), а A - е набор от ръбове. Матрица на цената се обозначава като C = $[C_{ij}]$, където C_{ij} - съответства на цената на преместване от възел i към възел j. Началната точка (източник) е S, а дестинацията е D. Индикаторът на връзката I_{ij} показва дали съществува маршрут между възел i и възел j. Ако маршрутът съществува, тогава I_{ij} = 1, в противен случай I_{ij}= 0 (препятствие) [86].

Процесът преминава през следните стъпки:

Инициализация на популацията. При формиране на популация се използват хромозоми с различна дължина. Наборът от генотипове на първоначалната популация са маршрути от S до D, като максималната дължина на хромозомата не трябва да надвишава N. Фигура 1.10 показва схемата за формиране и кодиране на генотипа на първоначалната популация [86].



Фигура 1.10 Формиране кодирането на генотипа в първоначалната популация [86].

По този начин всеки ген е графичен възел и тяхната последователност (хромозома) съответства на един от възможните маршрути.

Има два възможни подхода за формиране на първоначалната популация:

- евристичен - ефективен е в случай на възможност за формиране на голям брой решения. В този случай се оформя по-точен набор от решения (пъти, близки до оптималните) с по-малко времеви загуби;
- произволен – ефективен при работа в по-малко сложни конфигурации на работното пространство.

Фитнес функция. Фитнес функцията (фитнес функция) се въвежда за оценка на „пригодността“ на индивида в популацията, тоест съответствието на всеки маршрут от набора с критерия „оптималност“ (по минималната дължина на пътя) [86]:

$$f_i = \frac{1}{\sum_{j=1}^{l_i-1} C_{g_i(j),g_i(j+1)}},$$

където l_i е дължината на i-тата хромозома; g_i(j) е генът на j-тата хромозома със сериен номер j в генната последователност; C е цената на връзката (link cost) между j и j + 1 възли (гени) на дадения път (i-та хромозома).

Избор. Изборът на индивиди за по-нататъшно преминаване се извършва по метода на турнирен подбор по двойки въз основа на фитнес функцията.

Пресичане. На етапа на пресичане се формира ново поколение (набор от маршрути) с помощта на комбинация, смесване или мутация, като вероятността за мутация се задава в диапазона от 5 до 10%.

След образуването на ново потомство, най-лошата част от генотиповете се заменя с нови. Повтарянето на основните етапи на алгоритъма се извършва до получаване на пътя с най-ниска обща цена или след преминаване през определения брой поколения (цикли на еволюция) [86].

Използването на този ГА за решаване на проблема за търсене при определени условия дава високо увеличение на производителността в сравнение с алгоритъма на „Dijkstra“: при сравнително тестване на алгоритми върху графики до 50 възела, ГА решава проблема средно 2 пъти по-бързо, с сравнима точност на крайното решение [86]. Има обаче и редица недостатъци, които трябва да се има в предвид при дизайн на системи за навигация:

- трудността при определяне на необходимия брой генотипове за работни пространства с различни размери (брой възли) за формиране на крайното решение за възможно най-кратко време;
- в големи работни пространства с увеличаване на дължините на хромозомите (много възможни пътища) броят на генотиповете нараства експоненциално, което се отразява негативно на цялостната производителност;

- използването на ГА в динамични работни пространства, включително за групово управление, е трудно.

1.4.3. Алгоритми с размита логика (Fuzzy logic)

Размитата теория е представена за първи път от Zadeh [50] за справянето с несигурността. Размита логика е система, основана на правила, които могат да разчитат на практическия опит на оператора, особено полезна за улавяне на знанията на опитен оператор [25]. Подходът на размитата логика имитира начина на вземане на решения при хора, което включва всички междинни възможности да варират в степен между 0 и 1.

Решаването на проблема за планиране и проследяване на пътя с алгоритъм базиран на контролер с размита логика се основава на три основни етапа [87-A]:

Дефиниране на входни и изходни променливи. Контролера на размита логика при [87-A] съдържа:

- Изходен сигнал (резултантна на контролера) включва: „V“ – линейна скорост на МР по X и Y; „ω“ - е ъглова скорост на движение на МР;
- входни параметри на размития регулатор: „θe“ – грешка при ориентацията и “R”;

Задаване на функцията и правила с входни и изходни променливи. Входните и изходните сигнали съответстват на логико-лингвистични променливи, чиито стойности се определят от набори от термини:

- за “R” - OFF (O), Too Very Near (TVN), Very Very Near (VVN), Very Near (VN), Near (N), Far (F), Very Far (VF), Very Very Far (VVF), Too Very Far (TVF);
- за “θe” Too Very Far Negative (TVFN), Very Very Far Negative (VVFN), Very Far Negative (VFN), Far Negative (FN), Near Negative (NN), Very Near Negative (VNN), Very Very Near Negative (VVNN), Too Very Near Negative (TVNN), OFF (O), Too Very Near Positive (TVNP), Very Very Near Positive (VVNP), Very Near Positive (VNP), Near Positive (NP), Far Positive (FP), Very Far Positive (VFP), Very Very Far Positive (VVFP), Too Very Far Positive (TVFP).

Дефиниране на правилата за размити изводи (дефазификация). Правилата за размити изводи отразяват връзката между входните и изходните набори от термини:

(IF R is F AND θe is FN THEN Vx is OFF, „ω“ is N), - когато резултантната “R” е далеч и грешката „θ“ е далеч отрицателна, тогава линейната скорост по оста „X“ е изключена/забранена, а ъгловата скорост е отрицателна.

Така въз основа на процеса на размиване на входните променливи, се получава общо 153 - размити производствени правила, които са извлечени за контролера [87-A]. По време на работа на алгоритъма на първо място се извършва търсене по променливата „целева“, което прави възможно по-ефективното използване на изчислителните ресурси на МР.

Основните отличителни черти на алгоритмите за планиране, базирани на размита логика, са:

- умерено потребление на MR хардуерни ресурси;
- изчислителните разходи са практически независими от размера на матрицата за конфигурация на работната зона.

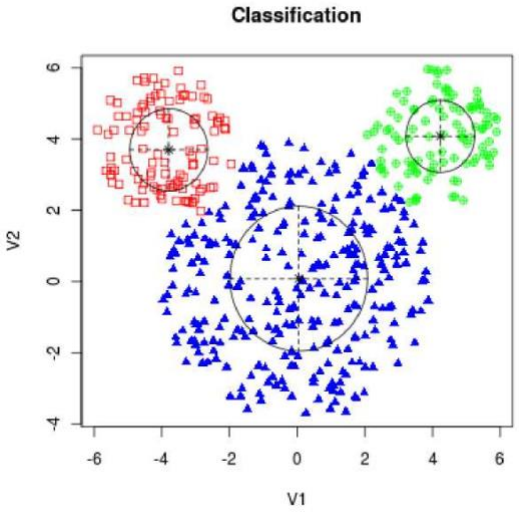
Тези характеристики благоприятно отличават размитите алгоритми от други интелигентни подходи, особено когато се използват за контрол група роботи с ограничени изчислителни способности. Като основен недостатък на алгоритъма за планиране, базиран на размита логика, може да се отбележи не толкова високото качество на изграждане на траектория в сравнение с други интелигентни подходи. Въпреки това, в рамките на проблема за планиране, размитият алгоритъм може успешно да се използва в комбинация с други видове алгоритми за изграждане на траектория,

например за локална корекция на траекторията при избягване на сложни препятствия [87].

1.4.4. Машина с поддържащи вектори (МПВ)

Машина с поддържащи вектори (МПВ) или познато на английски: support-vector machine (SVM) е метод за машинно самообучение. „Машините с поддържащи вектори“ са класификатори за определяне на максимален марж, които получават оптимална разделителна хипер-равнина между наборите от данни [89]. МПВ е разработен основно за решаване на проблема с класификацията на данните и е успешно приложен от лабораторията AT & T в техния проект за оптично разпознаване на символи (OCR). Методът на опорните вектори основно се използва за решаване на задачи като класификация и регресия. По-късно беше разширен, за да се справи с проблемите с машинното обучение и прогнозирането. През последните няколко години МПВ също се прилага успешно в изследователските области, отнасящи се до мултисензорно информационно сливане, зрение на роботите, взаимодействие с човешки робот и планиране и навигация на мобилен робот [89].

МПВ представя обучаване от примери представени като точки в многомерно пространство. Примерите са проектирани в това многомерно пространство по такъв начин, че примерите от различни класове да са възможно най-добре разделени помежду си. Те трябва да бъдат линейно разделими от хипер-равнина. Тази хипер-равнина трябва да се избере по такъв начин, че да се намира възможно най-далеч от примерите и на двата класа. Класификацията на нов пример става като той се проектира в същото пространство и се определя класа му според това от коя страна на хипер-равнината се намира. Примерна схема на работа на алгоритъма за двуизмерно пространство е представена на фигура 1-11. Методът на опорните вектори може да работи не само за линейна класификация, но са необходими модификации на първоначалната му формулировка. k-Means е метод за разделяне при извличането на данни. Този метод разделя данните на k на брой клъстера с помощта на Евклидовото подобие в разстоянията. Препятствията за роботите биват класифицирани или групирани на основата на характеристики в клъстерите.



Фиг. 1-11 Клъстеризация на данните в три различни клъстера (K=3)

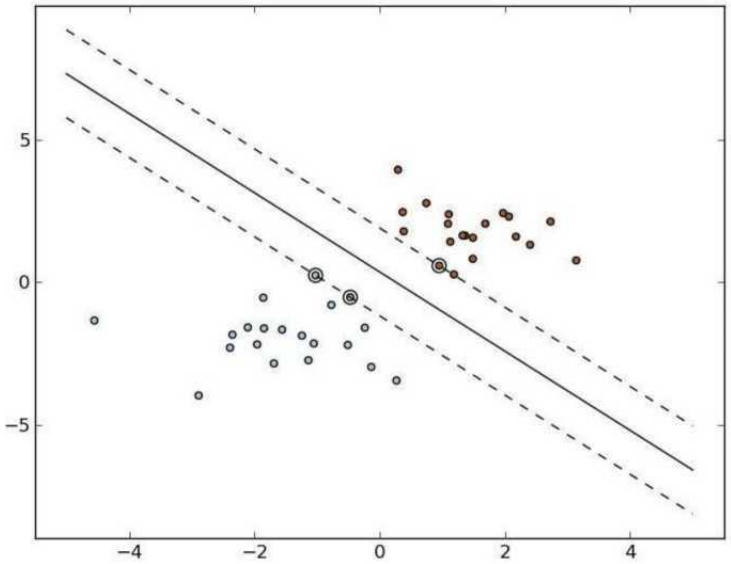
Евклидовото уравнение което се използва в случая за намиране на разстоянието между два обекта е:

$$D(a,b)= D(b,a)= |a-b|= \sqrt{\sum_{i=1}^{n}(bi - ai)^2}$$

където a и b са двата обекта, а bi и ai са техните координати в n- мерното пространство.
Основни стъпки за клъстеризация на данните чрез k-means:

- Избира се произволен брой центрове за клъстерите
- Назначава се всеки обект към най-близкия център с помощта на Евклидовото уравнение
- Премества се всеки център в средата на назначените към него обекти

Втора и трета стъпки се повтарят докато разликата в промяната на местоположението на центровете е по-малка от предварително определен праг.
Като предимство се счита относителната ефективност при групиране на незашумени данни, а като недостатък - не може да се справи със зашумени данни.



Фиг. 1-12 SVM работа на алгоритъм за двуизмерно пространство

За да се приложи МПВ за планиране на пътя, цялата среда, която роботът открива, се разделят на два класа набори от данни [88] и МПВ се използва за определяне на максималната хипер-равнина на марж между наборите от данни, принадлежащи към двата класа. Тази хипер-равнина ще представлява път без сблъсък, ако приемем, че има такъв. Конвенциите, следвани за резултатите от симулацията, са показани на фиг. 1-11. Двата класа с етикети -1 и +1 са представени съответно в червени (+) и зелени (x) точки, а пътят на МПВ е представен от сините (*) точки. Описание на алгоритъма за намиране на път с избягване на препятствията [89]: Всички препятствия трябва да бъдат етикетирани като положителни или отрицателни, преди да се приложи МПВ. Един от начините е да тествате всички възможни модели на етикети и да изберете най-добрия, който осигурява най-краткия път. Това обаче е очевидно непреодолимо, когато броят на препятствията е голям. Следователно се използва рандомизиран подход, при който ограничен брой модели на препятствия (етикетиране) се избират на случаен принцип и се тестват. За всеки избран модел се генерира положителни и отрицателни проби и се подават на МПВ, за да изчисли разделителна повърхност.
Схемата на алгоритъма е както следва:

- определят се виртуални препятствия около старта и целта;
- създава се първоначалния модел на препятствие и се генерират положителни и съответно отрицателни проби;
- задават се направляващи проби, за да огради и определи възможния обходен

регион;
- прилага се МПВ към генерираните проби и извличане на осъществим път чрез анализиране на научения модел;
- изпробват се и други модели, докато условието за прекратяване не е удовлетворено.

В случаите когато МПВ се използва за намиране на път и избягване на препятствия, калкулирания път, получен от МПВ, е гладък, без инциденти и улеснява непрекъснатото движение на робота дори и в условия на неясна среда.

Установено е [88], че изчислителната сложност за класификация на нови точки от данни нараства с увеличаването на размера на набора от данни, съдържащ по-рано класифицираните точки. Може да се зададе подходящо ограничение за размера на данните, за да се направят изчисленията по-бързи. За това МПВ най-често се използва в ситуации, които изискват както спазване на лентата, така и избягване на препятствия, използването на МПВ ще опрости проблема. Това би елиминирало обхвата на всякакви конфликти, които могат да възникнат поради използването на два различни алгоритъма за задачите.

1.5 Други алгоритми за колективна интелигентност и интелигентни агенти за навигация за ГМР

Различни трудове засягаща груповата роботика през последното десетилетие предложи нови стохастични подходи за контрол и управление на движението групи роботи. Те включват интерактивно търсене на оптимизирано решение и се ползват успешно при задачи свързани с намиране на най-кратък път. Един от първите подходи е описан в работата на К. Рейнолдс [1], където авторът е подчертал свойствата на движението на групата, които трябва да се спазват, така че да се считат за валидни и подчертава местните правила, на които агентите в групата трябва да се подчиняват, за да получат тези свойства. Съвкупното движение на симулираното ято се създава от поведенчески модел, подобен на този в естествено ято - птиците сами избират курса си. Всяка симулирана птица е реализирана като независим агент, който се ориентира според местното си възприятие за динамичната среда, законите на симулираната физика, които управляват нейното движение, и набор от поведения, програмирани предварително в нея от „аниматора“. Съвкупното движение на симулираното ято е резултат от плътното взаимодействие на относително простото поведение на отделните симулирани птици. Движението на база такива свойства може да се причисли към моделите на стадно управление, които се наблюдават в животинския свят – птиците например. „Стада“ или „рояка“ частици (в случая от n-на брой мобилни роботи) са популярна изследователска тема когато става въпрос за група роботи. В рамките на този подход съществуват методи на база на интелигентност на „рояка частици“[52, 53]. В рояка (групата) трябва да се правят безопасни движение, когато всеки изпълняват определена задача, докато поведението не трябва да зависи от броя роботи в стадото и да е устойчиво на възникнали повреди. Най-съвременните алгоритми за клъстеризация, базирани на инструмента интелигентност на рояка частици [10], са „оптимизация на рояка частици“ (Particle Swarm Optimization - PSO) [20] и „оптимизация на колонията от мравки“ (Ant Colony Optimization - ACO). ACO е мета евристика за решаване на комбинаторни оптимизационни проблеми [AARTON2020]. Те се причисляват към еволюционният клас от методи както са ГА. За разлика от алгоритмите, базирани на генетични методи, при оптимизиране на база „рояка частици“ се използват агенти, които се движат и задействат в свободното пространство. Тези индивиди в повечето случаи са моделирани по животни. След серия от повторения, движението на тези индивиди към целта създава

модел, който се сближава с общия целеви път.
Алгоритми за оптимизация, които симулират подобно поведение на мравки, бяха предложени в началото на 90-те години. Първата статия за “алгоритмите за мравки” беше публикувана в международно списание през 1996 г. [96] и след няколко години бяха необходими нови изследвания (Swarm Intelligence и Ant Algorithms). Както обикновено, ACO алгоритъмът се състои от три стъпки: избор на модел, актуализиране на феромони и итерация (Blum, 2005) [97]. На първия етап изборът на модел представлява серия от промени в позицията на мравка, които се определят от феромона, базиран на стохастичния механизъм. В ACO алгоритъм, процесът на актуализиране на феромони е много важен. Стойностите се актуализират от всички мравки, които преминават през възела. Поради количеството „феромон“, натрупан във възела, може да стане така, че да се объркат другите мравки, избирайки неоптимизиран път по погрешка. Затова може да включи и качествен параметър – примерно близост до обект с определена геометрия, която вече е забелязана от „стадото“.
В статията [2] също се предлага подход, основан на местните правила, но той се възпроизвежда от поведения, подобни на състоянието на материята (ето защо се причисляват към физически вдъхновени алгоритми).
Матарич [3] конструира алгоритъм, който изрично решава проблемите на движението в група (на първо място, избягване на сблъсъци). Предложен е модел на запазването на енергията на групово ниво, като целта е минимизиране на интерференцията между индивидите. На ниво агенти това се превъплъщава директно в целта за —достигане — придвижване без—сблъсъци - тоест безопасно движение на отделните агенти вътре в групата. Важно е да се отбележи, че все още има много малко реални проекти с използването на тези подходи.
Някои от по-известните приложения са два проекта за наблюдения на водни площи: в [4] рояци роботи трябва да се координират, за да наблюдават и покриват непрекъснато възможно най-голяма част от зададена зона; в проекта CoCoRo [5] има група от подводни роботи с различна функционалност като се извършва мониторинг на акваторията и се взимат проби от морското дъно. И в двете работи се прилагат правила за поведение на „стадото“ („рояка частици“). Подобни решения са търсени и актуални не само при подводни изследвания или военни и спасителни операции. Наборът от задачи за групи роботи не се ограничава само до задачи за използване следващи водни пространства. Изобщо всякакви приложения на мобилни роботизирани платформи в условия затворена среда попадат в полезрението на това изследване. В публикациите [6], [7] има формални задачи, които могат да бъдат решени от групи роботи: събиране в една точка, формиране на модели, вериги, самостоятелно сглобяване, събиране на ресурси, координиране насочено движение, избягване на препятствия, разгръщане, колективно изследване. Развитие на територията, колективен транспорт, разпределение на задачите. В скорошно проучване [8] в областта на логистиката се предлага използване на модулни роботи – нещо съвсем актуално и популярно днес.
Виртуалните сили на роботите могат да възпроизвеждат реални физически сили между молекули [9] (възприемането на робот за материална точка), или отразяват правилата за взаимодействие между роботи и препятствия. По същество системата действа като симулация на молекулярна динамика ($\vec{F} = m*\vec{a}$). Така, на абстрактно ниво агентите се третират като физически частици. Това позволява този алгоритъм да бъде въплътен в превозни средства, вариращи по размер от нано роботи до сателитни системи.

1.6 Сравнителен анализ на наличните подходи

Различните техники от изкуствения интелект предлагат различни предимства и

недостатъци за динамично планиране на пътя на мобилния робот.
В рамките на решаването на проблема за планиране на траектории на движение за група от МР могат да се разграничат примерни критерии, които биха предопределят ефективността при прилагането на всеки един от алгоритмите за планиране като:

- висококачествена конструкция на траекторията, т.е. алгоритъмът дава възможност за формиране на траектории, близки до оптималните (най-къси или близки до най-късите при условията на дадена конфигурация на работната зона).
- умерено потребление на МР хардуерни ресурси.
- лесно формализиране на задачата за планиране (т.е. простота математическо представяне на конфигурацията на работната зона).
- ефективност (скорост на изчисляване на траекторията) в условия на сложна конфигурация на работното пространство (т.е. при наличие на голям броят на препятствията или препятствията със сложна форма).
- възможност за ефективно приложение за планиране на траектория в условия на динамично променяща се конфигурация на работната зона (т.е. при наличие на движещи се препятствия, както и други МР).

По долу в Таблица 1-1 са показани едни от основните предимства и недостатъци на някой от типичните алгоритми.

Метод	Предимство	Недостатък
Традиционни подходи (алгоритмите Дейкстра и A*)	Познати и масово използвани; Лесни за изчисление;	Трябва да се комбинират с други подходи; При големи карти значително се увеличава времето за решаване на задачата; Алгоритъмът може да не работи правилно, когато работи в динамично променящо се работно пространство или планиране на групова траектория, където е необходимо да се вземе вземат предвид допълнителни правила за разрешаване на конфликтни ситуации между агенти;
Изкуствени потенциални полета (традиционен подход)	Като част от групата на на класическия алгоритмите като подход те са едни от първите внедрявани, най-адаптивни и масово използвани; Намирането на път до целта не изисква сложни математически изчисления и от там има занижени изисквания към изчислителната мощ към хардуера. Може да се използва в комбинация с други.	Методът на потенциалните не позволява маневриране и навигация около по-сложни пространствени конструкции - препятствия и роботи; Осигурява ефективно формиране на траектории на роботи само в случаите, когато контурите на елементите на околната среда са описани с изпъкнали многоъгълници или кръгове. В противен случай могат да възникнат т. нар. „локални минимуми“, за чийто изход са необходими допълнителни евристични правила [57].

Невронни мрежи „Хопфийлд“	Предполага използване на паралелни изчислителни техники, което го прави удачно решение в динамични работни пространства; Самоадаптивност и самооранизация - актуализиране на картата в реално време при динамични среди и без необходмиост от човешка намеса; Намира се оптимална трактория при сравнително добра производителност.	Сложност на хардуерното внедряване с високо компютърно време - до 7 математически операции за всеки неврон; Високи изисквания към изчислителната мощ на хардуера; Трудност при построяване на траекторията при обход на препятствия със сложна форма – особено при малки пространства;
Алгоритми с размита логика (Fuzzy logic)	Няма високи изисквания към изчислителните хардуерни ресурси - не зависят от размера на конфигурационната матрица на работната зона; Масово използване при група роботи с ограничени изчислителни възможности на бордови компютри (микроороботи) при конструиране на траекторията в сравнение с други интелигентни подходи; Успешно се използват в комбинация с други видове подходи за изграждане на траектория за избягване на сложни препятствия.	Като основен недостатък на алгоритъма за планиране, базиран на размита логика, може да се отбележи не толкова високото качество на на конструиране на траекторията в сравнение с други интелигентни подходи. Въпреки това, в рамките на проблема с планирането, алгоритъмът може успешно да се използва в комбинация с други видове алгоритми за изграждане на траектория, например за локална корекция на траекторията при избягване на сложни препятствия възникнали неочаквано в полезрението на агента.
Машина с поддържащи вектори (МПВ)	Калкулирания път е гладък	Ефективността при откриване на правилния път напълно зависи от обучението и данните на системата следователно и изчислителната сложност за класификация на нови точки от данни нараства с увеличаването на размера на набора от данни.
Генетични алгоритми	Устойчивост, адаптивност към околната среда; Осигурява оптимални решения дори за сложни изчислителни проблеми; С паралелни технологии позволява гъвкав глобалното търсене; Използването на този ГА за решаване на проблема с търсенето при определени условия дава високо увеличение на производителността в сравнение с алгоритъма на Dijkstra: средно 2 пъти по-бързо, със сравнима точност.	Трудност при определяне на необходимия брой генотипове за работни пространства с различни размери (брой възли), за да се формира крайното решение за възможно най-кратко време; В големи работни пространства с увеличаване на дължината на хромозомите (много възможни пътища), броят на генотипите нараства експоненциално, което се отразява негативно на цялостната производителност; Използването на ГА в динамични работни пространства е трудно.

Таблица 1-1 Сравнителна таблица с предимства и недостатъци на основните методи на навигация при УДМР

В момента методите, базирани на изчислителна интелигентност (computational intelligence), се развиват допълнително. Разработват се и методи, използващи хибриден подход, съчетаващи класически подходи за глобално планиране с подходи за местно планиране, използващи сензорни данни. Значителен брой съществуващи алгоритми поддържат възможността за препланиране, но това изисква актуализиране на глобална информация за околната среда по всяко време. По-адекватна ситуация за реални задачи е, когато роботът има достъп до глобална информация за околната среда в началото на задачата, като в хода на изпълнение тази информация се допълва само локално, поради сензорни данни. В изследванията не се обръща достатъчно внимание на разработването на подходи, които осигуряват хомогенна интеграция на решения на различни навигационни проблеми. Изключение правят задачите за локализация и картографиране, които се решават едновременно. Проблеми с локализацията, планирането на пътя и управлението на движението в работите трябва да се разглеждат поотделно, без да се отчита взаимното влияние на грешките, възникващи на всеки етап. Съществуващите проучвания обръщат малко внимание на решаването на проблема с непълнотата и неяснотата на информацията за околната среда. Решенията, възникнали въз основа на неправилно възприета информация за околната среда, имат определени последици, борбата срещу които трябва да се основава на динамично ситуационно препланиране. По-горе в проучването ни бе установено, че методологията на мулти-агентни системи изтъква над традиционните системи, базирани на изкуствен интелект при системите за навигация. Най-важните аспекти на интелигентните подходи са висока точност, самообучение и устойчивост. Описаните в източниците практики показват, че резултатите за откриване на пътя, използващи системи, базирани на множество агенти, постоянно се увеличават, тъй като процентът на допуснати грешки при тях намалява.
Несъмнено подходите, основаващи се на ИИ, потенциално биха могли да постигнат по-голяма гъвкавост, което ще ги направи още по-популярни в близко бъдеще. Както бе споменато по-горе, интелигентните алгоритми имат редица предимства пред класическите методи за планиране, по-специално те позволяват да се намали количеството на изчисленията, а също така могат да работят в условия на недостатъчна информация за конфигурацията на работното пространство. Заслужава да се отбележи обаче, че проучването разглежда основно методи за планиране, които използват дискретизацията на работното пространство, т.е. методи за неточно позициониране. В същото време съществуват редица класически алгоритми, които осигуряват висока точност на позициониране, а също така отчитат сложната форма на препятствията (която може да бъде приблизително апроксимирани чрез дискретизация) [68, 69, 70, 71]. Такива алгоритми (например техники, използващи „потенциални полета“ [74]) имат значителни ограничения при работа с карти с висока плътност на препятствията: поради всеобхватното разпределение на „отблъскващото“ поле в близост до тези препятствия има висока вероятност от локални минимуми, които правят невъзможно следването на пътя. Независимо от това, тези методи могат да бъдат ефективно комбинирани с алгоритмите за планиране, обсъдени по-горе и използвани за избягване на местни препятствия и както ще видим по-долу в тази дисертация техниката на ИПП може да приложи с успех при модела на следване на лидера във формация от МР, където или да замести НМХ с цел опростяване на математическите калкулации и бързина или да се комбинира с него за подобряване на резултатите в лоша среда.

1.6.1 Предимства при използването на невронни мрежи и интелигентни навигационни подходи в роботиката

Теорията на невронните мрежи е активно развиваща се област на науката, чиито основни

перспективи са свързани със сложни практически проблеми, чието решение не може да бъде получено под формата на математически зависимости.
Първата работа, която положи основите на последващото развитие на теорията за изкуствените модели на неврони и невронни мрежи, се счита за статията „Логически изчисление на идеите, присъщи на нервната дейност“ на Уорън Маккулок и Уолтър Питс, публикувана през 1943 г. [75]. Основният принцип на теорията на McCulloch и Pitts е, че апаратът на математическата логика може да се приложи за описване на невронната активност на мозъка: всеки неврон (с определени ограничения) е представен като праг логически елемент, работещ на принципа „всичко или нищо“ (закон за всичко или нищо), а мозъкът, съответно, като мрежов набор от тези елементи [76].
По-нататъшното развитие е свързано с изследванията на Ф. Розенблат. Резултатът от неговата работа е една от първите изкуствени невронни мрежи, която е способна на възприятие („възприятие“) и формиране на реакция на възприемания сигнал, този модел се нарича перцептрон (PERCEPTRON) [77].
Последващата работа на Мински и Пиперт [78], която показа ограничените изчислителни възможности на конвенционалния персептрон [79], предизвика известно прекъсване в изследванията в областта на теорията на НМ, но от началото на 80-те години на миналия век настъпва нов етап в развитието на модели на невронни мрежи е започнало и продължава и до днес. В много отношения се свързва с произведенията на Т. Кохонен („самоорганизиращи се карти“) [80], Д. Хопфийлд („Мрежа на Хопфийлд“) [46], К. Фукушима („неокогнитрон“) [82] , R. Hecht-Nelson („backpropagation network”) [83] и други.
В момента има голям брой различни видове невронни мрежи и алгоритми, предназначени за решаване на различни проблеми. Тези модели се различават по структурата на връзките, броя на слоевете неврони, както и правилата за определяне на теглата и методите на обучение. Тези методи за решаване на проблеми се използват активно в различни области на дейност, но има редица условия, които определят целесъобразността на използването на тези подходи на невронната мрежа:

- разглежданият проблем не подлежи на формализиране, т.е. съдържа елементи на несигурност, които не са формализирани от традиционните математически методи [63];
- разглежданият проблем е формализиран, но в момента няма апарат за решаването му;
- съществува подходящ математически апарат за решаване на разглеждания проблем, но изпълнението на изчисления с негова помощ върху съществуващата хардуерна база е неподходящо, на базата на съществуващите времеви, енергийни и др. ограничения. В тази ситуация е необходимо или да се опростят алгоритмите, което намалява качеството на решението, или да се приложи подходящ подход на невронна мрежа, при условие че осигурява необходимото качество на задачата;
- възможността за паралелизиране на изчислителни процеси (например за анализ на динамични процеси, като се вземат предвид различни външни смущения), което значително увеличава скоростта и ефективността на решаването на проблема [67].

Поради тези предимства, НМ се използват интензивно при нелинейно управление на сложни динамични обекти, обработка на изображения и разпознаване на изображения, текст и глас [84], както и адаптивно филтриране, идентификация и финансово прогнозиране [85].
В много отношения разширяването на обхвата на областите на приложение на НМ е

свързано с изследвания в областта на повтарящите се НМ, чиято архитектура прави възможно запомнянето и възпроизвеждането на цели последователности от изходни сигнали за един входен сигнал. Тази функция значително разширява възможностите за използване на подобна архитектура на невронната мрежа за изследване и симулация на динамични процеси, както и за изграждане на ефективни системи за управление, и както се наблюдава включително в роботиката.

1.7 Дефиниране и формулиране на проблема, цели и задачи

В момента МР и роботизираните системи (РС) се използват широко в различни сфери на човешката дейност, освен това са идентифицирани задачи, чието ефективно решение е възможно само при организиране на хомогенни и хетерогенни колективи от роботи. Към роботизираните агенти в условия на колективна работа се налагат следните изисквания: миниатюризация, наличие на развити средства за комуникация, висока степен на точност и автономност и т.н., което от своя страна води до появата на редица специфични проблеми за решаване от единичните и колектива като цяло.

Усилията на много изследователски лаборатории в областта на роботиката са насочени към изграждане на автономни колективи от роботи за различни цели и към решаване на проблеми, свързани с тази цел. Уместността на тази област се дължи на факта, че когато роботите се използват за изпълнение на повечето от задачите, които включват тяхното автономно функциониране за дълго време (монтаж, строителство, превоз на товари, проучване на терена и др.), колективните решения са по-ефективни в сравнение с един робот, ниска цена, висока надеждност и устойчивост на външни смущения.

Когато група мобилни роботи (МР) работи в недостатъчно позната среда, проблемът с планирането на безопасни траектории, позволяващи изграждането на безпроблемни маршрути за пътуване, става спешен. Ето защо и една от базовите функционалности на МР е правилното планиране на движението на отделните индивиди в рояка. Доброто познаване на позицията на всеки индивидуален робот чрез използване на разнообразие от сензори и възможността за локализиране е необходимо условие за реализиране на всяко задание, включително при управление на движението им особено когато те работят в група и трябва да се съобразяват с останалите. Една от най-често срещаните причини за грешки при мобилните роботи е неправилно изпълнение на навигационните операции във реално време. Такива грешки са по чести и възможни в непозната среда, където от критическо значение са безопасността на персонала и правилното съхранение на манипулираните обекти.

За да се разбере какво е „роева“ / колективна роботика, е дадено определение, взето от Sahin [13]: „роева“ роботика е изследване на това как голям брой относително прости физически представени агенти могат да бъдат проектирани така, че желаното колективно поведение да се появи от локалните взаимодействия между агентите и между агентите и околната среда. При проектирането, анализа и прилагането на методи и алгоритми за управление на група мобилни роботи възникват редица проблеми. Голяма част от проблемите са свързани с специфичността на този тип МР, които имат:

- по-голям радиус на действие, постигнат поради разпръскването на роботи по цялата работна площ;
- съдържат разширен набор от функции, изпълнявани в случай на използване на хетерогенна група роботи, т.е. роботи с различни функционални цели;
- по-голяма вероятност за изпълнение на задача в екстремни ситуации (например в космоса, под вода или в зона на радиоактивно замърсяване). Това предимство се дължи на възможността за преразпределение на задачите между роботи в група

в случай на повреда на някои от тях [14].

1.7.1 Обхват и предмет на темата на изследване

Локализация, навигация и координация на обекти в пространството са проблеми, занимавали човечеството непрекъснато. В миналото навигацията се е считала за едно от най-важните изкуства, наред с военното дело и медицината. Управление на движението на група роботи/агенти е широка област в роботиката съдържаща много отворени въпроси и за това е и с голям потенциал за развитие в бъдеще. Абревиатурата "SWARM" като „колония" или „рояк" и интелигентност на рояка описани в книгата на B. K. Panigrahi, Y. Shi и M.-H. Lim е иновативна парадигма за решаване на оптимизационни проблеми, организирани от изследването на колонии или рояк от социални организми. Изследванията на социалното поведение на организмите (индивидите) в рояка подтикнаха към проектирането на много ефективни алгоритми за оптимизация при групите. Най-често такива агенти се уповават на локални правила за взаимодействие на роботите един с друг в заобикалящата среда и възникващо последващото поведение, произтичащо от прилагането на тези правила. По този начин параметрите на един алгоритъм нямат изрична интерпретация по отношение на поведението на групата, което затруднява единно решение на задачата за навигация.

