Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Robert Tovornik

Izvajanje vzporednih planov za sodelovanje skupine robotov

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: akad. prof. dr. Ivan Bratko

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Naloga je implementirati usklajeno gibanje mobilnih robotov na pravokotni mreži. Naloga robotov je, da čim prej dosežejo svoje ciljne položaje v mreži, pri čemer se morajo izogibati drug drugemu. Za simbolično planiranje robotskih akcij uporabite standardne hevristične algoritme ali prilagodite že obstoječi program. Posebej se v svojem delu posvetite izvajanju vzporednih planov z enostavnimi roboti Thymio tako, da bodo akcije časovno usklajene, s čimer bo zagotovljena varnost in zanesljivost izvajanja. Pri tem uporabite že vgrajene senzorje teh robotov.

Zahvaljujem se mentorju, akad. prof. dr. Ivanu Bratku za strokovno pomoč pri izdelavi diplomskega dela. Zahvaljujem se asistentu, mag. Domnu Šoberlu za pomoč in nasvete pri tehnični izvedbi. Zahvaljujem se družini in prijateljem za podporo in vzpodbudo, ki so mi jo nudili tekom študija.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
IR	infrared radiation	infrardeča radiacija
WIFI	Wi-Fi	brezžična tehnologija za omrežno komunikacijo
USB	Universal serial bus	univerzalno serijsko vodilo

Povzetek

Naslov: Izvajanje vzporednih planov za sodelovanje skupine robotov

Avtor: Robert Tovornik

V okviru danega diplomskega dela smo se odločili implementirati usklajeno premikanje preprostih robotov Thymio na pravokotni mreži. V postopku priprave izvedbenih planov smo za osnovo vzeli abstraktne plane za reševanje variant igre osmih ploščic. Izdelali smo algoritem za pretvorbo abstraktnih planov v izvedbene plane za robote, ki fizično izvedejo premike ploščic. Abstraktnim planom smo dodali potrebne poteze rotacij in mirovanja z ozirom na ohranitev časovne usklajenosti izvedbenih ukazov posameznih robotov, s čimer zagotovimo varno premikanje robotov. Robote Thymio smo tekstovno sprogramirali v programskem jeziku Aseba kot končne avtomate. Avtomat deluje v dveh nivojih, globalno in lokalno in se odziva na dogodke, kot je zaznava bližine drugega robota. Globalni nivo skrbi za sinhronizacijo akcij abstraktnega plana. Lokalni nivo skrbi za izvedbo akcij in se odziva na spremebe okolja. Implementirani algoritem smo ovrednotili na nalogah gibanja do štirih robotov na mrežah velikosti do 3x3.

Ključne besede: planiranje, usklajeno, vzporedno, izvedba, robot, Thymio.

Abstract

Title: Execution of parallel plans for cooperation of a group of robots

Author: Robert Tovornik

Within the given diploma thesis, we decided to implement a coordinated movement of simple Thymio robots on a rectangular grid. We took the abstract plan of the eight puzzle game variant and used it as a basis in the process of forming execution plans. We created an algorithm for transforming abstract plans into execution plans for robots that physically perform the movements of tiles. We added the necessary moves of rotations and hibernation to the abstract plan, taking into consideration the preservation of harmonized execution commands of individual robots, therefore ensuring the safe movement of robot. The Thymio robots were programmed as finite-state automaton through text, using Aseba programming language. The automaton operates in two layers, both globally and locally and responds to events such as detecting proximity of another robot. The global layer ensures the synchronization of abstract plan actions. The local layer executes the actions and responds to environmental changes. The implemented algorithm was evaluated on the tasks of moving up to four robots on grids of size up to 3x3.

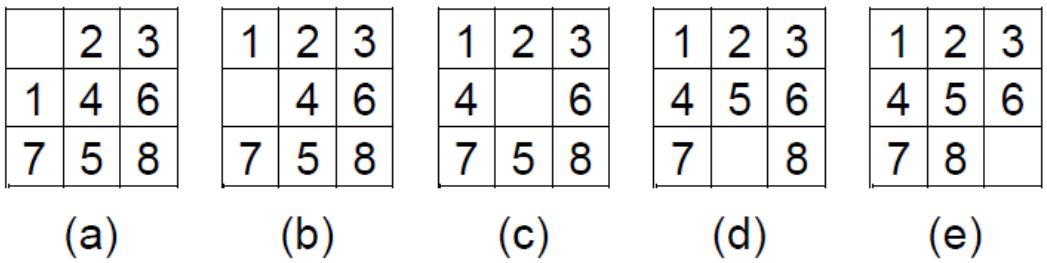
Keywords: planning, harmonized, parallel, execution, robot, Thymio.

Poglavje 1

Uvod

1.1 Uvodni primer

Igra osmih ploščic je preprosta igra, v kateri prerazporejamo osem ploščic na igralnem polju velikosti 3x3 tako, da ob vsaki potezi premaknemo eno ploščico na trenutno prosto polje. Cilj igre je prerazporediti ploščice v zahtevan vrstni red s čim manjšim številom potez [5]. Slika 1.1 prikazuje preprost primer, kako v štirih potezah podano začetno postavitev pripeljemo do ciljne postavitve.



Slika 1.1: Primer igranja igre osmih ploščic. Sličica (a) predstavlja začetno postavitev ploščic, sličica (e) pa želeno ciljno postavitev ploščic. Sličice (b), (c) in (d) predstavljajo vmesne zaporedne postavitve ploščic, vsaka po izvedeni eni potezi igre [5].

1.2 Kratek pregled tematike

V okviru diplomskega dela smo se ukvarjali z varianto igre osmih ploščic, kjer smo ploščice nadomestili z enostavnimi roboti (Thymio). Število robotov je v teh variantah igre manjše od osem, kar omogoča vzporedno premikanje robotov in zato so plani za reševanje naloge paralelni. Take variante igre osmih kvadratov bomo v nadaljevanju imenovali roboti na pravokotni mreži. Pozornost smo namenili planiranju in izvedbi časovno usklajenih planov za posamezne robote.

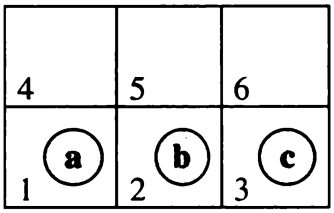
Pred fizično izvedbo smo morali pripraviti izvedbene plane robotov. Izvedbeni plani robotov so sestavljeni iz sekvence izvedbenih ukazov, naslovljenih na vsakega posameznega robota posebej. Izvedbeni ukazi so: premik naprej, rotacija levo, rotacija desno in miruj (eno potezo). Izvedbene plane smo pripravili na podlagi abstraktnih planov, ki jih je v okviru svojega diplomskega dela že pripravil Jernej Janež [4]. Janežev optimalni abstraktni plan za vsakega posameznega robota poda seznam koordinat, ki tekom izvajanja ponazarjajo zaporedne položaje robotov na mreži. Vendar Janeževi abstraktni plani ob planiranju izvajanja ne upoštevajo omejitev realnega sveta, kot na primer omejitve okolja, fizične omejitve robotov, omejitev komunikacije,.. Podane abstraktne plane kot vhodne podatke posredujemo našemu algoritmu, ki plane pregleda, uskladi, vstavi potrebne izvedbene korake "rotacij"in "mirovanja"ter kot izhodni podatek vrne končne izvedbene plane robotov. Primer abstraktnega in izvedbenega plana je prikazan v poglavju 1.3 Primer izvajanja.

Za fizično demonstracijo izvedbe planov smo uporabili preproste robote Thymio. Robote Thymio smo tekstovno sprogramirali v programskem jeziku Aseba, kot končne avtomate. Ob izvajanju planov delujejo roboti časovno usklajeno in se odzivajo na senzorične spremembe, kar zagotavlja natančnejšo izvedbo. Podrobnosti delovanja so opisane v poglavju 4 Planiranje in praktična izvedba.

1.3 Primer izvajanja

Za primer izvajanja smo izbrali zelo nazoren primer, ki ga podrobneje v svoji knjigi [3] opiše že profesor Bratko.

Primer: Podano je igralno polje v velikosti 2x3. V drugi vrstici polja, vzporedno stojijo trije roboti a, b in c, kot to prikazuje Slika 1.2. Cilj naloge je, da robota a in c medsebojno zamenjata svoja položaja.



Slika 1.2: Slika prikazuje začetni položaj robotov na mreži velikosti 2x3. Oznake a, b in c so imena treh robotov, števke 1,2,..,6 označujejo posamezna igralna polja v mreži [3].

V knjigi [3] najdemo abstrakten plan, ki opisuje potrebno zaporedje premikov za dosego rešitve problema. Na kratko ga opišemo z akcijami v obliki "oznaka robota,trenutno polje robota,naslednje polje robota", npr. akcijo robota c, ki se v eni potezi premakne iz polja številka 6 na polje številka 5 zapišemo kot: c,6,5. Zaporedje korakov v času ponazorimo s pomišljajem (–), sočasne korake pa z "in". Primer celotnega abstraktnega plana:

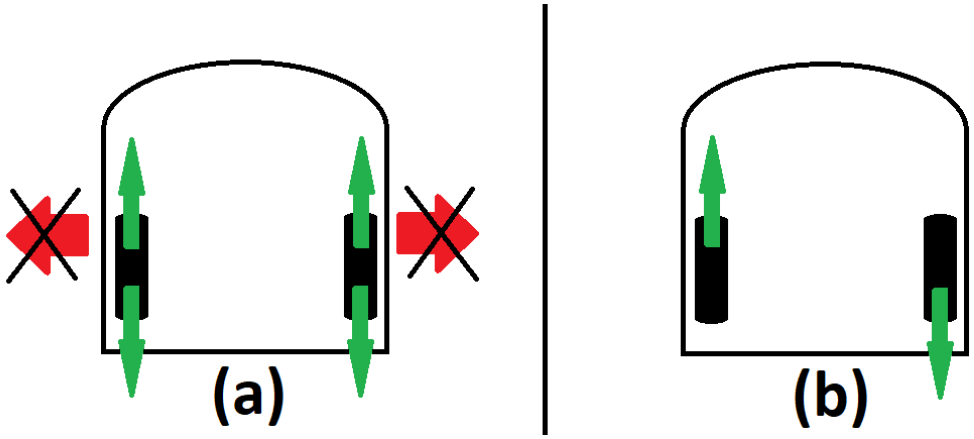
```
c,3,6 - c,6,5 - c,5,4 - b,2,5 - a,1,2 - a,2,3 in c,4,1
```
(1.1)

Z namenom, da se tudi robot b vrne na prvotni položaj, abstraktnemu planu na konec dodamo še akcijo b,5,2:

```
c,3,6 - c,6,5 - c,5,4 - b,2,5 - a,1,2 - a,2,3 in c,4,1 - b,5,2
```
(1.2)

Takšen plan predpostavlja, da imajo roboti zmožnost takojšnjega premikanja v vse smeri (navzgor, navzdol, levo, desno). V praksi pa so takšni roboti redki. Večina robotov, podobno kot avtomobili, premika vstran (levo ali desno) zaradi

bočne postavitve koles ni zmožna opraviti, za takšno potezo je naprej potreben korak rotacije v smer premika.



Slika 1.3: Shematika rotacije koles pri robotih Thymio. Sličica (a) prikazuje zmožnost rotacije koles okoli osi in nezmožnost premikanja vstran. Sličica (b) prikazuje primer rotacije koles ob rotaciji robota v desno smer.

