FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Multi-agentní hledání cest s produkcí a konzumací agentů
Student:	Tomáš Holas
Vedoucí:	doc. RNDr. Pavel Surynek, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Znalostní inženýrství
Katedra:	Katedra aplikované matematiky
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem práce je navrhnout nové nebo modifikovat existující algoritmy pro multi-agentní hledání cest se spojitým časem (MAPF-R), které počítají s možností produkování agentů v určitých místech prostředí (tj. agent se objeví) a konzumací agentů v cílových místech (tj. agent zmizí). Motivací pro tuto úlohu plánování pohybu robotů pro skladovou logistiku v rámci ohraničené části skladu, kde tato hranice tvoří místo produkce a konzumace robotů. Úkoly uchazeče budou následující:

1. Seznámí se s literaturou týkající se řešení problému MAPF-R, zejména s řešiči založenými na splnitelnosti
2. Navrhne a implementuje nový nebo modifikaci existujícího algoritmu pro variantu MAPF-R s produkcí a konzumací agentů
3. Navržený algoritmus otestuje v modelových scénářích s reálnými roboty typu Ozobot podobně, jako je tomu v práci [1], tedy v simulacích s otevřenou smyčkou řízení

[1] Simulation of Centralized Algorithms for Multi-Agent Path Finding on Real Robots, bakalářská práce, FIT ČVUT, 2020.

Elektronicky schválil/a Ing. Karel Klouda, Ph.D. dne 2. prosince 2021 v Praze.

Bakalářská práce

MULTI-AGENTNÍ HLEDÁNÍ CEST S PRODUKCÍ A KONZUMACÍ AGENTŮ

Tomáš Holas

Fakulta informačních technologií
Katedra aplikované matematiky
Vedoucí: doc. RNDr. Pavel Surynek, Ph.D.
10. května 2022

Obsah

Seznam obrázků

Seznam tabulek

Seznam algoritmů

Chtěl bych poděkovat svému vedoucímu bakalářské práce doc. RNDr. Pavlu Surynkovi, Ph.D. za odborné vedení, za pomoc a rady při zpracování této práce.

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, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 10. května 2022 .....

Abstrakt

Obsahem této bakalářské práce je vytvoření řešiče pro ovládání více agentů, který počítá s produkcí a konzumací agentů. Řešič sestaví plán pro spojitý čas a tento plán je následně demonstrován pomocí robotů typu ozobot. Multi-agentní hledání cest je problematické kvůli možným kolizím robotů. Kolize je potřeba vyřešit pro bezproblémový průchod robotů skladištěm, což způsobí zefektivnění celé logistiky skladu.

Klíčová slova mapf, ozobot, multi-agentní hledání cest, skladová logistika, spojitý čas

Abstract

The scope of this bachelor thesis is to create a multi-agent control solver that accounts for production and consumption of agents. The solver builds a plan for continuous time and this plan is then demonstrated using ozobots. Multi-agent pathfinding is problematic due to possible robot collisions. The collisions need to be resolved for the smooth passage of robots through the warehouse, which will make the entire warehouse logistics more efficient.

Keywords mapf, ozobot, multi-agent path finding, warehouse logistics, continuous time

Seznam zkratek

MAPF	Multi-Agent Pathfinding
MAPF-R	Multi-agent Pathfinding with continuous time
CBS	Conflict-based search
CCBS	Continuous-time conflict-based search
RGB	Red, Green, Blue
LED	Light-Emitting Diode
DA	Dijkstrův algoritmus

Úvod

Multi-agentní hledání cest, neboli plánování pohybu více agentů najednou, je v dnešní době velké téma. Spousta skladových jednotek používá k logistické přepravě zboží roboty. Ty je potřeba ovládat buď centralizovaným nebo decentralizovaným způsobem. Ve většině případů se využívá centralizovaný systém, protože roboti pro přesun zboží mohou být mnohem jednodušší, jelikož je řídí centrální počítač.

Motivací pro tuto práci bylo skladiště firmy Amazon. Ve skladu, v místech, kde se zboží vykládá z regálů do přepravek a následně je posláno k zabalení, roboti dovážející produkty tvoří dlouhé fronty a tím zdržují celkovou logistiku skladu. Cílem je tedy odbavit agenty co nejrychleji a tím zefektivnit celou logistiku.

Tato práce se zabývá vyřešením tohoto problému, kdy se agenti objevují na okraji vykládacího místa a zanikají po dovezení zboží. Úkolem práce tedy je naprogramovat řešič, který s touto variantou počítá a vytvoří validní plán pro agenty. Úkolem není ovládat agenty po dovezení zboží, ale jen během jeho přepravy z místa objevení do místa zaniknutí.

Kapitola 1
Teorie v pozadí

Na začátku kapitoly budou popsány úplné základy z teorie grafů, protože se od ní odvíjí celá práce. Úvodem budou vysvětleny základní pojmy z teorie grafů, následovat bude kapitola o multi-agentním hledání cest a na závěr příklady z praxe. Informace do této kapitoly byly čerpány z přednášek předmětu Algoritmy a grafy 1 (BI-AG1), Fakulta informačních technologií, České vysoké učení technické v Praze.

1.1 Základní teorie

1.1.1 Neorientovnaý graf

Neorientovaný graf , v práci dále jen jako **Graf**, je uspořádaná dvojice (V, E), kde

- V je neprázdná konečná množina **vrcholů** (uzlů),
- E je množina **hran**.

Hrana je neuspořádaná dvojice vrcholů {u, v}. Lze si snadno povšimnout, že počet hran je shora omezen hodnotou $\binom{V}{2}$. Pro každé dva vrcholy tam hrana buď je, nebo není. Zdola omezen není. Můžeme tedy mít graf, který má samé vrcholy a žádné hrany.

Množinu vrcholů grafu G budeme značíme jako V(G) a množinu hran jako E(G). Počet vrcholů značíme $|V(G)|$ a počet hran $|E(G)|$.

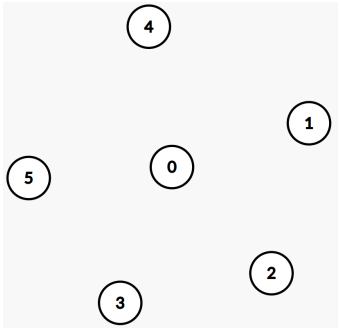
Značení Nechť e = {u, v} je hrana v grafu G. Pak řekneme, že

- vrcholy u a v jsou **koncové vrcholy** hrany e,
- u je **sousedem** v v G (a naopak).

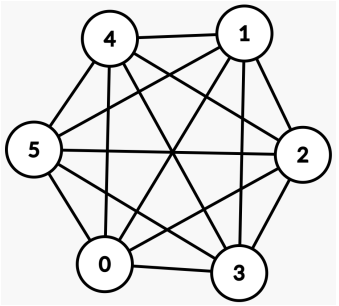
Sousední vrcholy jsou takové vrcholy, mezi kterými vede hrana.

Úplný graf K_n je takový graf, který splňuje následující vlastnosti

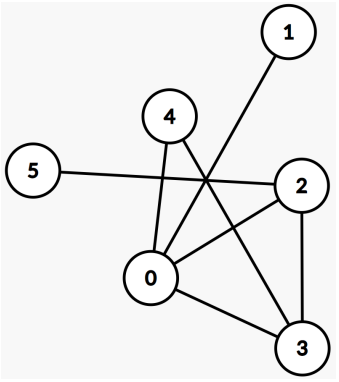
- $n \geq 1$,



(a) minimální graf na pěti vrcholech



(b) maximální graf na pět vrcholech (K_5)



(c) náhodný neorientovaný graf na pěti vrcholech

Obrázek 1.1 Příklady neorientovaných grafů

- K_n je graf (V, $\binom{V}{2}$), kde $|V|$ = n.

Jinak řečeno, jedná se o graf, ve kterém mezi každými dvěma vrcholy vede hrana.

Cesta P_m délky $m \geq 0$ je graf ({0, . . . , m}, {{i, i + 1} | i ∈ {0, . . . , m - 1}}), kde **m je počet hran**.

1.1.2 Orientovaný graf

Orientovaný graf G je uspořádaná dvojice (V, E), kde

- V je neprázdná konečná množina **vrcholů** (nebo také uzlů),
- E je množina **orientovaných hran** (nebo také šipek).

Orientovaná hrana (u, v) ∈ E je **uspořádaná** dvojice různých vrcholů u, v ∈ V . Říkáme, že u je **předchůdce** v a v je **následník** u. Platí tedy E ⊆ V × V.

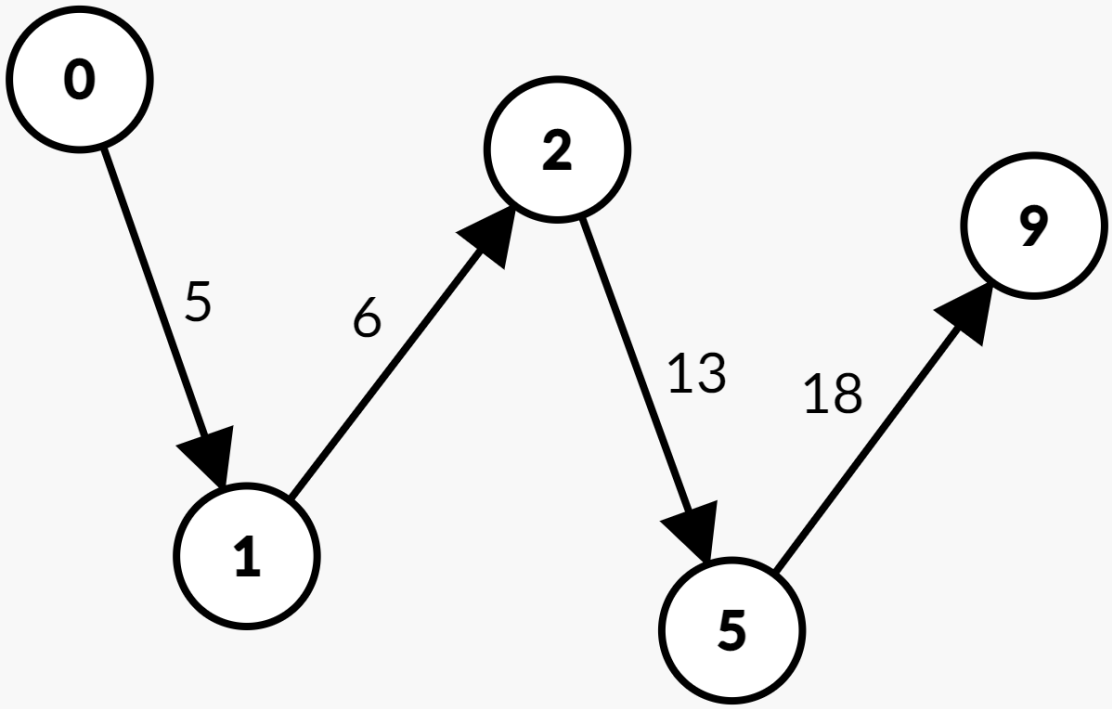
Orientovaná cesta P_m délky $m \geq 0$ je graf ({0, . . . , m}, {(i, i + 1) | i ∈ {0, . . . , m - 1}}), kde **m je počet hran**.