Ето защо и привлича вниманието на много учение хора поради предимствата, които се предлагат като групи от примитивни роботи при решаване на сложни проблеми в посока кооперииране (например устойчивост на аварии, способност за решаване на няколко проблема едновременно, мащабируемост на капацитета, и т.н.).

В практиката съществуват много техники, които можем да разделим на класически и иновативни, според подхода ползван за навигация. В тази глава ще се разгледа единствено и само пряко свързаните подходи и техники с тематиката на дисертационния труд. По-подробна информация за базовите навигационни методи може да бъде намерена в [18][19]. Ще се разгледа, също така и аспекти на кооперативната роботика, и по-точно кои са предимствата и пречките при навигация и планиране на движение на множество роботи във формация.

През последните две десетилетия много внимание беше отделено на изследванията в областта на технологиите за кооперииране при мобилните роботи. Ползите от използването на такива групи са ясни. Първо, това е по-голям радиус на действие, второ, разширен набор от изпълнявани функции и накрая, по-висока вероятност за успешно завършване на задачите. Характеристики на рояка мобилни кооперативни роботи:

- Изпълняват задачи без централен контрол или източник на данни (база данни, т.н.;
- Ограничена комуникация;
- Не е необходим (изричен) модел на средата;
- Имат сензори за възприятие за околната среда (усещане);
- Способност за реагиране на промените в околната среда;
- Социалните взаимодействия (локално споделени знания) осигуряват основата за решаване на проблема;
- Ефективността на усилията е свързана, но не зависи от степента или наличие на свързаност/мрежата между агентите и броя на взаимодействащите агенти;
- Силни примери са за решаване на аналогични проблеми в природата и физиката;
- Оцеляване в стохастична и враждебна среда;
- Социалното взаимодействие създава предпоставки за сложно поведение;
- Поведения, и нагаждане в условия на променени и динамична среда;

Ето защо предмета на изследването обхваща проблемни области при управлението на МКР като локализация, координация и навигация, сензорни системи и компютърно зрение, операционни системи и комуникационен интерфейс.
Предмета включва също така архитектурата, моделите и алгоритмите на системата за планиране, която осигурява коректността и безконфликтната траектория при движение в двуизмерно работно пространство с препятствия.

1.7.2 Обект, актуалност, цел и задачи на изследването

Обект на изследването са системите за планиране на траекторията на мобилни роботи в двуизмерно работно пространство с препятствия, действащи като част от група както автономни агенти децентрализирано.

Актуалност и значимост на дисертацията.

Едно от изначалните условия в роботиката е да има осигуряване на предсказуемост при планиране на движението и добро познаване на позицията на робота в реално време. Разбира се възможността за локализиране на МР в околната среда е необходимо приоритетно условие за реализиране на работното му задание и в дисертацията се приема за даденост наличието на прецизна локализация, която посочва конкретна моментна позиция на МР. И тъй като има много проучвания в областта на локализацията, смятаме, че именно планиране на оптимизираното и безаварийно придвижването, така че да се избягват възможните инциденти и да се използва опита на отделния индивидуален робот във всеки един момент би могло да има по-голям принос. Развитието на роботиката в последното десетилетие налага и необходимостта от създаване на нови подходи за решаване на проблема с груповия контрол чрез модерни мулти-агенти системи, които да предложат прогнозируемост и предсказуемост на навигационната динамика от начало до края на поставената задача. Вземането на решения в реално време трябва да става чрез количествени критерии коригирани с общото груповото поведение. Това означава, че процеса на навигация за всеки агент, ще включва получаване информация в реално време под формата на данни от други контролирани процеси и от други участници и е в състояние да влияе чрез управлението на тези процеси, допринасяйки за постигането на поставената цел. Ето защо в дисертацията се предлага актуално решение и ще си постави за цел намаляването на грешките при изпълнението на задание от групи агенти.

В резултат на направения обзор могат да се формулират група проблеми които да бъдат заложени като цели на дисертационния труд, а самите цели да се разбият на задачи, решението на които да доведат до постигане на поставените цели. Както видяхме се налага да планираме движенията в групата роботи така, че разположението винаги да е благоприятно за изпълнение на конкретната задача както и се избягват инциденти при навигацията и да поддържаме ниска грешка при отчитането на всяка позиция на МР.

Целта на дисертацията е да се направи изследване на оптимизационни методи, алгоритми и системи от ИИ за управление на траекторията и движението при група мобилни роботи. Провеждането на тези изследвания трябва да допринесе за развитието на УНМР, като подобри и разшири система за навигация, която да се използва при мобилните роботи. В тази връзка резултатите и направените изводи ще са основа за предлагане на оптимизации и подобрение на вече съществуващи алгоритми свързани с ориентация, координация и придвижване в затворени пространства и избягване на препятствия и поведение на МР в група. Крайната цел е да бъде предложен модел - алгоритъм за децентрализирано планиране на траекторията (т.н. „планировчик“) *и избягване на* препятствия от роботите с помощта на т.н. *„невронна карта“* с подобрена НМХ.

Във връзка с **основната цел** са формулирани следните **изследователски задачи в дисертацията**:

- Анализ на настоящото състояние на научните изследвания и проблемите в областта на УНМБ в условията на непозната среда и колективна работа в екип;
- Изследване на методи за подобряване на планиране на процеса на навигация при голям брой мобилните колаборативни роботи;
- Разработване и предлагане на метод за управление на множество от МР в условия на работа в екип и кооперирање, на основата, на който ще се разработи софтуерна автономна система за динамично планиране на пътя (решаване на навигационна задача в реално време) с избягване на сблъсъци в затворено пространство;
- Създаване на виртуална постановка с разработения алгоритъм за движение на МР и провеждане на реални експерименти и визуализиране на резултатите от проведените изследвания и разработки в реално време;
- Предлагане на актуална компютърна стимулационна навигационна платформа даваща възможност за експериментиране и потвърждаване на аналитично изведените модели за навигация и съпътстващите и закономерности.

1.8 Мотивация и методи на изследване

Основен проблем при приложенията в непозната среда, е липсата на външна информация, която да бъде ползвана за коригиране на грешките в оценката за състоянието на робота. За определяне на позицията си, мобилния робот разчита на сензорна информация. Сценарии с кооперирање между МР налага сензорната информация на всеки агент/робот да бъде споделена с останалите. Сензорите за вътрешно състояние се характеризират с висока честота на актуализация, което поставя допълнително условие за бърз обмен на данни и обработка в реално време. Налага се внедряване на подобрени подходи в областта на системите за навигация за кооперативни роботи, като управление на комуникацията на база „събития“, обмяна на данни между приложенията с мултинишкови процеси. Така системата може да бъде напълно самостоятелна, без да зависи от наличие на конкретна подреденост в работната среда, служеща за ориентация. Подобни решения са търсени и актуални при дълбоко-подводни изследвания, изучаване на други планети или изобщо приложения, при които липсва видимост към сателит, какъвто е общия случай на работа в затворена среда.

От друга страна, най-бързите сфери за развитие на изкуствения интелект са могли да бъдат по-ефективно използвани като се има предвид, че интелигентните агенти могат да бъдат използвани при изпълнението на решения за УДМР, за да се постигне бързодействие и при използване на минимален хардуерен ресурс.

За да може успешно да се реализира целта ще бъдат използвани мулти-агентни технологии от изкуствения интелект с възможност за самообучение на база опит споделяне между роботи в условия на вътрешна непозната и неструктурирана среда. За оптимизация на движението, координиране и позициониране на роботите ще бъдат използвани методи от ИИ, които ще са база за създаване на софтуерната автономна навигационна система (САНС), която действа самостоятелно и в реално време чрез алгоритъм на модифицирана НМХ предложен в Глава 2.

1.9 Структура на дисертационния труд

Дисертацията е базирана върху научно-изследователска работа. Съдържа литературен обзор, предложен собствен метод и модел на управление на движението на МР (УДМР)

в условия на „рояка частици“. Разработена е софтуерна имплементация за верифициране на предложения модел и са обсъдени получените резултати и направените симулации. Показани са методите на изследване

Настоящата дисертационна работа е структурирана както следва:

ГЛАВА 1: Обзор и анализ на състоянието на проблема. Представено е въведение и терминология при мобилните роботи. Анализирани са типовете мобилни роботи според тяхното предназначение и е систематизирано текущо състояние на методите от ИИ използвани в роботика и е изложена систематизация на видове групово управление при МР. Представени са както традиционните подходи като дърво на решенията и гради, транспортна задача, така и интелигентни системи - невронни мрежи, генетични алгоритми, изкуствени имунна системи, други стохастични алгоритми като Fuzzy logic, Naive Bayes, SVM, k-means както и такива за колективна интелигентност (Swarm Intelligence) като Particle Swarm Optimization (PSO). Направен е сравнителен анализ с предимствата и недостатъците за тяхното използване като подходи в ИИ при навигация и координация на роботизираните мобилни системи опериращи в условия на група. Дефиниран е проблема, представени са целите и мотивацията на дисертационния труд.

ГЛАВА 2: Фокусира внимание върху теорията т.н. „невронна карта“ и невронна мрежа на НМХ, основа за разработване на система за навигация и избягване на препятствия в Глава 3. Разгледани са различни възможни подходи при управление на навигацията. Представен е математически модел на алгоритъм „невронна карта“ на основа на НМХ, който ще се използва за глобално и локално планиране за група МР в дисертацията. Описан е конструктор на пътя, който реализира този алгоритъм на управление на навигационна система за група от роботи. Предложени са критерии за оптималност на модела НМХ и са предложени подходи за модифициране. Представена е методология за прилагане на паралелни изчислителни технологии която ще се използва за внедряване на софтуерната част на системата за планиране. В тази глава са представени и симулации на описания модел с НМХ на различните нива в системата реализирани с помощта на програмния език Python. След задълбочен анализ в тази глава е представен нов оптимизиран и модифициран *модел за изграждане на невронна карта, с който да се експериментира в дисертацията*.

ГЛАВА 3: Трета глава е посветена на разработения експериментален прототипен алгоритъм за планиране на траекторията за групата МР. Описани са техническите изисквания към алгоритъма като задание. Описват са всички модули, контролери, входни и изходни променливи и параметри на навигационна система. Разяснени са възможностите и принципа на работа на операционна система за управление на роботи – ROS, като среда за управление и контрол на МР. В тази глава са описани лабораторната обстановка, направените експерименти и симулации съобразно предложения в глава 2 модел на модифицирана НМХ, като се демонстрират различни подхода в стимулационна среда. Представена е средата и архитектура за провеждане на експерименталната постановка с нейните ограничения. За представяне на резултатите се използва симулационна платформа „Webots” с примерен модел на мобилен робот „E-puck“ [106], който се управлява от разработения за целите на дисертационния труд навигационния програмен код и навигатор наречен – „HNav“ за УДМР. Изложен е резултата от проведените научни експерименти като са систематизирани, анализирани и обобщени. Направени са изводи с критичен поглед по отношение на направените изследвания. Отчетени са подобрението в поведението на групата МР - точността на навигация и планиране на пътя, успеваемостта при разпознаване на средата, прецизността на предложения иновативен УДМР посредством различни устройства и интерфейси.

Към дисертацията е приложен списък с цитирани източници (литература и онлайн адреси), списък с авторските публикации, списък с фигурите и списък с таблиците.

1.10 Изводи по първа глава

Въпреки голямото количество литература и съществуващи приложения в тази област, проблемът за управление движението на група роботи в непозната работна среда е все още актуален. Повечето срещани системи са или неефективни или твърде сложни и като цяло може да се каже, че липсват ефективни решения. Добрите математически модели в практиката са базирани на итеративна оптимизация за намиране на решение на система от нелинейни уравнения. Това в случая многократно усложнява системата, тъй като намирането на решение е най-елементарната стъпка в нея, например в сравнение с оценяването на неопределеността й. Може би най-слабата страна на съществуващите приложения е оценката на неопределеността в осъществената локализация. За да може тази статистическа информация да бъде ползвана оптимално в хибридни навигационни системи, първо е нужно тя да бъде максимално точна, и второ да се изчислява максимално бързо и точно с ниски изчислителни ресурси. При кооперативната навигация прецизно позиционирания от изключителна важност и е нужно да се вземат предвид възможните грешки. Анализа на грешката при навигацията при наличие на неопределеност в позициите на роботите не е тривиален, а информацията за това е сравнително оскъдна в литературата. Най-сложната част от проблема на децентрализирано УДМР в условия на кооперативност и група от агенти е тя да бъде прецизна, бърза за изчисление и щадяща ресурси (батерия, процесорна мощ или памет). Тъй като УДМР зависи от конкретното взаимното положение на роботите в групата всеки един момент, то това състояние трябва да бъде следено и управлявано по време на изпълнение на общата за групата задача чрез добро планиране и обмен на сензорна информация – данни в реално време. Планирането на движения в група от роботи, с локализационни традиционни подходи, е сложна оптимизационна задача. В съществуващите приложения липсва подробна информация по решаване на проблема с традиционните алгоритми.

От една страна е нужно даден робот да бъде преместен по оптимизирана траектория, по която трябва да се избягват препятствия с отлична прецизност. От друга, докато извършва движението си, роботът трябва да предава данни от сензорите и да я споделя в реално време с други роботи за да се прецизира непрекъснато локализиране на всички в групата. Обикновено за подобни оптимизационна задачи се прилагат итеративни числени методи, оценяващи изходната неопределеност в множество точки. Това налага оценяването на неопределеността да е възможно най бързо, с цел да се ускорят въпросните оптимизационни алгоритми. Решаване на навигационната задача – най-кратък път от една до др. точка чрез НМХ покрива повечето от критериите, като бързина на изграждане на път, малка грешка/отклонение, добра устойчивост при много препятствия и използване на ограничени компютърни ресурси.

ВТОРА ГЛАВА. ХИБРИДНА МУЛТИ-АГЕНТНА СИСТЕМА ЗА ПЛАНИРАНЕ ТРАЕКТОРИИ ЗА ГРУПА МОБИЛНИ РОБОТИ

Във втората глава се анализира система за планиране на траекториите при група МР (роева роботика) с помощта на интелигентен подход - НМ „Хопфийлд“ „невронна

карта“. Описва архитектурата на основните функционални блокове. Извършва се задълбочен анализ на всеки един от разгледаните подходи по отделно. Прави се избор на подхода за изграждане на *системата за навигацията и планиране на траекторията на група мобилни роботи* (СНПТГМР), като основна за по-нататъшното изследване. Извършва се критичен анализ на предложения подход. Очакванията са, че реализацията на софтуерна част в дисертацията ще наложи използване на паралелни графични изчислителни мощности и технологии (GPU).

2.1 Определяне на критерии за оптималност при внедряването на УДНМР

При решаване на задачи от тип навигацията се изисква тя да е оптимална според някакъв критерий за оптималност, като най-краткото разстояние на пътуване и/или време. Високите изчислителни и представителни разходи и бавната реакция са типични в този случай. Когато става дума за критерии за избор, може да избират параметри измежду сходимост на процеса на самообучение на НМ, изчислителна достатъчност и т.н. с цел да постигнем желания резултат от внедряването на избрания модел. Т.е. има определена комплексност при определянето на критериите, които ще засегнат представения модел в частност.

Избора на сходимост например ще е условие за спиране на алгоритъм за оптимизация, където е разположена стабилна точка (в случай избрания път до целта) и е малко вероятно по-нататъшните итерации на алгоритъма да доведат до по-нататъшно негово подобрение.

Може да се измери и изследва сближаването на алгоритъм за оптимизация емпирично, като например използване на криви на обучение. Освен това би могло да се проучи конвергенцията на алгоритъм за оптимизация аналитично, като доказателство за конвергенция или средна изчислителна сложност на случая.

В рамките на решаването на проблема за планиране на траектории на движение за група МР може да разграничим следните възможни критерии, които определят ефективността на избрания алгоритъм и съответно подход за планиране и навигация:

- типа формация, която заема групата МР за постигането на целта – децентрализирана формация (всеки МР сам преценява как да стигне до целта) или във формация с цел оптимизиране част от калкулациите при изчисляването на пътя до целта и избягването на препятствия по пътя;
- максимална точност също е много важен критерий, зависещ от коренното приложението и вида на робота, е постигането на максимална точност. Висококачествено изграждане на траектория, т.е. избор на алгоритъмът, който дава възможност за формиране на траектории, близки до оптималните (най-къси или близки до най-късите в условията на дадената конфигурация на работната зона) но за сметка на повече ресурс необходим за тези изчисления. При въвеждане му също се получават по сложни задачи. Особено в условия сътрудничество и групиране между голям брой роботи, където трябва да се съобразяват и зависят един с друг при изпълнение на задачите си [39][40][41]. Тогава се налага включване на нови и по нетрадиционни подходи които да могат да извършат необходимите многобройни изчисления по-бързо и с по-малък наличен ресурс;
- умерено потребление на МР хардуерни ресурси – оптимизирано от гледна точка на енергийна консумация;
- лесна формализиране на задачата за планиране (т.е. простота математическо

представяне на конфигурацията в работната зона);
- скорост (скорост на изчисляване на траекторията) в сложна конфигурация на работното пространство (т.е. при наличие на голям броят на препятствията или препятствията със сложна форма);
- възможност за ефективно приложение за планиране на траекторията в условия на динамично променяща се конфигурация на работната зона (при наличие на подвижни препятствия, както и много други МР в близост).

Този списък се свежда до повечето класическите критерии, които се използват при навигация на МР:

- постигане на минимална консумирана енергия;
- минимално време за изпълнение – например за достигане до целта;
- минимална измината дистанция;
- други изисквания, приложими за всеки отделен конкретен проблем, който МР трябва да разреши.

Има голямо разнообразие от алгоритми, които позволяват решаването на проблема с намирането на път/траектория в съответствие с избраните по горе критерии. Въпреки че повечето от тези алгоритми са модификации на „основните“ методи за планиране на пътя, оптимизирани за конкретните условия [27] – хардуер, бързина, ефективност и т.н. Ето защо изборът на подход за внедряване за системата за планиране ще се определи от следната йерархия от критерии:

1 *минимизация на фалшивите положителни резултати – генериран път до целта;*
2 *изисквания за бързина на решаване на задачата за позициониране и съответно бързина на изчисленията за избраните оптимизационни алгоритми. Например, силният селективен натиск ще доведе до по-бърза сходимост и вероятно преждевременно сближаване. По-слабият натиск може да доведе до по-бавна сходимост (и съответно по-големи изчислителни разходи), въпреки че ще намери глобалния оптимум;*
3 *изчислителна достатъчност чрез асинхронна мултинишкова паралелизация и възможност на хардуерната база на МР;*
4 *гъвкавост и възможност за работа в условия на динамична среда – бързо променяща се конфигурация в работната зона и среда (размер, местоположение на препятствията, брой на МР, различни формирования/групи на МР);*
5 *енергийна интензивност (количеството енергия, изразходвана от мобилния агент спрямо планираните енергийни ресурси, с които разполага за изпълнение на задачата);*
6 *скорост на постигане на консенсус при взимането на решения между членове на формацията/екип (няма да се разгледа в изследването и се приема за равна на единица;*
7 *устойчивост на децентрализираната система от външна намеса;*
8 *надеждност към системата за навигация.*

Това са *критериите, които ще бъдат взети под внимание в изследването и конкретно при направените теоретични модели в Глава 2 и извършените експерименти в Глава 3* за да се удовлетвори желана оптимална траектория на групата МР в сложна среда.

Въз основа на същите тези критерии ще са и възможни следните типове **управление и придвижване** при изграждането на система за планиране на хибридна навигация [43]:

- централизирано управление;
- хибридно управление;
- децентрализирано управление.

- синхронно придвижване;
- асинхронно придвижване.

Самите МР също може да се придвижват синхронно или асинхронно независимо от това какво е тяхното управление – централизирано или не.

Нека се разгледат тези техники по-подробно по долу за да се представи потенциала на критериите в посока подобрение на алгоритмите за навигация.

2.2 Архитектура, формация и групиране на мобилни роботи

Много често типа управление (представен по горе) се свързва с това как МР се движат в работното пространство. Управлението на структурата - формацията на група мобилни роботи [104] представя как мобилните роботи се придвижват в пространството – синхронизирано или асинхронно. Примерно децентрализираната или т.н. асинхронна контролна структура е най-използваната структура позната при мулти-агентни системи и може да се разглежда като противоположна на централизирания подход. Така всеки агент/МР бива представен като отделна единица, която взима сама решението за това как да се придвижва в пространството, оптимизирайки движението си спрямо зададената цел. Такъв подход има приложение при по-големи системи, отколкото при една синхронна и централизирана структура. Но има риск до намалена стабилност и по-бавна конвергенция на модела за намиране на път до целта.

При подхода лидер-последовател [104], избрания лидер, генерира референтна траектория за останалата група роботи, а всички останали действат като последователи, които трябва да спазват определено относителното отстояние по отношение на лидера. Всъщност, след като се установи предложението за формация и фигура от лидера, желаното отделяне и желаното относително позициониране на последовател тази формация се следва чрез избиране на модел за управление на всеки последовател, основаващ се на неговата относителна позицията спрямо лидера. Така стабилността на формацията също е гарантирана, т.е. цялата група може да постигне и поддържа желаната формация през целия път. И тогава проблемът с навигацията на формацията може по същество да се разглежда като традиционен проблем за намиране на траектория до целта и да се разреши по тривиален начин.

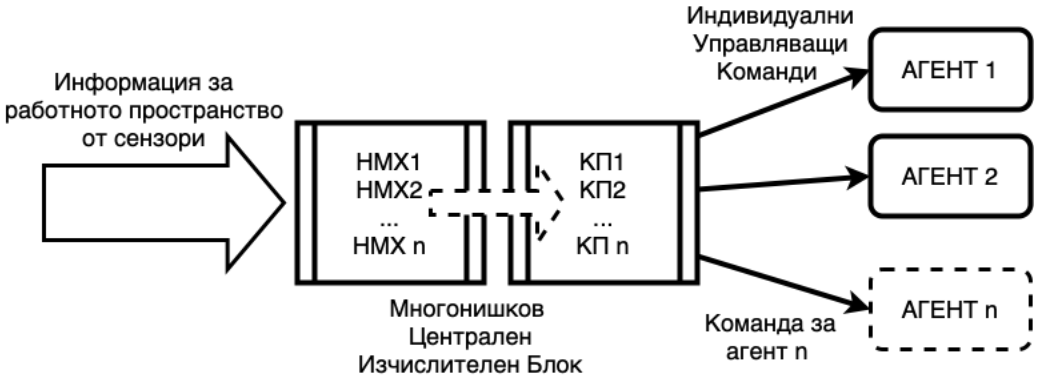
Роля на атрактивната потенциална функция в представения хибриден модел за навигация няма да бъде на фокус и изследването няма да обхваща координация между отделните роботи, които трябва да поддържат конкретна геометрична формация в координатна система (работно пространство). Координация и комуникация между МР по време на движението им ще липсва и колизии/инциденти ще се избягват от алгоритъма, който се представя в дисертацията. *Другия момент е, че ъгловата скорост, която може да е функция от атрактивната потенциална функция (Раздел 1.3.1) също я приемаме за фиксирана и не попада в полезрението на модела и направените експерименти.*

Изследването няма да обхваща сценария „лидер-последовател“, тъй като се търси решение за глобалната навигация за един МР - лидера без да се навлиза в детайли по управлението на формированията.

2.3 Централизиран подход при управление на навигацията

Изграждане на система за планиране за централизирания модел включва решаване на

проблема за групово позициониране от един център [43] и централизиран подход. По този начин всички изчислителни операции и системите за планиране се изпълняват от централен изчислителен блок (ЦИБ), който функционира в многонишков режим, и който комуникира индивидуално с всеки от агентите. Въз основа на наличната информация за конфигурацията на работното пространство ЦИБ извършва формиране на невронни карти за всеки от агентите, както и изчисляване на траектории, като се вземат предвид ъгъла, скоростите и разрешаването на възможни конфликтни ситуации. Трябва също да се отбележи, че изчислителните операции нужни за калкулация на НК и КП за всеки агент се извършват паралелно в отделни асинхронни нишки с достъп до общи софтуерни ресурси. Агентите от своя страна приемат само управляващи сигнали чрез канал за данни и в съответствие с тях задействат задвижващите механизми, тоест безусловно следват „инструкциите“ на центъра [AARTON2021].
Изпълнението на системата за планиране според централизирания подход е показано на Фигура 2-1.



Фигура 2-1. Блокова схема на централизирана система за планиране на траектория.

Основните предимства на това внедряване на системата за планиране включват:

- лекота на внедряване на хардуера;
- лекота на повишаване на производителността на цялата система чрез повишаване производителността на хардуер на ЦИБ;
- ниска цена;
- по-ниска енергийна интензивност при МР;
- висока производителност.

Разбира се също така се отбелязват и редица недостатъци като:

- слаба надеждност – при проблем с ЦИБ всички роботи напълно спират да функционират. Работата на системата може да се резервира но това би разрешило само възникнал хардуерен проблем;
- високи изисквания към изчислителните възможности на централния процесор на ЦИБ (с голямо вземане на проби от работното пространство, тоест с увеличаване на точността на движенията) и RAM (при голям брой агенти);
- и от тук невъзможност за динамична мащабируемост на системата;
- ограничен размер на общата зона за движение поради необходимостта от организиране на дистанционни комуникационни канали с ЦИБ и в резултат на това необходимостта от инсталиране на мощни енергоемки приемо-предавателни устройства на агентите;
- невъзможност за работа в неизвестна досега конфигурация на

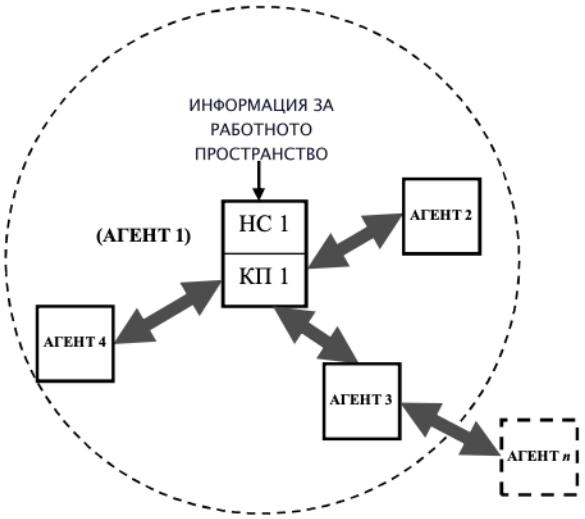
работната зона или при наличие на динамични препятствия, които биха претоварили системата;
- сложността на реализацията на софтуерната част, тъй като е необходимо да се използва технологията на многонишково програмиране, както и по-комплексно и сложно синхронизиране на изчислителни нишки за осигуряване на достъп до общите данни.

Макар и най-популярен, централизирания подход има нужда от информацията за средата в реално време и тя трябва да е достатъчна, за да могат действията на групата да бъдат планирани предварително. Това предполага добра комуникация в реално време и безпроблемна изградена инфраструктурна свързаност с ЦИБ. ЦИБ също трябва да е с подсигурена резервираност в хардуера и виртуализацията.
Основното предимство на централизирания подход е неговата лекота на използване и имплементиране на системата за групово управление на група МР.

2.4 Децентрализиран подход

Съществуват два вида децентрализирани структури: колективна и с поведение на ято. Децентрализираната стратегия предполага наличието на информационен обмен между участниците, който, от своя страна, също се нуждае от съответна правилна комуникация в реално време и сигурност.
Предимството на този подход е свързано с възможността за оптимизация на колективните действия на колаборацията. Но, въпреки сравнително високата ефективност на взаимодействието в групата, този подход изисква осигуряване на защита на информационния канал от външни взаимодействия, тъй като нарушаването на работата на канала за обмен на информация ще предизвика нарушаване на процеса на обмен на информация между всички участници.
От гледна точка на устойчивостта към откази стратегията на ятото, при която участниците не обменят информация помежду си, но могат да анализират състоянието на околната среда, на която влияят останалите участници, има предимство по отношение на устойчивостта пред колективната предвид факта, че при нея отсъства канал за обмен на информация. Ятото предполага обособено съществуване на всеки участник в системата, где отделните участници нямат възможности да обменят информация между себе си, но са способни да анализират измененията на състоянието на околната среда, на която влияят останалите участници от групата. Резултат от този анализ е решение за съответни действия на този участник, които трябва да доведат до достигане на общата цел посредством измененията на състоянието на този участник и влиянието на другите участници от групата на околната среда.
Освен това, липсата на такъв обмен подобрява общото време на реакция на всички участници. При този метод може да се организира обща „грижа“ за безопасността на агентите, при която всеки агент “се грижи” за другите агенти, отговаряйки за тяхната безопасност, следейки измененията на състоянието на системата. Моделът за управление на безопасността, който лежи в основата на този метод, бива наричан Police Office Model (POM). В POM всички агенти имат еднаква структура, но могат да притежават различни функционални възможности. Всеки агент се състои от две части: основна и допълнителна. Разликата между тях е в това, че основната част е достъпна само на агента – собственик, докато допълнителната е открита за всички. Това дава възможност непрекъснато да се проследява състоянието на всеки от агентите и в случай на нарушение на техническото състояние да се информира пункта за управление, след което информацията, се предава от повредените устройства се класифицира като несигурна и ще се игнорира от другите устройства.

При стратегията на ятото, въз основа на всички получени данни i-тият агент изчислява целевите функции за всички възможни решения за дадения момент и от наличния вектор на решенията (възможно получен в резултат на обучение или самообучение) избира действие с най-голяма стойност.
При внедряването на представената система за планиране с децентрализирана схема всеки агент има собствена система за планиране на траекторията с изчислителен блок НМ и КП на борда. Ето защо може да се класифицирана като *самоорганизираща се – тип „рояк“ или „ято“,* което се определя от по-гъвкавите й реакции към външни влияния, разширени възможности за адаптация и натрупване на опит.
По време на изпълнение на задачата за навигация и позициониране всеки агент, използвайки информационно взаимодействие по безжични комуникационни канали с други (най-близки) членове на групата или координатора, самостоятелно коригира траекторията си и разрешава възникващи конфликтни ситуации [AARTON2019]. Този подход до голяма степен е изграден на базата на мулти-агентен метод за групов контрол, чиито характеристики бяха описани в първа глава (раздел 1.1.3) [73].
Изпълнението на децентрализирана система за планиране на траекторията е показано на Фигура 2-2.



Фигура 2-2. Структурата при децентрализирана система за планиране на траектория базирана на НМХ.

Основните предимства на децентрализирания подход за внедряване на системата за планиране на траекторията за групата МР са:

- безотказна работа и надеждност – отказ на един агент (или няколко) не оказва влияние върху решаването на проблема с позиционирането на други агенти;
- възможност за работа в неизвестна досега конфигурация на работното пространство;
- възможност за решаване на проблеми с позиционирането в работната зона от всякакъв размер.

  Въпреки очевидните предимства трябва да се отбележи, че има и някои недостатъци:
- необходимост от въвеждане на алгоритъм за консенсусен контролер с памет (база данни), за да е сигурно, че винаги ще има непрекъснатост и сигурност на процеса на комуникация и съхранение на данните (примерно за постигане на общо мнение по въпроса за режима, в който ще се движат МР или при преноса и управление на всички одометричните данни);
- ограничени изчислителни ресурси на агента поради особено строги

изисквания за енергийна ефективност на мобилните бордови системи и в резултат на това по-ниска производителност;
- сложност и високата цена на внедряването на хардуерната платформа на агента.

2.5 Хибриден подход за локална и глобална навигация

Както посочихме в началото един робот, който се движи в реално време се нуждае от обработка на данните постъпващи от сензорите му за да планира движението си - *локално навигация*. При *глобална навигация* се изгражда карта на цялото работно пространство без да се знае ситуацията на място. Така, при прилагане на хибридния подход системата за планиране би се реализирала с един общ ЦИБ, който комуникира с локалните модули КП и НМХ (невронна карта) на всеки отделен агент. Всеки МР ще играе роля на разпределен ресурс с негова локална навигация отговорна за навигацията в групата и поддържане на формацията. В тази конфигурация всеки МР е оборудван с *локална бордова изчислителна единица* и формира контрола на оформената група/екип или формирование. *Активния лидер* е онзи, който взима решение, комуникира с останалите лидери и върху него се прилага глобалната навигация (или от локания изчислителен блок приоритетно или от ЦИБ). Останалите от формированието агенти следват лидера чрез локален изчислителен блок/модул (ЛИМ или ЛИБ), който се грижи за локалната навигация в групата (следването на лидера). Сензорна информация се приема от отделният агент, който стартира процеса на изчисляване на траекторията и създава невронна карта. Дисертацията само разглежда алгоритъма за придвижване на индивидуалния агент-лидер. При реални (не лабораторни условия) стъпките са както следва:

А/ Избрания лидер във така сформираната група от агенти създава НК и изпълнява локалния КП на база сензорната информация постъпила в НМХ.

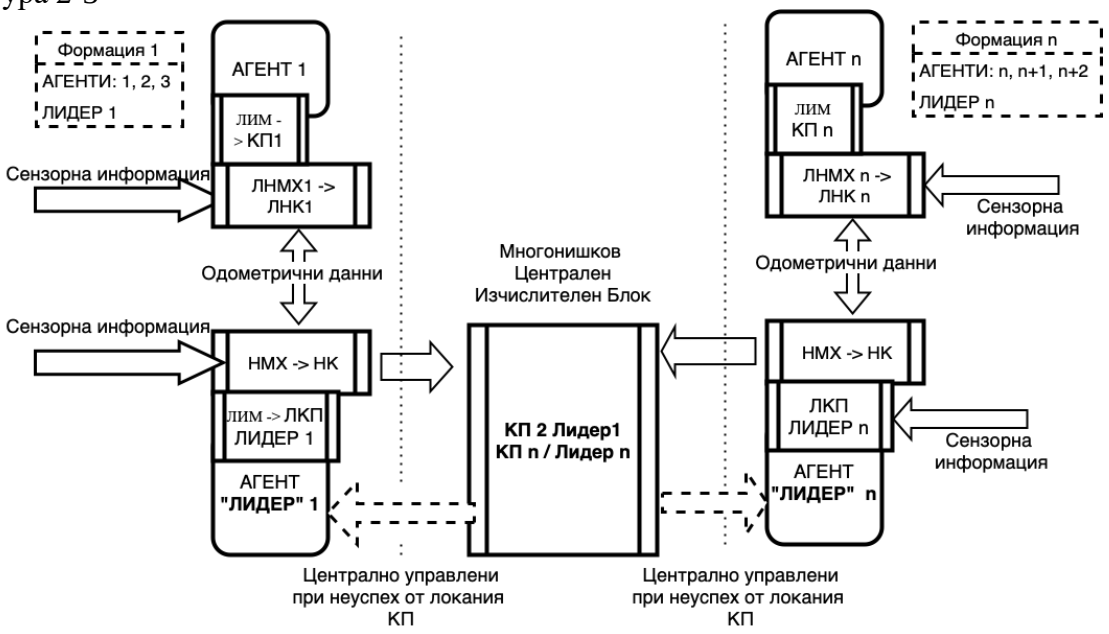
Б/ Локалната навигация се изпълнява приоритетно и сформираната група обменя одометрични данни с лидера, като по този начин всички уточняват своите координати и позиции спрямо останалите във формированието. Данните се подават на ЛНМ и се генерират НК. Следва процеса на КП във всеки индивидуален агент от групата за изчисляване на следващата му стъпка;

В/ Управляващите команди на ЦИБ, които пристига при лидерите не подлежат на задължително изпълнение, тъй като в процеса на движение в конкретната група да е в автономните режима на работа - или да коригира или формира собствена нова невронна карта в зависимост от новата информация за работното пространство или обект, идваща от сензорите. Също така, в случай на откриване на препятствия или непознат обект за обследване от сензорна информация за работното пространство, агентът-лидер съвместно с останалите от екипа да вземат самостоятелно решение за преминаване в режим на „обследване на препятствието“ или към нов режим „избягване на внезапен сблъсък“ за да се отдалечат от препятствието и съответно от крайната цел с условието за връщане към целевата траектория след извършване на тази т.н. „маневра“.

