FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Kompilace algoritmů hledání cest pro skupinu malých mobilních robotů
Student:	Nestor Popov
Vedoucí:	doc. RNDr. Pavel Surynek, Ph.D.
Studijní program:	Informatika
Studijní obor:	Znalostní inženýrství
Katedra:	Katedra aplikované matematiky
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

Cílem práce je prozkoumat možnosti kompilace algoritmů hledání cest případně výsledných plánů pro skupinu malých mobilních robotů. Předpokládáme, že každý robot se má přesunout na určité místo, přitom nesmí dojít ke kolizi mezi roboty. Dostupné roboty mají omezené možnosti programování a pouze základní sadu senzorů. Je tedy nutné navrhnout způsob kompilace vhodného existujícího algoritmu pro multi-agentní plánovaní cest nebo plánu, který produkuje tak, aby výsledek provedený roboty odpovídal jejich bezpečnému přesunu na cílové pozice. Úkoly pro řešitele jsou následující:

1. Prozkoumat existující algoritmy pro multi-agentní hledání cest.
2. Seznámit se s robotickým vybavením dostupným na fakultě, konkrétně s OZOBOTy typu EVO.
3. Prozkoumat možnosti kompilace algoritmů hledání cest či jimi produkovaných plánů pro OZOBOTy s využitím jejich omezeného programového vybavení.
4. Úspěšnost kompilace ověřit se skutečnými roboty v relevantních navigačních scénářích.

Seznam odborné literatury

[1] Guni Sharon, Roni Stern, Ariel Felner, Nathan R. Sturtevant: Conflict-based search for optimal multi-agent pathfinding. Artif. Intell. 219: 40-66 (2015)

[2] Jiaoyang Li, Pavel Surynek, Ariel Felner, Hang Ma, T. K. Satish Kumar, Sven Koenig: Multi-Agent Path Finding for Large Agents. SOCS 2019: 186-187

[3] Anton Andreychuk, Konstantin S. Yakovlev, Dor Atzmon, Roni Stern: Multi-Agent Pathfinding with Continuous Time. IJCAI 2019: 39-45

Ing. Karel Klouda, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 4. února 2020

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Kompilace algoritmů hledání cest pro skupinu malých mobilních robotů

Bc. Nestor Popov

Katedra aplikované matematiky
Vedoucí práce: doc. RNDr. Pavel Surynek, Ph.D.

3. června 2020

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 3. června 2020 . . . . . .

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Popov, Nestor. *Kompilace algoritmů hledání cest pro skupinu malých mobilních robotů*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

V této práci autor zkoumá možnosti kompilace algoritmů multi-agentního hledání cest na skupině malých robotů – Ozobot Evo. Každý robot má zadané počáteční a cílové pozice na předem zadané fyzické mapě, která je reprezentací mřížkového grafu. Tito roboti se následně musí přemístit do svých cílových pozicí, přitom nesmí dojít ke kolizi mezi nimi. V práci je implementován framework pro roboty Ozobot Evo, který umožnuje vyplnit předem spočítané cesty pro agenty z algoritmů multi-agentního hledaní cest. Funkčnost tohoto frameworku je oveřená v různých experimentech.

Klíčová slova multi-agentní hledání cest, Ozobot, kompilace algoritmu hledání cest, programování Ozobotů

Abstract

Compilation possibilities of multi-agent pathfinding algorithms on group of small robots – Ozobot Evo are researched in this thesis. Starting and goal positions on predifined map which is representation of grid graph are assigned to each robot. Theese robots should move to their goal positions following the condition that no collision between robots could occur. In this work framework for robots Ozobot Evo is implemented, which allows theese robots to

follow pathes that were the output from multi-agent path finding algorithms. Functionallity of this framework is verified in relevant experiments.

Keywords Multi-agent pathfinding, Ozobot, compilation of pathfinding algorithms, programming of Ozobots

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Multi-agentní hledání cest (MAPF) je téma z oboru umělá inteligence, které má velké množství různých uplatnění v běžném životě.

V rámci daného tématu je vymyšleno hodně různých algoritmů, které mají slabé a silné stránky, jako je např. optimalita nebo rychlost vyplnění algoritmu.

Tyto algoritmy mohou najít uplatnění v různých odvětvích: komerční (např. plánování cest pro roboty ve skladištích Amazonu [1]), zábavní (např. počítačové hry [2]), letecké (řízení vzduchové dopravy [3]) atd.

V této práci jsou následující cíle:

- prozkoumat možnosti kompilace výsledných plánů z algoritmu MAPF na skupině malých robotů – Ozobot Evo (dále jen Ozobot),
- implementovat funkční framework, který umožní vyplňovat plán z algoritmu MAPF na těchto robotech,
- v rámci relevantních experimentu ověřit funkčnost frameworku

Tito programovatelní roboti byli zvoleni pro danou práci, neboť jsou velmi dostupní pro veřejnost tím, že jsou levní vzhledem k podobným programovatelným robotům: např. jsou 7krát levnější než roboti e-puck a 29krát levnější než roboti Khepera IV. Proto by bylo vhodné analyzovat, jaké možnosti tito roboti mají a jestli jsou vhodní pro realizování výsledných plánů z problematiky MAPF.

Obsahem teoretické části této práce je náhled do problematiky MAPF. V rámci této části jsou zavedený důležité definice potřebné pro MAPF a popsané některé algoritmy pro řešení problémů MAPF.

Praktická část je zaměřena na specifikaci robotů Ozobot a analýzu jejich programovacích možností. Následně se práce věnuje návrhu a implementaci funkčního frameworku, který umožní Ozobotům vyplňovat cesty, vytvořené pomoci algoritmů MAPF. Funkčnost frameworku je ověřená v rámci relevantních experimentů.

Kapitola 1

Cíl práce

Cílem teoretické části je zavést důležité pojmy z problematiky MAPF a představit běžné algoritmy, které jsou využívané pro řešení problémů MAPF. Následně je potřeba vybrat vhodné algoritmy pro praktickou část práce.

Cílem praktické části je seznámit se s roboty Ozobot (fyzické vlastnosti a programovací možnosti) a následně implementovat funkční framework, který umožní vyplňovat řešení, vytvořené pomocí algoritmu MAPF na těchto robotech. Funkčnost tohoto frameworku bude otestována v relevantních experimentech.

Kapitola 2

Teoretická část

2.1 Definice zakladních pojmů

2.1.1 Grafy

Důležitým pojmem při práci s problémem hledání cest je *graf*. Proto je potřeba definovat tento pojem a jiné související pojmy. Definice jsou převzaté z [4].

2.1.1.1 Definice grafu

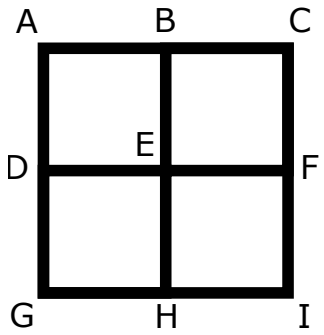
Graf je uspořádaná dvojice $G = (V, E)$, kde V je množina vrcholů a E je množina hran. Množina hran v tomto případě může být jak množinou vybraných dvouprvkových podmnožin množiny vrcholů v případě neorientovaného grafu, nebo množinou vybraných dvouprvkových uspořádaných dvojic $x \in V \times V$ v případě orientovaného grafu. Kromě formálního matematického zápisu lze také reprezentovat graf pomocí obrázku.

Vrcholy, které jsou spojeny hranou, jsou *sousední*. Množina vrcholů se značí $V(G)$. Množina hran se značí $E(G)$.

V této práci je často využíván *mřížkový graf*. Mřížkový graf je graf, jehož vrcholy jsou umístěny na 2D mřížce. Hrana tohoto grafu se může, ale nemusí vyskytnout jenom mezi horizontálně-sousedními nebo vertikálně-sousedními vrcholy na mřížce. Tím pádem do každého vrcholu mohou vést nejméně 0 nejvýše 4 hrany. Příklad mřížkového grafu 3 x 3 na množině vrcholů $\{A, B, C, D, E, F, G, H, I\}$ je na obrázku. 2.1.

2.1.2 MAPF

V této kapitole jsou zavedeny definice problematiky MAPF, jež jsou používány v dané práci. Obsah této kapitoly je vytvořen s využitím článku [5], [6] a knihy [7] Úkolem v problému MAPF je naplánovat cesty pro několik agentů, a to za předpokladu, že agenti budou sledovat tyto cesty souběžně a nesmí dojít ke kolizi (konfliktu) mezi nimi.



Obrázek 2.1: Mřižkový graf rozměru 3x3

Klasický MAPF problém lze popsat takto:

- Vstup do problému je trojice: $< G, s, t >$, kde $G = (V, E)$ je neorientovaný graf, $s : \{1, ..., k\} \to V$ je zobrazení pro počateční pozice agentů, $t : \{1, ..., k\} \to V$ je zobrazení pro cílové pozice agentů,
- Akce je zobrazení $a : V \to V$ takové, že $a(v) = v'$. $a(v)$ má následující význam: jestli se agent nachází ve vrcholu v a vyplní akci a, tím se dostane do vrcholu v'. Každý agent má dva typy akcí: pohyb a čekání. Při pohybu agent se přemístí z vrcholu v do sousedního vrcholu v' v grafu. Při čekání agent zůstává ve stávajícím vrcholu,
- Je předpokládáno, že čas je diskrétní a v každém okamžiku času,se každý agent nachází právě v jednom vrcholu grafu a může vyplnit právě jednu akci.

k-agentový stavový prostor tohoto problému je množina všech stavů, které představují různé způsoby, jak umístit tyto k agenty do $|V|$ vrcholů, s podmínkou že se v jednom vrcholu může nacházet nejvíce jeden agent. V počátečním a konečném stavu se agent i nachází v vrcholech $s(i)$ respektive $t(i)$. Stavový prostor tvoří orientovaný graf, ve kterém vrcholy jsou stavy a hrany mezi vrcholy je množina akcí, kterou musí vyplnit příslušní agenti, aby dostali do následujícího stavu.

Maximální faktor větvení b_{max} v k-agentovém stavovém prostoru je roven

$$b_{max} = a_k$$

kde a je počet všech dostupných akcí pro jednoho agenta.

Skutečný faktor větvení b_{skut} může být menší než maximální faktor, protože při k agentech mají někteří agenti menší počet legálních akcí. Stále platí vztah

$$b_{skut} = \mathcal{O}(b_{max})$$

což znamená, že s přibývajícím počtem agentů roste exponenciálně počet vrcholů grafu stavového prostoru.

Pro posloupnost akcí $\pi = (a_1, ..., a_n)$ a agenta i označíme jako $\pi_i[t]$ polohu agenta i po vyplnění prvních t akcí z posloupnosti akcí π s podmínkou, že počáteční poloha agenta i je $s(i)$. *Výsledný plán agenta* je taková posloupnost akcí π pro agenta i, že po vyplnění této posloupnosti agent i bude umístěn v $t(i)$. *Řešením* je množina k výsledných plánů agentů pro každého agenta.

2.1.2.1 Typy konfliktů v MAPF

Řešení v MAPF problému musí být bezkonfliktní, takže má smysl popsat, jaké běžné typy konfliktů mohou nastat mezi agenty.

Předpokládáme, že π_i a π_j jsou dva výsledné plány pro dva agenty i a j resp.

Vrcholový konflikt nastává mezi π_i a π_j, právě tehdy, když podle těchto plánů, agenty i a j plánují přemístit do stejného vrcholu v v nějaký okamžik času t.

Formálně: vrcholový konflikt mezi π_i a $\pi_j \Leftrightarrow (\exists t)\ (\pi_i[t] = \pi_j[t]\ )$,

Sledovací konflikt nastává mezi π_i a π_j, právě tehdy, když na základě těchto plánů agent i plánuje obsadit vrchol, na kterém se nacházel agent j v minulém okamžiku času.

Formálně: sledovací konflikt mezi π_i a $\pi_j \Leftrightarrow (\exists t)\ (\pi_i[t] = \pi_j[t+1]\ )$,

Konflikt výměny nastává mezi π_i a π_j, právě tehdy, když na základě těchto plánů, agent i plánuje obsadit vrchol, na kterém se nacházel agent j v minulém okamžiku času a agent j plánuje obsadit vrchol, na kterém se nacházel agent i v minulém okamžiku času.

Formálně: konflikt výměny mezi π_i a $\pi_j \Leftrightarrow (\exists t)\ (\pi_i[t] = \pi_j[t+1] \wedge \pi_i[t+1] = \pi_j[t]\ )$.

2.2 Algoritmy pro řešení MAPF problémů

Algoritmy pro řešení MAPF problémů lze rozdělit z hlediska optimálnosti na:

- optimální,
- neoptimální.