Dodatno so lahko roboti omejeni tudi pri gibanju vzvratno, posebej kadar imajo detekcijske senzorje nameščene zgolj na sprednji strani robota. Uporabljeni roboti Thymio imajo takšno omejitev. V praksi to pomeni, da sta namesto koraka navzdol (vzvratno) potrebna dva zaporedna koraka rotacije, levo–levo ali desno–desno. Posledično se izvedbeni plani hitro začnejo daljšati in razlikovati od abstraktnih planov. Zato je v primeru izvedbenih planov sočasno oziroma usklajeno delovanje še toliko pomembnejše.

Vzporedno usklajen izvedbeni plan pridobimo s pomočjo našega algoritma. Podan abstrakten plan algoritem sprejme kot svoj podatkovni vhod. V primeru abstraktnega plana (1.2) moramo abstraktni plan še preoblikovati v primerno koordinatno obliko, kot to prikažemo v poglavju 2.2.1 Primer prilagojenega abstraktnega plana. Algoritem podani plan nato preoblikuje, doda potrebne manjkajoče korake in optimizira vzporedno izvajanje. Algoritem predpostavlja, da so v začetnem stanju in končnem stanju vsi roboti obrnjeni navzgor. Sledi končna oblika plana, ki je nekoliko drugačna. Namesto skupnega, časovno zaporednega plana premi-

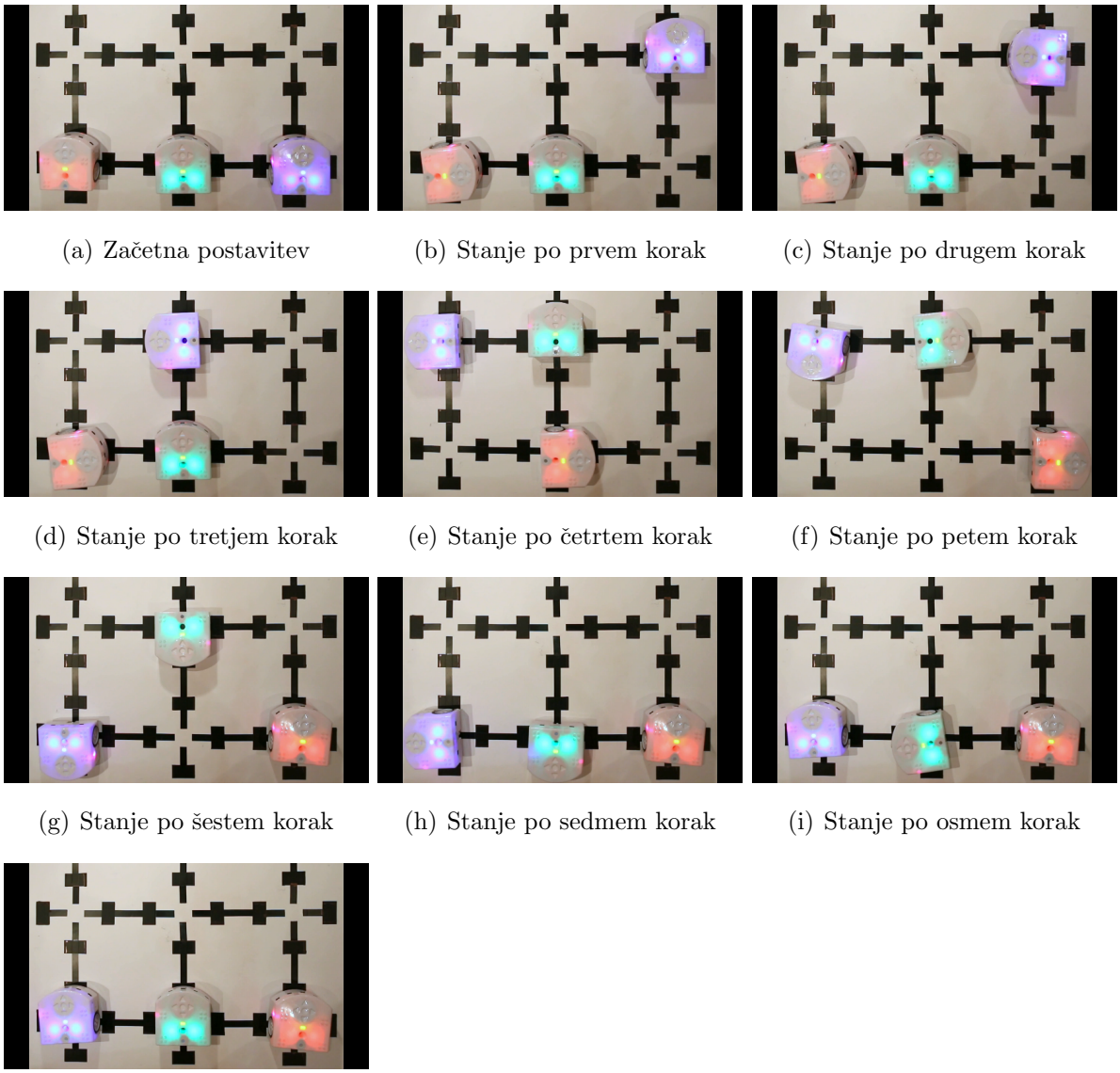
kov dobimo za vsakega posameznega robota plan ukazov, ki je časovno usklajen z vsemi preostalimi roboti. Časovno usklajenost zagotovimo tako, da vsem izvedbenim ukazom namenimo enak časovni interval za izvedbo. S poskušanjem smo ugotovili, da je za naš primer najboljša izbira omejitev časovnega intervala na 4.5 sekunde.

Izvedbeni plan v obliki akcij posameznih robotov za nalogo na sliki 1.2

- Robot a: Rotacija desno, Miruj, Miruj, Premik, Premik, Rotacija levo
- Robot b: Miruj, Miruj, Miruj, Premik, Rotacija desno, Rotacija desno, Premik, Rotacija desno, Rotacija desno
- Robot c: Premik, Rotacija levo, Premik, Premik, Rotacija levo, Premik, Rotacija desno, Rotacija desno

Dobljene izvedbene plane podamo vsem robotov ter jih sočasno poženemo. Dejansko izvajanje ukazov "premik"in "rotacija"je izvedeno s končnim avtomatom, ki upošteva informacije iz robotovih senzorjev. Ti detektirajo položaj robota iz barve podlage in bližino drugih robotov, kar robotom omogoča zanesljivejšo izvedbo.

Slika 1.4 prikazuje realizacijo izvedbenega plana z uporabo robotov Thymio, ki plan paralelno izvedejo skozi časovno zaporedje korakov.



(a) Začetna postavitev (b) Stanje po prvem korak (c) Stanje po drugem korak

(d) Stanje po tretjem korak (e) Stanje po četrtem korak (f) Stanje po petem korak

(g) Stanje po šestem korak (h) Stanje po sedmem korak (i) Stanje po osmem korak

(j) Po devetem koraku - končna postavitev

Slika 1.4: Slikovno zaporedje izvedbe tretjega scenarija z uporabo robotov Thymio.

Prvi vtis po preoblikovanju planov nam pusti občutek, da smo izvajanje pravzaprav zgolj podaljšali, namreč iz predhodnih šestih oziroma sedmih časovnih enot, smo izvajanje podaljšali na devet časovnih enot. Vendar moramo primerjati z izvedbo, kjer vzporedni premiki niso dovoljeni, prav tako niso dovoljeni premiki na trenutno zasedena polja. V takšnem primeru bi namesto porabljenih 9 časovnih enot izvedba trajala 23 zaporednih časovnih enot. Več o časovni optimalnosti v poglavju 5.1 Evalvacija optimalnosti.

Poglavje 2

Teoretična osnova

2.1 Prilagoditev pravil usklajenega izvajanja

Osnovna pravila igre osmih ploščic bi nas omejevala pri usklajenem izvajanju, zato smo osnovna pravila nekoliko prilagodili oziroma nekatera dodali.

1. pravilo: **Število ploščic na igralni površini**

   V osnovni igri je na igralni površini vedno prisotnih osem ploščic. Če pravilo prenesemo na poljubno velikost igralne površine, velja razmerje: na igralni površini velikosti $n * n$ je prisotnih $n * n - 1$ ploščic.

   Pravilo smo prilagodili tako, da velja: pri velikosti igralne površine $n * n$ je na igralni površini lahko prisotnih od 2 do $n * n - 1$ ploščic.

2. pravilo: **Začetna in končna postavitev ter orientacija**

   Osnovnim gradnikom ploščicam smo dodali lastnost orientacije. Dodano pravilo zahteva, da so ob začetku in ob zaključku igre vse ploščice obrnjene v isto smer. Privzeta smer orientacije določa usmeritev proti severu (navzgor).

3. pravilo: **Sprememba veljavnih premikov**

   Veljavne osnovne premike (navzgor, navzdol, levo, desno) nadomestimo z enim osnovnim premikom naprej v smeri trenutne orientacije ploščice. Kot

nadomestilo preostalim dodamo nov veljaven premik "rotacijo". Poteza rotacije dovoljuje obrat okoli svoje osi za kot 90 stopinj in je samostojna poteza, številno enakovredna potezi premika.

4. pravilo: **Sočasno premikanje ploščic**

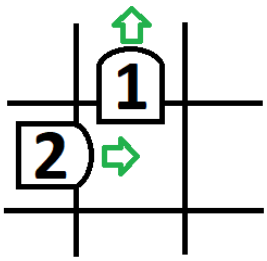
   V osnovni igri velja, da lahko ob vsaki potezi sočasno premaknemo zgolj eno ploščico. Novo prilagojeno pravilo dovoljuje, da lahko sočasno premaknemo vse prisotne ploščice, v kolikor to za posamezno ploščico dovoljujejo ostala pravila.

5. pravilo: **Prosta oziroma zasedena polja**

   Osnovna pravila ne dovoljujejo premikov ploščic na polja, ki so trenutno zasedena. Novo pravilo takšen premik ploščice dovoljuje, v kolikor se tudi ploščica, ki trenutno zaseda polje, v sklopu naslednje poteze premakne na drugo prosto polje, s čimer posledično sprosti trenutno zasedeno polje.

6. pravilo: **Velikost ploščic**

   Ploščice so po novem pravilu po velikosti manjše kot mrežna polja za toliko, da je omogočena sočasna sprostitev in zasedba polja. Poseben poudarek velja na premikih, kjer dve ploščici zasedata/sproščata isto polje v pravokotni si smeri premikov (slika 2.1). Alternativno lahko velikosti ploščic in mrežnih polj ostaneta enaki, vendar moramo med polji zagotoviti prehodno območje. Primer takšnega polja prikazuje slika 3.2.



Slika 2.1: Zasedanje/sproščanje istega polja v pravokotni si smeri premikov ploščic.

2.2 Abstraktno planiranje

Pod abstraktno planiranje štejemo planiranje, ki poišče optimalno rešitev v domeni osnovnih pravil igre osmih ploščic. Pri iskanju takšnih rešitev se algoritem ne obremenjuje s problemi realnega sveta. V našem primeru to pomeni, da se pri planiranju ne obremenjuje z izvedbenimi specifikami izbranih robotov Thymio, temveč predpostavi idealne izvedbene okoliščine.

Pri realizaciji oziroma izvedbenem planiranju pa moramo vnaprej upoštevati še dodatne faktorje, kot na primer: čas potreben za izvedbo posamezne poteze, hitrost prehoda robotov med polji, trenutna in potrebna orientacija robota, časovna usklajenost premikov,... Zato abstraktne plane uporabimo zgolj kot podlago za določitev izvedbenih planov.