Г/ Ако това условие не е изпълнено и агентът не е в състояние да продължи да се движи по текущата траектория, тогава лидера предава последните актуализирани данни (текущи координати и карта) на централния изчислителен блок (ЦИБ), който преизчислява траекторията но вече като взема предвид информация от други МР и цялостната ситуация в работното пространство. ЦИБ връща следващата стъпка изчислена от общия централен конструктор на пътя обратно към агента.

Във сравнение с централизирания подход, този принцип на внедряване на системата за планиране, предлага положителни характеристики в допълнение като *висока енергийна ефективност на агентите, по-добра мащабируемост и хардуерната производителност на системата като цяло само с добавяне на още агенти към формированието.*

Принципът на изпълнение на система за планиране по хибридна схема е показан на Фигура 2-3



Фигура 2-3. Блокова схема на хибридна система за планиране на локална и глобална траектория с НМХ.

Следните основни предимства предопределят хибридната система:

- способност за работа в неизвестна досега конфигурация на работната зона или при наличие на динамични препятствия;
- Значително намаляване на изискванията за производителност на хардуерното помещение части от системата с увеличаване на броя на агентите и увеличаване на дискретизацията на работната зона, поради разделянето на ресурса за изграждането на НМ между агентите.

  Въпреки това остават и редица недостатъци, присъщи на системите с признаци на централизация:
- ниска надеждност (възможно е частично решаване на проблема с помощта на допълнителен блок за математически аритметики - GPU);
- ограничен размер на общата зона за движение поради необходимостта от организиране на дистанционни комуникационни канали с ЦИБ и в резултат на това необходимостта от инсталиране на мощни и енергоемки приемо-предавателни устройства на агентите;
- по-сложен хардуерен дизайн на членовете на групата поради наличието на изчислителен блок за НМ, както и необходимостта от инсталиране на допълнителни бордови сензорни устройства;
- сложността на реализацията на софтуерната част, тъй като е необходимо да се използва технологията на многонишково програмиране и синхронизиране на изчислителните нишки за обработка на данни от агентите в реално време.

2.6 Избор на основен подход за внедряване на система за навигация

Като се вземат предвид посочените критерии (Раздел 2.9.1), в рамките на изследователската работа ще се спрем на хибридна схема като основен подход при изграждане на система за планиране за тестване на алгоритъма за управление на навигацията на група МР. Смятаме, че хибридната схема би могла да реши основните проблемни областни като сигурност и възможност за разширяване на хардуерния капацитет в реално време докато трае целия процес.
На основата на тях ще бъдат направени експерименти в Глава 3.
Изборът ни върху този подход се дължи на редица ключови предимства, които ще се представим:

- лекота на разширяване на хардуерните ресурси на цялата система в широк обхват чрез увеличаване на изчислителните възможности на ЦИБ (всъщност конвенционален компютър, който може да бъде виртуализиран);
- минимални изисквания към хардуерните възможности на агентите, което значително намалява цената на системата и опростява отстраняването на грешки в софтуерната част на системата за планиране. Не е необходимо да се използват високоскоростни бордови компютри, комуникационни съоръжения и подходяща система за захранване (въпросът за икономическата осъществимост и хардуерните възможности на агентите се увеличава с увеличаване на броя на агентите в групата, както и с намаляване на размера на самите агенти);
- бързина и надеждност на трансфера на данни между потоци поради споделен достъп до общите хардуерни ресурси на ЦИБ, което дава възможност за експериментално тестване на алгоритъма за планиране на траекторията и логиката за разрешаване на конфликти в групата, елиминирайки възможни грешки при трансфера на данни по време на информация обмен между агенти.

Осигуряването на споделен достъп до общи хардуерни ресурси в асинхронен режим е ключова характеристика на системите за планиране, базирани на централизирани и хибридни схеми за управление. Тази функция се осигурява от използването на паралелни изчислителни технологии при реализацията на софтуерната част на системата за планиране.
Също така си струва да се отбележи, че въз основа на анализа на съществуващите методи за управление на групата (раздел 1.1.3), децентрализираните и мулти-агентни подходи са най-обещаващите.
При внедряване на софтуерната част на хибридна система за планиране предопределя също така да се използват паралелни изчислителни технологии. Паралелизацията при всеки агент осигурява споделен асинхронен достъп за:

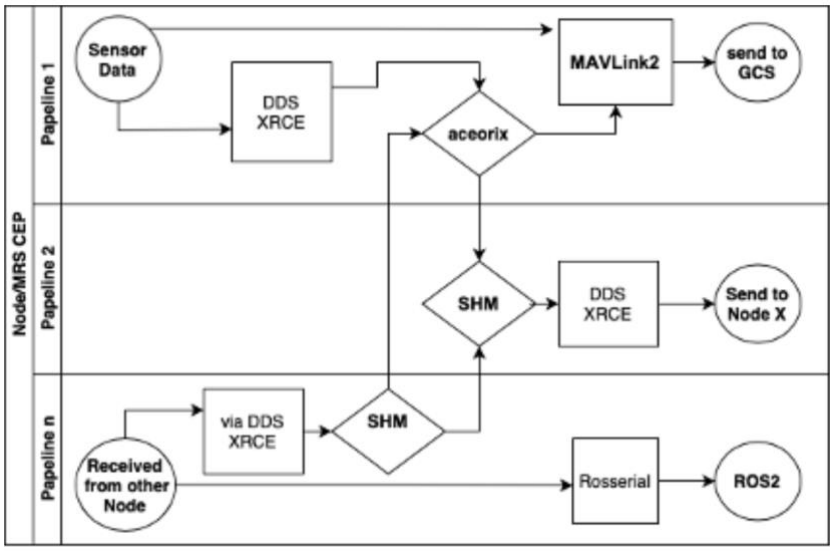
- канали за предаване на данни от други агенти в рамките на неговата бордова система [AARTON2021];
- изпълнение на изчислителните процеси свързани с алгоритмите.

Нека по-нататък да се разгледат особеностите на прилагането на паралелна изчислителна технология, като вземем предвид спецификата на обекта - екип от роботи. Както видяхме по-горе децентрализирана система ще има нужда от внедряване на *консенсусен алгоритъм (контролер)*, който осигурява целостта на системата от агенти) веднъж и втори път за потвърждение и постигане на консенсус в мнението на участниците, когато има формацията и когато се налага преминаване от един режим към друг. Това допълнително усложнява системата и поради тази причина консенсусен алгоритъм (контролер) няма да може да бъде разгледан в тази дисертация макар и да съществува като съставен компонент в използваната база данни. Също така основен

фокус в дисертационната работа ще бъде разглеждането на *глобалните навигационни техники* без да се навлиза в дълбочина и преглед на локалната и местна навигация във сформираната група от мобилни роботи.

2.7 Прилагане на паралелни изчислителни и комуникационни технологии при внедряване на СУНМР

Споменавайки за паралелни изчисления, организирани в съвременните операционни системи, е необходимо да се определи концепцията за процес. Процесът е съвкупност от ресурси, необходими за изпълнение на командите на определена програма. Ресурсите могат да се разбират например като виртуална памет, I/O манипулатори, област на паметта за изпълнявана програма, манипулатори на сигнали и т.н. По този начин процесът е блок от изпълними команди, който има отделно адресно пространство [84]. МР трябва в допълнение да поддържат едновременна комуникация в реално време с различни софтуерни приложения на бордовия. компютър или извън конкретния него със заобикалящите го агенти - със сензорна, или други видове информация. В дисертацията само ще маркираме възможността да се използва паралелна архитектура на процеси за SALP [AARTON2021] изпълнявани на различните МР без да се впускаме в детайли. Блоковата схема на примерна архитектурата (фигура 2-4) показва управлявана от събития хибридна комуникация с протоколи за комуникация като MAVLink, DDS-XRCE през архитектура PC SHM и rosserial. Тази схема е основа за внедряването на *нишкова комуникационна инфраструктура* при децентрализираните системи и конкретно в текущото проучване се ползва от ROS [47] .



Фигура 2-4. Блокова схема на SALP в паралелна архитектура [AARTON2021].

2.7.1 Въвеждане използване на паралелна архитектура за изчисления на алгоритъма НМХ.

Съвременният хардуер поддържа паралелни изчисления, т.е. едновременно изпълнение на няколко процеса. В същото време паралелизмът на ниво инструкции определя производителността на процесора [84]. Независимо от това, въпреки общата хардуерна поддръжка за паралелни изчисления, софтуерната паралелна обработка на данни може да бъде реализирана по различни начини и избраният метод за изпълнение ще зависи от

специфичните нужди, по-специално:
1/ Необходимостта от изпълнение на няколко задачи (процеси) за по-малко време, тоест повишена производителност на цялата система. Тук повишаването на производителността се постига от факта, че всеки процесор (ядро на процесора) е ангажиран с обработката на отделна задача. В този случай разпределението на процесите се извършва с помощта на планировчик на задачи (обикновено вече вграден в ОС или дори в микро архитектурата на процесора), който разпределя натоварването между процесорните ядра (както физически, така и логически) [100, 101].
2/ Необходимостта да се изпълни една задача за по-малко време. В този контекст използването на паралелизъм в рамките на една задача, за да се завърши за по-малко време, ще се нарича "паралелно (многонишково) програмиране" [100, 102].
Въз основа на горните нужди могат да се разграничат два основни подхода към софтуерната реализация на изчислителния паралелизъм: многопроцесорност и многонишковост [100,102], които ще бъдат демонстрирани в програмния код използван в Глава 3. Несъмнено необходимостта от избор на метод за осигуряване на процес на паралелизъм ще възникне при внедряване на разглежданата система за планиране на навигацията (особено при избор хибридна схема на изграждане) и кой подход ще бъде уместен до голяма степен зависи от сложността на изчислителната задача, възложена на централната банка и на хардуера.
Многонишковият подход е подходящ, ако:
– изчисленията се извършват на еднопроцесорна и едноядрена хардуерна платформа или на многоядрена платформа, но с ограничени възможности за консумация на енергия и консумация на RAM (т.е. бордова компютърна система MP);
- ниска дискретност на работното пространство с малък брой агенти, когато задачата за позициониране може да бъде представена като общ процес на цялата система с вътрешна многонишкова обработка, където обменът на данни между нишките може да се организира с помощта на общи области на паметта и глобални променливи, както е посочено на Фигура 2-4 по горе.
Многопроцесният подход е уместен, ако:
– изчисленията се извършват на многопроцесорна или многоядрена хардуерна платформа без строги ограничения за консумацията на енергийни ресурси и RAM;
– има висока дискретност на работното пространство или голям размер на работното пространство. В този случай подзадачата за обработка на конвейера на данни „невронна мрежа → конструктор на път” за всеки агент може да се разглежда като отделен процес. В този случай обменът на данни между процесите най-вероятно ще бъде организиран с помощта на интерфейси за мрежово програмиране чрез т.н. sockets - „сокети“ или „събития“ управлявани в реално време в отделни многонишкови процеси [AARTON2021]. Софтуерните гнезда могат да осигурят обмен на данни чрез мрежови протоколи (TCP, UDP, т.е. на транспортно ниво), което също така дава възможност за паралелизиране на процеси не само в рамките на един компютър, но и между машини, тоест бордови компютърни системи на отделни роботи (в група) и по този начин прилагат методи за клъстериране за изчислителни процеси в рамките на една разпределена система [AARTON2021].
Нишката е отделен поток на изпълнение. Това означава, че програмата ще изпълнява две задачи, които ще се изпълняват наведнъж. Но за повечето реализации в Python 3 различните нишки всъщност не се изпълняват по едно и също време - те само изглеждат за външния наблюдател като такива.
Може да приемаме че нишка, която е подкрепена хардуерно с два (или повече) различни процесора, изпълняваща програма, всеки от които изпълнява независима задача по едно

и също време. Нишките може да се изпълняват на различни процесори, но те ще работят само един след един – което не е най-подходящо при изпълнение на задачи в реално време.
Едновременното изпълнение на множество задачи изисква нестандартна реализация при програмиране с езика Python, който ще се използва и се решава с допълнителни подходи както ще се разгледат по-долу в Глава 3.
Задачи, които прекарват голяма част от времето си в чакане на външни събития, обикновено са добри за имплементиране с „нишки“. Проблеми, които изискват тежки изчисления на процесора и прекарват малко време в чакане на външни събития, може изобщо да не работят по-бързо. Това важи за код, написан на Python и работещ на стандартната реализация на CPython. Ако нишките са написани на езика „C“, те имат способността да се изпълняват едновременно. Ако се използва друга реализация на Python, трябва да се прецени дали и как тя се справя с нишките.
Ако се използва стандартна реализация на Python и се пише само на Python трябва да се разчита на модула за многопроцесорна обработка.
Въпреки очевидните на пръв поглед предимства на многопроцесния подход при организиране на паралелни изчисления, той има и своите трудности при. реализацията. Основният проблем е свързан с факта, че преместването на процес в паметта на компютъра или последователната обработка на процеси изисква значителни изчислителни ресурси на системата [100]. Също така, за обмен на данни между процесите е необходимо да се разработят допълнителни софтуерни интерфейси и протоколи за комуникация между приложенията в отделните агенти [AARTON2021].
Във връзка с горните трудности при организирането на многопроцесни изчисления е много по-лесно да се работи с данни, които са обекти в рамките на общ процес, които работят в неговата среда, в общо адресно пространство, тоест нишки. Всяка нишка има своя собствена област на паметта, разпределена от адресното пространство на процеса. По този начин превключването между нишки и преместването на нишки в паметта е по-лесно и по-бързо. Също така, моделът на многонишкова паралелизация го прави много по-ефективен за обмен на данни между нишки чрез използване на общо адресно пространство на паметта без нужда от допълнителни интерфейси [AARTON2021] - фигура 2-4.
При работа с асинхронни потоци [102] трябва да се вземат предвид и следните характеристики на използването на софтуерни и хардуерни ресурси:

- всяка нишка има нужда от място за работа с данни, така че всяка нишка заема RAM, дори ако тази памет не се използва и нишката „спи“;
- ако има „изтичане“ на памет в една от нишките, тогава целият изчислителен процес ще стане излишен и следователно ще бъде унищожен (например услуги за балансиране на натоварването, намиращи се във всички съвременни операционни системи) заедно с други нишки;
- ако две нишки използват един и същ ограничен ресурс, може да възникне „състезание“ на нишки за ресурса (състояние на състезание). За борба с това явление се използват синхронизатори: флагове, които позволяват текущия достъп до ресурса на една от нишките да блокират достъпа до останалите. Въпреки това, ако нишките не са правилно синхронизирани, може да възникне застой на нишки - ситуация, при която нишка A чака нишка B, а B чака A.

2.7.2 Програмна реализация на паралелни изчислителни процеси

В повечето случаи целта на прилагането на технологиите за паралелно изчисление е увеличаване на ефективността и производителността на системата. В общ случай за

оценка на ефективността на паралелната програма се изчислява като прираст на ефективността или ускорение. Това е времето, извършено за решението на задачата с помощта на последователното изпълнение на операции - T_s,, отнесено към времето за изпълнение на решението с помощта паралелни изчисления – T_p [100, 101]:

$$S = \frac{T_s}{T_p}. \quad (2.1)$$

По този начин с помощта на паралелизация на програмния код и увеличаването на количеството изчислителна мощ (количества на процесори и/или ядра) е възможно намаляване на времето на решаване на задачата в пъти. В този случай, „идеалната“ отложена паралелна програма при решаване на задачите на ръст на ефективността, която ще бъде директно пропорционална на количеството процесори (p). В идеалния случай ще има линеен прираст на производителност, а величината S и p ще бъдат равни. Така е, например, при изпълнението на програмата на 2-х процесорната система, с която се получава 2-х кратен прираст. Но такава линейна зависимост в реалните системи е малко вероятна, така че е необходимо време за обработка на служебната информация, която снижава общата производителност на системи [101]. Така, най-важният фактор, който влияе отрицателно на прираста при паралелизация, е част от софтуерния код, изпълняващ се само в режим на последователност. Правило, което определя това влияние, се нарича закономерност на „Амдал“. Закон на „Амдал“ определя максимален теоретичен ръст на програмните решения при използването на паралелни апаратно-програмни средства, сравнено по отношение на аналогично програмно решение, изпълняващо се само последователно [100, 101].

Примерно при задача за изпълнение на решение, което се изисква за време „T“. Общият обем на решаващата задача може да се представи във вид на величина f = 1, където при последователни изчисления ще бъде равна f_s, а при паралелни съответно $f_p = 1 - f_s$. Пример за последователни изчисления се явява например обработката на служебните данни при входно/изходните операциите, стартът на програмата, а също и обработката на резултатите от паралелните изчисления. При увеличеното количество процесори в системата се съкращава време само за паралелен изчисление, при това общо време T_p изпълнение на програмата за изчисление с количеството процеси/ядра (p) ще бъде определено по-нататък [101, 102, 103]:

$$T_p = \left(f_s + \frac{f_p}{p}\right) T_s = \left(f_s + \frac{1 - f_s}{p}\right) T_s$$

Тогава, в съответствие с (2.1), се извежда:

$$S = \frac{T_s}{\left(f_s + \frac{1 - f_s}{p}\right) T_s} = \frac{1}{\left(f_s + \frac{1 - f_s}{p}\right)}$$

Ако количеството на процесите p ще се стреми към безкрайност, то:

$$S = \frac{1}{f_s}, \quad (2.2)$$

За да има растеж на ефективността при растежа на паралелизма, ще се ограничи последователната част от алгоритъма, например: ако кодът е паралелен на 80% (последователна част 20%, съответно), това е максималният ръст на ефективността, в

съответствие с (2.2), ще бъде S = 1/0,2 = 5 при безкрайно голям брой процесори. Ако в системата присъстват два процеса, максималният ръст ще бъде:

$$S = \frac{1}{\left(f_s + \frac{1 - f_s}{p}\right) T_s} = \frac{1}{\left(0{,}2 + \frac{1 - 0{,}2}{2}\right)} \approx 1{,}67,$$

Тук също струва, че полученото значение – това е само теоретична величина, т.н. закон на „Амдал“ не отчита издържащите, възникващи в системата при управлението на разпределените потоци, изчислен (синхронизация, комуникация и т.н.) [95].
По този начин можете да се направят и следните изводи:

- в независимост от принципа на реализацията на системата за планиране е необходимо прилагането на технологиите, паралелизацията на изчислението, така както за постигането на коректни и ефективни решения на позиционирането на групата е необходимо асинхронен достъп до данните, съдържащи информация за други агенти, а също така е необходимо да се осигури обработка на сензорна информация и формиране на управляващи сигнали в асинхронен режим;
- избор на методика за паралелизация на изчисления определя обема на задачите за позициониране, решаваната група (размер и конфигурация на работните пространства, наличие на динамични промени и т.н.), а също и количеството участници в групата;
- при паралелизацията на изчислението е необходимо максимално съкратена част на алгоритъма, допускайки прилагането на само последователни изчисления, за да се получи максимален прираст на професионалната ефективност при увеличените количества процесори на апаратната платформа ЦБ (при хибриден и централизиран подход) или на бордовия хардуер на агента (при реализацията на децентрализираната на навигационна система).

2.7.3 Разпределен консенсус и съхранение на данните при УДМР

При изграждането на една разпределена и децентрализирана система трябва да се предвиди да е устойчива на грешки и прекъсвания. Тоест, ако един конкретен възел в създадената мрежа отпадне или дял, това да не повалява на коректната работа. Клъстерът от възли, участващи в разпределения протокол за консенсус, трябва да постигне съгласие по отношение на стойностите и след като това решение бъде постигнато, този избор е окончателен.
Алгоритмите за разпределен консенсус често приемат формата на репликирана „държавна машина“ и журнал. Всяка „държавна машина“ приема входове от своя дневник и представлява стойността(ите), която(и) трябва да бъде репликирана, например под формата на „хеш“ таблица. Те позволяват на колекция от машини да работи като съгласувана група, която може да оцелее при неуспехите на някои от членовете си.
Консенсусът е основен проблем в устойчивите на грешки разпределени системи. Консенсусът включва множество агенти - нодове, които се договарят за стойностите. След като вземат решение за стойност, това решение е окончателно. Типичните консенсусни алгоритми напредват, когато по-голямата част от техните сървъри е достъпна; например, клъстер от 5 „нода“ може да продължи да работи, дори ако 2 от тях се повредят. Ако повече агенти (в разглеждания в дисертацията сценарий) се провалят, те спират да напредват (но никога няма да върнат неправилен резултат).
В дисертацията пример за така хеш таблица ще се явява дневникът, който включва команди за запис в базата данни. Използва се алгоритъм за консенсус, за да се договорят командите и самите данни във файлове на сървърите. Алгоритъмът за консенсус трябва

да гарантира, че ако някоя държавна машина приложи set x на 3 като n-та команда, никоя друга щатна машина никога няма да приложи различна n-та команда. В резултат на това всяка държавна машина обработва една и съща серия от команди и по този начин произвежда една и съща серия от резултати и достига до една и съща серия от състояния. Два добре известни алгоритъма за разпределен консенсус са Paxos и Raft. Paxos се използва в системи като Chubby от Google, докато Raft обикновено се разглежда като по-разбираем и по-прост за изпълнение от Paxos и за това може често да бъде използван. В напия случай всеки МР със своите сензорните и навигационни данни ще се налага да ги съхранява и предава. Данните са от типа ключ – стойност (примерно координатната система със стойности за „x“ и съответно „y“) и лесно може да интегрира база данни с консенсусен алгоритъм. Не по-различно е предаването и съхранението на данни свързани със състоянието на невронната карта. Всеки неврон има своите координати в работното пространство и съответно стойност двоична стойност. Така създадена картата трябва да се пренесе между агентите или до/от ЦБУ.

Децентрализирания клъстер Raft има възможност да изпълни режим на база за съхранение от *типа „ключ – стойност“ (КС)*. За целта ще се използва библиотеката „RocksDB“ в “Python”, която предоставя постоянно съхранение на стойността на ключовете, където ключовете и стойностите са произволни байтови масиви. Ключовете се подреждат в хранилището за стойности на ключове в съответствие с определена от потребителя функция за сравнение в базата „RocksDB“ (https://github.com/facebook/rocksdb). Тази база се поддържа от инженерния екип на Facebook Database Engineering и е заимствана от „LevelDB“ с отворен код, създадена от Санджай Гемават и Джеф Дийн (Google). „RocksDB“ е основния градивен елемент за създаването на бърз сървър за ключ-стойност, особено подходящ за съхраняване на данни на флаш устройства. Има възможност за многонишкови процеси, което го прави особено подходящ за съхраняване на множество терабайти данни в една база данни.

Паралелно “CockroachDB” (https://github.com/cockroachdb/cockroach), ще може да се използва като SQL база данни, която съхранява записи в журнала/дневника на “Raft” и използва същата сървърна система за съхранение на данни – “RocksDB”. “CockroachDB” е разпределена SQL база данни, изградена върху транзакционно и силно съгласувано хранилище за ключ-стойност. Мащабира се хоризонтално. Удобството е, че на високо ниво “CockroachDB” преобразува SQL заявките на клиентите в данни за ключ-стойност (КС), които се разпределят между възлите и се записват на диск. и предоставя познат SQL API за структуриране, манипулиране и заявки за данни.

2.8 Моделиране на НМХ

Общия проблем за достигане до целта (т.н. *хибридна навигация* или т.н. *многослойно планиране*) при група МР се демонстрира практически в Глава 3 **без да отчитаме специфичността на двата компонента** (навигационни под задачи):

- *глобална навигация* – изграждане на невронна карта за придвижването на групата МР в работното пространство до достигането на целта;
- *локална навигация* – сформиране на „екип“ и групиране МР и неговото придвижване във формирование.

Поради необходимостта от представяне на много детайли свързани с локалната навигация в тази дисертация само ще набележим основните й компоненти без да навлизаме в детайли. Така като няма да може да навлезем в детайли в консенсусния алгоритъм и комуникационните техники при децентрализираните системи (Фиг. 2-4).

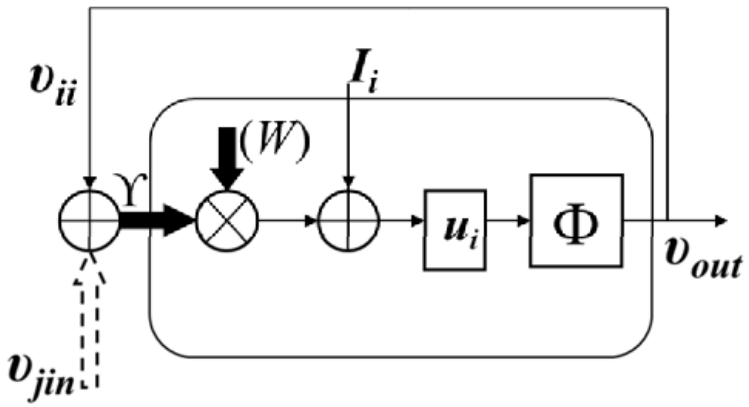
В трета глава ще представим различни експерименти и ще визуализираме графически математическия модел НМХ - т.н. „невронна карта“ (НК) докато в Глава 2 се изследват

невродинамиката на НМХ за изграждане на „невронна карта“ за глобална навигация до целта, като част от представената ни система за навигация в дисертацията. Тества се стабилността на посочените модели в симулационна среда като са представени резултатите от математическото моделиране, потвърждаващи ефективността на оптимизирана невронна карта за *автономна навигация на група мобилни роботи*. Въз основа на резултатите, разглежданите модели са оптимизирани, което дава възможност представеения подход да се използва за конструиране на неконфликтни траектории за МР движещи се в динамична и променяща се среда.
За експериментиране на различните нива и визуализиране на резултатите в Глава 2 и 3 се използва специално разработен програмен код на програмния език „Python“ в среда за програмиране „PyCharm”, като част от кода е приложен в *Приложение А* към дисертацията.

2.8.1 Структура и изследване на невронна карта на основата на НМХ

Както вече беше споменато в глава I (раздел 1.4.1), основната идея при метода за планиране на СПТ с невронна карта е динамично представяне на двуизмерно дискретно работно пространство. Така невронната мрежа, използвана за решаване на проблема с планирането, ще има определена топология, а броят на невроните в мрежата като матрица ще бъде равен на броя на дискретните клетки на самото работното пространство [60].
Нека има някакъв външен източник на информация (локална сензорна система на МР), на базата на който е възможно да се определи местоположението на МР, местоположението на целта и препятствията в рамките на дадена експеримент. След това, използвайки тази информация, е възможно да се формира вектор от входни сигнали за тази НМ. Ако някоя от входните стойности е различна от нула, тогава процесът на активиране започва в мрежата чрез директни и обратни връзки сигналът се разпространява в цялата НМ, докато се достигне състоянието на стабилност според критерия за стабилност на енергията - Ляпунов, [61]. Т.е. оптимизирането на решението се извършва на базата на минимизиране на енергийната функция на Ляпунов, конструирана за синтезираната НМ. *Ако входния вектор на НМ във вид на матрица в състояние на стабилност съвпада по размер с топологичното представяне на дискретното работно пространство (т.е. броят на матричните елементи съвпада с броя на дискретните клетки), следва да се приеме, че получената матрица е „невронна карта“, която може да се използва в бъдеще за изчисляване на траекторията до дадена цел.*
Разглеждана като част от системата за навигация НМХ, като *изчислителен процес се състои от следните параметри* (фигура 2-5)*:*

- неврон “i” - характеризира с входния вектор “Y”;
- собствен сигнал за обратна връзка v_{ii} на свързаните неврони;
- входните сигнали – v_{jin};
- “n” е размерността на входа и тегловите вектори;
- матрица с тегловните коефициенти Wi;
- обобщен нетен (мрежов) сигнал u_i;
- активираща функция Ф;
- матрица с отмествания (bias) Ii;
- състояние на собствен изходен сигнал на НМ - изхода v_{out}

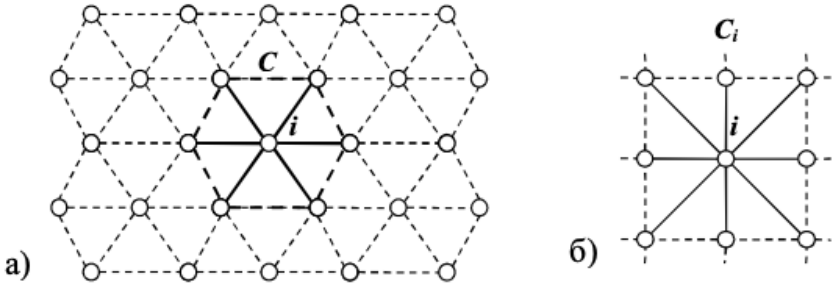


Фигура 2-5. Основният нелинеен процесор или неврон на Хопфийлд.

Обратната връзка преставляваща v_{ii} и входните сигнали v_{jin} на невроните, образуват входния вектор на сигналите Y, който от своя страна се претегля от матрицата с тегловните коефициенти Wi сумирани с отместване (bias) - Ii, образуват скаларен изход u_i (2-5). Той преминава през нелинейната активираща функция Ф [73, 46]:

$$u_i = \text{Wi} * \text{Y(t)} + \text{Ii(t)} \qquad (2.1)$$

Трябва да се отбележи, че векторът на тежестта "W" не би трябвало да е във функция на времето. Функцията за активиране "Ф" определя праговите стойности на невронните сигнали и следователно естеството на енергийното взаимодействие между всички неврони. Така тази функция до голяма степен определя набор от възможни мрежови състояния и съответно конвергенция [73, 46]. Ако се разглежда плоско - двумерно работно пространство „C", равномерно разделено на „n" дискретни клетки, тогава съответното НМХ също ще се състои от „n" неврони, равномерно разпределени (образно) върху центровете на дискретните клетки на това пространство. Всеки неврон ще има връзки (както директни, така и обратни) само с неврони от съседни клетки. Така мрежата ще бъде подредена структура под формата на "решетка". В зависимост от зададената геометрична форма на дискретните клетки, НМ ще има различна топология [73] (Фигура 2-6).



Фигура 2-6. Възможни мрежови топологии за разпределяне на двумерно пространство Ci: а) шестоъгълна; б) ортогонална/

Както може да се види от фигурата по горе, всеки неврон „i" взаимодейства само със своите съседи в своето подмножество („домейн") – „Ci" - общо 9 неврона. Тази функция осигурява „вълновата" природа на разпространението на сигнала в мрежата (невродинамиката на разглеждания НМХ ще бъде разгледана по-подробно в трета глава).

В рамките на тази изследователска работа ще приемем ортогоналната дискретизация и разпределение на работното пространство като база (Фигура 2-6). По този начин

разглежданият модел НМХ също ще има ортогонална топология, при която всеки „домейн“ ще се състои от девет неврона: централният и съответно 8 съседни (за осемте възможни посоки на движение) [73, 49].
Обучението на НМХ принадлежи към класа на самоорганизиращи се мрежи, за които се прилага методологията на обучение без супервайзор. Тази техника се основава на така нареченото правило на „Хеб“[95]: синаптичната връзка, свързваща два неврона, ще бъде засилена, ако и двата неврона изпитват възбуждане в процеса на обучение [73]. Въз основа на това правило изходният сигнал на разглежданата НМ ще бъде генериран от мрежата независимо (приемайки ненулев входен сигнал) и ще се определя от конфигурацията на връзките между невроните и предавателната функция.

2.8.2 Топологическото представяне и математически модел на НМХ

Въз основа на избраната топология (раздел 2.8.1), функцията на разстоянието между невроните i и j в описаната мрежа е функция на познатото евклидово разстояние:

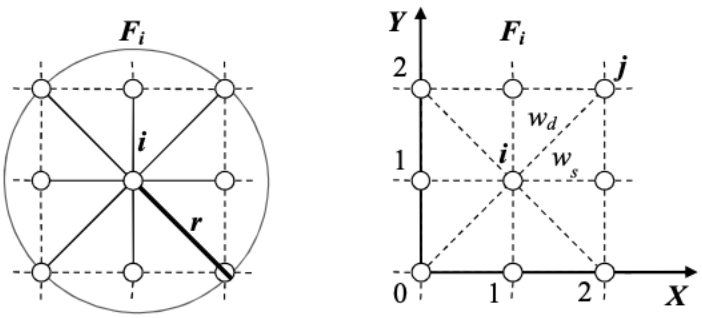
$$d(i,j) = \sqrt{\left(X_i - X_j\right)^2 + \left(Y_i - Y_j\right)^2}, \qquad (3.1)$$

където [i] и [j] са векторите на състоянието на i-тия и j-тия неврон, съответно [73].
Структурата на НМ гарантира, че всеки неврон i е свързан само с подмножество от съседни неврони, образувайки домейн F_i, а всеки невронен регион F е свързан само с подмножество от съседни невронни региони. В рамките на домейна, стойностите на теглата и „околността“ за всеки неврон се определят от функцията f(d) [73]:

$$F(d) = \begin{cases} \frac{1}{d}, & 0 < \mathrm{d} \leq \mathrm{r}, \\ 0, & \mathrm{d} = 0 \text{ или } \mathrm{d} > \mathrm{r}, \end{cases} \qquad (3.2)$$

където r - е радиусът на невронната област F; d - е разстоянието между невроните.
Изразът (3.2) определя физическото значение на невронните връзки в невронната мрежа, което е топологично представяне на дискретно работно пространство: теглото на невронната връзка може да бъде представено като „широчина на честотната лента“ между два неврона в пространството; колкото по-голямо е разстоянието между тези неврони, толкова по-малка е „широчината на честотната лента” (теглото) на връзката между тях. По този начин, за всяка двойка неврони, теглото на връзката е обратно пропорционално на разстоянието между тях.
Нека сега се разгледа ортогонална невронна област Fi в двуизмерна координатна система и стойностите на теглата на връзката за съседни неврони (Фигура 2-7):
Да вземем теглата за връзките с хоризонтален и вертикален (пряк) контакт (W_S) в невронната област F, при x и y = 1. Тогава, в съответствие с (3.1):

$$d(i,j) = \sqrt{(1-2)^2 + (1-2)^2} = \sqrt{2} = 1{,}4142$$



Фигура 2-7. Архитектура на невронна област „F“ в 2D декартова координатна система

и теглото на диагоналните връзки (W_d) съгласно (3.3): 1 / 1,4142 = 0,7071. Както се вижда получените тежести са симетрични за всички i и j в подмножеството F, където: x(i, j) = x(j, i).