2.2.1 Optimální algoritmy

Existují dva hlavní přístupy při řešení MAPF problémů optimálně:

- Převod problému MAPF na jiný problém, např. SAT[1],
- Vyhledávání v k-agentovém stavovém prostoru.

Tato práce se bude věnovat řešení MAPF problémů pomocí vyhledávání v k-agentovém stavovém prostoru, proto jsou dále popsány algoritmy, které hodí pro tento účel.

2.2.1.1 A*

A* je klasický případ algoritmu pro vyhledávání v grafu.

Tento algoritmus vyhodnocuje vrcholy pomocí vztahů $f(n) = g(n) + h(n)$, kde $g(n)$ je cena cesty od počátečního vrcholu, do vrcholu n a $h(n)$ je heuristický odhad vzdálenosti od vrcholu n do cílového vrcholu. Navíc, jestli $h(n)$ bude splňovat podmínku konzistentnosti – pak A* je optimální. [7].

Algoritmus udržuje seznam otevřených vrcholů a seznam uzavřených vrcholů. V každém kroku algoritmus expanduje takový vrchol ze seznamu otevřených vrcholů, $f(n)$ kterého je minimální. Expandováním algoritmus přidává do seznamu otevřených vrcholů všechny sousedy expandovaného vrcholu, kteří nejsou obsaženi v seznamu uzavřených vrcholů. Expandovaný vrchol se následně přidává do seznamu uzavřených vrcholů. Důležitou vlastností je to, že jestli heuristika použitá pro algoritmus je konzistentní, pak v okamžiku, kdy algoritmus expanduje nějaký vrchol, je optimální cesta k tomuto vrcholu nalezená. [7]

Tento algoritmus lze použít pro řešení problémů MAPF, avšak v jeho klasické formě se příliš nepoužívá, z toho důvodu, že velikost stavového prostoru je exponenciálně závislá na počtu agentů, seznam uzavřených vrcholů není udržitelný v paměti počítače pro velké problémy MAPF. Dále je faktor větvení exponenciálně závislý na počtu agentů, proto např. pro mřížkový graf a 20 agentů počet sousedů počátečního stavu bude

$$5^{20} = 9.53 \times 10^{14}$$

Vygenerovat jenom tyto stavy už stává výpočetně obtížné. [8]

Pseudokód algoritmu A* [9] je na obrázku 2.2.

2.2.1.2 CBS

Popis algoritmu CBS a jeho pseudokód 2.3 je převzat z [6].

CBS (Conflict based search) je algoritmus, který řeší MAPF problém pomocí jeho dekompozice na velké množství jedno-agentových problémů hledání cesty s omezeními (constraints). Každý z těchto problemů lze vyřešit za čas úměrný rozměru mapy a délce řešení, ale počet takových problémů v rámci MAPF může narůstat exponenciálně.

[1] anglický: boolean satisfiability problem, český: problém splnitelnosti booleovské formule

Obrázek 2.2: Pseudokód algoritmu A*

```
open ← inicializace_prioritní_fronty
dist ← inicializace_tabulky()
prev ← inicializace_tabulky()
zařadít_do_fronty(open, S, h(S))
dist[S] ← 0
while ¬prazdný(open) do
    x ← vybrat_z_fronty open
    if x je cilový vrchol then
        return rekonstruovat_cestu(prev, x)
    foreach y ∈ sousedy(x) \ closed do
        d' ← dist[x] + c((x,y))
        if y ∉ open ∨ dist[y] ≥ d' then
            dist[y] = d'
            prev[y] = x
            if y ∉ open then
                zařadít_do_fronty(open, y, d'+h(y))
            else
                aktualizovat_klič(open, y, d'+h(y))
    closed ← closed ∪ {x}
```

Omezení může být *vrcholové* nebo *hranové*. Vrcholové omezení je uspořádaná trojice (i, v, t), kde agentu i je zakázáno navštěvovat vrchol v v čase t. Hranové omezení je uspořádaná čtveřice (i, v_1, v_2, t), kde agentu i je zakázáno pohybovat se z v_1 do v_2 v čase t.

Klíčová myšlenka tohoto algoritmu je udržovat a zvětšovat množinu omezení a hledat výsledné plány agentů, které budou konzistentní s těmito omezeními. Jestli tyto plány mají v sobě konflikty, a tím pádem nejsou validní, konflikty jsou vyřešeny přidáním nových omezení. V daném kontextu konfliktem může byt *vrcholový konflikt* nebo *hranový konflikt*.

Vrcholový konflikt je uspořádaná čtveřice (i, j, v, t), kde agenti i a j se nacházejí ve vrcholu v v čase t. Hranový konflikt je uspořadaná pětice (i, j, v_1, v_2, t), kde agent i se pohybuje z v_1 do v_2 a agent j se pohybuje z v_2 do v_1 v čase t.

CBS pracuje ve dvou úrovních: *vyšší*, ve které algoritmus hledá konflikty a přidává omezení, a *nižší*, ve které algoritmus hledá výsledné plány pro individuální agenty, které jsou konzistentní s množinou omezení.

Na vyšší úrovni algoritmus prohledává binární strom, který se nazývá *strom omezení*. Každý vrchol tohoto stromu obsahuje v sobě následující informaci:

množina omezení – každé z těchto omezení patří nějakému jednomu agentu. Kořenový vrchol obsahuje prázdnou množinu omezení. Dítě vrcholu vždy dědí množinu omezení jeho rodiče a přidává navíc jedno omezení pro jednoho agenta,

řešení – množina k vysledných planů všech agentů, které jsou konzistentní s množinou omezení v daném vrcholu. Tyto cesty jsou hledané v nižší úrovni,

celková cena řešení – součet cen výsledných plánů všech agentů z řešení, kde cena je délka řešení.

Pro prohledávání tohoto stromu se používá algoritmus best first search (BFS). Algoritmus BFS vybírá takové vrcholy ze stromu omezení, které mají nejmenší celkovou cenu řešení. Jestli jsou ceny rovné, pak se vybírá vrchol s menším množstvím konfliktů. Jestli množství konfliktů je stejné, pak se vybírá vrchol podle techniky FIFO[2].

Vrchol, vybraný v rámci vyšší úrovně, je následně zpracován v rámci nižší úrovně. Nižší úroveň vrací nejkratší cestu pro každého agenta i, která je konzistentní s množinou omezení daného vrcholu. Následně jsou tyto cesty validované.

Proces validaci kontroluje, jestli neexistují žádní dva agenti i a j, kteří se ve stejný čas budou nacházet ve stejném vrcholu v nebo se budou pohybovat po stejné hraně $\{v_1, v_2\}$. V případě, že takoví agenti neexistují, vrchol je validní a řešení je vráceno.

V opačném případě se do stromu omezení přidávají nové dva vrcholy, které dědí množinu vybraného vrcholu s jedním omezením navíc. Do jednoho vrcholu se přidává omezení (i, v, t) pro vrcholový konflikt nebo (i, v_1, v_2, t) pro hranový konflikt. Do druhého vrcholu se přidává omezení (j, v, t) pro vrcholový konflikt nebo (j, v_2, v_1, t) pro hranový konflikt. Po přidání těchto vrcholů do stromu omezení se pokračuje v hledání.

Na nižší úrovni lze pro vyhledávání výsledného plánu agenta i využít libovolný algoritmus vyhledávání pro jednoho agenta, ve kterém je potřeba ověřovat, že omezení jsou splněna. Takovým algoritmem může byt například A* s potřebnými úpravami.

2.2.2 Neoptimálni algoritmy

2.2.2.1 Hierarchický kooperativní A*

Popis algoritmu je převzat z [10].

V HCA* MAPF problém je rozdělen na řadu jedno-agentových problémů vyhledávání cesty. Toto vyhledávání je provedeno v 3D prostoru (dvě dimenze jsou prostorové, jedna dimenze je časová) a je realizované pomocí algoritmu

[2]First In First Out

Obrázek 2.3: Pseudokód algoritmu Conflict-based search

```
Vstup: MAPF instance
Root.omezení = Ø
Root.řešení = najit individualni cesty agentu pomoci lowlevel()
Root.cena = soucet cen individualnich cest agentu v Root.solution
vložit Root do OPEN
while OPEN není prazdný do
    P ← nejlepší vrchol OPEN// reseni s nejmeni cenou
    Validovat cesty v P dokud nenastane konflikt.
    if P nemá konflikt then
        return P.řešení// P je cíl
    C ← prvni konflikt (i, j, v, t) in P
    foreach agent i in C do
        A ← nový vrchol
        A.omezení ← P.omezení + (i, v, t)
        A.řešení ← P.řešení
        Aktualizovat A.řešení pomoci volani lowlevel(i)
        A.cena = soucet cen individualnich cest agentu(A.řešení)
        if A.cena < ∞ then
            vložit A do OPEN
```

A*. Tuto proceduru lze nazvat *prostorove-časovy A**, aby se odlišila od obvyklého používání A*, jež nazýváme *prostorový A**.

Po výpočtu cesty nějakého agenta je tato cesta poznamenána do speciální struktury – *rezervační tabulky*. Tato tabulka představuje seznam zarezervovaných vrcholů pro konkrétní agenty v nějakém vybraném čase. Takže při plánování cesty jednotlivými agenty pomoci prostorově-časového A* jsou zarezervované vrcholy z této tabulky považované za neprůchodné v uvedeném čase pro všechny agenty, kromě toho agenta, který zarezervoval vrchol.

Libovolná přípustná heuristika může být použita v tomto algoritmu pro hledání jedno-agentových cest. Příliš prosté heuristiky jako Manhattonova vzdálenost může způsobit slabý výkon.

Jednou z nejlepší heuristik pro tento algoritmus je *heuristika skutečné vzdálenosti*. Tato heuristika představuje nejkratší vzdálenost do cíle v takovém abstraktním modelu, ve kterém se ignorují jiní agenti, ale neignoruji se překážky. Takže v podstatě je to vzdálenost, která by se našla pomocí prostorového A*. Tuto heuristiku je potřeba spočítat v každém vrcholu, který byl přidán do seznamu otevřených vrcholů v prostorově-časovém A*.

Počítání této heuristiky od vrcholu, ve kterém potřebujeme tuto heuristiku, není nejlepším způsobem a může způsobit slabší výkon. Lepším způsobem je použít reverzní obnovitelný A* (RRA*).

RRA* provádí modifikované vyhledávání A* v opačném směru. Vyhledávání začíná v cílovém vrcholu pro agenta i – $t(i)$, a míří k počáteční pozici agenta i – $s(i)$. Místo toho, aby algoritmus skončil ve vrcholu $s(i)$, vyhledávání pokračuje, dokud zadaný vrchol N nebude expandován. Jestli je přitom pro vyhledávání použita konzistentní heuristika, jako např. manhattonovská heuristika, potom z vlastností A* plyne, že jakmile je vrchol expandován, pak nejkratší cesta k tomuto vrcholu je nalezena. Takže v okamžiku expandování vrcholu N je známa nejkratší cesta od N do $t(i)$.

RRA* vrací vzdálenost na požadavek. V okamžiku, kdy je potřeba zjistit vzdálenost od vrcholu N do $t(i)$, algoritmus kontroluje, jestli vrchol N již je v seznamu uzavřených vrcholů – v tomto případě vzdálenost je známá a může být vrácena. Jestliže tomu tak není, pak RRA* pokračuje ve vyhledávání až vrchol N nebude expandován.

2.2.2.2 Greedy CBS

Optimální MAPF algoritmus CBS, který byl popsán v 2.2.1.2, může být následně relaxován, čímž, ztratí svou optimalitu, ale bude schopen najít řešení rychleji. K popisu algoritmu byl použitý článek [11].

Greedy CBS (GCBS) používá skoro stejný framework jako CBS s jediným rozdílem – na vyšší a nižší úrovni GCBS preferuje expandovat vrcholy, které s větší pravděpodobností budou produkovat validní (ale není nutné optimální) řešení rychleji.

Při relaxování vyšší úrovně hlavní myšlenka je upřednostňování takových vrcholů ze stromu omezení, které jsou nejbližší k validnímu řešení. Každý necílový vrchol stromu omezení obsahuje nevalidní řešení, které má v sobě vnitřní konflikty. Tyto konflikty mohou sloužit pro spočítání *heuristiky konfliktů*, která umožní upřednostňovat nejméně konfliktní vrcholy ze stromu omezení, které nejvíce pravděpodobně povedou k validnímu řešení. Takovouto heuristikou může být počet párů agentů, kteří mají alespoň jeden konflikt mezi sebou. Tato heuristika má nejlepší rovnováhu mezi jednoduchostí a výkonem [11],

Při relaxování nižší úrovně lze také použít heuristiku konfliktů. To znamená, že při hledání cesty pro jednoho agenta (např. pomocí A*), jsou upřednostňovány stavy, které nejsou v konfliktu s jinými agenty.