1.1.3 Hledání cest

Hranově ohodnocený graf G = (V, E), je neorientovaný graf takový, že ke každé hraně e ∈ E přiřadíme číselnou váhu w(e), kde w: E → R.

Ohodnocení hrany budeme reprezentovat jako její délku. Tuto délku si lze vyložit mnoha způsoby. Ať už délku mezi dvěma vrcholy, nebo čas, za který tento úsek agent ujede.

vzdálenost d(u, v) je minimum z délek všech uv-cest (cest z u do v) pro libovolné dva vrcholy u, v, v případně $+\infty$, pokud žádná uv-cesta neexistuje.



Obrázek 1.2 Příklad ohodnocené orientované cesty délky 42

Nejkratší uv-cesta je libovolná uv-cesta, jejíž délka je rovna vzdálenosti d(u, v).

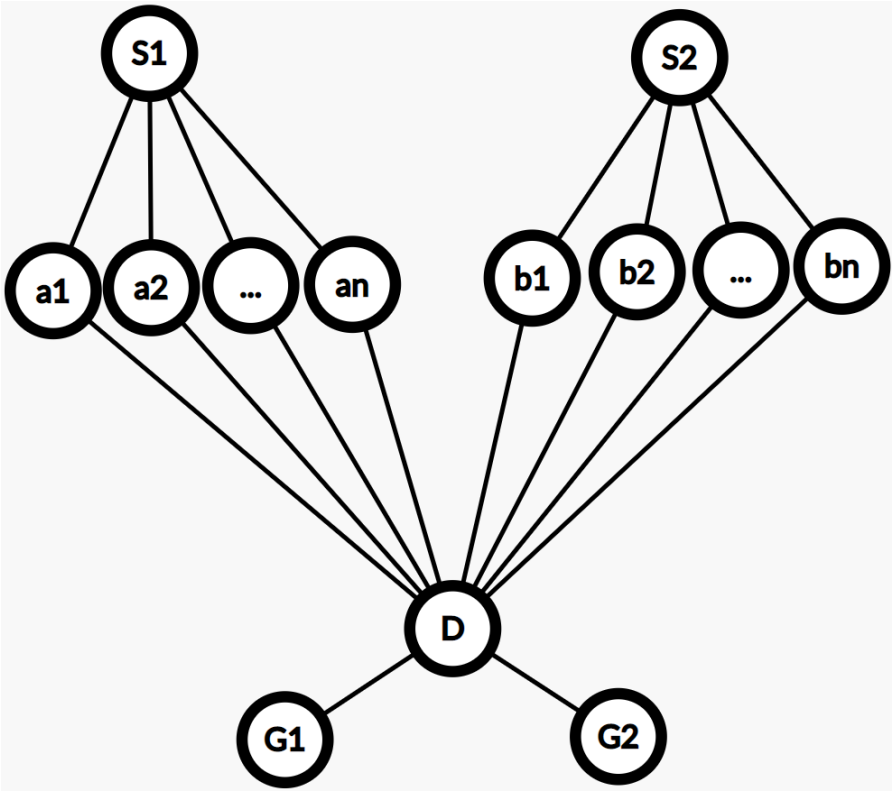
Trojúhelníková nerovnost Pokud jsou délky hran kladné, musí pro ně platit trojúhelníková nerovnost: $d(u, v) \leq d(u, w) + d(w, v)$ pro libovolné u, v, w ∈ V.

1.2 Multi-agentní hledání cest (MAPF)

Cílem multi-agentního hledání cest (MAPF) je přepravit všechny agenty ze svých počátečních pozic na jejich koncové pozice. Problém je reprezentován na grafu G = (V, E), kde jsou umístěni agenti z množiny A = $\{a_1, a_2, ... a_K\}$ na jednotlivých vrcholech. Na každém vrcholu může být pouze jeden agent. Počáteční konfigurace agentů může být zapsána jako α_0 : A → V. Úplně stejně můžeme definovat koncovou konfiguraci jako α_+ : A → V. Cíl MAPF můžeme přepsat na problém převedení počáteční konfigurace na konečnou za pomocí akcí 1.2.1. Veškeré informace k celému MAPF a MAPF-R pochází z těchto zdrojů: [1, 2, 3].

1.2.1 Akce

Každý agent může vykonat akci *pohyb*, nebo akci *čekat*. Při akci pohyb, je agent přesunut na jeden ze sousedních vrcholů a při akci čekat agent zůstane na stejném vrcholu. Obě tyto akce mají stejnou cenu.



Obrázek 1.3 Příklad MAPF problému

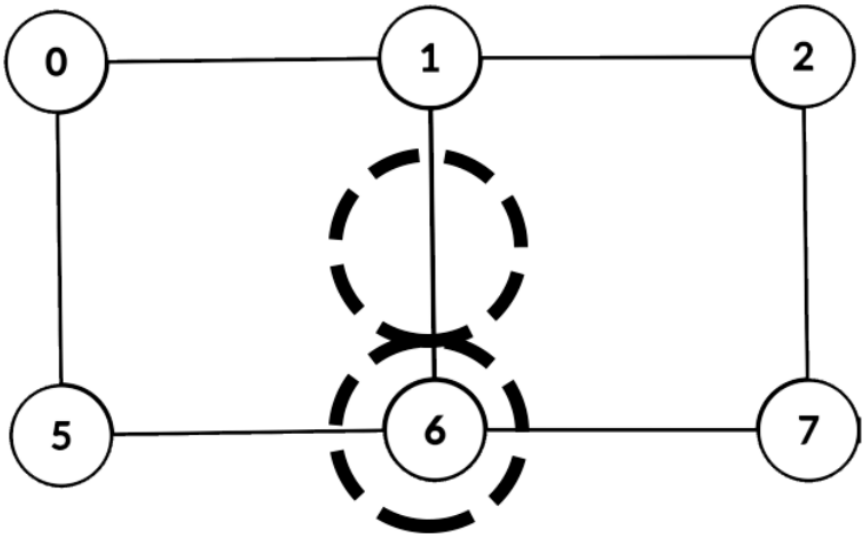
1.2.2 Omezení

Pro úspěšné nalezení plánu je vyžadováno, aby žádní dva agenti nebyli ve stejnou chvíli na stejném místě. Existují i varianty, kdy jednu hranu mohou využít dva agenti, ale touto variantou se bakalářská práce nezabývá. Důležitým omezením je to, že počet agentů musí být striktně nižší, než je počet vrcholů. Pro variantu bez produkce a konzumace agentů existuje další omezení. Toto omezení zakazuje stejnou počáteční a cílovou pozici.

Pro tuto práci, kde výsledek je simulován na robotech typu ozobot, který bude popsán v kapitole 2, předpokládáme, že každý agent má stejný tvar a rychlost pohybu. Dále předpokládáme, že všichni agenti se pohybují po stejném grafu.

1.3 Multi-agentní hledání cest - spojitý čas (MAPF-R)

MAPF-R je definován stejně jako MAPF. MAPF-R stejně jako MAFR pracuje nad neorientovaným grafem G = (V, E), množinou agentů A = $\{a_1, a_2, ... a_K\}$ a počáteční i koncovou konfigurací: α_0 : A → V a α_+ : A → V. Rozdíl je v tom, že nepracuje s diskrétní jednotkou, ale se spojitým rozsahem času. Nadále MAPF-R počítá s rozměry agenta a jejich rychlostí. V našem případě jsou rozměry všech agentů a rychlost totožné. Algoritmus počítající se spojitým časem se nazývá CCBS. [1][4]



Obrázek 1.4 Způsobená kolize, přestože agenti nesdílí ani vrchol ani hranu

▶ **Definice 1.1.** ***(MAPF-R)*** *Multi-agentní hledání cest se spojitým časem je pětice* Σ*-R =* $(G = (V, E), A, \alpha_0, \alpha_+, \rho)$ *kde* G, A, α_0, α_+ *jsou ze základní definice MAPF a* ρ *určuje rozšíření pro spojitý čas následovně:*

1. $\rho.x(v)$, $\rho.y(v)$ *pro každý* $v \in V$ *reprezentuje souřadnice vrcholu* v *v 2D poli*
2. $\rho.velocity(a)$ *pro každý* $a \in A$ *reprezentuje konstantní rychlost agenta* a
3. $\rho.radius(a)$ *pro každý* $a \in A$ *reprezentuje průměr agenta* a

1.3.1 Řešení

Řešením problému je takzvaná kolekce plánů pro jednotlivé agenty $\Pi = [\Pi(a_1), \Pi(a_2), ..., \Pi(a_K)]$, které jsou vzájemně bezkolizní. Plán pro agenta $a \in A$ je posloupnost $\Pi(a) = [((\alpha_0(a), \alpha_1(a)), [t_0(a), t_1(a))); ((\alpha_1(a), \alpha_2(a)), [t_1(a), t_2(a))); ...; ((\alpha_{m(a)-1}, \alpha_{m(a)}(a)),(t_{m(a)-1}, t_{m(a)}))]$, kde $m(a)$ je délka jednotlivých plánů a každá dvojice $(\alpha_i(a), \alpha_{i+1}(a)), [t_i(a), t_{i+1}(a)))$ v posloupnosti odpovídá události průchodu mezi dvojicí vrcholů $\alpha_i(a)$ a $\alpha_{i+1}(a)$. Začínající v čase $t_i(a)$ a končící v čase $t_{i+1}(a)$ (bez tohoto času).

Jinak řečeno, plán pro agenta $a \in A$ je posloupnost přesunů mezi dvojicí vrcholů v daném časovém rozmezí, které odpovídá době přesunu agenta.

1.3.2 Kolize

Kolize ve spojitém čase na rozdíl od standardního MAPF může být způsobena rozměry agenta. V MAPF-R může kolize nastat i mimo samotný vrchol nebo samotnou hranu, ale také mezi dvojicí hran. Například pokud jsou dvě hrany moc blízko u sebe a v jeden okamžik po obou jede agent, který je větší než vzdálenost těchto dvou hran, vznikne kolize. Dva agenti způsobí kolizi, pokud se dotknou.

▶ **Definice 1.2.** ***(Kolize)*** *Kolize mezi jednotlivými plány* $\Pi(a) = [(\alpha_i(a),\ \alpha_{i+1}(a)),\ [t_i(a),\ t_{i+1}(a)))]_{i=0}^{m(a)}$ *a* $\Pi(b) = [(\alpha_i(b),\ \alpha_{i+1}(b)),\ [t_i(b),\ t_{i+1}(b)))]_{i=0}^{m(b)}$ *nastane za následující podmínky:*

1. $\exists i \in \{0,\ 1,\ \ldots,\ m(a)\}$ *a* $\exists j \in \{0,\ 1,\ \ldots,\ m(b)\}$ *takové, že:*
   a. $dist([x_{\alpha_i(a)}, y_{\alpha_i(a)}; x_{\alpha_{i+1}(a)}, y_{\alpha_{i+1}(a)}]; [x_{\alpha_j(b)}, y_{\alpha_j(b)}; x_{\alpha_{j+1}(b)}, y_{\alpha_{j+1}(b)}]) < r_a + r_b$
   b. $[t_i(a), t_{i+1}(a)) \cap [t_j(b), t_{j+1}(b)) \neq \emptyset$

   (vrcholová nebo hranová kolize - dva agenti jsou současně na stejném vrcholu, stejné hraně nebo přecházejí po dvou hranách, které jsou k sobě příliš blízko)

1.4 Příklady z praxe

S problémy multi-agentního hledání cest se člověk může setkat na několika místech. Ať už se jedná o ovládání jednotlivých robotů převážejících zásilky ve skladu, pohyb jednotek v počítačové hře, koordinaci několika filmařských dronů nebo dokonce vyřešení hry patnáctka.