Kot smo omenili že v uvodnem delu, je abstraktno planiranje v okviru svojega diplomskega dela izvedel Jernej Janež [4]. Pri postopku planiranja se je večinoma držal osnovnih pravil igre osmih kvadratov, poleg pa vključil še prilagojeni pravili o številu ploščic (1. pravilo) in sočasnem premikanju ploščic (4. pravilo). Pri planiranju je uporabil znani hevristični algoritem za iskanje najkrajših poti A*, pri čemer je za hevristično oceno izbral štiri različne pristope. Za potrebe optimalnosti smo mi uporabili različico MAX - maksimalno, ki pri ocenjevanju dolžine plana uporablja najdaljšo najdeno razdaljo robota do svojega cilja in s tem zagotovi optimalno rešitev.

Janežev algoritem deluje tako, da ustvari začetno postavitev robotov, preveri veljavnost oziroma rešljivost igre glede na ustvarjeno postavitev ter poišče najkrajše poti k rešitvi igre. Izhod so poti posameznih robotov v obliki zaporednih koordinat premikov robotov po igralnih plošči. Rezultat Janeževega algoritma predstavlja vhodne podatke našega algoritem za izvedbeno planiranje.

2.2.1 Primer prilagoditve abstraktnega plana

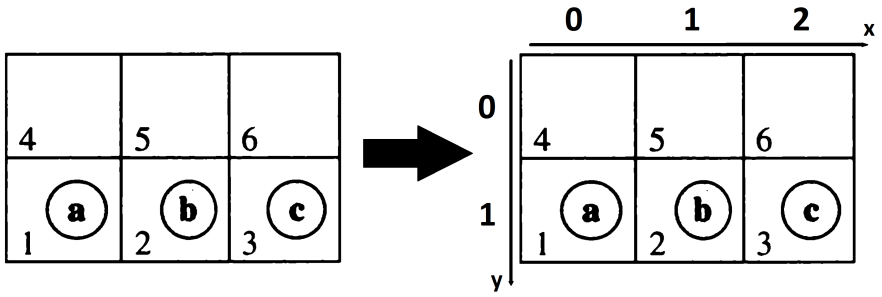
Za ponazoritev prilagoditve abstraktnega plana, ki bo primeren vhodnim podatkom našega algoritma privzemimo primer plana (1.2) iz uvodnega poglavja 1.3:

```
c,3,6 - c,6,5 - c,5,4 - b,2,5 - a,1,2 - a,2,3 in c,4,1 - b,5,2
```

Ker naš algoritem kot vhod sprejme zaporedne koordinate položajev robotov, moramo plan najprej prilagoditi oziroma zapis preoblikovati v enak format, kot ga na izhod poda v svojem algoritmu Jernej Janež. To storimo v dveh preprostih korakih.

1. korak: **Določitev koordinatnega sistema**

   Na podani igralni površini določimo natančne položaje z določitvijo koordinatnega sistem. Koordinatni sistem je standarden z izhodišem v zgornjem levem kotu. Horizontalo določa koordinata x in narašča z leve proti desni, vertikalo pa določa koordinata y, ki narašča od zgoraj navzdol.



Slika 2.2: Določitev koordinatnega sistema. Koordinatni sistem opredeljujeta osi x in y, z izhodiščem [0,0] v zgornjem levem kotu.

2. korak: **Preoblikovanje zapisa v abstraktni plan**

   Glede na koordinatni sistem odčitamo položaj robota v obliki **[y, x]**. Začetni položaji naših robotov so: robot a – [1, 0], robot b – [1, 1] in robot c – [1, 2].

   Podane začetne položaje uporabimo za orientacijo pri preoblikovanju zapisa plana. Glede na podan vrstni red premikov sledimo položajem posameznih robotov ter jih zabeležimo. Za podani primer dobimo naslednji plan:

   - robot a: [[1, 0], [1, 0], [1, 0], [1, 1], [1, 2]]
   - robot b: [[1, 1], [1, 1], [1, 1], [0, 1], [1, 1]]

- robot c: [[1, 2], [0, 2], [0, 1], [0, 0], [1, 0]]

Prilagojen zapis abstraktnega plana predstavlja vhodni podatek našega algoritma.

2.3 Optimizacijski kriterij

Glede na potrebe izvedbe v realnem svetu obstaja mnogo med seboj zelo različnih potencialnih kriterijev optimizacije, npr. časovna zahtevnost izračuna plana, računska zahtevnost, dejanski čas izvedbe,.. Ker nas zanima predvsem izvedba planov v realnem svetu, smo se odločili da kot kriterij optimalnosti izberemo čas izvedbe.

Časovna enota optimizacijskega kriterija je ena poteza robota oziroma izvedba enega ukaza, kar predstavlja globalni časovni okvir 4.5 sekunde. Poteze so globalne, torej nastopijo pri vseh robotih hkrati. Znotraj ene globalne poteze lahko delujejo vsi roboti, če jim izvedbeni plan tako narekuje.

V okviru optimizacijskega kriterija želimo doseči minimalno število globalnih potez, potrebnih za dosego končne postavitve robotov. Podrobneje o implementaciji v poglavju 4 Planiranje in praktična izvedba.

Poglavje 3

Roboti, okolje ter omejitve

3.1 Roboti Thymio

Robot Thymio (Slika 3.1), je preprost, 11x11 centimetrov velik, spredaj polkrožno zaobljen, dvoosni robot. Sprednji del robota sloni na plastični izboklini, ki omogoča drsenje po podlagi. Zadnji del robota sestavljata dve pogonski kolesi, vsako z lastnim elektromotorjem ter zmožnostjo vrtenja naprej in nazaj. Robot Thymio je opremljen s številnimi preprostimi senzorji (IR senzorji, mikrofon, akcelerator, idr.).

Slika 3.1: Robot Thymio s senzorji [1].

Za našo izvedbo najpomembnejša oprema je komplet devetih infrardečih senzorjev: pet distančnih IR senzorjev nameščenih na sprednji strani robota, dva distančna IR senzorja nameščena na zadnji strani robota ter dva reflektno–ambientna IR senzorja nameščena spredaj, na spodnji ploskvi robota usmerjena pravokotno na podlago. Distančni senzorji so namenjeni detekciji ovir. Bližnje objekte zaznajo na razdalji do 10 centimetrov, odvisno predvsem od materiala objektov (zaradi različne odbojnosti svetlobe). Dva reflektno–ambientna IR senzorja omogočata zaznavanje ambientne barve oziroma odbojnosti podlage pod robotom. Primarno sta senzorja namenjena razlikovanju med svetlo in temno podlago. Naprednejše zaznave podlage, na primer barvne raznolikosti ne omogočata. Sprednji in zadnji distančni IR senzorji hkrati služijo tudi namenu komunikacije med roboti, vendar je le-ta močno omejena z bližino in natančno medsebojno poravnavo senzorjev robotov, ki komunicirata.

Komunikacija z računalnikom poteka preko USB vmesnika, z največ enim robotom naenkrat. Druge oblike komunikacije roboti ne poznajo.

Programiranje robotov je predvideno v privzetem jeziku Aseba kot tekstovno programiranje, omogočeno pa je tudi programiranje z uporabo vizualnih blokov.

3.2 Okolje in pravila okolja

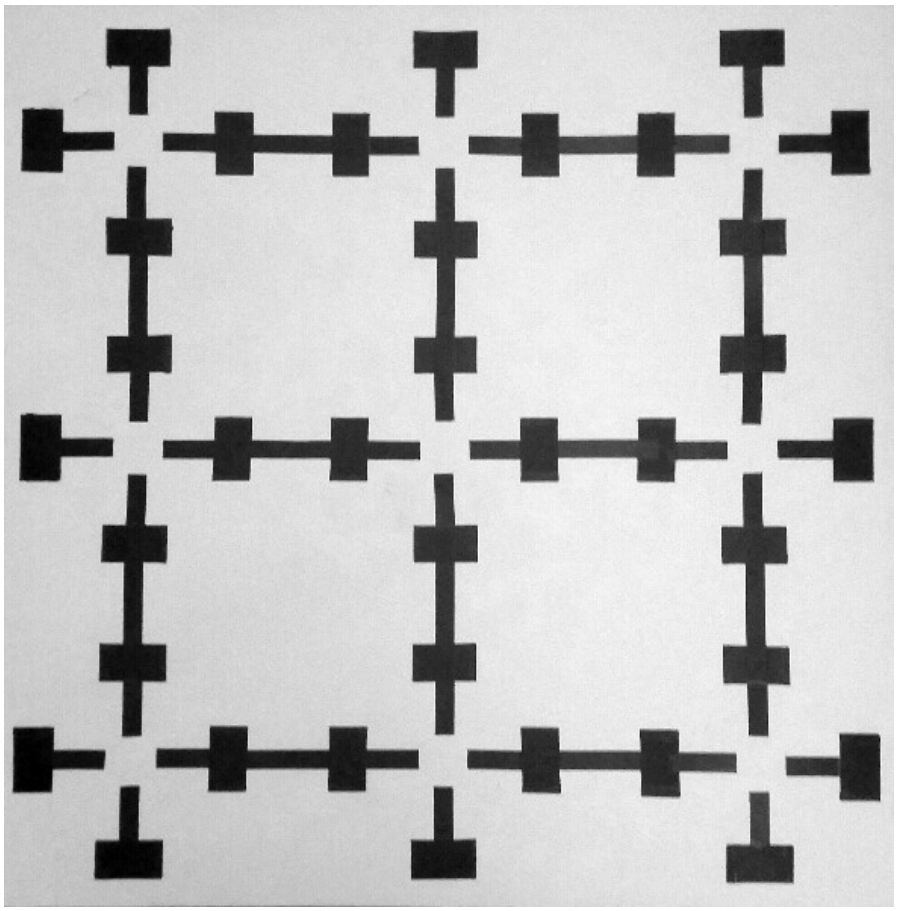
Osnovno okolje igre osmih ploščic predstavlja preprosta igralna površina velikosti 3x3, brez robov, zgolj z ločilnimi črtami.

Vendar pa je takšno okolje za preproste robote Thymio dinamično prezahtevno oziroma premalo informativno. Ob prilagojenih pravilih o orientaciji in določenem položaju se roboti ne bi uspeli orientirati brez dodatnega nadzornega sistema, npr. kamere. Kamera s pregledom nad celotno površino bi dejanski položaj robotov lahko spremljala v vsakem trenutku. Vendar bi za avtomatizacijo postopka potrebovali zelo učinkovit sistem sledenja gibanju, kar bi v primeru majhnega števila robotov bilo izvedljivo, z naraščanjem števila robotov pa bi se tudi izvedljivost močno otežila. Zato smo problem rešili tako, da smo izvedbeno okolje prilagodili omejitvam robotov.

Reševali smo tri glavne probleme:

- Zagotovitev učinkovitega premikanja robotov med polji
- Zagotovitev učinkovite izvedbe rotacije znotraj polja
- Zagotovitev učinkovite poravnave robotov med premiki znotraj polja

Vsem trem zahtevam smo uspeli ugoditi tako, da smo ustvarili okolje v katerem je gibanje robotov omejeno in vodeno.



Slika 3.2: Ustvarjena pravokotna mreža - izvedbeno okolje za robote Thymio

Da bi takšno gibanje čim bolje zagotovili, smo v kombinaciji z vodili uporabili

oba reflektno-ambientna IR senzorja na spodnji strani robota, ki sta nam omogočila dobro razlikovanje med svetlo in temno podlago in s tem sledenje robota vodilu.

Za postopek vodenja robotov, smo uporabili črni lepilni elektro trak, ki smo ga nalepili na kakovosten bel papir (šeleshamer). Bel papir oziroma bela podloga kvadratne oblike je velika 70x70 centimetrov. Na podlagi smo z elektro trakom oblikovali devet manjših kvadratov velikosti 12x12 centimetrov ki predstavljajo igralna polja. Polja omejujejo štirje pravokotniki velikosti 3x5 centimetrov, ki predstavljajo necelostne stranice kvadratov.

