VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY
FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

PLÁNOVÁNÍ CESTY ROBOTU POMOCÍ MRAVENČÍCH ALGORITMŮ

ROBOT PATH PLANNING BY MEANS OF ANT ALGORITHMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR
Bc. MARTIN PĚNČÍK

VEDOUCÍ PRÁCE
SUPERVISOR
RNDr. JIŘÍ DVOŘÁK, CSc.

BRNO 2014

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2013/2014

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Martin Pěnčík

který/která studuje v **magisterském navazujícím studijním programu**

obor: **Aplikovaná informatika a řízení (3902T001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Plánování cesty robotu pomocí mravenčích algoritmů

v anglickém jazyce:

Robot path planning by means of ant algorithms

Stručná charakteristika problematiky úkolu:

Úkolem systému pro plánování cesty robotu je najít cestu z výchozího do cílového bodu tak, aby se robot nedostal do kolize se známými překážkami, aby nebyla porušena případná omezení kladená na pohyb robotu a aby byla optimalizována nějaká kriteriální funkce. K možným přístupům k řešení tohoto problému patří také metody rojové inteligence, jako např. mravenčí algoritmy.

Cíle diplomové práce:

1. Analyzovat přístupy k plánování cesty robotu.
2. Popsat metody rojové inteligence a jejich aplikace na plánování cesty.
3. Navrhnout a implementovat systém pro plánování cesty robotu pomocí mravenčích algoritmů.
4. Provést ověřovací a srovnávací experimenty.

Seznam odborné literatury:

MOHEMMED, A. W.; SAHOO, N. C.; GEOK, T. K. Solving shortest path problem using particle swarm optimization. Applied Soft Computing, vol. 8, 2008, pp. 1643-1653.
TAN, G.-Z.; HE, H.; SLOMAN A. Ant Colony system algorithm for real-time globally optimal path planning of mobile robots. Acta Automatica Sinica, vol. 33, 2007, pp. 279-285.
ZHU, Q.; HU, J.; CAI, W.; HENSCHEN, L. A new robot navigation algorithm for dynamic unknown environments based on dynamic path re-computation and an improved scout ant algorithm. Applied Soft Computing, vol. 11, 2011, pp. 4667-4676.

Vedoucí diplomové práce: RNDr. Jiří Dvořák, CSc.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2013/2014.

V Brně, dne 15.11.2013

L.S.

Ing. Jan Roupec, Ph.D. Ředitel ústavu	prof. RNDr. Miroslav Doupovec, CSc., dr. h. c. Děkan fakulty

Abstrakt
Tato diplomová práce se zabývá plánováním cesty robotu. Obsahuje přehled obecných přístupů pro plánování cesty. Dále popisuje metody rojové inteligence a jejich aplikace na plánování cesty robotu. Práce obsahuje návrhy změn pro mravenčí algoritmy a prezentuje výsledky experimentů provedených pomocí implementovaných algoritmů.

Summary
This thesis deals with robot path planning. It contains an overview of general approaches for path planning and describes methods of swarm intelligence and their application for robot path planning. This paper also contains proposals of adjustments for ant algorithms and it presents experimental results of algorithm implementation.

Klíčová slova
Plánování cesty, robot, rojová inteligence, mravenčí algoritmy

Keywords
Path planning, robot, swarm intelligence, ant algorithms

PĚNČÍK, M. *Plánování cesty robotu pomocí mravenčích algoritmů.* Brno, Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2014, 63 s. Vedoucí RNDr. Jiří Dvořák, CSc.

Já, Martin Pěnčík, prohlašuji, že jsem diplomovou práci vypracoval samostatně a že jsem uvedl všechny použité prameny a literaturu.

Brno 30. 5. 2014

Bc. Martin Pěnčík

Děkuji vedoucímu diplomové práce RNDr. Jiřímu Dvořákovi, CSc. za cenné rady a připomínky. Dále děkuji Ing. Radku Zavřelovi za přizpůsobení podmínek v zaměstnání pro dokončení této práce. V neposlední řadě děkuji své rodině za podporu poskytnutou při studiu.

Bc. Martin Pěnčík

Obsah

1. Úvod

Diplomová práce se zabývá využitím rojové inteligence pro řešení problému plánování cesty robotu se zaměřením na mravenčí algoritmy.

V úvodu práce je vysvětlen problém plánování cesty robotu a jsou uvedeny aplikace plánování cesty a robotiky obecně. Druhá kapitola podrobněji pojednává o plánování cesty robotu a o využívaných přístupech. Třetí kapitola se věnuje definici rojové inteligence a popisu jejích metod s uvedenými odkazy na aplikace pro hledání cesty robotu. Čtvrtá kapitola podrobně popisuje dva návrhy z literatury s uvedenými navrženými úpravami. Pátá kapitola popisuje implementaci algoritmů. Šestá kapitola obsahuje popis a výsledky provedených srovnávacích experimentů. Závěrečná část shrnuje výsledky práce.

1.1. Plánování cesty

Některé z nejdůležitějších úkolů autonomní robotiky leží v oblasti automatického plánování pohybu. Cílem je umožnit zadání úkolu ve vysokoúrovňovém jazyce a zařídit, aby robot úkol následně převedl na řešení v podobě řady jednoduchých, vykonatelných pohybů. Typickým úkolem je najít cestu pro robot, ať už je to robotická paže nebo mobilní robot, z jedné konfigurace do jiné tak, aby se robot vyhnul překážkám.

Od tohoto základního úkolu dospělo plánování k řešení obrovského množství variací problému, jež umožňují aplikace v oblastech, jako je animování digitálních postav, plánování chirurgických zákroků, automatické ověření rozložení pracovišť v továrnách, mapování neprozkoumaného prostředí, navigace přes měnící se prostředí, plánování postupu montáže a návrh léčiv. Nové aplikace přinášejí nové otázky, které musí být řešeny pomocí algoritmů plánování pohybu.

Protože akce v reálném světě podléhají fyzikálním zákonům, neurčitostem a geometrickým omezením, při návrhu a analýze algoritmů pro plánování pohybu vyvstává unikátní kombinace otázek z oblasti mechaniky, teorie řízení, geometrie a počítačových věd. Dopad automatického plánování pohybu tedy přesahuje původní aplikační oblast.

Možnost sestavení počítačem řízených mechanických systémů, které snímají okolí, plánují a vykonávají svůj vlastní pohyb, přispěla k rozvoji vědy položením základních otázek, které jinak nemusely vyvstat. [3]

1.2. Příklady využití plánování cesty a robotiky obecně

1.2.1. Problém stěhování klavíru

Jedná se o základní problém plánování cesty. Je dáno trojrozměrné tuhé těleso, např. mnohostěn, a množina známých překážek. Cílem je nalezení bezkolizní cesty pro všesměrové volně se pohybující těleso z počáteční pozice do cílové pozice. Překážky jsou nepohyblivé a jejich pozice a tvar jsou detailně známy. Provedení plánované cesty je absolutně přesné. Tento způsob se nazývá off-line plánování, protože plánování je dokončeno v předstihu před vykonáním pohybu.

Existuje několik variací tohoto problému. Jedním z nich je problém stěhování pohovky, kde se těleso pohybuje v rovině mezi rovinnými překážkami. Další variací je zobecněný

problém přesunu (*generalized mover's problem*), kde se robot může skládat z množství tuhých těles spojených klouby, např. robotická paže. [3]

1.2.2. AERCam

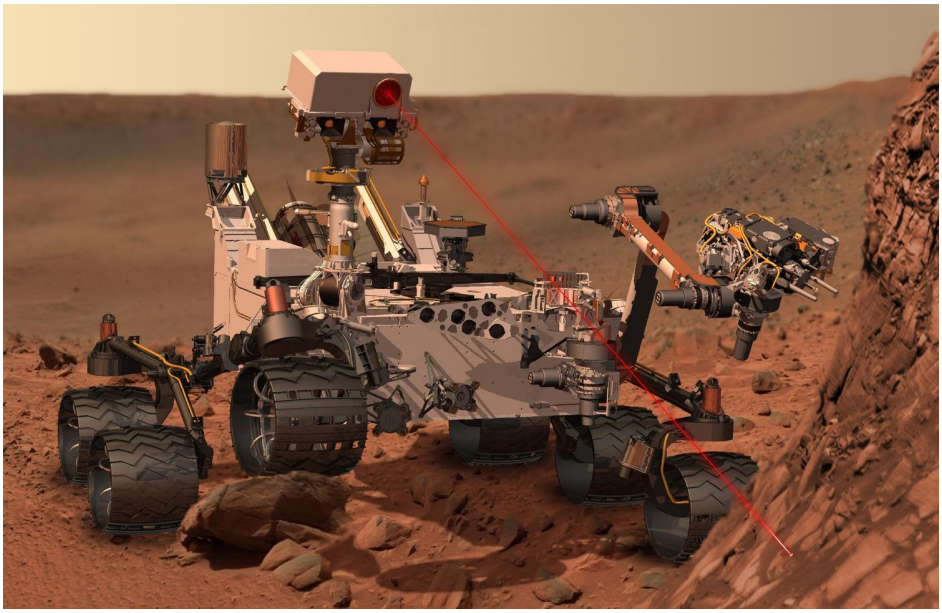
AERCam (*Autonomous Extravehicular Activity Robotic Camera*) je autonomní robotická kamera pro vizuální kontroly vnějších prostorů ve vesmíru, vyvíjená v NASA. Jedná se o volně létajícího robota kulového tvaru vybaveného 12 chladnými plynovými tryskami, které umožnují generování síly a momentu v jakémkoli směru. Funkční typ AERCam Sprint autonomní není. [1]

Plánování pohybu kamery je podobné problému stěhování klavíru. Protože pohyb je vyvolán tahem trysek, je nutné naplánovat také rychlost pohybu, ne pouze dráhu, po které se bude robot pohybovat. Řízení bere v potaz také dynamické vlastnosti robotu. Problém stěhování klavíru řeší pouze geometrii a kinematiku pohybu. [3]

1.2.3. Průzkum planet

Jedním z největších úspěchů nasazení robotů je průzkum Marsu. Během posledních let bylo vysláno několik průzkumných robotů. V roce 1997 byl na Mars dopraven *Sojourner*, který poskytl fotografie okolního terénu. *Sojourner* se nedostal příliš daleko od místa přistání, jeho pohyb byl generován off-line na Zemi. V roce 2004 byli na Mars dopraveni další robotičtí průzkumníci: *Spirit* a *Opportunity*. Jejich úkolem bylo především prověření výskytu vody v minulosti na Marsu. [3]

Poslední mobilní robotickou sondou je *Curiosity*. Ta dosedla na povrch Marsu v roce 2012. Mezi její úkoly patří zjištění, zda se na Marsu vyskytoval život, geologický průzkum, charakterizování klimatu a příprava pro průzkum prováděný lidmi. [13]



Obrázek 1.1: Sonda *Curiosity* na povrchu Marsu. [23]

1.2.4. Odstraňování min

Podle humanitární organizace CARE je na světě rozmístěno přibližně 110 milionů pozemních min (rok 2013). [10] Roboti mohou hrát klíčovou roli při rychlém a bezpečném odstraňování min z postiženého regionu. Prvním podstatným krokem je nalezení min. Při odminování musí robot projít senzorem nad všemi body dané oblasti, ve které je podezření na výskyt min. Aby toho bylo dosaženo, musí robot postupovat podle opatrně volené cesty přes zkoumanou oblast. Naplánovaná cesta musí pokrýt všechny body oblasti. Princip pokrývání oblasti má využití i v jiných oblastech, včetně úklidu podlahy, sekání trávníku a sklízení úrody. Ve všech těchto případech musí robot znát svou polohu pro zaručení celkového pokrytí. [3]

1.2.5. Stacionární průmyslové manipulátory

Stacionární robotické manipulátory vykonávají celou řadu úkolů, včetně montáže, svařování a natírání. Při natírání musí robot rovnoměrně nanést nátěr přes všechny body cílového povrchu. Pokrytí povrchu představuje nové úkoly. Je třeba zajistit rovnoměrné rozložení barvy, což je mnohem složitější požadavek než pouhé pokrytí. Cílový povrch většinou není rovný a robot musí příhodně využít své stupně volnosti k vedení stříkací pistole nad povrchem.

Zavádění robotů do průmyslu je důsledkem ekonomických faktorů. Z tohoto důvodu je kladen velký důraz na minimalizaci času potřebného k vykonání úkolu. Minimalizace času se dosahuje časově optimálními plány pohybu. Další úkoly mohou mít užitek v jiných druzích optimality, například minimalizaci spotřeby energie nebo paliva. [3]

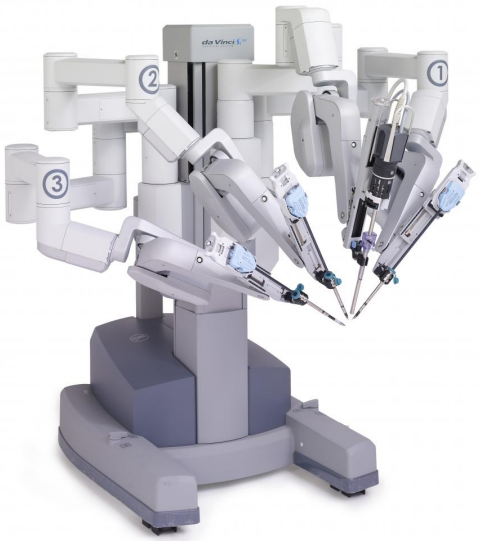
1.2.6. Robotická chirurgie

Roboty jsou stále více využívány při invazivních i neinvazivních chirurgických zákrocích. Příkladem neinvazivního použití je stereotaktická radiochirurgie, která využívá překřížené ionizující záření pro léčbu nádorů mozku (*Cyberknife* [37], *Gamma Knife* [11]). Mezi invazivní zákroky, které mohou být prováděny za pomoci robotů, patří řezání a šití (*da Vinci Surgical System* [8]). Využití robotů umožňuje minimalizovat narušení okolních tkání a zkrátit tak dobu rekonvalescence. [3]

1.2.7. Výzkum léků

Důležitou součástí návrhu léčiv a studia chorob je porozumění toho, jak se proteiny skládají do své přirozené nebo nejstabilnější konfigurace. Při výzkumu jsou proteiny považovány za kloubově spojené segmenty. Výzkumníci používají plánování pohybu pro zjištění možných cest pro složení původně rozloženého řetězce proteinu.

Ve farmaceutickém průmyslu se proteiny kombinují s menšími molekulami pro vytvoření komplexů, které jsou důležité pro prevenci a léčení chorob. Metody plánování pohybu se používají pro analýzu pohybů vazeb molekul, čímž umožňují simulované testování léčiv dříve, než budou syntetizovány v laboratoři. [3]



Obrázek 1.2: Chirurgický robot *da Vinci Surgical System*. [24]

1.2.8. Další

Uvedený výčet není zdaleka vyčerpávající. Mezi další aplikace lze zahrnout: *BigDog* – čtyřnohý robot pro přenášení nákladů těžkým terénem, *Segway PT* – dvoukolový elektrický dopravní prostředek, *ASIMO* – humanoidní robot, *Predator*, *Reaper* a množství jiných bezpilotních letounů, generování pohybu postav a objektů v animovaných filmech a počítačových hrách.

2. Přístupy k plánování cesty robotu

Neformální definice problému plánování cesty je dle [16] následující: pro zadané počáteční umístění robotu vypočítej, jak se má robot postupně pohybovat, aby se dostal do cílového umístění takovým způsobem, aby se vůbec nedotkl překážek.

Algoritmy pro plánování cesty obsahují následující vstupní parametry: počáteční umístění robotu, cílové umístění robotu, geometrický popis robotu a překážek ve volném prostředí. Výstupem algoritmu je přesný popis způsobu, jak postupně přesouvat robot z počáteční pozice do cílové pozice, aniž by se dotkl oblasti obsazené překážkou.

2.1. Robot a prostředí

Pro jednoznačné zadání a následné řešení úkolu je nutné přesně popsat jak robot, tak i prostor, ve kterém se robot může pohybovat. Prostor, ve kterém se robot pohybuje, se nazývá pracovní prostor. Pracovní prostor lze reprezentovat jako dvourozměrný či trojrozměrný euklidovský prostor. V pracovním prostoru se kromě robotu mohou vyskytovat také překážky omezující jeho pohyb. Pro popis robotu a prostoru slouží konfigurace robotu a konfigurační prostor.

Konfigurace robotu je úplná specifikace pozice každého bodu robotu. Konfigurační prostor robotu je prostor všech možných konfigurací robotu. Konfigurace je bodem v tomto abstraktním konfiguračním prostoru. Rozměr konfiguračního prostoru je dán počtem stupňů volnosti robotu. Rozměr konfiguračního prostoru také udává minimální počet parametrů potřebných pro jednoznačné určení konfigurace. Pokud například uvažujeme bodový robot, který se může posouvat v rovině (bez úvahy natočení), jednoznačné určení polohy je možné popsat pomocí 2 souřadnic relativně vůči pevnému souřadnému systému. Konfigurační prostor tohoto robotu je dvourozměrný.

Některé konfigurace v konfiguračním prostoru znamenají srážku robotu s překážkou. Z toho důvodu se definuje konfigurační prostor překážek, což je množina všech konfigurací, kdy by došlo ke srážce s překážkou. Geometrická interpretace je taková, že překážka a robot by se překrývaly. Dále definujeme volný konfigurační prostor jako množinu všech konfigurací, ve kterých nedojde ke kolizi s překážkou.

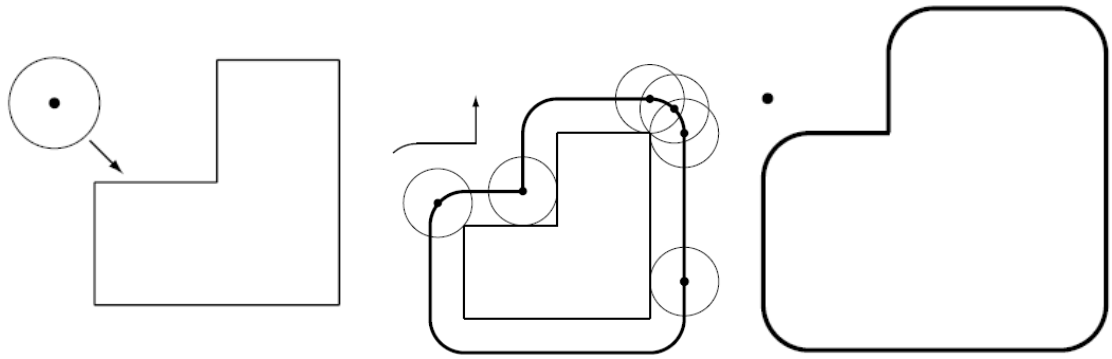
Smyslem konfiguračního prostoru je reprezentace stavu robotu jako bodu. Tímto způsobem je hledání cesty robotu v pracovním prostoru převedeno na hledání cesty pro bod v konfiguračním prostoru. Úkol nyní můžeme formulovat tak, že hledáme řadu na sebe navazujících konfigurací ve volném konfiguračním prostoru, která vytvoří spojitou cestu z počáteční konfigurace do cílové konfigurace. [3]

Obrázky 2.1 a 2.2 ilustrují vztah mezi pracovním a konfiguračním prostorem.

2.1.1. Omezení robotu

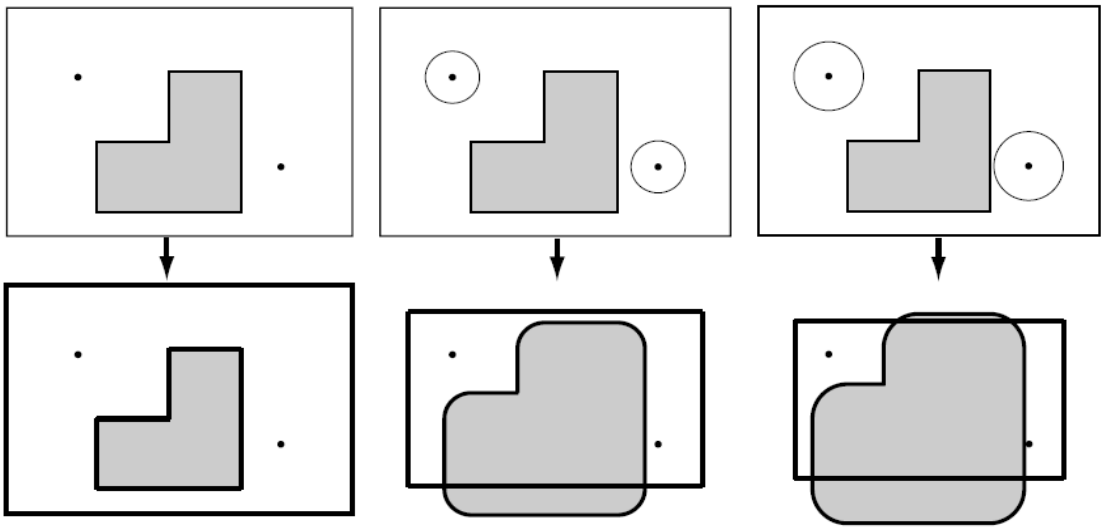
Robot nemusí být při svém pohybu omezen pouze přítomností překážky. Další omezení mohou vyplývat z vlastností robotu.

Robot, jehož počet stupňů volnosti rychlosti odpovídá počtu stupňů volnosti pozice, se nazývá všesměrový neboli holonomní. Předcházející definice holonomního robotu znamená, že robot může měnit svou rychlost nezávisle ve všech směrech.



(a) (b) (c)

Obrázek 2.1: (a) Kruhový robot se přibližuje k překážce v pracovním prostoru. (b) Posouváním robotu po hranici překážky a zaznamenáváním křivky zanechané středem robotu se vytvoří pro robot zakázaný konfigurační prostor překážky. (c) Plánování pohybu pro kruhový robot v pracovním prostoru (a) bylo přetrasformováno na plánování pohybu pro bodový robot v konfiguračním prostoru. [3]



(a) (b) (c)

Obrázek 2.2: Horní řada zobrazuje pracovní prostor, dolní řada znázorňuje konfigurační prostor pro (a) bodový mobilní robot, (b) kruhový mobilní robot, (c) větší kruhový mobilní robot. [3]

Pokud robot nemůže měnit svou rychlost nezávisle ve všech směrech, jedná se o robot neholonomní. Do kategorie neholonomních robotů patří např. roboty typu auto a roboty typu tank. Roboty typu auto se nedokáží otáčet na místě. Změny orientace mohou dosáhnout pouze příslušným natočením kola nebo nápravy následovaným dopředným nebo zpětným pohybem. Roboty typu tank se mohou otáčet na místě, mohou se však pohybovat pouze ve směru rovnoběžném s pásy. [36] Dalším příkladem neholonomního robotu může být model letadla. V případě neholonomích robotů musí plánovač cesty respektovat omezenou manévrovatelnost robotu a generovat cestu splňující jeho pohybová omezení.

Podrobnější rozbor neholonomních mobilních robotů je možné nalézt v literatuře: [28], [25].

2.1.2. Zpracování prostředí

Plánování cesty se obvykle skládá ze dvou hlavních kroků: předzpracování a zpracování dotazu. V kroku předzpracování se získá reprezentace volného konfiguračního prostoru mezi všemi překážkami v oblasti. Následuje fáze zpracování dotazu, kde se přistoupí k prohledávání grafu nebo k použití funkce pro nalezení cesty z počátečního bodu do cílového. [33]

Cílem předzpracování je vytvořit model vnějšího světa, ve kterém se následně hledá cesta. Modelem světa je obvykle grafová struktura nebo funkce definovaná nad konfiguračním prostorem.

Základní členění metod plánování cesty je následující:

1. Mapy cest – spojitost volného prostoru je zachycena pomocí grafu nebo sítě cest.
2. Dekompozice do buněk – prostor je rozdělen na jednoduché části (buňky), spojitost je reprezentována grafem sousednosti buněk.
3. Potenciálová pole – nad prostorem je definována funkce, která má globální minimum v cílové konfiguraci a maxima v prostoru překážek. [29]

2.2. Mapy cest

Prostředí je zpracováno do mapy cest, což je grafová struktura, kde vrcholy jsou body volené v závislosti na typu metody. Vrcholy grafu jsou spojeny hranou, existuje-li mezi nimi cesta, po které robot může přejít. Mapa cest připomíná síť silnic a dálnic. V této analogii je hledání cesty převedeno na úkol skládající se z nalezení cesty z výchozího bodu na dálnici, přemístění po dálnici do bodu, který je nejblíže k cíli, a z tohoto bodu do cíle. Metoda mapy cest patří mezi úplné metody, což znamená, že pokud cesta existuje, metoda ji nalezne. [33]

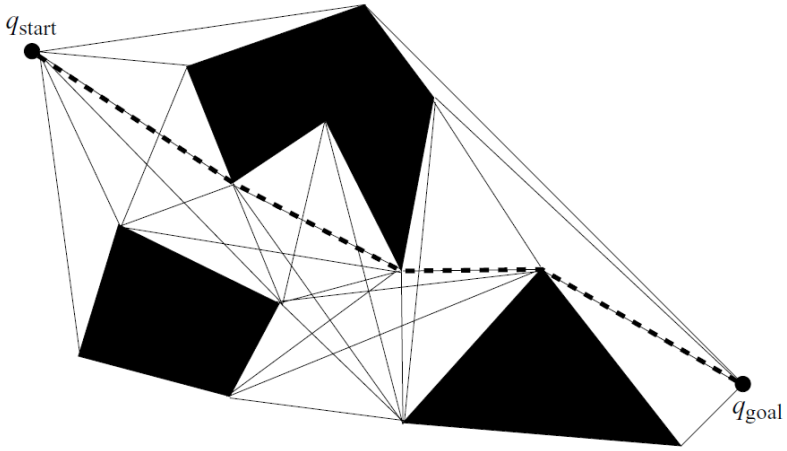
Plánování cesty pomocí mapy cest sestává ze 3 hlavních kroků:

1. Vytvoření mapy cest.
   (a) Uzly jsou body ve volném konfiguračním prostoru (nebo na jeho hranici).
   (b) Dva uzly jsou spojeny hranou, pokud je mezi nimi volná cesta.
2. Spojení startovního bodu robotu k mapě cest v pomocném bodě q_1 a cílového bodu v pomocném bodě q_2.
3. Nalezení cesty na mapě cest mezi body q_1 a q_2.