Синаптичната функция е тази, която претегля входния вектор на сигналите за обратна връзка от невроните (съгласно (1.1) и (2.1)). За да се изчисли нейната стойност се използва нормализираното скаларно произведение представено като матрица с теглата към всеки входен вектор [73]:

$$\mathrm{Y(t)} \times W_i = [v_1\ v_2\ v_3\ \dots\ v_n] \times \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \cdot \\ \cdot \\ \cdot \\ w_n \end{bmatrix} = v_1w_1 + v_2w_2 + v_3w_3 + \cdots + v_nw_n.$$

Получената стойност се нормализира (нормализирано векторно произведение) [92]:

$$norm = \frac{v_1w_1+v_2w_2+v_3w_3+\cdots+v_nw_n}{w_1+w_2+w_3+\cdots+w_n}.$$

Въз основа на избраната мрежова топология (раздел 2.8.1), горното уравнение добива вида:

$$norm = \frac{v_1w_1+v_2w_2+v_3w_3+\cdots+v_8w_8}{w_1+w_2+w_3+\cdots+w_8}. \qquad (3.3)$$

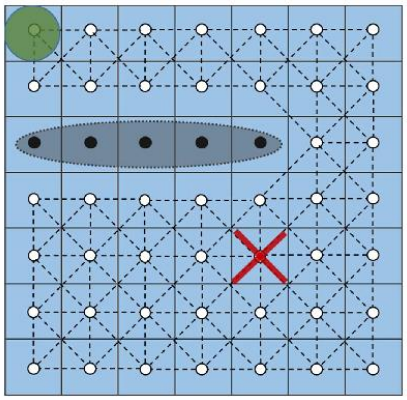
Нормализираната стойност се добавя към стойността на входния сигнал I_i, съответстващ на неврона и образува скаларната стойност – u_i [73]:

$$u_i = norm + I_i,$$

Както видяхме в раздел 2.8.1 НМ може да се използва за да представи дискретно работно пространство. Нека текущото разположение на агента (МР), целите и местоположението на препятствията са дефинирани в дадено (избрано) дискретно (ортогонално) пространство „F“. Нека има НМХ с дадена топология (раздел 3.1.1), чиито неврони са равномерно разпределени върху дадено пространство „F“ по такъв начин, че всяка невронна мрежа да се намира в центъра на съответната дискретна пространствена клетка. Въз основа на това представяне неврон или група неврони (в зависимост от разделителната способност на мрежата) ще бъдат разположени в клетка на мястото на агента. Ще бъдат определени и невроните, разположени на мястото на целта и препятствията – съответно неврон тип „цел“ и невроните тип „препятствия“ (Фигура 2-

8).
Очевидно при създаването на невронната карта стойността на един неврон не трябва да съответства (едновременно) на агент и цел, или цел и препятствие. Идентифицирането на невроните по тяхното "предназначение" се извършва с помощта на входния вектор на първоначалното състояние на мрежата (I_1, I_2,..., I_n) [60].
Всеки елемент от входния вектор $\boldsymbol{v_i}$ е равен на първоначалното състояние на съответния неврон със стойности вариращи в диапазона реални числа [0, 1]. Състоянието на целевия неврон винаги ще бъде установено в 1 а състоянието на препятстващите неврони винаги ще бъде 0 – това са неактивни неврони, „изключени“ от мрежовата структура. Тогава, в зависимост от вектора на входния сигнал, стойността на $\boldsymbol{u_i}$ ще бъде определена, както [73]:

За целеви неврон $u_i = norm + 1$,
За всички останали неврони $u_i = norm$,



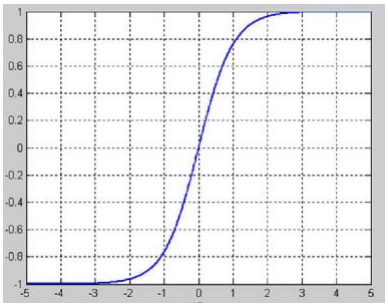
- област с неактивни неврони (препядствие); - агент (MP); - цел; - работно пространство; ∗ - неактивен неврон ; **o** – активиран неврон (свободно пространство);

Фигура 2-8. Представяне на дискретно работно пространство с невронна мрежа (невронна карта)

Първоначалното състояние на останалите неврони също ще бъде равно на 0, но тяхната стойност ще се промени, когато мрежата навлезе в равновесно състояние и ще се определя от активиращата функция „Ф“ (или т.н. трансферната функция на неврона), която преобразува всеки елемент от входния вектор $\boldsymbol{u_i}$ в изходния вектор $\boldsymbol{v_{out}}$:

$$v_{out} = \Phi(u_i)\,,$$

Както бе споменато по-горе, състоянието $\boldsymbol{v_i}$ на неврон i приема стойността в диапазон от реални числа [0, 1]: „1“ означава, че невронът е максимално активиран, а „0“ е инактивиран неврон.
Следователно *функцията активиране Ф(u)* трябва да е равна на „0“ за нулев сигнал и да нараства монотонно с положителен входен сигнал, насищайки до „–1“ и „+1“, съответно от –∞ до +∞. Пример за такава функция е хиперболичната тангенциална функция представена на фигура 2-9 [94]:



Фигура 2-9. Хиперболична тангенциална активираща функция

Така тази функция ще бъде представена математически [73]:

$$\Phi(u_i) = th(u_i) = \frac{e^{2u_i}-1}{e^{2u_i}+1}, \tag{3.4}$$

Както бе отбелязано по-горе (раздел 1.4.1), общото състояние на мрежата ще бъде описано с вектора Y(t) = [$\mathcal{V}_1$ (t), $\mathcal{V}_1$ (t), $\mathcal{V}_3$ (t),..., $\mathcal{V}_n$ (t)] с всички негови възможни стойности, които определят фазовото пространство на състоянията „S“ на системата [73].

Както е показано по-горе в този раздел, невродинамиката на всички невронни мрежи на Хопфийлд се характеризира с правилото на „Хеб“ [95], както и изпълнението на условието за конвергенция. Това включва допълнително усилване на синаптичните връзки между невроните, които изпитват многократно задействане, и също така гарантира, че мрежата накрая се установява в стабилна позиция в пространството на състоянието. По долу ще се представи изпълнението на тези условия за дадена конфигурация на НМХ.

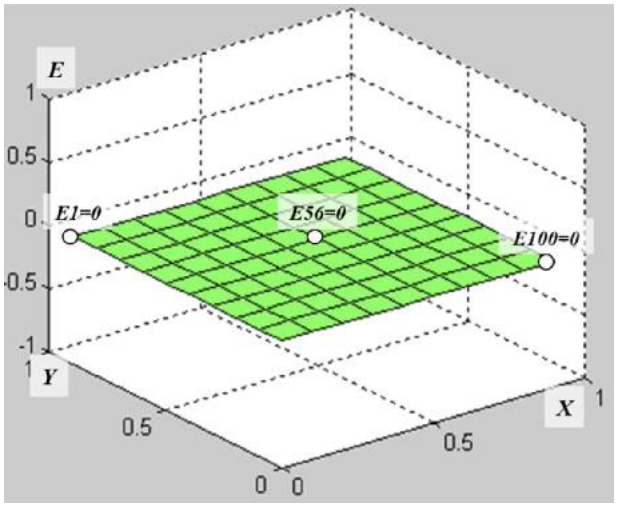
2.8.3 Невродинамика на класическа НМХ

За да се изследва енергийните взаимодействия между невроните в мрежата, ще конструираме математически модел на примерна НМХ от 100 неврона чрез софтуерен подход за моделиране с приложение на Python (**Приложение А** към Дисертация). Ще визуализираме самия процес на активиране и до достигане състояние на равновесие на невронната мрежа благодарение на графичния пакет „PyGame”.

Въз основа на дадената ортогонална топология и стойностите на теглата на връзките на невроните (раздел 2.8.1), мрежата може да бъде представена като невронна мрежа 10×10, където всеки неврон е разположен в центъра на отделна клетка и е свързан само на подгрупа от неврони, разположени в най-близките клетки (фигура 3.2).

Всеки неврон в процеса на активиране ще приеме свои собствени енергийни стойности, според които се формира матрица от сигнали (*активационна матрица*), чийто размер е равен на размерността на невронната мрежа. Въз основа на тази матрица можете да изградите триизмерна повърхност, чиито върхове „E“ ще съдържат съответните стойности на матрицата (стойност на сигнала на съответния неврон) - тази повърхност ще представлява т.н. „невронна карта“ за дадена набор от входни сигнали.

За случая, когато няма установени препятствия и цели, входните сигнали за всички неврони са равни на нула (I_1 = 0, I_2 = 0, I_3 = 0..., I_n = 0) и, ако мрежата не е била активирана преди, тогава нейното състояние ще остане нула. В този случай невронната карта ще изглежда така както е на фигура 2-10 [93].



Фигура 2-10. Нулево състояние на мрежата „Хопфийлд“ като „невронна карта“ с 100 неврона

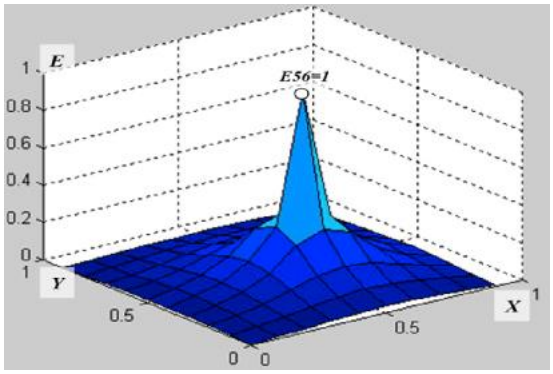
Предполага се, че е зададена цел, тогава еквивалентният на мишената-неврон ще получи максимален вход (напр. клетка 56, I_{56} = 1). Разпространението на този сигнал в мрежата през невроните, свързани в дискретното пространство като процес на активиране ще продължи, докато не бъде изпълнено условието за конвергенция (1.3), т.е. докато мрежата достигне равновесие.

По-долу са стойностите на матрицата за активиране за случай I_{56} = 1 (Таблица 2-1), получена при извършена симулация с програмен код/скрипт „Python3“ (***Приложение А***):

0.0034	0.0071	0.0113	0.0155	0.0187	0.0198	0.0185	0.0151	0.0104	0.0052
0.0071	0.0153	0.0249	0.0349	0.0429	0.0459	0.0426	0.0341	0.0229	0.0113
0.0113	0.0249	0.0423	0.0625	0.0799	0.0868	0.0794	0.0612	0.0393	0.0187
0.0155	0.0349	0.0625	0.1017	0.1417	0.1598	0.1412	0.1001	0.0585	0.0267
0.0187	0.0429	0.0799	0.1417	0.2591	0.3103	0.2585	0.1398	0.0752	0.0330
0.0198	0.0459	0.0868	0.1598	0.3103	**1.0000**	0.3097	0.1578	0.0819	0.0355
0.0185	0.0426	0.0794	0.1412	0.2585	0.3097	0.2579	0.1393	0.0748	0.0328
0.0151	0.0341	0.0612	0.1001	0.1398	0.1578	0.1393	0.0985	0.0573	0.0261
0.0104	0.0229	0.0393	0.0585	0.0752	0.0819	0.0748	0.0573	0.0366	0.0173
0.0052	0.0113	0.0187	0.0267	0.0330	0.0355	0.0328	0.0261	0.0173	0.0085

Таблица 2-1. Стойности на матрицата за активиране при I_{56} = 1

Фигура 2.11 показва невронна карта, изградена според данните за активиращата матрица от Таблица 2-1:



Фигура 2-11. Тип невронна карта при активиране на 56-ти неврон

Ако се активират няколко неврона, тогава на повърхността на всеки от тях ще се появят пикови стойности и получената повърхностна форма ще бъде представена като сливане на активиращите вълни и ще се появят няколко върха. В дисертационната работа не се очаква такъв сценарий и затова няма да се разглежда в проучването.

Тъй като в дисертацията се разглежда софтуерният модел на дискретна НМХ (раздел 1.4.1, фигура 1-9), процесът на активиране на мрежата също има дискретен (итеративен) характер. В този случай Δt на обратните връзки на невроните ще бъде равно на компютърното време за изчислението на $\boldsymbol{v_i}$ (формула 3.4) за всички неврони на мрежата.

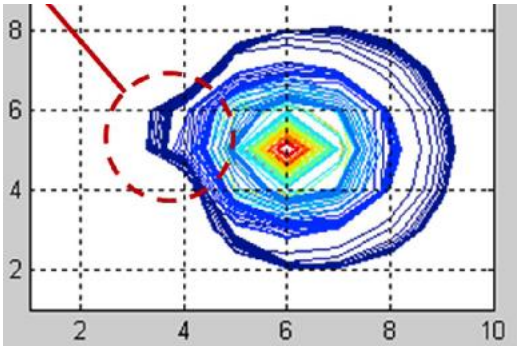
Също трябва да отбележи, че при извършване на машинни изчисления в машата програма на "Python3" се налага използването на променливи от двоен тип (число с двойна точност с плаваща запетая) за обработка и запазване на сигналите на невроните. Този тип променлива има размер от 8 байта и съответната граница на точност е $\approx 10^{-16}$, и следователно сигналът на неврон ще се счита за различен от нула, ако $E_i > 10^{-16}$. По този начин естеството на разпространението на сигнала в мрежата ще бъде *последователно на стъпки и етапи*: на първия етап целевият неврон и най-близките неврони в съседство ще имат ненулев сигнал, който от своя страна ще предаде сигнала на своите съседи на следващия етап и т.н. В резултат на това НМ трябва да премине през няколко „етапа", преди да завърши активирането.

Както бе споменато по-горе (раздели 2.8.1. и 2.8.2), невроните с препятствия са неактивни неврони и тяхното състояние винаги е настроено на „0" и активирането не се разпространява през тях - тези неврони са „изключени" от мрежата. Фигура 2-12 показва активационната повърхност и графично представяне на вълновата природа на разпространението на активиране със затворено пространство като препятствие (препятствието включва всички неврони до 4-та колона).

Фигура 2-11 показва как възпрепятстващите неврони действат като "бариери" за активиране, които въпреки това могат да се разпространяват около тях чрез ненулеви неврони (Фигура 2-12).

Трябва да се отбележи, че колкото е по-голям броят на нулевите неврони, толкова повече време (брой итерации) ще е необходимо да завърши активирането цялата мрежа. Този ефект е свързан с намаляване на „каналите" (броя на връзките) на разпространение на сигнала поради невроните, „изключени" от структурата на мрежата препятствия (Фигура 2.10). Също така, в съответствие с (формула 3.3), нулевите неврони ще понижат нивото на сигнала на съседните неврони и съответно местата, в които се намират, като по този начин невроните с препятствия ще образуват области с намален потенциал около тях [73]. В тази посока има възможност да се оптимизира алгоритъма като изчисленията

не са за цялото работно пространство а частично до достигането на конкретна цел, след което да следва прекратяване на алгоритъма.



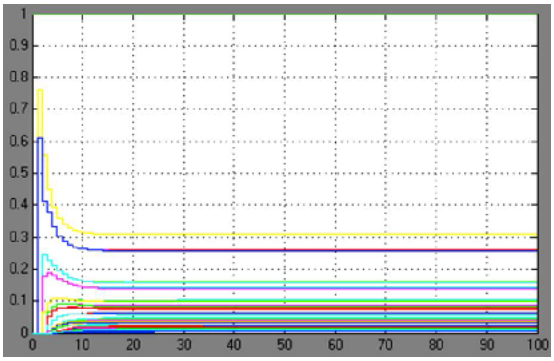
Фигура 2-12. Активационна повърхност и график за разпространение на активиране за отворено препятствие (неврони 44 и 54)

По долу се разглежда процеса на конвергенция, който ще трябва да се минимизира съгласно поставените критерии в раздел 2.9.1. Както бе споменато по-горе в този раздел 2.8.1, при разглеждане на процеса на активиране е необходимо да се вземе предвид и изпълнението на условието за сближаване (формула 1.3). Предполага се, че ако мрежата е стабилна, тогава ще приеме стабилно състояние във фазовото пространство, тоест всички неврони в мрежата трябва да достигнат постоянни енергийни стойности, които да не се променям при последваща итерация. В Глава 1 (раздел 1.4.1) темата за стабилността на НМХ бе разгледана и е дадено *основното правило*, което гарантира конвергенцията на мрежата и следователно нейната стабилност: *стабилността на мрежата е гарантирана, ако матрицата на тегловните коефициенти W е симетрична и всички диагонални елементи са равни на нула*. За да се провери това условие, се създава невронна мрежа с архитектура както на таблица 2-2 но с размери 4×3 с общо 12 неврона. Както се вижда от таблицата 2-2, матрицата е симетрична и всички нейни диагонални елементи са равни на нула. Защото всички синаптични връзки на всеки неврон са едни и същи независимо от техния брой а това гарантира конвергенция и предполага, че разглежданата архитектура НМХ е стабилна.

```
neuron 0:  [0   , 1   , 0   , 0   , 1   , 0.7, 0   , 0   , 0   , 0   , 0   , 0]
neuron 1:  [1   , 0   , 1   , 0   , 0.7, 1   , 0.7, 0   , 0   , 0   , 0   , 0]
neuron 2:  [0   , 1   , 0   , 1   , 0   , 0.7, 1   , 0.7, 0   , 0   , 0   , 0]
neuron 3:  [0   , 0   , 1   , 0   , 0   , 0   , 0.7, 1   , 0   , 0   , 0   , 0]
neuron 4:  [1   , 0.7, 0   , 0   , 0   , 1   , 0   , 0   , 1   , 0.7, 0   , 0]
neuron 5:  [0.7, 1   , 0.7, 0   , 1   , 0   , 1   , 0   , 0.7, 1   , 0.7, 0]
neuron 6:  [0   , 0.7, 1   , 0.7, 0   , 1   , 0   , 1   , 0   , 0.7, 1   , 0.7]
neuron 7:  [0   , 0   , 0.7, 1   , 0   , 0   , 1   , 0   , 0   , 0   , 0.7, 1]
neuron 8:  [0   , 0   , 0   , 0   , 1   , 0.7, 0   , 0   , 0   , 1   , 0   , 0]
neuron 9:  [0   , 0   , 0   , 0   , 0.7, 1   , 0.7, 0   , 1   , 0   , 1   , 0]
neuron 10: [0   , 0   , 0   , 0   , 0   , 0.7, 1   , 0.7, 0   , 1   , 0   , 1]
neuron 11: [0   , 0   , 0   , 0   , 0   , 0   , 0.7, 1   , 0   , 0   , 1   , 0]
```

Таблица 2-2. Тегловна матрица от 12 неврона НМХ

За да се тества правилото за стабилност, се изгражда НМХ с размери 10 × 10 в разработената системата за моделиране (използвана е библиотека на Pyton3 – „oscilloscope“) на различни типове НМХ (**Приложение А**). В съответствие с избраната архитектура и стойности на коефициента на тегло (раздел 3.1.1) ще се разгледа преходния процес на промяна на състоянието на невроните в мрежата (фигура 2-13):

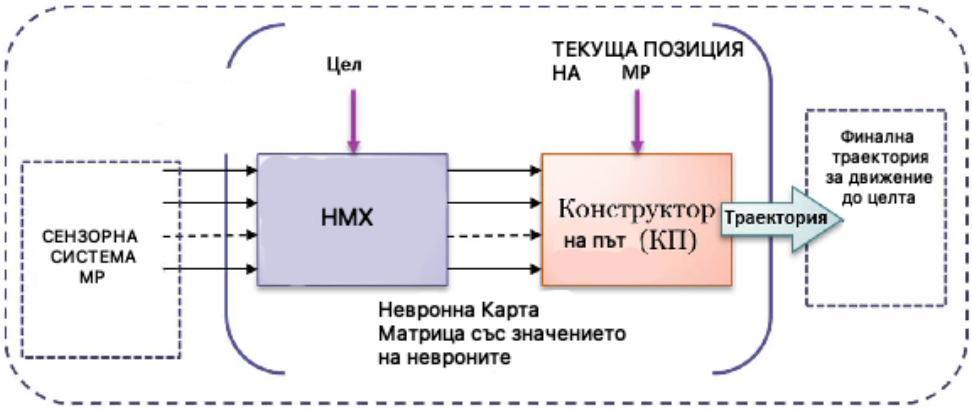


Фигура 2-13. Показания на осцилографа за процеса по промяна на състоянието на НМХ до постигане на стабилност

Както може да се види от фигура 3.6, всички неврони в мрежата накрая ще приемат стабилно състояние, следователно системата е стабилна.
За да се провери коректността на изходните данни на разглежданата архитектура на НМХ, ще се симулира алгоритъма на системата за планиране на траекторията с подхода и метода на „невронна карта“.

2.8.4 Архитектура на „планировчик“ (КП) с невронна карта

За да опишем принципа на действие на системата за планиране на траекторията, базирана на метода „невронни карти“, където се представя МР, разположен в двуизмерно дискретно работно пространство (С).
Системата за планиране на траекторията (СПТ) за единичен МР се състои от два основни блока: невронна карта и конструктор на пътя (КП). Архитектурата на системата е показана на Фигура 2-14 [60, 73].



Фигура 2-14. Архитектура на система за планиране на траекторията в група от МР, базирана на невронна карта

Системата работи по следния алгоритъм [93]:

- определя се целта и текущата позиция на МР;
- на базата на сензорна информация за С се определя конфигурацията
- работно пространство (местоположение на препятствия);
- се формира входният сигнал за НМ „Хопфийлд“, докато входният сигнал невронът, съответстващ на целевата клетка, ще бъде по-голям от 0 (точка на активиране);
- изчакване на приключване на процеса на активиране;

- формиране на матрица от изходни сигнали (невронна карта);
- невронната карта се подава към СР блока;
- получаване на крайната траектория на движение към дадена цел.

Можем да кажем, че подобна система за планиране на траектории (Фигура 2-1) осигурява събиране, обработка и предаване на информация, необходима за вземане на глобални решения и изпълнение на локални задачи (по-специално задачи за позициониране).

СПТ има следните основни характеристики и свойства:

- *Делимост* – системата може да се представи като състояща се от множество подсистеми и елементи: НМ „Хопфийлд“, който генерира карта и КП, който изгражда траектория по нея; от своя страна НМ се състои от множество взаимосвързани основни елементи на неврони и т.н.
- *Цялостност* - свойствата на цялата система зависят от свойствата на нейните елементи, което означава, че промяната в свойствата на елементите води до промяна в свойствата на цялата система. Например, при промяна на архитектурата на невронната мрежа, невронната карта ще бъде различна със същата конфигурация на работното пространство, което ще доведе до необходимост от промяна на функциите за изчисляване на траекторията на КП блока. В разглежданата система тези фактори до голяма степен се определят от сближаването на НМ, което от своя страна на свой ред, зависи от коректността на входните данни, идващи от източника на информация.
- *Възникване* - свойствата на цялата система не са сума от свойствата на нейните елементи, така че решаването на проблема с планирането на траекторията е невъзможно без информационния канал и връзки между елементи, които осигуряват трансформация и структуриране на входа и изхода данни за обработка от съответните елементи (по-специално връзката между източника на информация и НМ, връзката на НМ към КП и връзката на конструктора с подсистемата за управление на робота).
- *Комуникация* – системата не е изолирана, взаимодейства с околната среда; средата и системата си влияят взаимно. Взаимодействието на системата с околната среда възниква както в началния етап на решаване на проблема с планирането на траекторията (при получаване на информация за конфигурацията на работното пространство чрез източника на информация), така и в процеса на формиране на крайното решение (когато роботът се движи по изчислените стъпки на траекторията).
- *Йерархия* - системата, от една страна, се състои от подсистеми, а от друга, самата тя е част от по-обща система (например част от системата за управление от групата МР).

Въз основа на факта, че е използвана НМ „Хопфийлд“, в системата, тя има свойството да запомня изображения („памет“) [94, 46]. Така генерираните невронни карти могат да се използват многократно и да действат като вид база от знания, на базата на които системата ще изгради или коригира траекторията в процеса на движение на агента (т.е. „да взема решения“). Въз основа на това свойство може да се кажем че системата може да се разглежда като интелигентна.

Също така, разглежданата система несъмнено ще принадлежи към класа на сложните. Принадлежността към този клас се посочва от сложността на функционирането (поведението) на тази система, което до голяма степен се определя от набора от възможни състояния на НМХ и невродинамиката. Невродинамичните процеси от своя страна се определят както от структурните особености на основните изчислителни

елементи на мрежата (невроните), така и от структурната организация на цялата мрежа (топологията).

2.8.5 Модул „конструктор на пътя“ (КП)

Както бе споменато в раздел 2.8.3, по-нататъшната обработка на изходните данни от НМ и изчисляването на траекторията се извършва от отделна изчислителна подсистема наречена - „конструктор на пътя“. В роботиката често КП се нарича „планировчик“ или „планер“. Задачата му е да изчисли път за изпълнение на някаква целева функция. Пътят може да бъде известен и като маршрут, в зависимост от избраната номенклатура и алгоритъм. Два примера може да са изчисляване траекторията и пътя до целта (например от текуща позиция до крайната цел) или с пълно покритие (докато не се обходи цялото свободно пространство). Планерът трябва да има достъп до състоянието на околната среда и данните от сензори, буферирани в база данни достъпни от него. Планирането най-кратко може да се опише така:

- Изчислете най-краткия път
- Изчислете пълен път на покритие
- Изчисляване на пътища по редки или предварително дефинирани маршрути

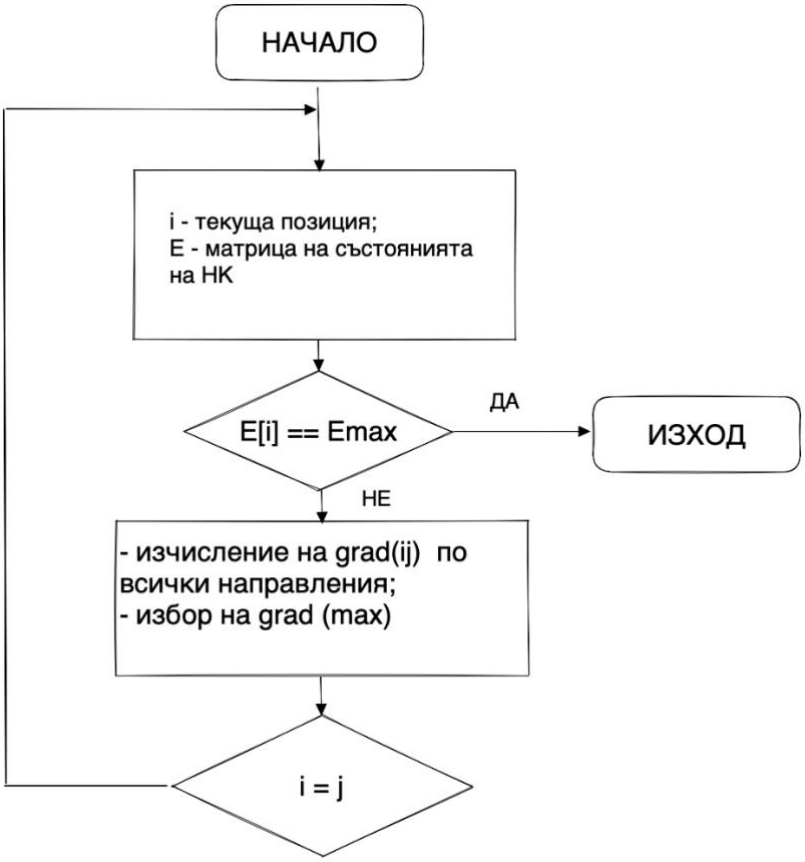
Общата задача в от „планировчика“ е да изчисли валиден и потенциално оптимален път от текущата поза до целева поза. Съществуват обаче много класове планове и маршрути, които се поддържат.

В зависимост от количеството налична информация за конфигурацията на работното пространство, модулът КП може да изчисли цялата траектория (ако местоположението на препятствията е известно напълно и тази конфигурация на работното пространство е статична) или да изчисли отделни части от траекторията (местоположението на препятствията не е известно предварително, т.е. конфигурацията на работното пространство е динамична).

В дисертацията изчисляването на траекторията започва след формирането на матрицата на състоянието на НК (невронна карта) от началната позиция на агента. При всяка нова стъпка се изчисляват 8 стойности: градиенти (grad_{ij}) във всички възможни посоки от текущата стойност на неврон “i” към съседни (“j”-ти неврон). След това се избира максималната стойност на градиента (grad_{max}) и се извършва преход в тази посока към нова позиция (i = j). Търсенето продължава, докато се достигне целевият неврон с максимална енергийна стойност (E_{max}) [73]. При изграждането на система за изчисляване на траектории за група МР в общо работно пространство „С“, на КП се възлага и ключовата функция за разрешаване на конфликтни ситуации и избягване на препрядствия, като се вземат предвид приоритетите и съответната динамична корекция на траекториите. В зависимост от задачите, които трябва да се решават, както и информационните и изчислителни ресурси, са възможни различни подходи за внедряване на системата за планиране на групата МР. По-долу ще се представят хардуерните и софтуерните характеристики необходими при изграждането на системата когато има различни подходи.

Както бе споменато по-горе, функциите по обработка на изходните данни на НМ и конкретното изграждане на траекторията се изпълняват от блока „конструктор на пътя“ (КП).

Базовия алгоритъм на процеса е показан на фигура 2.15.



Фигура 2-15. Обобщен алгоритъм на „конструктора за път“

Матрицата на състоянието на невроните (невронна карта), образувана от НМХ, се прехвърля в КР блока. Тук се изчислява траекторията до целта с помощта на тази матрица и чрез подхода на търсене „максимален градиент“ в посока от неврон i, съответстващ на текущата дискретна клетка, към съседен неврон “j”. След определяне на максималната стойност, операцията се повтаря за j-тия неврон и така нататък, докато се намери целевият неврон и се изгради крайната траектория.
Ако i и j са два съседни неврона, тогава градиентът в посока от i към j ще бъде определен по формулата:

$$grad\ (i,j) = (E_i - E_j)w_{ij}, \tag{3.5}$$

където E_i е стойността на активиране на i-тия (текущ) неврон, E_j е стойността на активиране на j-ти неврон; а w_{ij},е коефициентът на тежест на връзката между i-тия и j-тия. Трябва да се отбележи, че при завършване процеса на активиране, енергийната стойност на невроните, свързани с нулеви неврони (разположени близо до препятствия), винаги ще бъдат по-ниски от стойността на невроните, свързани с ненулеви. Този ефект е вид „близка предвидливост” на агента осигурява плавно избягване на препятствия по траектории, близки до оптималните.
Алгоритъма за планиране на глобалния път от КП е показан по долу в таблицата 2-3:

Require: [cur_pos (номер текуща позиция)];
[target (номер целева позиция)];
[E (матрица на активация)];
If (Ecur_pos <> 0) **OR** (Ecur_pos <> Etarget)
While $E_{cur_{pos}}$ **=** E_{target} **do**
- изчисляване на grad(i,j) за всяко от възможните направления;

```
    - избор на максимална стойност;
    - определяне на номера на следващата позиция (next_pos);
    -  cur_pos = next pos
  end while
end if
```

Таблица 2-3. Обобщен алгоритъм за изчисление на глобалния път от КП

Ако целта е недостижима (агентът е заобиколен от препятствия), стойността на неврона на местоположението на агента ще бъде 0 и всички неврони в околната среда ще бъдат в нулево състояние, тъй като вълната на разпространение от целевия неврон няма да мине през препятствията и да достигнете до тях. Това условие се проверява в самото начало на алгоритъма и ако е изпълнено, КП ще съобщи за невъзможност да се изчисли траекторията.

Ако мрежа с големи размери (няколко хиляди неврона) с голям брой невронни препятствия се използва за формиране на матрицата на състоянието на невроните, тогава съществува риск от локални екстремуми. В този случай, за да се избегне зацикляне при изчисляване на градиентата в областта на екстремума, е необходимо да се добави операцията за нулиране на стойността на невроните на вече изчислените клетки (Ecur_pos = 0, cur_pos = next_pos). Методите за елиминиране на локални екстремни стойности ще бъдат разгледани по-долу.

2.9 Резултати от моделирането с невронна карта

Програмата за моделиране на разглеждания метод на планиране и синтез на пътя е реализирана с специално разработен софтуер (**Приложение А** към дисертацията) а резултатите в Глава 2 са графически визуализирани с помощта на библиотеките „pygame“ и „matplot“ в “python”.

Приемат се следните условия при моделирането:

- конфигурацията на средата е известна предварително;
- МР знае своята позиция във всеки един момент;
- МР се разглежда като точков и има размери, които не надвишават размера на един дискретни клетки на работното пространство;
- агента може да се движи във всяка възможна посока към съседните клетки;
- работното пространство - НМХ е с размер 10×10 с ортогонална топология. По този начин невроните на мрежата са разположени в 2-измерна правоъгълна решетка в центъра на дискретните клетки на работното пространство и за всеки неврон има 8 съседи, следователно, 8 възможни посоки за движение на робота.