Glavno vodilo, kateremu robot sledi tako med premikanjem znotraj polj, kot tudi pri prehodu med polji, predstavlja 1.5 centimetra široki črni trak. Znotraj polj se vodilo, ki usmerja gibanje robota, prekine za dolžino štirih centimetrov.

Končno obliko izvedbenega okolja predstavlja slika 3.2.

3.2.1 Kratka obrazložitev okoljskih meril

Velikost kvadratov – Roboti so veliki 11x11 centimetrov, kvadrati so veliki 12x12 centimetrov. Dodaten centimeter je podan zaradi napak pri rotaciji, časovnega zamika pri zaustavitvi robota ter časovnega zamika pri senzoričnih zaznavah.

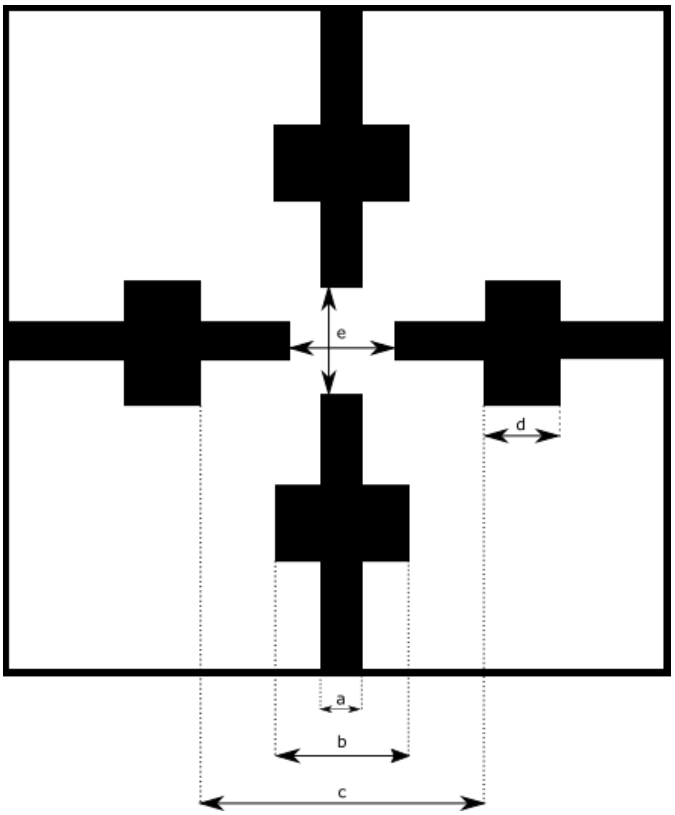
Stranice kvadratov – Pri stranicah kvadratov dolžine 12 centimetrov smo obdržali zgolj sredinskih 5 centimetrov, kar robotu omogoča, da med rotacijo sledi vzorcu prehoda: temna podlaga – svetla podlaga – temna podlaga, ne da bi se vmes naključno pojavila temna podlaga. Več o rotacijah v poglavju 4.2 Implementacija in delovanje robotov Thymio.

Širina vodila – Vodilo, črni električni trak meri v širino 1.5 centimetra, kar ustreza razmaku sprednjih reflektno-ambientnih IR senzorjev. To omogoča, da robot zanesljivo sledi črnemu vodilu.

Prekinitev vodila – Vodila znotraj kvadratov so potrebna, da robot ohrani orientacijo tudi med premikanjem znotraj kvadrata. Vmesna prekinitev v dolžini 4

centimetrov je nujna, saj robot znotraj polja sledi beli podlagi ne glede na smer iz katere se robot pripelje v polje. Brez potrebne prekinitve bi se robot zaustavil na sredini kvadrata.

Material – Črni električni trak smo za vodilo izbrali iz dveh razlogov: črna barva pri zaznavah IR senzorja zagotovi dober kontrast ambientne svetlobe v primerjavi z belo podlago, zgornja plast električnega traku pa, v primerjavi z zgolj črno pobarvanim papirjem, zagotovi zelo dober kontrast pri odstotku svetlobne odsevnosti materiala.



Slika 3.3: Igralno polje mreže in pripadajoče mere: a – širina vodila: 1.5 cm, b – dolžina kvadratka: 5 cm, c – velikost polja: 12 cm, d – širina kvadratka: 3 cm in e – prekinitev vodila: 4 cm.

3.3 Fizične omejitve

Roboti Thymio so preprosti roboti, v prvi vrsti namenjeni izobraževalnim programom, zato so tudi opremljeni s senzorji povprečne kvalitete. Posledično so senzorji dosti občutljivejši na zunanje vplive, ki se pojavijo tekom izvajanja nalog in zato niso najbolj primerni za dela, ki zahtevajo visoko natančnost in zanesljivost.

Kvaliteto senzorjev smo upoštevali pri načrtovanju izvedbe. Pri premikih robota smo zagotovili hitrost, ki spodnjima IR senzorjema omogoči dovolj časa za kakovostno odčitavanje podlage pod seboj. Vseeno včasih pride do napake, večinoma zaradi odboja luči pod napačnim kotom ali pa zaradi prisotnosti kakšne smetke, prahu. Posledica so napačne vrednosti odsevnosti in/ali barve.

Podoben problem se pojavi tudi pri uporabi horizontalnih distančnih senzorjev, saj je odbojnost IR signala močno odvisna od oblike in materiala zaznanega objekta. Ob različnih materialih objektov pride tudi do razlike v zaznavi oddaljenosti objekta od robota [2].

Na problem smo naleteli tudi pri uporabi vgrajenega mikrofona. Vzrok se nahaja v poziciji mikrofona znotraj robota in slabi zvočni izolaciji okolice. Tekom premikanja robotov motorji v kombinaciji s tresljaji oddajajo dovolj močen zvok, da mikrofon, ki je od motorjev oddaljen zgolj nekaj centimetrov, ne razloči več zunanjih zvokov.

Glavna pomanjkljivost robotov Thymio, ki je močno ovirala tako planiranje kot tudi samo izvedbo, pa je komunikacija. Vsa komunikacija med posameznimi roboti poteka preko sedmih horizontalnih IR senzorjev. Namenjeni so kratki in preprosti izmenjavi informacij o senzoričnih zaznavah. Dodatno je ta vrsta komunikacije otežena, saj IR sprejemniki in oddajniki zaradi načina delovanja zahtevajo, da sta ob izmenjavi informacij robota s svojimi senzorji medsebojno povsem poravnana (poravnava senzorja robota, ki informacijo oddaja in senzorja robota, ki informacijo prejema).

Za precizno premikanje robotov po igralni površini takšen način komunikacije ni uporaben, saj zahteva visoko stopnjo prostega gibanja znotraj polj in tesno bližino robotov. Dodatno je IR komunikacija omejena, saj ne omogoča sočasne komunikacije z več deležniki, temveč poteka istočasno zgolj med dvema robotoma.

Za skupino robotov postane takšna oblika komunikacije časovno prezahtevna.

Delno otežena je tudi komunikacija robota z računalnikom. Problem leži v načinu identificiranja posameznih robotov. Vsi komunikacijski vmesniki so tovarniško enako poimenovani s privzetim imenom "Thymio-II". Ker ni unikatnega identifikatorja, je v poljubnem trenutku z računalnikom nemogoče povezati več kot enega robota Thymio.

Poglavje 4

Planiranje in praktična izvedba

4.1 Algoritem: izvedbeno planiranje

Pod pojem izvedbenega planiranja smo uvrstili planiranje, ki v svoji osnovi temelji na planih abstraktnega planiranja, vendar ne predpostavlja idealnih okoliščin teoretičnega sveta, temveč upošteva izvedbene pogoje in omejitve navedene v poglavju 3 Roboti, okolje ter omejitve.

Ustvarjeni algoritem, katerega vhodni podatki so abstraktni plani in katerega izhodni podatki so izvedbeni plani v obliki izvedbenih ukazov, je dokaj preprost. Sestavljen je iz štirih glavnih korakov:

- Pregled sekvence abstraktnih planov, določitev sekvence orientacij vseh robotov
- Pregled zaporednih orientacij robotov, dopolnitev s potezami rotacij in mirovanja
- Pregled vseh izvedbenih planov posameznih robotov, časovna in prostorska uskladitev ter časovna optimizacija planov
- Dokončna pretvorba pozicij robotov in vmesnih ukazov v izvedbene plane

Izvorna koda algoritma se nahaja na github repozitoriju [7]: `https://github.com/roberttovornik/multi-robot-parallel-plan-execution.git`

Z uporabo postopka opisanega poglavju 2.2.1 Primer prilagoditve abstraktnega plana, prilagodimo primer abstraktnega plana iz uvodnega poglavja 1.2 v obliko primerno vhodnim podatkov našega algoritma (zapis abstraktnega plana v obliki koordinat).

Robot	Koordinate robotov na podlagi abstraktnega plana
a	[[1, 0], [1, 0], [1, 0], [1, 1], [1, 2]]
b	[[1, 1], [1, 1], [1, 1], [0, 1], [1, 1]]
c	[[1, 2], [0, 2], [0, 1], [0, 0], [1, 0]]

Ker smo v primeru 1.2 uporabili mrežo velikosti 2x3, pri implementaciji pa mrežo velikosti 3x3 ploščice, prištejemo vrednostim koordinate "y"vrednost 1, da ponazorimo zamik zaradi dodane vrstice.

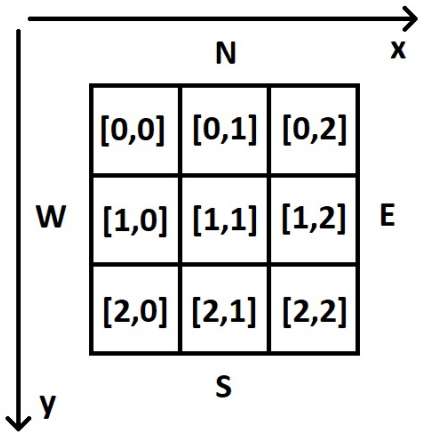
Dobimo sledeč plan, ki predstavlja vhodne podatke našega algoritma:

Robot	Koordinate robotov po prilagoditvi mreže
a	[[2, 0], [2, 0], [2, 0], [2, 1], [2, 2]]
b	[[2, 1], [2, 1], [2, 1], [1, 1], [2, 1]]
c	[[2, 2], [1, 2], [1, 1], [1, 0], [2, 0]]

Abstraktni plan kot vhodni podatek podamo v naš algoritem.

4.1.1 Pregled sekvence abstraktnih planov, določitev sekvence orientacij vseh robotov

V prvem koraku algoritma določimo orientacije robotov po celotni izvedbeni poti robotov. Predpostavimo začetno orientacijo vseh robotov proti severu. Oznake so v angleškem jeziku in predstavljajo: N - sever, S - jug, W - zahod ter E - vzhod.



Slika 4.1: Igralna površina velikost 3x3 s podanimi orientacijskimi oznakami: N - sever, S - jug, W - zahod ter E - vzhod. Polja so označena s koordinatami [y,x], kjer x narašča v smeri proti desni in y v smeri navzdol.

Orientacije robotov določamo glede na potrebne premike med polji in trenutno orientacijo. Primer: Če je robot na trenutni poziciji obrnjen v smeri proti severu, naslednja pozicija pa je pred njim (koordinata y je za vrednost ena manjša), potem se orientacija ohrani. V primeru, da je robot na trenutni poziciji obrnjen proti severu, naslednja pozicija pa se nahaja eno polje levo ali desno od trenutne pozicije robota (x koordinata je za vrednost ena manjša ali večja), potem se orientacija robota v naslednjem koraku spremeni proti zahodu ali vzhodu.