Výsledkem je cesta vedoucí ze startovního bodu do cílového. [6] Mapy cest se dělí na deterministické a pravděpodobnostní. Mezi deterministické mapy cest patří graf viditelnosti, graf tečen a Voroného diagram.

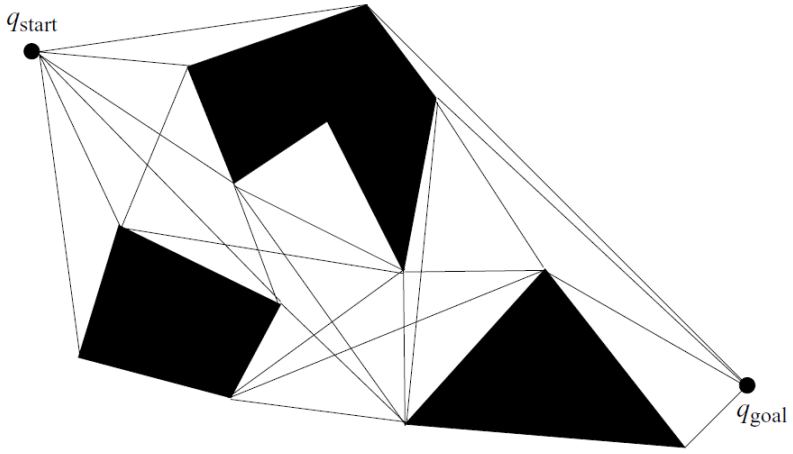
2.2.1. Graf viditelnosti

Graf viditelnosti se používé zejména pro prostředí, kde jsou překážky definovány jako mnohoúhelníky. Uzly grafu viditelnosti odpovídají vrcholům překážek. Dvojice uzlů je spojena hranou, pokud jsou součástí jedné strany překážky nebo se úsečka mezi nimi neprotíná překážku (tj. jsou viditelné). Nejkratší cesta v mapě cest je také globálně nejkratší cestou (platí pouze pro 2D prostředí). Zahrnutím startovního a cílového bodu mezi uzly mapy cest se získá požadované napojení na mapu cest. [6]



Obrázek 2.3: Graf viditelnosti. Tenké úsečky značí hrany grafu viditelnosti pro prostředí se třemi překážkami, reprezentovanými vyplněnými mnohoúhelníky. Tučně čárkovaná lomená čára znázorňuje nejkraší cestu mezi startovním a cílovým bodem. [3]

Redukovaný graf viditelnosti je podgrafem grafu viditelnosti. Někdy se označuje jako graf tečen. Sestává pouze z uzlů, které jsou v konvexních vrcholech překážek, a z hran, které jsou tečnami překážek, tj. z hran, jejichž prodloužení je ve volném prostoru. [6]

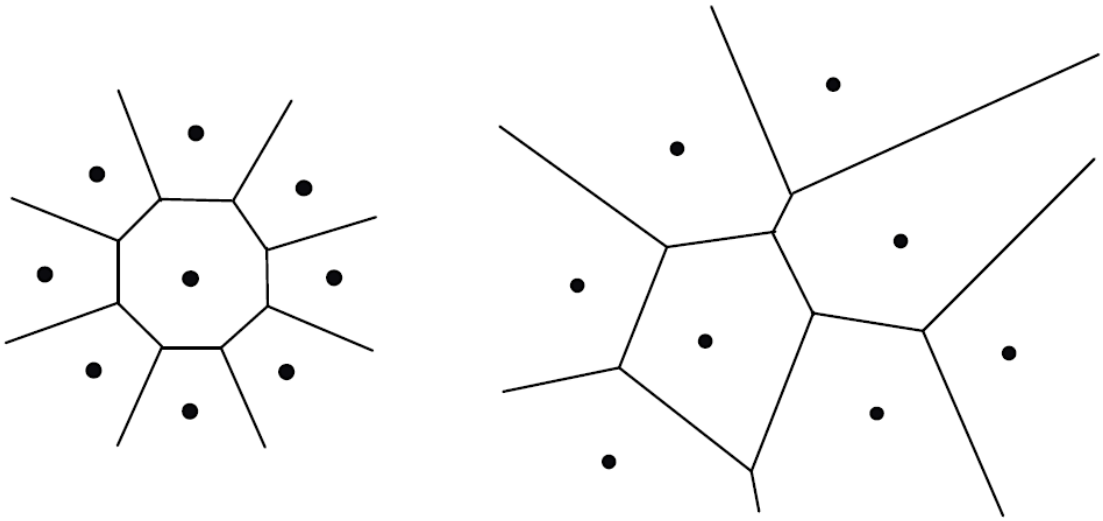


Obrázek 2.4: Redukovaný graf viditelnosti. [3]

2.2.2. Voroného diagram

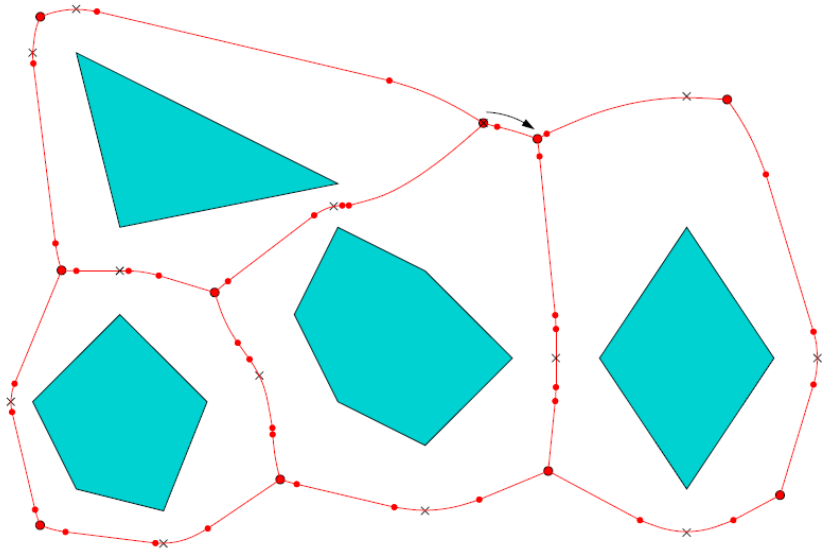
Voroného diagram je geometrická rovinná struktura, která rozděluje rovinu na části nebo území. Tvar diagramu je odvozen z rozmístění tzv. generujících bodů. Rovina je rozdělena takovým způsobem, že každý bod je přiřazen k nejbližšímu generujícímu bodu. Body, které

se nacházejí ve stejné vzdálenosti od více generujících bodů, tvoří Voroného diagram. Jedná se o body, které jsou na rozhraní území přiřazených ke generujícím bodům. Hrany Voroného diagramu tvoří body na rozhraní dvou území. Vrcholy jsou body, které jsou na rozhraní více než dvou oblastí. Při vytvoření grafu jsou mezi uzly zahrnuty také počáteční a cílová pozice robotu. Pohyb po Voroného diagramu odpovídá situaci, kdy robot udržuje od překážek co největší odstup.



Obrázek 2.5: Příklady Voroného diagramů pro body. [26]

Zobecněný Voroného diagram rozšiřuje generující body do rovinných útvarů.

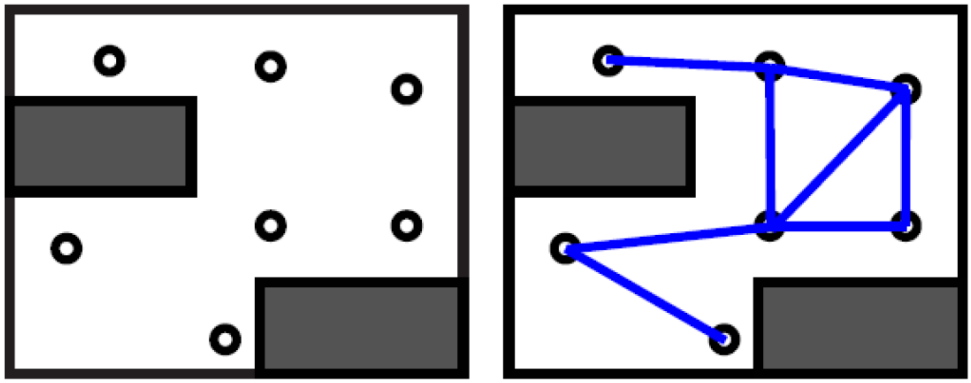


Obrázek 2.6: Zobecněný Voroného diagram pro překážky reprezentované mnohoúhelníky. [27]

2.2.3. Pravděpodobnostní mapy cest

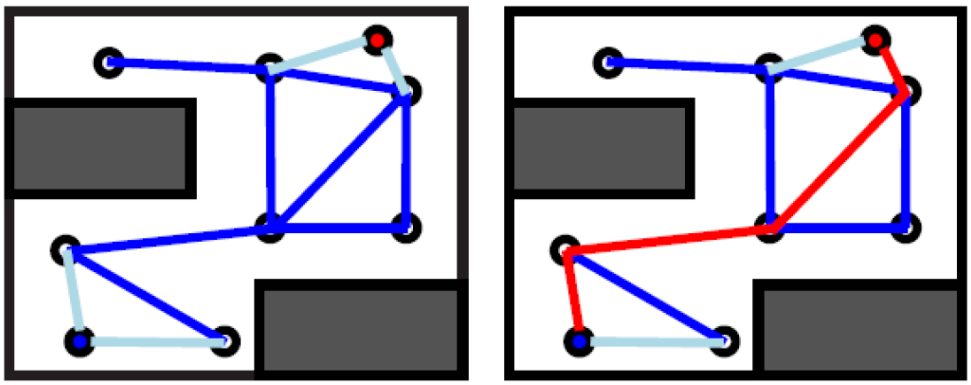
Ve fázi předzpracování se náhodným způsobem generuje mapa cest, tato fáze se označuje jako učící. V druhé fázi, tzv. dotazovací, se hledá cesta ve vygenerované mapě (grafu).

V první fázi se náhodně vybírají konfigurace robotu. U vybraných bodů se testuje, zda se nacházejí ve volném konfiguračním prostoru. Pokud ano, jsou zařazeny mezi vrcholy mapy cest. Následuje hledání přímé bezkolizní spojnice nově vybraného vrcholu k vrcholům dříve vybraným. Pokud je taková spojnice nalezena, je zařazena mezi hrany grafu. Fáze končí, jakmile je dostatečným způsobem popsán volný pracovní prostor, nebo je splněno jiné kritérium ukončení (vyčerpání času, paměti aj.).



Obrázek 2.7: Učící fáze pravděpodobnostních map cest. [27]

Ve druhé fázi je prvním krokem napojení starovní a cílové pozice do grafu. To se provádí stejným způsobem jako přidávání hran v předchozí fázi. V případě, že se napojení nezdaří, je možné zopakovat první fázi. Pokud je napojení do grafu úspěšné, řešením je nalezení cesty v grafu. [27] [28]

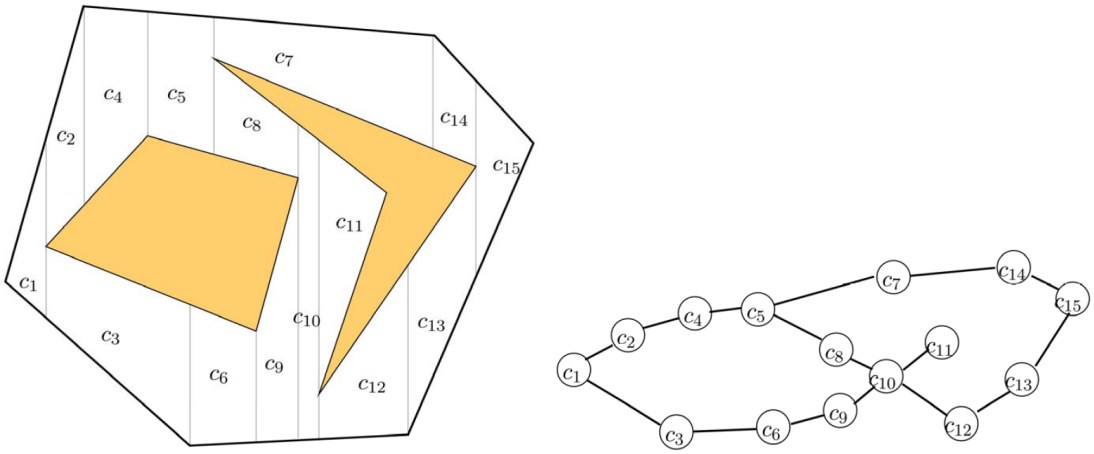


Obrázek 2.8: Dotazovací fáze pravděpodobnostních map cest. [27]

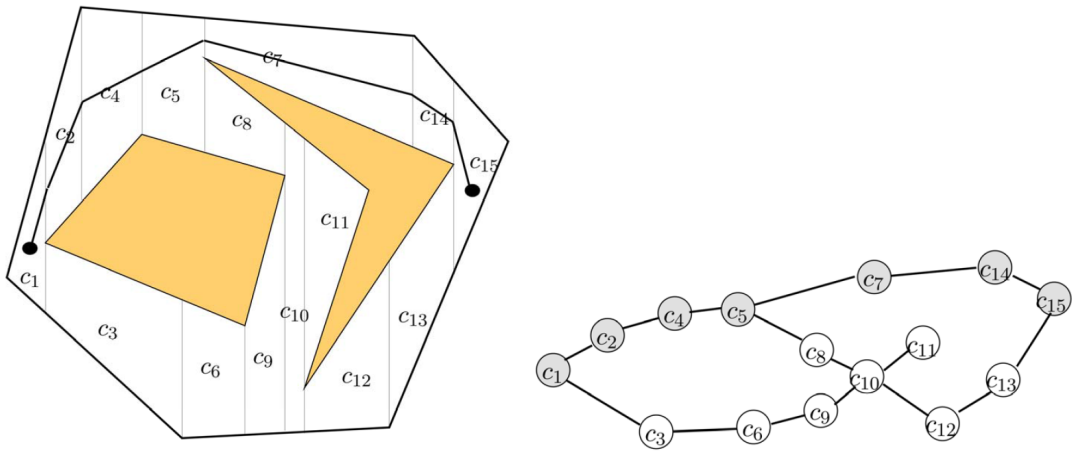
Do kategorie pravděpodobnostních map cest patří také metoda rychle rostoucích náhodných stromů (RRT). Metoda je popsaná v části 2.5.

2.3. Dekompozice do buněk

Základem rozkladu do buněk je rozdělení prostoru na jednoduché části (buňky), jejichž spojitost je poté reprezentována grafem sousednosti buněk. Tvar buněk se liší dle typu metody. Přesný neboli exaktní rozklad využívá buňky nestejného tvaru a velikosti. Tvarem buněk je v tomto případě většinou lichoběžník nebo trojúhelník. Dalším typem rozkladu je přibližný nebo také aproximativní rozklad do buněk. Tvarem buněk aproximativního rozkladu je většinou čtverec. U buněk se určí, zda obsahují či neobsahují překážku. Vrcholy



Obrázek 2.9: Lichoběžníková dekompozice a odpovídající graf sousednosti. [4]



Obrázek 2.10: Cesta rozloženým prostředím a odpovídající cesta grafem. [4]

grafu jsou buňky neobsahující překážky. Vrcholy grafu reprezentující buňky jsou spojeny hranou, pokud odpovídající buňky mají společnou volnou hranici. [33]

Hledání cesty se pak skládá ze dvou kroků: Nejdříve proběhne rozhodnutí, ve kterých buňkách se nachází start a cíl cesty. Následuje nalezení cesty mezi buňkou obsahující robot a buňkou obsahující cíl v grafu spojitosti. [4]

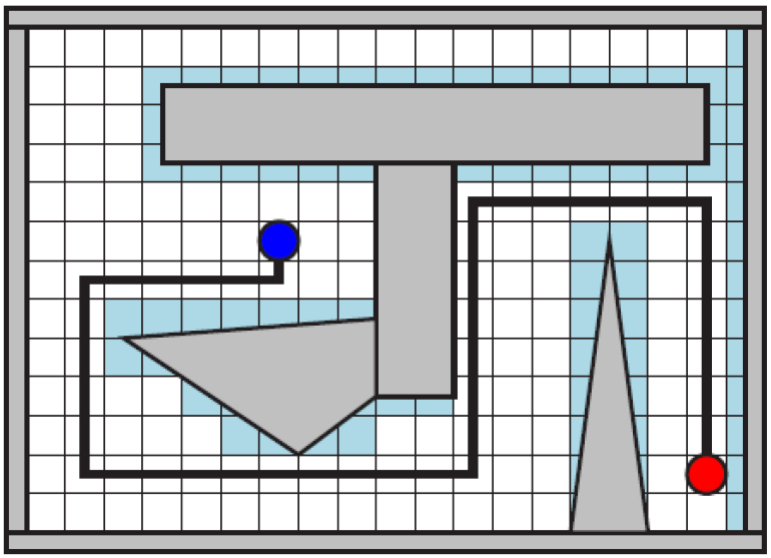
2.3.1. Exaktní rozklad do buněk

Hlavní myšlenkou je rozdělení volného prostoru do nepřekrývajících se buněk, které přesně odpovídají tvaru prostředí. Většinou se používá v prostředích, kde jsou překážky reprezentovány mnohoúhelníky. Buňky mají potom tvar lichoběžníků nebo trojúhelníků. Buňky jsou sousední, pokud sdílí společnou hranici. [4]

2.3.2. Aproximativní rozklad do buněk

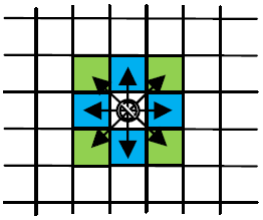
Prostředí je rozděleno do buněk stejného tvaru. Nejčastěji používaným tvarem buněk je čtverec. Jedná se o diskretizaci spojitého prostoru. Přesnost aproximace je dána velikostí buněk: čím jsou buňky menší, tím přesnější je rozklad. Se snižující se velikostí buněk ale narůstá jejich počet, takže se zvyšuje výpočetní i paměťová náročnost. Nepřesnost tohoto

typu rozkladu je dána faktem, že při konečném rozlišení jsou jako nepřístupné označené i ty buňky, které jsou překážkou obsazeny jen částečně.



Obrázek 2.11: Aproximativní rozklad na čtvercové buňky konstantní velikosti. [27]

Velikost buněk nemusí být konstantní. Při variantě rozkladu do kvadrantů se jemnější diskretizace provádí pro tzv. smíšené buňky. To jsou takové, které jsou obsazené překážkou pouze z části. U buněk, které jsou buď zcela obsazeny překážkou, nebo jsou úplně neobsazené, není třeba v rozkladu pokračovat. [25]

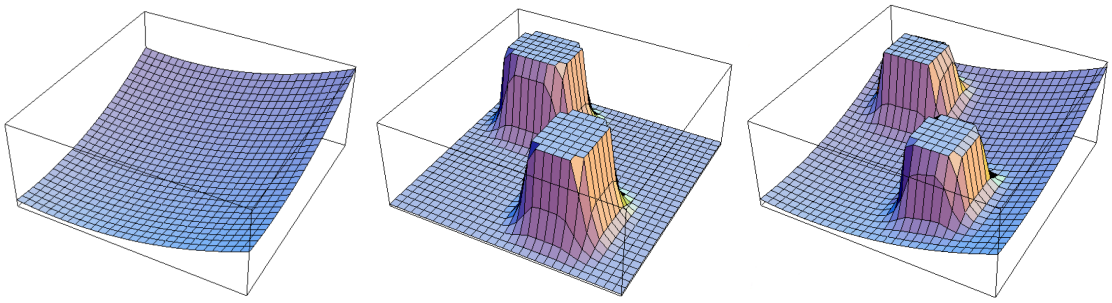


Obrázek 2.12: Možné směry pohybu v prostředí reprezentovaném mřížkou. [28]

Takto rozdělený prostor může být popsán kartézským souřadným systémem, kdy jednotkou délky je velikost strany buňky. Pohyb v tomto diskretizovaném poli buněk je většinou omezen na 8 směrů, případně na 4 směry. V případě 4 povolených směrů není možný diagonální přechod. V případě osmisměrného pohybu se někdy vyskytuje podmínka pro diagonální přechod. Aby byl diagonální přechod umožněn, je nutné, aby byly volné také přiléhající buňky v horizontálním i vertikálním směru. Podmíněný diagonální přechod odpovídá situaci, kdy se uvažuje robot nenulové velikosti. Naopak čistě osmisměrný pohyb odpovídá bodovému robotu.

2.4. Potenciálová pole

Plánování cesty metodou potenciálového pole je založeno na jednoduché úvaze. Základní idea spočívá v tom, že na robot působí pružina, která jej přitahuje k cíli a současně směrem od překážek. Jiné přirovnání je takové, že robot se pohybuje po zakřiveném povrchu, který klesá směrem k cíli cesty. Cíl se nachází v nejnižším bodě povrchu, cesta směrem k překážkám vede naopak vzhůru. Obě zmíněné analogie (pružina a zakřivený povrch) jsou



(a) (b) (c)

Obrázek 2.13: Celková funkce potenciálového pole (c) vzniká složením funkce udávající spád ke globálnímu cíli (a) a funkce označující nárůst polenciálu v okolí překážek. [5]

způsobem uložení potenciální energie. Robot se pohybuje do konfigurace s nižší energetickou hladinou, tj. pohybuje se ve směru největšího úbytku energie (ve směru nejstrmějšího klesání). [5]

Ve fázi předzpracování je definována funkce potenciálového pole pro konfigurační prostor. V okolí překážek se vyskytují maxima funkce, v cíli je globální minimum. Celková funkce potenciálového pole je složena ze dvou funkcí. První funkce popisuje růst potenciálu s narůstající vzdáleností od cíle. Druhá funkce popisuje růst potenciálu s klesající vzdáleností k překážkám. Nevýhodou metody je možnost uvíznutí v lokálním minimu, která musí být řešena jinými přístupy.

2.5. Metoda RRT

Metody založené na vzorkování opouštějí myšlenku explicitně definovného volného prostoru a prostoru obsazeného překážkami. Jediným zdrojem informací o dostupnosti nebo nedostupnosti konfigurace je algoritmus detekující kolizi. Detekční algoritmus je černá skříňka, která zkoumá konfigurační prostor a rozhodne, zda konktrétní konfigurace (případně i její blízké okolí) leží ve volném konfiguračním prostoru.

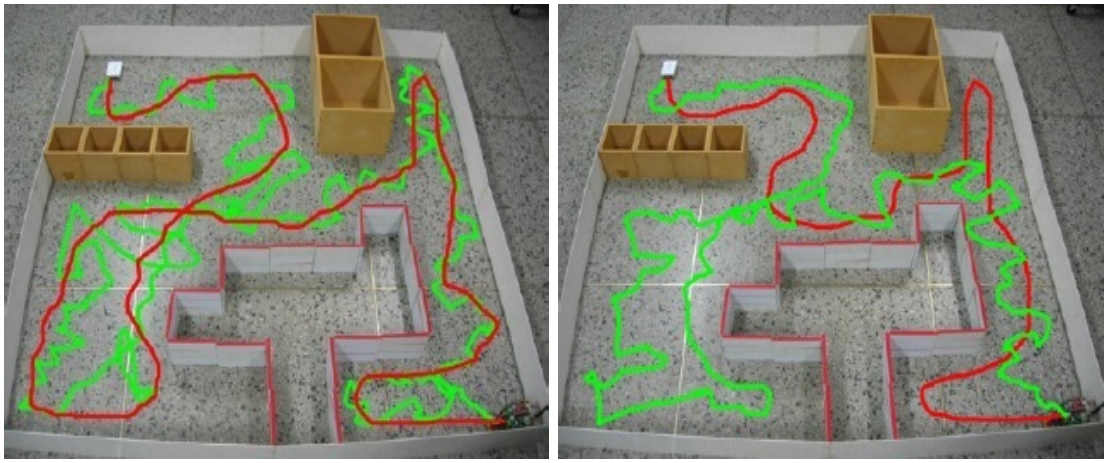
Plánovací algoritmy využívají detekční funkci pro postupný průzkum volného konfiguračního prostoru. Tento způsob plánování připomíná pohyb robotu s malým dosahem senzoru při průzkumu neznámého prostředí.

Příkladem metod založených na vzorkování je metoda rychle rostoucích náhodných stromů (*rapidly exploring random trees* – RRT). Myšlenkou je agresivní průzkum konfiguračního prostoru postupnou expanzí z počáteční konfigurace.