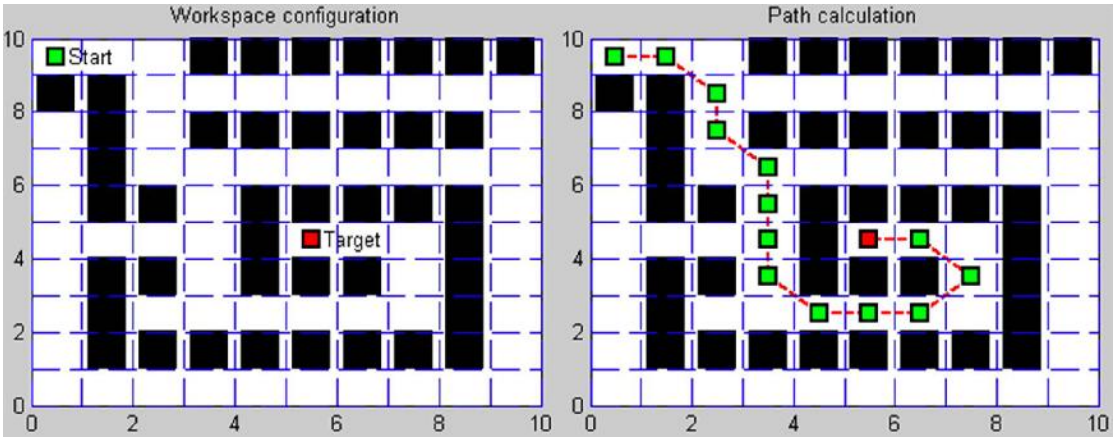
Установяват се следните входни променливи:

- **target (цел)** – номер на целевата позиция;
- **start (начало)** – номер на началната (текуща) позиция;
- **obst (препятствие)** е вектор, чиито елементи са числата на местоположението на препятствията, при **obst = 0**, => не се задават препятствия.

Изходни променливи са:

- **cycles (цикъл)** е броят на стъпките за активиране (итерациите), необходими на мрежата завършване на процеса на активиране (dostigane стабилност в мрежата);
- **steps (стъпка)** - броят на стъпките в траекторията до достигане на целта.

Резултатите от конкретната симулация са представени по-долу на фигури 2-16:



цел	начало	препятствия	цикли	стъпки
56	1	4,5,6,7,8,9,10,11,12,22,24,25,26,27,28,29, 32,42,43,45,46,47,48,49,55,59,62,63,65,66, 67,69,72,79,82,83,84,85,86,87,88,89	19	13

Фигура 2-16. Планирането на глобална траектория при сложна конфигурация на работното пространство

2.9.1 Критерии за подобряване алгоритъма НМХ

Изводите направени до момента водят до заключение, че за придвижване на рояк мобилни роботи в области с висока насищане на препятствия е препоръчително да се разгледат две взаимосвързани задачи:

а) планиране на траекторията на групата като единна структура в работното пространство – това което се постига с невронната карта и алгоритъм на НМХ, представена като глобална навигация или т.н. конструктор на пътя (КП);

б) планиране на траекторията на всеки мобилен робот поотделно, в рамките на разглежданата структура/формация като локална навигация. Т.е. обособяване на клъстери от МР и въвеждането на лидер за обособената локална формация. Освен с НМХ при *локална навигация* е възможно да се съчетае НМХ с ИПП за подобряване времето за изчисление при определени ситуации – голям бр. препятствия и голям бр. мобилни роботи. За съжаление тази хипотеза с тип двуслоен планировчик няма да бъде разгледан в дисертацията.

Резултатите от симулацията, представени в раздел 2.8.3, потвърждават работоспособността на системата за планиране, базирана на НМХ в работна среда със статични препятствия. Въпреки това, за правилната работа на тази система в рамките на концепцията за изграждане на неконфликтни траектории за МР в група, е необходимо *веднъж да се оптимизират алгоритмите на системата и на втори план – да се подобри паралелизацията на изчислителните процеси*. По-специално, необходимо е да се разработи алгоритъм за взаимодействие на НМХ с бордовата сензорна система МР, с помощта на който би било възможно да се получат действителни входни данни за НМ при условия, когато конфигурацията на препятствията в работната зона не е известна предварително. Приема се позицията на роботите във всеки един момент за известна но за препятствията винаги има неяснота и следва да се приема, че се работи в условия на несигурност. Допълнително големия брой роботи в движение усложнява и забавя генериране на такава карта. Алгоритмите за работа на самата невронна мрежа също трябва да бъдат оптимизирани, така че активационната матрица да се актуализира в

съответствие в реално време със постъпващата сензорната информация.
Както показаха измерванията в хода на оценката на броя итерации, необходими на НМ, за да влезе в равновесно състояние (раздел 2.8.3), по-голямата част от изчислителното време при конструирането на траектория и нейното коригиране се изразходва за образуването на нова „невронна карта“ (Eout) като матрица от стойности на невроните, получени в резултат на процеса на активиране. Активирането от своя страна е итеративен процес на взаимодействие на невроните в НС чрез синаптични връзки и обратна връзка, докато невроните достигнат стабилни изходни стойности (раздел 1.4.1). По този начин времето, необходимо за формиране на невронна карта, е продължителността на процеса на активиране, който въз основа на итеративния характер на курса може да бъде изразен със следната формула:

$$T_A = t_c . N. \quad (3.6)$$

където t_c е средната продължителност на един цикъл на активиране (време за изчисление Ф(u) за всички неврони на всеки етап); „n“ - е броят на итерациите (цикли). Следователно, за да се подобри производителността на системата за планиране в динамично променящо се работно пространство, е необходимо да се модифицира общия алгоритъм за генериране на невронна карта по такъв начин, че *изчислителното време на цикъла на активиране (t_c) и броят на тези цикли (n) са сведени до минимум.*
Ето защо смятаме, че предложените нововъдения по-долу като подход биха подобрили изпълнението на базовия алгоритъм за навигация с **НМХ:**

- ***частична корекция на картата*** при засечена промяна в работното пространство – раздел 2.10;
- използване на ***линейна предавателна функция*** *с насищане* за активационна функция Ф(x) в НМХ за скъсяване продължителността на процеса на активиране при конфигурации с много на брой агенти и препятствия в работното пространство – раздел 2.11;
- изпълнението на изчислителните процеси да се извърши чрез много на брой нишки т.е. да се изпълни паралелен режим на работа на навигационна система (демонстриран в Глава 3) с паралелизация както на кода на алгоритъма така и на комуникацията между отделните услуги (управление на комуникацията на база на събития). Тези техники ще бъдат по-детайлно разгледани в Глава 3 – раздели 3.2.1 и 3.2.4 но поради няма да бъда анализирани в експерименталната част.

2.10 Въвеждане частична корекция в невронната карта

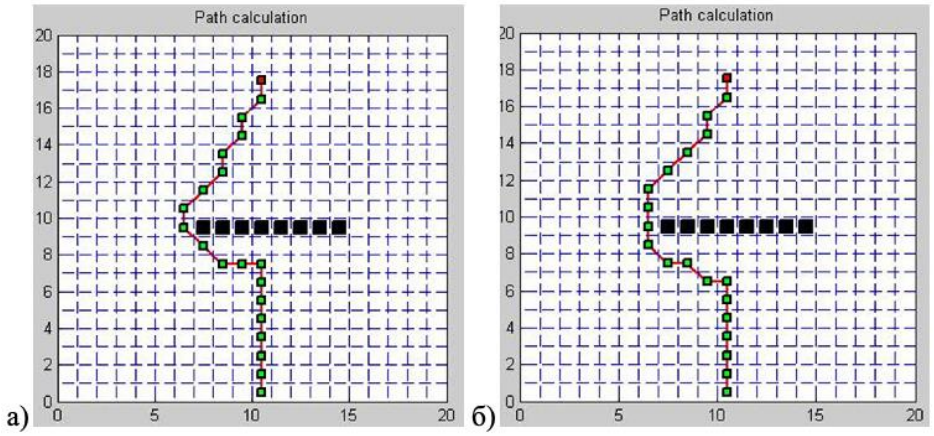
В зависимост от скоростта на бордовите изчислителни системи и чувствителността на МР сензорните системи, както и скоростта на движението на МР, е възможно да се използват два метода за коригиране на траекторията с цел избягване/заобикаляне на препятствието, които се изпълняват в реално време от навигационните системи на рояка МР:

- частична корекция на картата – при ниска изчислителна скорост, както и при ниска чувствителност (радиус) на сензори (например само IR далекомери) или работа на сензори в условия на смущения (шум, светлина и др.);
- формиране на нова карта - при високи изчислителни възможности

на МР и с високоефективна и скъпа сензорната система (системата има голям обхват и шумът се филтрира, например, чрез сравняване на данни от различни източници за идентифициране на препятствия: далекомер и 2D лазерно-оптичен скенер “LIDAR” за точно замерване на разстояния над 50м. разстояние).

Ето защо нека да се разгледат тези два случая:

А/ създаване на частичната корекция на картата – това значително намалява времето за изчисляване на траекторията. Резултатите от моделирането на процеса на настройка на едно работно пространство 20×20 са показани във фигура 2-17 по долу:



Фигура 2-17. частична корекция на невронната карта: а) 1 итерация (показания енкодер: 29.3); б) 5 итерации (показания на енкодера: 28.3)

Както се вижда, „плавността на байпаса“ ще се определя от показанията на енкодера на сензора за дистанция. Енкодерът показва цялата дължина на пътя (приемайки, че един преход в посока напред е равно на 1), включително пътя, изминат при завои: колкото по-дълъг е пътят, толкова по-малко „плавност“. Следователно за въвеждането на такава частична корекция на картата, която спестява ресурси на процесора и предлага съответно по-голяма скорост на създаване на траекторията, трябва да включва алгоритъм в следните стъпки:

1) Първоначалната карта (E_{out}) се формира на основа на начална конфигурация на работното пространство;
2) Картата се запазва в паметта ($E_m = E_{out}$);
3) Изчислява се траекторията на E_m и движението на МР започва към целта;
4) Ако се достигне целевата позиция – следва изход от алгоритъма;
5) Ако се открие ново препятствие на траекторията ($C_{obst} = true$), в Em стойностите на сигналите на съответните неврони се нулират чрез индекси от масива Obst{}: $E_{m_{Obst1}}$ = 0, $E_{m_{Obst2}}$ = 0,..., $E_{m_{ObstN}}$ = 0;
6) Извършва се корекция на невронната карта чрез подаване на модифицираната матрица на състоянието (E_m) на входа на НМХ (E_{out}= NET(E_m)) „n“ брой пъти;
7) Преминаване към стъпка 2.

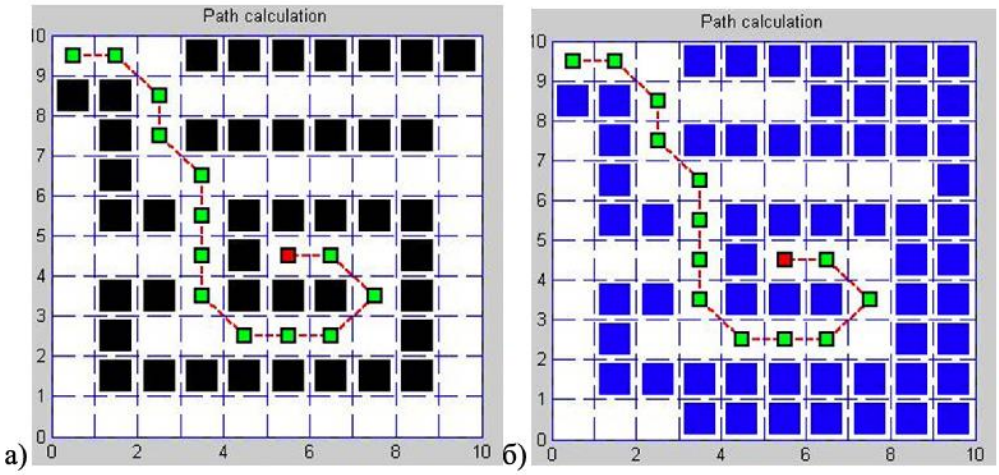
Така, колкото по-голям е „n“, толкова по-силен е сигналът на невроните, които са най-близо до препятствието, намалява и зоната с намален потенциал се разширява около нулевите неврони. По този начин НМ ще коригира траекторията на МР в посоката на препятствието и колкото по-голяма е площта с намален потенциал (т.е. колкото по-голямо е „n“), толкова по-рано ще бъде коригирана траекторията. Образно казано, „n“ влияе върху „плавността” на избягването на препятствието;

Въз основа на получените резултати можем да заключим, че частичната корекция на невронната карта намалява плавността на избягването на препятствието. Също така, като недостатък на този метод за корекция в движението, заслужава да се отбележи трудността при изграждане на траектория при заобикаляне на препятствия със сложна форма (особено в малко пространство за преминаване/придвижване).

Б/ когато се използва метод за създаване на нова цялостна карта като корекция на траекторията, системата работи по следния алгоритъм:

1) Формира се невронна карта (E_{out}), като се вземе предвид известната в момента конфигурация на работното пространство;
2) Изчислява се траекторията и започва движението на МР към целта;
3) При достигане на целевата позиция – изход от алгоритъма;.
4) Ако се открие ново препятствие, индексите, съответстващи на тях позициите на невроните се записват като списък в отделен масив Obst{};
5) Ако текущата траектория пресече препятствие, тогава се формира нова невронна карта (E_{out}) с нулиране на невроните по индекси от масива Obst{}.
6) Преминаване към стъпка 2.

Също така си струва да се отбележи, че формирането на невронна карта тук възниква, когато първоначалното състояние на мрежовите неврони е нула, но не се активира цялата невронна мрежа, а само онези неврони, които са необходими за изграждане на траекторията. Математическото моделиране показа, че модифицирането на условието за сближаване практически не оказва влияние върху качеството на изграждането на траекторията, т.е. КП правилно заобикаля препятствията и крайната траектория остава близка до оптималната. Сравнителните резултати от симулацията са представени на фигура 2-18 (б - сините квадратчета показват неактивни неврони).
Този ефект е възможен поради факта, че се активират само онези неврони на мрежата, които са необходими за изграждане на траектория от текущото местоположение на агента, като по този начин се намалява както броя на повторенията в процеса на активиране, така и броя на активираните неврони, респективно и броя на изчислителните операции за претегляне и прилагане на трансферните функции. По този начин този метод за формиране на невронна карта може ефективно да се използва за коригиране на траекторията в *динамично променящо се работно пространство*.



Фигура 2-18. Резултатът от изграждането на траектория в работното пространство 10×10: а) пълно активиране с 19 итерации; б) частично активиране с 13 итерации

2.11 Въвеждане на линейна предавателна функция с насищане в НМХ

Повишаване производителността на НМ за формирането на изходния сигнал е също възможно с оптимизирането на трансферната функция на невроните, което ще сведе до минимум времето за изчисление на цикъла на активиране (t_c) и следователно общото време на формиране на невронната карта T_A (съгласно (3.6)).

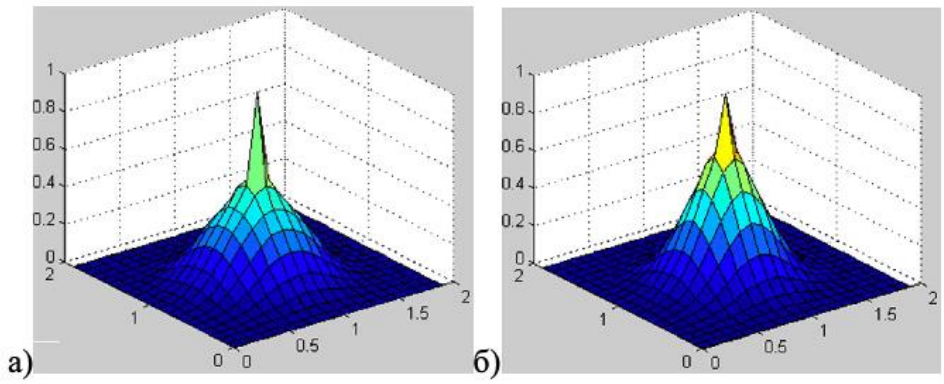
Както вече споменахме, основното условие, при което е възможно образуването на невронна карта е, че функцията за активиране Ф(x) трябва да е равна на 0 за нулев сигнал и да нараства монотонно с положителен входен сигнал, насищащ към 1 до +∞. Първоначално най-точно беше избрана хиперболичната тангенциална активираща функция (3.4) отговарящи на всички необходими условия. Тази функция осигурява максимална плавност на разпределението на енергията на активиране на невроните в мрежата и по този начин ви позволява да формирате траектории, които са близки до оптималните. Очевидният недостатък на функцията обаче е *сложността на хардуерната реализация и високата цена на компютърното време* - до 7 математически операции на неврон (което е значително, като се има предвид големия брой неврони в мрежата).

Като алтернатива може да се използва *линейна трансферна функция с насищане* вместо формула (3.4) както следва при следните правила:

$$\Phi(u_i) = 0, \quad \text{ако } u_i \leq 0,$$
$$\Phi(u_i) = u_i, \quad \text{ако } 0 < u_i \leq 1,$$
$$\Phi(u_i) = 1, \quad \text{ако } u_i > 1.$$

Както се вижда функция $\Phi(u_i)$ е много по-лесна за хардуерно изпълнение, ще изисква по-малко разходи за компютърно време (1 математическа операция вместо 7 в съответствие с (3.4) и в същото време удовлетворява основните изисквания за конструиране на невронна карта.

Общ изглед на графиките на функциите е показан на Фигура 2-19.



Фигура 2-19. Повърхност на активиране на невроните: а) хиперболична тангенциална функция; б) линейна функция с насищане

Времето се увеличава с усложняването на конфигурацията на работното пространство, т.е. като същевременно усложнява формата на препятствията между агента и целта. Този ефект се обяснява с факта, че тъй като конфигурацията на работното пространство става по-сложна (т.е. броят на невроните в мрежата и броят на препятствията се увеличават), ще са необходими повече повторения за завършване на процеса на активиране. Тъй като

линейна функция с насищане се характеризира с по-високи стойности на невронните сигнали в областта на насищане, разпространението на сигналите в мрежата е по-бързо, което прави възможно образуването на невронна карта (съгласно (3.7)) за по-малко време.
Както се вижда от фигура 2-19, хиперболичната тангенциална функция се характеризира с по-голяма „плавност“ на промените на сигнала в мрежата, но в същото време „забавя“ увеличаването на сигнала на невроните, разположени в насищането регион, което се потвърждава и от получената активираща повърхност. Този ефект (в съответствие с раздел 3.1.1) също ще повлияе на общото време за активиране на НМ. Според резултатите от симулацията, която е направена с разработения софтуер-програма на „Python3“ (***Приложение А***) беше установено опитно, че използването на линейна функция с насищане като функция за активиране може да увеличи производителността на мрежата до 2,5 пъти (в зависимост от размерите и конфигурацията на работното пространство).

2.12 Изводи по глава втора

1. В процеса на сравнителен анализ на навигационните архитектури бяха идентифицирани основните критерии, които пряко влияят върху избора на един или друг подход на изпълнение: скорост, сложност, консумация на енергия и надеждност. *Хибридния и децентрализиран* подход се приемат като основни подходи за внедряване на една системата за планиране за групата МР.
2. Високите изисквания за гъвкавост на взаимодействието между членовете на групата (агенти – МР) и ЦИБ, както и за скоростта на реакция на цялата система от външни динамични промени и смущения, налагат използването на техники за изчислителна паралелизация. Въз основа на избраната схема за управление, паралелизацията на изчисленията ще се извърши програмно, т.е. използвайки многонишкови изчислителни технологии, докато за да се постигне максимална производителност, части от кода с последователни изчисления трябва да бъдат намалени колкото е възможно повече.
3. Като цяло предложените архитектури на системата за планиране са в състояние да решат поставените цели и задачи на изследването. Независимо от това е необходимо да се прецизира структурата на отделните елементи на системата, като се вземат предвид спецификите на решаваните задачи за подобряване на ефективността, стабилността и стабилността на системата за различни условия на околната среда и броя на роботите в групата.
4. Подхода на разглеждане динамиката на невронните карти за формиране на навигационен път може да предостави подобрение и оптимизация на времето с избягване на локални минимуми. Поради голямата, неограничена тестова зона, няма случаи на мобилни роботи от екипа да достигнат локален минимум и да заседнат. Въпреки това едно бъдещо добавяне и комбиниране с метода на „отблъскващите полета“ би улеснил МР да промени посоката и скоростта си при необходимост за да излезе от „задръстване“. Атрактивното поле ще привлече МР бързо и директно към целта. Счита се, че при взаимодействието между отблъскващите и атрактивните полета успешно ще маневрира МР в граници следвайки лидера в сценарии с формации от МР.
5. Изграждането на математически модел и експерименталното изследване на невродинамиката на НМ на Хопфийлд разкри, че активирането на мрежата има итерационен характер. Колкото по-голяма е стойността на тегловните коефициенти на връзките на невроните, толкова по-малък е броят на итерациите, необходими за активиране.
6. Стигнахме до извода, че е нужно въвеждане на допълнителни подобрения в

архитектура на системата за планиране, отчитащи особеностите на движението на агент от групата МР в динамично променящо се работно пространство и информационна недостатъчност и в частност отново с НМХ. Такова едно подобрение би могло да бъде паралелната обработка и изчисления за създаване на НК.
Инженерната методология за хардуерно-софтуерна реализация на предложения модел и резултатите от нейната работа ще бъдат разгледани в трета глава.

ТРЕТА ГЛАВА. ПРОТОТИП НА СОФТУЕРНА СИСТЕМА ЗА НАВИГАЦИЯ ПРИ ГРУПА МОБИЛНИ РОБОТИ. ЕКСПЕРИМЕНТАЛНА ПОСТАНОВКА И АНАЛИЗ НА РЕЗУЛТАТИТЕ

Трета глава е посветена на разработения експериментален прототипен алгоритъм за планиране на траекторията за групата МР предложен в Глава 2 и наричан за краткост контролер - "**HNAV**" („Hopfield Navigation"). Алгоритъма е реализиран на езика за програмиране Python (CPython) версия 3.4, така че кода да може да се изпълнява много-нишково и на много-процесорни и многоядрени платформи. Този нов алгоритъм (плановик) гарантира, че траекториите на роботите се създават възможно най-бързо, без опасност от сблъсъци, реорганизирайки образуваните формации, когато се изисква от условията на околната среда – спазване дистанции и разстояния между препятствия и роботи като се използва цялата налична информация за работната среда и всички агенти в реално време. Той се справя дори и с движещи се препятствия като част от променящата се околната среда тъй като картите се опресняват в реално време (не се предвиждат експерименти с придвижване във формация).
Така предложения плановик – конструктор на пътя (КП) намалява времето за опресняване на невронната карта, тъй като както видяхме в раздел 2.10 не се налага създаване на изцяло нова невронна карта. *Както ще се види в експерименталната постановка по долу оптимизацията постига (t* n, където n е бр. на роботите в групата.*
В тази глава също се посочват детайли свързани с приложението за 2D графична стимулация на мобилни роботи "**Webots",** което визуализира движението на група мобилни роботи в пространството в реално време. Контролера HNAV е отворен софтуер и може да комуникира с всяка външна навигационната система. "Webots" също може да комуникира с HNAV (който се изпълнява за всеки агент като отделен контролер) по-отворен комуникационен интерфейс както директно така и през навигационни платформи и операционни системи като ROS2. Тази комуникация е детайлно описана в раздел 3.2.4
Създадения код за приложението – контролер за навигация „HNav" е приложен в (**Приложение А**) към дисертацията.

3.1 Архитектура на експерименталната постановка

3.1.1 Дефиниции на модела и ограничения в експерименталния подход

Всеки агент/МР в условия на реално придвижване трябва притежава следните базови функционални модули:

- комуникационен;
- безжична комуникация за обмена на данни в реално време;
- видео камера с висока резолюция – над 5 мегапиксела;
- ултразвукови датчици-сензори за ориентация за къс и дълъг обхват;
- инерционни сензори: 9-DOF сензор с 3-осен акселерометър, 3-осен магнитометър и 3-осен жироскоп;
- лазерен сензор на база лазерна технология – LiDAR;
- диференциално управление на движението с PID контролер за два двигателя;
- два позиционни (hall) сензора с PWM изхода за отчитане на импулси при ротационното движение на всяко от двете колела;

В експериментална постановка в Глава 3 се приемат ограничения и специфични условия наложени върху агентите/МР и околната среда за да се извърши експериментална постановка и да се стигне до изводите свързани основно с подобрения алгоритъм за конструиране на пътя. Някои от тях са:

- *1/ Работно пространство* (среда) – работното пространство се дискретизира (ортогоналната дискретизация) в 2D равнина, където всеки дискретизиран квадрат е част от първи квадрант в декартова координатна система с координата x и y или за улеснение за целите на симулациите направени по долу – с номера на полето в работното пространство;
- *2/ Препятствия* (O):
  - „n" на брой;
  - в начално състояние локализирани и позиция на (O) – x ostart/y ostart, и заемат квадрати в работното пространство (в първи квадрант);
  - препятствията са само фиксирани и неподвижни след стартиране на процеса на симулация и се маркират в началото на процеса на изграждане на пътя и не може да търпят промени по време на изпълнение на програмата;
- *3/ Цел* (T): достигането до крайната цел е глобална задача за МР. В рамките на построяването на пътя е възможно да изникват и др. подзадачи като: заснемане на препятствие, заемане на определена формация от група МР и т.н. но тези допълнителни техники няма да се разглеждат в дисертацията.
  - една единствена цел - един МР може да има само една цел като не се разглежда сценарии „multi-target";
  - координатите на целта са известни;
  - глобалния път до целта се определя чрез алгоритъм предложен в дисертацията въз основата на т.н. „невронна карта" и разгледана в детайли в Глава 2 и демонстрирана в Глава 3 в експерименталната част;
  - целта е може да бъде различна за всеки отделен агент/робот;
  - целта има координати по x, y на декартовата координатна система – отново в първи квадрант;
  - целта заема само по една позиция - квадрат в дискретната среда;
- *4/ Формация от МР - лидер и последователи* (МР) – не се разглеждат на този етап формациите поради ограничения обем на дисертацията.

Приема се и се разглежда т.н. *глобален плановик* за изграждане на път единствено и само за лидера-МР. Няма да се навлиза в проучване на условия за оформяне на екипите, избор на лидер измежду всички МР и избор на конкретния навигационен режим, всеки от който е с различна цел – избягване на сблъсъци между роботите/препятствия, следване на лидера си, обследване на препятствие и т.н.. Както се разбра от Глава 2, че този проблем може да се разреши от т.н. локален планер на траектория (примерно с ИПП) но

няма да бъде разглеждан в дисертационния материал. Точният локален плановик отчита ограниченията и генерира осъществими локални траектории в реално време. В реални условия роботите с помощта на локален плановик трябва не само да изпълняват плавни осъществими траектории, но и да могат да избягват движещи се препятствия съобразно режимите, които са изброени по долу. В рамките на рояка от МР всеки агент взема решение къде да продължи, използвайки както локалната информация за околността, така и глобалната цел (посока и скорост на движение). При такъв сценарий всеки агент може да планира избягването на препятствията, като се има предвид позицията на най-близките препятствия и съседните агенти или се засекат подвижни обекти.

Локалния планер или локален КП, разполага с реална и актуална информация на състоянието само в близкото пространство и може да проектира пътя на робота напред много бързо и без нужда от големи ресурси, изчислявайки локално осъществим път при всяка итерация. *Конструктор на пътя* (КП) ще оперира само в **един от възможните различни режими** в зависимост от текущата задача, която ще изпълнява мобилния робот. И това е т.н. **„проследяване“** на целта.

В реални условия извън постановката описани в Глава 2 и 3 биха се очаквали **повече поведения** от агента и съответно различни режими:

- „проследяване“ на цел - **M2G** (move2goal);
- „обследване“ на обект – сканиране форма и вида на достигнато препятствие - **BGB** (boundary finding behavior);
- инцидентно „избягване на сблъсъци“ за избягване от обект при засечено непредвидено/неочаквано преместване и движение - **RAB** (runaway behavior);
- „следване на лидера“ в групата, в който режим се поддържа или сформира или прегрупира в определена геометричната форма - **FL** (follow the leader);
- „случайна посока“ – избор на случайна посока при необходимост когато имаме локален минимум и всички подходи са изпробвани и няма изход от ситуацията.

❖ 5/ *Мобилен робот (МР - агент)*:

- Разглежда се мулти-агентна система от „n” брой МР. За целите на симулациите и проучването ни ще бъде използван разработения софтуерен КП или т.н. глобален „планер“ – контролер “**HNAV**”, където пътя се изчислява *чрез алгоритъм, предложен в дисертацията като модифициран вариант на невронна мрежа „Хопфийлд“ с подобрена активационна функция и оптимизирано генериране на невронната карта;*
- при създаване на невронната карта МР се разглежда като точково твърдо тяло без специфична форма и размери, което заема 1 квадрант (1 неврон от невронната карта).
- Но реални условия извън направените симулации може да се приеме, че МР може да се впише в окръжност с радиус “r” като на центъра му се намира сензора за отчитане на обекти а в даден момент „t“ неговата позиция $p2$ в системата xOy е известно: $p2(t)$ = ($x2$,$y2$,θ2), където $x2$, $y2$ са координати на агента; θ2 е ориентацията, дадена от ъгъла между посоката на вектора на линейната скорост на агента v=< и оста Ox. Известни са също стойностите на линейните $v2$ и ъгловите ω2 скорости на агента и неговата постоянна надлъжна дължина $l2$;
- приемаме, че има достатъчно голям бр. от мобилни роботи с общо радарно покритие от минимум 10% от общата площ на работното пространство;
- знаят се координатите MR във всеки един момент от време – записва се в базата номера на полето – брои се от ляво на дясно като най- горния ляв ъгъл

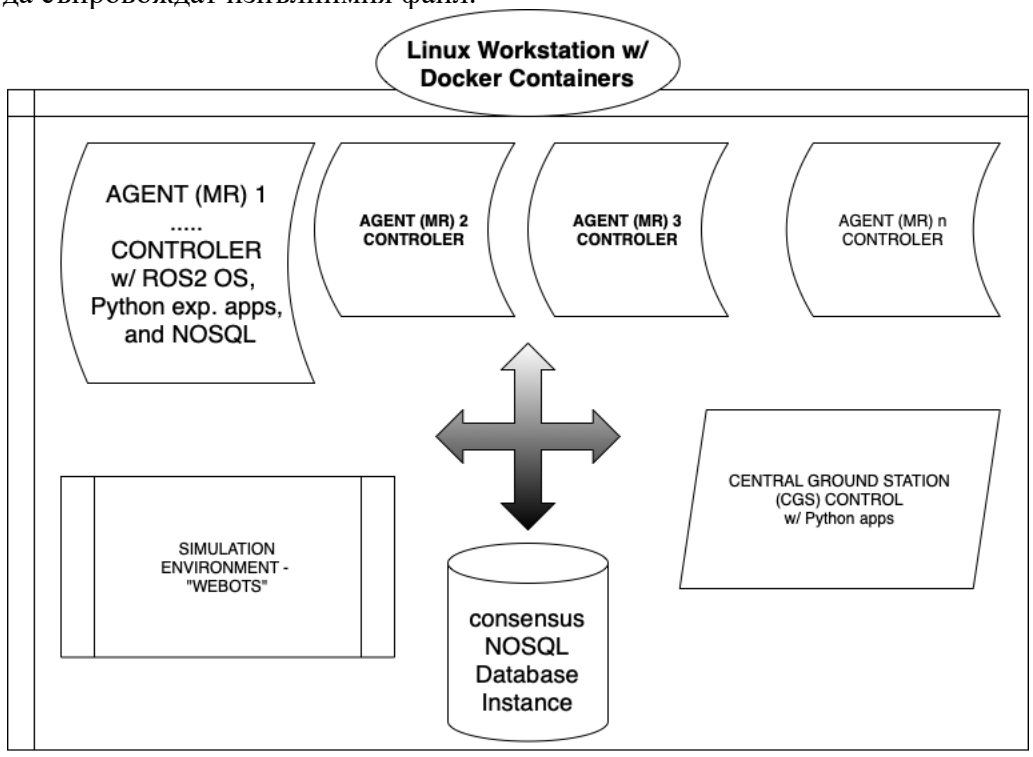
е с номерация 1 а най-долния десет е крайния;
    - възможни са общо 8 посоки на движение в пространството – всички диагонали плюс основните четири посоки – север, юг, запад и изток;
- 6/ *Локалната навигация* свързана с формиране и управление на формацията (поведението на МР във формация) както и процесите на постигане на консенсус между агентите във формацията за смяна на режима/поведението при децентрализирано управление няма да се разглеждат в дисертация;
- 7/ *Пътна карта* (с информация за наличните структури/препятствия във всеки един момент) няма да има нужда да се опреснява при експериментите, тъй като се приема, че са предварително зададени. В децентрализиран режим на управление е необходимо да се води информация за препятствията и останалите МР в реално време. Това подобрявала ориентация и съкращава времето за изграждане на пътя;
- 8/ *Сензорна информация* на МР – за конкретните експерименти с разработения алгоритъм в Глава 2 не се използват сензори. Самите координати на всички препятствията и целите при експерименталната постановка са ясни във всеки един момент;
- 9/ Комуникация и инфраструктура по време на експериментите са детайлно разгледани в раздел 3.2.1. При децентрализирана системата за планиране на движение на голям брой агенти/МР се налага да има комуникационен блок с специфично телеком оборудване (в случая безжично) във всеки един от агенти, така че да се подсигури бордова система с непрекъсваемо предаване и приемане на данни в реално време към централизираната база данни или към другите участници-агенти. Тъй като агентите трябва да се движат автономно по изчислена от тях траектория в условия на недостатъчна информация за работното пространство проблем с комуникацията или липса на данни би навредило на цялата група. И когато информацията за текущото местоположение на други агенти не е налична или е неправилна, например при наличие на радиосмущения или повреда на бордовите комуникации е невъзможно правилното разрешаване на конфликтни ситуации и оптимизиране на пътя съобразно конкретната ситуация. При инцидент в безжичната комуникация появата на други агенти по дадена траектория ще се възприема от МР като динамично непредвидено препятствие и траекторията ще бъде коригирана с помощта на техниките и алгоритъма ни за преодоляване на инциденти с помощта на вградените сензори. В представената опитна постановка и с цел улесняване на извеждането на резултатите комуникацията между стимулационния софтуер и алгоритъма ще става в локална мрежа и стабилна комуникационна среда:
    - Вътре между процесите на приложенията в контролира на МР и отделно между агента и CGS;
    - При опитната постановка в глава 3 не се разглежда възможна комуникацията между агентите;
    - Приема се, че агента е всъщност е самата програма алгоритъма, която се изпълнява на 4 ядрена система. А CGS е стимулационния софтуер, който се изпълнява на настолен компютър в една мрежата;
    - На разположение е централна консенсусна БД, в която се записва координатите на всеки МР при всяка стъпка;
- 10/ Път до целта - пътя до целта – разглежда се само глобалния път до целта;
- 11/ *Конфликтни ситуации* между роботите при изграждане на пътя - при генериране на НК е възможно да се създаде конфликтна ситуации при движението на МР когато съседни агенти имат общо поле за следващ ход. За целта в представения алгоритъм се представят сценарии за разрешаване на ситуации описани в Раздел 3.3.2.