2.2.3 Výběr algoritmu

Problém hledání optimálního řešení algoritmu je NP-těžký, proto optimální algoritmy jsou primárně využívané u problémů, které generují relativně malý stavový prostor. U těchto problémů je algoritmus schopen doběhnout v rozumné době.

Velmi často lze vystačit s neoptimálním, ale bezkonfliktním řešením tohoto problému. Neoptimální algoritmy jsou schopné dodat řešení MAPF problému v dost kratší době než optimální algoritmy, ale toto řešení nemusí být právě optimální.

Pro použití frameworku, který je implementován v praktické části není důležitá optimalita jako taková, proto v případech, ve kterých optimální algoritmus není schopen doběhnout v rozumné době, není velký problém použít neoptimální MAPF řešič. Stále jsou však upřednostňované optimální řešiče, protože jsou schopné generovat nejkratší řešení a tím pádem by program pro Ozoboty spotřeboval méně paměti.

Kapitola 3

Praktická část

3.1 Popis problému

V rámci praktické části je úkolem specifikovat Ozobota a analyzovat jeho možnosti (fyzické a programovací). Dále vzhledem k možnostem Ozobota je potřeba navrhnout framework, který bude běžet na Ozobotech a pomocí kterého Ozoboti budou vyplňovat výsledný plán z MAPF algoritmů s co nejmenším možným počtem kolizí.

S využitím návrhu je potřeba implementovat framework. Testování frameworku je provedeno pomocí relevantních experimentů. V rámci testovaní je také potřeba použít nějaký existující řešič MAPF problémů nebo naimplementovat vlastní.

3.2 Specifikace robotu Ozobot

Ozobot je malý programovatelný robot, který vypadá následovně 3.1.

Robot se umí pohybovat jak dopředu, tak i dozadu. K pohybu využívá své hlavní aktuátory – dvě kola. Pomocí těchto kol se robot také umí otáčet kolem své osy stejným způsobem, jako to dělá tank: jedno kolo se točí dozadu, druhé se točí dopředu. Rychlost, se kterou se robot buď pohybuje, nebo otáčí, a také délku pohybu lze nastavovat v rámci jeho programování. Další aktuátory, které robot má, jsou LED indikátory a zvukové reproduktory.

Robot má v sobě dva druhy senzorů, a to senzory přiblížení, dva jsou vepředu a dva vzadu, a optické senzory. Optické senzory se nachází přímo pod robotem, jsou to senzory rozpoznání barev a senzory rozpoznání čáry. Všechny tyto senzory lze využít při programování robota.

Senzory přiblížení jsou využívané pro detekci překážek ze čtyř stran: zleva dozadu nebo dopředu a zprava dozadu nebo dopředu. Senzory rozpoznání čáry slouží k detekci čáry pod robotem. S využitím tohoto senzoru se Ozobot může úspěšně pohybovat po čáře popř. detekovat protínání dvou čar. Barevný senzor se používá primárně pro rozpoznání barvy čáry.

Obrázek 3.1: Programovatelní robot Ozobot, převzato z: https://files.Ozobot.com/stem-education/educator-botcamp.pdf

Robot má vnější tlačítko, které je primárně určeno k zapínání/vypínání Ozobota. Toto tlačítko se dá následně přeprogramovat.

Ozobot má také vestavěný bluetooth modul. Tento modul se využívá primárně pro spojení robota s mobilním zařízením a nahrání programu.

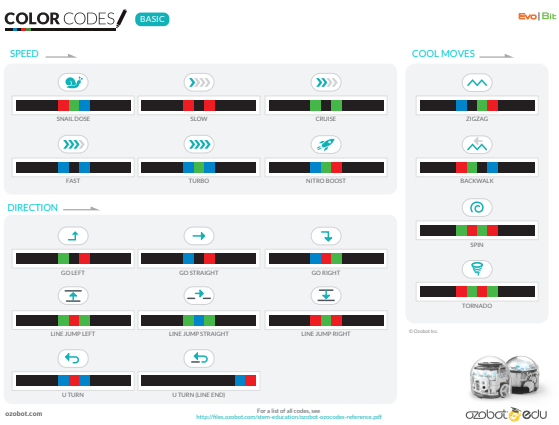
3.2.1 Způsoby programování robota Ozobot

Celkem existuje dva způsoby programování Ozobota:

- Pomocí barevných kódů,
- Pomocí vizuálního editoru „Ozoblockly Editor".

Programování pomocí barevných kódů tento způsob programování využívá toho, že Ozobot má vnější barevné senzory. Pomocí těchto senzorů robot může rozpoznávat barevné kódy, které představují nějakou malou posloupnost barevných obdélníčků. Tato posloupnost jednoznačně určuje nějakou konkrétní akci Ozobota. Ozobot má v sobě předinstalovanou sadu různých příkazů, které jsou zakódované pomocí posloupnosti různých barev. Příklad některých akcí je na obrázku 3.2

Následně musí být tyto barevné kódy umístěny na černé čáře, po které se Ozobot pohybuje. V okamžiku, kdy Ozobot projede nějakou barevnou posloupnost a zpracuje ji, začne vyplňovat akci odpovídající tomuto barevnému kódu. Tento způsob programování je určen primárně



Obrázek 3.2: Příklad některých barevných kódu, převzato z: https://files.Ozobot.com/stem-education/Ozobot-ozocodes-reference.pdf

k vzdělávacím účelům – např. pro výuku programování u dětí, kvůli jeho intuitivní pochopitelnosti,

Programování pomocí webového editoru ozoblockly editor je vizuální editor, ve kterém se dá vytvořit program pro Ozoboty v speciálním jazyce – Ozoblockly. Ozoblockly je procedurální vizuální programovací jazyk.

Uživatel má možnost vytvořit vlastní účet v editoru, pod kterým následně může vytvářet a uchovávat programy. Program pro Ozoboty se sestavuje ze sekvence bloku, které jsou sloučené do kategorie pro lehčí vyhledávání v rámci programování. Celkem je 15 kategorií bloků pro Ozobot EVO:

Pohyb bloky, které jsou určeny pro volný pohyb Ozobota,

Jízda po čáře bloky, které jsou určeny pro pohyb Ozobota s využitím navigace po čáře,

Světelné efekty bloky, pomocí kterých se dá využívat LED senzory Ozobota,

Zvuky bloky, pomocí kterých se dá využívat zvukových reproduktorů Ozobota,

Senzory bloky, pomocí kterých se dá využívat senzorů přiblížení Ozobota,

Tlačítko bloky, pomocí kterých se dá přeprogramovat vnější tlačítko Ozobota,

Časování bloky, které umožňují Ozobotu pracovat s reálným časem – napr. čekání,

Ukončení bloky, které slouží pro vypínání Ozobota,

Logika bloky, které reprezentují různé logické operátory a podmíněné operátory, které existují ve většině procedurálních programovacích jazyků,

Cykly bloky pro práci s cykly (FOR, WHILE),

Matematika bloky pro manipulaci s čisly,

Proměnné bloky pro práci s proměnnými,

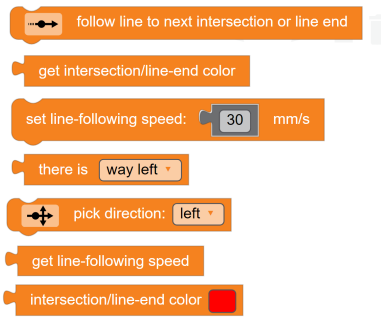
Funkce bloky pro práci s funkcemi,

Pole bloky, pomocí kterých se dá alokovat pole a pracovat s ním,

Seznam bloky pro práci se seznamem, který má rozšířenou funkcionalitu vzhledem k poli.

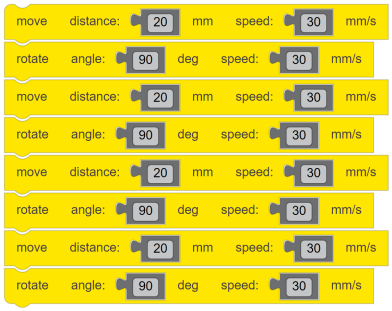
Pro lepší představitelnost na obrázcích 3.3 a 3.4 jsou bloky z kategorie pro pohyb Ozobota: „pohyb“ a „jízda po čáře“.

Obrázek 3.3: Bloky z kategorie „Pohyb“



Obrázek 3.4: Bloky z kategorie „Jízda po čáře“

Programování v editoru je spojení těchto předem definovaných bloků. Příklad jednoduchého programu, ve kterém Ozobot projede trasu ve formě čtverce a tím se vrátí na svoje původní místo, je na obrázku 3.5



Obrázek 3.5: Program výtvořený v webovém editoru

V editoru existuje funkce náhledu programu v syntaxi jazyka JS[3]. Program na obrázku 3.5 by vypadal při náhledu v JS takto 3.6. Pro lepší přehlednost budou níže různé ukázky programu pro Ozoboty demonstrovány s použitím této funkcionality. Rozpracovaný program lze serializovat a stáhnout jako XML soubor, který má speciální strukturu a následně zpátky nahrát tento XML soubor do editoru. Struktura XML souboru s programem je jednoduchá. Bloky ve vizuálním editoru, ze kterých se tvoří program, jsou reprezentovány pomoci elementu s tagem `block` v XML. Uvnitř tohoto elementu se mohou nacházet jiné elementy s tagem `block` a to v obalujícím elementu s tagem `next`, nebo jiné pomocné elementy s hodnotami, které popisují tento block.

Posloupnost bloku v programu je reprezentovaná rekurzivním vnořením elementů `block` do sebe. Kostru této struktury lze vidět na obrazku 3.7

V rámci dané práce bude využito programování pomocí webového editoru. Tento typ programování je rychlejší vzhledem k programování pomocí ba-

[3]Programovací jazyk JavaScript

Obrázek 3.6: Přehled programu z obrazku 3.5 v jazyce JS

```
move(20, 30);
rotate(90, 30);
move(20, 30);
rotate(90, 30);
move(20, 30);
rotate(90, 30);
move(20, 30);
rotate(90, 30);
```

Obrázek 3.7: Kostra struktury XML soubora s programem

```
<block>
    ...
    <next>
        <block>
            ...
        <block>
    </next>
    <value>
    ...
    </value>
</block>
```

revných kódů a zároveň umožnuje vytvářet komplexnější programy než s využitím prvního způsobu.

3.2.2 Nahrání programu do Ozobotů

Webový editor poskytuje dvě moznosti pro nahrání hotového programu do Ozobota:

Přes barevné senzory Ozobota malá část obrazovky začne rychle blikat posloupnost barev, která představuje vlastní zakódovaný program. Ozobot následně má možnost přečíst tuto posloupnost barev pomocí svých barevných senzorů, dekódovat ji, a tak dostane kompletní program do své paměti. Následně lze spustit přečtený program pomocí dvojitého zmáčknutí tlačítka na Ozobotu,

Pomocí bluetooth spojení daný způsob umožňuje předat program do Ozobota s využitím technologie bluetooth. K tomu je potřeba mít moderní mobilní zařízení (mobil, tablet atd.) a nainstalovat aplikaci od vývojářů Ozobotů – Ozobot EVO (aplikace je veřejné dostupná v Apple store nebo Play marketu). Pomocí aplikace lze připojit a řídit více Ozobotů najednou. Po vstupu do svého účtu uživatel dostává přístup ke všem svým programům a má možnost nainstalovat a spustit libovolný program na všech Ozobotech synchronně.

Kvůli tomu, že je potřeba mít centrální řídicí prvek pro spuštění programu na všech Ozobotech synchronně (což v tomto případě je aplikace Ozobot EVO), budou programy nahrávány přes bluetooth spojení.

Výrazným nedostatkem aplikace je to, že není možné spustit na všech Ozobotech, které jsou připojené k zařízení s programem, různé programy synchronně (ale existuje možnost spustit nějaký vybraný program na všech Ozobotech synchronně). Jelikož je při vyplňování plánu potřeba aby Ozoboty začinali vyplňovat plán syncrhonně, je nutné na všech Ozobotech nainstalovat

stejný program. To má nevýhodu – každý Ozobot bude mít ve své paměti cesty jiných Ozobotů, které nepotřebuje, protože výsledné řešení již bylo bezkonfliktní.