Prozkoumané území je označeno pomocí stromového grafu s kořenem v počáteční konfiguraci. Každá iterace rozšiřuje strom o listové uzly a hrany, které je spojují ke zbytku stromu. Hrany jsou bezkolizními cestami mezi dvěma konfiguracemi. Algoritmy RRT vybírají náhodným způsobem konfiguraci průběžných cílů (z celého konfiguračního prostoru, nikoli pouze z volného). Následně se snaží těchto cílů dosáhnout expandováním nejbližších uzlů ze stromu. Díky volbě průběžných cílů vzniká tendence k prozkoumání neprozkoumaných částí konfiguračního prostoru. Po dostatečném množství iterací algoritmus pokryje volný konfigurační prostor. [16]

Článek [39] se zabývá využitím RRT pro plánování cesty robotu v neznámém prostředí. Oproti základnímu algoritmu RRT uvádí srovnání s modifikací pro omezení velkého počtu generovaných uzlů stromu. Generování uzlů je omezeno následujícím způsobem. V

Obrázek 2.14: V počátečních iteracích RRT rychle pronikne do neprozkoumaných částí (vlevo). Následuje zhuštění vzorkování v dosažených částech (vpravo). [16]



Obrázek 2.15: Srovnání cest robotu se základním algoritmem RRT (zelená) a s omezením generování uzlů směrem k překážkám (červeně). [39]

případě, že strom narazí na překážku, zmenší se rozsah povolených směrů pro generování dalších uzlů. Robot se tak pohybuje v určitém stálejším směru a nepřibližuje se zbytečně k překážce. Výsledkem je omezení doby slepého prohledávání a zmenšení paměťových nároků potřebných pro uchování uzlů stromu.

2.6. Genetické algoritmy

Genetické algoritmy jsou heuristické optimalizační metody, které využívají mechanismy analogické k biologické evoluci. Řešení jsou reprezentována pomocí chromozomů. Chromozom je pole hodnot, které jednoznačně určuje nějaké řešení problému. Jednotlivé prvky chromozomu se nazývají geny. Kvalita daného řešení je vyjádřena hodnotou vhodnosti (*fitness*).

Principem genetického algoritmu je tvorba nové generace řešení z generace předchozí. Pro vytvoření nové generace se používají různé operátory, které jsou aplikovány na předchozí generaci. Klasické operátory jsou selekce, křížení a mutace. Dále se mohou vyskytovat specializované operátory při řešení specifických problémů.

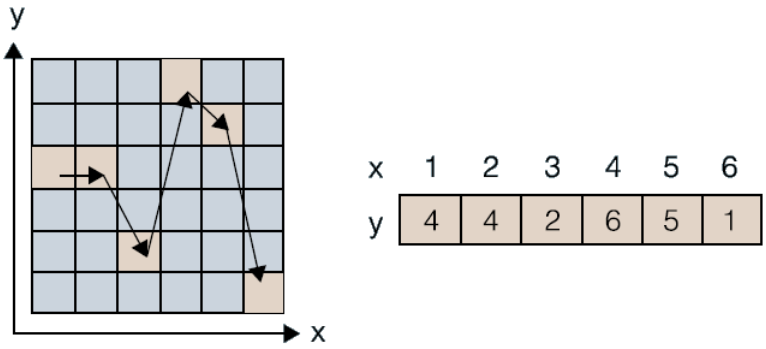
Selekce spočívá ve výběru chromozomů, ze kterých se bude vytvořena následující generace. Výběr se řídí různými způsoby, které zohledňují kvalitu řešení reprezentovaného chromozomem. Příkladem je turnajový výběr, kdy se z populace náhodně vybere několik řešení. Jako rodič je vybráno řešení s nejlepší hodnotou *fitness*.

Křížení kombinuje části (většinou) dvou vybraných rodičů. Opět existuje celá řada způsobů provedení. Biologickou analogií je bodové křížení, kdy se rodičovské chromozomy rozdělí na dvě části. Potomek je potom vytvořen ze dvou částí, jež každá pochází z jiného rodiče.

Mutace spočívá v náhodné změně některých genů nově vzniklého jedince. Mutace může být provedena například tak, že se u nového jedince vždy změní jeden náhodně vybraný gen.

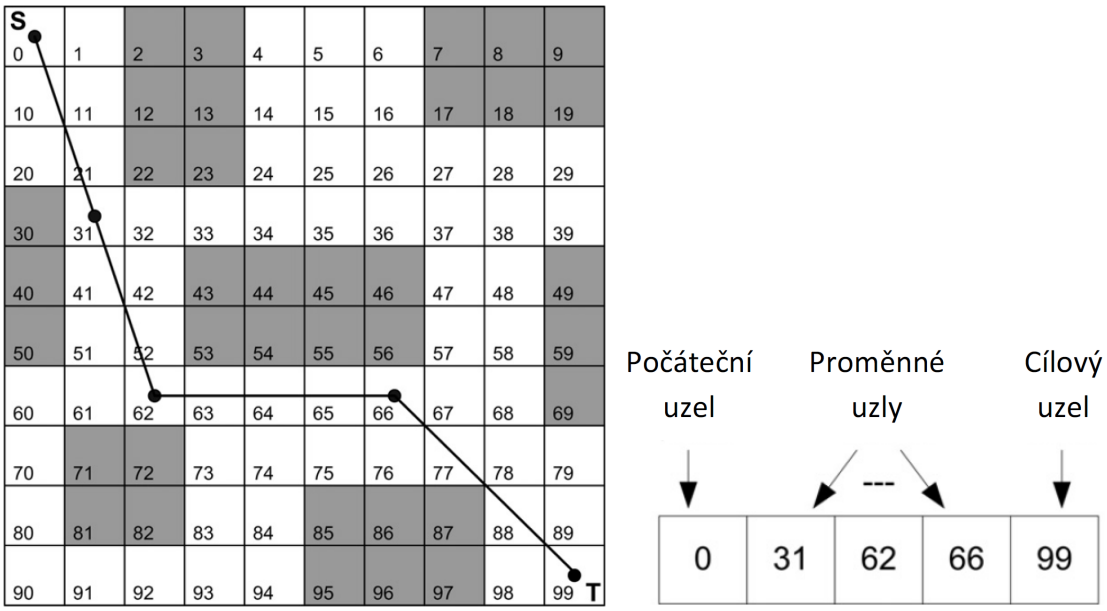
Po vytvoření nové generace nastává výměna populace. Jednou z možností je úplná generační výměna, kdy jsou všechna řešení předchozí generace nahrazena řešeními novými. Tento způsob znamená ztrátu všech řešení z minulé generace, včetně těch nejlepších. Varianty postupné náhrady populace v různé míře uchovávají dobrá řešení z předchozí generace.

Algoritmus začíná vytvořením počáteční generace. Následuje aplikace výše popsaných operátorů na populaci. Po výměně populace se ověří, zda bylo naplněno nějaké kritérium ukončení. Mezi kritéria ukončení patří například dosažení maximálního počtu generací, nalezení dostatečného řešení nebo pouze malé změny kvality řešení v průběhu daného počtu iterací. Výstupem genetického algoritmu je řešení s nejlepší hodnotou *fitness*. [18] [14]



Obrázek 2.16: Příklad reprezentace cesty v mřížce chromozomem. Pozice genu určuje souřadnici x, hodnota genu udává souřadnici y. [18]

Článek [32] představuje nový způsob mutace pro plánování cesty robotu pomocí genetického algoritmu. Metoda spočívá v posunu souřadnic mutovaného uzlu cesty o jedno pole. Z možných osmi posunů je vybrán ten, jehož výsledkem je řešení s nejlepší hodnotou *fitness*.



Obrázek 2.17: Jiný způsob reprezentace cesty mřížkou pomocí chromozomu. [32]

3. Rojová inteligence

Kapitola popisuje základní principy rojové inteligence, přehled metaheuristik, které lze zahrnout do rojové inteligence, a využití některých typů pro plánování cesty robotu.

3.1. Charakteristika metaheuristik

Heuristika je definována jako metoda, která vede k rychlým rozhodnutím s využitím malého množství informací a výpočetního výkonu. [15] Heuristiku je možné chápat jako odhad založený na neúplných znalostech o problému.

Metaheuristika je souhrn principů použitých pro určení heuristických metod, které je možné použít pro řešení širokého pole problémů. Jinými slovy, metaheuristika je hlavní vysokoúrovňový rámec algoritmu, který může být aplikován na různé optimalizační problémy s relativně malým množstvím modifikací nutných pro adaptaci na řešení specifického problému. Příkladem metaheuristik je simulované žíhání, tabu prohledávání, evoluční algoritmy nebo optimalizace mravenčí kolonií. [21]

Metaheuristiky lze rozdělit na dva hlavní typy: První typ je založený na jednom řešení, druhý typ na populaci řešení. Prohledávací proces prvního typu začíná v jednom možném řešení, které je následně iterativně vylepšováno. Příkladem tohoto typu je simulované žíhání. Oproti tomu metaheuristiky založené na populaci řešení začínají s větším množstvím (populací) náhodných počátečních řešení. V průběhu iterací se vylepšují všechna řešení v populaci. Metaheuristiky založené na populaci skýtají několik výhod v porovnání s těmi založenými na jednom řešení: Větší množství řešení umožňuje sdílet informace o prohledávaném prostoru a zaměřit se na jeho perspektivnější části. Populace řešení je méně náchylná k uvíznutí v lokálním optimu. Metaheuristiky založené na populaci mají většinou lepší diverzifikaci. [19]

Jedním ze směrů populačně zaměřených metaheuristik je rojová inteligence. Definice rojové inteligence je podle [2] následující: Rojová inteligence je kolektivní inteligence, která se objevuje u skupin jednoduchých agentů. Inspirací pro rojovou inteligenci jsou většinou přírodní kolonie, hejna, stáda – obecně skupiny živočichů.

3.1.1. Intenzifikace a diverzifikace

Intenzifikace a diverzifikace jsou dvěma klíčovými složkami metaheuristik. Intenzifikace, která se též označuje jako využívání (*exploitation*), používá lokální informace při vyhledávacích procesech pro nalezení lepších řešení. Lokální informací může být derivace hodnotící funkce nebo variace ceny pohybu v prostředí.

Účelem diverzifikace, která se také označuje jako prohledávání (*exploration*), je důkladnější prozkoumání dané oblasti a generování různorodých řešení.

Přílišná intenzifikace vede k rychlejší konvergenci optimalizačního procesu, ale může mít za následek předčasnou konvergenci k lokálnímu optimu, nebo dokonce ke špatnému řešení. Na druhou stranu vysoká míra diverzifikace zvyšuje pravděpodobnost nalezení globálního optima, ale často zpomaluje rychlost konvergence. Z těchto důvodů je nutné najít rovnováhu mezi intenzifikací a diverzifikací nebo využíváním a prohledáváním.

V průběhu hledání je nutné použít vhodný mechanismus nebo kritérium výběru nejlepších řešení. Nejčastěji používaným kritériem je přežití nejschopnějších (*survival of the*

fittest), to znamená aktualizovat a uchovat průběžné nejlepší řešení. Dalším přístupem, který se často využívá, je elitismus, což je předání více nejlepších řešení dalším generacím prohledávání. [38]

3.1.2. Techniky zavedení náhodnosti

V algoritmech inspirovaných jevy v přírodě je možné nalézt znaky náhodnosti. Příkladem může být horolezecký algoritmus s náhodnými startovními pozicemi. Kladem této jednoduché strategie je jak efektivita, tak většinou i jednoduchá implementace.

Složitějším způsobem zavedení náhodnosti je použít ji v různých částech algoritmu a použít různá rozdělení pravděpodobnosti. Příkladem používaných rozdělení jsou rovnoměrné, normální (Gaussovo) rozdělení nebo Lévyho rozdělení. Shrnuto, techniky náhodnosti jsou účinnou a důležitou složkou globálních prohledávacích algoritmů. [38]

3.2. Algoritmy rojové inteligence

Nyní následuje přehled algoritmů, které dle literatury [38] spadají do kategorie rojové inteligence.

3.2.1. Mravenčí algoritmy

Mravenčí algoritmy, zejména optimalizace mravenčí kolonií, napodobují chování skupiny mravenců při hledání potravy. Mravenci používají feromon jako chemický nositel informace. Koncentraci feromonu je možné považovat za indikátor kvality řešení. Tvorba řešení je spojena s koncentrací feromonů, vyhledávací algoritmus často generuje cesty obsahující nejvyšší koncentrace feromonů. Mravenčí algoritmy jsou obzvlášť vhodné pro diskrétní optimalizační problémy, převedené na hledání cesty v grafu. Významnými charakteristikami mravenčích algoritmů jsou: pravděpodobnost výběru cesty, rychlost vypařování feromonu a aktualizace feromonu s ohledem na kvalitu nalezené cesty. Existuje několik způsobů řešení těchto problémů. [38]

Každý mravenec vytváří dílčí řešení – cestu z počátečního uzlu do cílového. Při tvorbě řešení se mravenec rozhoduje, do kterého sousedního uzlu se přesune podle základního vztahu:

$$p_{ij}^{k} = \begin{cases} \frac{\tau_{ij}^{\alpha}}{\sum_{l \in N_i^k} \tau_{il}^{\alpha}}, & \text{pokud } j \in N_i^k; \\ 0, & \text{pokud } j \notin N_i^k; \end{cases} \tag{3.1}$$

kde N_i^k je okolí mravence k, který se nachází v uzlu i. Okolí uzlu i zahrnuje všechny uzly přímo spojené jednou hranou s uzlem i. Některé modifikace do okolí uzlu i nezahrnují předchůdce, ze kterého se mravenec dostal do uzlu i. Řešení je vytvořeno v okamžiku, kdy mravenec dosáhne cílového uzlu. Při tvorbě řešení mohou vznikat v nalezené cestě cykly, které je nutné odstranit.

Následuje úprava hodnoty feromonů. Hodnoty koncentrace feromonů jsou ovlivněny dvěma jevy. Prvním jevem je vypařování feromonu. Míra vypařování feromonu je určena hodnotou konstanty ρ. Po ukončení každé iterace hledání je hodnota feromonu snížena na všech hranách následujícím způsobem:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij}, \tag{3.2}$$

kde $\rho \in (0, 1]$ je vstupní parametr určující míru vypařování.

Druhým jevem, který má vliv na koncentraci feromonů, je nanesení feromonu na hrany, které byly součástí dílčích řešení. Koncentrace feromonu na hranách, které v řešení zůstaly po odstranění cyklů, jsou navýšeny o hodnotu $\Delta\tau^k$:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau^k. \tag{3.3}$$

Důležitým aspektem je volba $\Delta\tau^k$. Nejjednodušším případem je použití stejné, konstantní hodnoty pro všechny mravence. Právě způsobem volby $\Delta\tau^k$ se liší různé typy mravenčích algoritmů. Velmi často je volena v závislosti na délce nalezené cesty, zejména jako nepřímo úměrná vůči délce cesty. V algoritmu *Ant System* je dána výrazem $1/C^k$, kde C^k je délka cesty mravence k.

Koncentraci feromonů je dále možno upravovat pomocí démonových akcí (*daemon action*). Příkladem démonové akce je navýšení hodnoty feromonu na průběžně nejkratší nalezené cestě, která se přenáší a aktualizuje mezi jednotlivými iteracemi hledání. Varianta *Elitist Ant System* navyšuje koncentraci feromonu na hranách nejkratší nalezené cesty po každé iteraci o hodnotu e/C^{bs}, kde C^{bs} je délka průběžně nejkratší cesty a e je parametr definující váhu navýšení feromonu na nejkratší cestě.

Další variantou algoritmu *Ant System* je *Rank-Based Ant System*, kde se hodnoty feromonů navyšují pouze několika nejlepším řešením nalezených v poslední iteraci. Těmto řešením se přidělují váhy dle jejich kvality. Váhy jsou vzestupnou řadou přirozených čísel, kdy váhu 1 má nejhorší z akceptovaných řešení. Navíc se zde využívá stejného principu jako v algoritmu *Elitist Ant System*, tj. navýšení feromonů na průběžně nejlepší nalezené cestě. Průběžná nejkratší nalezená cesta má nejvyšší váhu. [9]

Článek [30] popisuje využití algoritmu *Ant colony system* (ACS) pro globálně optimální plánování pohybu mobilních robotů v reálném čase. Metoda sestává ze tří základních kroků:

1. Vytvoření modelu volného prostoru robotu pomocí teorie grafů MAKLINK.
2. Nalezení suboptimální bezkolizní cesty v modelu volného prostoru pomocí Dijkstrova algoritmu.
3. Nalezení globálně optimální cesty s využitím algoritmu ACS pro optimalizaci subobtimální cesty.

Článek [41] se zabývá popisem algoritmus RNA (*Robot navigation ant algorithm*). Jedná se o mravenčí algoritmus pro navigaci robotu v diskretizovaném, neznámém a dynamickém prostředí. Neznámé prostředí znamená, že robot má omezený dohled. Robot nezná rozložení překážek na celé scéně, pouze v oblasti dohledu. Význačnými rysy algoritmu jsou:

- Určení lokálního cíle v případě, že globální cíl se nachází mimo oblast dohledu robotu.
- Předvídání kolize s dynamickou překážkou a přeplánování cesty tak, aby ke kolizi nedošlo. RNA předpokládá výskyt nejvýše jedné dynamické překážky v oblasti dohledu.

Plánování cesty ve známé oblasti je prováděno algoritmem MSAC (*multi-scout ants cooperation*). MSAC využívá dvě skupiny mravenců, kteří jednotlivě využívají hladovou strategii a navzájem spolupracují sdílením tabu tabulky, čímž omezují opětovné hledání v již prozkoumané oblasti. Algoritmus je podrobně popsán v části 4.1.

Článek [12] popisuje využití algoritmu *Simple ant colony optimization* (SACO) pro plánování cesty robotu. Dále jsou uvedeny úpravy a zjednodušení, které dávají algoritmu další vlastnosti. Prvním vylepšením je možnost při výběru dalšího kroku dát váhu vzdálenosti vybíraného uzlu od cíle. To umožňuje preferovat uzly blíže k cíli. Využití této možnosti je vhodné ve volných scénách bez výskytu lokálního minima. Druhou úpravou je využití paměti pro uchování prohledaných pozic. Několik prvních prozkoumaných pozic je v modelu prostředí dočasně označeno jako nedostupné, což omezuje stagnaci algoritmu. Zvláštností a zjednodušením algoritmu je výskyt feromonu v uzlech místo obvyklé přítomnosti feromonu v hranách grafu scény. Výše zmíněný článek se také zaměřuje na způsob hodnocení kvality nalezené cesty s využitím fuzzy regulátoru a jeho ladění. Podrobný popis (bez využití fuzzy regulátoru) je uveden v části 4.4.

3.2.2. Včelí algoritmy

Algoritmy inspirované včelami jsou více různorodé, některé používají feromon, většina však ne. Téměř všechny včelí algoritmy jsou založené na napodobení hledání potravy. Často se využívají následující charakteristické vlastnosti:

- Vrtivý včelí tanec (*waggle dance*), pomocí kterého si včely navzájem předávají informace o místě výskytu potravy.
- Polarizace světla sloužící jako reference pro směr pohybu.
- Maximalizace výnosu nektaru pro simulaci rozdělení včel hledajících potravu v okolí květin a tím i v prohledávaném prostoru.

Různé varianty včelích algoritmů používají lehce odlišné vlastnosti chování včel. Algoritmy založené na chování včely medonosné (*honeybee*) přiřazují včely hledající potravu k různým zdrojům potravy (květům) tak, aby maximalizovaly celkový výnos nektaru.

Příkladem včelího algoritmu, který využívá feromony, je algoritmus virtuálních včel (*virtual bee algorithm*).

Algoritmus umělé včelí kolonie ABC (*artificial bee colony*) rozděluje včelí kolonii do tří skupin: včely přenášející potravu, pozorovací včely a včelí průzkumnice. Oproti algoritmu *honeybee*, který využívá pouze dva druhy včel (včely přenášející potravu a průzkumné včely), používá algoritmus ABC vyšší specializaci. [38]

Aplikací včelích algoritmů pro plánování cesty robotu se zabývají například články [22] a [7]. První článek popisuje metodu o dvou krocích. V prvním kroku se vytvoří počáteční bezkolizní cesta, v druhém kroku se pomocí kolonie včel původní cesta optimalizuje. Druhý článek charakterizuje využití algoritmu rozmnožování včel (*honey-bee mating algorithm*) pro navádění robotu.

3.2.3. Netopýří algoritmus

Netopýří algoritmus je inspirovaný chováním netopýrů a využíváním echolokace – schopnosti navigovat se vlastním sluchem podle odrazů zvuku jejich pískotu od předmětů. Netopýři používají echolokaci pro zjišťování pozice kořisti, překážek a míst k odpočinku.

Netopýří algoritmus využívá tři idealizovaná pravidla:

1. Všichni netopýři používají echolokaci pro vnímání vzdálenosti, současně poznají rozdíl mezi kořistí a bariérami okolí.

2. Netopýr náhodně poletuje rychlostí v_i v pozici x_i s neměnným rozsahem frekvencí $[f_{min},\ f_{max}]$, proměnlivou mírou emise $r \in [0, 1]$ a hlasitostí A_0 a hledá kořist v závislosti na blízkosti cíle.
3. Přestože hlasitost se může v mnohých ohledech lišit, předpokládá se rozsah od velké kladné hodnoty A_0 do minimální hodnoty A_{min}.

Tato pravidla je možné zapsat pomocí následujících vztahů:

$$f_i = f_{min} + (f_{max} - f_{min})\varepsilon, \quad v_i^{t+1} = v_i^t + (x_i^t - x^*)f_i, \quad x_i^{t+1} = x_i^t + v_i^t \tag{3.4}$$

kde ε je náhodné číslo vybrané pomocí rovnoměrného rozdělení a x^* je průběžné nejlepší nalezené řešení. Hlasitost a míra emise se může v průběhu iterací lišit následovně:

$$A_i^{t+1} = \alpha A_i^t, \quad r_i^t = r_i^0[1 - \mathrm{e}^{-\beta t}] \tag{3.5}$$

kde α a β jsou konstanty. V nejjednodušších případech je možné použít $\alpha = \beta = 0{,}9$. [38]

Aplikací plánování cesty bezpilotního vojenského letounu pomocí netopýřího algoritmu se zabývá článek [34]. Článek také popisuje modifikaci netopýřího algoritmu pomocí mutace.

3.2.4. Optimalizace hejnem částic

Optimalizace hejnem částic je založeno na chování hejna ptáků nebo ryb. Algoritmus prohledává prostor kriteriální funkce pomocí úprav trajektorií jednotlivých agentů, nazývaných částice. Trajektorie je po částech vytvářena pomocí pozičních vektorů kvazistochastickým způsobem.

Pohyb částice sestává ze dvou hlavních složek: stochastické a deterministické. Každá částice je přitahována k pozici současného průběžného globálního optima g^* a k pozici svého průběžného lokálního optima x_i^*. Částice má zároveň tendenci pohybovat se náhodně. Vektor x_i označuje polohu částice, v_i značí její rychlost. Nový vektor rychlosti je dán následujícím vztahem:

$$v_i^{t+1} = v_i^t + \alpha\varepsilon_1[g^* - x_i^t] + \beta\varepsilon_2[x_i^* - x_i^t], \tag{3.6}$$

kde ε_1 a ε_2 jsou dva náhodné vektory, jejich složky mají hodnoty mezi 0 a 1. Alfa a beta jsou parametry učení nebo urychlující konstanty. Typické hodnoty jsou $\alpha \approx \beta \approx 2$.

Počáteční rozmístění částic by mělo být relativně rovnoměrné, aby bylo prozkoumáno co největší území. Počáteční rychlost částice může být nulová, $v_i^{t=0} = 0$. Následující pozice je potom dána vztahem:

$$x_i^{t+1} = x_i^t + v_i^{t+1}. \tag{3.7}$$

Přestože v_i může být jakákoli hodnota, většinou je ohraničená nějakým rozsahem $[0,\ v_{max}]$. Možným vylepšením je zavedení funkce setrvačnosti, která stabilizuje pohyb částic, což má vést k rychlejší konvergenci. [38]

Článek [20] popisuje využití optimalizace hejnem částic pro plánování cesty přes dynamické prostředí. Článek [40] používá optimalizaci hejnem částic pro plánování cesty v nebezpečném a neurčitém prostředí.

3.2.5. Algoritmus světlušek

Algoritmus světlušek napodobuje chování světlušek, které se řídí jasem vydávaného světla jednotlivých mušek. Základem algoritmu jsou tři pravidla:

1. Světlušky jsou jednopohlavní, takže se navzájem přitahují bez ohledu na své pohlaví (ani není rozlišeno).
2. Atraktivita je přímo úměrná jasu, což znamená, že klesá se vzrůstající vzdáleností. Pro jakékoli dvě svítící světlušky platí, že ta s menším jasem se bude pohybovat za tou s větším jasem. Pohyb světlušky s největším jasem je náhodný.
3. Jas světlušky je dán kriteriální funkcí dle okolního prostředí.

Pohyb světlušky i, která je přitahovaná atraktivnější světluškou j (s vyšším jasem) je dán vztahem

$$x_i^{t+1} = x_i^t + \beta_0 \mathrm{e}^{-\gamma r_{ij}^2}(x_j^t - x_i^t) + \alpha \varepsilon_i^t, \tag{3.8}$$

kde β_0 je atraktivita ve vzdálenosti $r = 0$, γ je parametr udávající vliv euklidovské vzdálenosti světlušek r_{ij} na pokles jasu. Třetí člen udává náhodnost pohybu: α je parametr vlivu náhodnosti, ε_i^t je vektor náhodných čísel vybraných pomocí rovnoměrného nebo normálního rozdělení pravděpodobnosti v čase t. Pokud $\beta_0 = 0$, pohyb je zcela náhodný. [38]

Plánováním cesty s využitím algoritmu světlušek se zabývá článek [17], návrh obsahuje popis adaptivní změny parametrů v průběhu hledání. O plánování cesty bezpilotního vojenského letounu pomocí algoritmu světlušek pojednává článek [35]. Cesta, kterou algoritmus generuje, zohledňuje nebezpečná místa s různou známou mírou rizika.