3.2 Среда за провеждане на експериментите

За разработването и тестването на подобрения алгоритъм на НМХ в многонишков режим е необходимо да се избере *софтуерна платформа*, която да е специално развита за целите на имплементиране на техники и алгоритми от изкуствения интелект. И на второ място да се избере *симулационна среда,* която да демонстрира навигация и управление на всички мобилни роботи едновременно в мулти-агентен сценарий– Фигура 3-1.

Поради посочените по горе описани специфични изисквания цялостното програмно осигуряване (логика, анализ и визуализация) на платформата е реализирано на езика „Python“ и развойна среда “Pycharm”. Програмен език „Python“ дава широки възможности за оптимизация на програмния код по бързодействие и ефективност, управление на оперативната памет, директна връзка с операционната система, позволяваща от своя страна достъп до хардуерните ресурси, паралелно изпълнение на изчислителните задачи и др. Допълнителна мотивация за избора на Python е съвместимостта на кода за компилатори за всички познати платформи.

Като програмна среда е използван „PyCharm“, който е функционално богат текстов редактор, разполагащ с множество средства за улесняване и автоматизиране на програмното осигуряване за широк диапазон програмни езици. Но най-важно е че симуалционния софтуер „Webots“ също може да се изпълни в „PyCharm“. Нужно е да отбележим, че всички горепосочени програмни библиотеки и средства се разпространяват безплатно. Предимство на библиотеката е, че е статично свързана с кода на приложението и не са нужни допълнителни динамично свързани библиотеки, които да съпровождат изпълнимия файл.

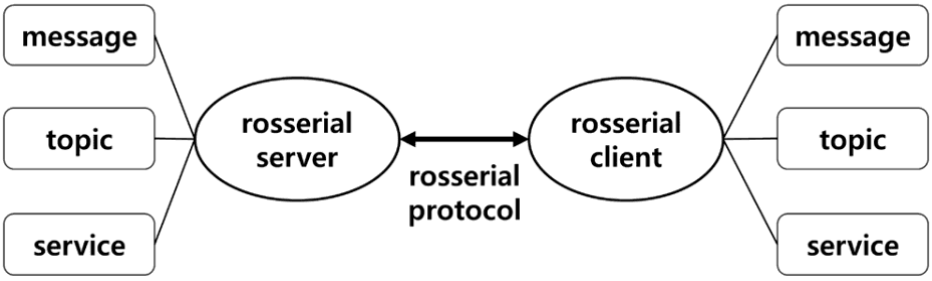


Фигура 3-1. Компоненти в експерименталната постановка

3.2.1 Комуникационна инфраструктура

Всички агенти както и CGS контролера представляват „Docker” инстанции в

експерименталната постановка. Така бързо и лесно може да се симулират всички процесите на самостоятелен контролер за комуникация. Между отделните агенти има възможността да се използва т.н. класически ROS2 serial „сокет“ за комуникация с отворен мрежов порт резервиран за конкретния отделния агент. Такъв подход е показа на фигура 3.2 по долу:



Фигура 3-2. ROS serial клиент – сървър подход за комуникация между отделните агенти

На втори план трансфера на данни между самите процеси и приложения на ниво ОС на агент както и във всеки модул или контролер от агента също изисква паралелизация, бързина и сигурност. Ето защо в посочената постановка ще се ползва комуникация тип „вътрешните процеси“ – „Inter process Communication (IPC)“. Това е колекция от методи и техники, които осигуряват комуникация между два процеса с различно пространство в паметта (на ново ОС) без значение как програмите функционират и какъв език използват независимо една от друга. Независимостта кара програмите написани на различни езици да комуникират лесно и сигурно през IPX. В резултат, на което IPC позволява изпълнението на едновременни задачи [107] паралелно с директен достъп до паметта.

Блоковата диаграма на паралелната архитектура, която се използва за комуникация между агентите е основана на събития (event driven) и е показана на фигура 2-4 (раздел 2.7) [AARTON2021]. Всеки един от агентите комуникира както с останалите в групата така и комуникира самите приложения които се изпълняват на процесорната система на агента. В примера са използвани протоколи за съобщения като: „MAVLink“, „DDS-XRCE“ през „“ и „rosserial“ протокол. Процеса по изграждане на комуникационен модел преминава през следните етапи:

- Създаване на модел на решение с правила в „Drools“ [108]. В тази стъпка се дефинират параметри като входни/изходни данни, събития, възли, комуникационните протоколи;
- при настъпване на очакваното събитие започва самостоятелен процес - „възел“, активиран от конкретно приложение (което в случая трябва да осъществи комуникация) – фигура 2-4 (раздел 2.7) [AARTON2021]. Това са аритметичните стъпки в реално време (в многонишков програмен режим) – на ангийски „pipeline“, които ще обработват конкретното събитие. В експериментите в Глава 3 се използва „Eventerpretor“ [109] в случаи когато се ползват по мощни процесори и „TaskManagerIO“ [110] като приемливо решение за процесори с по-ниска мощност, които работят с операционни системи както „FreeRTOS“. Всеки процес се изпълнява в отделна нишка. Възможно е да има и да се управляват множество самостоятелни „pipline” в един агент. Създадените възли остават в изчакващ режим и не са активни, докато не пристигнат нови данни в свързаните неблокиращи опашки;
- Процес 1: данните от сензора се предават на централния изчислителен блок (ЦИБ). DDS-XRCE протоколът се грижи за споделянето на данни от „pipeline“ 2,

използвайки комуникация от типа IPC в реално време.“AceOrix” от Eclipse.org [111] и представлява *SHM*. Прехвърляне на съобщения става със закъснение по-малко от 1 μs;

- ❖ Процес 2: отговаря за повторното изпращане на данните от сензора до всички други възли в агента чрез IPC SHM и DDS-XRCE протокол, управляващ втория процес;
- ❖ Процес "n": Управление на съобщението с навигационни данни, получено от всички възли в мрежата и прехвърлянето му към компонентите ROS2. В случай на затруднения с комуникацията, този възел ще пренасочи съобщението към ЦИБ.

Посочената блокова схема за интегриране и поддържане на комуникационните канали на ниво МР са детайлно разгледани в мои публикувани изследвания: [AARTON2019], [AARTON2021], [AARTONCS2020] и [AARTONTU2021].

3.2.2 Операционна система и среда за комуникация

ОС за всеки един индивидуален агент е ROS (Robotic Operation System) [48]. Тази операционна система съдържа различни функционални модули и позволява да установите взаимодействие между тях. Първоначално ROS е разработена в лабораторията за изкуствен интелект на Станфордския университет и в момента се разработва и поддържа от общността на „Open Robotics Foundation“. В последствие и в последната година този проект се финансира от Европейски фондове като “Horizont 2020”.

ROS2 съдържа много модули за изпълнение на широк спектър от задачи в областта на роботиката, като задачата за едновременна локализация и картографиране. ROS е фокусиран върху unix-подобни системи, но основните поддържани системи са Ubuntu Linux/Debian Linux, а други като MacOS X са експериментални. В момента има версии и за микро-рпоцесорни решения.

Взаимодействието и комуникацията между модулите в ROS се осъществява на принципа на абонамент (съобщения които са за конкретното „заглавие“). Всеки модул в ROS може да създава теми в раздел или т.н. секция, в които се пишат съобщения в определен стандартизиран формат, докато други модули могат да четат тези съобщение, когато се абонират за съответната секция. Това е разяснено по горе в раздел 3.2.1

3.2.3 Графична платформа за симулация

Експерименталното потвърждаване на аналитичните резултати, изведени във втора Глава и за визуализация на движението на групата мобилни робот, ще се използва специализирана платформа за компютърна симулация на кооперативна навигация при множество от роботи – „Webots“ [112]. За целта се използва контролер-симулатор в платформата „Webots“. Генерирането на входни данни, анализа им, обработката на резултатите, както и построяването на графики и други общи функционалности се реализират от общи за симулатора софтуерни модули, всички те написани на езика „Python“.

Симулирането на кооперативна навигация се характеризира с висока изчислителната сложност и голям обем на обработваната информация. Затова контролера за управление на навигацията ще бъде реализиран в среда „PyCharm”. PyCharm е междуплатформена интегрирана среда за разработка (IDE), специално за езика Python. В същото време, за да бъде симулацията достатъчно интерактивна, е нужно тя да се проведе в реално време и с потребителски интерфейс, позволяващ бързо и интуитивен контрол и добавяне на различни конфигурации. Симулационен софтуер „Webots“ е разработен в сътрудничество с швейцарския федерален технологичен институт в Лозана, щателно тестван, добре документиран и непрекъснато поддържан [112]. Ще се съобразим с

наличните имплементирани функционални модули и списъка от сензори и мобилни роботри. Компания, която стои зад този проект - Cyberbotics е разработила „Webots“ [112], като софтуер за симулация на мобилна роботика и предоставя среда за бързо създаване на прототипи за моделиране, програмиране и симулиране на мобилни роботи. Наличните библиотеки за роботите позволяват създадените симулационни контролни програми, които последствие да може да се прехвърлят към истински мобилни роботи. „Webots“ позволява да се дефинират и модифицират различни роботи, споделящи една и съща среда. За всеки агент (робот или друг участник/инструмент в средата) може да се определят редица свойства, като форма, цвят, текстура, маса, триене и физически параметри т.н.

Платформата разполага с графичен интерфейс, позволяващ наблюдение и управление на симулацията в реално време – времетраене, брой итерации. За реализацията му е използвана графичната библиотека OpenGL, позволяваща построяването на двумерни и тримерни изображения, множествено буфериране на видео паметта, хардуерно ускорение на графичните изчисления. Допълнително предимство на OpenGL пред други графични библиотеки е, че повечето операционни системи го поддържат фабрично и не е необходима никаква допълнителна инсталация, нито промени по програмния код при компилация на различни платформи.

Резултатите от симулациите, освен графично, могат да бъдат експортирани на твърдия диск под различни файлови формати, включително видео.

В по сложни сценарии всеки робот може да се програмира с голям брой налични сензори и задвижващи механизми, като може да се използват различни среди за разработка. „Webots“ има характеристики и предимства които включват:

- пълна библиотека от сензори и изпълнителни механизми при създаването на модела;
- възможност за програмиране на роботите на C, C++, Python, Java, или от друг софтуер чрез TCP/IP;
- използване библиотека катто ODE (Open Dynamics Engine) за точна физична симулация;
- създават се AVI или MPEG симулационни филми за уеб и публични презентации;
- бързо и лесно локализиране в работното пространство;
- позволява да се симулират мултиагентни системи за *swarm* сценарии;
- 2D и 3D среда за визуализация на симулациите.

В конкретния частен случай симулация на код-контролерите за „Webbots“ става едновременно на една и съща машина и тази симулация обработва и визуализира няколко MP и техните т.н. контролер с КП, които се управляват от външни контролери, написани като код в средата „Pycharm”. Тази операция може да се повтори в нов терминал за всеки робот участващ в симулацията.

3.2.4 Паралелизация на навигационната система и софтуерния код за КП

Както вече беше споменато в Глава 2, софтуерната реализация на системата за планиране трябва да се извършва с помощта на паралелно изчислителни технологии за да се реализира асинхронен контрол на агентите и поради необходимостта от паралелна обработка на голямо количество данни. В същото време, както е посочено в раздел 2.7, при внедряването на такава система са възможни два основни подхода:

- многонишков подход в случай, когато софтуерният пакет е разработен на еднопроцесорна система но се изпълнява с няколко нишки;

- или многопроцесорен когато има многопроцесорна система.

В същото време трябва да се има предвид, че многопроцесорна система може да се изгради както с *централизиран подход* - с централен контролер с няколко процесора, така и *децентрализирано*, когато имаме много агенти отговорни за различни функции-роли (в групата), всеки от които има вградена изчислителна мулти-процесорна система. В последния случай е необходима универсална архитектура за софтуерно планиране, която осигурява паралелна обработка на данни, като се вземат предвид различни платформи за хардуерно изпълнение. Такава архитектура е показана на Фигура 3-3.

В раздел 2.7 бе споменато, че рамката на това изследване ще включва *хибридно* управление като основен подход при внедряването на системата за планиране и следователно е възможно да се демонстрират едновременно ***техниките за многонишково и многопрецесорно програмиране*.** Многонишкови функционалности при изпълнение на софтуера - програмата се постигат с основния модул за нишки в езика „Python“ наречен - “**threading**”. При извършването на експериментите се правят няколко различни опита с посочените модули за да изберем най-добрия подход за алгоритъма, които ще се изпълняват на МР.

По време на експериментите се прилагат паралелните изчислителни операции в три посоки:

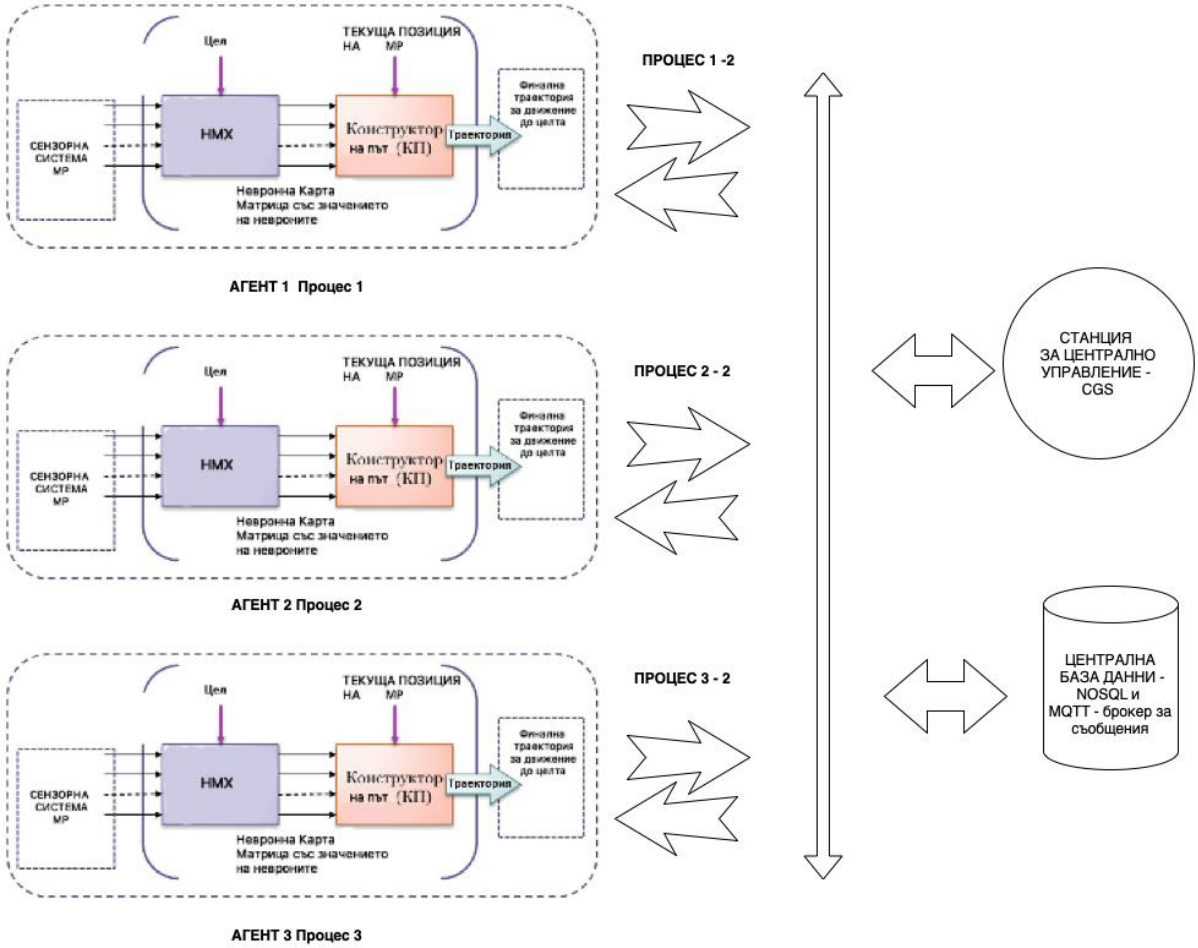
- ❖ Тъй като изграждане на траектория за всеки агент се извършва в отделна изчислителна нишка, обменът на информация (комуникация) между централния контролер за управление отделните приложения за навигация в агента се осъществява чрез обща област в паметта - фигура 2-4 (раздел 2.7) [AARTON2021].
- ❖ Докато само методът на обмен на данни между агентите се различава (в зависимост от хардуера за реализация). В случай на хибридна система за управление комуникацията между агентите и/или станцията за централно управление (CGS) става с взаимодействие чрез мрежови “сокети” отново в отделни паралелни нишка заедно с останалите процеси представени във фигура 3-3.
- ❖ Посредством използването на стандартните паралелни директиви на OpenMP използвани в CPython, основната част от програмния код с алгоритъма на НМХ и КП също се изпълнява в отделни нишки – Фигура 3-4 и 3-9. Съществото в представения сценарий е, че и програмата „конструктор на пътя” се изпълнява в една изчислителна нишка / процес - върху мулти-процесорен контролер. Както се вижда от фигура 3.9, единствените части на програмата, които не могат да се изпълняват в една нишка, са въвеждането на първоначални данни и подпрограмата за обучение на невронна мрежа.

Както вече бе представено всички пресмятанията, изложени в Глава 3, са извършени с помощта на паралелен код – мултипроцеси (паралелизиране на функциите с алгоритмите) усъвършенстван и разширяван многократно по време на изследванията. Използван е програмният език Python, а паралелизацията е направена с използване на стандартна библиотека за активиране на асинхронен паралелен режим в Python – “multiprocessing” и framework “Ray”. Модула „Ray“ предлага редица предимства на това пред многопроцесорния модул:

- същият код ще може да работи както на многоядрена машина така и на клъстер от машини (едно и мулти-ядрени);
- процесите споделят ефективно данни през споделена оперативна или бърза флаш памет и технология на сериализация с нулево копие на „key value store“;
- в допълнение към дистанционното извикване на функции, класовете могат да бъдат инстанцирани дистанционно.

Очаква се с въвеждането на паралелни изчислителни операции и многонишково

програмиране, на всяка стъпка на изчисление на пътя на предложения плановик допълнително да се повиши бързината на оценка на конфликта и взаимодействие с общата база.



Фигура 3-3. Блокова схема на паралелните процеси в експерименталната установка

3.3 Софтуерен контролер „HNAV" за управление на МР с модифицирана НМХ

Формирането и въвеждането на входните данни е основна стъпка в разработения софтуер- програма. Дефинираните начални променливи са разделени на два типа в зависимост от обхвата:

- глобални - достъпни за всички нишки (т.е. разположени в зона на споделена памет);
- локални - променливите стойности са налични само за конкретна нишка.

Минималният набор от входни параметри, необходими от програмата за изчисляване на неконфликтни траектории, е даден в Таблица 3-1. Също така, като допълнителни параметри, коефициентът на усилване „K" и приоритетният вектор „P" могат да бъдат въведени в програмата.

Коефициентът на усилване е множител на тегловните коефициенти на невронните връзки, с увеличаване на което се засилват синаптичните връзки на невроните и съответно намалява времето за влизане на НС в равновесието. Ако не е зададено усилване, то по подразбиране е 1.

Приоритетният вектор е програмно едномерен масив с размерни единици, съдържащи приоритетни стойности за всеки агент, т.е. P[0] е стойността на приоритета на първия

агент, P[1] е приоритетът на втория и т.н. Векторът на приоритета е незадължителен входен параметър, тъй като приоритетите се присвояват на агентите автоматично в зависимост от скоростите на роботите. Както се вижда от таблица 3-1, приоритетът на робота е обратно пропорционален на неговата скорост, т.е. колкото по-ниска е скоростта, толкова по-висок е приоритетът. Това условие е въведено поради факта, че агент с по-висока скорост на движение може да маневрира по-бързо в случай на конфликтни ситуации, което ще осигурява решение на общия проблем с позиционирането за по-кратък период от време. Въпреки това, всеки критерий (или набор от критерии) за оптимално приоритизиране може да се използва, като например „важността“ на изпълнението на задача или типа робот, ако планирането на траекторията се извършва за хетерогенна група. По този начин въвеждането на приоритети дава възможност да се определят гъвкави правила за взаимодействие между агентите в групата в процеса на движение в общо работно пространство, а също така осигурява изпълнението на обща (за цялата група) задача в най-кратки срокове.

Променлива	**Описание**
N	Цяло число, определящо размерът ма дискретното работно пространство по хоризонтала
M	Цяло число, определящо размерът ма дискретното работно пространство по вертикала
space	Двуизмерен N×M масив, който дефинира индекса c_{nm} за всяка клетка от дискретното работно пространство или индекса за всеки неврон. Попълва се с цели числа в ред от 1 до N×M
obst	Едномерен масив от индекси на неврони с нулеви препятствия. В процеса на изграждане на траекторята може да се промени, ако конфигурацията на работното пространство е напълно неизвестна.
units	Броят на агентите, еквивалентно на броя на изчислителните нишки.
start	Едномерен масив, съдържащ индекси на *начални* позиции за всеки агент (лидер-робот), т.е. start[0] е индексът на началната позиция на първия агент, start[1] е индексът на втория и т.н.
target	Едномерен масив, съдържащ индекси на *целеви* позиции за всеки агент, т.е. target[0] е индексът на целевата позиция на първия агент, target[1] е индексът на втория и т.н.
spd	Едномерен масив, съдържащ стойностите за скоростта на всеки агент, т.е. spd[0] е скоростта на първия агент, spd[1] е скоростта на втория и т.н.
P	Едномерен масив, съдържащ приоритетните стойности за всеки агент, т.е. P[0] = 1/spd[0] е приоритетът на първия агент, P[1] = 1/spd[1] е приоритетът на втория и т.н.
net_weights	Едномерен масив с размерност 9, съдържащ теглата на връзката в НМХ.
shared_paths	Двуизмерен масив с размери 3×i (където i е броят на агентите), всеки ред от който съхранява стойностите на текущата позиция на агент i - shared_paths[i][0], следваща стъпка - shared_paths[i][1], вероятна (предварително изчислена) shared_paths[i][2] стъпка от съответния път до целта. Стъпките в реда shared_paths[i] са стойностите на индексите от пространствения масив, които съставляват част от траекторията на

	агент i, т.е. shared_paths[i] ={cmn0, cmn1, cmn2}.

Таблица 3-1. Глобални променливи използвани в програмата за създаване на НК

3.3.1 Обучение на НМ и създаване на невронна карта

Коефициентите на тежест се задават според дадената топология, която случая е ортогонална. Масивът „net_weights“ се съхранява в споделена област на паметта и е достъпен само за четене за всички изчислителни нишки, което елиминира системни грешки по време на паралелен достъп до данни.

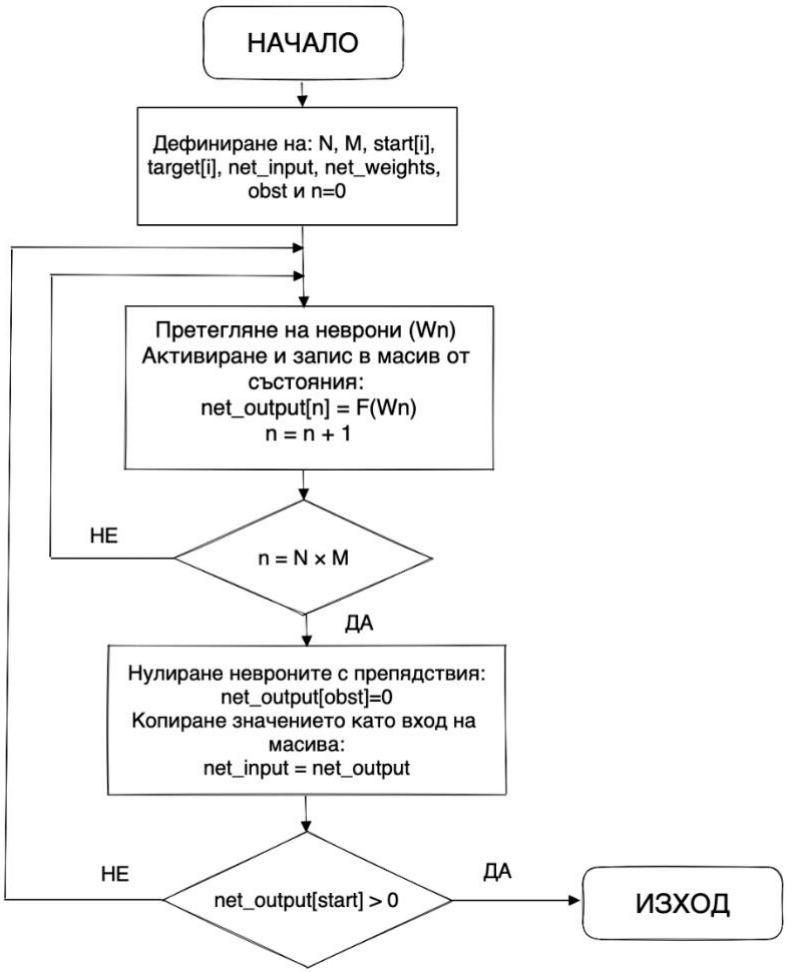
Софтуерният блок НМ се стартира в отделна нишка за всеки агент. Изчислителният процес на блока е софтуерна реализация на математическия модел НМ „Хопфийлд“ (НМХ), описан в глава 1 (раздел 1.4.1) и 2 (раздел 2.8.1). Първоначално на входа на НМ се подава нулевата матрица (масив) net_input с размерност N×M (в съответствие с индексната матрица на пространството на работното пространство). Освен това, всяка стойност на този масив, започвайки от активираната (net_input[target] = 1), се претегля от съответния вектор на тегловните коефициенти „net_weights“, умножен по коефициент - K (ако е необходимо) и заместен във функцията за активиране F. Получените стойности съответно се записват в масива „net_output“ (също с размери N × M) и след това този масив отново се подава към входа като „net_input“, докато стойността на неврона, съответстваща на началната позиция, е нула (net_output[start] = 0). По този начин, софтуерният блок НМ реализира алгоритъм за генериране на невронна карта. Блоковата диаграма на алгоритъма НМХ е показан на фигура 3-4.

Генерираният изходен масив от състоянието на невроните net_output се подава като входен аргумент към блока на програмата КП, където се изчислява траекторията. Общият алгоритъм на КП операцията е описан в глава 2. Има допълнителни процедури за проверка за неконфликтност и коригиране на всяко от изчислена за траекторията както и осигуряване на комуникацията между агентите. Взаимодействието на всеки КП с обща област на паметта се дължи на необходимостта от получаване на данни за текущото местоположение на други агенти, както и информация за следващата изчислителна стъпка на всеки агент за проверка коректност и неконфликтност на формираната траектория. Както може да се види от Таблица 3-1, обменът на информация е организиран чрез глобалния двуизмерен масив “shared_paths”, съхраняван в паметта.

Всеки изчислителен процес на КП има достъп до всички стойности на масива “shared_paths”, както и до стойностите на скоростите (“spd”) и приоритетите (“P”) на агентите. Въз основа на тези данни, КП проверява всяка вероятна стъпка (“cmn2”) с подобни стойности, изчислени в други нишки, за да я провери за коректност и неконфликтност (Фигура 3-9) [93].

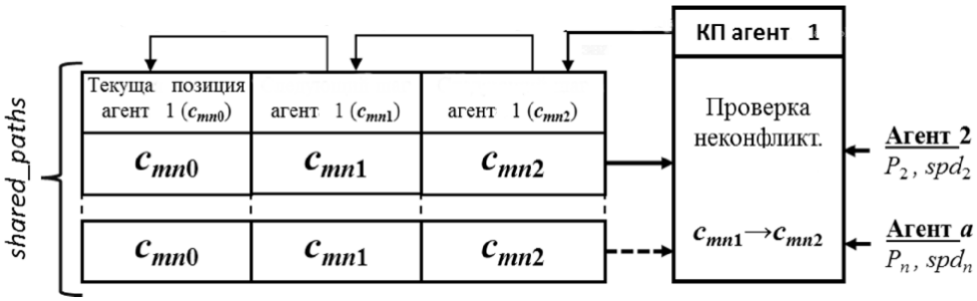
Основната разлика между стойностите “cmn1 = shared_paths[i][1]” и “cmn2 = shared_paths[i][2]” е, че стойността на следващата стъпка “shared_paths[i][1]” вече е проверена за коректност и ще да не се променя, докато като вероятна стъпка “shared_paths[i][2]” е стойността за проверка за предпазване от сблъскване, която вероятно ще бъде преизчислена.

Както се вижда от фигура 3-5, ако стойността на “shared_paths[i][2]” е преминала теста, тогава стойностите се изместват по низа на масива, тоест вероятната стъпка замества стойността на следващата стъпка от агент, а предишната стойност на “shared_paths[i][1]” става агент на текущата позиция.



Фигура 3-4. Блокова схема на алгоритъма на работа на програмния блок НМ

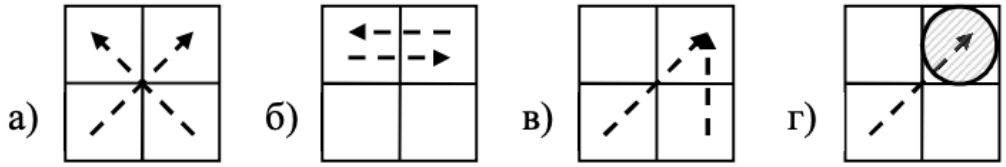
Изместването на стойностите се случва в момента, когато агентът започне да се движи към следващата изчислена безконфликтна позиция.



Фигура 3-5. Организация на масива „shared_paths“

3.3.2 Допълнителен модул „разрешаване на конфликтни ситуации“

Въз основа на избора ни за ортогоналната дискретизация на работното пространство се оформят четири вида конфликтни ситуации между двама агенти посочени графически на Фигура 3-6.

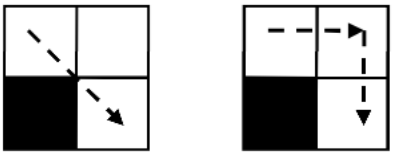


Фигура 3-6. Възможни конфликтни ситуации при траектории: а) кръстато; б) противоположно; в) комбинирано; г) пресичане на стабилната фиксирана позиция

Разрешаването на конфликтни ситуации (а), (б) или (в) ще се извърши с помощта на приоритетния вектор P: за агент с по-ниска стойност на приоритет (P[n]), преходът в тази (конфликтна) посока ще бъде блокирани, а траекторията съответно коригирана. Ако векторът P не е определен или приоритетите на агентите са еднакви, тогава и двата агента ще променят посоката на движение и ще коригират траекторията.

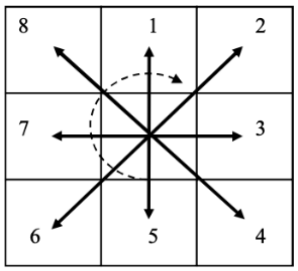
Ако един от агентите стои неподвижно, което е възможно, например, в случай на извънредна ситуация, в очакване на нова задача или при условие, че агентът е достигнал крайната точка на траекторията и върши някаква работа, тогава това ще се определи като "стабилна позиция" на този агент. Агент в стабилна позиция ще бъде възприет от другите членове на групата като пречка, така че всеки агент ще коригира траекторията, ако пресече тази позиция (г), независимо от стойността на приоритета.

В процеса на формиране на пътя е възможно агентът да премине през остър ъгъл на зоната на препятствието по време на диагонално движение. Тази ситуация също се идентифицира също като конфликт, което означава, че траекторията трябва се коригира както е показано на Фигура 3-7.



Фигура 3-7. Корекция на пътя, преминаващ през остър ъгъл

Както вече беше споменато в глава 2 (раздел 2.8.2 и 2.8.3), изчисляването на стъпката на траекторията се извършва чрез изчисляване на градиентите във всички възможни осем посоки (d) и определяне на максималната стойност „$grad_d$“. В този случай стойностите на градиента се изчисляват по посока на часовниковата стрелка с начална стойност на "12 часа" (Фигура 3-8).



Фигура 3-8. Процедурата за изчисляване на стойностите на градиентите

Тъй като разглежданите конфликтни ситуации възникват между най-близките агенти, за разрешаване на конфликта ще бъде достатъчно да се промени посоката на траекторията на един от агентите от текущата позиция. В този случай използването на алгоритми за динамична корекция на невронната карта (раздели 2.10, 2.11) е неподходящо и промяната в посоката на движение на текущата стъпка се извършва с помощта на КП чрез принудително задаване на градиента стойност в грешна посока към минималната й

стойност ($grad_d$ = –1) . Въз основа на (раздел 2.8.2), отрицателната стойност на градиента във всяка посока ще се интерпретира от системата като наличие на препятствие в тази посока:

$$grad_d = -1 \rightarrow E_d = 0,$$

където E_d е стойността на неврона в дадената посока.