Vse preostale pogoje si je mogoče ogledati v izvorni kodi [7]. Za podan primer, dobimo naslednje podatke:

Robot	Koordinate robotov	Pripadajoče orientacije
a	[[2, 0], [2, 0], [2, 0], [2, 1], [2, 2]]	['N', 'N', 'N', 'E', 'E', 'N']
b	[[2, 1], [2, 1], [2, 1], [1, 1], [2, 1]]	['N', 'N', 'N', 'N', 'S', 'N']
c	[[2, 2], [1, 2], [1, 1], [1, 0], [2, 0]]	['N', 'N', 'W', 'W', 'S', 'N']

4.1.2 Pregled zaporednih orientacij robotov, dopolnitev s potezami rotacij in mirovanja

Drugi korak algoritma predstavlja pregled zaporednih orientacij posameznih robotov ter dopolnitev poti robotov z ukazi rotacij in mirovanja. Pogoji so preprosti. Primer: če je trenutna orientacija robota N – sever, naslednja orientacija pa W – zahod, vstavimo ukaz leva rotacija. Enako velja za preostale pogoje, ki so navedeni v izvorni kodi [7]. Izjema pri rotacijah se pojavi, ko gre za spremembo orientacije za 180 stopinj (popolni obrat). V takšnem primeru robotu dodamo dva zaporedna ukaza rotacije v desno.

Ukaze mirovanja vstavimo vsem preostalim robotom ob isti potezi, kadar za danega robota vstavljamo ukaz rotacije. Izjema so roboti, ki tudi sami v tej potezi izvajajo ukaz rotacije. Ukazi mirovanja so pomembni za ohranitev enakega vrstnega red izvajana, kot ga predvideva privzeti abstraktni plan. V primeru popolnega obrata oziroma dvojne rotacije vstavimo preostalim robotov dva zaporedna ukaza mirovanja.

Za naš primer algoritem vrne naslednji vmesni izpis.

Robot	Koordinate robotov z vstavljenimi potrebnimi ukazi rotacij in mirovanja
a	[[2, 0], [2, 0], 'miruj', [2, 0], 'rotacija', [2, 1], 'miruj', 'miruj', [2, 2]]
b	[[2, 1], [2, 1], 'miruj', [2, 1], 'miruj', [1, 1], 'rotacija', 'rotacija', [2, 1]]
c	[[2, 2], [1, 2], 'rotacija', [1, 1], 'miruj', [1, 0], 'miruj', 'miruj', [2, 0]]

4.1.3 Pregled vseh izvedbenih planov posameznih robotov, časovna in prostorska uskladitev ter časovna optimizacija planov

Tretji korak algoritma skrbi za časovno optimizacijo in usklajenost posameznih planov robotov z vsemi preostalimi roboti. Preverimo tudi, da roboti ne mirujejo po nepotrebnem. Algoritem iterira skozi posamezne plane robotov ter za vsako potezo mirovanja preveri, ali jo je mogoče izpustiti. Potezo mirovanja je mogoče

izpustiti, kadar zaradi izpuščenega ukaza mirovanja in s tem predhodne izvedbe naslednjega ukaza, robot, pri katerem želimo odstraniti potezo mirovanja, s svojo predčasno izvedbo naslednjega ukaza ne bi oviral delovanja drugih robotov oziroma kadar ne bi zasedel polja igralne površine, ki jo v danem trenutku že zaseda drugi robot. Algoritem iterira dokler prihaja do eliminacije katerekoli poteze mirovanja.

Po končanem tretjem koraku dobimo naslednji izpis:

Robot	Koordinate in ukazi po postopku optimizacije
a	[[2, 0], 'miruj', 'miruj', 'rotacija', [2, 1], [2, 2]]
b	[[2, 1], 'miruj', 'miruj', 'miruj', [1, 1], 'rotacija', 'rotacija', [2, 1]]
c	[[2, 2], [1, 2], 'rotacija', [1, 1], [1, 0], [2, 0]]

Če pogledamo rezultate optimizacije, lahko opazimo, da smo število globalnih potez skrajšali iz osmih na sedem. Prav tako smo optimizirali delovanje robota a in c, ki sta v določenih korakih neodvisna od robota b in zato svojo nalogo opravita hitreje. Robota a in c oba svojo nalogo opravita v petih potezah namesto v osmih potezah. Pri štetju korakov je potrebno paziti, prva koordinata označuje zgolj začetni položaj robota.

4.1.4 Dokončna pretvorba pozicij robotov in vmesnih ukazov v izvedbene plane

Četrti korak algoritma predstavlja dokončno preoblikovanje planov v izvedbene plane. Delimo ga na dva podkoraka. Prvi podkorak poskrbi, da robot potezo rotacije vedno izvede pred potezo mirovanja, v kolikor je rotacija potrebna. S tem zagotovimo pripravljenost robota na izvajanje naslednje poteze in občasno hkrati eliminiramo nepotrebno potezo mirovanja, namreč v kolikor s tem ne onemogočamo izvedbe drugih robotov je dovoljeno združiti potezo rotacije in mirovanja v zgolj potezo rotacije. Dodatno je v primeru predčasne rotacije robot pripravljen na spremljanje dogajanja na igralni plošči v smeri pred robotom. Drugi podkorak opravi prehode skozi posamezne plane robotov ter informacije o trenutni pozicij, poziciji v naslednjem koraku in/ali morebitnem ukazu združi v sekvenco izvedbenih

ukazov. Ukazi so podani v numerični obliki z oznakami, ki predstavlja ukazno stanje avtomata. Oznake predstavljajo sledeča stanja: 1 - premik naprej, 2 - leva rotacija, 3 - desna rotacija in 4 - poteza mirovanja. Ob končnem pregledu ukazov, robot preveri še končne orientacije robotov. Po konvenciji morajo biti vsi roboti usmerjeni navzgor, zato po potrebi doda zadnje ukaze rotacij, do poravnave orientacije.

Za podan primer nam robot vrne naslednji izhod.

Robot	Izvedbeni ukazi v obliki globalnih stanj avtomata
a	[3, 4, 4, 1, 1, 2]
b	[4, 4, 4, 1, 3, 3, 1, 3, 3]
c	[1, 2, 1, 1, 2, 1, 3, 3]

4.1.5 Izpis dokončne poti robotov v obliki koordinatnih poziciji na mreži

V zadnjem koraku lahko združimo informacijo pozicij abstraktnih planov ter izvedbenih ukazov z namenom pridobitve sekvence položajev za posamezne robote.

Za naš primer so pozicije robotov sledeče.

Robot	Končne pozicije robotov tekom izvedbe
a	[[2, 0], [2, 0], [2, 0], [2, 0], [2, 1], [2, 2], [2, 2]]
b	[[2, 1], [2, 1], [2, 1], [2, 1], [1, 1], [1, 1], [1, 1], [2, 1], [2, 1], [2, 1]]
c	[[2, 2], [1, 2], [1, 2], [1, 1], [1, 0], [1, 0], [2, 0], [2, 0], [2, 0]]

4.2 Implementacija in delovanje robotov Thymio

Algoritem, ki ga opišemo v prejšnjem poglavju, nam kot rezultat vrne izvedbene plane oziroma sekvence izvedbenih ukazov posamezni robotov v obliki oznak končnih stanj. Takšno obliko smo želeli zaradi načina implementacije delovanja samih robotov.

Kot omenjeno v uvodnem poglavju, smo robote Thymio programirali kot končne avtomate. Razlog za takšno implementacijo je omejenost programskega jezika Aseba, ki se uporablja za programiranje danih robotov. Programski jezik je implementiran tako, da omogoča programiranje zgolj v obliki odzivnosti robota na dogodke. Večino časa delovanja se odziva na dva tipa dogodkov: časovni dogodek, ki ga proži notranja ura robotov ter zaznavni dogodek spremembe vhodne vrednosti senzorjev. Ker je tudi komunikacija robotov zelo omejena, kot je razloženo v poglavju 3, smo se odločili implementirati zaprt sistem (končni avtomat) z vnaprej podanim načrtom delovanja, ki deluje v dveh nivojih.

Prvi, globalni nivo skrbi za časovno usklajeno prehajanje zaporednih ukaznih stanj vseh robotov. Ukazna stanja so določena kot: stanje premika, stanje leve rotacije, stanje desne rotacije in stanje mirovanja. Časovna usklajenost delovanja robotov je dosežena tako, da delovanje vseh robotov poženemo sočasno. Pred delovanjem so roboti v stanju pripravljenosti in imajo v dosegu zadnjih IR senzorjev postavljeno oviro. Stanje pripravljenosti je časovno neomejeno in poteče ob odstranitvi ovire, zato moramo oviro pri vseh robotih odstraniti istočasno. Avtomat preide v stanje delovanja. Ob nastopu stanja delovanja se požene notranji časovnik robotov, ki šteje čas v milisekundah. S poskušanjem smo ugotovili, da je optimalen čas za prehajanje globalnih ukaznih stanj 4.5 sekunde. Za povprečno izvedbo ukaza potrebuje robot 3.5 sekunde nato pa ima na voljo še dodatno sekundo prostega časa, v primeru daljšega čakanja drugih robotov. Po preteku 4.5 sekund se sproži časovni dogodek, trenutno globalno stanje poteče in nastopi naslednje ukazno stanje. Izvedbeni ukazi se hranijo v seznamu na robotu. Števec stanj predstavlja tudi indeks izvedbenega ukaza v seznamu. S prehodom stanj na globalnem nivoju se števec povečuje in globalni izvedbeni ukaz se zamenja.

Drugi, lokalni nivo skrbi za delovanje znotraj globalnega stanja. Delovanje je odvisno od ukaznega stanja v katerem se robot nahaja. Podstanja globalnih stanj se spreminjajo z vhodimi zaznavami senzorjev.

Vsa štiri globalna stanja uporabljajo spredaj, spodaj nameščena reflektno–ambientna IR senzorja, saj ob premikanju robot sledi vodilu po podlagi preko zaznav bele in/ali črne podlage. Senzor zaznava reflektivnost podlage na lestvici od

vrednosti 0 do približno 950, kjer vrednost 0 predstavlja črno podlago, vrednost 950 pa svetlo belo podlago. Med črno in belo podlago razlikuje na podlagi podanega numeričnega praga vrednosti 500.

Zaradi preprostosti bomo v tem poglavju navedena reflektno–ambienta senzorja naslovili zgolj kot senzorja, kadar gre za drug senzor bo posebej poudarjeno.

4.2.1 Stanje - premik naprej

Stanje premika naprej predstavljata dve lokalni podstanji: stanje premika med polji in stanje premika znotraj polja. Za premik med polji uporabimo tehniko sledenja črni črti. Robot je primarno postavljen na podlago z obema senzorjema na črni podlagi (pravokotna stranica polja). Nato vklopimo oba motorja in pričnemo z gibanjem toliko časa, da oba senzorja zaznata belo podlago. Z gibanjem nadaljujemo s tehniko sledenja oziroma v našem primeru izogibanja črnemu vodilu. Robot se v tej fazi pravilno premika, kadar oba senzorja zaznavata belo podlago. Če levi senzor zaznava črno podlago, desni pa belo, je potreben popravek smeri v levo (dodamo moč desnemu motorju) in obratno, če levi zaznava belo podlago in desni črno podlago, uporabimo popravek smeri v desno (dodamo moč levemu motorju) do izravnave tj. zaznave bele podlage na obeh senzorjih. Ko oba senzorja hkrati ponovno naletita na črno podlago, vemo, da smo dosegli stranico naslednjega polja.

Sledi lokalno podstanje premika znotraj polja. Postopek je podoben, vklopimo motorje in se gibamo do dosega bele podlage, nato sledimo sredinskemu vodilu z morebitnimi vmesnimi popravki, ki so enaki prejšnjemu podstanju. Ob ponovni zaznavi črne podlage na obeh senzorji smo dosegli nasprotno stranico polja, kar pomeni, da je robot uspešno izvedel potezo premika naprej. Motorje ugasnemo.