Samozřejmě lze také mít několik různých instancí aplikací, připojit každého Ozobota k právě jedné aplikaci, nahrát na každého Ozobota různý program a následně se pokusit spustit tyto programy synchronně. Tento způsob je ale složitě realizovatelný z technického pohledu – je k tomu třeba tolik mobilních zařízení, kolik je Ozobotů, a proto se práce tímto nebude zabývat.

3.3 Návrh frameworku

Cílem je navrhnout takový framework, který umožní vyplnit výsledný plán z MAPF algoritmů na robotech s co nejvyšší přesností, a tím pádem s co nejmenším počtem kolizí mezi Ozoboty.

Dostupná paměť pro program uvnitř Ozobota je relativně malá (≈1MB). Kvůli tomu, že každý program bude obsahovat v sobě cesty všech Ozobotů (z důvodu synchronního startu), je potřeba navrhnout framework kompaktně. Jelikož se na každém Ozobotu spustí stejný program, je potřeba objevit, jak rozlišovat konkrétní cestu, kterou musí Ozobot vyplňovat v tomto programu.

3.3.1 Mapa

Ozoboty potřebují fyzické prostředí, ve kterém se budou pohybovat. Jelikož algoritmus MAPF hledá řešení pro agenty na mřížkovém grafu, toto prostředí má být fyzickou reprezentací použitého mřížkového grafu na papíru. Označíme toto prostředí termínem mapa. S ohledem na možnosti Ozobota je mapa vytvořena následujícím způsobem:

- Každý vrchol grafu na mapě je označen protínáním dvou čar (dále jen vrchol mapy). Pro Ozobota je důležité toto označení, aby mohl správně rozpoznat vrchol. Protínání ve formě „L“ není validní a Ozobot může selhat při rozpoznání vrcholu pomocí svých senzorů,
- Hrana grafu je na mapě reprezentována jako čára spojující dva vrcholy (dále jen hrana mapy). Kvůli tomu, že hrany v původním grafu nemají váhu, pak všechny tyto hrany mapy musí mít stejnou délku,
- Čára musí být dostatečně tlustá (šířka přibližně 0,5 cm).

Vzhledem k abstraktnímu modelu MAPF ve kterém agenti nemají fyzický rozměr, Ozoboti zabírají nějaké fyzické místo, proto délka jedné hrany mapy nesmí být příliš malá. Pokud tomu tak není Ozoboti mohou dostát do fyzické kolize.

3.3.2 Režimy pohybu Ozobota

Aby se Ozobot mohl dostat z počáteční pozice do libovolného dostupného vrcholu mapy, musí mít minimálně dvě akce: pohyb dopředu a otáčení. Pohybovat se Ozobot může ve dvou režimech: volný pohyb nebo pohyb po čáře. Příkazy pro Ozobota ve volném pohybu jsou zobrazeny na obrázku 3.3 Hlavním nedostatkem volného pohybu je nepřesnost pohybu – kvůli hardwaru různí Ozoboti mohou projíždět různou délku v rámci stejného pohybového příkazu. Tím pádem se Ozoboti budou častěji dostávat do kolize, nebo se pohybovat mimo hranu mapy.

Pohyb po čáře, příkazy jsou zobrazeny na obrázku 3.4 garantuje, že se Ozobot bude pohybovat přímo po hraně mapy až do následujícího vrcholu, protože ta je tvořena čarou. V následujícím vrcholu znovu zvolí nějakou akci atd. Nedostatkem tohoto režimu je, že se Ozobot neumí pohybovat dozadu (pohyb dozadu je povolen jenom v režimu volného pohybu), což by pomohlo snížit počet otáčení (a tím cenu cesty) při pohybu od startu do cíle.

Samotné otáčení (bez následujícího pohybu) Ozobot může vyplňovat jenom v režimu volný pohyb. V režimu pohyb po čáře sice existuje otáčení, ale vždycky je následováno pohybem. Nakonec pro implementaci pohybu mezi vrcholy mapy je vhodný režim „pohyb po čáře“, pro implementaci rotace - režim „volný pohyb“.

3.3.3 Akce Ozobota

S využitím informací z sekcí 3.3.2 Ozobot bude mít následující akce:

1. Pohyb po čáře dopředu do vedlejšího vrcholu mapy.
2. Otáčení o 90 stupňů doleva.
3. Otáčení o 90 stupňů doprava.
4. Čekání.

Jelikož algoritmus MAPF je diskrétní, je důležité, aby každá akce trvala stejný čas. Předpokládejme že zvolený čas na každou akci – N sekund a je známá délka hrany mapy – e mm. Z těchto údajů lze vypočítat ostatní potřebné parametry pro Ozobota:

- Rychlost pohybu dopředu v mm/s:

$$\frac{N}{e}$$

- Pro vypočítání rychlostí otáčení je potřeba znát vzdálenost mezi koly Ozobota. Tato délka je 23 mm po změření měřítkem (11,5 mm je délka do středu Ozobota – rádius).

Z toho plyne že při otáčení o 90 stupňů na místě Ozobot projíždí

$$\frac{2 \cdot \pi \cdot 11,5}{4}$$

což je zhruba 18,064 mm. Tím pádem výsledná rychlost je $\frac{18,064}{N} mm/s$.

Je důležité poznamenat, že tyto teoretické výpočty nemusí přesně platit v praktických scénářích, proto je potřeba tyto parametry dodatečně odkalibrovat. Při kalibraci je potřeba upravit parametry tak, aby každá akce Ozobota trvala co nejblíže k referenční době trvání akce.

3.4 Implementace frameworku

3.4.1 Vyplnění cesty Ozobotem

Cesta se pro Ozobota skládá z posloupnosti akcí, vysvětlených v sekci 3.3.3. Pro uchovávání cesty v programu Ozobota je využíván datový typ List (seznam), který garantuje pořadí elementů. Maximální délka seznamu je omezena na 127 prvků, což by stačilo pro nevelké mapy. Kód pro zpracování cesty Ozobotem je na obrázku 3.8. Podprogram projde každý element v seznamu od začátku a podle hodnoty elementu Ozobot, na kterém běží tento program, vyplní odpovídající akci.

3.4.2 Konfigurace Ozobota

Kvůli tomu, že všichni Ozoboti budou vyplňovat stejný program, je potřeba navrhnout jakým způsobem předat do Ozobota informaci o tom, kterou cestu robot musí vyplňovat. Autorovi práce se nepodařilo najít žádný způsob, jak rozlišit Ozoboty za běhu programu (nemají sériové číslo nebo ID), proto vymyslel konfigurační funkci, která využívá toho, že Ozobot má programovatelné vnější tlačítko.

Před spuštěním programu bude mít uživatel čas, aby zadal pozici každému Ozobotu, která nabývá hodnot 0 až 5 a odpovídá pořadí seznamu souřadnic agentů z definice problému MAPF 3.4.3, a to zmáčknutím vnějšího tlačítka – jedno zmáčknutí inkrementuje pozici Ozobota o 1 mod 6. Po zvolení pozice musí uživatel dát Ozobota na vrchol na mapě, který odpovídá jeho počáteční pozici. Ozobot musí být umístěn na sever.

Po skončení konfiguračního času bude do seznamu přidána cesta, která odpovídá dané pozici, a Ozobot ji začne vyplňovat pomocí podprogramu z 3.4.1. Podprogram pro přidání cesty je na obrázku 3.10. Konfigurační funkce je na obrázku 3.9.

3.4.3 MAPF řešič

K získání bezkonfliktních cest mezi roboty je potřeba použít nějaký MAPF řešič. Pro účely testování frameworku byly naimplementovány dva různé řešiče

Obrázek 3.8: Podprogram, který se používá pro vykonaní cesty Ozobotem

```
while (i < LIST.length) {
  /*akce pohyb dopředu po čare*/
  if (getFromListAtIPosition() == 0) {
    /*nastaví se rychlost pohybu po čare*/
    setLineFollowingSpeed(s);
    /*tento přikaz spolu s nasledujicim umožní*/
    /*pohybovat po čare dopředu*/
    findLine();
    followLine();
  }
  /*akce otačení doprava*/
  else if (getFromListAtIPosition() == 1) {
    /*otačení o 90 stupnu s rychlosti n mm/s*/
    rotate(90, n);
  }
  /*akce otačení doleva*/
  else if (getFromListAtIPosition() == 2) {
    /*otačení o -90 stupnu s rychlosti n mm/s*/
    rotate(-90, n);
  }
  /*akce čekaní*/
  else if (getFromListAtIPosition() == 3) {
    /*čekaní t sekund*/
    wait(t);
  }
  i = i + 1;
}
```

Obrázek 3.9: Funkce pro konfiguraci Ozobotu

```
function tenSecondConfig() {
  for (var count = 0; count < 100; count++) {
    /*čekaní 100 ms*/
    delay(10);
    /*nastaví barvu odpovidajici počtu zmačknutí tlačitka*/
    setTopColorAccordingToPosition(get_button_press_count() % 6);
    /*nastaví pozice Ozobotu*/
    robotNumber = get_button_press_count() % 6;
  }
}
```

Obrázek 3.10: Přidaní cesty Ozobota do seznamu

```
if (robotNumber == 0) {
  appendToList(0);
  appendToList(1);
  ...
}
else if (robotNumber == 1) {
  ...
}
...
```

v jazyce Java: optimální – CBS z 2.2.1.2 a neoptimální HCA* z 2.2.2.1. Vlastní implementace je použita, neboť je potřeba řešič, který bude řešit nadstandardní MAPF problem[4] ,ve kterém je přidána navíc jedna akce pro agenty – otáčení, a se akce pohyb dovoluje pohybovat jen ve směru, kterým se robot dívá.

Vstupem řešičů je textový soubor obsahující definici MAPF problemu – informaci o grafu a agentech. Graf se zadává následujícím způsobem: na začátku se zadávají rozměry mřížkového grafu, pak následuje počet agentů a jejich počáteční a cílová pozice pro každého z těchto agentů. Dále je počet překážek v grafu a seznam jejich pozice. Překážkou se rozumí vrchol, do kterého se agent nesmí pohybovat. Na konci je počet vynechaných hran, po kterých se Ozobot nesmí pohybovat, a jejich seznam.

Příklad definice problému s popisem (komentář je označen následujícím způsobem: /* komentář */) je na obrázku 3.11.

Vystup řešiču se zapíše do java struktury `List<List<DirectionCoord>>`. Vnořený list – `List<DirectionCoord>` v sobě obsahuje posloupnost souřadnic Ozobota, která reprezentuje výslednou cestu agentu v MAPF problému. Tím pádem vnější list v sobě obsahuje jednotlivé cesty pro každého agenta.

3.4.4 Předání cest do programu Ozobota

Výstupem MAPF řešiče, který byl implementován pro testování frameworku je seznam výsledných plánů agentů pro každého agenta. Kvůli tomu, že program Ozobota může být reprezentován jako soubor v XML formátu s určitou strukturou, pro vložení cest do frameworku by se velmi hodila XSLT[5].

Na vstupu této transformace je potřeba mít datový XML soubor, který má obsahovat v sobě posloupnost akcí pro každého Ozobota a XSL[6] šablonu, která určuje pravidla pro transformace. Kvůli tomu, že program pro Ozoboty zůstává stejný a mění se jenom výsledné cesty pro různé instance problému

[4]Klasicky MAPF problem je popsan v 2.1.2.
[5]eXtensible Stylesheet Language Transformation
[6]eXtensible Stylesheet Language

Obrázek 3.11: Přiklad definici problemu pro MAPF algoritmus

```
/*Rozmery mřižkového grafu*/
7 7
/*Pocet agentu*/
6
/*seznam souradnic startu a cile techto 6 agentu*/
3 2 5 2
4 0 1 3
5 1 1 6
1 5 3 2
4 2 1 5
0 1 6 3
/*Počet překažek*/
1
/*souřadnice překažky*/
3 3
/*Počet zablokovanych hran*/
1
/*Souřadnice dvou vrcholů mezi kterými je zablokovana hrana.*/
1 3 2 3
```

Obrázek 3.12: Datový XML, obsahující v sobe vysledné plany všech agentů

```
<robots>
    <robot number = "0">
        <action>...</action>
        ...
    </robot>
    <robot number = "1">
        ...
    </robot>
    ...
</robots>
```

MAPF, vstupem do transformace bude datový XML soubor, obsahující seznam akce pro každého Ozobota.

Potřebná struktura datového XML souboru je zobrazená na obrázku 3.12. Element `robots` obsahuje v sobě každého robota pro danou mapu. Atribut `number` u elementa `robot` určuje pozici tohoto robota. Element `action` odpovídá akci, kterou musí provést Ozobot a může nabývat hodnot 0 až 3, kde:

0 – Pohyb po čáře dopředu do vedlejšího vrcholu,

1 – Otáčení o 90 stupňů doleva,

Obrázek 3.13: Definice problemu pro příklad cesta „res/simple_example.txt“

```
2 2
2
0 0 1 1
1 1 0 0
0
0
```

2 – Otáčení o 90 stupňů doprava,

3 – Čekání.