13	2	6	9
15	10	4	8
3	12	7	11
1	5	14	

(a) příklad počítačové hry StarCraft 2 [5] **(b)** hra patnáctka [6]

Obrázek 1.5 Vyobrazení počítačové hry a hry patnáctka, kde je využito multi-agentní hledání cest

Kapitola 2
Ozobot

V této kapitole bude popsán ozobot jako takový, jak s ozobotem pracovat, jeho využití pro vizualizaci algoritmů a využití pro tuto práci.

Ozobot je skvělá pomůcka do škol. Malý programovatelný robot, který se dá perfektně využít pro vizualizaci algoritmů, pro hledání cest. Jsou dva typy ozobotů, typ Evo a typ Bit. V celé práci nadále bude hovořeno o typu Evo, jelikož na tomto typu robota bude následně simulován výsledný algoritmus.

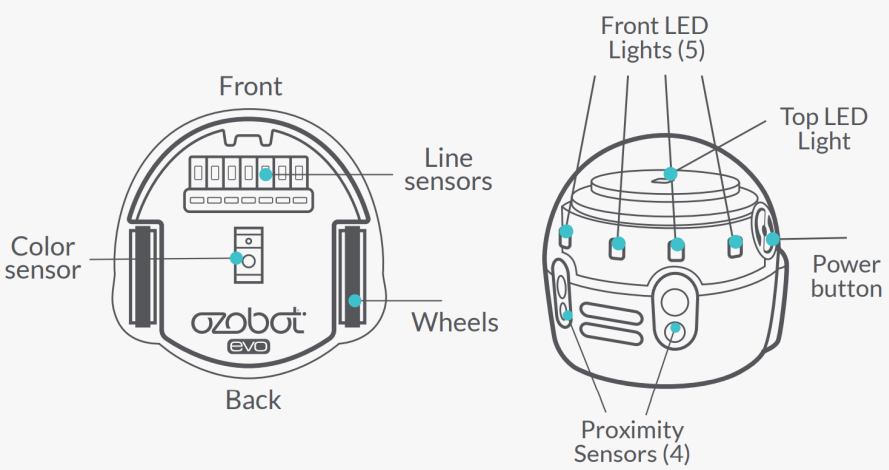
2.1 Popis Ozobota

Ozobot má tvar polokoule o průměru 34mm. Na spodní ploché straně jsou dvě kola, která mu umožňují pohyb po vodorovném povrchu. Kola jsou pogumovaná. Toto pogumování umožňuje ozobotovi snadný pohyb i po lesklých materiálech jako jsou monitory nebo lesklé desky stolu. Zatáčení provádí pohybem každého kola jinou rychlostí. Toto umožní jak otočení na místě, tak plynulý pohyb po křivce.

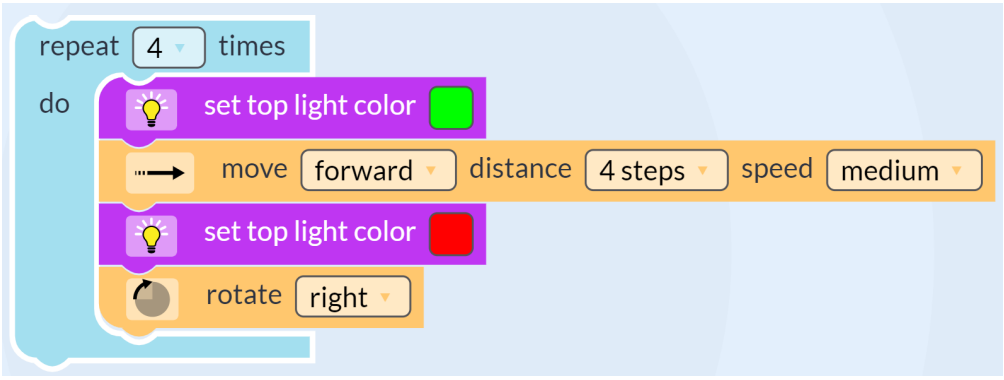
Na této straně nalezneme také senzory pro sledování čáry, po které se ozobot pohybuje a senzory na sledování barev čáry. Tento optický senzor dokáže rozpoznat osm barev: červenou, modrou, zelenou, žlutou, bílou, černou, magentovou a azurově modrou. Pokud by senzor zaznamenal jinou barvu, klasifikuje jí do těchto osmi barev v závislosti na odstínu. Senzor je neskutečně důležitý jak pro naprogramování ozobota, tak pro ovládání jeho rychlosti nebo ovládání jeho dalších periferií jako jsou LED diody nebo reproduktor. Vedle tohoto senzoru nalezneme bílou LED diodu, kterou si ozobot v případě potřeby přisvětluje povrch pro lepší čtení barev.

Na vrchu ozobota nalezneme RGB LED diodu. Vpředu nalezneme dalších pět RGB LED diod. Ozobot je vybaven čtyřmi senzory přiblížení. Dva vpředu a dva vzadu. Na jedné straně se nachází vypínací tlačítko. Toto tlačítko slouží buď k zapnutí robota, spuštění programu nebo pro kalibraci. Na druhé straně se nachází reproduktor. Vzadu se nachází micro USB konektor, který slouží pro nabíjení robota. *Na jedno nabití baterie vydrží 60–90 minut.* [7]

Veškerá elektronika je ochráněna polykarbonátovou slupkou.



Obrázek 2.1 Obrázek ozobota a jeho senzorů [7]



Obrázek 2.2 Příklad programu v OzoBlockly [3]

2.2 Práce s Ozobotem

Před samotnou prací s robotem je potřeba robota nakalibrovat. To se provede v závislosti na zobrazovacím médiu pro ozobota. Pokud se má ozobot pohybovat po displeji, kalibrace bude prováděna na bílem podkladu. Pokud se ovšem ozobot má pohybovat po papíře, je třeba ho umístit na černý kruh. Tento kruh ozobot během své kalibrace přejede. Začátek kalibrace proběhne po podržení vypínacího tlačítka po dobu dvou vteřin. Po této době by se měla horní LED rozblikat. Robot se mezitím zapne. Je ale důležité tlačítko v této době nepustit. Pokud kalibrace proběhla úspěšně, horní LED se po dvou vteřinách rozbliká zeleně. V opačném případě červeně a je potřeba kalibraci opakovat. [8]

Po nakalibrování je robot připravený na plnění základního programu, nebo na nahrání nového programu. Nový program se dá nahrát nablikáním nového programu pomocí programu od výrobce OzoBlockly. Tento program je určen převážně pro vytváření nového programu pro ozoboty. [9]

2.3 Pohyb Ozobota

Pokud ozobot nezaregistruje žádnou čáru, kterou by měl sledovat, zůstává v klidovém stavu. Jakmile senzor zaznamená čáru, rozjede se po ní. Čára má však omezení. V ideálním případě by měla být 5mm široká, všude stejně tlustá a pokud zatáčí, tak vnitřní úhel musí být alespoň 45°. Pokud by čára tvořila písmeno „U“, jednotlivé svislé čáry musí mít mezi sebou rozestup alespoň 25mm.

Ozobot je schopen rozpoznávat i různé barevné kódy. *„Barevný kód je sekvence 2–4 barev, které je ozobot schopen přečíst pomocí optického senzoru a následně na ně reagovat pomocí přeprogramovaného chování a to zpomalením, zrychlením, změnou směru nebo speciálním pohybem.“* [10]

2.4 Využití v této práci

Pro demonstraci výsledků byl zapotřebí robot, který se dokáže okamžitě rozjet, má konstantní rychlost a stejné rozměry jako ostatní roboti stejného typu. Všechny tyto podmínky splňují právě ozoboti. Na svou rychlost se rozjedou během zlomku vteřiny, udržují svou konstantní rychlost a všichni mají stejnou velikost.

Pomocí vizualizéru 4 je zobrazena mapa 4.1. Na dané počáteční stanice jsou položeni ozoboti. Následně je puštěna simulace, při které ozoboti sledují vykreslovanou čáru až do cílové stanice. Odtamtud je ozobot ručně odebrán, aby byla demonstrována konzumace agenta. Tím, že je ozobot odebrán, uvolní prostor pro případné další ozoboty projíždějící kolem, nebo dokonce přes dané dlaždice.

Kapitola 3

Skladová logistika

S problémy týkajícími se skladové logistiky, se potýká velké množství firem. Ať je zboží převáženo jakýmkoliv způsobem, najít optimální řešení není snadné. Firma Amazon se potýkala s problémem, jak rychle a efektivně zboží přepravit ze skladu na jiné místo. Zpočátku bylo zboží přepravováno ručně. To byl nejjednodušší způsob, ale také nejpomalejší. Tato metoda se s narůstajícím množstvím zboží stávala nepoužitelnou. Firma se proto v roce 2012 spojila s firmou *Kiva Systems*. Toto spojení zajistilo Amazonu přístup ke spoustě robotů, které tato firma vyvíjela. Amazon následně zrušil prodej robotů a veškeré si nechával pro své účely. Firma byla přejmenována na „Amazon Robotics“, kde nadále probíhá vývoj a výroba robotů.

3.1 DU-1000

DU je zkratka pro Drive Unit a 1000 je maximální váha v librách (450kg), kterou je stroj schopný uzvednout. Tento robot váží 146,5kg a jeho maximální rychlost je 4,69km/h. Původně byl navrhnut pro fungování decentralizovaně, avšak Amazon pro jejich ovládání používá centralizovaný systém. Jednotka má jednu kameru směřující k zemi pro snímání značek na podlaze, jednu kameru směřující nahoru, aby dokázala číst kódově označené regály, infračervené senzory a nárazníky pro detekci nárazů. Jednotka má šest koleček, dvě hlavní poháněcí a další čtyři pro bezproblémové otáčení. Díky těmto kolečkům, je jednotka schopna otáčení o 360° na jednom místě. Hydraulický zvedák, kterým jednotka zvedá regály, se také může otočit o 360°, aby se nemusel celý robot otáčet při vykládání zboží.

V roce 2014 pracovalo ve skladišti Amazonu 14 000 těchto jednotek. V současnosti jich je přes 200 000. Toto množství jednotek napomohlo Amazonu v roce 2022 úspěšně přepravit 6,5 miliardy balíčků.[11] [12] [13]

3.2 Logistika distribučního centra v Amazonu

Většina informací pochází z [14], kde je perfektně ukázána celá logistika distribučního centra Amazonu v Itálii.

Obrázek 3.1 Obrázek DU-1000 jednotek [12]

3.2.1 Naskladnění zboží

Zboží je do distribučního centra dopraveno pomocí kamionů. Každý kamion je vyložen a jednotlivé palety se zbožím dočasně odloženy na speciálním odkladovém místě. Po příjmu je zboží přemístěno na přijímací stanici. Na stanici je každé zboží vybaleno z přepravních krabic a zadáno do systému. Pokud se jedná o úplně nové zboží, které v systému ještě není, je potřeba ho změřit a zvážit. K tomu slouží takzvaný *cubiscan* [15]. Všechny změřené hodnoty se zadají do systému. Všechny tyto informace jsou nezbytné pro pozdější kontrolu, o které je psáno v kapitole 3.2.6.