3.2.6. Vlčí algoritmy

Poměrně novým typem rojové inteligence jsou algoritmy napodobující chování vlků. Algoritmus vlčího hledání (*Wolf Search Algorithm*) napodobuje způsob, jakým vlci pátrají po potravě a jak se vyhýbají predátorům. Základem algoritmu jsou tři pravidla:

1. Oblast dohledu vlka je omezena poloměrem v. Vlk ví pouze o těch společnících, kteří jsou v oblasti dohledu. Velikost kroku s, se kterou se vlk pohybuje, je většinou menší než dohled.
2. Kvalitu řešení měří kriteriální funkce, jejíž hodnota závisí na pozici vlka. Vlk se vždy snaží dostat na lepší území, ale pokud má možnost připojit se k jinému vlkovi, byť na horším území, tak to udělá. Pokud je v dohledu více vlků, připojí se k tomu, který je na lepším území.
3. Je možné, že v některou chvíli vlk vycítí přítomnost nepřítele. V tom případě uteče na náhodnou pozici mimo oblast dohledu. [31]

Dalším vlčím algoritmem je optimalizace vlka obecného (*Grey wolf optimizer*). Algoritmus napodobuje hierarchickou strukturu vlčí smečky, která rozlišuje vlky alfa, beta, delta a omega. Tři nejlepší řešení jsou v tomto pořadí alfa, beta a delta. Všechna ostatní řešení mají označení omega. Vlci omega následují vlky alfa, beta a delta, kteří řídí lov. Mezi další prvky, které algoritmus napodobuje, patří hledání kořisti, obklíčení kořisti a útok na kořist. [19]

3.2.7. Další algoritmy

Existuje mnoho jiných algoritmů rojové inteligence, například umělý imunitní systém, kukaččí hledání, algoritmus opylení květu, algoritmus roje krilu. Účinnost výše uvedených algoritmů je možné přičíst faktu, že jsou založeny na imitaci úspěšných jevů vyskytujících se v přírodě, které se vyvíjely miliony let a obstály v evolučním výběru.

4. Návrh systému pro plánování cesty robotu

Kapitola popisuje návrh systému pro plánování cesty. Byla implementována dvě řešení pro hledání cesty ve známém prostření a dále řešení pro simulaci pohybu v neznámém dynamickém prostředí. Návrh vychází z článků [41] a [12] a z literatury [9]. Cílem bylo implementovat níže uvedené algoritmy a navrhnout možná vylepšení.

4.1. Algoritmus RNA

Robot se pohybuje po dvourozměrném diskretizovaném poli. Pole obsahuje konečný počet statických a dynamických překážek a je popsáno kartézským souřadným systémem se středem v počátečním bodu robotu. Jednotkou souřadného systému je délka kroku robotu Δ. Pole je tak rozděleno do mřížky, jejíž dílky mají velikost $\Delta \times \Delta$. Oblast dohledu je omezena velikostí dosahu senzorů robotu r. Robot nezná počáteční rozmístění statických ani dynamických překážek.

Algoritmus bere v úvahu následující předpoklady: Přestože ve skutečnosti je oblast dosahu senzorů robotu kruhová, používá se čtvercový tvar, což napomáhá vyobrazení a použití diskretizovaného dvourozměrného pole. Předpokládá se, že vzdálenost mezi libovolnými dvěma dynamickými překážkami je větší než $2r$. To znamená, že během kterékoli plánovací fáze se v oblasti dohledu vyskytuje nejvýše jedna dynamická překážka. S tímto omezením počítá lokální plánování cesty při detekci dynamické překážky. Dalším předpokladem je, že robot se pohybuje rychlostí o konstantní velikosti. Trasy a rychlosti dynamických překážek nejsou robotu známy. Při předvídání kolize se uvažuje konstantní rychlost a směr pohybu dynamických překážek. Pokud se dynamická překážka nepohybuje předvídaným způsobem, metoda tuto skutečnost zohlední v dalších krocích, protože plánování cesty robotu je přepočítáváno po každém vykonaném kroku. Předpokládá se, že počáteční a cílová pozice robotu není obsazená statickou překážkou.

Globální navigační algoritmus sestává z několika částí: První část řeší případ, kdy globální cíl leží vně oblasti dohledu robotu. Je tedy nutné převést globální cíl na pole mřížky poblíž oblasti dohledu tak, aby bylo možno jej použít jako prozatímní lokální cíl navigace. Další část rozebírá případ, kdy se v dohledu robotu vyskytuje dynamická překážka. Robot provede odhad, zda může nastat kolize, a případně rozhodne, jak jí předejít. Je důležité zopakovat, že tento navigační algoritmus je opakován po každém vykonaném kroku robotu.

4.1.1. Metoda pro volbu lokálního podcíle

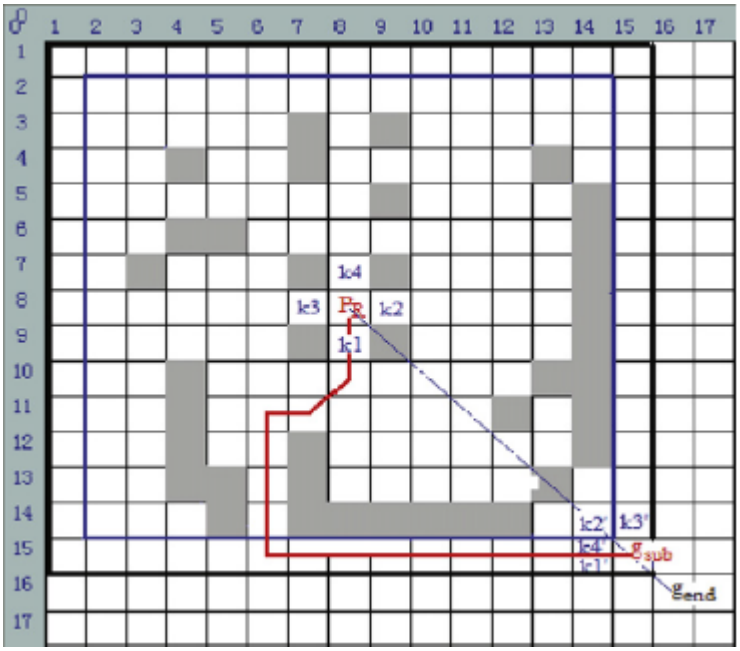
Nejprve robot získá informace o svém okolním prostředí v oblasti dohledu a převede je do diskrétního modelu mřížky relativně vůči své současné pozici, která je umístěna v prostředním bodu mřížky. Velikost polí mřížky je daná délkou kroku robotu.

Dále robot plánuje lokální cestu přes oblast dohledu pomocí algoritmu MSAC (popsaného v sekci 4.1.4). Protože prostředí, v němž se robot pohybuje, je zpravidla větší než oblast dohledu, globální cíl v ní nemusí ležet. V tomto případě je nezbytné určit lokální cíl pro plánování cesty a poté algoritmem MSAC cestu do tohoto cíle.

Metoda určení lokálního cíle je následující. Z polí, která se nacházejí těsně za hranicí dohledu robota, se vybere takové, jež má nejkratší vzdálenost ke globálnímu cíli.

Výběr lokálního cíle z polí, která leží těsně za hranicí oblasti dohledu, zajišťuje, že robot nalezne vhodnou cestu, i pokud jsou pole za hranicí oblasti dohledu obsazena překážkami. Ve skutečnosti by pole vně oblasti dohledu, včetně lokálního cíle, mohla být obsazena překážkami. Avšak aby bylo algoritmu MSAC umožněno nalezení cesty, metoda předpokládá, že pole obsazená nejsou.

Protože se nová cesta přepočítává po každém vykonaném kroku robotu, přítomnost překážek v polích poblíž lokálního cíle je vyřešena později s pohybem robotu, jakmile jsou překážky detekovány. S použitím výše zmíněných předpokladů má algoritmus MSAC možnost vždy nalézt prozatímní cestu k lokálnímu cíli bez ohledu na počet překážek, které ve skutečnosti leží mimo oblast dohledu. A robot tak může vždy vykonat první krok po nalezené cestě.



Obrázek 4.1: Určení lokálního cíle. [41]

Obrázek 4.1 schématicky zobrazuje určení lokálního cíle g_{sub} odvozeného z pozice robotu P_R a pozice globálního cíle g_{end}. Čtverec s černým tučným okrajem označuje oblast dohledu rozšířenou o jedno pole v každém směru. Samotná oblast dohledu je označena modrou čarou. Dle předpokladů v této rozšířené oblasti nejsou překážky. Červená čára značí cestu vygenerovanou lokálním navigačním algoritmem.

4.1.2. Strategie předpovědi kolizí a jejich předcházení

Robot zkoumá okolní prostředí svými senzory v průběhu každého kroku. Předpokládejme, že robot detekuje novou překážku. Nejprve ji považuje za neurčitou a naplánuje cestu s předpokladem, že překážka je nepohyblivá. Po uplynutí krátkého časového okamžiku je pozice překážky detekována znovu a robot může posoudit, zda se jedná o statickou nebo dynamickou překážku. Pokud je překážka dynamická, robot zkontroluje možnost kolize.

Z rozdílu dvou pozic dynamické překážky robot určí velikost a směr její rychlosti. Předpokládá se přímý směr pohybu a konstantní rychlost překážky. Metoda předpovědi

a zabránění kolize je založena na základě vztahu mezi plánovanou cestou robotu a předpokládanou dráhou překážky. Vztah je vyhodnocen jako jeden z následujících případů:

1. Cesta robotu a dráha překážky nemají žádný společný bod.
2. Dráhy se protínají v jednom bodě.
3. V některé části plánované cesty se robot a dynamická překážka pohybují stejným směrem a jejich trasy se zcela nebo částečně překrývají.
4. V některé části plánované cesty se robot a dynamická překážka pohybují opačným směrem a jejich trasy se zcela nebo částečně překrývají.

Pro rozhodnutí, který z případů nastává, robot vypočítá svou předpokládanou pozici na plánované cestě po každém kroku a odpovídající pozice dynamické překážky podél její lineární cesty. Čas dosažení těchto bodů je určen z podílu drah a odpovídajících rychlostí. Podle vztahu množin bodů obsazených robotem a bodů obsazených překážkou je určen jeden z výše uvedených případů a jemu odpovídající strategie odvrácení kolize.

Pokud množiny neobsahují žádný společný bod, kolize nehrozí a robot může vykonat první krok po plánované cestě.

Pokud množiny obsahují právě jeden společný bod, nastává druhý případ. Nyní se vyhodnotí, v jakém předpokládaném čase se ve společném bodě bude vyskytovat robot a překážka. Pokud je absolutní hodnota rozdílu časů větší než doba potřebná pro vykonání jednoho kroku, ke kolizi nedojde a robot může pokračovat v původně plánované trase. V opačném případě robot zůstane na svém místě po dobu potřebnou pro vykonání jednoho kroku a poté může pokračovat v naplánované trase.

V případě, že množiny obsahují více společných bodů, nastává jeden z případů 3 nebo 4. Opět je vyhodnoceno, v jakém čase se ve všech společných bodech drah bude vyskytovat jak robot, tak překážka. Pokud se v některém bodě čas výskytu robotu a čas výskytu překážky rovnají, je tento bod označen jako P_0. Pro případy 3 a 4 je pole P_0 dočasně označeno jako pro robot zakázaná pozice, tj. jako statická překážka. Poté se pomocí algoritmu MSAC nalezne nová bezkolizní cesta a robot se může pohybovat podle nové cesty bez nebezpečí kolize.

Mezi nevýhody výše uvedeného přístupu patří: 1) Omezení na jednu dynamickou překážku. 2) Vzorkování pozice překážky pouze v době dokončení kroku robotem. 3) Nevyřešený případ, kdy se dynamická překážka v době vzorkování nenachází přesně v diskretizovaném poli.

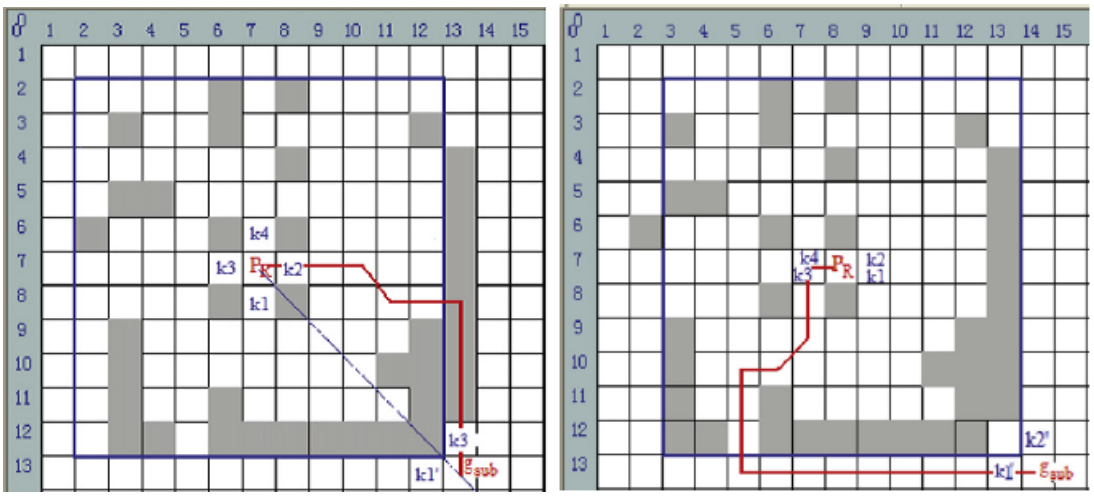
Druhá nevýhoda se projeví v případě, kdy má překážka více než dvojnásobně vyšší rychlost než robot. V tom případě může vzorkování pro krok robotu způsobit, že pozice výrazně rychlejší překážky, přestože se dráhy kříží, nemusí být odhadnuta na pozici robota.

4.1.3. Kroky globálního navigačního algoritmu

Kroky globálního algoritmu RNA jsou následující:

Krok 1: Robot rozdělí oblast dohledu na jednotlivá pole mřížky souřadného systému a vypočítá souřadnice těchto polí relativně ke své současné pozici.

Krok 2: Pomocí metody popsané v části 4.1.1 se jako lokální navigační cíl určí pole vně oblasti dohledu robotu, které má vůči globálnímu cíli nejmenší vzdálenost.



(a) Plán cesty pro oblast dohledu. (b) Nová oblast dohledu po 1 kroku.

Obrázek 4.2: Oscilace pohybu robotu. [41]

Krok 3: Robot použije svůj senzor pro zjištění informací o překážkách uvnitř současné oblasti dohledu. Pokud jsou detekovány nějaké překážky, jsou určeny jejich souřadnice a jsou klasifikovány jako statické nebo dynamické.

Krok 4: Robot pomocí algoritmu MSAC naplánuje lokální cestu ze své současné pozice do lokálního cíle bez úvahy pohybu dynamických překážek.

Krok 5: Robot detekuje, zda se v oblasti dohledu vyskytuje pohybující se překážka. Pokud ne, následuje krok 7. V opačném případě robot analyzuje rychlost a směr pohybu překážky vzhledem k naplánované cestě. Poté použije strategii pro předejití kolize popsanou v části 4.1.2 a rozhodne, který případ nastal. Pokud nastane případ 1, následuje krok 7. Pokud je případ 2 vyhodnocen jako hrozící kolize, pak krok 7 následuje po vykonání čekací strategie. Pro případy 3 a 4 je určen bod kolize P_0.

Krok 6: Robot označí bod kolize P_0 jako statickou překážku a poté znovu naplánuje lokální bezkolizní navigační cestu.

Krok 7: Pokud je globální cíl detekován v oblasti dohledu, pak robot pokračuje v cestě ke globálnímu cíli po plánované cestě, přičemž prověřuje okolí kvůli nově objeveným dynamickým překážkám, kterým by bylo nutné se vyhnout. V opačném případě následuje krok 1 po přesunu do následujícího bodu plánované lokální navigační cesty.

Cesta je přeplánována v průběhu každého kroku pohybu robotu, takže za speciálních podmínek může nastat oscilace.

Obrázek 4.2 znázorňuje ukázku situace, která by bez opatření vedla ke vzniku oscilace v pohybu robotu. Lokální cíl je označen g_{sub}, pozice robotu je označena P_R, plánovaná lokální cesta je označena červenou čarou. Oblast dohledu je ohraničena modrou čarou. Výchozí situace je zobrazena na obrázku 4.2a. Po vykonání prvního kroku doprava po nalezené cestě se robot dostává do pozice na obrázku 4.2b. Nyní je vidět, že první krok nově nalezené cesty vede zpět na pozici zobrazenou na obrázku 4.2a. Robot se tedy bude pohybovat zpět, pokud je předchozí cesta nedostupná (přehrazená nově detekovanou překážkou) nebo delší než jiná v nové oblasti dohledu. To znamená, že nastává oscilace.

V těchto případech nepřináší nově nalezená cesta mnoho užitku. Z toho důvodu jsou ty pozice, ze kterých se robot pohybuje zpět, označeny jako dočasně zakázané, tj. označeny jako statické překážky.

Článek [41] uvádí, že tato strategie obecně řeší problém oscilace. Experimentálně bylo zjištěno, že problém bodové oscilace, podobný situaci na obrázku 4.2, je skutečně vyřešen. Ale při řešení problému lokálního minima, kdy souvislá překážka přehrazuje cestu k lokálnímu cíli a kdy celá překážka není v dohledu robota, oscilace stále vznikají. Nevznikají na rozhraní dvou bodů, ale na rozhraní oblastí, kde robot po vykonání kroku od lokálního cíle ztratí část překážky z dohledu a následuje krok přímo k lokálnímu cíli. Tímto způsobem se robot pohybuje tam a zpět. Při každém návratu je obsazený bod označen dočasnou překážkou. Robot pak pomocí dočasných překážek musí přehradit rozhraní oblastí, aby nalezl správnou cestu.

4.1.4. Algoritmus MSAC

MSAC (*multi-scout ants cooperation*) je algoritmus pro hledání cesty ve známém statickém prostředí. Stručně se dá popsat jako multiagentový systém, který využívá současné dopředné a zpětné vyhledávání, agenti využívají znalosti ze sdílené tabu tabulky a jednotlivě využívají hladovou strategii.

Při hledání spolupracují dvě skupiny mravenčích průzkumníků, které hledají z opačných konců cesty. Každý mravenec vytváří cestu tak, že jako další krok vybere pole ze současného okolí, které je nejbližší k cílovému bodu. Podmínkou je, aby toto pole nebylo dříve navštíveno jiným mravencem. Množina polí navštívených kterýmkoli mravencem se ukládá do tabu tabulky. Mravenci spolupracují sdílením této globální tabulky, čímž minimalizují opakované prohledávání již prozkoumaného prostoru. Pokud by po vyloučení zakázaných polí z tabu tabulky mravenec neměl žádnou možnost pohybu, může se přesunout i na pole, které již navštívil jiný mravenec. V tomto případě se pole vybírá náhodně, náhodný výběr zvyšuje diverzifikaci hledání.

Když mravenec zjistí, že vstoupil do slepé uličky, začne se z ní vracet. Souběžně s návratem jsou pole, která jsou součástí slepé uličky, označena jako nepřístupná. Tímto způsobem se zamezí zbytečné opětovné prohledávání neperspektivního prostoru dalšími mravenci.

Algoritmus je pro obě skupiny identický s výjimkou jejich startovního a cílového bodu. Bude popsán algoritmus pro první skupinu, která provádí dopředné hledání. Hlavní vnější smyčka, která je tvořena kroky 2–6, nalezne jednu nebo více cest mezi startovním a cílovým bodem. Nalezené cesty nemusí být nejkratšími. Z toho důvodu se vnější smyčka několikrát opakuje, nejvýše však do hodnoty voleného parametru MAX. Uvnitř vnější smyčky se provádí kroky 3–5 pro každého mravence. Kroky určují výběr následujícího pole pro přesun mravence. Proces je ukončen, buď když mravenec z jedné skupiny narazí na cesty mravence z druhé skupiny, nebo když počet kroků mravenců překročí parametr S_{max}. Typická hodnota parametru MAX je 3 nebo 4, doporučená hodnota S_{max} je $N \times N/2$, kde N je počet polí v jedné řadě oblasti dohledu robotu (řádku nebo sloupci, větší z obou hodnot). Počet mravenců v každé skupině je označen m.

Kroky algoritmu MSAC jsou následující:

Krok 1: Nastav počitadlo iterace hledání potravy $n = 0$ a počitadlo kroků $S = 0$. Nastav délku průběžné doposud nalezené nejkratší cesty $L_d = \infty$. Vytvoř tabulku cest pro každého mravence a sdílenou tabu tabulku a nastav je všechny jako prázdné.

Krok 2: Umísti m mravenců na startovní bod, přidej startovní bod do tabu tabulky a také do cest všech mravenců. Nastav $k = 1$; k roste vzestupně přes všechny mravence této skupiny.

Krok 3: Označ současnou pozici mravence k jako g_k. Jsou nalezeny dvě následující množiny:

1) Množina $W(g_k)$ sousedních polí, která nejsou obsazena překážkami.
2) Množina $Z(g_k)$ sousedních polí, která nejsou obsazena překážkami a současně zatím nebyla prozkoumána žádným mravencem.

Krok 4: Vyber pole g_{next}, kam se přesune mravenec k. Jsou tři případy:

Případ 1: Pokud $|Z(g_k)| >= 1$, znamená to, že existuje alespoň jedno pole, které není obsazené překážkou a současně zatím nebylo prozkoumáno žádným mravencem. Z množiny $Z(g_k)$ vyber jako g_{next} takové pole, které které minimalizuje vzdálenost zkoumaného pole a cílového bodu hledání.

Případ 2: Když $|Z(g_k)| = 0$ a $|W(g_k)| > 1$, znamená to, že všechna okolní neobsazená pole již byla prozkoumána a nemůže být vybráno žádné nové neprozkoumané pole. Avšak existuje více okolních prozkoumaných polí. Pole g_{next} je vybráno náhodně z již prozkoumaných polí.

Případ 3: Pokud $|Z(g_k)| = 0$ a $|W(g_k)| = 1$, znamená to, že mravenec vstoupil do slepé uličky. Současná pozice mravence k g_k je označena jako dočasná statická překážka a následující pozice mravence g_{next} je určena jedinou volnou pozicí v okolí.

Krok 5: Přesuň mravence k do vybrané pozice g_{next} a přidej g_{next} do tabu tabulky a do cesty mravence k. Když $k < m$, pak nastav $k = k + 1$ a pokračuj na krok 3.

Krok 6: Poté, co byli přesunuti všichni mravenci v obou skupinách, nastav $S = S+1$ a zkontroluj, zda se setkali mravenci z rozdílných skupin. Za setkání se považuje stav, kdy se v okolí mravence z jedné skupiny nachází alespoň jeden bod z cesty mravence druhé skupiny.

Pokud se setkali 2 mravenci z odlišných skupin, vytvoř nalezenou cestu spojením cest mravenců a urči délku L nalezené cesty. Porovnej L s průběžnou nejkratší nalezenou délkou L_d. Když $L < L_d$, dosaď $L_d = L$ a ulož nově nalezenou cestu jako průběžně nejkratší cestu a pokračuj na krok 7. Jinak pokud $S < S_{max}$, nastav $k = 1$ a pokračuj na krok 3.

Krok 7: Nastav $n = n + 1$. Když $n >= MAX$, plánování je u konce. Poslední průběžná uložená cesta je plánovaná cesta. Když $n < MAX$, vymaž tabu tabulku a tabulku cest pro všechny mravence a pokračuj krokem 2.

Algoritmus využívá znalost o vzdálenosti zkoumaných bodů od cíle jako heuristickou funkci (krok 4, případ 1) za účelem rychlejší konvergence. Algoritmus také využívá prvek náhody (krok 4, případ 2) pro zvýšení diverzifikace hledání a snížení pravděpodobnosti uvíznutí v lokálním optimu. Neobvyklou vlastností mravenčího algoritmu je absence feromonu, čímž odpadá nutnost aktualizovat jeho hodnoty. Důsledkem strategie návratu

ze slepých uliček a jejich označení jako nepřístupných je zabránění zbytečného prohledávání neužitečného prostoru. Využití sdílené tabu tabulky zmenšuje četnost opakovaného duplicitního hledání. Všechny tyto metody výrazně zlepšují konvergenci algoritmu.

4.2. Navržené úpravy pro algoritmus RNA

4.2.1. Více dynamických překážek v oblasti dohledu

Algoritmus RNA byl upraven pro zohlednění více dynamických překážek v oblasti dohledu. Plánování s ohledem na dynamické překážky bylo změněno. Kroky 5 a 6 algoritmu RNA byly nahrazeny následujícím postupem.

Pokud se v dohledu robotu vyskytuje dynamická překážka, robot zná její rychlost i směr. Ve skutečnosti je rychlost robotu získávána na základě rozdílů polohy z alespoň dvou měření. Tento proces získávání informace není simulován, jednoduše se předpokládá, že robot zná polohu překážky, velikost její rychlosti i směr pohybu.

V případě detekce dynamické překážky jsou jednotlivé body nalezené nejkratší cesty ohodnoceny časem výskytu robotu. Při kontrole se jako interval nutný pro průchod bere ohodnocení předcházejícího uzlu a uzlu následujícího. To je odrazem skutečnosti, že robot potřebuje mít uzel uvolněn, již když na něj vstupuje. Stejně tak je nutné, aby na něj dynamická překážka nevstoupila dříve, než bude dosaženo uzlu následujícího. Podobně při přechodu v diagonálním směru je nutné, aby pro přechod byly volné i odpovídající hlavní směry. Pokud osm možných směrů označíme jako růžici kompasu, pak pro možný průchod na severovýchod je nutné, aby byly volné tři sousední uzly, a to severní, východní a severovýchodní.