Jelikož výstupem MAPF řešiče je seznam souřadnic pro každého Ozobota, je také potřeba v tomto modulu převést tyto seznamy do potřebného datového XML formátu. Pro převod seznamu výsledných plánů agentů, byla vytvořena metoda `Document generateDataXml (List<List<DirectionCoord>)`.

Xsl šablona by měla obsahovat v sobě části programu ze sekce 3.4.1 a 3.4.2, a další pomocné funkce v XML formátu. Po provedení XSLT by měl vzniknout kompletní a fungující program ve formátu XML, který lze následně importovat do webového editoru.

Získat části programu popsané v minulém odstavci ve speciálním formátu lze jednoduchým způsobem – je nutné pro to jednou vytvořit program ve vizuálním editoru a stáhnout ve formátu XML. Následně při vytváření XSL šablony jako kostru stačí použít tuto XML reprezentaci programu a pomocí XSLT na nutné místo vložit cesty Ozobota z datového XML. Také je využito možnosti XSLT procesoru předat parametry do některých bloků, jako např. rychlost pohybu, délku čekání a rychlost otáčení. Tímto způsobem se získá framework, ve kterém lze doplnit cesty Ozobota a potřebné parametry pro Ozoboty.

3.4.5 Návod k použití frameworku

V této kapitole na jednoduchém příkladu chce autor ukázat, jak spočítat výsledné plány agentů pro předem definovaný problém, a následně je vyplnit pomocí frameworku. Definice problému pro příklad je na obrázku 3.13. Mapa představuje čtverec, počáteční pozice Ozobotů jsou levý horní vrchol a pravý dolní vrchol. Ozoboti mají za cíl se vyměnit mezi sebou.

Soubor s definicí problému (komentáře v souboru nejsou povolené) je potřeba přidat do složky `res/`. Soubor musí byt zakódován do UTF-8[7].

Dále je potřeba nakonfigurovat MAPF řešič pomocí souboru `res/config.properties`. V tomto souboru lze zadat název souboru s mapou pro problém MAPF, název algoritmu, který bude použit pro tento MAPF

[7]Unicode Transformation Format, 8-bit

Obrázek 3.14: Konfigurační soubor pro příklad, cesta „res/config.props“

```
#map and algorithm
mapName=race.txt
#Possible variants are: CBS, HCA
mapfAlgorithm=CBS
#ozobot parameters
straightMoveSpeed=35
waitAfterMove=5
rotateSpeed=15
waitAfterRotate=85
waitActionSeconds=1
#milliseconds=waitActionMilliSeconds * 10
waitActionMilliSeconds=86
```

problém a potřebné parametry pro Ozoboty. Pro tento příklad konfigurační soubor bude vypadat následujícím způsobem 3.14.

Pro spuštění řešiče s následujícím vytvořením programu pro Ozoboty z výsledných plánů agentů je potřeba spustit main metodu ve třídě `PathCalculcatorMain.java`. Po doběhnutí programu vznikne soubor s příponou `.ozocode`, který bude uložen do složky s programem. Obsahem tohoto souboru je reprezentace hotového programu pro Ozoboty v jazyce XML. Tento soubor lze následně nahrát přes webový editor ozoblockly.com do svého účtu pomocí vyplnění následující posloupnosti akcí: „Programs“ → „Open program“ → „Open from file“ → „Save as“.

Nakonec je potřeba zapnout Ozoboty, připojit je přes aplikaci Ozobot Evo do nějakého mobilního zařízení, otevřít v aplikaci záložku „Programs and tricks“ a spustit potřebný program. Až program začne běžet na Ozobotech, Ozoboti se rozsvítí žlutou barvou, což odpovídá první pozici z 3.4.2. V tento okamžik má uživatel čas (výchozí nastavení je 30 sekund), aby zadal pozici všech Ozobotů a umístil je na odpovídající počáteční pozici. Po skončení doby konfigurace Ozoboti začnou vyplňovat plány odpovídající jejich pozici.

Program lze také spustit pomoci souboru `program.jar`, který představuje zkompilovaný program. K tomu je potřeba mít nainstalovanou Java 8. Spustit program následně lze pomoci příkazu `java -jar program.jar`. Při otevření souboru `program.jar` pomoci kompresního programu[8] lze také editovat konfigurační soubor a definice problemu MAPF.

3.5 Experimenty nad frameworkem

Pro ukázku a otestování frameworku je vytvořeno pět experimentů. Experimentů se může účastnit různý počet Ozobotů, ale nejvýše 6 různých Ozobotů.

[8] např. WinRar

Pro generaci výsledných plánů agentů je použita vlastní implementace optimálního algoritmu – CBS a neoptimálního algoritmu HCA*.

S využitím vzorců z 3.3.3 by měly být parametry Ozobotů následující:

- rychlost pohybu dopředu – 35 mm/s
- doba trvaní každé akci Ozobota – $\frac{65}{35} = 1,86s$,
- rychlost otáčení – $\frac{18,06}{1,86} = 9,73mm/s$. Jelikož minimální rychlost otáčení smí nabývat pouze hodnot 15 až 85, potom rychlost otáčení je nastavena na 15 mm/sec. V tomto případě je nutné počkat $1,86 - \frac{18,06}{15} = 650ms$ po otáčení,
- čekaní – 1,86 s.

Dále je potřeba odkalibrovat každou akci, aby trvala přesně 1,86 s. Pro kalibraci byla každá akce natočena pomocí snímacího zařízení a následně bylo přidáno zdržení pro akci, která trvalá méně než 1,86 s. Z kalibrace byly získány následující parametry:

- rychlost pohybu dopředu – 35 mm/s s následným čekáním 50 ms,
- rychlost otáčení – 15 mm/s s následným čekáním 850 ms,
- čekaní – 1,86 s.

Tyto hodnoty parametrů byly použity v každém experimentu.

Pro experimenty 3.5.1, 3.5.2, 3.5.3, 3.5.4 byla provedena 2 měření (z kapacitních důvodů). V každém z těchto experimentů byly výsledky z měření seskupeny do tří tabulek: každá tabulka obsahuje výsledky z obou měření: číslo před lomítkem patří 1. měření, číslo za lomítkem patří 2. měření.

První tabulka obsahuje hodnotu uplynulého času od začátku vyplnění plánů na Ozobotech do okamžiku, kdy Ozobot n (sloupec tabulky) vykonal akci i z jeho plánu (řádek tabulky). Poslední sloupec této tabulky je referenční sloupec, který obsahuje předpokládaný odhad uplynulého času po provedení akci i libovolným Ozobotem. Pro všechny experimenty čas trvaní jedné akce činí 1,86 s, proto každý záznam tohoto sloupce lze spočítat jako $i \times 1,86s$.

Druhá tabulka je spočítaná s využitím první tabulky. Každý záznam v sobe obsahuje rozdíl uplynulého času po vyplnění Ozobotem n akci i a referenčního času po vyplnění akci i.

Každý záznam třetí tabulky v sobe obsahuje rozdíl doby vyplnění akce i Ozobotem n a referenční doby vyplnění jedné akci – 1,86 s.

Oproti předchozim experimentům pro experiment 3.5.5 bylo provedeno 10 měření, protože na této mapě Ozoboty vyplňují dělši vysledné plány agentů a také se dochází k víc kooperaci mezi nimi. Tabulky stejného charakteru jako v minulých experimentech jsou umístěny z kapacitních důvodů do souboru Excel v příloze. Výsledky těchto tabulek jsou zagregovány do tří tabulek.

Každý záznam první tabulky obsahuje průměr uplynulého času od začátku vyplnění plánů na Ozobotech do okamžiku, kdy Ozobot n vykonal akci i z jeho plánu mezi těmito 10 meření.

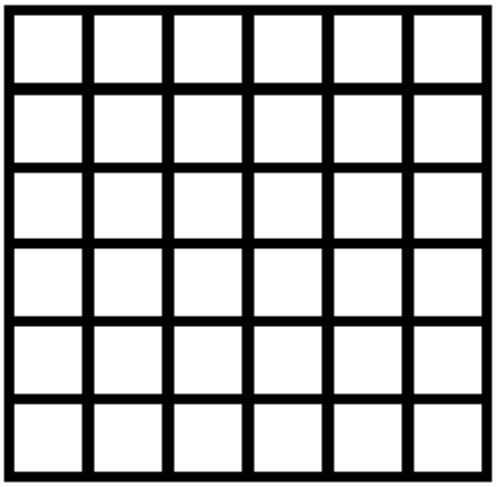
Druha tabulka obsahuje výběrový průměr rozdílů doby vyplnění jedné akce i Ozobotem n od referenční hodnoty 1,86 mezi 10 měřeními.

Jelikož rozdíly doby trvaní akcí i Ozobotem n od referenční hodnoty v těchto 10 meřeních jsou mezi sebou nezávislé, lze pro každý výběrový průměr z 2. tabulky sestavit interval spolehlivosti, což je provedeno ve 3. tabulce.

V každém měření pro každou akci Ozobotu (otáčení, pohyb dopředu, čekání) je také spočítán rozdíl průměrné doby trvaní vybrané akce od referenční doby vyplnění jedné akce – 1,86 s. Podle toho se dá poznat, jaké akce nejvíce způsobují odchylku od vyplnění výsledného plánu agenta v každém měření.

3.5.1 Experiment 1 – Obdélnik

Obrazek: 3.15



Obrázek 3.15: Mapa 1

Počet účastníků: 6

Název souboru s definici problemu: `rectangle.txt`

Popis: Mapa ve formě obdélníku reprezentuje mřížkový graf se všemi hranami, úplně bez překážek. Startovní a cílové pozice Ozobotů jsou vybrané náhodně.

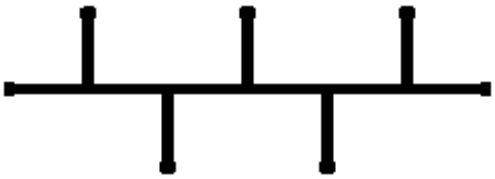
Tabulky s vysledky experimentu: 3.2, 3.3, 3.4.

Rozdil průmerné doby trvaní akci a referenční doby trvaní akci:

1. Pohyb dopředu: -0,005s / 0,0008s
2. Otačení: 0,008s / 0,011s
3. Čekaní: 0,017s / -0,017

3.5.2 Experiment 2 – Tunel

Obrazek: 3.16



Obrázek 3.16: Mapa 2

Počet účastníků: 6

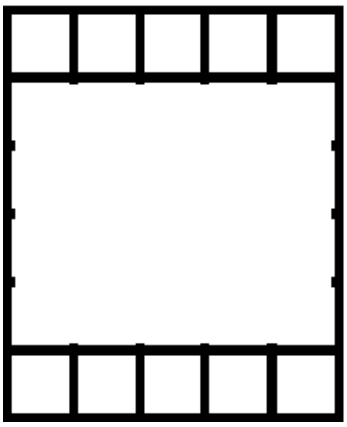
Název souboru s definici problemu: `tunnel.txt`

Popis: Jeden z Ozobotů má za cíl projet horizontální čarou zleva doprava, na které se nachází 5 jiných Ozobotů. Těchto 5 Ozobotů musí propustit Ozobota, který jede horizontální čarou a musí se vrátit zpátky na svoje místo.

Tabulky s vysledky experimentu: 3.5, 3.6, 3.7.

Rozdil průmerné doby trvaní akci a referenční doby trvaní akci:

1. Pohyb dopředu: -0,013s / -0,005s
2. Otačení: 0,012s / 0,007s
3. Čekaní: -0,0005s / 0,034s



Obrázek 3.17: Mapa 3

3.5.3 Experiment 3 – Závod

Obrazek: 3.17

Počet účastníků: 6

Název souboru s definici problemu: `race.txt`

Popis: Ozoboty mají za cíl přemístit se z horní části mapy na dolní a umístit se tam ve stejném pořadí jako nahoře.

Tabulky s vysledky experimentu: 3.8, 3.9, 3.10.