4.2.2 Stanje - leva rotacija

Stanje leve rotacije je implementirano dokaj preprosto. Robot začne tako, da oba senzorja zaznavata črno podlago (trenutno stranico) in v danem trenutku vključi oba motorja. Levi motor se vrti vzvratno, desni motor se z enako močjo vrti naprej. Robot se vrti okoli svoje osi in znotraj polja. Ko robot zazna belo podlago na obeh

senzorji zabeleži, da se nahaja sredi obrata. Ko naslednjič z obema senzorjema zazna črno podlago (naslednja stranica) robot ugasne motorje in zaključi delovanje. Robot je uspešno izvedel potezo leve rotacije.

4.2.3 Stanje - desna rotacija

Stanje desne rotacije je implementirano na povsem enak način, kot je implementirano stanje leve rotacije. Edina razlika je v smeri vrtenja motorjev. Pri desni rotaciji levi motor vrti kolo naprej, desni motor pa z enako močjo vrti kolo vzvratno. Zopet gre za rotiranje okoli osi in znotraj polja. Ob ponovni zaznavi črne podlage z obema senzorjema ugasne motorje.

4.2.4 Stanje - mirovanje

Stanje mirovanja pomeni, da mora robot zaradi sledenja ukazom sinhronizacije abstraktnega plana v dani potezi/stanju mirovati. Robot z delovanjem še ni zaključil.

4.2.5 Dodatni stanji - stanje pripravljenosti in zaključno stanje

Dodatno stanje pripravljenosti, kot opisano v uvodnem delu poglavja služi zagonu sistema robotov in nam omogoča istočasni zagon večjega števila robotov s sočasno odstranitvijo ovire. Zaključno stanje doseže robot, ko je svojo nalogo dokončno opravil, torej ko je izvedel vse podane ukaze. V zaključnem stanju miruje in čaka na zaključek delovanja preostalih robotov.

4.2.6 Varnostno stanje - distančni senzorji

V stanju premika naprej, ves čas delovanja robota v ozadju vzporedno deluje nadzorni sistem za preprečitev trkov. Distančni senzorji ves čas preverjajo prostor 3-5 centimetrov pred robotom in v primeru zaznave ovire zasilno zaustavijo motorje.

Ob sprostitvi prostora pred robotom se motorji ponovno vklopijo in robot lahko prosto nadaljuje izvajane trenutnega ukaza.

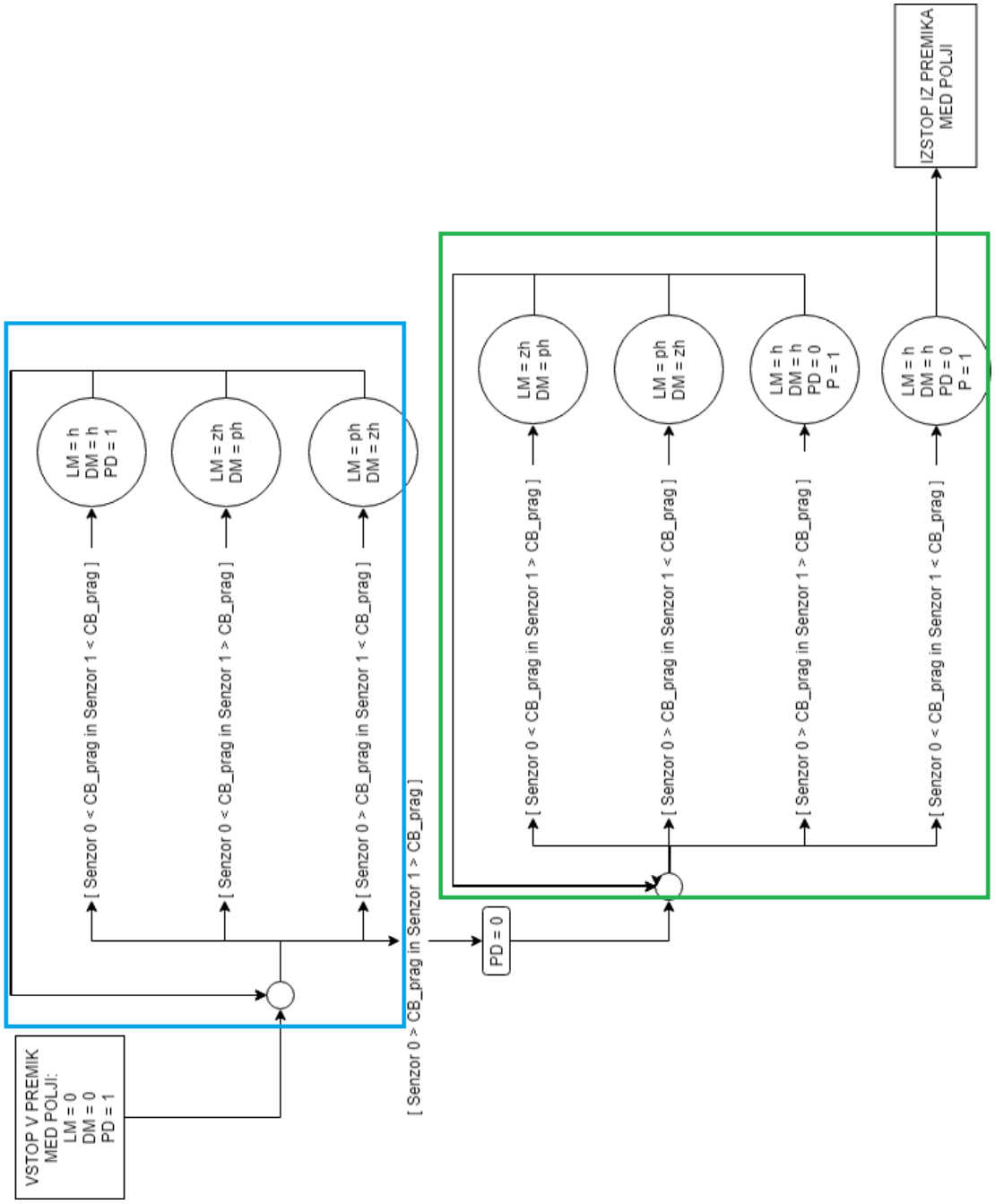
Celotno delovanje podrobneje prikazujeta sliki 4.2 in 4.3, ki prikazujeta izvedbene diagrame prehajanja stanj avtomata robotov.

Tehnične specifike delovanja robotov, ki so bile pridobljene s poskušanjem in oznake, ki se pojavijo na diagramih, podane so v obliki "[ime-oznake]":

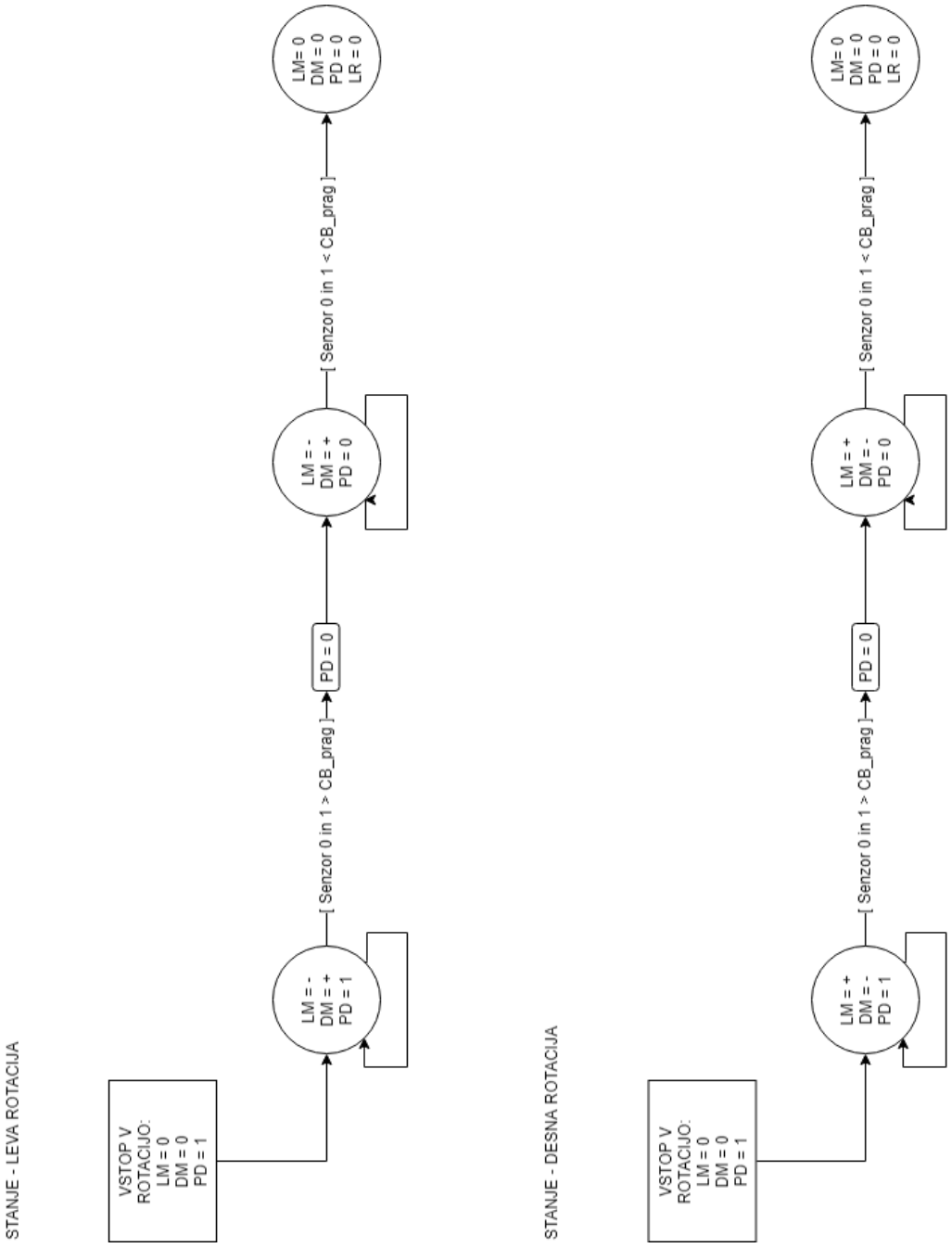
- Hitrost premikanja. Možne vrednosti od -500 do 500.
  - Privzeta hitrost [h] = 300, levi motor = [LM], desni motor [DM]
  - Popravki ob sledenju: pospeševalno kolo [ph] = privzeta hitrost / 2, zavirajoče kolo [zh] = - (privzeta hitrost/ 4)
  - Hitrost ob rotacijah: pospeševalno kolo [+] = (privzeta hitrost / 2) + 50, zavirajoče kolo [-] = - ((privzeta hitrost / 2) + 50)
- Časovni interval proženja časovnega dogodka: 4500 ms
- Vrednosti primerjalnih pragov:
  - Razlikovanje med črno in belo podlago. Možne vrednosti od 0-črna do 950-bela: črno beli prag [CB-prag] = 500
  - Ocenjevanje bližine objektov. Možne vrednosti od 0-daleč do 6000-blizu: sredinski senzor = 3000 (cca. 3cm) in ostali senzorji = 2500 (do cca. 5 cm)

Poleg navedenih tehničnih oznak se pojavijo še nekatere druge:

- PD – označuje "Prvi dotik". Pri stanju premika označuje prehod med polji (zaznavo nove stranice). Pri rotaciji označuje zaznavo vmesne bele podlage. Vrednosti 1 in 0 prestavljata veljavnost in neveljavnost pogoja.
- Levi in desni reflektno–ambientni IR senzorja: oznaki Senzor0 in Senzor1



Slika 4.2: Diagram prehajanja stanj znotraj globalnega stanja (ukaza) premik naprej. Modri okvir prikazuje delovanje prvega podstanja – prehod med polji oziroma sledenje vodilu. Zeleni okvir predstavlja delovanje drugega podstanja – premik robota znotraj polja.