Po zadání do systému je zboží posláno v černé přepravce po dopravníkovém pásu na oddělení zaskladnění.

3.2.2 Zaskladnění

Oddělení zaskladnění neboli „stow“ je místo určené pro ukládání věcí do skladu. Zde zaměstnancům přijede zboží v černých přepravkách. Každé zboží je vyndáno z přepravek, naskenováno a uloženo do žlutých regálů. Více o těchto regálech bude v kapitole 3.2.3. Po naskenování zboží zaměstnancem je systémem přidělené místo v regálu. Zaměstnanec tam zboží vloží a potvrdí vložení do správné police načtením QR kódu. Pokud systém vyhodnotí, že je daný regál plný, nechá tento regál odvézt robotem a následně nechá přivézt nový regál.

3.2.3 Sklad

Ve skladu jsou uložené jednotlivé položky zboží ve žlutých regálech - „pots“. Každý regál má přibližně sto polic. Tyto police jsou označeny číslem a písmenem. Písmeno udává pořadí police odspodu a číslo udává pořadí zprava doleva. Každá strana regálu je jiná a má různou velikost polic, aby i velké zboží mohlo být umístěno do těchto regálů. V jednotlivých policích je zboží umístěno *náhodným způsobem* („všechno je všude“). To znamená, že v několika regálech může být stejné zboží. To umožňuje více robotům vyzvednout stejné zboží a nemusí tak čekat na dokončení předchozí objednávky. Každý z regálů má vespod čárový kód, podle kterého robot přesně ví, o který regál se jedná a jakým směrem je regál otočený. Řídící systém má přehled o zboží. Pošle informaci robotovi o tom, kde zboží nalezne, a ke které vykládací stanici ho má dovézt.

Obvykle ve skladišti bývá kolem 6 000 regálů, které obsluhuje 2 000 robotů.

3.2.4 Vykládací stanice

Ze skladu roboti vezou regály se zbožím k vykládacímu místu. Zboží vezou širokou halou, aby byl prostor pro případné vyhnutí se jiným robotům. U každého vykládacího místa pracuje jeden člověk. Pracovník přesně ví, které zboží má vyložit, protože mu počítač přesně ukáže číslo a písmeno police, ve které se zboží nachází. Toto zboží vezme z dovezeného regálu a vloží jej do připravené přepravky, kterou vloží na přepravní pás. Ten přepravku dopraví na balicí místo. Odbaven je vždy pouze jeden robot. Teprve když je z regálu od prvního robota vše požadované vyloženo, může přijet další robot.

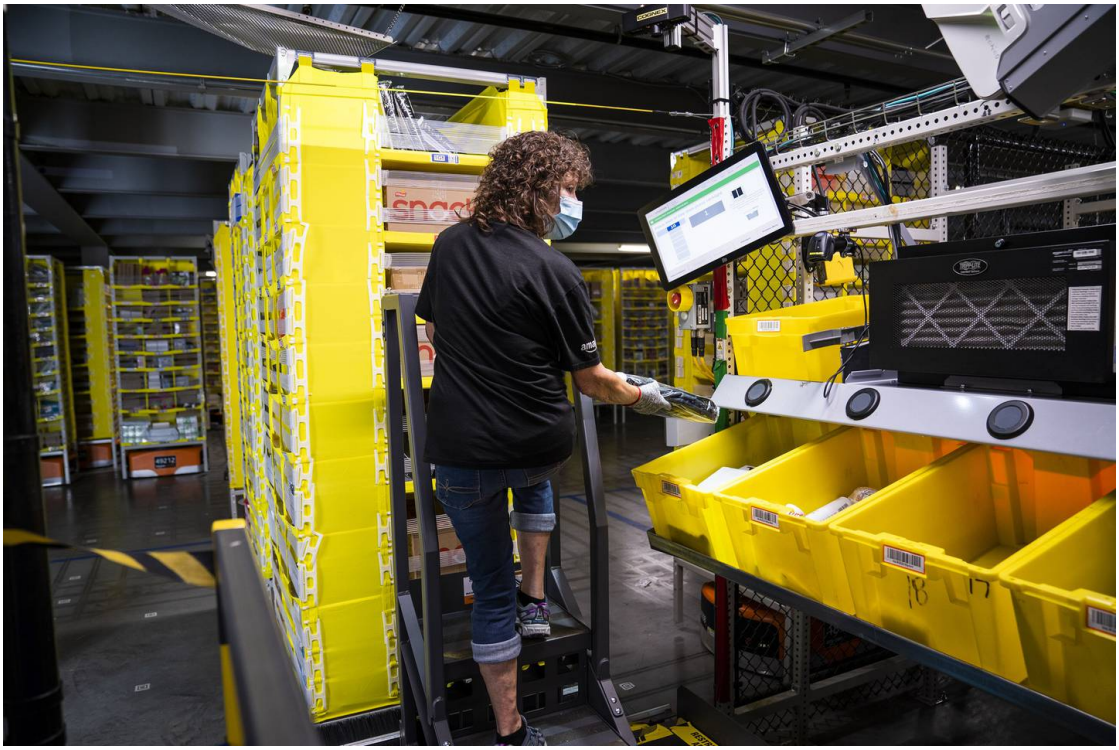
3.2.5 Balící stanice

Balení zboží na balící stanici probíhá manuálně. Zaměstnanec naskenuje přepravku, která mu byla přivezena po dopravníkovém pásu. Následně systém zaměstnanci určí optimální velikost krabice pro zabalení zboží. Jednotlivé papírové krabice jsou na balícím místě složeny a označeny barevným dvouznakovým kódem. Tento kód má na první pozici písmeno a na druhé pozici číslovku. Pracovník vloží do krabice zboží a vyplní přebytečný prázdný prostor papírovou výplní. Když je vše v krabici, zaměstnanec si nechá z *páskovačky* vytisknout pásku. Tato páska má přesné rozměry, aby byla optimálně dlouhá na danou krabici.

Prázdná přepravka je pracovníkem umístěna na kraj balící stanice, odkud je jinými pracovníky odebírána a přepravena zpět k vykládacímu místu.

Celý proces balení zásilek je anonymní. Žádný zaměstnanec neví, komu balí zboží. Aby bylo možné zásilku správně doručit, zaměstnanec nalepí čárový kód na zabalenou zásilku. Kód je generován systémem, aby nedošlo k záměně zboží.

Zabalená zásilka je následně předána na spodní dopravníkový pás, odkud je dopravena na „SLAM“.



Obrázek 3.2 Vyobrazené vykládací místo [16]

3.2.6 SLAM

Zkratka SLAM znamená „Scan, Label, Apply, Manifest“. Tento stroj zváží každý balíček a porovná změřenou váhu s očekávanou vahou, včetně hmotnosti balení. Pokud hmotnost neodpovídá, je balíček předán k manuální kontrole. Pokud váha odpovídá, načte SLAM štítek přilepený na balícím místě a připevní na zásilku štítek s doručovacími údaji. Tento štítek nikdy není domáčknut. Rameno, které připevňuje štítek se zastaví pár centimetrů nad zásilkou a štítek vzduchovým proudem připevní, aby nedošlo k poškození zásilky.

Zákazník má možnost zrušit objednávku ještě předtím, než projede SLAMem. Tato zrušená zásilka je vyřazena a odvezena pomocí dopravníkového pásu na speciální místo, kde je zboží opět umístěno do skladu. Tento dopravníkový pás dosahuje rychlosti až 30km/h[17]

3.2.7 Odchozí vykládací rampy

Poté co zboží opustí SLAM, je dopraveno do třídící jednotky („shipsorter“). Tato jednotka vytřídí jednotlivé zásilky podle druhu přepravy, místa doručení nebo doby, kdy mají být doručené.

Roztříděné zásilky jsou předány přepravcům a následně převezeny na doručovací depa, odkud jsou jednotlivé zásilky přepraveny na místa doručení.

Kapitola 4
Vizualizer

Obsah této práce bude demonstrován pomocí vizualizeru, který navrhl student Jan Chudý ve své práci [18]. Jeho vizualizer na vstupu požaduje mapu v přesném formátu a s přesným pojmenováním. Dále potřebuje na vstup plán z řešiče a počáteční konfiguraci zobrazovacího média, na kterém bude celá simulace zobrazena. Pokud dostane všechny tyto parametry a všechny zpracuje úspěšně, zobrazí mapu a po stisknutí klávesy spustí simulaci.

4.1 Mapa

Mapa je tvaru obdélníku. V obdélníku jsou jednotlivé čtvercové dlaždice, které reprezentují jednotlivé vrcholy grafu. Dvě přilehlé dlaždice mají mezi sebou buď zeď, a nebo nic. V případě zdi mezi jednotlivými vrcholy nevede hrana. V opačném případě tam vedou dvě orientované hrany. Každá z hran vede opačným směrem. Pokud mezi jednotlivými dlaždicemi vede hrana, agent má možnost se přesunout z jedné dlaždice na druhou za cenu odpovídající době jízdy mezi jednotlivými dlaždicemi. Celá mapa je ohraničena zdí, aby bylo na první pohled vidět, kde mapa začíná a kde končí. Mapa je ve formátu txt s koncovkou „.ozomap“. Název souboru je důležitý, protože si z něj jak vizualizer, tak řešič bere důležité parametry.

Formát vstupu vypadá následovně:

- šířka mapy
- x
- výška mapy
- podtržítko
- počet agentů a na konec připsané „a“
- podtržítko
- název mapy

Ve výsledku vstup vypadá třeba následovně: **3x1_2a_example.ozomap**. Jedná se tedy o mapu s šířkou tři a výškou jedna tedy o celkových třech dlaždicích, na které jsou umístěni dva agenti.

```
V =
(0,1,0)
(1,0,2)
(2,2,1)
E =
{0,1}
{1,2}
```

Obrázek 4.1 Příklad zapsání jednoduché mapy

Na prvním řádku souboru se nachází „V =“. Následuje výpis jednotlivých vrcholů. Jednotlivé hodnoty jsou odděleny čárkami.

Formát vrcholu vypadá následovně:

- (
- číslo vrcholu,
- číslo agenta, který má zde počáteční pozici,
- číslo agenta, jež má zde cílovou pozici
-)

Jednotlivý vrchol může vypadat následovně: **(1,0,2)**. Tedy na vrcholu jedna nemá počáteční stanici žádný agent, ale cílovou stanici zde má agent číslo dva.

Následuje jeden řádek s „E =“. Poté výpis jednotlivých hran. Obě hodnoty jsou odděleny čárkou.

Formát hrany vypadá následovně:

- {
- číslo vrcholu, ze kterého vede hrana,
- číslo vrcholu, do kterého vede hrana
- }

Jednotlivá hrana může vypadat následovně: **{0,1}**. Tedy z vrcholu nula vede hrana do vrcholu jedna a obráceně.