Rozdil průmerné doby trvaní akci a referenční doby trvaní akci:

1. Pohyb dopředu: -0,005 / -0,007
2. Otačení: -0,003s / 0,007s
3. Čekaní: 0,005s / 0,017s

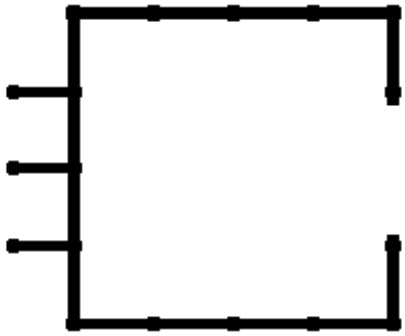
3.5.4 Experiment 4 – Kruhový tunel

Obrazek: 3.18

Počet účastníků: 3

Název souboru s definici problemu: `roundtunnel.txt`

Popis: Ozobot v pravém horním rohu se musí vyměnit s Ozobotem v pravém dolním rohu. Ozobot ve středu mapy se chová jako brána, která musí propustit tyto Ozoboty v potřebný okamžik, a na konci se vrátit na svoje počáteční místo.



Obrázek 3.18: Mapa 4

Tabulky s vysledky experimentu: 3.11, 3.12, 3.13

Rozdil průmerné doby trvaní akci a referenční doby trvaní akci:

1. Pohyb dopředu: 0,003s / 0,008s
2. Otačení: -0,011s / -0,011s
3. Čekaní: 0s / 0s

3.5.5 Experiment 5 – Želva

Obrazek: 3.19

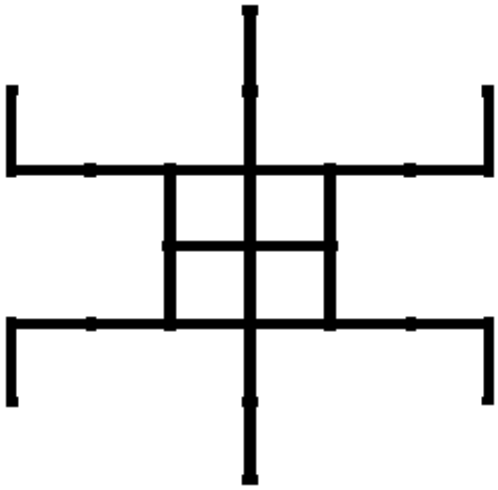
Počet účastníků: 6

Název souboru s definici problemu: `turtle.txt`

Popis: Na této mapě ve formě želvy Ozoboty v určitých dvojicích se musí vyměnit mezi sebou. Kvůli omezenému prostoru v centru mapy vzniká mezi Ozoboty velká kooperace.

Tabulky s vysledky experimentu: 3.14, 3.15, 3.16

Rozdil průmerné a referenční doby trvaní akcí pro každe meření: Tabulka 3.1



Obrázek 3.19: Mapa 5

Tabulka 3.1: rozdil průmerné a referenční doby trvaní akcí pro každe z 10 meření

	meření 1	meření 2	meření 3	meření 4	meření 5	meření 6	meření 7	meření 8	meření 9	meření 10	průmer
Pohyb dopředu	0,01	0,007	0,003	0,015	0,004	0,01	0,002	0,004	0,01	0,009	0,0074
Otačení	0,007	-0,008	-0,001	-0,003	-0,011	-0,006	-0,003	0,001	-0,018	-0,009	-0,0051
Čekaní	0	0	0,017	0	0	0	0	0	0	0	0,0017

3.5.6 Shrnutí výsledků experimentu

V rámci experimentů nedošlo k žádnému konfliktu mezi Ozoboty. Z druhé tabulky pro experimenty 3.5.1, 3.5.2, 3.5.3, 3.5.4 a také z tabulek v příloze pro experiment 3.5.5 je vidět, že odchylka času od referenčního času leží v intervalu (-0,342, 0,408), což znamená, že celou dobu experimentů se Ozoboti neodchylovali od předpokládaného časového plánu více než na *408 ms*.

Z třetí tabulky pro experimenty 3.5.1, 3.5.2, 3.5.3, 3.5.4 a také z tabulek v příloze pro experiment 3.5.5 je vidět, že rozdíl doby trvaní libovolné akce od referenčního času 1,86 leží v intervalu (-0,154, 0,256), takže maximální odchylka doby trvání akcí od referenční činí *256 ms* napříč všemi experimenty.

Průměrná doba trvání pro následující akce napříč všemi měřeními ze všech experimentů leží v intervalu:

1. Pohyb dopředu (-0,013, 0,0101)

2. Otáčení (-0,0177, 0,0137)

3. Čekaní (-0,017, 0,034)

Příčinou těchto odchylek může být:

Hardware Ozobotů ozobot nemůže pokaždé vyplňovat stejnou akci ideálně. Například slabý náboj baterie může způsobit slabší výkon. Také vlastnosti hardwaru mohou být odlišné u různých Ozobotů.

Mapá pro Ozoboty některé hrany mapy mohou byt kratší nebo delší. Jelikož v této práci jsou nakresleny mapy ručně, pak vliv tohoto faktoru je silnější, než jestli kdyby byly mapy vytištěny.

Počateční umístění Ozobotů nepřesné umístění Ozobotů na počáteční pozici může způsobit odchylku pro akci „pohyb dopředu".

Kalibrace čím lépe jsou odkalibrované parametry Ozobotů, tím budou menší odchylky.

Zaznamové zažizení experimenty byly natáčeny na mobilní kameru. Použití kamery s vyšší snímkovací frekvencí by pomohlo změřit experimenty s větší přesností.

Pro experiment 3.5.5 jsou dále okomentovány výsledky z tabulky 3.16. Některé z intervalů spolehlivosti v této tabulce nemají obsaženu hodnotu 0. Bylo zjištěno, že všechny takové intervaly patřily k akci „pohyb dopředu". Pro Ozoboty 4,5,6 i jejich první akci a také Ozoboty 1,2,3 i jejich třetí akci jsou odchylky způsobené nepřesným umístěním Ozobotů na počáteční pozici. Ostatní odchylky jsou způsobeny především nedostatečnou kalibrací a nepřesnými hranami nakreslené mapy.

3.5.7 Tabulky s výsledky experimentů

Tabulka 3.2: experiment 1 – uplynuly čas od začátku spuštění programu na Ozobotech do okamžiku, kdy Ozobot n vykonal akci i z jeho plánu

	1 – žlutý	2 – purpurový	3 – světle modrý	4 – modrý	5 – oranžový	6 – červený	reference
0	0 / 0	0,03 / 0,03	0,03 / 0,07	0 / 0,03	0 / 0,03	0,03 / 0,03	0
1	1,91 / 1,91	1,88 / 1,88	1,91 / 1,98	1,88 / 2,01	1,88 / 1,95	1,98 / 1,84	1,86
2	3,79 / 3,75	3,75 / 3,75	3,79 / 3,82	3,79 / 3,89	3,75 / 3,79	3,75 / 3,69	3,72
3	5,6 / 5,63	5,63 / 5,63	5,53 / 5,53	5,67 / 5,8	5,49 / 5,53	5,63 / 5,56	5,58
4	7,44 / 7,47	7,47 / 7,47	7,41 / 7,41	7,51 / 7,71	7,37 / 7,41	7,47 / 7,37	7,44
5		9,32 / 9,35	9,32 / 9,28	9,35 / 9,52	9,25 / 9,32	9,32 / 9,25	9,3
6		11,13 / 11,19	11,16 / 11,13	11,19 / 11,36	11,09 / 11,19	11,26 / 11,13	11,16
7		12,97 / 13,1	13,04 / 13		12,93 / 13,04	13,14 / 12,97	13,02
8		14,81 / 14,98	14,88 / 14,85		14,78 / 14,91	14,98 / 14,85	14,88
9			16,72 / 16,69		16,65 / 16,79	16,79 / 16,69	16,74
10			18,6 / 18,57			18,67 / 18,57	18,6
11			20,48 / 20,41				20,46
12			22,39 / 22,32				22,32

Tabulka 3.3: experiment 1 – rozdíl uplynulého času po vyplnění Ozobotem n akci i a referenčního času po vyplnění akci i

	1 – žlutý	2 – purpurový	3 – světle modrý	4 – modrý	5 – oranžový	6 – červený
0	0 / 0	0,03 / 0,03	0,03 / 0,07	0 / 0,03	0 / 0,03	0,03 / 0,03
1	0,05 / 0,05	0,02 / 0,02	0,05 / 0,12	0,02 / 0,15	0,02 / 0,09	0,12 / -0,02
2	0,07 / 0,03	0,03 / 0,03	0,07 / 0,1	0,07 / 0,17	0,03 / 0,07	0,03 / -0,04
3	0,02 / 0,05	0,05 / 0,05	-0,05 / -0,05	0,09 / 0,22	-0,09 / -0,05	0,05 / -0,02
4	0 / 0,03	0,03 / 0,03	-0,04 / -0,04	0,07 / 0,27	-0,07 / -0,04	0,03 / -0,07
5		0,02 / 0,05	0,02 / -0,02	0,05 / 0,22	-0,05 / 0,02	0,02 / -0,05
6		-0,04 / 0,03	0 / -0,04	0,03 / 0,2	-0,07 / 0,03	0,1 / -0,04
7		-0,05 / 0,08	0,02 / -0,02		-0,09 / 0,02	0,12 / -0,05
8		-0,07 / 0,1	0 / -0,03		-0,1 / 0,03	0,1 / -0,03
9			-0,02 / -0,05		-0,09 / 0,05	0,05 / -0,05
10			0 / -0,04			0,07 / -0,04
11			0,02 / -0,05			
12			0,07 / 0			

Tabulka 3.4: experiment 1 – rozdíl doby vyplnění jedne akce i Ozobotem n a referenční doby vyplnění jedné akce – 1,86

	1 – žlutý	2 – purpurový	3 – světle modrý	4 – modrý	5 – oranžový	6 – červený
0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
1	0,05 / 0,05	-0,02 / -0,02	0,02 / 0,05	0,02 / 0,12	0,02 / 0,05	0,09 / -0,05
2	0,02 / -0,02	0,02 / 0,02	0,02 / -0,02	0,05 / 0,02	0,02 / -0,02	-0,08 / -0,02
3	-0,05 / 0,02	0,02 / 0,02	-0,12 / -0,15	0,02 / 0,05	-0,12 / -0,12	0,02 / 0,02
4	-0,02 / -0,02	-0,02 / -0,02	0,02 / 0,02	-0,02 / 0,05	0,02 / 0,02	-0,02 / -0,05
5		-0,02 / 0,02	0,05 / 0,02	-0,02 / -0,05	0,02 / 0,05	-0,02 / 0,02
6		-0,05 / -0,02	-0,02 / -0,02	-0,02 / -0,02	-0,02 / 0,02	0,09 / 0,02
7		-0,02 / 0,05	0,02 / 0,02		-0,02 / -0,02	0,02 / -0,02
8		-0,02 / 0,02	-0,02 / -0,02		-0,02 / 0,02	-0,02 / 0,02
9			-0,02 / -0,02		0,02 / 0,02	-0,05 / -0,02
10			0,02 / 0,02			0,02 / 0,02
11			0,02 / -0,02			
12			0,05 / 0,05			

Tabulka 3.5: experiment 2 – uplynuly čas od začátku spuštění programu na Ozobotech do okamžiku, kdy Ozobot n vykonal akci i z jeho plánu

	1 – žlutý	2 – purpurový	3 – světle modrý	4 – modrý	5 – oranžový	6 – červený	ref
0	0 / 0,03	0 / 0,03	0 / 0	0 / 0	0 / 0	0 / 0	0
1	1,91 / 1,89	1,98 / 2,08	1,84 / 1,81	1,88 / 1,84	1,84 / 1,84	1,88 / 1,89	1,86
2	3,79 / 3,75	3,86 / 3,96	3,69 / 3,72	3,86 / 3,89	3,69 / 3,72	3,86 / 3,86	3,72
3	5,7 / 5,6	5,73 / 5,87	5,39 / 5,43	5,7 / 5,73	5,39 / 5,43	5,7 / 5,8	5,58
4	7,51 / 7,44	7,58 / 7,78	7,3 / 7,3	7,58 / 7,58	7,3 / 7,24	7,51 / 7,64	7,44
5	9,35 / 9,25		9,18 / 9,21	9,45 / 9,39	9,15 / 9,04	9,38 / 9,49	9,3
6	11,23 / 11,16		10,99 / 11,02		10,95 / 10,89	11,19 / 11,36	11,16
7	13,1 / 13,04					13,04 / 13,17	13,02

Tabulka 3.6: experiment 2 – rozdíl uplynulého času po vyplnění Ozobotem n akci i a referenčního času po vyplnění akci i