Potom, co je plánovaná cesta robotu časově ohodnocena, následuje predikce pohybu dynamických překážek. Predikce pohybu dynamických překážek považuje směr i velikost rychlosti za konstantní. Robot provádí interní simulaci pohybu překážek. Pro každou dynamickou překážku se testují body, po nichž dle predikce přejde, a tyto body jsou průběžně ohodnoceny časem, podobně jako cesta robotu. V případě, že se časové intervaly označující výskyt robotu a překážky překrývají v některém bodě, znamená to, že je předpovězena kolize. Takový bod je označen jako bod kolize. Postupně se prozkoumají dráhy všech dynamických překážek v dohledu. Pokud při kontrole nebyl zjištěn žádný kolizní bod, robot provede první krok z původně plánované cesty. V opačném případě robot interně dočasně označí všechny body kolize jako statickou překážku. Následně cestu přeplánuje pomocí algoritmu MSAC a opět testuje, zda nedojde ke kolizi. Tento proces přeplánování a hledání bodů kolize se opakuje tolikrát, dokud robot nenajde bezkolizní cestu, nebo dokud není překročen hraniční počet kontrol, který je parametrem algoritmu. Může se stát, že robot nenalezne bezkolizní cestu, ať už z důvodu překročení maximálního počtu přeplánování, nebo kvůli situaci, kdy robot nemá prostor k úniku. Pokud nastane tato situace, robot se zastaví a čeká po dobu jedné časové jednotky. Dále pokračuje stejným způsobem, jako kdyby robot vykonal krok, a to krokem 1 algoritmu RNA.

4.2.2. Řešení případu nenalezení cesty přes statické prostředí

Podobným způsobem byl upraven také krok 4. Původní algoritmus nepočítá s případem, že by robot nenašel cestu přes statické překážky. Tento případ byl ošetřen následovně. V případě, že se v oblasti dohledu vyskytují dočasné překážky přidané z důvodu zabránění

oscilace, je poslední takto umístěná překážka odebrána. (Jedná se pouze o interní úpravu, nejde o zásah do okolního prostředí.) Ať už byla dočasná překážka odstraněna, nebo nebyla, následuje nové volání algoritmu MSAC. V tomto speciálním případě je maximální počet kroků, které mohou mravenci vykonat, navýšen o jednu polovinu oproti obvyklé doporučené hodnotě. To pro případ, kdyby bylo nenalezení cesty způsobené právě omezeným dosahem mravenců. Pokud ani po těchto úpravách nedojde k nalezení cesty, robot začne čekat. Po uplynutí časové jednotky zkusí znovu vyhledat cestu. Při každém čekání se navyšuje hodnota udávající počet za sebou následujících čekání. Pokud tato hodnota převýší hranici zadanou jako parametr, simulace končí se zprávou, že po stanovenou dobu nebyla nalezena cesta. Pro úplnost je vhodné poznamenat, že pokud robot čeká z důvodu nenalezení bezkolizní cesty s ohledem na předpokládaný pohyb dynamických překážek, není tento čas započítán do doby čekání vůči stanovené hranici.

Dále byl upraven krok 7 algoritmu RNA. Není ošetřen případ, kdy nalezená bezkolizní cesta vede přes rozšířenou oblast dohledu. Pokud by tomu tak bylo, robot by při následování této cesty mohl narazit do překážky. Jedná se o případ, kdy cíl je sice v oblasti dohledu, robot je však od něj oddělen překážkou, která by přesahovala i do rozšířené oblasti dohledu. Případ je vyřešen tak, že po vykonání kroku robotu (pohybu) následuje krok 1 algoritmu, dokud není dosaženo globálního cíle.

4.2.3. Popis chování dynamických překážek

Pro simulaci pohybu v dynamickém prostředí bylo nutné navrhnout chování dynamických překážek. Velikost dynamických překážek je totožná s velikostí robotu, což je 1 pole v diskretizovaném prostoru. Překážky se pohybují v přímém směru konstantní rychlostí. V případě, že by dynamická překážka měla v příštím kroku narazit do statické překážky, do robotu, případně na okraj prostředí, náhodně změní směr svého pohybu. Pokud se kříží dráhy dynamických překážek, přednost má překážka s nižším identifikačním číslem. Překážka s vyšším číslem změní svůj směr v případě, že se měly dráhy křížit. Každá dynamická překážka je na počátku simulace definována svou pozicí, směrem a velikostí rychlosti. Zobrazení simulace je vzorkováno v čase, kdy robot dokončí přechod do další pozice v mřížce prostředí. Protože rychlost překážek a robotu je obecně jiná, nastávají případy, kdy překážka není v době zobrazení přesně na pozici definované celočíselnými souřadnicemi. V tomto případě jsou takovou překážkou obsazeny dvě nebo čtyři pole prostředí. Dvě pole jsou obsazena, pokud jedna souřadnice není celočíselná, čtyři pole jsou obsazena, pokud ani jedna ze souřadnic není celočíselná.

Pohyb dynamických překážek je vykonáván postupnými kroky o jedno pole. Dráha, kterou překážka urazí mezi zobrazeními, závisí na délce plánovaného kroku robotu. Rychlost robotu je brána jako jednotková. Časový interval, který uplyne mezi zobrazeními, a tedy i vzorkováním, je určen směrem pohybu robotu. V případě, že se robot pohybuje v přímém směru, urazí dráhu o velikosti jedné jednotky a časový interval je roven jedné. Pokud se robot pohybuje v diagonálním směru, urazí dráhu o velikosti druhé odmocniny ze dvou jednotek. Časový interval je pak roven odmocnině ze dvou. Rychlost dynamických překážek je zadána relativně k rychlosti robotu. Dráha, kterou dynamická překážka mezi vzorkováním urazí, je potom součinem časového intervalu a rychlosti překážky. Pokud se dynamická překážka na počátku pohybu nachází mimo celočíselné souřadnice, ověří se, zda může dokončit započatý krok. Pokud ne, posune se o celou zbývající vzdálenost a po tomto necelém kroku stále zůstává mimo celočíselné souřadnice. V opačném případě je

dokončen krok. Pokud se překážka nachází v celočíselných souřadnicích, ověřuje se, zda zbývající dráha postačuje na příští krok. Pokud ano, krok se vykoná a délka kroku se odečte od zbývající dráhy. Takto se postupuje, dokud není zbývající dráha nulová. Pokud je plánovaný krok delší než zbývající dráha, vykoná se pouze částečný krok a překážka zůstane v pozici mimo mřížku.

4.3. Navržené úpravy pro algoritmus MSAC

4.3.1. Tabu tabulka

Slabým místem původního algoritmu MSAC je rozpoznávání setkání mravenců. Setkání mravenců je dle definice stav, kdy se v okolí mravence z jedné skupiny nachází pole z cesty mravence druhé skupiny. Tato definice je důležitá z toho důvodu, že při hledání se nemusí setkat přímo mravenci, ale jeden z mravenců narazí na cestu mravence z jiné skupiny. Tento stav umožňuje spojit jejich cesty do výsledné nalezené cesty.

Následuje popis počátečního řešení problému. Pro všechny mravence v obou skupinách se po vykonání každého kroku kontroluje, zda se některý bod z okolí mravence (včetně současné pozice mravence) nevyskytuje v cestě některého mravence z jiné skupiny. Tomu odpovídá složitost až $2 \cdot 9 \cdot m^2 \cdot S$ v každém kroku. Zahrnuje dvě skupiny, devět okolních polí, m mravenců v každé skupině, S je pořadí kroku od počátku hledání. Celková složitost v celém průběhu hledání je $18 \cdot m^2 \cdot \sum_{S=1}^{S_{end}} S$, kde S_{end} je pořadí kroku (iterace), ve kterém bylo setkání detekováno.

Za účelem snížení složitosti bylo navrženo účelnější využití tabu tabulky. Původní tabu tabulka měla formu matice o rozměrech prohledávaného prostředí. Prozkoumaná pole byla označena hodnotou 1, neprozkoumaná 0. První úpravou bylo zaznamenání informace o tom, která skupina dané pole prozkoumala jako první. Tomu odpovídá označení: 0 neprozkoumáno, 1 prozkoumáno první skupinou, 2 prozkoumáno druhou skupinou.

Po vykonání kroku se v tabu tabulce zkontroluje, zda se v okolí mravence vyskytuje pole prozkoumané jinou skupinou. Pokud ano, zaznamená se, který mravenec se setkal s druhou skupinou včetně pole setkání.

Tento způsob znamená, že složitá kontrola se provádí pouze v případě, kdy je jisté, že povede k nalezení cesty. Po každém kroku se vykonává kontrolní proces se složitostí $18m$. Pro $2m$ mravenců se porovnává 9 polí vůči tabu tabulce. Po detekci setkání následuje hledání, který mravenec z druhé skupiny prozkoumal bod setkání. To znamená kontrolu pro každý bod setkání až $m \cdot S$. Popsaný způsob využití tabu tabulky vedl ke zřetelnému zrychlení algoritmu, stále se ale po detekci kolize hledalo naslepo ve všech cestách mravenců druhé skupiny.

Závěrečným zlepšením tabu tabulky je následující uspořádání. Jedná se o záznam jak označení skupiny, tak i označení mravence, který jako první pole prozkoumal. Tabu tabulka je implementována ve formě matice prvků datového typu *uint8* (osmibitové kódování celých čísel bez znaménka). V popisovaném návrhu je využití jednotlivých prvků matice následující. Hodnota 0 značí neprozkoumané pole mřížky. Hodnoty v rozsahu 1–127 značí pole prozkoumané první skupinou a hodnota je přímo identifikačním číslem mravence. Hodnota 128 není využita. Rozsah 129–255 označuje pole prozkoumaná druhou skupinou, hodnota udává identifikační číslo mravence navýšené o 128. Hodnota 128 není využita z toho důvodu, že počet mravenců je v obou skupinách vždy stejný.

Nyní je setkání detekováno se stejnou složitostí jako v druhém případě. Rozdíl je v tom, že po detekci kolize už stačí najít pouze bod setkání v cestě jednoho známého mravence z druhé skupiny. Z uvedené změny plyne omezení maximálního počtu mravenců, které však není nijak dramatické (127 pro každou skupinu, [41] uvádí jako typickou hodnotu 4).

Výstupem je cesta vytvořená z cest dvou mravenců, kteří se setkali. Pro zopakování, setkáním mravenců je myšlena situace, kdy se v okolí mravence z jedné skupiny vyskytuje pole cesty mravence z druhé skupiny. Vysledná cesta proto není pouhým spojením celých dvou cest mravenců. Jedna z cest může být využita pouze částečně.

Vliv typu tabu tabulky na rychlost hledání je uveden v části 6.1.1.

4.3.2. Podmíněný přenos informace o slepých uličkách mezi iteracemi

Druhým navrženým vylepšením je přenos informace o nalezených slepých uličkách mezi iteracemi hledání. Oproti původnímu návrhu ze [41], kdy se informace o slepých uličkách udržovala v rámci jedné iterace, se toto vylepšení jevilo jako logické. Problematická se ukázala situace, kdy se pozice startu nebo cíle hledání vyskytovala ve slepé uličce. Ta byla v průběhu hledání označena jako dočasná neprůchozí pozice. V následující iteraci hledání došlo k chybě, protože mravenci neměli žádná volná okolní pole, kam by se mohli přesunout, a výsledkem bylo nenalezení cesty v následujících iteracích hledání.

Původní myšlenkou bylo v případě obsazení počáteční či cílové pozice zjistit, které souvislé uskupení bodů slepých uliček zasahuje na kritická místa. Do další iterace by se pak přenesly všechny ostatní body slepých uliček. Tato analýza se však ukázala být složitou a výpočetně náročnou. Z toho důvodu bylo přistoupeno k řešení, že nalezené body slepé uličky se v další iteraci využijí pouze v případě, kdy žádná z nich neblokuje startovní nebo cílovou pozici.

4.3.3. Zpracování nalezených cest

Další úpravou je zpracování nalezených cest. V průběhu algoritmu MSAC jsou generovány cesty, které se konfrontují s dosavadní průběžnou nejkratší nalezenou cestou. Generované cesty mohou obsahovat cykly. Cykly v nalezené cestě vznikají typicky ve dvou případech. Prvním případem je situace, kdy část cesty mravence vedla do slepé uličky a zpět. Druhou možností je případ, kdy se mravenec pohyboval po území, které již jeho skupina prozkoumala. Výběr dalších polí je v tomto případě náhodný a mohou vzniknout cykly. Bez odstranění cyklů z generovaných cest by mohla být krátká cesta zamítnuta jako delší, pokud by obsahovala zbytečný cyklus uměle navyšující její délku.

Cestu zbavenou cyklů je možné ještě dalším způsobem vylepšit, a to mírným vyhlazením. Vyhlazení je provedeno následujícím způsobem. Cesta se postupně od startovní pozice uzel po uzlu kontroluje, zda se v nejbližším dostupném okolí vyskytuje více než jeden uzel, přes který cesta prochází. Pokud ne, zkoumaný uzel se z cesty pro tuto kontrolu vyřadí a pokračuje se na další uzel (vyřazený uzel však pro výstup zůstává součástí cesty). Pokud ano, znamená to, že v cestě se vyskytuje část, kterou je možné odstranit při zachování spojitosti cesty. Tímto způsobem lze cestu zkrátit a mírně vyhladit. V cestě se ponechá ten sousední uzel, který má nejvyšší index, tj. ten, který je z hlediska časové posloupnosti nejblíže k cíli. Ostatní sousední uzly jsou z cesty odstraněny. Potom se stejně jako v předchozím případě zkoumaný uzel pro potřebu kontroly z cesty vyřadí.

Výše uvedené zpracování generovaných cest stojí jistý výpočetní čas. Z toho důvodu byla přidána možnost toto zpracování povolit nebo zakázat. V případě, že by byla rychlost výstupu důležitější než garance, že bylo uděláno maximum pro dosažení nejlepšího výsledku.

4.3.4. Podmíněný a nepodmíněný diagonální pohyb

Původní algoritmus MSAC uvažoval osmisměrný pohyb, kdy diagonální přechod byl podmíněný uvolněnými odpovídajícími hlavními směry. To odpovídá situaci, kdy je velikost robotu rovna velikosti mřížky. Byla implementována i možnost výběru čistě osmisměrného pohybu bez výše zmíněné podmínky. Tato situace odpovídá bodovému robotu v prostředí s rozšířenými překážkami o hodnotu větší než poloměr robotu (není povolen ani dotyk překážky).

4.4. Algoritmus SACO

Jednoduchý algoritmus optimalizace mravenčí kolonií SACO (*simple ant colony optimization algorithm*) [12] je implementací metaheuristiky optimalizace mravenčí kolonií, která přizpůsobuje chování skutečných mravenců pro hledání řešení problému nejlevnější cesty v grafech.

Několik umělých mravenců tvoří řešení optimalizačního problému pomocí vytváření a výměny informací o kvalitě těchto řešení. Tímto je napodoben způsob komunikace skutečných mravenců.

Mravence považujeme za bodového mobilního robota ve 2D prostředí. Specifikace pozice robota vůči fixnímu souřadnicovému systému se nazývá konfigurace q, která je dána rovnicí:

$$q = (p, \theta) = (x, y, \theta), \tag{4.1}$$

kde $p = (x, y)$ je pozice robotu a θ je orientace.

Množina všech možných konfigurací se nazývá konfigurační prostor. Některé body konfiguračního prostoru nejsou robotu přístupné z důvodu přítomnosti překážky. Množina všech přípustných konfigurací robotu se nazývá volný konfigurační prostor, jedná se o všechny konfigurace s výjimkou těch pozic, které jsou obsazeny překážkami. Řešením hledání cesty je nalezení posloupnosti Q navazujících bodových konfigurací q ve volném konfiguračním prostoru z počáteční konfigurace q_s do cílové konfigurace q_g.

Pokud uvažujeme robot o poloměru R, pak některá řešení Q nebudou vyhovovat omezení vyplývající z nenulové velikosti robotu. Hledání cesty pro robot o nenulovém poloměru se řeší zmenšením volného konfiguračního prostoru o body, které jsou nepřístupné z důvodu velikosti robotu. Tohoto zmenšení lze docílit expanzí všech překážek do všech směrů o velikost R. Nyní lze použít stejný postup jako pro robota s nulovým poloměrem. Řešení v tomto redukovaném volném konfiguračním prostoru je označeno $Q(R)$.

Kroky SACO jsou popsány následovně:

1. Každá hrana (i, j) je spojena s koncentrací feromonu označenou jako τ_{ij}.

2. Mravenci $k = 1, \ldots, n_k$ jsou umístěni do výchozí pozice hledání.

3. V každé iteraci nebo epoše všichni mravenci vytvoří cestu do cílového uzlu. Pro výběr dalšího uzlu se používá pravděpodobnostní vztah:

$$p_{ij}^k(t) = \begin{cases} \dfrac{\tau_{ij}^\alpha(t)}{\sum\limits_{j \in N_i^k} \tau_{ij}^\alpha(t)}, & \text{pokud } j \in N_i^k; \\ 0, & \text{jinak;} \end{cases} \tag{4.2}$$

kde N_i^k je množina uzlů dostupných z uzlu i, kde se nachází mravenec k. Jedná se o sousední uzly neobsazené překážkou. τ_{ij} je koncentrace feromonu hrany (i, j), α je kladná konstanta posilující vliv koncentrace feromonu. Uzly sousedící s uzlem i, které jsou obsazeny překážkou, mají nulovou pravděpodobnost výběru.

4. Jsou odstraněny cykly v cestách a vypočteny váhy pro nalezené cesty $f(x^k(t))$.

5. Hodnoty feromonu jsou upraveny vlivem vypařování dle vztahu:

$$\tau_{ij}(t) \leftarrow (1 - \rho)\tau_{ij}(t), \tag{4.3}$$

kde ρ je míra rychlosti vypařování feromonové stopy. Vypařování je součástí algoritmu z toho důvodu, aby mravenci prozkoumali více řešení a aby bylo zabráněno předčasné konvergenci k sub-optimálním řešením. Pro $\rho = 1$ je hledání zcela náhodné. Vypařování zabraňuje konvergenci k lokálním optimům. Bez vypařování se cesty prvních mravenců stanou příliš atraktivními pro následující mravence. Pomocí tohoto opatření není prozkoumávání prostoru příliš omezeno.

6. Koncentrace feromonu je aktualizována dle vztahu:

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \sum_{k=1}^{n_k} \Delta\tau_{ij}^k(t). \tag{4.4}$$

7. Algoritmus může být ukončen jedním ze tří způsobů:

   (a) Pokud bylo dosaženo maximálního počtu iterací nebo epoch.

   (b) Pokud bylo nalezeno řešení s přijatelným ohodnocením $f(x^k(t)) < \varepsilon$.

   (c) Pokud všichni mravenci v průběhu jedné epochy nalezli stejné řešení.

4.4.1. SACOdm

Algoritmus SACOdm (dm značí vzdálenost a paměť – *distance*, *memory*) [12] se od algoritmu SACO liší ve dvou bodech. První odlišností je změna vztahu (4.2) pro volbu dalšího uzlu. Vztah je upraven tak, aby byly preferovány uzly blíže k cílovému uzlu. Toto vylepšení se lépe projeví ve scénách s převahou volného, prostupného prostředí. Modifikace je popsána následovně: ξ_j je euklidovská vzdálenost mezi cílovým uzlem a uzlem j, β je konstanta působící na vliv ξ, platný rozsah pro β je $\langle 0; \infty)$. Nový vztah pro výběr následujícího uzlu je:

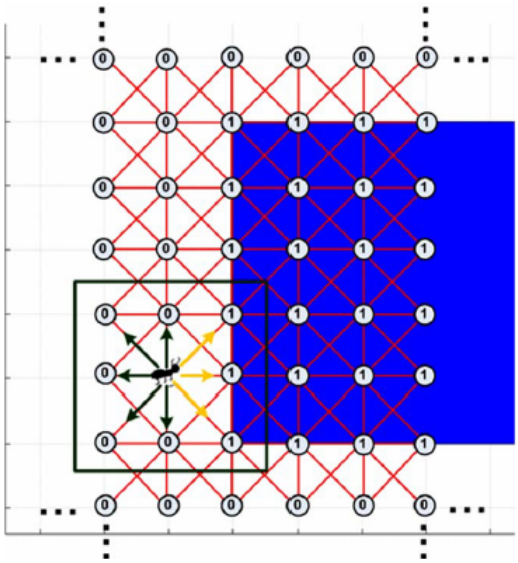
$$p_{ij}^k(t) = \begin{cases} \dfrac{\tau_{ij}^\alpha(t)}{\xi_j^\beta \sum\limits_{l \in N_i^k} \tau_{il}^\alpha(t)}, & \text{pokud } j \in N_i^k; \\ 0, & \text{jinak.} \end{cases} \tag{4.5}$$

Je nutné poznamenat, že součet všech hodnot $p_{ij}^k(t)$ pro uzel i se po této úpravě obecně nemusí rovnat jedné. Z toho důvodu se už nejedná o pravděpodobnost. Pravděpodobnost výběru uzlu obdržíme poměrem hodnoty $p_{ij}^k(t)$ vůči součtu těchto hodnot pro všechny sousední uzly, kdy součet odpovídá 100 %. Hodnota $p_{ij}^k(t)$ pro uzly obsazené překážkou se rovná nule.

Druhou změnou je zápis prozkoumaných uzlů do paměti a jejich označení jako nepřístupných. Tímto se zabraňuje stagnaci algoritmu. Úprava používá hodnotu γ, což je referenční hodnota pro kapacitu paměti. Na začátku prohledávání se udržuje seznam již prozkoumaných uzlů, které jsou dočasně označeny v pracovním prostředí jako překážky. Když počet kroků při tvorbě řešení dosáhne hodnoty γ, uzly ze seznamu jsou opět uvolněny pro hledání.

4.4.2. Pracovní prostředí

Scéna, ve které se hledá cesta, je diskretizovaná do obdélníkové matice, kde 0 označuje dostupný uzel a 1 překážku. Pro nenulový poloměr robotu se použije upravená matice s překážkami rozšířenými o hodnotu větší než poloměr robotu. Hodnoty feromonu jsou uloženy v matici o stejných rozměrech, které má matice scény. Oproti obvyklému pojetí, kdy je hodnota feromonu přiřazena hranám, se hodnota feromonu v uzlu se použije pro všechny hrany, které do tohoto uzlu směřují. Mravenci při tvorbě řešení vybírají následující uzel v okně 3×3 uzlů. Následující uzel je zvolen z až 8 sousedních uzlů pomocí vztahu (4.5). Výběr z menšího počtu uzlů je způsoben výskytem překážek v sousedních uzlech, případně pozicí mravence na okraji scény.



Obrázek 4.3: Pozice mravence v diskretizovaném prostředí.

Na obrázku 4.3 je zobrazen mravenec v diskretizovaném prostředí. Modrý vyplněný obdélník je překážka. Tmavě zelený čtverec ohraničuje oblast, ze které mravenec vybírá

následující uzel. Pouze uzly s hodnotou 0 jsou přístupné, tyto možné kroky jsou zobrazeny tmavě zelenými šipkami. Uzly s hodnotou 1 jsou nedostupné, neproveditelné přechody jsou naznačeny žlutými šipkami.

4.5. Navržené úpravy pro algoritmus SACO

4.5.1. Průběžný seznam zakázaných polí

První návrh na změnu se týká využití paměti. SACOdm na počátku hledání zakazuje mravencům návrat do prvních γ navštívených uzlů. Místo toho bylo navrženo držet v paměti průběžný seznam posledních γ navštívených uzlů, které jsou označeny jako mravencům nepřístupné za pomoci dočasných překážek. V případě, že kvůli dočasným překážkám není žádný sousedící uzel dostupný, jsou všechny dočasné překážky na sousedních uzlech odstraněny. Toto opatření zmenšuje možnost vzniku cyklů, ale ne zcela. Právě v případě, kdy není žádný sousední uzel dostupný kvůli dočasným překážkám, po provedení kroku vznikne v cestě cyklus.

4.5.2. Zpracování nalezených cest

Dalším návrhem pro změnu je zpracování nalezených cest po jejich vytvoření. Součástí čtvrtého kroku algoritmu SACO je odstranění cyklů. Další úpravou cesty může být vyhlazení. Proces mírného vyhlazení je totožný s návrhem popsaným v části 4.3.3.

Drobnou úpravou je odstranění cyklů v nalezených cestách okamžitě po jejich vytvoření. Původní vytvořené cesty mohou být ve velké míře tvořeny cykly, a pokud překročily velikost původně prelokovaného místa v paměti, v době mezi jejich vytvořením a úpravou zbytečně zabírají paměť. Nečeká se na vytvoření všech řešení, cesty jsou zpracovány okamžitě po svém vygenerování (odstraněním cyklů a mírným vyhlazením).

4.5.3. Odlišné ohodnocení cest, doplnění hodnoty aktualizace feromonu

Kvalita cest je ohodnocena délkou cesty, nikoli využitím fuzzy regulátoru, jehož návrh a ladění řeší článek [12].

Zmíněný článek neobsahuje definici hodnoty aktualizace feromonu $\Delta\tau$ v závislosti na kvalitě cesty. Vztah byl doplněn z literatury [9] dle algoritmu *Ant system* jako převrácená hodnota délky cesty, s možností zvolit způsob aktualizace algoritmu *Elitist ant system*. Ten po každé iteraci posiluje feromonovou stopu na průběžně nejlepší nalezené cestě s vahou, kterou je možné zadat jako parametr.