В този случай стойностите на невронната карта (НК) няма да се променят, защото за да се коригира посоката на движение на текущата стъпка, пред робота се инсталира „виртуална пречка“.
Условното блокиране на диагоналното преместване има следния вид:

$$grad_d = -1 \rightarrow ((grad_{d-1} = -1) \vee (grad_{d+1} = -1)) \wedge (d \in \{1,3,5,7\}),$$

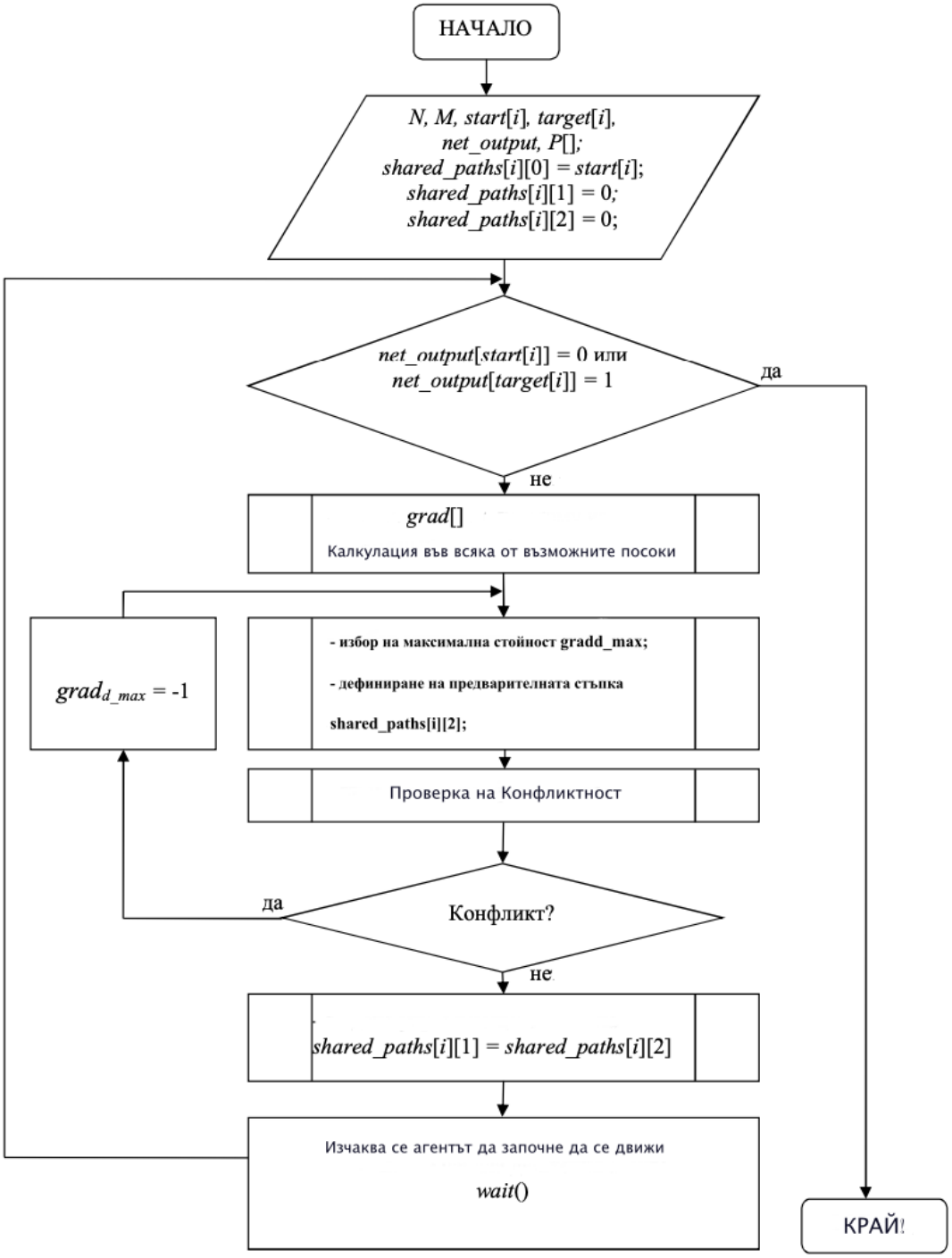
където d е индексът на диагоналната посока.

3.4 Експериментална постановка с контролер за навигация “HNav”

Този раздел е посветен на самата експерименталната част в дисертацията. Тук са описани проведените експерименти с модифицираният алгоритъм HMX представен в Глава 2. Основната цел е да се разкрият както предимствата така и възможните ограниченията при използването им при решаването на навигационни задачи в условия на „рояк“ мобилни роботи при различните условия и подходи. За проектиране и изпълнението на предложената система е използван мулти-агент базирани технологии [42] и програмен код за изграждане на алгоритъма за достигане на целта.
Разработеното многоннишково приложение (част от кода му е показан в **Приложение А** към дисертацията) - контролер за навигация “**HNav**” е съвместимо с навигационните техники заложени в операционната система ROS2 и включва модифицирана HMX - Раздел 2.8 с програмиран алгоритъм за избягване на препятствията в многонишкова комуникация между агентите.
Приложението представлява иновативна навигационна система (глобален мулти-агент плановик) за придвижване на група МР до цел в непозната среда, използвайки модифицирания алгоритъм на HMX представен в Глава 2. Алгоритъма включва КП с подход за избягване на препятствия средата в реално време. В отделните симулации по долу са описани само някой по-значими моменти от разработването на навигационния симулатор чрез среда „Webots“. Част от програмния код на КП, който е основата на навигационното приложение е приложен в **Приложение А**.

3.4.1 Тестове и експерименти на „HNAV” в различни ситуации

Опитната постановка включва изчисления при няколко различни сценария, където присъстват различен брой и конфигурация мобилни роботи, цели и препятствия за демонстриране на възможностите на навигационна система базирана на HMX под формата контролер за симулация на алгоритъма „**HNAV**“ в среда “**Webots**”.
В този раздел се представят резултатите от извършените експерименти.
Хардуерни спецификации на тестовия стенд върху който се извършват всички симулации има следната спецификация: Processor Intel(R) Xeon(R) - CPU X5660



Фигура 3-9. Блокова схема на алгоритъмът на робот в отделна изчислителна нишка (i) на КП

@ 2.80GHz (2 processors), RAM 24,0 GB; System type 64-bit operating system - win10. Графичният интерфейс за въвеждане на началните данни в програмата следва структурата от Таблица 3-2. След което ще бъдат установени следните параметри за агента (има експеримент само с един агент за първия пример):

Параметър	Стойност
Размер на работното пространство	20*20
Брой на агентиите (MP)	1

Скорост	2
Приоритет	
Начални позиции	1
Цели	381
Препятствия	101 – 219
Функция на активация	Линейна

Таблица 3-2. Меню за начални/входни данни за контролера "HNAV" и конфигурация на работното пространство.

Ако полетата за въвеждане на скорости и приоритети не са попълнени, скоростите и приоритетите на всички агенти автоматично ще бъдат зададени на 1.
Важно е да се отбележи, че мярката за скорост се приема като елементарна единица на дискретната работа в пространство, изминато за 1 s. Една елементарна единица е минимумът разстоянието между центровете на две близки дискретни клетки, т.е. посока в права линия.

Първо ще се провери правилността на програмата при планирането траекторията на един МР с едно препятствие в работното пространство, което се дефинира като размер предварително посочени в Таблица 3-2. След изпълнение на програмата ни от **Приложение А** в средата „Pycharm" с тези входни променливи, контролер "HNAV" връща изходни данни посочени на таблица 3-3.
Както се вижда от фигура 3-10, програмата работи правилно и изгражда маршрут, близък до най-краткия. Все пак трябва да се отбележи, че по-малката дължина на пътя не гарантира решаването на проблема с позиционирането за по-малко време, тъй като роботът променя посоката на движение и се нуждае от време, за да прави тези завои.

Фигура 3-10. КП с избягване на едно препятствие и използване на контролера "HNAV"

Брой стъпки	Брой завои	Дължина на пътя	Време (s)
40	15	47,04	31,15

Таблица 3-3. Изходни данни след симулацията в режим на един МР и едно препятствие в работно пространство 20x20

Времето за маневрата завой от своя страна е 0,4–0,5T, където T е времето преминаване към следващата позиция. Така при планирането траекторията трябва да се търси и цели намаляване броя на завоите. За да се изпълни това условие се налага добавяне на функция „контрол на посоката на движение" на МР (като функция „изглаждане"), което

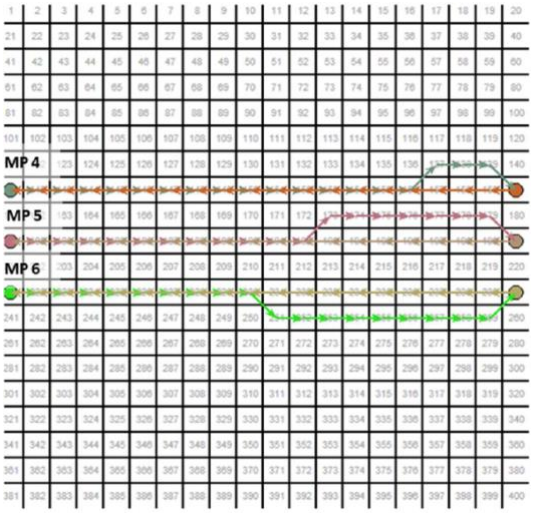
намалява броя на завъртанията на робота и по този начин „изглажда“ траекторията. Намаляването на броя на завоите се постига от факта, че при изчисляване на всяка следваща стъпка, роботът ще вземе предвид текущата си посока с по висок приоритет. Това условие се удовлетворява с помощта на КП, който, активира функцията „изглаждане“ и изчислява градиент от текущ посока $grad(i,j)_{cur}$ съгласно следната формула:

$$grad\ (i,j)_{cur} = (\ Ei - \ Ej) * (Wij * \sqrt{2}), \qquad (3.7)$$

При анализа на резултатите може да се направи следното заключение: *функцията за изглаждане* значително намалява броя на завъртанията на робота, което позволява да се получи печалба във времето от 5-10%, въпреки непроменена или дори малко по-голяма дължина на трасето. Затова в следващите експерименти, когато ще се разглеждат две и повече препятствия ще се вземе в предвид изведеното във формула 3.7 т.н. „изглаждане“. Нека сега се разгледа случай в който се търси път на група роботи с различна скорост в общо работно пространство с ситуация на конфликти и пресрещане – Таблица 3-4 с входните променливи, Фигура 3-11 с графичното представяне на пътя и Таблица 3-5 с изходните резултати).

Параметтър	Стойност
Размер на работното пространство	20*20
Брой на агентиите (MP)	6
Скорости	0.2, 0.5, 0.8, 1, 1, 1
Приоритет	
Начални позиции	200, 240, 141, 181, 221, 160
Цели	141, 221, 160, 200, 240, 181
Препятствия	няма
Функция на активация	Линейна
Изглаждашта функция	Включена

Таблица 3-4 с зададените входни променливи за Фигура 3-11



Фигура 3-11 - КП “HNAV” при многоагентен режим и възможни конфликти и взаимодействие на 2 групи агенти в насрещен трафик

Агент	Конфликтен ход	Дължина на пътя	Време (s)
1	-	19	99,39
2	-	19	39
3	-	19	25,33
4	156→157	19,82	22,5
5	191→192	19,82	22,5
6	230→231	19,82	22,5

Таблица 3-5. Изходни данни свързани с Фигура 3-11

Както може да се види от Фигура 3-1, роботите в групата отляво имат една и съща скорост, освен това по-висока от скоростта, на който и да е от роботите от групата вдясно. Приоритетите не са присвоени на агентите ръчно, но както е променихме в началото на раздела в този случай програмата присвоява приоритети на роботите автоматично, където в този случай приоритетът на робота е обратно пропорционален на скоростта му. На фигурата се вижда, че по-бързите роботи променят маршрута си, избягвайки насрещни сблъсъци, така че не се налага бавните агенти да изразходват време за маневриране, което намалява общото време за изпълнение на задачата за придвижване. Финалното изпълнението на самата задача се оценява по времето за изпълнение на най-бавния агент.

Нека сега да се разгледа работата ситуация на системата с работно пространство, където присъстват и препятствия (Фигура 3-12, Таблица 3-6 и 3-7).

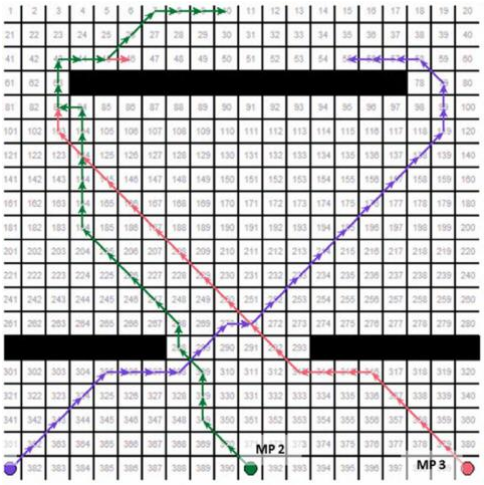
Параметтър	Стойност
Размер на работното пространство	20*20
Брой на агентиите (MP)	3
Скорости	2, 1, 2
Приоритет	1,3,2
Начални позиции	381, 291, 400
Цели	55, 10, 46
Препятствия	281-287, 294-300, 64-77
Функция на активация	Линейна
Изглаждашта функция	Включена

Таблица 3-6 с зададените входни променливи за Фигура 3-12

Както може да се види от Фигура 3-12, системата правилно изгражда без конфликти маршрути за група роботи, като се вземат предвид препятствията. Също така си струва да се отбележи необходимостта от коригиране на маршрута за агент 2 – въпреки факта, че е с най-висок приоритет в групата, при движение пресича крайната позиция на агент 3, който е достигнал местоназначението до този момент и така, че агент 2 коригира своя маршрут в зависимост от приоритета.

Агент	Конфликтен ход	Дължина на пътя	Време (s)
1	270→251	30,21	18,79
2	45→46	28,73	35,6
3	-	28,79	17,85

Таблица 3-7. Изходни данни свързани с Фигура 3-12



Фигура 3-12. Алгоритъма "HNAV" в условия на няколко МР и три препятствия

3.4.2 Експериментално сравнение на HNAV с класическите алгоритми за търсене

Използвайки модифицирания математически модел на невронна карта на Хопфийлд предложена в тази дисертация ще се извърши и експериментално сравнение с класическия алгоритъм за търсене A*, като се използва разработен софтуер с отворен код на написан на C# наличен в платформата за отворен код Github на адрес: https://github.com/yatima1460/AStar) (част от кода е приложен в **Приложение Б** като референция) и където за евристика на A* ще се използва функцията „разстояние Манхатън" [34].

Хардуерни спецификации на тестовия стенд е същия както до сега – Processor: Intel® Xeon® CPU X5660 @2.80GHz (2 processors), RAM24,0 GB; System type: 64-bit operating system win10.

Алгоритмите бяха сравнени в рамките на решаването на задачата за търсене траектории в четири различни (по сложност) конфигурации в работно пространства с размери 20×20:

- заобикаляне на просто препятствие;
- търсене на изход от стаята;
- търсене на траектория за заобикаляне на няколко препятствия;
- търсене на траектория за заобикаляне на няколко препятствия (вариант 2);

При решаването на тези проблеми броят на участващите неврони и възли графиката за намиране на траекторията се увеличава, когато конфигурацията става по-сложна препятствия. Резултатите от сравняването на производителността на алгоритмите са представени в таблица 3-8. Въз основа на представените резултати от компютърна симулация – чрез изпълнение на кода от **Приложение А** и **Приложение Б** в конфигурация на т.1-4, можем да се изведе извода, че времевата ефективност на алгоритъма за намиране на пътя на базата на модифицираната "неврона карта" на Хопфийлд и в частност представения контролер "HNAV" (кода в **Приложение А**) е *повече от 4 пъти по-висока от тази на алгоритъма A**, при относително еднакво качество на построената траектория т.е. решението и на двата алгоритъма е близко до оптималното.

Предимството на алгоритъма на невронната мрежа е **в скоростта на планиране поради**

факта, че поради оптимизацията на математическия модел на невронната мрежа – алгоритъмът използва само основни математически операции, работещи при това с прости типове данни (масиви и прости променливи), докато в алгоритъм A* използва ресурсоемки операции за сортиране и търсене в голяма база данни. От друга страна една допълнителна паралелизация на процесите на изчисления при A* няма да доведе до същите резултати както при изчислителна паралелизация на "неврона карта" на Хопфийлд.

Конфигурация на работното пространство	**Врене за решение на задачата в ms (бр на стъпките)**	
	Алгоритъм A*	**"неврона карта" на Хопфийлд**
задача 1	26,37 (33)	7,22 (33)
задача 2	28,57 (29)	6,95 (29)
задача 3	36,27 (28)	7,81 (27)
задача 4	39,86 (30)	8,23 (31)

Таблица 3-8. Резултати от сравняване на производителността на алгоритмите за планиране на пътя – A и модифицирана "неврона карта" на Хопфийлд.*

3.5 Резултати от извършените експерименти с контролера "HNAV"

В хода на теоретичната работа в Глава 2 и извършените експериментални изследвания в Глава 3 бяха получени резултати и изведени изводи на основата на анализа на основните подходи за управление на група от МР посочени по долу. Изследван бе въпросът за ефективността на различни алгоритми за планиране за решаване на проблема с позиционирането на група от МР. Въз основа на резултатите от анализа бе разработено софтуерно решение на системата за планиране на траекторията за групата МР на базата на невронна мрежа ‚Хопфийлд" (НМХ) наречен „HNav".

Разработения алгоритъм и архитектура на системата за планиране на навигация „HNav", взема предвид особеностите на движението на агент от групата на МР доказа своите предимства при условия динамично променящо се работно пространство и информирана недостатъчност.

Резултатите от модификациите на НМХ са представени в таблица 3-9 по-долу:

Модификация	**Резултат**
Подмяна на функцията за активиране	Подобряване на производителността на невронната мрежа до 2,5 пъти. Намаляване на риска от локални максимуми при сложни конфигурации на препятствия и голям броя на итерациите, необходими за изграждане на карта.
Изменение условията на активация	1,3 до 3 пъти намаление на Времето за активиране на НМ в зависимост от на разстоянието между агента и целта.

Опростяване на тегловата функция	Умерено използване на оперативни системна памет поради възможната “компресия” на матриците с коефициентите на тежест.

Таблица 3-9 – Резултати при извършените експерименти с навигационния контролер модифицирана “неврона мрежа” Хопфийлд “HNav”

1) Въз основа на експериментално изследване на невродинамиката на невронната мрежа ‚Хопфийлд“ беше извършена модификация на математическия модел на мрежата, по-специално: функцията на активиране на невроните беше заменена и условието за конвергенция на мрежата беше променено. Софтуерна симулация на системата за планиране на траекторията за един агент-робот, използващ модифицирана невронна мрежа ‚Хопфийлд“, показа, че модификациите позволяват да се сведе до минимум времето, необходимо за формиране на невронна карта.

2) Архитектурата и алгоритмите на системата за планиране са разработени, като се вземат предвид особеностите на движението на агент от групата МР в динамично променящо се работно пространство и информационна недостатъчност.

3) Предложен е метод за генериране и планиране на траектория за МР група на базата на универсален софтуерен модел, който осигурява изграждането на траектории на движение, като се вземат предвид зададените правила за безконфликтно взаимодействие на агентите в ортогонално дискретно работно пространство. Също така на базата на тези правила за взаимодействие е получен метод за синтезиране на траекториите на МР, което позволява да се получи допълнително съкращение във времето на движение до 10%.

4) След проведените експериментални симулации с предложената в Глава 2 модифицираната невронна мрежа ‚Хопфийлд“ под формата на код-програма на „Python“ в среда „Pycharm“ бе установено, че средната продължителност на времето за един цикъл на активиране е намалено до 1,5–2,5 пъти, а броят на циклите, необходими за формиране на невронна карта, намаля с 1,3–3 пъти (в зависимост от конфигурацията на работното пространство). Създаден е и виртуален стенд, на базата на който е извършен анализ на ефективността на предложените методи за оптимизиране на системата за планиране на невронни мрежи, алгоритми и техники. Визуализирани са резултатите от експериментална проверка на системата за планиране по време на работата на групата МР, която прилага централизирани и децентрализирани подходи за управление, въз основа на които може да се заключи, че предложената архитектура на системата за планиране, методите и алгоритмите за изчисляване на траекториите и по-специално логиката за преодоляване на конфликтни ситуации и избягване на препятствия работят правилно.

5) Оценка на изчислителната сложност и производителност на алгоритъма.
Определяне на класа на сложност на алгоритъма е функция на зависимостта на обема на изчислителните ресурси от обема на входни данни: O(f(X)) или границата на f(X), където $X \to \infty$, където X е измерението на входните данни. Извършва се оценка за „най-лошия случай“ на изпълнение алгоритъм, тоест с такъв набор от входни параметри, кога да се реши задача ще изпълни най-голям брой изчислителни операции. Чрез апроксимиране на общия обем изходни данни за разглеждания алгоритъм, може да се приеме, че

размерът на входните данни ще бъде определен стойности за вземане на проби от работното пространство M и N (размера по вертикал и хоризонтал), тогава приемаме:

$$X = M \times N$$

Въз основа на структурата на алгоритъма (Фигура 4.1) и цялото изчисление процесът може да бъде разделен на 2 последователно изпълнявани части от кода:

1. формиране на невронна карта;
2. изчисляване на траекторията,

Извършихме тези процедури чрез прилагане на подхода на многонишково и едновременно калкулиране на невронните карти на МР, осигурявайки отделно и паралелно изчисление за всеки отделен „HNAV“ контролер.

Ето защо по време на експериментите и постановката показа, че средно 53% от времето за изпълнение се пада на процедурата за пресмятане на невронната мрежа за всеки агент.

Ефективността на тази техника на изчисленията се характеризира с увеличаване на производителността (S), тоест съотношението на времето, прекарано в решаване на проблема с помощта на последователен алгоритъм, според времето, изразходвано за Решение, използващо паралелни изчисления – описани в Раздел 3.2.4. Последователната част на алгоритъма включва операции за въвеждане и инициализиране на данните - (Фигура 3-3). След това, вземайки дела на операциите за въвеждане на данни, равен на не повече от 5% (0,05), се определя f(s) за разглеждания алгоритъм, въз основа на (2.2).

При този подход, максималното постигнато ускорение на производителността на алгоритъма при клас на сложност $\mathbf{P} \rightarrow \infty$ ще бъде:

$$S = 1/f(s) = 1/0{,}05 = \mathbf{20}.$$

Следователно, при повишаване на броят процесори в системата, максималното (теоретично) повишаване на производителността на алгоритъма може да бъде до 20 пъти.

3.6 Изводи по глава три

Експериментално се установи, че предлагания КП-контролер на основата на модифицирана HMX, решава задачата за намиране на най-кратък път в дискретно работно пространство с ортогонална топология стабилно.

Въз основа на резултатите от експериментално изследване в Глава 3 беше доказано, че реализирането на математически модел на модифицираната HMX с нов подход за активиране на мрежата от неврони и, който отчита спецификата на хардуерната реализация успешно се справя с задачите:

1. промяна на условието за сближаване, в резултат на което се активират само невроните, необходими за изчисляване на траекторията, за изграждане на картата;
2. активиращата функция на невроните бе променена (опростена), което увеличи интензивността на енергийните взаимодействия между невроните на мрежата;

Успяхме да докажем очакваните предимствата на този подход - изисква по-малко изчисления и може да се използва за приложения за глобална навигация в дискретно пространство.

Създадената в хода на работата система за планиране на невронни мрежи има перспективи за по-нататъшно развитие, а разработените инженерни методи позволяват изграждането на такива системи за почти всички софтуерни и хардуерни платформи на бордови информационни и контролни системи МР. Характерна особеност на предложеното решение е неговата специализация за групово придвижване на роботи.

Оценка на изчислителната сложност и възможност за подобряване на производителността на алгоритъма „HNAV“ демонстрира добра възможност за оптимизация чрез внедряването на техники за паралелна работа на процесите в

контролера и използвайки възможностите на езика “Python” в многонишкови разпределени изчисления.
В хода на експериментите се установява, че при кратък път, малки пространства и в ситуации с голям брой препятствия и брой агенти се появяват много на брой локални максимуми и намирането на път може да се окажат проблем и за НМХ. В такива ситуации може да се:

- установи условност в мулти-агентна система – установяване на максимален бр. от мобилни роботи за определената общата площ на работното пространство;
- съчетае с алгоритъм за търсене като ИПП (описан детайлно в Глава 1).

Това ще възпрепятства трайно блокиране на планера в локалните минимуми. Т.е. възможно е МР да открие локален минимум но това няма да го изложи на опасност от „засядане“, тъй като ще се включи още един подход за търсене на път и техника да го изведе от това положение.

БЪДЕЩО РАЗВИТИЕ

В днешни дни мобилните роботи изпълняващи мисии в условия на корперативност и информационна недостатъчност ще изпитват все по-голяма нужда от бързодействие при получаване на пълна и детайлна картина към случващото в работното пространство. Това води до нужда от все по-бърз обем от данни в реално време между роботите и от там необходимост до стимулационни стендове, на които да се демонстрират възможните сценарии и извършат изпитанията. Предложената в дисертационния труд *хибридна система и архитектура за навигация* с избягване на препятствията може да се използва в напълно функционираща система и среда, в която да се симулира самообучаваща се среда и различни алгоритмите за навигация и контрол на множество МР.

Използвания симулатор „Webots“ в среда и операционна система ROS2 и разработения алгоритъм предлага бъдещи възможности за симулации за постигане на оптимизация в бъдещи сценарии в условията на информационна недостатъчност.

Разработения комуникационен сценарии с предложения в Глава 3 контролер „HNAV“ може да се използва и за експериментални сценарии като:

– локална навигация с възможно управление на траекторията на движение на членовете на групата да си поставят цели;

– отчитане на енергийните ресурси (както собствени, така и на групата като цяло), когато се решава проблема с планирането на траекторията;

– решение на различни задачи за локализация и картография (SLAM).

Като част от по-нататъшни изследвания за алгоритъма на НК „Хопфийлд“ може да се планира бъдеща разработка алгоритъм за планиране на динамично усилване синаптични връзки, използвани за всеки мрежов домейн поотделно. Тази модификация на системата на невронната мрежа ще позволи да се изчисли траекторията с отчитане на различната проходимост при конкретните участъци в дискретно работно пространство, т.е. в реални условия като се вземат предвид терена, вида на почвата и т.н.

Освен това може да се предвижда усъвършенстване на представената методология в условия на планиране траекторията в триизмерно работно пространство с потенциал за развитие в областта на управление на многоженство безпилотни апарати (дронове) в малки пространства.

При сценарии на локална навигация въвеждането ИПП съвместно с алгоритъма на „невронна карта“ за управление движението на МР може да бъде допълнително имплементирано с цел оптимизиране на калкулациите при управление на движението на много голям брой агенти в динамично в сложно работното пространство с много препятствия. ИПП има добавената стойност при подобряване фалшивите положителни резултати, ускоряване на сходимостта на процеса на създаване на навигационния път и по този начин занижаване на изискванията към изчислителна достатъчност необходима при ситуации с много препятствия и МР-ти.

ПРИНОСИ

ЗАКЛЮЧЕНИЕ - РЕЗЮМЕ НА ПОЛУЧЕНИТЕ РЕЗУЛТАТИ

В заключение следва да се отбележи, че в съответствие с целта и поставените задачи в дисертационния труд е извършен анализ на основните подходи за управление на група мобилни роботи, като е изследван въпросът за ефективността на различни алгоритми за планиране и решаване на проблема с позиционирането на група от МР. Получени са следните научни, научно-приложни и приложни резултати, които се представят в резюме според изискванията на чл. 27 (2) от Правилника за приложение на ЗРАСРБ:

НАУЧНИ

- Предложен е подход за оптимизиране на обучението на НМХ свързано със създаването на НК, която може да се ползва при навигационни задачи – *Раздел 2.9.1;*
- Предложено е оригинално решение, а именно: метод за изчисление на НМХ в дисертацията, като комбинация от свойства на универсалност, гъвкавост и мащабируемост, необходими за ефективното и бързо решаване на проблема с груповия контрол в „рояк“, където е различно съотношението на броя цели и роботи – *Раздел 2.10 и 2.11*;

НАУЧНО-ПРИЛОЖНИ

- Разработен е нов, по-систематичен и ефективен модел за навигация и координация при МР в група чрез НМХ - контролер „HNav“ – *Раздел 2.9, 2.10 и 2.11;*
- Реализиран е нов модел планиране на пътя за МР в група в децентрализиран или хибриден режим на работа на основата на мулти-агенти базирани на *автоасоциатор и модел на „Хопфийлд“*. Този контролер - алгоритъм може да се изпълнява на многопроцесорни и многоядрени нови хардуерни платформи като: ARM и RISC-V. Новия модел предлага по-добра мащабируемост, гъвкавост и пълна сигурност използвайки възможностите на многонишковото програмиране и управление/програмиране на основата на събития (event-driven) – *Раздел 2.4, Фигура 2-2, [AARTON2021];*
- Доказано е предимството на демонстрирания модел по отношение на възможните грешки при навигация. Използването на поведенчески техники за анализ елиминира високия процент на неверни данни. Елиминира необходимостта от боравене с огромното количество данни, необходими, за да се самообучават отделните МР чрез паралелна архитектура за изчисления на предложения алгоритъма – *Раздел 2.6 и 2.7;*
- Доказано е експериментално ефективността на надграждането на мулти-агентна система, при което е реализирано значително подобрение на координацията между МР, бързодействието и времето за реализиране на алгоритъма – *Раздел 2.8 и 2.9.*;
- Разработен е алгоритъм, които симулира ефективността на предложения хибриден модел – „HNav“. Изградени са различни сценарии за симулация в средата и ОС за роботи които лесно могат да се внедрят и интегрират в ОС за роботизирани системи - „ROS” – *Глава 3.*

ПРИЛОЖНИ

- Представени са архитектура и алгоритми за работа на системата за планиране, като са взети предвид особеностите на движението на агент от групата МР в

условия на динамично променящо се работно пространство и информационна недостатъчност – *Фигури 2-3 и 2-15*;

- Разгледана е възможността за решаване на навигационни задачи при разпределени цели и децентрализирано и хибридно управление на определените групи роботи, благодарение на симулационната среда чрез, която се анализират резултатите на един или друг робот по отношение на критериите за оценка на ефективността на избрания предварително път до целта – *Раздел 2.8.5;*
- Приложен е метод за синтезиране на система за планиране на траекторията на група МР за базата на универсален софтуерен модел, който осигурява изграждането на траектории на движение с отчитане на зададените правила. Въз основа на представените правила за взаимодействие е получен метод за синтезиране на траектории, който намалява времето на движение до 10% - *Формула (3.7), Раздел 3.4.1, Фигура 3.10, Таблица 3.3;*
- Проведени са експерименти с моделиране на модифицираната неврона мрежа Хопфийлд, в резултат на което е установено, че средната продължителност на цикъла на активиране е намален до 1,5–2,5 пъти, а броят на циклите, необходими за формиране на невронна карта, са намалени с 1,3–3 пъти (в зависимост от конфигурацията на работното пространство) поради използването на частична активация на невроните в НК (частичната корекция на картата**)** – *Глава 2, Фигура 2-1;*
- Приложена е модификация на математическия модел на мрежата, чрез замяна на функцията на активиране на невроните и условието за конвергенция на мрежата, при което софтуерното моделиране на система за планиране на траектория за един робот-агент с езика phyton, демонстрира възможност да се достигне теоретично до 20% допълнителна оптимизация във времето за изпълнение на алгоритъма при повишаване на броят процесори в системата – *Раздел 3.5*

Създадената по време на работа *система за планиране и навигация* има перспективи за по-нататъшно развитие, а разработените техники позволяват изграждането на подобни системи за почти всички софтуерни и хардуерни платформи на бордови информационни и контролни системи при управлението на група МР (swarm).