Slika 4.3: Diagram prehajanja podstanj globalnih stanj leve rotacije in desne rotacije.

Poglavje 5

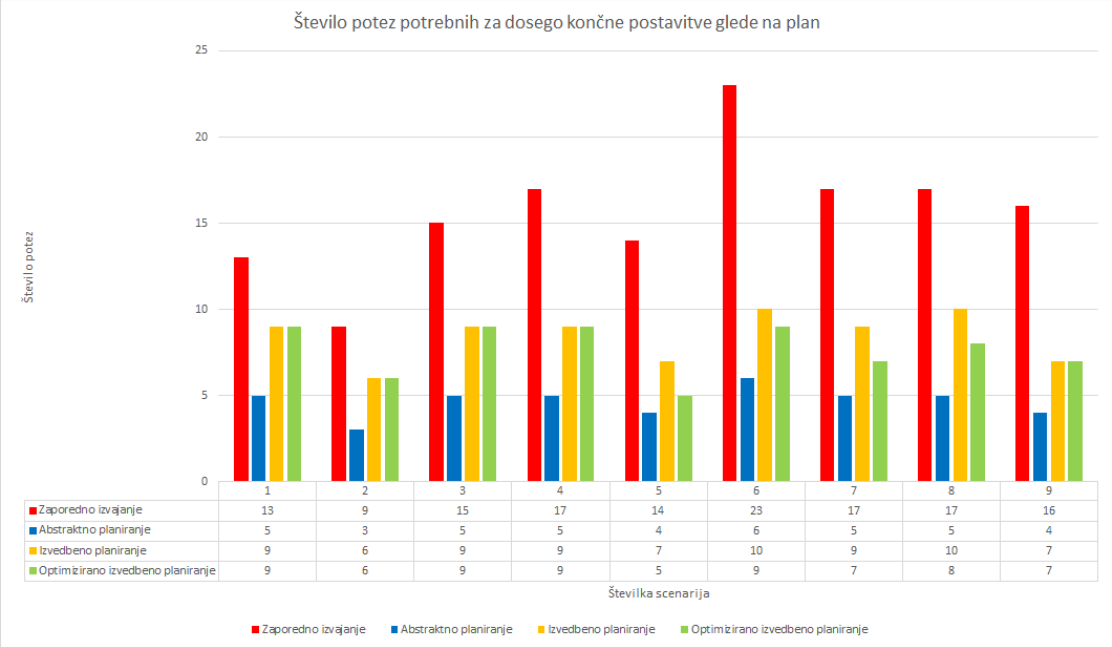
Evalvacija

5.1 Evalvacija optimalnosti

V okviru diplomskega dela smo izvedli devet različnih scenarijev vzporednega planiranja in izvedbe z roboti Thymio. Posnetke praktičnega dela izvedbe, si je mogoče ogledati na spletnem portalu Youtube [8]: `https://www.youtube.com/watch?v=Rlp3eDsAy_U`.

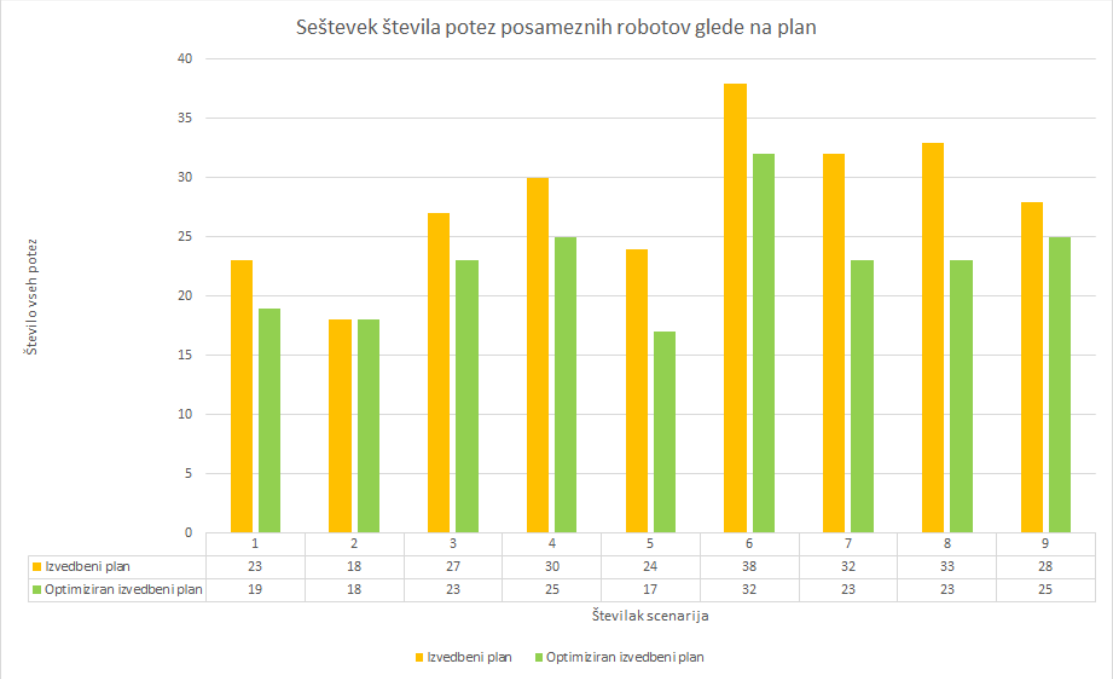
Optimizacijski kriterij, ki smo ga opredelili v podpoglavju 2.3, predvideva vzporedno gibanje robotov po pravokotni mreži z minimalnim številom globalnih potez za dosego želene končne postavitve robotov. Z namenom ugotovitve optimalnega izvajanja primerjamo število korakov brez vzporednega izvajanja, število korakov abstraktnih planov ter število korakov izvedbenih planov.

Tu gre poudariti, da ne moremo govoriti o optimalni izvedbi planov za dane primere gibanja robotov po pravokotni mreži, temveč optimiziramo zgolj izvedbene plane, ki so izpeljani iz optimalnih abstraktnih planov, vendar to ne zagotavlja optimalnosti izvedbenih planov v smislu čim manjšega števila globalnih potez. Namreč z upoštevanjem dodatnih ukazov rotacij in mirovanja bi z abstraktnim planiranjem lahko prišli do drugačnih planov, ki bi bili krajši od naših izvedbenih planov in bi potencialno lahko bili boljši.



Slika 5.1: Graf prikazuje število potrebnih potez za dosego končne postavitve. Primerja zaporedno izvedbo, abstraktno planiranje, izvedbeni plan ter optimiziran izvedbeni plan.

Iz grafa lahko razberemo prednosti vzporednega izvajanja. Čeprav je izvedbeno planiranje zahtevnejše kot abstraktno planiranje, lahko z optimizacijo in dobrim planiranjem število potez zmanjšamo dovolj, da se močno približamo številu teoretičnih potez. To je dober rezultat, saj abstraktno planiranje ne upošteva potrebnih potez rotacij, ki se jim v praktični izvedbi ne moremo ogniti. Vsekakor pa je vzporedno izvajanje gibanja robotov po pravokotni mreži veliko učinkoviteje kot zaporedno izvajanje po osnovnih pravili igre osmih ploščic.



Slika 5.2: Graf prikazuje seštevek vse potrebnih potez posameznih robotov za dosego končne postavitve. Primerja preprosto izvedbeno planiranje in optimizirano izvedbeno planiranje.

Če podrobneje pogledamo optimizacijo, torej plan poti vsakega posameznega robota in ne zgolj število globalnih potez, opazimo da se število potez znatno zmanjša. To je v določenih situacijah še pomembnejši podatek, kot število globalnih potez. Nakazuje namreč, da so posamezni roboti pogosteje in hitreje na voljo za nadaljnje izvajanje novih planov, saj so cilj dosegli hitreje. V okolju z dinamičnim izvajanje bi to pomenilo visoko prednost in dodatno učinkovitost izvedbenih robotov.

V grafu lahko zaznamo tudi primer, ko algoritem ni najbolj učinkovit. Do takšnega primera pride, kadar je za izvedbo potrebno zelo veliko število rotacij in zelo majhno število premikov.

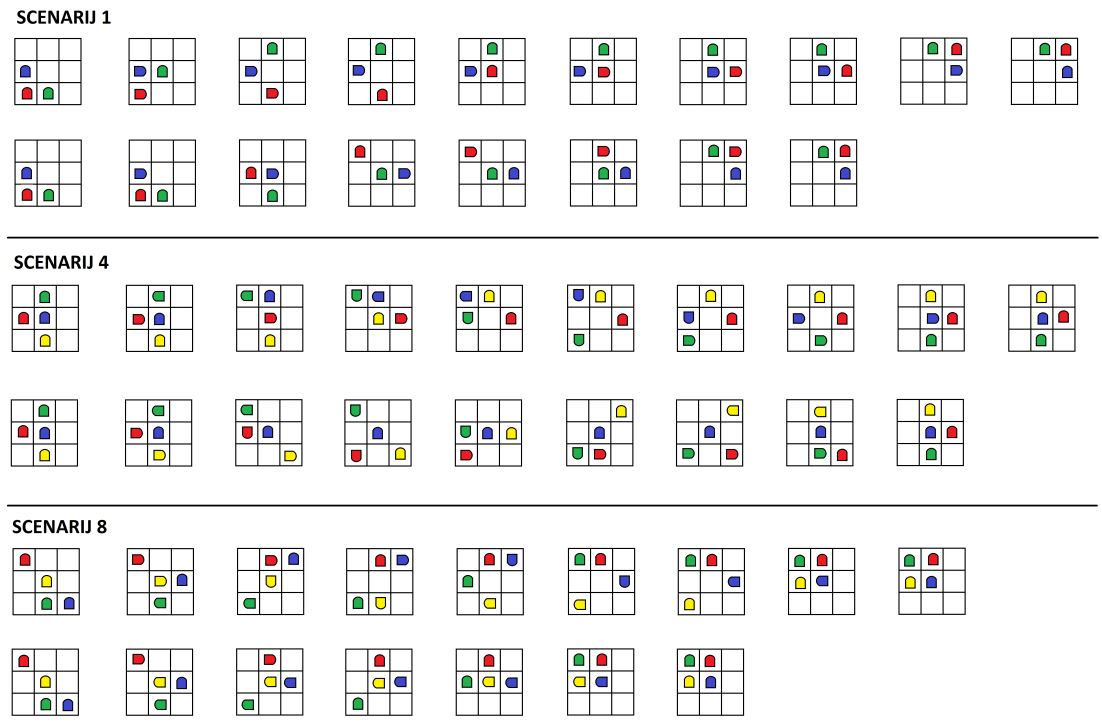
5.1.1 Ugotavljanje optimalnosti izvedbenih planov

S podrobnejšo ročno analizo izvedbenih planov smo poskušali ugotoviti globalno optimalnost delovanja robotov. Iskali smo kombinacije izvedb, kjer bi izvajanje lahko opravili v manjšem številu globalnih potez, kot izvedbeni plani na podlagi abstraktnih planov.