4.2 Konfigurační soubory

Na vstupu je požadován pouze jeden konfigurační soubor. Přesto je potřeba nastavit ještě jeden, ze kterého bere potřebné parametry pro řešič. Jeden je předán při startu parametrem, a to konfigurační soubor s nastavením zobrazovacího média. Druhý, který si vizualizer načítá sám, se jmenuje „simulator.ini“.

V prvním je potřeba nastavit rozlišení zobrazovacího média a jeho skutečnou velikost v milimetrech. Skutečnou velikost lze vypočítat na stránce http://screen-size.info.

```
[display] ; Display configuration (http://screen-size.info/)
;Dimensions in pixels
resolution_width = 2560
resolution_height = 1080

;Dimensions in millimeters
display_width = 677
display_height = 290
```

V druhém nastavuje vzhled mapy, cestu k řešiči a styl zobrazení jednotlivých vodících čar pro ozoboty.

```
[ozobot]
; Output configuration for OzoBot
; Real world dimensions in millimeters
    line_width = 5
    wall_width = 2
    tile_border_width = 0.5
    tile_size = 52
    color_code_radius = 6.5
    intersection_width = 25

[solver]
; External solver configuration
; BoOX solvers: mapf_solver_boOX, rota_solver_boOX
; BoOX algorithms: cbs, cbs+, cbs++, smtcbs, smtcbs+, smtcbs++
; production-consumption: not needed = empty
    path = /home/holas/Desktop/me/
    solver = main.py
    algorithm = empty

[simulator]
    agent_type = animated
    ; Time in milliseconds that should take Ozobot to go between tiles
    step_time = 1620
    ; Time lag of the tail (cannot be greater than step_time)
    tail_lag = 600
    ; Flags
    display_borders = true
    display_walls = true
    direction_preview = true
    colors = true
```

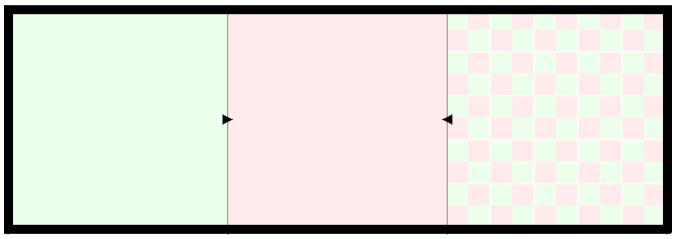
4.3 Funkcionalita

Pokud jsou oba konfigurační soubory nastavené, vizualizer po spuštění rozparsuje jednotlivé parametry. Pokud vše proběhne bez chyby, odešle mapu do řešiče. Z řešiče obdrží plán. Pokud je plán ve správném formátu a vizualizer ho dokázat rozparsovat, je zobrazena mapa.

4.3.1 Barevné dlaždice

Aby se vizuálně oddělily počáteční a koncové pozice agentů, vizualizer je zobrazuje jinou barvou. Klasická dlaždice, která nereprezentuje ani počátek ani cíl, je bílá. Dlaždice, která reprezentuje počátek, je světle zelená a dlaždice reprezentující cíl, je světle červená. Pokud existuje dlaždice, která je jak počáteční, tak cílová, zobrazuje

se jako zeleno-červená se šachovnicovým vzorem. Tyto barvy nemá možnost bez zásahu do kódu změnit.



Obrázek 4.2 Možnosti dlaždic

4.3.2 Umístění agenta

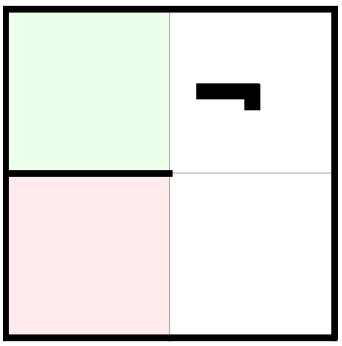
Vizualizer byl navrhnut pro agenty typu ozobot 2. Každá dlaždice má tedy rozměry, na které se ozobot perfektně vejde.

Po spuštění vizualizeru je zobrazena mapa. Je potřeba umístit ozoboty na zelené dlaždice, a to následovně: ozobot musí být nasměrován tím směrem, kterým ukazuje malá černá šipka na jedné z hran zelené dlaždice. Je potřeba aby byl ozobot přesně ve středu. Pokud by byl ozobot moc vzadu, nemusel by zachytit navigující čáru a mohl by zůstat na stejné místě.

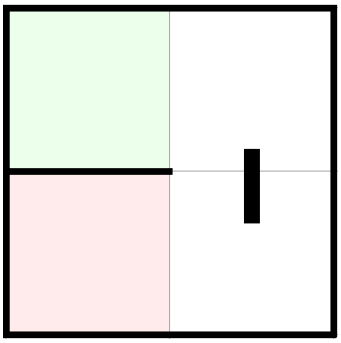
4.3.3 Pohyb agenta

Vizualizer zajišťuje pohyb ozobota tím, že vykresluje danou cestu ve středu dlaždic. Podporované jsou tři typy vykreslení *animated, dummy a ozobot*. To, který způsob bude použit, se nastavuje před spuštěním samotné aplikace v *simulator.ini* 4.2 v sekci *[simulator]* parametr *agent_type*.

animated v tomto nastavení je pod ozobotem generována krátká čára, která se před ním objevuje a za ním zaniká. Nevýhodou je možnost ujetí čáry při záseku ozobota. Pokud ozobot čáru ztratí, stojí na jednom místě a nepokračuje svojí vypočítanou cestou. Tím může vzniknout kolize s jiným ozobotem.



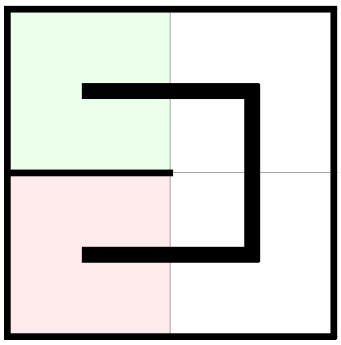
(a) nastavení animated, zatáčka



(b) nastavení animated, rovina

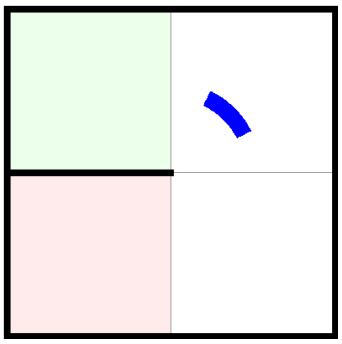
Obrázek 4.3 nastavení: animated

dummy v tomto nastavení je celá trasa ozobota vykreslena najednou jako jedna dlouhá, nepřerušovaná čára. Je nevhodné pro větší počty agentů, protože překrývající se čáry mohou způsobit nepřehlednost. Výhodou je, že čára nemůže ozobotovi ujet, a tím se nestane, že by ozobot zůstal stát uprostřed mapy.

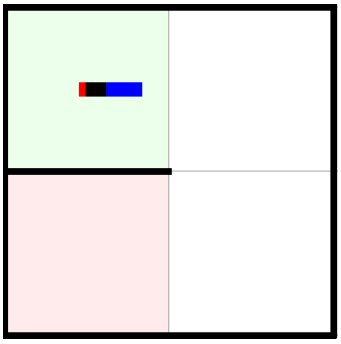


Obrázek 4.4 nastavení: dummy

ozobot v tomto případě, je potřeba ozobota přeprogramovat. Vizualizer generuje červeno-černo-modré čáry s umělými křižovatkami, které dokáží upravit ozobotovu rychlost.



(a) nastavení ozobot, zatáčka

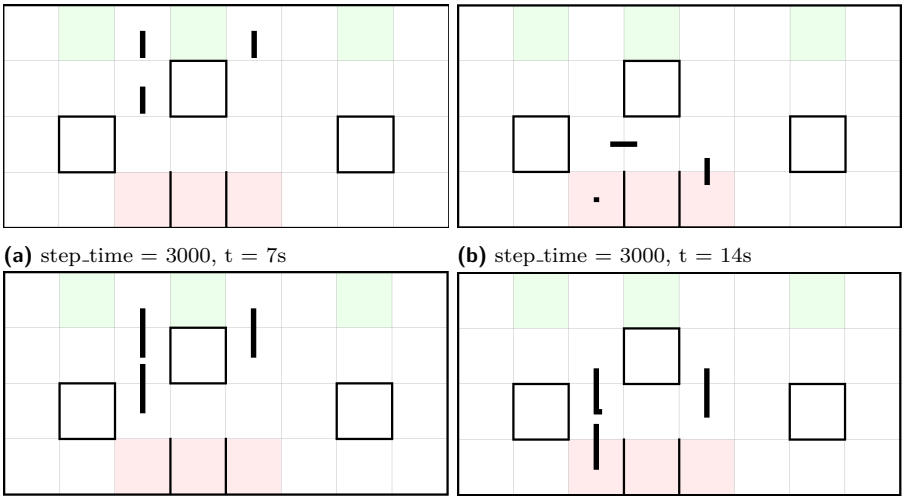


(b) nastavení ozobot, rovina

Obrázek 4.5 nastavení: ozobot

Dalšími podstatnými parametry jsou:

step_time, pomocí kterého je možné nastavit rychlost pohybu čáry, po které agent jede. Pokud bude hodnota příliš nízká, může agentovi ujet. Pokud čára agentovi ujede, zůstane stát na místě. Nejen že takový agent nedojede do své cílové stanice, navíc tím blokuje cestu všem ostatním agentům. Pokud bude hodnota příliš vysoká, agent se může pohybovat trhaně, nebo dokonce čáru předjet, a tím ji ztratit.



(a) step_time = 3000, t = 7s

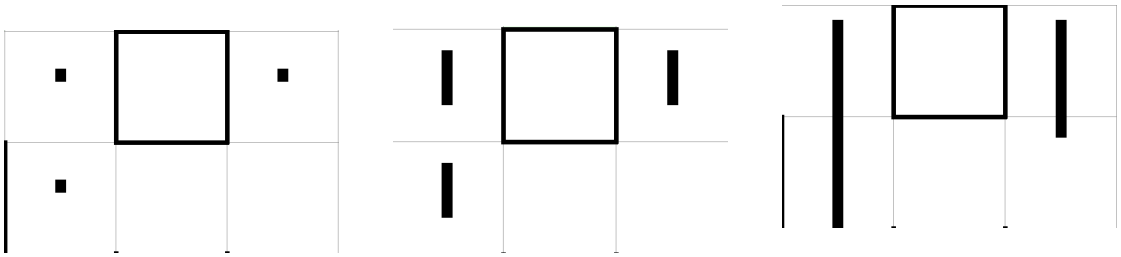
(b) step_time = 3000, t = 14s

(c) step_time = 1500, t = 5s

(d) step_time = 1500, t = 6s

Obrázek 4.6 Vyobrazení nastavení step_time v závislosti na čase

tail_lag, tímto parametrem je nastavena délka čáry. Pokud je čára příliš dlouhá, může se stát, že se spojí s čárou pro jiného agenta, který jede v těsné blízkosti za předchozím agentem. Toto chování je zobrazeno na obrázku 4.7. Bude-li čára příliš krátká, agent může ztratit tuto čáru v důsledku jemného záseku. Hodnota tohoto parametru musí být **menší než step_time**.