	1 – žlutý	2 – purpurový	3 – světle modrý	4 – modrý	5 – oranžový	6 – červený
0	0 / 0,03	0 / 0,03	0 / 0	0 / 0	0 / 0	0 / 0
1	0,05 / 0,03	0,12 / 0,22	-0,02 / -0,05	0,02 / -0,02	-0,02 / -0,02	0,02 / 0,03
2	0,07 / 0,03	0,14 / 0,24	-0,04 / 0	0,14 / 0,17	-0,04 / 0	0,14 / 0,14
3	0,12 / 0,02	0,15 / 0,29	-0,19 / -0,15	0,12 / 0,15	-0,19 / -0,15	0,12 / 0,22
4	0,07 / 0	0,14 / 0,34	-0,14 / -0,14	0,14 / 0,14	-0,14 / -0,21	0,07 / 0,2
5	0,05 / -0,05		-0,12 / -0,09	0,15 / 0,08	-0,16 / -0,26	0,08 / 0,19
6	0,07 / 0		-0,17 / -0,14		-0,21 / -0,27	0,03 / 0,2
7	0,08 / 0,02					0,02 / 0,15

Tabulka 3.7: experiment 2 – rozdíl doby vyplnění jedne akce i Ozobotem n a referenční doby vyplnění jedné akce – 1,86

	1 – žlutý	2 – purpurový	3 – světle modrý	4 – modrý	5 – oranžový	6 – červený
0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
1	0,05 / -0,01	0,12 / 0,19	-0,02 / -0,05	0,02 / -0,02	-0,02 / -0,02	0,02 / 0,03
2	0,02 / 0,01	0,02 / 0,02	-0,02 / 0,05	0,12 / 0,19	-0,02 / 0,02	0,12 / 0,11
3	0,05 / -0,02	0,02 / 0,05	-0,15 / -0,15	-0,02 / -0,02	-0,15 / -0,15	-0,02 / 0,09
4	-0,05 / -0,02	-0,02 / 0,05	0,05 / 0,02	0,02 / -0,02	0,05 / -0,05	-0,05 / -0,02
5	-0,02 / -0,05		0,02 / 0,05	0,02 / -0,05	-0,02 / -0,05	0,02 / -0,02
6	0,02 / 0,05		-0,05 / -0,05		-0,05 / -0,02	-0,05 / 0,02
7	0,02 / 0,02					-0,02 / -0,05

Tabulka 3.8: experiment 3 – uplynuly čas od začátku spuštění programu na Ozobotech do okamžiku, kdy Ozobot n vykonal akci i z jeho plánu

	1 – žlutý	2 – purpurový	3 – světle modrý	4 – modrý	5 – oranžový	6 – červený	ref
0	0 / 0	0 / 0	0,07 / 0	0 / 0	0 / 0	0,03 / 0	0
1	1,84 / 1,81	1,84 / 1,78	1,88 / 1,84	1,88 / 1,81	1,88 / 1,88	1,88 / 1,95	1,86
2	3,69 / 3,65	3,69 / 3,65	3,75 / 3,72	3,65 / 3,72	3,75 / 3,75	3,69 / 3,82	3,72
3	5,46 / 5,36	5,6 / 5,53	5,6 / 5,56	5,39 / 5,46	5,63 / 5,67	5,39 / 5,56	5,58
4	7,3 / 7,24	7,44 / 7,37	7,47 / 7,44	7,24 / 7,34	7,47 / 7,54	7,27 / 7,41	7,44
5	9,18 / 9,11	9,28 / 9,22	9,35 / 9,32	9,08 / 9,18	9,28 / 9,35	9,18 / 9,28	9,3
6	11,02 / 10,96	11,13 / 11,09	11,23 / 11,19	11,02 / 11,09	11,13 / 11,23	11,02 / 11,16	11,16
7	12,9 / 12,93	13,04 / 12,97	13,11 / 13,04	12,9 / 12,93	13 / 13,07	12,93 / 13,07	13,02
8	14,78 / 14,78	14,88 / 14,81	14,95 / 14,91	14,74 / 14,78	14,88 / 14,95	14,78 / 14,88	14,88
9		16,76 / 16,76	16,83 / 16,79	16,62 / 16,69	16,76 / 16,83		16,74
10		18,57 / 18,6	18,7 / 18,63	18,46 / 18,53	18,63 / 18,77		18,6
11		20,44 / 20,48	20,55 / 20,48	20,41 / 20,44	20,41 / 20,55		20,46
12		22,32 / 22,35	22,42 / 22,35	22,29 / 22,35	22,32 / 22,42		22,32
13		24,23 / 24,2	24,3 / 24,2	24,06 / 24,13	24,16 / 24,23		24,18
14			26,21 / 26,14	25,94 / 26,01			26,04
15			28,09 / 27,95	27,81 / 27,88			27,9
16			29,9 / 29,76	29,62 / 29,69			29,76

Tabulka 3.9: experiment 3 – rozdíl uplynulého času po vyplnění Ozobotem n akci i a referenčního času po vyplnění akci i

	1 – žlutý	2 – purpurový	3 – světle modrý	4 – modrý	5 – oranžový	6 – červený
0	0 / 0	0 / 0	0,07 / 0	0 / 0	0 / 0	0,03 / 0
1	-0,02 / -0,05	-0,02 / -0,09	0,02 / -0,02	0,02 / -0,05	0,02 / 0,02	0,02 / 0,09
2	-0,03 / -0,07	-0,03 / -0,07	0,03 / 0	-0,07 / 0	0,03 / 0,03	-0,03 / 0,1
3	-0,12 / -0,22	0,02 / -0,05	0,02 / -0,02	-0,19 / -0,12	0,05 / 0,09	-0,19 / -0,02
4	-0,14 / -0,21	0 / -0,07	0,03 / 0	-0,21 / -0,1	0,03 / 0,1	-0,17 / -0,03
5	-0,12 / -0,19	-0,02 / -0,09	0,05 / 0,02	-0,22 / -0,12	-0,02 / 0,05	-0,12 / -0,02
6	-0,14 / -0,2	-0,03 / -0,07	0,07 / 0,03	-0,14 / -0,07	-0,03 / 0,07	-0,14 / 0
7	-0,12 / -0,09	0,02 / -0,05	0,09 / 0,02	-0,12 / -0,09	-0,02 / 0,05	-0,09 / 0,05
8	-0,1 / -0,1	0 / -0,07	0,07 / 0,03	-0,14 / -0,1	0 / 0,07	-0,1 / 0
9		0,02 / 0,02	0,09 / 0,05	-0,12 / -0,05	0,02 / 0,09	
10		-0,04 / 0	0,1 / 0,03	-0,14 / -0,07	0,03 / 0,17	
11		-0,02 / 0,02	0,08 / 0,02	-0,05 / -0,02	-0,05 / 0,09	
12		0 / 0,03	0,1 / 0,03	-0,04 / 0,03	0 / 0,1	
13		0,05 / 0,02	0,12 / 0,02	-0,12 / -0,05	-0,02 / 0,05	
14			0,17 / 0,1	-0,1 / -0,04		
15			0,19 / 0,05	-0,09 / -0,02		
16			0,13 / 0	-0,14 / -0,07		

Tabulka 3.10: experiment 3 – rozdíl doby vyplnění jedne akce i Ozobotem n a referenční doby vyplnění jedné akce – 1,86

	1 – žlutý	2 – purpurový	3 – světle modrý	4 – modrý	5 – oranžový	6 – červený
0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
1	-0,02 / -0,05	-0,02 / -0,09	-0,05 / -0,02	0,02 / -0,05	0,02 / 0,02	-0,02 / 0,09
2	-0,02 / -0,02	-0,02 / 0,02	0,02 / 0,02	-0,09 / 0,05	0,02 / 0,02	-0,05 / 0,02
3	-0,08 / -0,15	0,05 / 0,02	-0,02 / -0,02	-0,12 / -0,12	0,02 / 0,05	-0,15 / -0,12
4	-0,02 / 0,02	-0,02 / -0,02	0,02 / 0,02	-0,02 / 0,02	-0,02 / 0,02	0,02 / -0,02
5	0,02 / 0,02	-0,02 / -0,02	0,02 / 0,02	-0,02 / -0,02	-0,05 / -0,05	0,05 / 0,02
6	-0,02 / -0,02	-0,02 / 0,02	0,02 / 0,02	0,09 / 0,05	-0,02 / 0,02	-0,02 / 0,02
7	0,02 / 0,12	0,05 / 0,02	0,02 / -0,02	0,02 / -0,02	0,02 / -0,02	0,05 / 0,05
8	0,02 / -0,02	-0,02 / -0,02	-0,02 / 0,02	-0,02 / -0,02	0,02 / 0,02	-0,02 / -0,05
9		0,02 / 0,09	0,02 / 0,02	0,02 / 0,05	0,02 / 0,02	
10		-0,05 / -0,02	0,02 / -0,02	-0,02 / -0,02	0,02 / 0,08	
11		0,02 / 0,02	-0,02 / -0,02	0,09 / 0,05	-0,09 / -0,08	
12		0,02 / 0,02	0,02 / 0,02	0,02 / 0,05	0,05 / 0,02	
13		0,05 / -0,02	0,02 / -0,02	-0,09 / -0,09	-0,02 / -0,05	
14			0,05 / 0,08	0,02 / 0,02		
15			0,02 / -0,05	0,02 / 0,02		
16			-0,05 / -0,05	-0,05 / -0,05		

Tabulka 3.11: experiment 4 – uplynuly čas od začátku spuštění programu na Ozobotech do okamžiku, kdy Ozobot n vykonal akci i z jeho plánu

	1 – žlutý	2 – purpurový	3 – světle modrý	reference
0	0,03 / 0,03	0 / 0,03	0,03 / 0	0
1	2,05 / 2,01	1,84 / 1,88	1,84 / 1,91	1,86
2	3,93 / 3,89	3,62 / 3,69	3,65 / 3,79	3,72
3	5,77 / 5,73	5,32 / 5,39	5,49 / 5,63	5,58
4	7,64 / 7,61	7,24 / 7,17	7,37 / 7,44	7,44
5	9,52 / 9,56	9,08 / 9,08	9,25 / 9,28	9,3
6	11,33 / 11,5	10,96 / 10,96	11,13 / 11,13	11,16
7	13,21 / 13,31	12,8 / 12,8	12,97 / 12,97	13,02
8	15,12 / 15,19	14,67 / 14,61	14,85 / 14,85	14,88
9	16,96 / 17,03	16,59 / 16,52	16,69 / 16,69	16,74
10	18,84 / 18,91	18,43 / 18,39	18,5 / 18,53	18,6
11	20,68 / 20,72	20,34 / 20,31	20,37 / 20,41	20,46
12	22,49 / 22,52	22,18 / 22,43	22,22 / 22,28	22,32
13	24,33 / 24,44	23,99 / 23,96	24,09 / 24,13	24,18
14	26,21 / 26,31	25,83 / 25,83	25,94 / 26	26,04
15	28,11 / 28,15	27,75 / 27,71	27,75 / 27,81	27,9
16	30 / 30,07	29,62 / 29,55		29,76
17	31,87 / 31,91	31,47 / 31,43		31,62
18	33,72 / 33,75	33,34 / 33,31		33,48
19	35,59 / 35,59			35,34
20	37,44 / 37,44			37,2
21	39,28 / 39,42			39,06
22	41,12 / 41,23			40,92

Tabulka 3.12: experiment 4 – rozdíl uplynulého času po vyplnění Ozobotem n akci i a referenčního času po vyplnění akci i

	1 – žlutý	2 – purpurový	3 – světle modrý
0	0,03 / 0,03	0 / 0,03	0,03 / 0
1	0,19 / 0,15	-0,02 / 0,02	-0,02 / 0,05
2	0,21 / 0,17	-0,1 / -0,03	-0,07 / 0,07
3	0,19 / 0,15	-0,26 / -0,19	-0,09 / 0,05
4	0,2 / 0,17	-0,21 / -0,27	-0,07 / 0
5	0,22 / 0,25	-0,22 / -0,22	-0,05 / -0,02
6	0,17 / 0,34	-0,21 / -0,21	-0,04 / -0,04
7	0,19 / 0,29	-0,22 / -0,22	-0,05 / -0,05
8	0,24 / 0,31	-0,21 / -0,27	-0,04 / -0,03
9	0,22 / 0,29	-0,15 / -0,22	-0,05 / -0,05
10	0,24 / 0,31	-0,17 / -0,21	-0,1 / -0,07
11	0,22 / 0,25	-0,12 / -0,16	-0,09 / -0,05
12	0,17 / 0,2	-0,14 / -0,22	-0,1 / -0,04
13	0,15 / 0,25	-0,19 / -0,22	-0,09 / -0,05
14	0,17 / 0,27	-0,21 / -0,21	-0,1 / -0,04
15	0,21 / 0,25	-0,16 / -0,19	-0,16 / -0,09
16	0,24 / 0,31	-0,14 / -0,21	
17	0,25 / 0,29	-0,16 / -0,19	
18	0,24 / 0,27	-0,14 / -0,17	
19	0,25 / 0,25		
20	0,24 / 0,24		
21	0,22 / 0,36		
22	0,2 / 0,31		