5. Popis programu

Navržený systém byl implementován v programovacím jazyce MATLAB. Algoritmy byly implementovány jako funkce, které jsou volány z předpřipravených skriptů s ukázkovými vstupy. Soubory jsou přiloženy na CD.

Následuje popis hlavních funkcí.

5.1. RNA

Funkce RNA simuluje pohyb robotu přes neznámé dynamické prostředí.

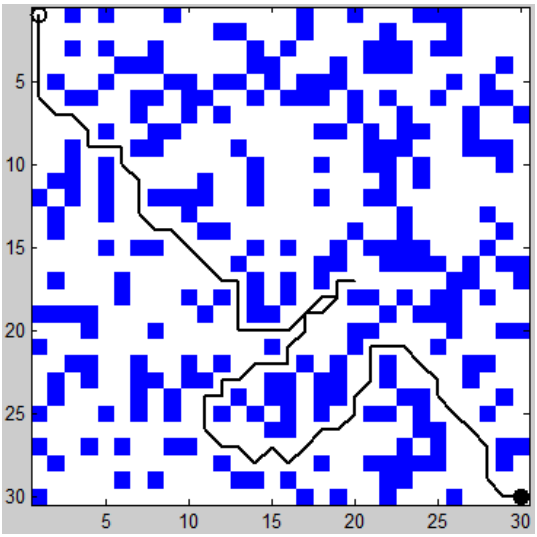
Vstupy funkce RNA jsou následující:

- Souřadnice startu a cíle – určují výchozí a cílovou pozici robotu v prostředí.
- Matice prostředí – obdélníková matice, 0 značí volné pole, 1 značí nepřístupné pole obsazené překážkou.
- Typ oscilace – určuje, zda je oscilace definována jako návrat do kterékoli dříve navštívené pozice nebo pouze do předcházející pozice. Dle vybraného typu oscilace si robot dočasně zakazuje pozice v prostředí. Zákaz je proveden tak, že daná pozice je dočasně označena jako obsazená překážkou.
- Maximální počet přeplánování – hraniční hodnota počtu přeplánování při předpovědi kolize s dynamickou překážkou. Při dosažení hraničního počtu provedených přeplánování pohybu se robot zastaví a počká jednotku času.
- Maximální počet čekání – hraniční hodnota ukončující algoritmus při opakovaném nenalezení cesty statickým prostředím.
- Dohled robotu – určuje velikost oblasti, ve které robot zná rozmístění překážek.
- Vlastnosti dynamických překážek – každá dynamická překážka je definována pomocí:
  - Souřadnic počáteční pozice.
  - Velikostí rychlosti – zadávána v poměru k rychlosti robotu.
  - Směrem počátečního pohybu.
  - Hranic omezující její pobyb – obdélníková oblast určená limitními hodnotami pozice dynamické překážky.
- Typ vizualizace – umožňuje zapnout vizualizaci globálního pobybu, procesu lokálního hledání a výsledku lokálního hledání.

Dalšími vstupy jsou také parametry determinující lokální hledání pomocí algoritmu MSAC. Přímo je třeba zadat počet mravenců m a hodnotu opakování hlavní smyčky hledání MAX. Další vstupy do funkce MSAC generuje sama funkce RNA na základě ostatních vstupů.

Varianta WF-RNA neobsahuje vstupy pro lokální hledání. WF je zkratkou pro *wave front planner*, což je obdoba prohledávání do šířky pro diskretizované prostředí mřížky. Vstupy pro toto hledání jsou pouze start, cíl a prostředí. Všechny jsou všechny generovány algoritmem RNA.

Výstupem funkce RNA je nalezená cesta, případně průběžná vizualizace procesu hledání cesty a pohybu robotu přes prostředí.



Obrázek 5.1: Globální cesta robotu (černá lomená čára) nalezená pomocí algoritmu RNA, dohled robotu 6. Bílá barva označuje volný prostor, modrá značí překážky. Startovní pozice v levém horním rohu, označena prázdným kroužkem. Cílová pozice robotu je v pravém dolním rohu, plný kroužek. Můžeme si všimnout, že robot v jednom okamžiku mířil do slepé uličky. Jakmile dohlédl na její konec, začal se pohybovat jiným směrem.

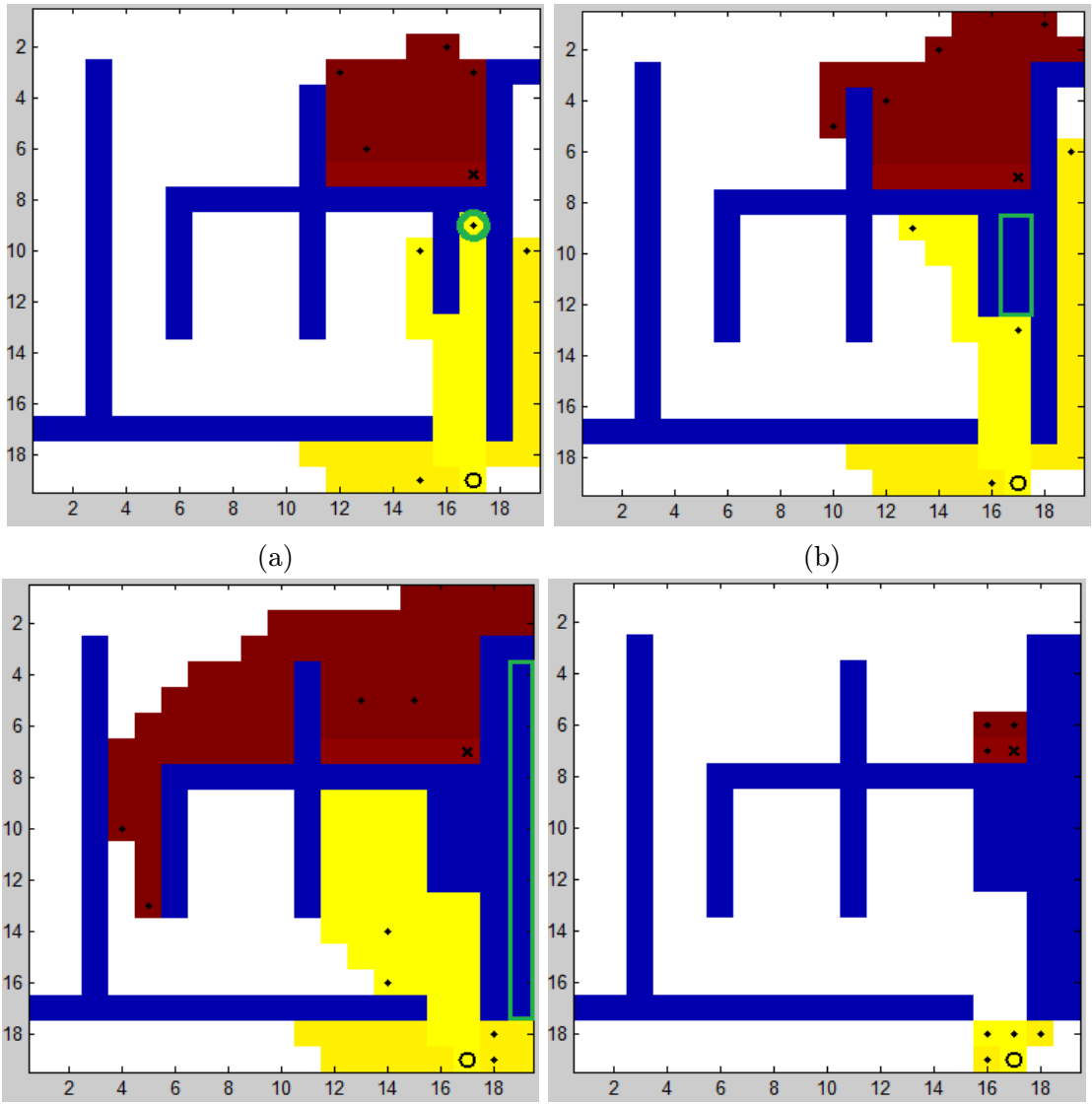
5.2. MSAC

MSAC je funkce pro hledání cesty ve známém prostředí.

Její vstupy jsou popsány následovně:

- Souřadnice startu a cíle – výchozí a cílová pozice robotu ve známém prostředí.
- Matice prostředí – obdélníková matice, 0 značí volné pole, 1 značí nepřístupné pole obsazené překážkou.
- m – počet mravenců v obou skupinách provádějících lokální hledání.
- S_{max} – maximální počet kroků mravenců v jedné iteraci hledání.
- MAX – počet iterací hledání.
- Podmíněný diagonální přechod – určuje definici sousedních polí dostupných pro přechod.
- Zpracování cest – volba zapnutí či vypnutí úpravy nalezencýh cest.
- Typ vizualizace – umožňuje zapnout vizualizaci průběhu lokálního hledání a výsledku lokálního hledání.

Výstupem algoritmu je nalezená cesta reprezentovaná seznamem lokálních souřadnic, případně vizualizace procesu a výsledku hledání.



(a) (b)

(c) (d)

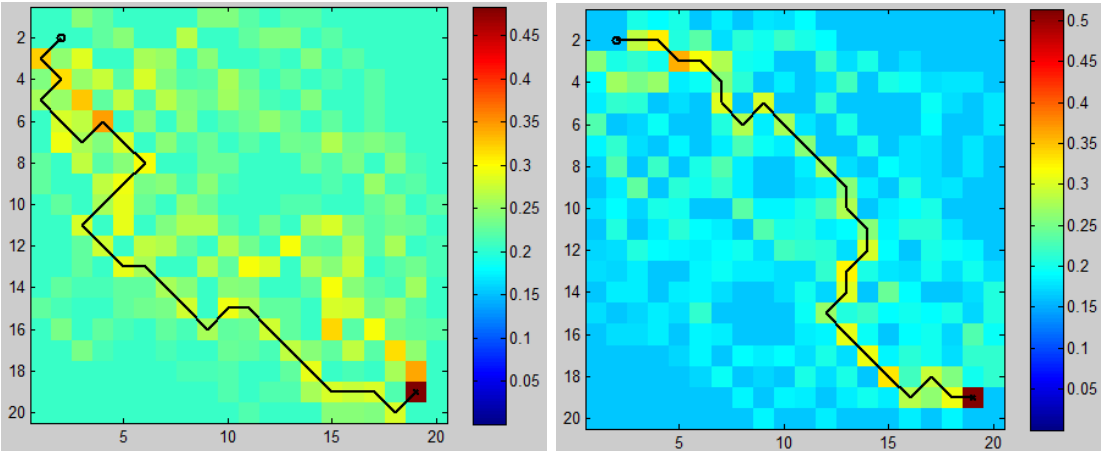
Obrázek 5.2: Ukázka přenosu slepých uliček do další iterace hledání v algoritmu MSAC. Modrá barva znázorňuje překážky, bílá přístupné neprozkoumané území, žlutá území prozkoumané první skupinou mravenců, tmavě červená území prozkoumané druhou skupinou. Start je označen černým kroužkem, cíl černým křížkem, černé tečky znázorňují pozici mravenců. (a) Mravenec označený zeleným kroužkem narazil na konec slepé uličky, jediné přístupné pole je již prozkounané. (b) Cestou zpět mravenec označil slepou uličku jako překážku - zvýrazněno zeleným obdélníkem. (c) Také mravenec v pravém sloupci označil slepou uličku jako nepřístupnou. (d) Na počátku další iterace jsou slepé uličky nepřístupné a mravenci mohou hledat cestu na perspektivnějších místech.

5.3. SACOdm

SACOdm je funkce pro hledání cesty ve známém prostředí.

Vstupy funkce jsou následující:

- Souřadnice startu a cíle – výchozí a cílová pozice robotu ve známém prostředí.



Obrázek 5.3: Průběžné výsledky hledání cesty algortimu SACOdm na volné scéně. Barva znázorňuje hodnotu feromonů v uzlech, černá lomená čára je průběžné nejlepší nalezené řešení.

- Matice prostředí – obdélníková matice, 0 značí volné pole, 1 značí nepřístupné pole obsazené překážkou.
- Počet mravenců provádějících hledání.
- Konstanty vlivu pro výběr dalšího pole dle vztahu 4.5.
  - α – konstanta posilující vliv koncentrace feromonů.
  - β – konstanta posilující vliv vzdálenosti vybraného uzlu od cíle.
- ρ – míra vypařování feromonu.
- γ – počet zakázaných polí pro návrat mravence.
- Typ seznamu γ – výběr varianty z možností:
  - Prvních γ navštívených polí pro začátek hledání.
  - Průběžný seznam posledních γ navštívených polí v průběhu celého hledání.
- Typ aktualizace feromonu – výběr z možností:
  - *ant system* – navýšení feromonů na všech cestách z iterace hledání dle jejich délky.
  - *elitist ant system* – stejné navýšení jako v předchozím případě s přídavným ohodnocením pro průběžně nejlepší nalezené řešení.
- Váha pro průběžně nejlepší nalezenou cestu, pokud je vybrán typ aktualizace feromonu *elitist ant system*.
- Počáteční hodnota feromonu.
- Počet iterací.
- Hodnoty pro předčasné ukončení hledání: absolutní dostatečná délka cesty, relativní dostatečná délka cesty. Relativní délka je vyjádřena poměrem vůči nekratší dosažitelné vzdálenosti při osmisměrném pohybu ve volné mřížce.
- Podmíněný diagonální přechod – určuje definici sousedních polí dostupných pro přechod.
- Typ vizualizace – umožňuje zapnout vizualizaci průběžných výsledků jednotlivých iterací hledání a celkového výsledku lokálního hledání.

Výstupem algoritmu je nalezená cesta reprezentovaná seznamem souřadnic, volitelným výstupem je průběžná vizualizace tvorby řešení.

6. Ověřovací a srovnávací experimenty

Kapitola popisuje provedené srovnávací experimenty a jejich výsledky. Experimenty byly provedeny na počítači s procesorem Intel Core i3 330M, 2,13 GHz a 4 GB operační paměti. Experimenty jsou provedeny s nepodmíněným diagonálním pohybem, s výjimkou srovnání cest algoritmu RNA a MSAC s výsledky z článku [41].

6.1. MSAC

6.1.1. Vliv verze tabu tabulky

Vliv implementace tabu tabulky na rychlost algoritmu MSAC. Vstupní parametry byly nastaveny na hodnoty: $MAX = 3$, $m = 4$, $S_{max} = 1250$. Měřil se čas potřebný pro vykonání 100 hledání ve scéně 1 (zobrazena na obrázku 6.2). Verze 1 je tabulka rozlišující pouze prohledáno nebo neprohledáno bez ohledu na skupinu; verze 2 je tabulka rozlišující skupinu, která pole prohledala; verze 3 je tabulka zaznamenávající jak skupinu, tak i mravence, který pole prohledal. Výsledky pro jednotlivé typy tabu tabulky shrnuje tabulka 6.1.

Tabu tabulka	Verze 1	Verze 2	Verze 3
Průměrný čas [s]	0,586	0,130	0,104

Tabulka 6.1: Průměrná doba potřebná pro nalezení cesty dle implementace tabu tabulky.

Z výsledku vyplývá, že závěrečná implementace tabu tabulky (verze 3) je v uvedeném experimentu více než pětinásobně rychlejší oproti výchozí verzi.

Ve všech ostatních experimentech bylo provedeno 10 měření a byly zaznamenány průměrné a minimální hodnoty nalezených cest a průměrná doba potřebná pro nalezení cest.

6.1.2. Vliv předzpracování nalezených cest

Pro tento experiment byly zvoleny tyto parametry: $MAX = 3$, $m = 4$, $S_{max} = 1250$.

Experiment sleduje vliv povolení a zákazu předzpracování nalezených cest na dobu potřebnou pro nalezení cesty a na délku cesty. Bylo provedeno 10 měření.

Předzpracování	Ano	Ne
Průměrný čas [s]	0,1010	0,0860
Průměrná délka cesty	73,84	73,84
Minimální délka cesty	73,84	73,84

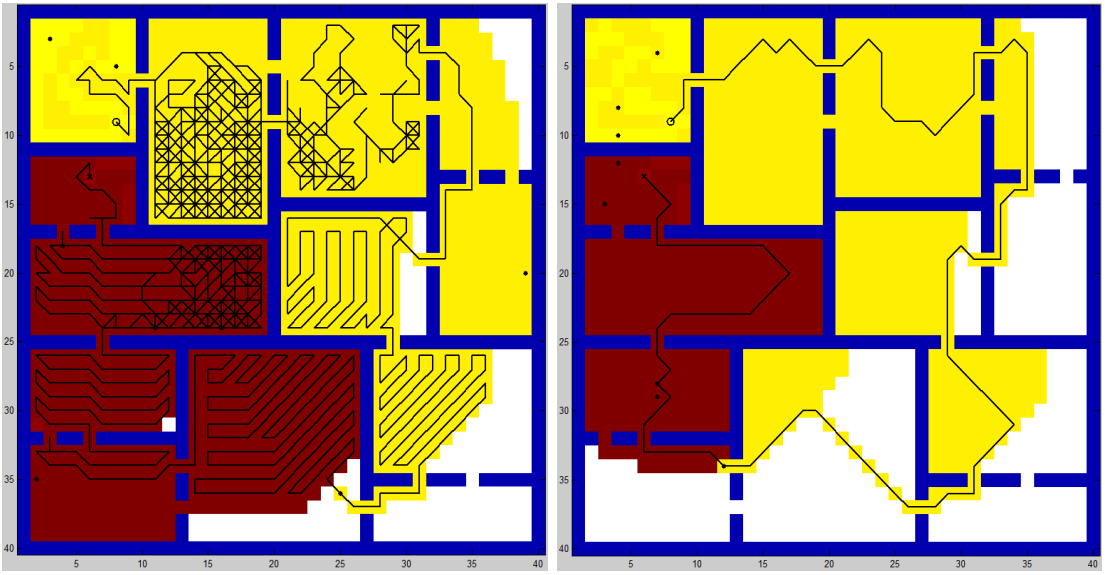
Tabulka 6.2: Vliv předzpracování nalezených cest, scéna 1.

Výsledky ukazují, že pro poměrně volnou scénu není žádný rozdíl v délce nalezených cest, bez ohledu na jejich předzpracování. V obou případech měla nalezená cesta ve všech 10 měřeních stejnou délku. Průměrná doba nalezení cesty byla pro předzpracování delší přibližně o 17 %. Dále byl experiment zopakován se stejnými parametry pro scénu 2 (převzata z [33]).

Předzpracování	Ano	Ne
Průměrný čas [s]	3,694	3,070
Průměrná délka cesty	149,12	1018,16
Minimální délka cesty	140,17	808,67

Tabulka 6.3: Vliv předzpracování nalezených cest, scéna 2.

Výsledky experimentu na scéně 2 ukazují významné rozdíly v kvalitě nalezené cesty. Průměrná délka cest bez použití předzpracování je více než šestinásobně kratší než délka cest s použitím předzpracování. Průměrný čas potřebný pro nalezení cesty je pro způsob s předzpracováním delší přibližně o 20 %. Z výsledku je patrné, že pro složité scény s lokálním minimem má předzpracování cesty značný vliv na její kvalitu.



(a) (b)

Obrázek 6.1: Scéna 2. Srovnání cest nalezených implementací algoritmu MSAC bez zpracování nalezených cest (a) a se zpracováním nalezených cest (b). Překážky jsou zobrazeny modře, bílá značí neprozkoumané prostředí. Oblast prozkoumaná první skupinou je zobrazena žlutě, oblast prozkoumaná druhou skupinou je znázorněna tmavě červeně. Konečná pozice mravenců je označena tučnými černými tečkami.

Další experimenty byly prováděny s použitím předzpracování.

6.1.3. Vliv počtu mravenců

Experiment byl proveden s těmito parametry: $MAX = 3$, $m = 4$, $S_{max} = 1250$.

Počet mravenců m	2	4	8
Průměrný čas [s]	2,521	3,682	6,167
Průměrná délka cesty	152,71	146,95	149,80
Minimální délka cesty	142,41	136,75	138,51

Tabulka 6.4: Vliv počtu mravenců, $MAX = 3$, scéna 2.

Výsledky dle očekávání ukazují, že časová náročnost roste se zvětšujícím se počtem mravenců, pro případy 2 a 8 mravenců je nárůst téměř dvouapůlnásobný. Minimální

i průměrná délka cesty byla nejnižší pro 4 mravence. Průměrná i minimální délka cesty pro 8 mravenců je delší o méně než 2 % oproti 4 mravencům. Pro 2 mravence je průměrná i minimální délka větší o přibližně 4 % oproti případu 4 mravenců.

Experiment byl zopakován se stejnými parametry, s výjimkou parametru MAX, pro který byla zvolena hodnota 1. Parametr MAX udává, kolikrát se opakuje hlavní smyčka hledání v rámci jednoho volání funkce MSAC.

Počet mravenců m	2	4	8
Průměrný čas [s]	0,875	1,212	1,937
Průměrná délka cesty	158,45	157,02	156,62
Minimální délka cesty	150,49	139,58	148,41

Tabulka 6.5: Vliv počtu mravenců, $MAX = 1$, scéna 2.

Oproti předchozímu případu je průměrný čas dle očekávání přibližně třetinový. V tomto experimentu průměrná délka cesty klesá s rostoucím počtem mravenců. Nejkratší minimální délka cesty byla v tomto případě nalezena 4 mravenci. Je nutné poznamenat, že v jednom hledání při využití 2 mravenců nebyla cesta vůbec nalezena. Toto hledání bylo ze zpracování výsledků vyřazeno. Zatímco relativní rozdíl v průměrné délce cesty se pro 2 mravence a 8 mravenců liší o méně než 1,5 %, průměrný čas vzrostl více než dvojnásobně.

Výsledky ukazují, že vyšší hodnota parametru MAX poskytuje i menšímu množství mravenců prostor pro nalezení dobrého řešení. V případě, kdy byl parametr MAX roven jedné, se více projevil vliv počtu mravenců na průměrnou délku cesty.

Průměrná délka cesty pro hodnotu parametru $MAX = 1$ je pro počet mravenců dva, čtyři a osm v tomto pořadí o přibližně 3,8 %, 6,9 % a 4,6 % delší než pro hodnotu 3.

Vliv počtu mravenců byl dále vyzkoušen na scéně 1.

Počet mravenců m	2	4	8
Průměrný čas [s]	0,060	0,100	0,206
Průměrná délka cesty	73,84	73,84	73,84
Minimální délka cesty	73,84	73,84	73,84

Tabulka 6.6: Vliv počtu mravenců, $MAX = 3$, scéna 1.

Pro scénu 1 se opakují výsledky z tabulky 6.2, kdy ve všech případech bylo nalezeno řešení délky 73,84. Průměrný čas potřebný pro nalezení cesty dle očekávání narůstá pro větší počet mravenců.

Počet mravenců m	2	4	8
Průměrný čas [s]	0,083	0,145	0,214
Průměrná délka cesty	65,05	65,05	65,05
Minimální délka cesty	65,05	65,05	65,05

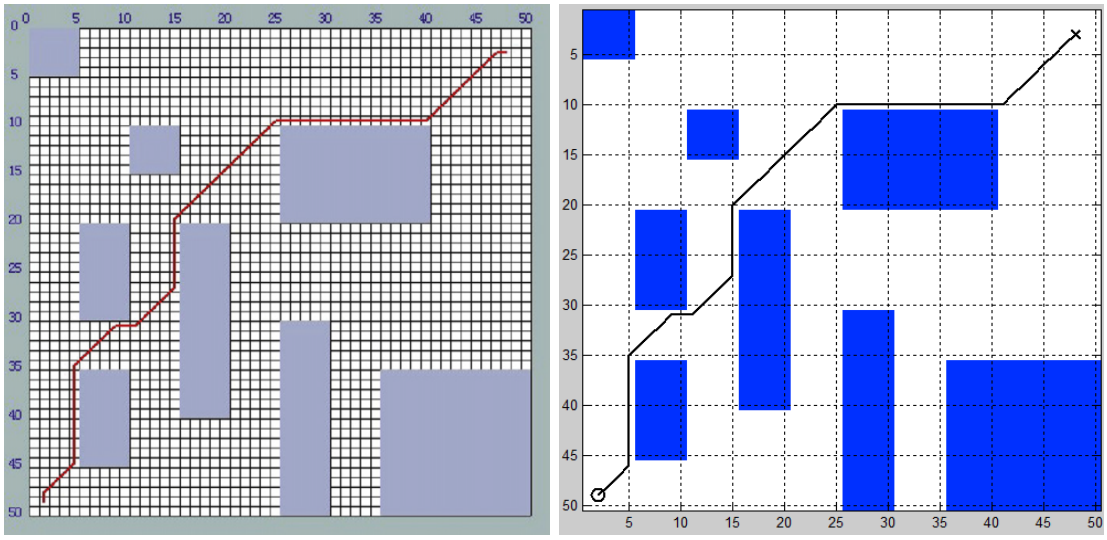
Tabulka 6.7: Vliv počtu mravenců, $MAX = 3$, volná scéna, vzdálenost startu a cíle 65,05.

Na volné scéně algoritmus MSAC ve všech případech nalezl nejkratší cestu. Průměrný čas narůstá pro zvyšující se počet mravenců.

Celkově lze říci, že nastavení parametrů má na průměrnou délku cesty poměrně malý vliv. Pro volnou scénu a pro otevřenou scénu 1 bez lokálního minima byla vždy nalezena nejkratší cesta bez ohledu na nastavení. Měnil se pouze čas potřebný pro nalezení cesty.