(a) tail_lag = 100

(b) tail_lag = 1200

(c) tail_lag = 2900, spojeni dvou čar do jedné

Obrázek 4.7 Vyobrazení jednotlivých nastavení pro tail_lag

Oba tyto parametry je nutné nastavit pro každý display na jiné hodnoty, protože vizualizér 4 neupravuje tyto parametry s velikostí a rozlišením obrazovky. Oba parametry jsou závislé, a proto při změně jednoho je dobré zkontrolovat a popřípadě upravit druhý.

Kapitola 5
Praktická část

Cílem bylo vytvořit nový, či modifikovat existující algoritmus, který počítá s produkcí a konzumací agentů. Toto zadání bylo splněno. Vznikl nový samostatný řešič v jazyce python, který s touto variantou počítá. Řešič je zjednodušenou variantou MAPF-R řešiče.

5.1 Dijkstrův algoritmus

Informace do této kapitoly byly čerpány z přednášek předmětu Algoritmy a grafy 1 (BI-AG1), Fakulta informačních technologií, České vysoké učení technické v Praze.

DA je algoritmus, který pro zadaný orientovaný graf a pro zadaný počáteční vrchol nalezne vzdálenosti od všech vrcholů k počátečnímu vrcholu. Algoritmus se chová obdobně jako algoritmus prohledávání do šířky, na ohodnoceném grafu. Funkcionalita je v 1 popsána pseudokódem.

Funkci DA si můžeme představit následovně: na každý vrchol vložíme budík, který je nastaven na dobu, za jak dlouho se z počátečního vrcholu dostaneme k danému vrcholu. Díváme se, na kterém vrcholu první zazvoní budík. První zazvonil na vrcholu A. Všem sousedům A k jejich původnímu času přičteme dobu, kterou nám trvalo dostat se z počátečního vrcholu do vrcholu A. Pokud se do nějakého budíku dostaneme z jiného vrcholu dříve, nastavíme tuto lepší hodnotu. Toto opakujeme, dokud nezazvoní všechny budíky.

5.1.1 Relaxace

▶ **Definice 5.1.** *Přepočítání ohodnocení H(w) následníka w vrcholu c v DA na jedenáctém řádku pseudokódu 1 budeme nazývat* ***relaxace následníka*** *w vrcholu c.*

5.1.2 Konečnost a správnost DA

Pro dokázání konečnosti a správnosti DA je potřeba dokázat čtyři vlastnosti.

```
Result: Pole vzdáleností H a pole předchůdců P
forall vrcholy v: do
    stav(v) := NOTFOUND;
    H(v) := +∞;
    P(v) := None;
end
stav(v0) := OPEN;
H(v0) := 0;
while Existují nějaké vrcholy ve stavu OPEN do
    vyber otevřený vrchol v, s nejmenší H(V);
    forall následníky w vrcholu v do
        if H(W) je větší, než H(v) + ohodnocení hrany (v, w) then
            H(w) := H(v) + + ohodnocení hrany (v, w);
            stav(w) := OPEN;
            P(w) := v;
        end
    end
    stav(v) := CLOSED;
end
```

Algoritmus 1: Dijkstrův algoritmus

▶ **Tvrzení 5.2** (**Vlastnost 1**). *Ohodnocení H(v) v průběhu DA nikdy neroste a je-li konečné, rovná se délce nejkratší dosud nalezené v_0v-cesty zachycené inverzí posloupnosti v, P(v), P(P(v)), ... , v_0.*

Na počátku výpočtu tvrzení platí pro v_0, neboť $H(v_0) = 0$. Pokaždé, kdy má dojít k relaxaci vrcholu, dojde ke snížení jeho H(w). H(w) je následníkem vrcholu v, který je otevřený. Tento vrchol má konečné H(w). H(v) odpovídá délce dosud nalezené nejkratší cesty z vrcholu v_0. Relaxace w znamená nalezení kratší cesty z v_0. Při každé relaxaci si DA zapamatuje předchůdce, ze kterého vede kratší cesta. Tuto informaci ukládá do pole P.

▶ **Tvrzení 5.3** (**Vlastnost 2**). *DA postupně otevírá všechny vrcholy dosažitelné z v_0 a jakmile je otevřený vrchol uzavřen, nemůže být znovu otevřen.*

První otevřený a uzavřený vrchol je vrchol v_0. Vrchol se stane otevřeným poté, co se stane předchůdcem dříve otevřeného vrcholu. Z toho vyplývá, že je dosažitelný z v_0. DA vybírá pro expanzi vždy otevřený vrchol s nejnižším H(v), všechny ostatní otevřené vrcholy c mají hodnotu $H(c) \geq H(v)$. DA takový vrchol uzavře. Tento vrchol nemůže být nikdy znovu otevřen, protože nesplní podmínku na jedenáctém řádku pseudokódu 1.

▶ **Tvrzení 5.4** (**Vlastnost 3**). *DA se v konečném čase zastaví.*

DA otevírá pouze dosažitelné vrcholy a uzavřený vrchol se zároveň znovu neotevře, viz 5.3. DA v každém kroku cyklu na řádcích 14-17 uzavře jeden vrchol. Počet kroků cyklu v DA je tedy shora omezen počtem vrcholů. Počet následníků v každé iteraci je také konečný.

▶ **Tvrzení 5.5** (**Vlastnost 4**). *DA uzavírá všechny dosažitelné vrcholy v pořadí jejich neklesající vzdálenosti od v_0 a po skončení je $H(v) = d(v_0, v)$.*

V každém kroku DA uzavře jeden otevřený vrchol. Tento vrchol v má právě nejnižší H(v) ze všech otevřených vrcholů. Hodnoty H(v) tvoří neklesající posloupnost. Uzavřený vrchol v nemůže být znovu otevřen, viz 5.3. H(v) tohoto vrcholu má na konci stále tuto hodnotu. To podle 5.2 je nejkratší cesta z počátečního vrcholu v_0. Kratší cesta tedy neexistuje, a proto H(v) = d(v_0, v).

5.2 Řešič

Je samostatný program, který na základě zadaného vstupu nalezne cesty pro všechny agenty na dané mapě a vytvoří plán. Tento plán je následně uložen do souboru a zároveň vypsán do konzole. Pokud budou agenti dodržovat zadaný plán, dopraví se ze zadaného místa na místo určení.

5.2.1 Vstup

Vstupem pro řešič musí být mapa uložená souboru formátu *.ozomap*. Tento formát je stejný, jako požaduje vizualizer 4. Struktura této mapy je popsána v 4.1.

5.2.2 Reprezentace mapy

DA pracuje nad orientovaným grafem. Je proto důležité mapu ze vstupu načíst a reprezentovat jako orientovaný graf.

5.2.3 Postup načítání mapy

Program otevře soubor. Nejprve si z názvu soubory vyčte všechny potřebné parametry, jako je velikost mapy a počet agentů. Následně jsou do paměti uloženy všechny počáteční a koncové pozice. Všechny tyto pozice jsou srovnány podle čísla agentů.

Následuje krok tvorby samotné mapy. Pro každou hranu načtenou ze souboru je natažena hrana mezi jednotlivými vrcholy, jedna pro každý směr. Pro hranu **{0,1}** vzniknou v paměti hrany **0** → **1** a **1** → **0** Pokud tyto vrcholy ještě neexistují, jsou vytvořeny a následně jsou nataženy hrany. Ohodnocení této hrany je pro všechny stejné, protože všechny dlaždice mají stejnou vzdálenost a agenti, kteří mají všichni stejnou rychlost, jí překonají za stejný čas.

Následně je načtená mapa uložena do souboru, pro případnou kontrolu, pokud by algoritmus nepracoval správně.

5.2.4 Reprezentace grafu v paměti

Mapa je v paměti reprezentována jako slovník. Klíčem tohoto slovníku je vrchol, ze kterého vedou hrany. Hodnota pro daný klíč pak odpovídá dalšímu slovníku, který má jako klíč vrchol, do kterého vede hrana. Hodnota tohoto vnořeného slovníku pro daný klíč pak odpovídá ohodnocení hrany.

Toto řešení umožňuje snadné dohledání všech sousedů daného vrcholu.

5.2.5 Hledání plánu

Řešič nalezne pro každého agenta zvlášť nejkratší cestu pomocí DA. Tyto cesty jsou nalezeny nezávisle na sobě, tudíž může a také nemusí existovat kolize. Délka cesty odpovídá době, za kterou Ozobot 2 přejede přes všechny dlaždice z počátku do konce.

```
forall agenty a: do
    Dijkstra(start_pos_a);
end
check_collision();
```

Algoritmus 2: Hledání cest

5.2.6 Řešení kolizí

V tomto okamžiku jsou nalezeny všechny (ne nutně disjunktní) cesty. Je tedy potřeba z těchto cest udělat disjunktní. Toho je docíleno tak, že skrz všechny kroky, v daném čase je kontrolováno, zda některé dvě cesty netvoří kolizi. Tato akce je popsáno v psuedokódu 4 Pokud některá z dvojic cest tvoří kolizi, je do cesty s menším indexem vsunuta akce *čekání* 1.2.1 a všechny jeho akce jsou o tuto dobu posunuty. Funkce je ukázána pseudokódem 5. Celý algoritmus opakujeme do doby, než jsou vyřešeny všechny kolize. Pokud jsou vyřešeny všechny kolize, plán je hotový. Výsledný plán je vypsán a uložen.

```
collisions := True;
while collisions do
    collisions := False;
    forall kroky v čase: do
        duplicities := najdi_kolize(krok v daný čas);
        if duplicities then
            vlož_čekání();
            posuň_kroky();
            collisions := True;
        end
    end
end
```

Algoritmus 3: Řešení kolizí

5.2.7 Výstup

Výstupem je hotový *diskrétní* plán, který je uložen do textového souboru a zároveň vypsán do konzole. Plán je diskrétní, protože vizualizer zobrazuje jen diskrétní plány. Tento plán může vypadat jako na obrázku 5.1.

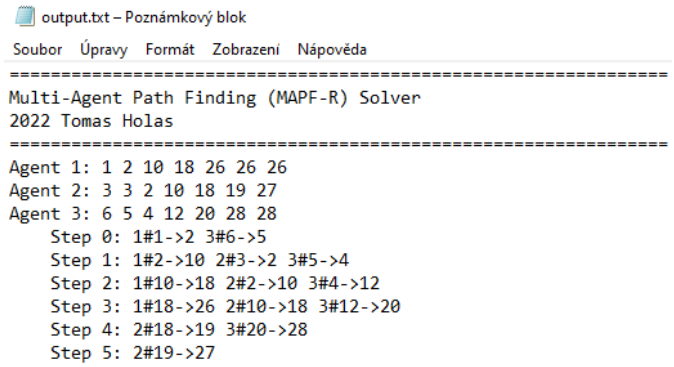
První čtyři řádky jsou pouze úvodní text, který budou mít veškeré plány společný.

```
Result: Pole kolizí
Input: pole moves = (agent, from_vertex, to_vertex)
forall moves m: do
    dictionary_of_arrays[to_vertex].append(index of m);
end
forall klíče d v dictionary_of_arrays: do
    if len(dictionary_of_arrays[d]) > 1 then
        return dictionary_of_arrays[d];
    end
end
return [];
```

Algoritmus 4: Nalezení kolizí

```
Input: t = time, agent, move = (agent, from_vertex, to_vertex)
if pokud t > velikost plánu then
    return;
end
if pokud je t = velikosti plánu then
    zvětšení velikosi plánu o 1 - přidání move na konec plánu;
    return;
end
přidání move v čase t;
forall move m: do
    if m.agent = agent then
        move := m;
        break;
    end
end
odstraň krok move z plánu;
setřiď kroky plánu v daném čase podle čísla agenta;
rekurzivní volání(t + 1, agent, move);
```

Algoritmus 5: Posunutí kroků



Obrázek 5.1 Příklad vyřešeného plánu

Jelikož je plán diskrétní, následuje výpis všech dlaždic, na kterých se daný agent v daném diskrétním čase má nacházet. Tento výpis pozic pro každého agenta musí být stejně dlouhý pro všechny agenty. Přestože by měl agent zmizet po dojetí do cílové pozice a nepřekážet ostatním, vizualizer stejně počítá s tím, že agent je stále na této pozici. Z tohoto důvodu jsou přidány další zbytečné pozice, se kterými řešič ale nepočítá.