Pri preiskovanju planov smo se odločili na izvedbo vseh devetih scenarijev, ki jih prikazuje naš posnetek izvajanja [8]. Pri šestih izmed devetih scenarijev smo ugotovili optimalnost izvedbe. To so scenariji 2, 3, 5, 6, 7 in 9. V scenarijih 1, 4 in 8 pa smo ugotovili, da je gibanje robotov po pravokotni mreži mogoče izpeljati v manj potezah, torej plani, ki so posledica pretvorbe abstraktnih planov v izvedbene v teh treh primerih niso optimalni. Slika 5.3 prikazuje primerjavo izvedbenih scenarijev in najdenih optimalnih rešitev.

V izvedbenih scenarijih številka 1 in 8 smo našli optimalne plane izvedbe, ki so za dve potezi krajši od izvedbenih planov dobljenih s pretvorbo optimalnih abstraktnih planov. V izvedbenem scenariju številka 4 pa smo našli optimalno rešitev, ki je za eno potezo krajša od izvedbenega plana našega algoritma. Z analizo ugotovimo, da gre v vseh primerih za plane, kjer je na poti posameznih robotov veliko število rotacij. Rotacije upočasnijo izvedbo planov. Optimalne izvedbe so tiste, kjer je bilo mogoče najti alternativno pot, ki je vsebovala več povezanih zaporednih premikov robota v isto smer in s tem manj rotacij.

Če postavimo grobo oceno, lahko trdimo, da naš algoritem za pretvorbo deluje blizu optimalnega planiranja. Razlog za takšno oceno je dejstvo, da je v šestih od devetih planov našel optimalno rešitev. Pri preostalih treh planih, kjer optimalne rešitve ni našel, pa je bili odstopanje minimalno - ena oziroma dve potezi.



Slika 5.3: Primerjava izvedbenih scenarijev z optimalnimi rešitvami izvedbe. Prikazane so primerjave treh izvedbenih scenarijev 1, 4 in 8. Pri vsakem posameznem scenariju je v zgornji vrstici prikazan naš izvedbeni plan, v spodnji vrstici pa najden optimalen plan.

5.2 Evalvacija izvedbe, varnosti

Praktično izvedbo igranja igre lahko opišemo kot povprečno. Preprosti roboti Thymio, predvsem zaradi omejene komunikacije ne morejo zagotoviti visoko precizne izvedbe. Za takšno izvedbo, bi potrebovali dodaten nadzorni sistem, ki bi spremljal dogajanje na igralni plošči v celoti in bi izvedbene plane lahko prilagajal sproti, glede na dano situacijo.

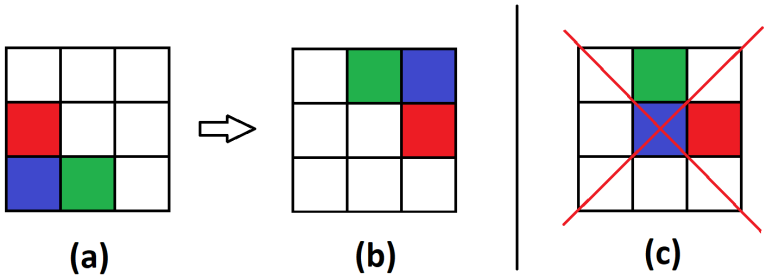
Na problem občasno naletimo tudi pri percepciji okolja. Različne odbojnosti materialov, distance, zamuda v zaznavi, umazanost ali obrabljenost pnevmatik, lahko pripeljejo do kritične napake, ki povzroči da robot med izvajanjem zaide

s poti. Ker nimamo nobenega zunanjega vira nadzora, se robot sam ne more vrniti na pravi položaj. Robot izgubi orientacijo in ga je potrebno ročno vrniti na položaj. Takšni dogodki so redki.

Če izvzamemo problem nadzora in komunikacije, pa so roboti kot avtomati implementirani zelo striktno z jasnimi določili, zato v sami izvedbi delovanja napaka ni mogoča. Roboti tekom delovanja sporočajo v kakšnem fazi delovanja se nahajajo. Vsakemu izmed robotov je dodeljena unikatna barvna oznaka RGB. Kadar barvna oznaka žari neprekinjeno robot izvaja operacijo premika ali rotacije, ko izvaja potezo mirovanja to nakaže z utripajočo barvno oznako.

Dodatno varnost izvedbe smo zagotovili s pomočjo preprostega sistema za preprečevanje trkov, ki v primeru zaznane ovire v smeri pred robotom, z dosegom nekje 3-5 cm, začasno zaustavi delovanje obeh motorjev in z delovanjem nadaljuje šele, ko se ovira umakne.

Glavna naloga izvedbenega planiranja poleg optimizacije je, da ohranimo pogoj dejanske izvedljivosti planov. Pri preiskovanju prostora najkrajših poti vse kombinacije niso vedno izvedljive, določenih končnih postavitev iz začetnih ni mogoče doseči [6]. Zato je pomembno, da ob postopku optimizacije ohranimo enak vrstni red in razmerje izvedbe med roboti. Namreč hitro se lahko zgodi, da s prehitevanjem oziroma optimizacijo lastne poti robot zapre pot drugemu robotu in mu onemogoči dokončanje izvedbe (Slika 5.4).



Slika 5.4: Slika prikazuje primer napačno optimiziranega plana. Sličica (a) prikazuje začetno postavitev, sličica (b) prikazuje želeno končno postavitev, sličica (c) pa primer napačne optimizacije. Rdeča in zelena ploščica svoj cilj dosežeta optimalno, modra ploščica pa svojega cilja ne more doseči.

5.3 Diskusija / izboljšave

Prostor za nadgradnjo izvedbenega sistema vsekakor obstaja. Velik preskok v zagotavljanju zanesljive izvedbe bi lahko omogočila že brezžični komunikacijska sposobnost, ki bi robotom omogočal sporočanje nevarnosti, neizpolnjevanja nalog, dinamičnega določanja izvedbe in dodeljevanja nalog, ter potencialno orientiranje ali določanje položaja s pomočjo informacij in zaznav preostalih robotov.

Delno oviro pri izvedbi oziroma že implementaciji sami predstavlja tudi programski jezik Aseba, ki je omejen in njegovo delovanje vodijo odzivi na dogodke. Če bi želeli sistem, ki bi bil sposoben samostojno na robotih adaptirati izvedbene plane ter jih prilagajati, bi morali dodati module, ki lahko uporabljajo naprednejši programski jezik, kot na primer Python ali C.

Tekom načrtovanja smo preizkušali več različnih načinov za vzpostavitev nadzorovane izvedbe s komunikacijo.

Glavni dve zamisli sistem sta bili:

- Samostojen sistem gruče Thymio z uporabo IR komunikacije
- Sistem vodenja s strani centralnega računalnika, ki dodeljuje naloge

Pri prvem, samostojnem sistemu smo si zamislili, da bi imeli ob strani igralne površine postavljenega robota nadzornika, ki bi vse preostale robote delavce usmerjal preko izdaje ukazov. Robot nadzornik bi bil povezan z računalnikom ali pa samostojen, poznal bi celovit plan in ga po delih predajal robotom delavcem. Razlogov za neuspeh je bilo več. Takšna izvedba bi bila časovno močno neučinkovita zaradi načina prejema informacij. Kratek doseg IR senzorjev bi zahteval, da se roboti pripeljejo do nadzornika in prejmejo ukaze, kar bi povzročilo izgubo orientacije. Drugi način, kjer bi uporabili princip igrice "telefončkov"in bi roboti delavci lahko drugim robotom delavcem predali ukaze vodje, bi povzročili veliko nepotrebnih potez rotacije in časovnega zamika. Pri uporabi IR komunikacije je problem tudi zahteva po medsebojni poravnavi senzorjev z namenom izmenjave informacij.

Drugi sistem pa je predvideval, da bi roboti bili povezani z računalnikom preko povezave USB in bi računalnik bil tisti, ki bi v celoti nadzoroval delovanje posa-

meznih robotov, planiral in predajal izvedbene ukaze direktno robotom. Vendar se je problem pojavil že pri začetku implementacije. Namreč vsi roboti Thymio imajo enak identifikator komunikacijskega vmesnika USB, kar pomeni da ne moremo naslavljati robotov po unikatnih identifikatorjih, niti ne morem na računalnik povezati več kot enega robota naenkrat.

Možnost za izboljšavo takšnega sistema pa se obeta v novi seriji robotov Thymio, ki omogoča komunikacijo robotov z računalnikom preko vgrajenih WiFi vmesnikov.

Kot se je izkazalo z analizo optimalnosti, bi dodatne izboljšave lahko dosegli tudi pri postopku planiranja. Namesto uporabljenih abstraktnih planov Janeža in pretvorbe le-teh v izvedbeno obliko, bi morali uporabiti algoritem, ki bi že ob planiranju upošteval dodatek k hevristično oceni, ki bi predstavljal ceno potrebnih rotacij na poti robota.

Poglavje 6

Zaključek

Cilj diplomskega dela je bil učinkovito implementirati izvajanje variante igre osmih ploščic z uporabo robotov Thymio. Implementacija nam je kljub preprostosti robotov Thymio in omejitvami tako robotov, kot tudi okolja, dobro uspela. Pokazali smo, da je vzporedno izvajanje tudi ob izvedbah realnega sveta veliko učinkovitejše kot zaporedno izvajanje ukazov. Ugotovili smo, da se lahko z dobrim načrtovanjem in optimizacijskim kriterijem močno približamo optimalni izvedbi, ki jo sicer zagotavlja pravilno abstraktno planiranje. Roboti Thymio se v obliki izvedbenih avtomatov pri delovanju dobro obnesejo, vendar so kot večina robotov deloma odvisni od vplivov izvedbenega okolja. Z implementacijo mehanizma za preprečevaje trkov smo zagotovili dodatno varnost pri gibanju robotov po pravokotni mreži. Delovanje bi bilo mogoče izboljšati z implementacijo centralnega nadzornega sistema ali z dodano sposobnostjo brezžičnega komuniciranja robotov. Po opravljeni analizi lahko potrdimo, da so izvedbeni plani po številu opravljenih potez zelo blizu optimalnim, v določenih primerih so optimalnim enaki, kadar niso pa so odstopanja zgolj manjša. Do odstopanj pride pri planih, ki vsebujejo veliko število rotacij. Algoritem za pretvorbo abstraktnih planov v izvedbene plane je zelo učinkovit, vendar ne zagotavljajo optimalne izvedbe, saj optimizira že obstoječe plane in ne išče novih rešitev v prostoru kombinacij gibanja robotov po pravokotni mreži.

Literatura

[1] Thymio graphic image. https://www.techykids.com/wp-content/uploads/2013/05/Thymio-Large_square-copy.jpg, 2018.

[2] S Adarsh, S Mohamed Kaleemuddin, Dinesh Bose, and K I Ramachandran. Performance comparison of infrared and ultrasonic sensors for obstacles of different materials in vehicle/ robot navigation applications. *IOP Conference Series: Materials Science and Engineering*, 149(1):012141, 2016.

[3] I. Bratko. *Prolog Programming for Artificial Intelligence*. International Computer Science Series. Addison-Wesley, 2011.

[4] Jernej Janež. Iskanje in izvajanje paralelnih planov pri variantah igre 8 kvadratov. Diplomska naloga, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2017.

[5] Rok Piltaver, Mitja Lustrek, and Matjaz Gams. Search pathology of 8-puzzle. 2007.

[6] Robert Sedgewick and Kevin Wayne. *Algorithms (Fourth edition deluxe)*. Addison-Wesley, 2016.

[7] Robert Tovornik. Github repozitorij - izvorna koda. https://github.com/roberttovornik/multi-robot-parallel-plan-execution.git, 2018.

[8] Robert Tovornik. Video: Parallel robot path plan execution - using thymio robots. https://www.youtube.com/watch?v=Rlp3eDsAy_U, 2018.