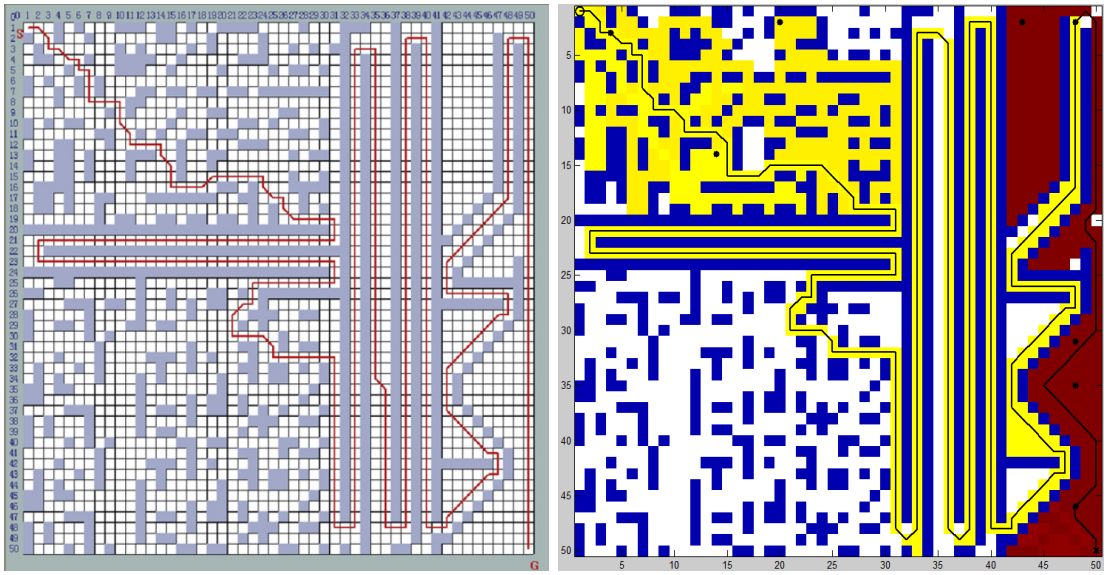
Ve scéně 2 se ukázala důležitost zpracování nalezených cest v případech, kdy se v prostředí vyskytují lokální minima. Zpracování cest zkrátilo průměrnou délku cesty šestinásobně. Nastavení ostatních parametrů ve scéně 2 vedlo k rozdílům v průměrné délce nalezené cesty v jednotkách procent.

6.1.4. Srovnání s předlohou



(a) (b)

Obrázek 6.2: Srovnání cest nalezených algoritmem MSAC z článku [41] (a) s implementovaným algoritmem (b), scéna 1. Překážky jsou zobrazeny modře, bílá značí prostupné prostředí.



(a) (b)

Obrázek 6.3: Scéna 3, srovnání cest nalezených algoritmem MSAC z článku [41] (a) s implementovaným algoritmem (b). Překážky jsou zobrazeny modře, bílá značí prostupné prostředí. V části (b) je oblast prozkoumaná první skupinou zobrazena žlutě, oblast prozkoumaná druhou skupinou je znázorněna tmavě červěně. Konečná pozice mravenců je označena tučnými černými tečkami.

Výsledné cesty nalezené pro scénu 1 znázorněné na obrázku 6.2 jsou téměř shodné. Liší se pouze v řazení diagonálního pohybu na počátku a konci cesty, což nemá vliv na její výslednou délku. Oproti tomu cesty nalezené pro scénu 3, zobrazené na obrázku 6.3, jsou odlišné. Největší odchylka se vyskytuje v pravé části (sloupec 50), kde algortimus předlohy nalezl přímou cestu bez vybočení.

Nalezení takové cesty nebylo dosaženo žádným nastavením parametrů. Možným způsobem pro dosažení obdobného řešení by bylo použít složitější zpracování cesty a vyhlazovat ji ve větším dosahu než pouze v nejbližším okolí. Článek [41] se však o žádném zpracování cesty nezmiňuje.

Doba, kterou algoritmus MSAC potřebuje k nalezení cesty, je v [41] uvedená jako nižší než 0,01 s ve všech případech, kdy je velikost mřížky menší než 50×50. Této rychlosti implementovaný algoritmus nedosahuje. Pro scénu 1 se doba potřebná pro nalezení cesty pohybuje kolem 0,1 s – podle toho, zda je zapnuté zpracování cest. Pro dosažení maximální rychlosti je nutné minimalizovat parametry $MAX = 1$ a $m = 1$. Po této minimalizaci je průměrná doba pro nalezení cesty rovna 0,0176 s při zpracování cest, bez zpracování cest je průměrná doba 0,00839 s. Tento přístup pro složitější scény selhává. Nízký počet mravenců a pouze jedno opakování vnejší smyčky nestačí pro nalezení cesty přibližně v 30 % případů (scéna 2).

Nižší dosažená rychlost je dána implementací v interpretovaném jazyce MATLAB. Přesto bylo dosaženo významného relativního zrychlení implementovaného algoritmu díky úpravě tabu tabulky. Dále došlo ke zkvalitnění generovaných cest pomocí jejich zpracování (odstranění cyklů a mírného vyhlazení).

6.2. SACOdm

Část srovnává implementaci algoritmu vůči výsledkům uvedeným v článku [12]. Ve všech experimentech byla počáteční hodnota feromonu zvolena 0,5.

6.2.1. Vliv koeficientu β

Účelem experimentu je srovnání výsledků algoritmu SACO oproti SACOdm. Algoritmus SACO je speciálním případem nastavení algoritmu SACOdm. Jedná se o případ, kdy $\beta = 0$ a $\gamma = 0$.

Podmínky jsou následující. Prostředím je volná scéna 50×50. Počet mravenců je 3, iterací 10, míra vypařování $\rho = 0,2$; $\alpha = 2$. Parametry β a γ jsou nastaveny na 0. Toto nastavení odpovídá základnímu algoritmu SACO.

Pro druhý běh experimentu použijeme stejné hodnoty, jen změníme hodnotu $\beta = 1$, tzn. dáme váhu pro preferenci uzlů, které se nacházejí blíže k cílovému uzlu. Zde už se jedná o algoritmus SACOdm.

β	0	1
Průměrný čas [s]	109,97	15,79
Průměrná délka cesty	97,02	95,17
Minimální délka cesty	88,81	87,88

Tabulka 6.8: Vliv parametru β, volná scéna, vzdálenost startu a cíle 65,05.

Podle očekávání váha přidaná uzlům blíže k cíli ve volné scéně vede k rychlejší konvergenci. Pro hodnotu $\beta = 1$ je průměrný čas nalezení cesty téměř sedmkrát menší. Podobně průměrná i minimální délka nalezené cesty byla pro případ $\beta = 1$ v obou srovnáních nepatrně lepší, relativní rozdíl průměrných i minimálních délek je menší než 2 %.

Následuje srovnání časové náročnosti s výsledky uvedenými v [12]: 53,99 s pro $\beta = 0$ a 4,95 s pro $\beta = 1$. Pro hodnotu $\beta = 0$ je průměrný čas potřebný k nalezení cesty více než dvojnásobný, pro $\beta = 1$ více než trojnásobný. Délku cest nelze srovnat, protože ve zmíněném článku jsou nalezené cesty ohodnoceny cenou udanou fuzzy funkcí, nikoli délkou cesty.

6.2.2. Vliv typu koeficientu γ

Experiment srovnává typ využití koeficientu γ. Nastavení parametrů je následující.

Počet mravenců je 3, iterací 15, míra vypařování $\rho = 0,2$, $\alpha = 2$, $\beta = 1$ a $\gamma = 7$. V prvním běhu je typ γ dle článku [12], tj. na počátku hledání má mravenec zakázáno se vrátit do prvních γ prošlých uzlů.

Pro druhý běh použijeme stejné hodnoty, ale změníme typ využití γ na průběžný seznam, tj. mravenec se v průběhu celého svého pohybu nesmí vrátit do průběžně posledních γ navštívených uzlů.

Typ γ	Prvních γ	Průběžný seznam γ
Průměrný čas [s]	32,83	18,99
Průměrná délka cesty	93,48	95,17
Minimální délka cesty	88,22	90,23

Tabulka 6.9: Vliv typu γ, scéna 1, $\gamma = 7$.

Z výsledku experimentu je patrné, že použití průběžného seznamu vede k rychlešímu nalezení cesty, ale na úkor kvality. Doba potřebná k nalezení cesty je o přibližně 42 % kratší. Oproti tomu průměrná délka i minimální délka cesty jsou delší, ale ne o více než 2,5 %.

Čas potřebný pro nalezení cesty je ve srovnání s výsledky uvedenými v [12] (12,14 s) pro typ prvních γ zakázaných uzlů více než dvaapůlkrát větší, pro typ průběžného seznamu je více než jedenapůlkrát větší.

6.2.3. Vliv typu koeficientu γ, jiná volba parametrů

Experiment je zopakováním předchozího případu s jinou volbou parametrů: 3 mravenci, 5 iterací, $\rho = 0,2$, $\alpha = 2$, $\beta = 1$ a $\gamma = 1$.

Typ γ	Prvních γ	Průběžný seznam γ
Průměrný čas [s]	10,89	9,26
Průměrná délka cesty	101,20	100,50
Minimální délka cesty	96,12	90,23

Tabulka 6.10: Vliv typu γ, scéna 1, $\gamma = 1$.

Pro nastavení $\gamma = 1$ je průběžný seznam o více než 17 % rychlejší, nalezená cesta je kratší o méně než 1 %. Průměrný čas pro nalezení cesty je v obou případech nižší než uvedený v [12] (12,08 s).

6.2.4. Vliv typu aktualizace feromonu

Experiment zkoumá vliv typu aktualizace feromonu s následujícími parametry: 3 mravenci, 5 iterací, $\rho = 0,2$, $\alpha = 2$, $\beta = 1$ a $\gamma = 1$, typ γ je průběžný seznam. Zkoumané typy jsou *Ant system* (AS) a *Elitist ant system* (EAS).

Typ aktualizace	AS	EAS	EAS
Váha nejlepší cesty	0	1	2
Průměrný čas [s]	8,85	8,74	8,63
Průměrná délka cesty	101,13	99,64	101,49
Minimální délka cesty	94,95	94,71	86,57

Tabulka 6.11: Vliv typu aktualizace feromonu, 5 iterací, scéna 1.

Průměrný čas klesá se zvyšující se vahou průběžné nejlepší cesty, nejdelší průměrný čas je oproti nejkratšímu o 2,5 % větší. Průměrná délka cesty byla nejmenší pro hodnotu váhy 1, největší pro váhu 2, rozdíl 1,8 %. Rozdíl pro váhy 0 a 2 je 0,4 %. Minimální délka cesty byla nalezena pro váhu 2, další v pořadí jsou o více než 9 % delší. Experiment byl zopakován pro 10 iterací:

Typ aktualizace	AS	EAS	EAS
Váha nejlepší cesty	0	1	2
Průměrný čas [s]	16,89	15,54	17,48
Průměrná délka cesty	95,39	94,92	98,04
Minimální délka cesty	89,05	87,98	91,30

Tabulka 6.12: Vliv typu aktualizace feromonu, 10 iterací, scéna 1.

Oproti předchozí situaci je nyní nejkratší průměrný čas pro hodnotu váhy 1, nejdelší pro váhu 2, který je delší o 12,5 %. Nejkratší průměrná délka cesty je opět pro hodnotu váhy 1 a nejdelší pro váhu 2, rozdíl činí 3,3 %. Minimální délky cest jsou seřazeny ve stejném pořadí. Rozdíl nejnižší a nejvyšší hodnoty je 3,8 %.

Příčinou vyšších hodnot průměrné délky cesty pro váhu 2 je pravděpodobně předčasná konvergence, kdy je po několik iterací stejná průběžná nejlepší cesta. To v kombinaci s vysokou váhou pro tuto cestu vede k příliš velkému nárůstu hodot feromonu na této cestě a znemožňuje nalezení lepších cest.

6.2.5. Vliv počtu mravenců

Hodnoty vstupních parametrů: 5 iterací, $\rho = 0,2$, $\alpha = 2$, $\beta = 1$ a $\gamma = 1$, typ γ je průběžný seznam, typ aktualizace feromonu je EAS s hodnotou váhy pro nejlepší cestu 1.

Mravenců	2	4	6
Průměrný čas [s]	5,99	10,74	17,29
Průměrná délka cesty	102,84	100,19	94,86
Minimální délka cesty	89,98	95,30	91,30

Tabulka 6.13: Vliv počtu mravenců, 5 iterací, scéna 1.

Průměrná doba potřebná pro nalezení cesty roste se vzrůstajícím počtem mravenců, průměrná délka cesty naopak klesá, což jsou očekávané výsledky.

Stejný scénář byl zopakován pro 10 iterací:

Mravenců	2	4	6
Průměrný čas [s]	11,34	22,68	34,02
Průměrná délka cesty	98,51	95,34	94,01
Minimální délka cesty	87,98	89,39	85,74

Tabulka 6.14: Vliv počtu mravenců, 10 iterací, scéna 1.

S navýšeným počtem iterací se na trendu potřebného průměrného času a průměrné délky cesty dle očekávání nic nezměnilo.

6.2.6. Uzavřená scéna, vliv parametru β

Experiment zkoumá výsledky algoritmu na uzavřené scéně s následujícím nastavením parametrů: 5 iterací, 3 mravenci, $\rho = 0,2$; $\alpha = 2$ a $\gamma = 100$, typ γ je průběžný seznam, typ aktualizace feromonu je EAS s hodnotou váhy pro nejlepší cestu 1. Nastavení vysoké hodnoty parametru gama do jisté míry brání ustrnutí v lokálním minimu.

β	0	1
Průměrný čas [s]	34,87	29,35
Průměrná délka cesty	153,43	153,54
Minimální délka cesty	144,41	138,65

Tabulka 6.15: Vliv parametru β, scéna 2.

Průměrná doba potřebná k nalezení cesty byla pro hodnotu $\beta = 0$ delší o 18,8 %. Relativní rozdíl průměrné délky cesty je zanedbatelný.

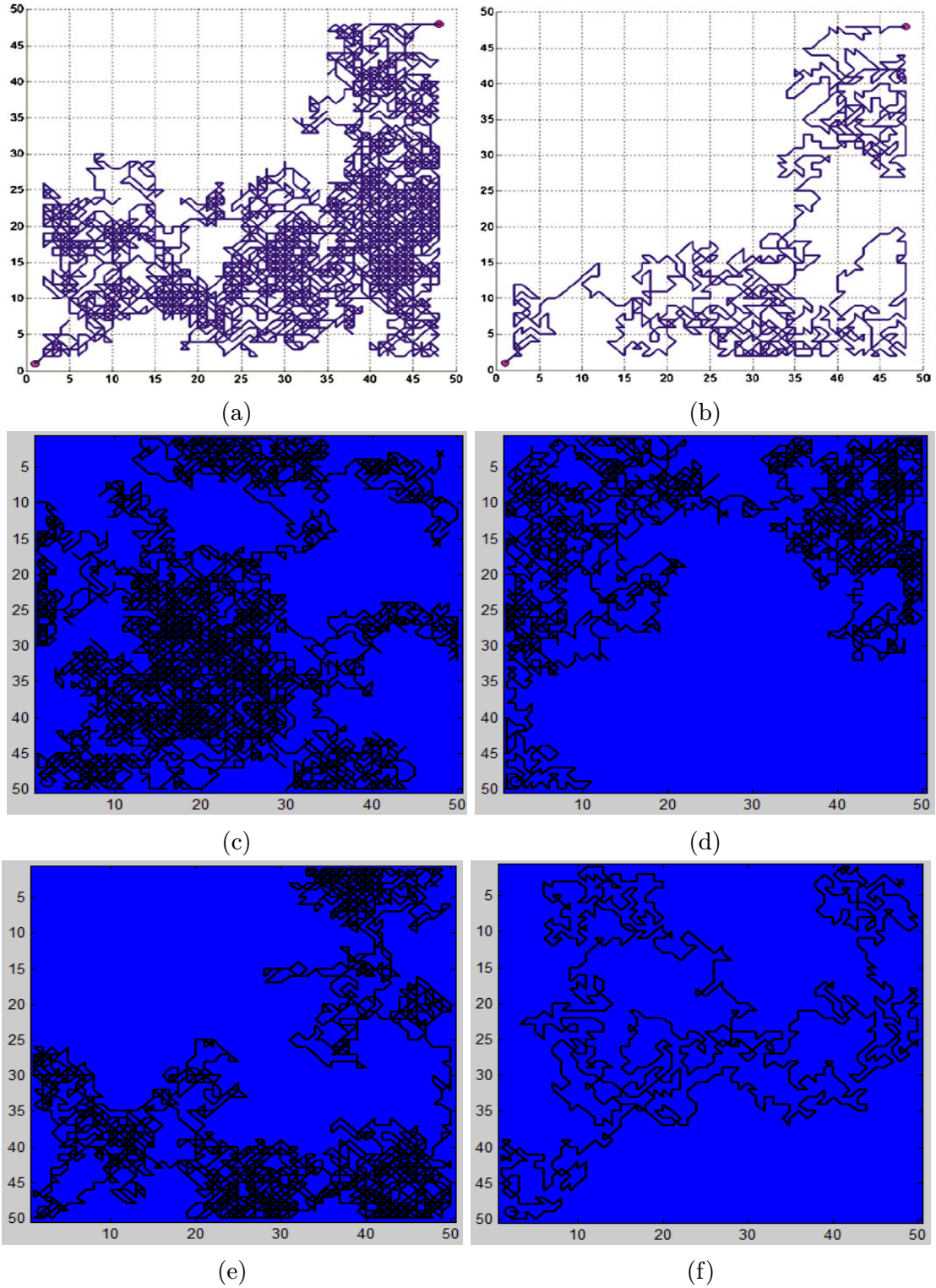
Experiment byl zopakován pro hodnotu $\gamma = 50$:

β	0	1
Průměrný čas [s]	36,04	41,39
Průměrná délka cesty	153,49	146,55
Minimální délka cesty	142,65	132,41

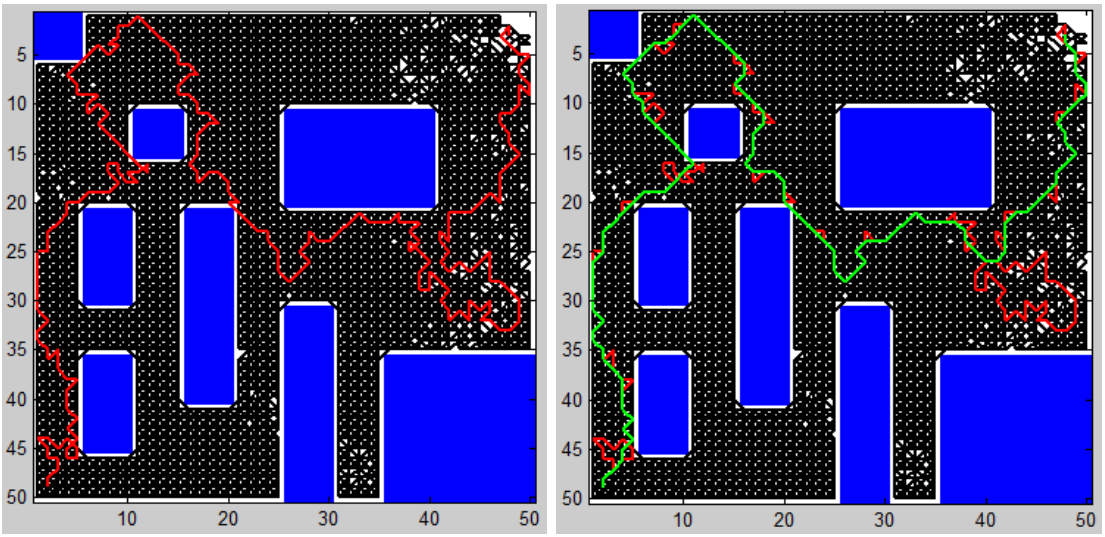
Tabulka 6.16: Vliv parametru β, scéna 2.

Průměrná délka nalezených cest je pro případy $\gamma = 100$ (bez ohledu na hodnotu β) a $\gamma = 50$ pro případ $\beta = 0$ téměř stejná. Zbývající případ $\gamma = 50$ a $\beta = 1$ je o 4,5 % kratší.

Výsledky ukazují, že volba parametrů má vliv převážně na dobu potřebnou pro nalezení cesty, zejména parametry β a γ (typ průběžného seznamu). Průměrná délka cesty se za pomoci parametrů mění v jednotkách procent. S navýšením počtu mravenců a iterací lze snížit průměrnou délku nalezených cest výměnou za delší dobu hledání.



Obrázek 6.4: Srovnání cest vygenerovaných prvním mravencem v první iteraci ve volné scéně, hodnoty parametrů jsou pro případy (a), (c), (e) $\beta = 0$ a $\gamma = 1$, pro případy (b), (d), (f) $\beta = 0$ a $\gamma = 100$. (a), (b) je převzato z [12]. (c), (d) je vygenerováno implementovaným algoritmem, kdy je typ seznamu γ shodný s [12]. (e), (f) je vygenerováno s průběžným seznamem γ. Z porovnání je vidět, že cesty vytvořené s vysokou hodnotou parametru γ (v pravém sloupci) mají tendenci méně tvořit cykly.



(a) (b)

Obrázek 6.5: Odstranění cyklů. (a) Černou čarou je znázorněna cesta s extrémním počtem cyklů (délka 58 895,5), vygenerovaná algoritmem SACOdm s parametry $\beta = 0$ a $\gamma = 0$ v první iteraci. Červenou čarou je znázorněna tatáž cesta po odstranění cyklů (délka 257,6). (b) Zelenou čarou je cesta po lehkém vyhlazení (délka 153,5).

6.3. Srovnání implementovaných algoritmů MSAC a SACOdm

Ve volné scéně nalezl algoritmus MSAC vždy nejkratší cestu délky 65,05 v časech řádu desetin sekundy. Nejlepší výkon algoritmu SACOdm byl na volné scéně následující: průměrná délka 95,17; minimální délka 87,88 v čase 15,79 s.

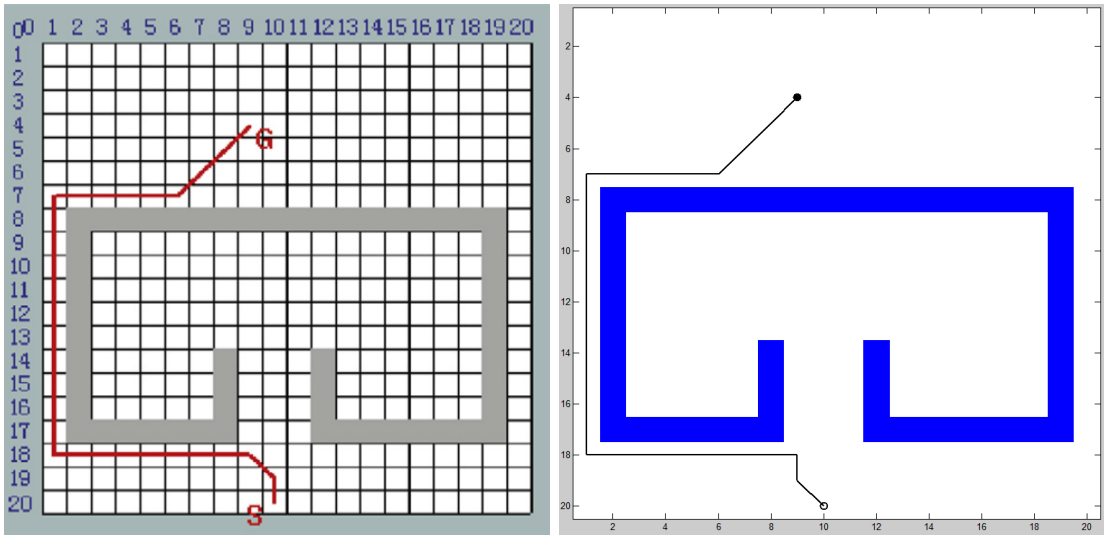
Ve scéně 1 nalezl algoritmus MSAC také vždy stejnou délku cesty, a to 73,84 opět v časech nepřesahujících 1 s. Nejlepší výsledek algoritmu SACOdm ze scény 1 dosáhl průměrné délky 94,92; minimální délky 87,98 v čase 15,54 s. Časová náročnost se pohybuje v jednotkách až desítkách sekund v závisloti na počtu iterací a mravenců.

Pro scénu 2 jsou výsledky algoritmu MSAC následující: nejlepší průměrná délka cesty 146,95; minimální nalezená dálka 136,75; čas nalezení cesty se pohyboval řádově v jednotkách sekund. SACOdm má pro scénu 2 nejlepší výsledek: minimální délka 132,41 a průměrná délka 146,55 v čase 41,39 s.

Ve volné scéně a ve scéně 1 jednoznačně dominuje algoritmus MSAC, který vždy nalezne nejkratší cestu a v mnohem kratší době než SACOdm. Pro scénu 2 byl ohledně délky nalezené cesty těsně úspěšnější algoritmus SACOdm, z hlediska délky hledání byl opět lepší algoritmus MSAC.