Následuje výpis jednotlivých kroků (každý krok odpovídá jednotce diskrétního času). Za krokem s číslem následuje výpis jednotlivých přesunů.

přesuny mohou vypadat následovně:

- číslo agenta
- #
- vrchol, ze kterého se agent přesouvá
- $->$
- vrchol, na který se daný agent přesouvá
- mezera

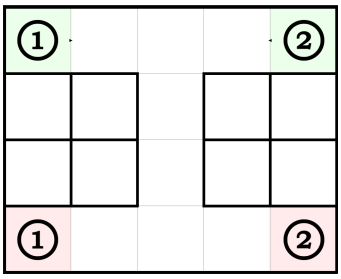
Pokud je v daném čase kroků více, kroky jsou vypsány hned za sebe. Čekání ve výpisu jednotlivých kroků není uvedeno.

Kapitola 6

Pokusy

Správnost algoritmu je demonstrována pomocí vizualizéru 4 a robotů typu ozobot 2. Pro ověření bylo vytvořeno šest map, které simulují skutečné situace, se kterými se roboti mohou v běžném provozu setkat.

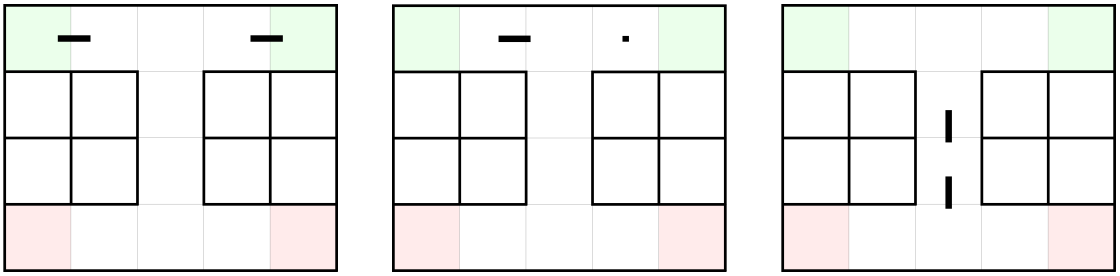
6.1 Mapa mustwait



Obrázek 6.1 Mapa mustwait

Tato mapa 6.1 simuluje situaci, kdy dva agenti mají být ve stejném čase na stejném vrcholu a je tedy nutné, aby jeden agent počkal, než projede druhý agent.

Jak je vidět na obrázku 6.2 oba agenti vyjedou ve stejný čas. Oba agenti potřebují na stejný vrchol. Projede tedy první agent. Druhý agent zatím vyčkává. Jen co první agent odjede, a tím uvolní prostor pro druhého agenta, vyjede druhý agent. Agent druhý je následně zařazen za prvního agenta a oba mohou dojet na své cílové stanice.



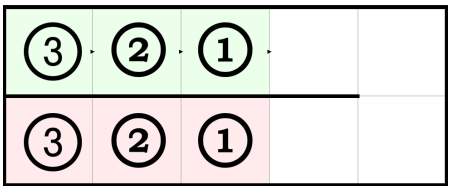
(a) oba agenti vyjedou ve stejnou dobu

(b) druhý agent čeká, než projede první

(c) druhý agent se zařadí za prvního

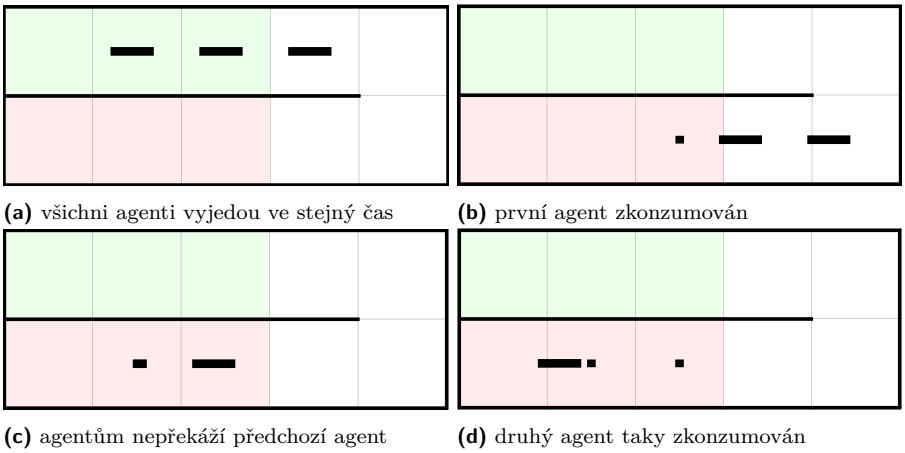
Obrázek 6.2 Ukázka čekání druhého agenta, než projede první

6.2 Mapa snake



Obrázek 6.3 Mapa snake

Každý agent, který dorazí na svou cílovou pozici, je zkonzumován. Konzumace znamená, že zmizí z dané mapy a tím uvolní cestu všem ostatním agentům. Tato mapa 6.3 má za úkol demonstrovat chování při konzumaci agentů. Obrázek 6.4 znázorňuje scénář, kdy první agent po dojetí do cílové stanice je zkonzumován. Druhý a třetí agent mohou tedy bez problémů projet dále. Následně i druhý agent dojede do své cílové stanice a je zkonzumován. Tím uvolní cestu třetímu agentovi. Třetí agent má volný prostor a dojede do své cílové stanice.



(a) všichni agenti vyjedou ve stejný čas

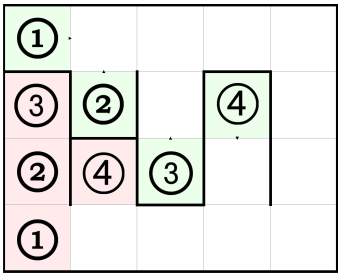
(b) první agent zkonzumován

(c) agentům nepřekáží předchozí agent

(d) druhý agent taky zkonzumován

Obrázek 6.4 Agenti pokračují v pohybu přes zkonzumované agenty

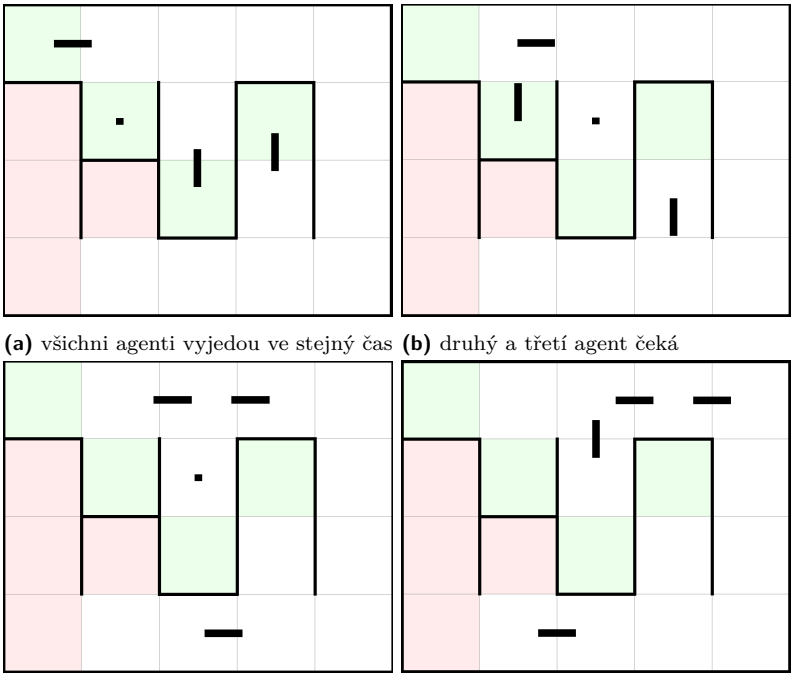
6.3 Mapa round



Obrázek 6.5 Mapa round

Tato mapa má ukázat, že řešič 5.2 zvládne vyřešit situaci, kdy musí jeden agent provést akci čekání dvakrát. Zároveň je ukázáno, že agenti, kteří nejsou blokováni, vykonají svou cestu bez jakýchkoliv obtíží.

Po celou dobu pokusu agent číslo čtyři není nikým blokován a tak celou trasu projede bez zastavení. První agent blokuje nejdříve druhého agenta. Druhý agent je nucen počkat, než první agent projede. Po průjezdu prvního agenta, druhý agent má volnou cestu a může jet. V této chvíli je třetí agent blokován prvním agentem a následně i druhým agentem. Musí tedy počkat, než projedou oba tito agenti. Po průjezdu těchto dvou agentů vyjede i třetí. Teď už agenty nic neblokuje a mohou dojet na své cílové stanice, kde jsou zkonzumováni.

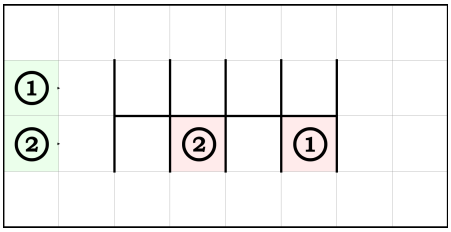


(a) všichni agenti vyjedou ve stejný čas **(b)** druhý a třetí agent čeká

(c) třetí agent čeká na průjezd agentů **(d)** žádný z agentů nemusí čekat

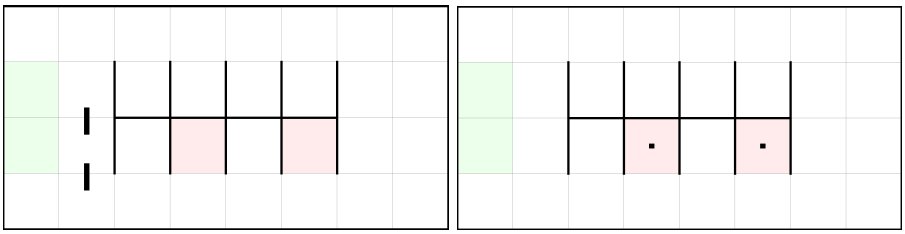
Obrázek 6.6 Třetí agent je nucen vyčkat na přesun ostatních agentů

6.4 Mapa load



Obrázek 6.7 Mapa load

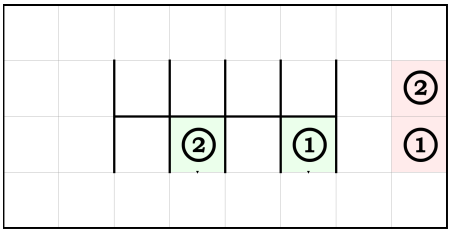
Tato mapa je zjednodušenou kopií skladiště v Amazonu. Cíle vyznačené na této mapě reprezentují jednotlivé žluté regály, ve kterých je uloženo zboží. Agenti vyjedou ze své počáteční stanice ve stejný čas a každý z nich má přidělený svůj regál, ke kterému musí dojet a zboží vyzvednout. Po vyzvednutí zboží začíná nový úkol; dovezení regálu na vykládací stanici. Tento úkol je popsán v sekci 6.5.