ИЗПОЛЗВАНА ЛИТЕРАТУРА

1. Reynolds C. Flocks, Herds, and Schools: A Distributed Behavioral Model, ACM SIGGRAPH Computer Graphics. 1987. 21(4). Pp. 25-34.
2. Spears W.M., Spears D.F., Heil R., Kerr W., Hettiarachchi S. An Overview of Physicomimetics, Lecture Notes in Computer Science. 2005. Vol 3342. Pp. 84–97.
3. Matari ́c, M. J. Designing and Understanding Adaptive Group Behavior, Adaptive Behavior. 1995. 4(1). Pp. 51—80. 1995
4. Duarte M., Costa V., Gomes J., Rodrigues T., Silva F., Oliveira S., Christensen A. Evolution of Collective Behaviors for a Real Swarm of Aquatic Surface Robots, pLoS ONE. 2016.
5. Schmickl T., Thenius R., Moslinger C., Timmis J., Tyrrell A., Read M., Hilder J., Halloy J., Campo A., Stefanini C., Manfredi L., Orofino S., Kernbach S., Dipper T., Suta-tyo D. CoCoRo - The self-aware underwater swar— // SASO 2011—- Fifth IEEE International Conference on Self-Adaptive and Self- Organizing Systems. 2011. Pp. 120–126.
6. Bayindir L., Sahin E. A review of studies in swarm robotics, Turkish Journal of Electrical Engineering and Computer Sciences. 2007. 15. Pp. 115-147.
7. Brambilla, M., Ferrante, E., Birattari, M. et al. Swarm robotics: a review from the swarm engineering perspective, Swarm Intelligence. 2013. 7. Pp. 1–41.
8. Gutelius B., Theodore N. The Future of Warehouse Work: Technological Change in the U.S. Logistics Industry, UC Berkeley Labor Center. 2019.
9. Spears, W.M., Spears, D.F., Hamann, J.C. et al. Distributed, Physics-Based Control of Swarms of Vehicles, Autonomous Robots. 20—. 17. Pp. 13——162.
10. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science, pp. 39-43. IEEE service center, Piscataway (1995)
11. Ellips Masehian, and Davoud Sedighizadeh, Classic and Heuristic Approaches in Robot Motion Planning – A Chronological Review, Ellips Masehian, and Davoud Sedighizadeh
12. Thomas J. Watson, T. Lozano-Perez, M. A. Wesley, An Algorithm for Planning Collision-free Paths Amongst Polyhedral Obstacles. ACM 22, (1979) pp. 560-570
13. J.F. Canny, "The Complexity of Robot ". (1988), MIT Press, Cambridge, Mass.
14. E. Şahin, “Swarm robotics: from sources of inspiration to domains of application,” in Swarm Robotics Workshop: State-of- the-Art Survey, E Şahin and W. Spears, Eds., Lecture Notes in Computer Science, no. 3342, pp. 10–20, Berlin, Germany, 2005.
15. I.A. Kalyaev, A.R. Gaiduk, S.G. Kapustyan., Models and algorithms of collective control in groups of robots, FIZMATLIT, 2009.
16. Masehian, E, and Amin Naseri, M. R. “A Voronoi diagram-visibility graph-potential field compound algorithm for robot path planning”, J. Rob. Sys. Vol. 21, No6, (2004), pp. 275-300
17. Asano, T., Asano, T, Guibas, L., Hershberger, J.“ and Imai,”H.“"Visibility-polygon search and Euclidean shortest path”, 26th Symp. Found. Comp. Science (1985) pp. 155-164
18. J. Borenstein, H.R. Everett, L. Feng, and D. Wehe, Mobile Robot Positioning: Sensors and Techniques, Invited paper for the Journal of Robotic Systems, Special Issue on Mobile Robots. Vol. 14 No. 4, Page(s): 231 – 249
19. Mohinder S. Grewal, Lawrence R. Weill, Angus P. Andrews, Global Positioning Systems, Inertial Navigation, and Integration, John Wiley & Sons, Inc. 2001, ISBN: 0-471-20071-9
20. Kennedy, J., and Eberhart, R. (1995). Particle Swarm Optimisation. In: Proceedings of the IEEE Conf. on Neural Networks, Perth.
21. Mondada, F., Bonani, M., Magnenat, S., Guignard, A. and Floreano, D., Physical connections and cooperation in swarm robotics, In Proceedings of the 8th Conference on Intelligent Autonomous Systems (IAS8), March 10-14, 2004, IOS Press, Amsterdam, NL, Page(s): 53-60
22. Y. Uny Cao, Alex S. Fukunaga, Andrew B. Kahng, Cooperative Mobile Robotics: Antecedents and Directions, Autonomous Robots, Vol. 4, Page(s):1–23 (1997)
23. В. Павлов, Р. Димитрова, Г. Павлова, „Интелигентни роботи и автоматизация на производството“, XXV МНТК „АДП-2016”.
24. В.И.Павлов, В. Николов, И.Чавдаров Ал.Вацкичев, „Системен подход в проектирането на мобилни роботи“, сб. Доклади “Роботика и мехатроника”, Дряновски манастир, 12-14

октомври, 2005.
25. Ullrich R., The Robotics Primer., New Jersey, Prentice-Hall, 1983
26. Vladimir L. Lumelsky, Sensing, intelligence motion: how robots and humans move in an unstructured world, John Wiley & Sons, Inc, 2006, ISBN 0-471-70740-6
27. J. Schwartz, J. Hopcroft, M. Sharir, eds. Planning, Geometry, and Complexity. Robot Motion Aspects, Ablex Publishing Corporation, Norwood, NJ, 1986, ISBN: 0893913618
28. M. LaValle, Planning Algorithms, Cambridge University Press 2006, ISBN 0-521-86205-1
29. J. Reif, Complexity of the Mover's Problem and generalizations, In Proceedings, of the 20th Symposium of the Foundations of Computer Science, 1979
30. J. Schwartz and M. Sharir. On the Piano Mover's problem. II. General techniques for computing topological properties of real algebraic manifolds. Advances in Applied Mathematics, Vol. 4, Pages: 298–351, 1983
31. De-Shuang Huang, Laurent Heutte and Marco Loog, Artificial Potential Field Based Path Planning for Mobile Robots Using Virtual Water-Flow Method, Advanced Intelligent Computing Theories and Applications. With Aspects of Contemporary Intelligent Computing Techniques, Springer Berlin Heidelberg 2007
32. S. S. Ge and Y. J. Cui, Dynamic Motion Planning for Mobile Robots Using Potential Field Method, Autonomous Robots, 2002
33. Diéguez, A. R., Sanz, R., and López, Deliberative On-Line Local Path Planning for Autonomous Mobile Robots, Journal of Intelligent and Robotic Systems, Vol. 53 , Issue 2 (October 2008), Pages: 145 – 168, ISSN:0921-0296
34. Roland Geraerts and Mark H. Overmars, The Corridor Map Method: Real-Time High-Quality Path Planning, IEEE International Conference on Robotics and Automation, Roma, Italy, 10-14 April 2007
35. A. Kamphuis and M. Overmars, Finding paths for coherent groups using clearance, in Eurographics/ ACM SIGGRAPH Symposium on Computer Animation, 2004, Page(s): 19-28
36. J. Barraquand, L. Kavraki, J.-C. Latombe, T.-Y. Li, R. Motwani, and P. Raghavan, A random sampling scheme for path planning, International Journal of Robotics Research, vol. 16, pages 759-744, 1997
37. Hart, P. E.; Nilsson, N. J.; Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems Science and Cybernetics, Vol. 2, Page(s): 100–107, 1968
38. Dechter, Rina; Judea Pearl, Generalized best-first search strategies and the optimality of A*, Journal of the ACM, Vol. 32, Pages: 505–536, 1985
39. A. Kamphuis and M. Overmars, Finding paths for coherent groups using clearance, in Eurographics/ ACM SIGGRAPH Symposium on Computer Animation, 2004, Page(s): 19-28
40. Correll, N. & Martinoli A. Collective Inspection of Regular Structures using a Swarm of Miniature Robots, Proc. of the Ninth Int. Symp. on Experimental Robotics ISER-04, Singapore. Springer Tracts in Advanced Robotics 6, Vol. 21, 2006
41. J. Spletzer, A. K. Das, R. Fierro, C. J. Taylor,V. Kumar, and J. P. Ostrowski, Cooperative Localization and Control for Multi-Robot Manipulation, Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems 2001, Vol. 2, 2001 Page(s):631 – 636
42. G. Tsochev, Roumen Trifonov, G. Popov, FIPA Standardization for Agent Based Technology, Computer&Communications Engineering V.10/1, 2016, ISSN 1314-2291, pp 32-34
43. O.V. Darintsev, Management system for a team of microrobots, newspaper: Artificial intelligence. 2006
44. Б.Г., Ильясов "Интеллектуальные системы управления. Теория и практика.", Москва: Радиотехника, 2009
45. A.B. Migranov, "Different approaches to control the movement of mobile robots based on soft computing technologies", 2012.
46. HopfieldJ.J. Neural Networks and Physical Systems with Emergent Collective Computational Abilities // Proceedings National Academy of Sciences. — 1982. — С. 2554-2558.
47. "ROS.org | About ROS," library Catalog: www.ros.org. [Online]. Available:

https://www.ros.org/about-ros/
48. ROS Robot Programming, First Edition, Published ROBOTIS Co. Ltd., ISBN: 979-11-962307-1-5, Authors: YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, TaeHoon Lim
49. K. Zheng, "ROS Navigation Tuning Guide," arXiv preprint arXiv:1706.09068, p. 23, 2017.S.
50. Rajasekaran и G. A. Vijayalakshmi Pai, Neural networks, fuzzy logic and genetic algorithm: synthesis and applications, phi learning pvt. ltd., 2003.
51. П. Георгиева, „Генетичните алгоритми като средство за решаване," Списание „Компютърни науки и комуникации", том 2, No 3, pp. 16-23, 2013.
52. Zhao, Y., Zheng, Z., Liu, Y. Survey on computational-intelligence-based UAV path planning. Knowledge-Based Systems. 2018. Vol. 158. P. 54–64.
53. Patle, B. K., Babu L, G., Pandey, A., A review: On path planning strategies for navigation of mobile robot. China Ordnance Society, 2019. 582–606 p.
54. Gazi V. Swarm Aggregations Using Artificial Potentials and Sliding Mode Control. Proc. of the IEEE Conf. on Decision and Control. Maui, Hawaii, 2003
55. Барбашова Т.Ф., Кирильченко А.А., Колганов М.А. Некоторые аспекты использования метода потенциалов при управлении мобильными роботами. – М.: Изд-во ИПМ им. М.В. Келдыша РАН, 2004.
56. Рыжова Т.П. Управление коллективом мобильных роботов . Труды международной научно-технической конференции "Экстремальная робототехника". – 2011..
57. Пшихопов В.Х., Медведев М.Ю. Управление подвижными объектами в определенных и неопределенных средах. – М.: Наука, 2011. – 350 с.
58. Mehdi Mouad, Lounis Adouane, Pierre Schmitt, Djamel Khadraoui, Philippe Martinet. Architecture Controlling Multi-Robot System using Multi-Agent based Coordination Approach. Submitted on 27 Feb 2018 , LAboratoire des Sciences et Mate´riaux pour l'Electronique et d'Automatique, CNRS Campus des Ce´zeaux Aubiere, FRANCE
59. Ozen, O., Sariyildiz, E., Yu, H., та ін. Practical PID controller tuning for motion control: Proceedings – 2015 IEEE International Conference on Mechatronics, ICM 2015, IEEE Inc., 09.April.15. P. 240–245.
60. R. Glasius — Nijmegen. Trajectory Formation and Population Coding with Topographical Neural Networks. Katholieke Universiteit Nijmegen, 1997.
61. O.V.Darintsev and A.B.Migranov, Using the Hopfield Neural Network to Select a Behaviour Strategy for the Group of Mobile Robots, 2021
62. P.E. Keller, K.L. Priddy. Artificial Neural Networks: An Introduction USA, Washington: SPIE, 2005.
63. Ionut B. Brandusoiu, Gavril I. Toderean, How to fine-tune neural networks for classification, The general association of engineers in Romania, Copyright © Authors, 2020
64. Simon X. Yang, Max Meng, "An efficient neural network approach to dynamic robot motion planning", Neural Networks, 2000.
65. Bueckert J, Yang SX, Yuan X, Meng MQH. Neural dynamics based multiple target path planning for a mobile robot. In: 2007 IEEE Inter- national Conference on Robotics and Biomimetics (ROBIO); 2007 Dec 5-18; Sanya, China. IEEE; 2007
66. Rojas R. Neural Networks - A Systematic Introduction, Berlin, New-York. Springer-Verlag, 1996
67. Subhrajit Bhattacharya, Siddharth Talapatra, Robot Motion Planning Using Neural Networks: A Modified Theory, Department of Mechanical Engineering, IIT Kharagpur
68. Li H, Yang SX, Biletskiy Y. Neural network based path planning for a multi-robot system with moving obstacles. In: 2008 IEEE International Conference on Automation Science and Engineering; 2008 Aug 23-26; Arlington, USA. IEEE; 2008.
69. Yuan X, Yang SX. Multirobot-based nanoassembly planning with automated path generation. IEEE ASME Trans Mechatron, 2007.
70. Zhu A, Cai G, Yang SX. Theoretical analysis of a neural dynamics based model for robot trajectory generation. In: IEEE 2002 International Conference on Communications, Circuits and Systems and West Sino Expositions; 2002 Jun 29-Jul 1; Chengdu, China. IEEE; 2002.
71. Ni J, Yang SX. Bioinspired neural network for real-time cooperative hunting by multirobots in unknown environments. IEEE TransNeural Netw 2011.

72. Shaorong Xie,"The obstacle avoidance planning of USV based on improved artificial potential field," in *IEEE International Conference on Information and Automation (ICIA)*, Hailar, China, 2014
73. Lagoudakis M.G. Mobile Robot Local Navigation with a Polar Neural Map, The Center for Advanced Computer Studies University of Southwestern Louisiana.
74. Johansson S.J., Johan Hagelbäck J. Using multi-agent potential fields in real- time strategy games, Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems. — Estoril, 2008. — C. 631-638.
75. P.E. Keller Artificial Neural Networks: An Introduction USA / P.E. Keller, K.L. Priddy — Washington: SPIE, 2005. — 180 c.
76. Warren S. Mcculloch, Walter Pitts, A logical calculus of the ideas immanent in nervous activity, Bulletin of mathematical biophysics, Vol. 5, 1943
77. Rosenblatt F. Principles of neurodynamics: perceptrons and the theory of brain mechanisms . Washington: Spartan Books, 1962. — 616 c.
78. S. Papert, M. Minsky, Perceptrons: An Introduction to Computational Geometry. Cambridge: MIT Press, 1969. — 258 c.
79. Jianchang Mao, Mohiuddin K.M., Anil K. Jain Artificial Neural Networks: A Tutorial. Computer. — 1996. — T. 29, No 3. — C. 31-44.
80. Kohonen T. Self-Organized Formation of Topologically Correct Feature Maps. Biological-Cybernetics. — 1982. — C. 59-69.
81. HopfieldJ.J. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. Proceedings National Academy of Sciences, 1982
82. Miyake S., Fukushima K. Neokognitron: A New Algorithm of Pattern Recognition Tolerant of Deformations an Shifts in Position // Pattern Recognition, 1982..
83. Hecht-Nielsen R. Counterpropagation networks // Applied Optics. 1987
84. Nvidia Developer, Convolutional Neural Network (CNN), https://developer.nvidia.com/discover/convolutional-neural-network, посетен януари, 2022
85. Md Shad Akhtar, Abhishek Kumar, Deepanway Ghosal, Asif Ekbal, Pushpak Bhattacharyya, A Multilayer Perceptron based Ensemble Technique for Fine-grained Financial Sentiment Analysis, Conference on Empirical Methods in Natural Language Processing, Sep/2017
86. E.S. Ramakrishna, C.W. Ahn , A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations. Evolutionary Computation, IEEE Transactions. 2002. C. 566 - 579.
87. Mohammed Al-DahhanMohammad, Muzakkir AliMohammad Muzakkir Ali, Path tracking control of a mobile robot using fuzzy logic, 13th International Multi-Conference on Systems, Signals & Devices (SSD), 2016.
88. Srinivas Tennety, Saurabh Sarkar, Ernest L. Hall, Manish Kumar. Support vector machines based mobile robot path planning in an unknown environment, Department of Computer Science, Department of Mechanical Engineering. University of Cincinnati, October 2009, DOI: 10.1115/DSCC2009-2703
89. J. Miura, 2006. “Support Vector Path Planning”. International Conference on Intelligent Robots and Systems, IEEE/RSJ, Beijing, China.
90. Akihiro Kishimoto, Kiyohito Nagano, Evaluation of Auction-Based Multi-Robot Routing by Parallel Simulation, Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2016)
91. M L Patkin and G N Rogachev, Construction of multi-agent mobile robots control system in the problem of persecution with using a modified reinforcement learning method based on neural networks, 2017 Workshop on Materials and Engineering in Aeronautics (MEA2017)
92. A.V.Timofeev, Multi-agent navigation and intelligent motion control of robots in a dynamic environment with obstacles, International Exhibition-Congress "Mechatronics and Robotics 2007", St. Petersburg, 2007.
93. O.V. Darintsev, A.B. Migranov, B.S. Yudinets. “A neural network algorithm for planning trajectories for a group of mobile robots”. Institute of Mechanics, Scientific Center RAS; Fima State Aviation Technical University, Ufa, Russia - http://dspace.nbuv.gov.ua/bitstream/handle/123456789/58823/19-Darintsev.pdf?sequence=1
94. Lippmann R.P., An introduction to computing with neural nets, ASSP Magazine, 1987, No4,

p. 3-21.
95. Hebb D.O. The Organization of Behaviour, New York: Wiley & Sons, 1949
96. Dorigo, M., Maniezzo, V., and Colorni, A., The Ant Sys- tem: Opti- mization by a Colony of Cooperating Agents, IEEE Trans. Systems, Man Cybernetics, Part B, 1996, vol. 26, no. 1, pp. 29–41.
97. ChristianBlum, Antcolonyoptimization:Introductionandrecenttrends, ALBCOM, 11 October 2005
98. O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” The international journal of robotics research, vol. 5, no. 1, pp. 90–98, 1986.
99. Yingchong Ma, Gang Zheng, Wilfrid Perruquetti, Zhaopeng Qiu, Motion planning for non-holonomic mobile robots using the i-PID controller and potential field, IEEE/RSJ Conference on Intelligent Robots and Systems (IROS), 2014, Sep 2014, Chicago, United States. hal-01071506
100. Intel Guide for Developing Multithreaded Applications, Intel, Published: 01/16/2012 , Last Updated: 06/01/2015,URL: https://www.intel.com/content/www/us/en/developer/articles/guide/guide-for-developing-multithreaded-applications.html?wapkw=Nuts%20and%20bolts%20of%20multithreaded%20programming (Accessible March, 2022)
101. Threads and threading, Microsoft, Article, 09/15/2021, URL: https://docs.microsoft.com/en-us/dotnet/standard/threading/threads-and-threading (Accessible March, 2022)
102. An Introduction to Asynchronous Programming in Python, Velotio Technologie, Aug. 24, 2018, URL: https://medium.com/velotio-perspectives/an-introduction-to-asynchronous-programming-in-python-af0189a88bbb (Accessible March, 2022)
103. Prof. Plamenka Borovska, TU-Sofia, available online URL: https://schupen.net/lib/tu/KST_all/Semesters/semestar%206/VPKS/Lections/PAR_PERF_BG.pdf (Accessible March 2022)
104. Khadir Besseghieur, Radosław Trebinski, Wojciech Kaczmarek, Jaroslaw , Leader-follower formation control for a group of ROS-enabled mobile robots, 6th International Conference on Control, Decision and Information Technologies (CoDIT), 2019
105. Xiaojing Fan, Yinjing Guo, Hui Liu, Bowen Wei, Wenhong Lyu, Improved Artificial Potential Field Method Applied for AUV Path Planning, Mathematical Problems in Engineering. Volume 2020, Article ID 6523158, https://doi.org/10.1155/2020/6523158
106. ROS2 Programming Interface for the E-puck2 Robo, Author: Darko Lukic, SS 2020, MT-S, No. DISAL-MP44, EFPL, Access link: https://disalw3.epfl.ch/teaching/student_projects/ay_2019-20/ss/CYBERBOTICS-DISAL-MP44_summary.pdf
107. M. Kashyian, S. L. Mirtaheri, and E. M. Khaneghah, “Portable Inter Process Communication Programming,” in 2008 The Second Interna- tional Conference on Advanced Engineering Computing and Applica- tions in Sciences, Sep. 2008, pp. 181–186.
108. Business Rules Management System (BRMS) “Drools”, https://www.drools.org
109. Eventerpretor - an open-source node-based, lightweight Complex Event Processor (CEP), https://github.com/andreas-seiderer/Eventerpretor
110. TaskManagerIO - library for Arduino, https://github.com/davetcc/TaskManagerIO/
111. Inter-process-communication (IPC) middleware for various operating systems – “AceOrix”, Eclipse.org, https://projects.eclipse.org/projects/technology.iceoryx, https://github.com/eclipse-iceoryx/iceoryx , https://iceoryx.io/v2.0.2/
112. Michel, O. / Cyberbotics Ltd - WebotsTM: Professional Mobile Robot Simulation, pp. 39-42, International Journal of Advanced Robotic Systems, Volume 1 Number 1 (2004), ISSN 1729-8806

СПИСЪК С ПУБЛИКАЦИИТЕ ПО ДИСЕРТАЦИОННИЯ ТРУД

1. [AARTON2019] Antouan Anguelov, Roumen Trifonov, Ognian Nakov, Emerging and secured mobile ad-hoc wireless network (MANET) for swarm applications, BCI'19: Proceedings of the 9th Balkan Conference on Informatics. September 2019 Article No.: 9. Pages 1–4. https://doi.org/10.1145/3351556.3351557

2. [AARTON2021] Antouan Anguelov, Roumen Trifonov, Ognian Nakov, Analysis of swarm application layer protocols (SALP) used in event-driven communication, 30 Sept.-2 Oct. 2021, Varna. International Conference Automatics and Informatics (ICAI). DOI: 10.1109/ICAI52893.2021.9639663

3. [AARTON2020] Antouan Anguelov, Roumen Trifonov, Ognian Nakov,Colony Intelligence for Autonomous Wheeled Robot Path Planning, 28th National Conference with International Participation (TELECOM). 29-30 Oct. 2020. DOI: 10.1109/TELECOM50385.2020.9299536

4. [AARTONCS2020] Antouan Anguelov, Roumen Trifonov and Ognian Nakov, Protocol Stack for Hybrid Short-Range Modular Wireless Transceiver, 9th International Scientific Conference, CS-20: International Scientific Conference Computer Science'2020, 18-21 October, 2020

5. [AARTONTU2021] Antouan Anguelov, Roumen Trifonov, Ognian Nakov, PROTOCOL ANALYSIS FOR SHORT-RANGE TRANSCEIVERS, Computer and communications engineering, Vol. 15, No. 1/2021

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А:

Контролер „HNAV“ – NMH.py

```
# space = 0 # матрица с индексите/номерата на работните пространства
#unit_id = 0 # идентификатор на агента
M = 10# вертикален размер на мрежата (ред)
N = 10 # хоризонтален размер на мрежата (колона)
#start_point = 91 # индекс на стартовата клетка (където се намира агента)
#target_point = 56 # индекс на целевата клетка (цел)
start = (0, 0) # # матричен индекс на началната клетка
k=1     # коефициент на усилване
actf_type='linear'  # избор linear или tanh тип функция активация
target = (5, 5)     # матричен индекс на целевата клетка (ред, колона) / (row, col)
dw = (1/(math.sqrt(2)))*k # тегло на връзката по диагонал * 1/2
sw = 1*k #тегло на възката по права
scw = 0*k #тегло на връзката към самия себе си неврон
#smoothing = 0 # включване/изключване на функцията за "изглаждане" на траекторията
#smooth = sw - dw # коефициент на "изглаждане" на траекторията
E = [[0]*N for i in range(M)] # начално състояние на НМ (E е невронна матрица/карта)
obst = [[[3], [0]],
                        [[3], [1]],
        [[3], [2]],
        [[3], [3]],
        [[3], [4]],
                                                                [[3],    [5]]]    #
списък с координати на препрядствията в E [col, row]
track =    [[[3], [0]],
                            [[3], [1]],
           [[3], [2]],
           [[3], [3]],
           [[3], [4]],
                                                                [[3],   [5]]]  #
списък с координати на следите в E [col, row]

domain = (dw, sw, dw, sw, scw, sw, dw, sw, dw) # списък на клетките в малкото
пространство F
obstacle = -1    #значение на препятствието в началото, после става E = 0;

col = start[1]     # колона с текущите позиции
row = start[0]
gradient = [0] * 8      # Буфер за градиентите
sum_gradient = 0
recalculate = False # променлива за необходимостта от преизчисляване на траекторията
cur_direction = 0 # начално направление (на 12ч.)
username = urllib.parse.quote_plus('antouan')
password = urllib.parse.quote_plus('boriss150')
   # Provide the mongodb atlas url to connect python to mongodb using pymongo
```

```
#client                                                                    =
MongoClient('mongodb+srv://%s:%s@phd.sb35c.mongodb.net/PhD?retryWrites=true&w=m
ajority' % (username, password))

#db = client['PhD']

def set_obstacles(self):     #функция + I (при препядствие E = 0, а при следа трябва да
натежи с коефц.)
   if self == - 1:
      ar = np.array (obst, dtype=int)
      for i in range(len(ar)):
         row = ar[i, 0]    # малко сметки да превърнем координатите в цели числа за E
         col = ar[i, 1]
         print(int(col), int (row))   #col, row, 0.0 е в май горе, най-ляво
         E[int(col)][int(row)]= -1  # Use ar as a source of indices, to assign 1

   #print(np.matrix(obst))
  # arr_obst = [[0]*len (obst) for i in xrange(obst [len (obst)])
   # obsticles = np.reshape(arr_obst, (N, M))  # списък с координатите на предпрядствията;

def get_normw(curY, curX):  # функция за изчисляване на теглата
   wesumm = 0
   wsumm = 0
   wcount = 0      #Разглежда се всеки съседен неврон
   #print('функция: изчисляване на теглата')
   for y in range((curY - 1), (curY + 1 + 1)):
      for x in range((curX - 1), (curX + 1 + 1)):     # ако координатите на тествания неврон
не излизат извън работната зона и не падат върху препядтвие
         if x >= 0 and y >= 0 and x < N and y < M:
            if E[y][x] != -1:
               #print (E[y][x], wcount, wsumm, wesumm, domain[wcount], y, x)
               wesumm += E[y][x] * domain[wcount]
               wsumm += domain[wcount]
         wcount += 1
   return wesumm / wsumm


def act_function(x):   # калкулация на функцията за активиране на неврон в зависимост
от линейна или нелинейна
   if actf_type == 'linear':
      #print('x:', x)
      if x<=0:
         x=0
      elif x > 1:
         x=1
   elif actf_type == 'tanh':
      x = (math.exp(2*x) - 1) / (math.exp(2*x) + 1)
      if x > 1:
         x=1
   #print('x:', x)
```

```
    return x

# Построяване на невронната карта
def net_activation(actf_type):
    cycles = 0 # брояч на цикъла на активиране
    cycles_max = (N * M) ** 2 # максимално допустимо количво цикли на активация
    if len(obst) > 0:
        set_obstacles (-1) # установяане "-1" за "изключване" на невроно-препятствие
    E[target[0]][target[1]] = 1 # установяване на активирания неврон
    m=1 #стъпка
    n=1
    while E[start[0]][start[1]] < 1e-15:
        print('цикли:', cycles)
        for y in range((target[0] - m), (target[0] + m + 1)):
            for x in range((target[1] - n), (target[1] + n + 1)):

                if x >= 0 and y >= 0 and x < N and y < M:
                    if E[y][x] == 1:
                        E[y][x] = 1
                    elif E[y][x] != -1:      # при -1 има препрядствие! => да се отрази
                        E[y][x] = act_function(get_normw(y, x))
                        #print(E[y][x])
                    #if len(obst) > 0:
                        #set_obstacles(0)
                if m < M:
                    m += 1
                if n < N:
                    n += 1

        cycles += 1
        print(np.round(E, 3))        #даните от матрицата се закръглят до 3ти знак след десетичната
        if cycles > cycles_max:
            break
    #print (E[start[0]][start[1]])
    #if len(obst) > 0:
        #set_obstacles(0)                # формиране на крайния изходен сигнал "net_output" (нулиране на предпрядствията)
    print(" number of activation cycles: " + str(cycles))

#start1 = act_function(x)
test = net_activation (actf_type)
#print (get_normw(y, x))
#print(np.round (E, 3))
```

ПРИЛОЖЕНИЕ Б:

Алгоритъм A* - Pathfinder.cpp /
https://github.com/yatima1460/AStar/blob/master/Source/Pathfinder.cpp

---

```
#ifdef GOOGLE_TEST
#include "Pathfinder.hpp"
#endif

#include <cmath>
#include <queue>
#include <unordered_map>

/*
============================================================

    ___       _
   /   \     /\| |/\
  / ^ \ \   \ `' /
 / /_\ \ \ |_    _|
/ _____ \ \ / , . \
/__/    \__\ \/|_|\/

Pathfinding implementation using A*

Supports checking a partial path first to avoid
the algorithm wasting time when the world is walkable but
the end is narrowly enclosed

Note: in general I tried to avoid as many IFs as possible
to not have failing CPU branch prediction in the hot path.

============================================================
*/

// ============ CONFIGS ============

/*
How much the dot product between the (start,end) and (current,end)
matters in the heuristics, multiplied by this value,
WARNING: don't make it too big otherwise you risk to overshoot
WARNING: using the dot product can increase performance but paths
could look "strange" in a game, not really human-like in some cases
better to use for things where no graphics is involved
*/
#define DOT_PRODUCT_WEIGHT 0.0005f

/*
How much of the world area to scan from the end before starting
To avoid situations where just the end is enclosed but the world is walkable
*/
```

```
#define CHECK_END_IF_NARROWLY_ENCLOSED_BEFORE_STARTING true
#define WORLD_MAP_END_ENCLOSED_WEIGHT 0.0005f

// ================================

/*
Can pollute the namespace, but at least for this test's code you get better readability
But be careful to use this in the real world
*/
using namespace std;

// Sometimes this saved like 20ms
// Hey it's a whole game frame time, even more!
#ifdef __GNUC__
#define likely(x) __builtin_expect(!!(x), 1)
#define unlikely(x) __builtin_expect(!!(x), 0)
#else
#define likely(x) (x)
#define unlikely(x) (x)
#endif

// Bundle most used data together to have fewer cache misses
struct NodeData
{
   int nDistance = numeric_limits<int>::max();
   float nScore = 0.0f;
   int nParentIndex = -1;
};

// Same as FindPath, but with a cutoff of nodes count discovered
int PartialFindPath(const int nStartIndex, const int nStartX, const int nStartY, const int nTargetIndex, const int nTargetX, const int nTargetY,
                    const unsigned char *pMap, const int nMapWidth, const int nMapHeight,
                    int *pOutBuffer, const int nOutBufferSize, const int nCutoff)
{

   unordered_map<int, NodeData> nodeMetadata;

   /*
   I actually wanted to use a Set with a custom comparator instead of a priority queue +
   an unordered_map to check if an element is inside in O(1)...
   but for Sets C++ also uses the comparator for equivalence using reflexivity
   instead of the element operator==
   two set keys are considered equal if !comp(a,b) && !comp(b,a))
   I think the problem is that C++ has strict weak ordering containers
   I read that C++2020 will fix this, so this can be improved even further
   */
   const auto openNodesComparator = [&nodeMetadata] (const int &nIndexLHS, const int &nIndexRHS)
   {
```

```
        const auto &nodeDataLHS = nodeMetadata[nIndexLHS];
        const auto &nodeDataRHS = nodeMetadata[nIndexRHS];
        return nodeDataLHS.nScore == nodeDataRHS.nScore ? nodeDataLHS.nDistance > nodeDataRHS.nDistance : nodeDataLHS.nScore > nodeDataRHS.nScore;
    };

    // Maybe a Fibonacci heap is better for this by having O(1) insert and O(logN) amortised extract?
    vector<int> openNodesContainer; // so clang-tidy doesn't complain
    priority_queue<int, vector<int>, decltype(openNodesComparator)> openNodesPriorityQueue(openNodesComparator, openNodesContainer);

    // Setup root node
    openNodesPriorityQueue.push(nStartIndex);
    nodeMetadata[nStartIndex].nDistance = 0;
    int nCurrentIndex = nStartIndex;

    // Useful for later for heuristics
    const auto nDeltaXStartTarget = nStartX - nTargetX;
    const auto nDeltaYStartTarget = nStartY - nTargetY;

    static const int NEIGHBOURS_X_OFFSET[4] = {+0, +1, +0, -1};
    static const int NEIGHBOURS_Y_OFFSET[4] = {-1, +0, +1, +0};

    // Stop if open nodes is empty, if we reached the destination or if we reached the cutoff
    while (unlikely(!openNodesPriorityQueue.empty() && nCurrentIndex != nTargetIndex && (int)nodeMetadata.size() < nCutoff))
    {
        // O(1) to get the node with the lower f because open is sorted
        nCurrentIndex = openNodesPriorityQueue.top();
        openNodesPriorityQueue.pop();

        // Calculate new Dijkstra's cost, it's a 2D grid so always +1
        const auto gCurrent = nodeMetadata[nCurrentIndex].nDistance + 1;

        // Get X and Y of current node from the index
        const int xCurrent = nCurrentIndex % nMapWidth;
        const int yCurrent = nCurrentIndex / nMapWidth;

        // Dot product between (start,end) and (current,end), used later for heuristics
        const auto nDeltaXCurrentTarget = xCurrent - nTargetX;
        const auto nDeltaYCurrentTarget = yCurrent - nTargetY;
        const auto dot = abs(nDeltaXCurrentTarget * nDeltaYStartTarget - nDeltaXStartTarget * nDeltaYCurrentTarget);

        for (int i = 0; i < 4; ++i)
        {
            const int new_x = xCurrent + NEIGHBOURS_X_OFFSET[i];
            const int new_y = yCurrent + NEIGHBOURS_Y_OFFSET[i];
            const int neighbourIndex = new_x + new_y * nMapWidth;
```

```
            // if neighbour inside the map && walkable && better cost
            if (likely(new_x >= 0 && new_y >= 0 && new_x < nMapWidth && new_y < nMapHeight)
                && static_cast<bool>(pMap[neighbourIndex])
                && unlikely(gCurrent < nodeMetadata[neighbourIndex].nDistance))
            {
                // save where we came from to reconstruct path later
                nodeMetadata[neighbourIndex].nParentIndex = nCurrentIndex;

                // save new better cost
                nodeMetadata[neighbourIndex].nDistance = gCurrent;

                // f = dijkstra + Manhattan + dot product between (start,end) and (current,end)
                const auto Manhattan = fabs(static_cast<float>(new_x - nTargetX)) + fabs(static_cast<float>(new_y - nTargetY));
                nodeMetadata[neighbourIndex].nScore = gCurrent + Manhattan + dot * DOT_PRODUCT_WEIGHT;

                // better cost so add it to open
                openNodesPriorityQueue.push(neighbourIndex);
            }
        }
    }

    // If we broke the while because we found the target
    if (nCurrentIndex == nTargetIndex)
    {
        // Reconstruct path
        int i = 0;
        while (likely(nodeMetadata[nCurrentIndex].nParentIndex != -1))
        {
            nCurrentIndex = nodeMetadata[nCurrentIndex].nParentIndex;
            if (likely(i < nOutBufferSize))
                pOutBuffer[i] = nCurrentIndex;
            i++;
        }
        return i == 0 ? -1 : i;
    }

    // If no path found
    return static_cast<int>(nodeMetadata.size()) < nCutoff ? -1 : -2;
}

int FindPath(const int nStartX, const int nStartY,
             const int nTargetX, const int nTargetY,
             const unsigned char *pMap, const int nMapWidth, const int nMapHeight,
             int *pOutBuffer, const int nOutBufferSize)

{
```

```
    // Consider simple case if target is the same as position
    if (unlikely(nStartX == nTargetX && nStartY == nTargetY))
        return 0;

    // Useful for later, so we don't calculate them every time
    const auto nStartIndex = nStartX + nStartY * nMapWidth;
    const auto nTargetIndex = nTargetX + nTargetY * nMapWidth;

#if CHECK_END_IF_NARROWLY_ENCLOSED_BEFORE_STARTING
    // Checks a bit of the world map around start before the normal search
    const int nCutoff = static_cast<int>(nMapWidth * nMapHeight * WORLD_MAP_END_ENCLOSED_WEIGHT);
    const auto nPartialResult = PartialFindPath(nStartIndex, nStartX, nStartY, nTargetIndex, nTargetX, nTargetY, pMap, nMapWidth, nMapHeight, pOutBuffer, nOutBufferSize, nCutoff);
    if (unlikely(nPartialResult == -1))
        return -1;
#endif

    // Reverse nTargetIndex and nStartIndex so the path doesn't need to be reversed later on
    return PartialFindPath(nTargetIndex, nTargetX, nTargetY, nStartIndex, nStartX, nStartY, pMap, nMapWidth, nMapHeight, pOutBuffer, nOutBufferSize, nMapWidth * nMapHeight + 1);
}
```