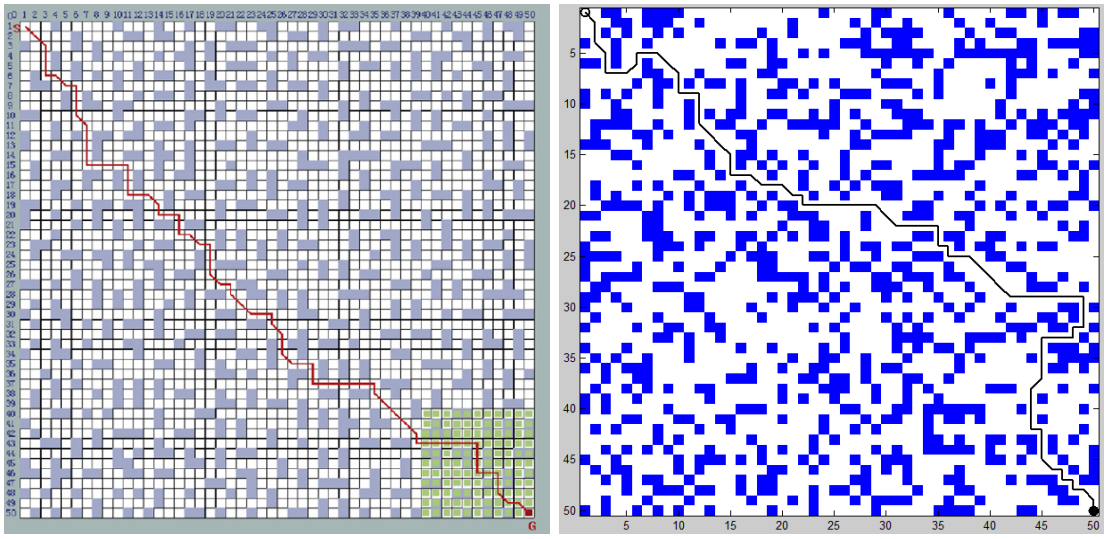
6.4. RNA

Srovnání výsledků scén ze [41] vůči implementaci.



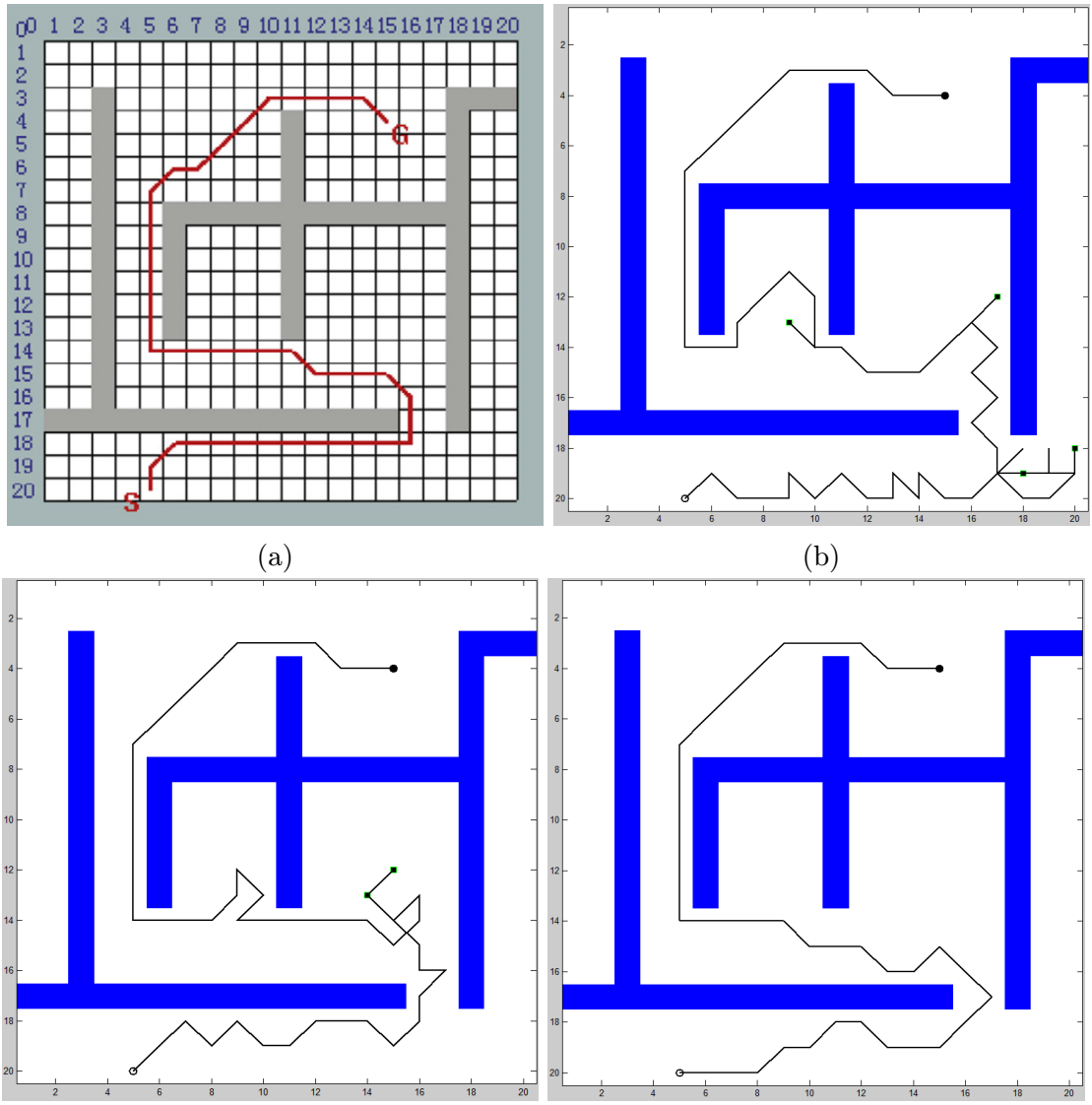
(a) (b)

Obrázek 6.6: Srovnání cest nalezených pomocí: (a) algoritmu RNA dle [41]; (b) implementovaného algoritmu RNA, dohled 10. Počáteční pozice je označena prázdným kroužkem, cílová pozice robotu výplněným černým kroužkem. Modrá barva značí překážku.



(a) (b)

Obrázek 6.7: Srovnání cest přes odlišná náhodná prostředí nalezených pomocí: (a) algoritmu RNA dle [41]; (b) implementovaného algoritmu RNA, dohled 10. Startovní bod v levém horním rohu, cíl v pravém dolním rohu.

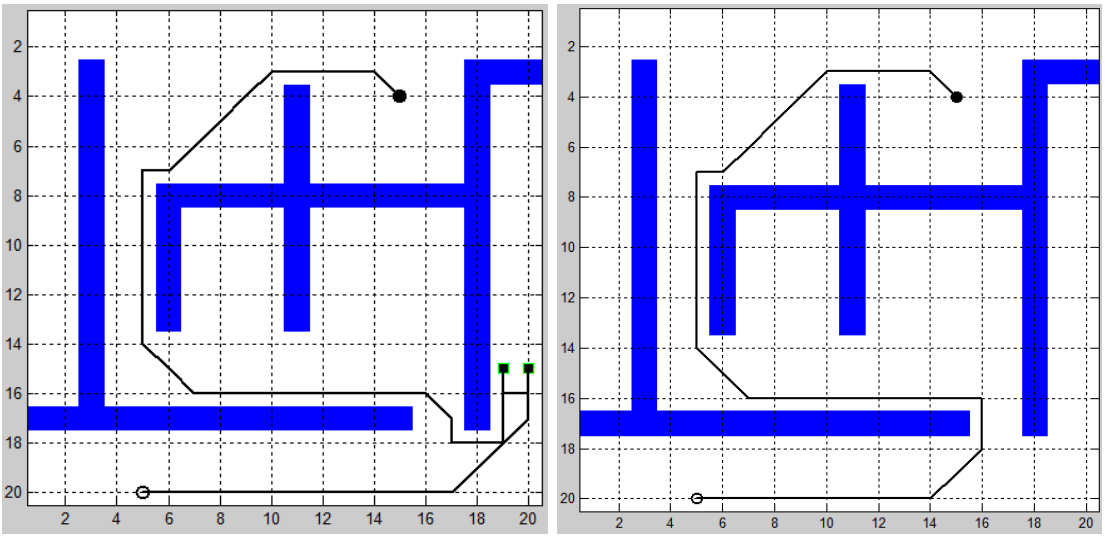


(a) (b)

(c) (d)

Obrázek 6.8: Srovnání cest nalezených pomocí: (a) algoritmu RNA dle [41] a implementovaného algoritmu s dohledem (b) 15; (c) 16; (d) 18.

Z obrázku 6.8 je patrné, že kvalita nalezené cesty pomocí algoritmu RNA narůstá se zvyšujícím se dohledem. Přesto i v případě (d), kdy byl dohled dostatečný pro dohlédnutí na všechny překážky, dochází ke zbytečným pohybům. Tyto pohyby jsou způsobeny omezenou mírou vyhlazení cesty v implementaci algoritmu MSAC. Pro ověření funkčnosti algoritmu RNA byla implementována varianta WF-RNA, WF je zkratkou pro *wave front planner*. WF je obdobou prohledávání do šířky pro prostředí diskretizované do mřížky, které dokáže nalézt nejkratší cestu (za cenu prohledání celého prostoru [3]). Právě pomocí WF bylo nahrazeno lokální prohledávání v algoritmu RNA.



(a) (b)

Obrázek 6.9: Cesty nalezené pomocí varianty WF-RNA s dohledem (a) 12; (b) 17. V případě (a) zkoušel robot projít slepou uličkou, když zjistil, že nemůže projít, udělal krok zpět a označil místo, odkud se vrátil, dočasnou překážkou (zeleně ohraničený čtverec). Vykonáním kroku zpět ale opět ztratil konec uličky z dohledu a znovu zkusil projít. Až po umístění druhé dočasné překážky robot našel správnou cestu. Robot se pohybuje optimálně pouze v případě, když má v dohledu celou konkávní překážku, které se má vyhnout.

7. Závěr

Cílem práce bylo analyzovat přístupy k plánování cesty robotu, popsat metody rojové inteligence a jejich aplikace na plánování cesty robotu. Dalším úkolem bylo navrhnout a implementovat systém pro plánování cesty robotu pomocí mravenčích algoritmů a na implementovaném systému provést ověřovací a srovnávací experimenty.

Rešeršní část popisuje obecné způsoby plánování cesty robotu, metody rojové inteligence a jejich aplikace na plánování cesty. Praktická část se zabývá návrhem systému pro plánování cesty robotu pomocí mravenčích algoritmů. Cílem bylo implementovat algoritmy z literatury a navrhnout možná vylepšení.

Byl implementován upravený algoritmus RNA, což je algoritmus pro plánování cesty v dynamickém, částečně neznámém prostředí. Byl rozšířen o plánování pro více dynamických překážek v oblasti dohledu. Byla ošetřena situace, kdy cesta není nalezena ve fázi plánování za předpokladu statického okolí. Pokud není cesta statickým prostředím nalezena po zadaný počet iterací, program skončí s upozorněním na vzniklou situaci.

K výsledkům prezentovaným v článku [41], kde je popsán původní algoritmus RNA, byla doplněna podmínka pro optimální chování robotu. Tato podmínka je taková, že dohled robotu musí být větší, než je velikost překážky ohraničující oblast s lokálním minimem.

Lokální hledání algoritmu RNA zajišťuje algoritmus MSAC. Tento multiagentový mravenčí systém využívá dopředného a zpětného hledání, kdy agenti sdílejí znalosti pomocí tabu tabulky a jednotlivě využívají hladovou strategii.

Pro implementaci algoritmu bylo navrženo několik úprav. Prvním návrhem byla změna využití tabu tabulky za účelem omezení slepého hledání při detekci setkání agentů z odlišných skupin algoritmu. Výsledky experimentů ukázaly výrazné zrychlení původní implementace. Dalším návrhem bylo zpracování generovaných cest před jejich porovnáváním s průběžně nejkratší nalezenou cestou (odstranění cyklů a mírné vyhlazení). Využití tohoto způsobu sice navyšuje časovou náročnost algoritmu, ale hlavně pro uzavřenou scénu s výskytem lokálních minim lze vidět výrazné zlepšení kvality výsledné cesty. Naopak pro relativné otevřené scény není navýšení časové náročnosti vyváženo kvalitativním zlepšením. Posledním návrhem byl přenos informací o slepých uličkách mezi iteracemi hledání.

Oproti výsledkům z článku [41] jsou nalezené cesty pro scény s výskytem lokálních minim odlišné. Pro otevřenou scénu jsou výsledné cesty srovnatelné. Jedním z výstupů práce je otázka, zda algoritmus MSAC generuje nejkratší cestu ve složitém prostředí s lokálními minimy bez dalšího zpracování cest.

Druhým implementovaným mravenčím algoritmem pro hledání cesty ve známém prostředí je SACOdm. Oproti návrhu z článku [12] byl upraven způsob ohodnocení cest. Namísto fuzzy přístupu byla použita varianta ohodnocení pouze na základě délky cesty. Byla navržena jiná varianta seznamu zakázaných polí. Původní seznam zakazoval mravencům vrátit se zpět pouze na počátku hledání. Navrženou změnou pro generování cest bylo použití průběžného zakázaného seznamu polí, tzn. že mravenci se nesmějí vrátit do několika posledních prozkoumaných polí v průběhu celého hledání. Zmíněný návrh urychluje nalezení cesty a zabraňuje ustrnutí v lokálním minimu.

Byl ověřen urychlující vliv parametru β, váhy pro hladovou strategii, na čas nalezení cesty a její kvalitu ve volné scéně. Dále byl zjištěn významný urychlující vliv parametru γ při použití typu průběžného seznamu.

Ve srovnání s výchozími algoritmy popsanými v literatuře jsou prezentované implementace pomalejší. To je dáno pravděpodobně použitím interpretovaného jazyka MATLAB.

Porovnáním algoritmů MSAC a SACOdm se ověřil výsledek ze [41], že algoritmus MSAC, který nevyužívá feromony, nalezne cestu mnohem rychleji než algoritmus SACOdm.

Možností pro rozšíření práce by bylo upravit plánování cesty pro neholonomní robot. Dalším možným rozšířením by byla implementace fuzzy hodnotící funkce pro algoritmus SACOdm. Obecnou možností pro metody rojové inteligence je optimalizace parametrů, případně jejich adaptivní nastavování dle vlastností scény.

Literatura

[1] AERCam Sprint. *NASA* [online]. 3-30-2008 [cit. 2014-05-08]. Dostupné z: http://aercam.jsc.nasa.gov/sprint/index.asp

[2] BONABEAU, E. *Swarm intelligence*. New York: Univerzity Press, 1999, 307 s. ISBN 01-951-3159-2.

[3] CHOSET, Howie. *Principles of robot motion: theory, algorithms and implementations*. Massachusetts: MIT Press, 2005, xix, 603 s. ISBN 02-620-3327-5.

[4] CHOSET, Howie. *Robotic Motion Planning: Cell Decompositions* [online]. 16--Jul-2007 [cit. 2014-05-23]. Dostupné z: http://www.cs.cmu.edu/ motionplanning/lecture/Chap6-CellDecomp_howie.pdf

[5] CHOSET, Howie. *Robotic Motion Planning: Potential Functions* [online]. 16--Jul-2007 [cit. 2014-05-25]. Dostupné z: http://www.cs.cmu.edu/ motionplanning/lecture/lecture%5cChap4-Potential-Field.pdf

[6] CHOSET, Howie. *Robotic Motion Planning: Roadmap Methods* [online]. 16--Jul-2007 [cit. 2014-05-23]. Dostupné z: http://www.cs.cmu.edu/ motionplanning/lecture/Chap5-RoadMap-Methods_howie.pdf

[7] CURKOVIC, P. Honey-bees optimization algorithm applied to path planning problem. *International Journal of Simulation Modelling* [online]. 2007-9-15, vol. 6, issue 3, s. 154-164 [cit. 2014-05-18]. DOI: 10.2507/IJSIMM06(3)2.087. Dostupné z: http://www.ijsimm.com/Full_Papers/Fulltext2007/text6-3_154-164.pdf

[8] Da Vinci Surgical System. *Da Vinci Surgery* [online]. © 2014 [cit. 2014-05-08]. Dostupné z: http://www.davincisurgery.com/da-vinci-surgery/da-vinci-surgical-system/

[9] DORIGO, Marco. *Ant colony optimization*. Cambridge: MIT Press, 2004, xiv, 305 s. ISBN 02-620-4219-3.

[10] Facts About Landmines. *CARE* [online]. © 2013 [cit. 2014-05-08]. Dostupné z: http://www.care.org/emergencies/facts-about-landmines

[11] Gamma Knife radiosurgery Definition. *Mayo Clinic* [online]. May. 23, 2013 [cit. 2014-05-08]. Dostupné z: http://www.mayoclinic.org/tests-procedures/gamma-knife--radiosurgery/basics/definition/prc-20014760

[12] GARCIA, M.A. Porta, Oscar MONTIEL, Oscar CASTILLO, Roberto SEPÚLVEDA a Patricia MELIN. Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation. In: *Applied Soft Computing* [online]. 2009, s. 1102-1110 [cit. 2014-05-14]. ISSN 15684946. DOI: 10.1016/j.asoc.2009.02.014. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S1568494609000349

[13] Goals - Mars Science Laboratory. *NASA* [online]. [2011] [cit. 2014-05-08]. Dostupné z: http://mars.jpl.nasa.gov/msl/mission/science/goals/

[14] GROSS, T. *Plánování cesty mobilního robota* [online]. Brno, Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2007 [cit. 2014-05-21]. Dostupné z: http://autnt.fme.vutbr.cz/szz/2007/DP_Gross.pdf. Diplomová práce. Vedoucí práce RNDr. Jiří Dvořák, CSc.

[15] KATSIKOPOULOS, K. V. a G. GIGERENZER. Behavioral operations management: a blind spot and a research program. *The Free Library* [online]. 2013, [cit. 2014-05-19]. Dostupné z: http://www.thefreelibrary.com/Behavioral operations management: a blind spot and a research program-a0318910311

[16] LAVALLE, Steven M. Motion Planning: The Essentials. *IEEE Robotics and Automation Society Magazine* [online]. 2011, vol. 18, č. 1, s. 79-89 [cit. 2014-05-18]. Dostupné z: http://msl.cs.uiuc.edu/ lavalle/papers/Lav11b.pdf

[17] LIU, Chang, Zhongqiang GAO a Weihua ZHAO. A New Path Planning Method Based on Firefly Algorithm. *2012 Fifth International Joint Conference on Computational Sciences and Optimization* [online]. IEEE, 2012, s. 775-778 [cit. 2014-05-18]. DOI: 10.1109/CSO.2012.174. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6274838

[18] MANIKAS, Theodore, Kaveh ASHENAYI, Roger WAINWRIGHT. Genetic algorithms for autonomous robot navigation. *IEEE Instrumentation* [online]. 2007, vol. 10, issue 6, s. 26-31 [cit. 2014-05-21]. DOI: http://dx.doi.org/10.12681/eadd/15018. Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04428579

[19] MIRJALILI, Seyedali, Seyed Mohammad MIRJALILI a Andrew LEWIS. Grey Wolf Optimizer. In: *Advances in Engineering Software* [online]. 2014, s. 46-61 [cit. 2014-05-10]. ISSN 09659978. DOI: 10.1016/j.advengsoft.2013.12.007. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0965997813001853

[20] NASROLLAHY, Amin Zargar a Hamid Haj Seyyed JAVADI. Using Particle Swarm Optimization for Robot Path Planning in Dynamic Environments with Moving Obstacles and Target. *2009 Third UKSim European Symposium on Computer Modeling and Simulation* [online]. IEEE, 2009, s. 60-65 [cit. 2014-05-18]. DOI: 10.1109/EMS.2009.67. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5358824

[21] Project Summary. *Metaheuristics Network* [online]. August 2004 [cit. 2014-05-20]. Dostupné z: http://www.metaheuristics.org/index.php?main=1

[22] SAFFARI, M. H. a M. J. MAHJOOB. Bee colony algorithm for real--time optimal path planning of mobile robots. *2009 Fifth International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control* [online]. IEEE, 2009, s. 1-4 [cit. 2014-05-18]. DOI: 10.1109/ICSCCW.2009.5379462. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5379462

[23] STROHL, Daniel. Jay Leno adds Mars Curiosity rover to his collection. *Hemmings Daily* [online]. 2013 [cit. 2014-05-25]. Dostupné z: http://blog.hemmings.com/index.php/2013/04/01/jay-leno-adds-mars-curiosity--rover-to-his-collection/

[24] *Summerlin Hospital Medical Center* [online]. © 2014 [cit. 2014-05-25]. Dostupné z: http://www.summerlinhospital.com/

[25] SVIDENSKÁ, A. *RRT plánování pohybu robota aplikovaná ve 2D prostoru* [online]. Brno, Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2007 [cit. 2014-05-21]. Dostupné z: http://autnt.fme.vutbr.cz/szz/2007/BP_Svidenska.pdf. Bakalářská práce. Vedoucí práce Ing. Radomil Matoušek Ph.D.

[26] ŠEDA, Miloš. *TEORIE GRAFŮ* [online]. Brno, 2003. Dostupné z: http://www.uai.fme.vutbr.cz/ mseda/TG03_MS.pdf

[27] ŠÍMA, M. *Plánování cesty robota ve spojitém prostředí* [online]. Brno, Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2006 [cit. 2014-05-21]. Dostupné z: http://autnt.fme.vutbr.cz/szz/2006/DP_Sima.pdf. Diplomová práce. Vedoucí práce RNDr. Jiří Dvořák, CSc.

[28] ŠINDELÁŘ, J. *Plánování cesty neholonomního mobilního robotu* [online]. Brno, Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2010 [cit. 2014-05-21]. Dostupné z: http://autnt.fme.vutbr.cz/szz/2010/DP_Sindelar.pdf. Diplomová práce. Vedoucí práce Ing. Petr Krček

[29] ŠVEC, P. *Plánování trasy robota pomocí Voroného diagramů a dalších prostředků výpočetní geometrie.* [online]. Brno, Masarykova univerzita v Brně, Fakulta informatiky, 2006 [cit. 2014-05-21]. Dostupné z: http://is.muni.cz/th/39233/fi_m/diplomka_Petr_Svec.pdf. Diplomová práce. Vedoucí práce Mgr. Petr Tobola, Ph.D.

[30] TAN, Guan-Zheng. Ant Colony System Algorithm for Real-Time Globally Optimal Path Planning of Mobile Robots. *ACTA AUTOMATICA SINICA* [online]. 2007, vol. 33, issue 3, s. 0279- [cit. 2014-04-10]. DOI: 10.1360/aas-007-0279. Dostupné z: http://www.aas.net.cn/qikan/epaper/zhaiyao.asp?bsid=12823

[31] TANG, Rui, Simon FONG, Xin-She YANG a Suash DEB. Wolf search algorithm with ephemeral memory. *Seventh International Conference on Digital Information Management (ICDIM 2012)* [online]. IEEE, 2012, s. 165-172 [cit. 2014-05-18]. DOI: 10.1109/ICDIM.2012.6360147. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6360147

[32] TUNCER, Adem a Mehmet YILDIRIM. Dynamic path planning of mobile robots with improved genetic algorithm. *Computers* [online]. 2012, vol. 38, issue 6, s. 1564-1572 [cit. 2014-05-22]. DOI: 10.1016/j.compeleceng.2012.06.016. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0045790612001255

[33] VESELOVSKÝ, M. *Plánování cesty robotu pomocí posilovaného učení* [online]. Brno, Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2013 [cit. 2014-05-21]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=66162. Diplomová práce. Vedoucí práce RNDr. Jiří Dvořák, CSc.

[34] WANG, Gaige, Lihong GUO, Hong DUAN, Luo LIU a Heqi WANG. A Bat Algorithm with Mutation for UCAV Path Planning. *The Scientific World Journal* [online]. 2012, vol. 2012, s. 1-15 [cit. 2014-05-18]. DOI: 10.1100/2012/418946. Dostupné z: http://www.hindawi.com/journals/tswj/2012/418946/

[35] WANG, Gaige, Lihong GUO, Hong DUAN, Luo LIU a Heqi WANG. A Modified Firefly Algorithm for UCAV Path Planning. *International Journal of Hybrid Information Technology* [online]. 2012, vol. 5, No. 3, s. 123-144 [cit. 2014-05-18]. Dostupné z: http://www.sersc.org/journals/IJHIT/vol5_no3_2012/11.pdf

[36] WINKLER, Zbyněk. Odometrie. *Robotika.cz* [online]. 2005, 2005-12-05 [cit. 2014-05-24]. Dostupné z: http://robotika.cz/guide/odometry/en

[37] What is CyberKnife. *CyberKnife* [online]. © 2008 - 2014 [cit. 2014-05-08]. Dostupné z: http://www.cyberknife.com/cyberknife-overview/what-cyberknife.aspx

[38] YANG, Xin-She. *Swarm intelligence and bio-inspired computation: theory and applications*. 1st ed. Boston: Elsevier, 2013, xxii, 422 p. Elsevier insights. ISBN 01-240-5163-4.

[39] TIAN, Yu, Lei YAN, Gun-Young PARK, Seung-Han YANG, Young-Suk KIM, Sang-Ryong LEE a Choon-Young LEE. Application of rrt-based local path planning algorithm in unknown environment. *Proceedings of the 2007 IEEE International Symposium on Computational Intelligence in Robotics and Automation* [online]. 2007, s. 456-460 [cit. 2014-05-21]. Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04269896

[40] ZHANG, Yong, Dun-wei GONG a Jian-hua ZHANG. Robot path planning in uncertain environment using multi-objective particle swarm optimization. *Neurocomputing* [online]. 2013, vol. 103, s. 172-185 [cit. 2014-05-18]. DOI: 10.1016/j.neucom.2012.09.019. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0925231212007722

[41] ZHU, Qingbao, Jun HU, Wenbin CAI a Larry HENSCHEN. A new robot navigation algorithm for dynamic unknown environments based on dynamic path re-computation and an improved scout ant algorithm. In: *Applied Soft Computing* [online]. 2011, s. 4667-4676 [cit. 2014-02-20]. ISSN 15684946. DOI: 10.1016/j.asoc.2011.07.016. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S1568494611002778