(a) všichni agenti vyjedou ve stejný čas **(b)** oba agenti vyzvedli regál

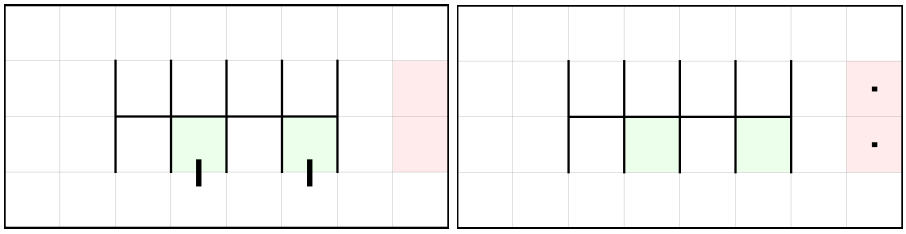
Obrázek 6.8 Proces vyzvedávání regálu ze skladu

6.5 Mapa unload



Obrázek 6.9 Mapa unload

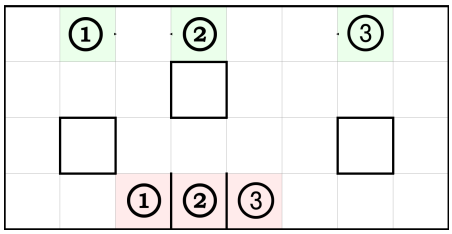
Tato mapa je opět znázornění zjednodušeného skladu Amazonu. Cílem této mapy je dopravit vyzvednuté regály se zbožím na vykládací stanici. Oba agenti vyjedou s regály ve stejný čas a dopraví daný regál na vykládací stanici. Na této stanici jsou následně zkonzumováni. Na této mapě nebylo potřeba žádné čekání robotů a místo demonstrace fungování řešiče ukazuje proces vyzvedávání zboží ve skladu Amazonu za pomocí robotů.



(a) všichni agenti vyjedou ve stejný čas **(b)** oba agenti dovezli regál

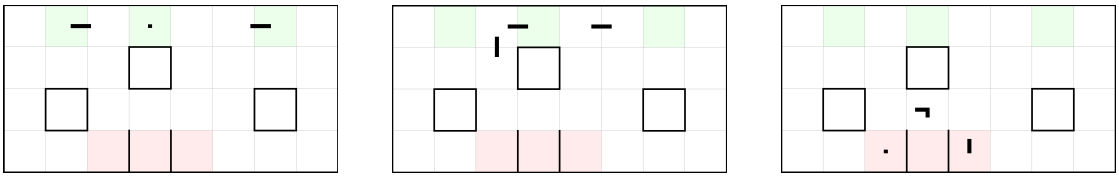
Obrázek 6.10 Proces převozu regálů ze skladu na vykládací stanici

6.6 Mapa amazon



Obrázek 6.11 Mapa amazon

Tahle mapa demonstruje pohyb agentů v prostoru, kde se nachází překážky. Cílem agentů je se vyhnout všem překážkám včetně ostatních robotů. První a třetí agent vyjedou ve stejný čas. Druhý agent nevyjede, protože mu cestu blokuje první agent a tak čeká. Po průjezdu prvního agenta se druhý agent zařadí. Všichni agenti mají volnou cestu a dojedou na své cílové stanice.



(a) první a třetí agent vyjedou ve stejný čas, druhý agent čeká **(b)** druhý agent čekat na průjezd prvního **(c)** druhý agent se zařadí za prvního

Obrázek 6.12 Ukázka čekání druhého agenta, než projede první

6.7 Problémy a výsledky pokusů

Při pokusech bylo potřeba se vypořádat s problémy. Jeden z problémů byl vyladění parametru *step_time*, který udává rychlost pohybu čáry. Tento parametr je potřeba nastavit znovu na každém zobrazovacím zařízení. Není tedy snadné přepnout na větší obrazovku bez přenastavení tohoto parametru.

Dalším velmi závažným problémem byl špatný pohyb ozobota po lesklé obrazovce. Pokud byla obrazovka lesklá, ozobot obtížně sledoval čáru. Často čáru

Tabulka 6.1 Výsledky jízdy ozobotů na různých displejích

displej	úspěch	neúspěch
FullHD lesklý	1	9
FullHD antireflexní	7	3
FullHD matný	10	0
4K lesklý	0	10

ztratil, nebo čáru ani nezaregistroval. Pokud byla stejná mapa i stejný ozobot testován na antireflexní obrazovce, výsledky byly mnohem lepší. Ozobot stále občas ztrácel čáru, ale počet neúspěšných pokusů byl minimální. Nejlepší výsledky se dostavily při použití matného displeje.

Pro testy byly použity čtyři displeje. Jejich výsledky jsou prezentovány v 6.1. Na všech displejích byli použiti stejní ozoboti. Každý pokus byl proveden na mapě 6.11 a pokusů bylo celkem 10. Tato mapa byla vybrána kvůli své jednoduchosti, nutnosti zatáčení agentů a zároveň protože všichni agenti nevyjeli ve stejný čas.

Závěr

Hlavním cílem práce bylo vytvořit algoritmus pro multi-agentní hledání cest, který počítá s produkcí a konzumací robotů. Nakonec byl vytvořen celý řešič obsahující tento algoritmus. V rámci práce byla vysvětlena teorie, bez které by nebylo možné porozumět zbytku práce, následně provedena rešerše ohledně problému, na jehož vyřešení práce pracuje, popis robotů, na kterých byl problém demonstrován a popis samotného řešení včetně pokusů.

Jelikož se jedná o první verzi řešiče, je zde prostor pro vylepšení. Tím by se mohl zabývat kdokoliv, kdo bude navazovat na tuto práci. Využití pro tuto práci nalezne jakákoliv firma, která pro svou skladovou logistiku využívá nedecentralizovaně řízené roboty.

Bibliografie

1. SURYNEK, Pavel. Multi-agent Path Finding with Continuous Time and Geometric Agents Viewed through Satisfiability Modulo Theories [online]. 2019 [cit. 2022-05-03]. Dostupné z: http://surynek.net/publications/files/Surynek_Continuous-MAPF_WoMAPF-2019.pdf.
2. SILVER, David. Cooperative Pathfinding. In: YOUNG, R. Michael; LAIRD, John E. (ed.). *Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference, June 1-5, 2005, Marina del Rey, California, USA* [online]. AAAI Press, 2005, s. 117–122 [cit. 2022-04-21].
3. SHARON, Guni; STERN, Roni; FELNER, Ariel; STURTEVANT, Nathan R. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence* [online]. 2015, roč. 219, s. 40–66 [cit. 2022-04-21]. ISSN 0004-3702. Dostupné z DOI: https://doi.org/10.1016/j.artint.2014.11.006.
4. ANDREYCHUK, Anton; YAKOVLEV, Konstantin; ATZMON, Dor; STERN, Roni. Multi-Agent Pathfinding with Continuous Time. In: [online]. 2019, s. 39–45 [cit. 2022-05-01]. Dostupné z DOI: 10.24963/ijcai.2019/6.
5. MORROLORD. How can I achieve a "formation move"in StarCraft 2? [Online]. 2015 [cit. 2022-05-09]. Dostupné z: https://gaming.stackexchange.com/questions/186429/how-can-i-achieve-a-formation-move-in-starcraft-2.
6. MILER, Tadeáš. How can I achieve a "formation move"in StarCraft 2? [Online]. 2013 [cit. 2022-05-09]. Dostupné z: https://hlavolam.maweb.eu/patnactka.
7. EVOLLVE, INC. *Educator's Guide* [online]. 2018 [cit. 2022-04-10]. Dostupné z: https://files.ozobot.com/stem-education/ozobot-educators-guide.pdf.
8. OZO EDU, INC. *Calibration* [online]. 2022 [cit. 2022-04-10]. Dostupné z: https://ozobot.com/support/calibration.
9. OZO EDU, INC. OzoBlockly [online]. 2022 [cit. 2022-04-10]. Dostupné z: https://ozobot.com/create/ozoblockly.
10. EVOLLVE, INC. *Color Code Guide* [online]. 2020 [cit. 2022-04-10]. Dostupné z: https://storage.googleapis.com/files.ozobot.com/resources/ozobot-color-code-guide.pdf.
11. MECHATRONICS AND ARTIFICIAL INTELLIGENCE ORGANIZATION. Drive Unit [online]. [B.r.] [cit. 2022-04-21]. Dostupné z: https://mecai.org/kiva.html.

12. BHASIN, Kim; CLARK, Patrick. How Amazon is creating a robot arms race where it always wins [online]. 2016 [cit. 2022-04-21]. Dostupné z: https://www.independent.ie/business/technology/news/how-amazon-is-creating-a-robot-arms-race-where-it-always-wins-34844243.html.
13. EDWARDS, David. Amazon now has 200,000 robots working in its warehouses [online]. 2020 [cit. 2022-04-21]. Dostupné z: https://roboticsandautomationnews.com/2020/01/21/amazon-now-has-200000-robots-working-in-its-warehouses/28840/.
14. AMAZON TOURS. *Virtuální prohlídka distribučního centra Amazon (záznam)* [video]. 2021 [cit. 2022-04-21]. Dostupné z: https://www.youtube.com/watch?v=4ZTLJ-5OtN4.
15. QUANTRONIX. CUBISCAN [online]. 2022 [cit. 2022-05-07]. Dostupné z: https://cubiscan.com/.
16. NAVIDI, Leila; TRIBUNE, Star. Marie Steele works Oct. 8, 2020, at Amazon's Shakopee, Minn., fulfillment center, during the coronavirus pandemic. [Online]. 2020 [cit. 2022-05-05]. Dostupné z: https://www.chicagotribune.com/business/ct-biz-covid-19-amazon-hiring-jobs-20201124-of5pj7pvqfg4tffeavwlt2hwsu-story.html.
17. ROSER, Christoph. The Inner Workings of Amazon Fulfillment Centers – Part 4 [online]. 2019 [cit. 2022-05-07]. Dostupné z: https://www.allaboutlean.com/amazon-fulfillment-4/.
18. CHUDÝ, Jan. *Simulation of Centralized Algorithms for Multi-Agent Path Finding on Real Robots*. 2020. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology.

Obsah přiloženého média

- `readme.txt` ..... popis obsahu média
- `BP` ..... zdrojová forma práce ve formátu LaTeX
- `src`
  - `src` ..... zdrojové kódy implementace
  - `ozobot-mapf-simulator-master` ..... zdrojové kódy vizualizéru
- `text` ..... text práce
  - `bakalářská-práce-holastom.pdf` ..... text práce ve formátu PDF