Tabulka 3.13: experiment 4 – rozdíl doby vyplnění jedne akce i Ozobotem n a referenční doby vyplnění jedné akce – 1,86

	1 – žlutý	2 – purpurový	3 – světle modrý
0	0 / 0	0 / 0	0 / 0
1	0,15 / 0,12	-0,02 / -0,02	-0,05 / 0,05
2	0,02 / 0,02	-0,09 / -0,05	-0,05 / 0,02
3	-0,02 / -0,02	-0,15 / -0,15	-0,02 / -0,02
4	0,02 / 0,02	0,05 / -0,09	0,02 / -0,05
5	0,02 / 0,08	-0,02 / 0,05	0,02 / -0,02
6	-0,05 / 0,09	0,02 / 0,02	0,02 / -0,02
7	0,02 / -0,05	-0,02 / -0,02	-0,02 / -0,02
8	0,05 / 0,02	0,02 / -0,05	0,02 / 0,02
9	-0,02 / -0,02	0,05 / 0,05	-0,02 / -0,02
10	0,02 / 0,02	-0,02 / 0,02	-0,05 / -0,02
11	-0,02 / -0,05	0,05 / 0,05	0,02 / 0,02
12	-0,05 / -0,05	-0,02 / 0,27	-0,02 / 0,02
13	-0,02 / 0,05	-0,05 / -0,33	0,02 / -0,02
14	0,02 / 0,02	-0,02 / 0,02	-0,02 / 0,02
15	0,04 / -0,02	0,05 / 0,02	-0,05 / -0,05
16	0,03 / 0,05	0,02 / -0,02	
17	0,02 / -0,02	-0,02 / 0,02	
18	-0,02 / -0,02	0,02 / 0,02	
19	0,02 / -0,02		
20	-0,02 / -0,02		
21	-0,02 / 0,12		
22	-0,02 / -0,05		

Tabulka 3.14: experiment 5 – průmer uplynulého času od začátku spuštění programu na Ozobotech do okamžiku, kdy Ozobot n vykonal akci i z jeho plánu mezi těmito 10 meření

	1 - žlutý	2 - purpurový	3 - světle modrý	4 - modrý	5 - oranžový	6 - červený	reference
0	0,0272	0,0238	0,0409	-0,0034	0,0069	0,0103	0
1	1,8973	1,8974	1,87	1,9657	2,0611	2,0066	1,86
2	3,7334	3,7296	3,7095	3,8187	3,9141	3,8632	3,72
3	5,5387	5,4774	5,5143	5,6854	5,7468	5,682	5,58
4	7,3898	7,3543	7,3815	7,5283	7,6082	7,559	7,44
5	9,2482	9,2108	9,2551	9,4052	9,4702	9,4017	9,3
6	11,1253	11,0562	11,1389	11,2583	11,3411	11,265	11,16
7	12,992	12,9339	12,9921	13,1352	13,2037	13,1079	13,02
8	14,8689	14,7937	14,8245	14,9815	15,0703	14,9713	14,88
9	16,6879	16,6434		16,8755	16,9441	16,821	16,74
10	18,5648	18,5135		18,7253	18,7694	18,7046	18,6
11	20,421	20,3837		20,6124	20,626	20,5646	20,46
12	22,264	22,247		22,4756	22,4484	22,404	22,32
13		24,1273				24,2749	24,18
14		26,001				26,1238	26,04
15		27,8882				27,9632	27,9
16		29,7949				29,8607	29,76
17		31,6762					31,62
18		33,5635					33,48

Tabulka 3.15: experiment 5 – vyběrový průmer rozdilu doby vyplnění jedne akce i Ozobotem n od referenční hodnoty 1,86 mezi 10 meření

	1 - žlutý	2 - purpurový	3 - světle modrý	4 - modrý	5 - oranžový	6 - červený
0						
1	0,0101	0,0136	-0,0309	0,1091	0,1942	0,1363
2	-0,0239	-0,0278	-0,0205	-0,007	-0,0139	-0,0034
3	-0,0547	-0,1122	-0,0552	0,0067	-0,0204	-0,0412
4	-0,0089	0,0169	0,0072	-0,0171	0,0014	0,017
5	-0,0016	-0,0035	0,0136	0,0169	0,002	-0,0173
6	0,0171	-0,0146	0,0238	-0,0069	0,0109	0,0033
7	0,0067	0,0177	-0,0068	0,0169	0,0026	-0,0171
8	0,0169	-0,0002	-0,0276	-0,0137	0,0066	0,0034
9	-0,041	-0,0103		0,034	0,0138	-0,0103
10	0,0169	0,0101		-0,0102	-0,0347	0,0236
11	-0,0038	0,0102		0,0271	-0,0034	3,55E-16
12	-0,017	0,0033		0,0032	-0,0376	-0,0206
13		0,0203				0,0109
14		0,0137				-0,0111
15		0,0272				-0,0206
16		0,0467				0,0375
17		0,0213				
18		0,0273				

Tabulka 3.16: experiment 5 – intervaly spoléhlivosti pro hodnoty z tabulky 3.15 na 5% hladině vyznámností

	1 - žlutý	2 - purpurový	3 - světle modrý	4 - modrý	5 - oranžový	6 - červený
0						
1	(-0,01, 0,03)	(-0,01, 0,04)	(-0,06, 0)	(0,08, 0,14)	(0,17, 0,22)	(0,1, 0,17)
2	(-0,04, 0)	(-0,06, 0)	(-0,04, 0)	(-0,02, 0,01)	(-0,04, 0,02)	(-0,03, 0,02)
3	(-0,08, -0,03)	(-0,13, -0,09)	(-0,09, -0,02)	(-0,02, 0,03)	(-0,04, 0)	(-0,06, -0,02)
4	(-0,05, 0,03)	(-0,01, 0,05)	(-0,02, 0,04)	(-0,03, 0)	(-0,01, 0,01)	(0,01, 0,03)
5	(-0,03, 0,02)	(-0,01, 0,01)	(0, 0,03)	(-0,01, 0,04)	(-0,01, 0,01)	(-0,03, 0)
6	(0, 0,04)	(-0,03, 0)	(0,01, 0,03)	(-0,02, 0,01)	(-0,02, 0,05)	(-0,01, 0,01)
7	(0, 0,02)	(-0,01, 0,04)	(-0,02, 0)	(-0,01, 0,04)	(-0,02, 0,02)	(-0,04, 0,01)
8	(0,01, 0,03)	(-0,01, 0,01)		(-0,05, 0,02)	(0, 0,02)	(-0,01, 0,01)
9	(-0,05, -0,03)	(-0,03, 0,01)		(0, 0,07)	(0, 0,03)	(-0,04, 0,02)
10	(0,02, 0,02)	(-0,01, 0,03)		(-0,05, 0,03)	(-0,05, -0,02)	(-0,01, 0,06)
11	(-0,03, 0,02)	(0, 0,02)		(0,02, 0,04)	(-0,02, 0,02)	(-0,02, 0,02)
12	(-0,03, 0)	(-0,03, 0,03)		(-0,01, 0,01)	(-0,05, -0,03)	(-0,03, -0,01)
13		(0,01, 0,03)				(0, 0,02)
14		(-0,01, 0,04)				(-0,02, 0)
15		(-0,01, 0,06)				(-0,05, 0,01)
16		(0, 0,09)				(0,03, 0,05)
17		(-0,01, 0,05)				
18		(0,02, 0,04)				

Závěr

V rámci návrhu a implementaci frameworku bylo zjištěno, že roboti Ozobot jsou sice použitelní pro problematiku MAPF, ale s některými výhradami. Hlavními nedostatky jsou: nemožnost spouštět různý kód na Ozobotech synchronně, malá kapacita baterie Ozobotu a malá dostupná vnitřní paměť Ozobotu. Toto spolu s nutností chránit všechny cesty pro Ozoboty dělá problémy při použití většího počtu agentů pro problematiku MAPF.

Podařilo se mi implementovat funkční framework, který umožňuje vyplnit výsledné plány z algoritmu MAPF na robotech Ozobot. Z výsledků experimentů nad tímto frameworkem se ukázalo, že pomocí frameworku Ozoboti vyplňují výsledné plány z MAPF s malými odchylkami od předpokládaného času vyplnění. Díky tomu se Ozoboti v experimentech nedostali do žádného konfliktu.

Pro budoucí práce bych sdílel několik doporučení:

- zavést interakci mezi Ozoboty s použitím senzorů přiblížení, aby snížilo riziko konfliktu mezi roboty,
- přidat grafické rozhraní pro zadávání problémů MAPF,
- navrhnout lepší protokol pro předání výsledných plánů MAPF do Ozobotu,
- s ohledem na přiciny odchylek z sekce s vysledky experimentů, řešit tyto přičiny (vytisknout mapy, pořidit lepši kameru atd.) a provést víc experimentů nad Ozoboti.

Bibliografie

1. LI, Jiaoyang; TINKA, Andrew; KIESEL, Scott; DURHAM, Joseph W; KUMAR, TK Satish; KOENIG, Sven. Lifelong Multi-Agent Path Finding in Large-Scale Warehouses. In: *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents a Multiagent SystemsRichland, 2020, s. 1898–1900. ISBN 978-1-4503-7518-4.
2. HAGELBÄCK, Johan. Hybrid pathfinding in StarCraft. *IEEE Transactions on Computational Intelligence and AI in games*. 2015, roč. 8, č. 4, s. 319–324. ISSN 1943068X.
3. BICCHI, Antonio; PALLOTTINO, Lucia. On optimal cooperative conflict resolution for air traffic management systems. *IEEE Transactions on Intelligent Transportation Systems*. 2000, roč. 1, č. 4, s. 221–231. ISSN 1558-0016.
4. PETR, Hliněný. *Zaklady Teorie Grafů pro (nejen) informatiky* [online]. 2010. Dostupné také z: `http://www.dlib.org/dlib/july05/lynch/07lynch.html`.
5. STERN, Roni et al. Multi-agent pathfinding: Definitions, variants, and benchmarks. In: *Proceedings of the Twelfth International Symposium on Combinatorial Search*. AAAI Press 2019, 2019, s. 151–159. ISBN 978-1-57735-808-4.
6. SHARON, Guni; STERN, Roni; FELNER, Ariel; STURTEVANT, Nathan R. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*. 2015, roč. 219, s. 40–66. ISSN 0004-3702. Dostupné z DOI: `10.1016/j.artint.2014.11.006`.
7. RUSSELL, Stuart J.; NORVIG, Peter. *Artificial intelligence: a modern approach*. Pearson, 2016. ISBN 9781292153964.

8. FELNER, Ariel; STERN, Roni; SHIMONY, Solomon Eyal; BOYARSKI, Eli; GOLDENBERG, Meir; SHARON, Guni; STURTEVANT, Nathan; WAGNER, Glenn; SURYNEK, Pavel. Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges. In: *Proceedings of the Tenth Annual Symposium on Combinatorial Search*. AAAI Press 2017, 2017, s. 29–37. ISBN 9781577357902.

9. SURYNEK, Pavel. Heuristicke prohledávání stavového prostoru [přednaška]. In: [online]. 2020. Dostupné také z: `https://courses.fit.cvut.cz/BI-ZUM/media/lectures/BI-ZUM_lecture-03_heuristic.pdf`.

10. SILVER, David. Cooperative Pathfinding. *the artificial intelligence for interactive digital entertainment conference*. 2005, roč. 1, s. 117–122. ISSN 2334-0924.

11. BARER, Max; SHARON, Guni; STERN, Roni; FELNER, Ariel. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In: *Proceedings of the Seventh Annual Symposium on Combinatorial Search*. AAAI Press 2014, 2014. ISBN 978-1-57735-676-9.

Příloha **A**

Seznam použitých zkratek

BFS best first search - algoritmus pro vyhledaní v stavovem prostoru

FIFO First in first out

JS Programovaci jazyk JavaScript

MAPF Z anglickeho Multi-agent pathfinding znamena vyhledavani cesty pro nekolik agentu.

SAT Boolean satisfiability problem

XML Extensible markup language

XSL eXtensible Stylesheet Language

XSLT eXtensible Stylesheet Language Transformation

Příloha **B**

Obsah přiloženého CD

- `program.jar` ..... spustitelný soubor
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
- `videos` ..... videa z experimentů
- `calculations` ..... soubory s vysledky experimentů