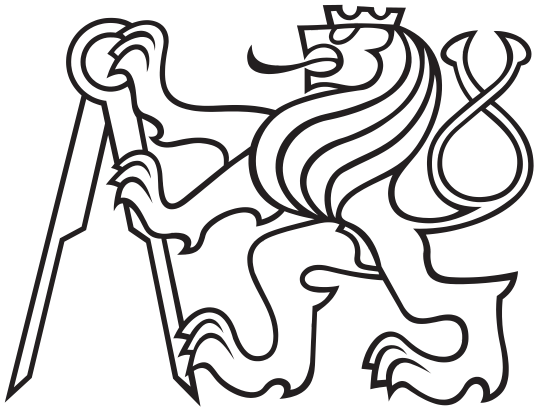
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická



BAKALÁŘSKÁ PRÁCE

Analýza použitelnosti pebble-motion algoritmů pro koordinaci trajektorií v multi-robotickém týmu

Jakub Vašek

Praha 2015

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Jakub V a š e k

Studijní program: Otevřená informatika (bakalářský)

Obor: Informatika a počítačové vědy

Název tématu: Analýza použitelnosti pebble-motion algoritmů pro koordinaci trajektorií v multi-robotickém týmu

Pokyny pro vypracování:

1. Seznamte se s algoritmy pro řešení pebble-motion problémů a s algoritmy pro koordinaci trajektorií v multi-robotickém týmu.
2. Navrhněte algoritmus, který bude řešit problém koordinace trajektorií v multi-robotickém týmu pomocí převodu na pebble-motion problém.
3. Analyzujte omezení navrženého algoritmu.
4. Navržený algoritmus naimplementujte.
5. Porovnejte vlastnosti (úspěšnost, kvalitu řešení a výpočetní nároky) navrženého algoritmu s vybranou existující technikou pro koordinaci trajektorií v multi-robotickém týmu.

Seznam odborné literatury:

[1] Chapter 7.2 -- Multiple Robots of Steven M. La Valle: Planning Algorithms. Cambridge University Press, 2006
[2] B. de Wilde, A. W. ter Mors and C. Witteveen: "Push and Rotate: a Complete Multi-agent Pathfinding Algorithm", JAIR, Volume 51, pages 443-492, 2014
[3] Glenn Wagner, Howie Choset: "Subdimensional expansion for multirobot path planning", Artificial Intelligence, Volume 219, February 2015, Pages 1-24, ISSN 0004-3702
[4] M. Čáp, P. Novák, A. Kleiner, and M. Selecký: "Prioritized planning algorithms for trajectory coordination of multiple mobile robots", IEEE transactions on automation science and engineering (in review), 2014

Vedoucí bakalářské práce: Bc. Michal Čáp, MSc.

Platnost zadání: do konce letního semestru 2015/2016

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 14. 1. 2015

Abstrakt

Cílem této práce je navrhnout algoritmus pro koordinaci trajektoriií v multi-robotickém systému. Navržený přístup je založen na převodu problému koordinace trajektorií na problém pebble-motion a následném vyřešení pebble-motion problému algoritmem Push&-Rotate. Výhodou tohoto přístupu je nižší časová náročnost. Při správné diskretizaci 2-d prostoru, kterou nazýváme jako přípustnou diskretizaci je ukázána korektnost algoritmu. Jeho kompletnost nám zajišťují takové diskretizace, které mají dva volné vrcholy a jsou zároveň přípustné. Algoritmus byl experimentálně porovnán v prázdném prostředí pomocí počítačové simulace. Experiment potvrzuje jeho kompletnost, ale ceny jeho trajektorií jsou až třikrát větší než u prioritizovaného plánování.

Abstract

The aim of this work is to design algorithm for coordination of trajectories in multi-robot system. Designed algorithm is based on the transfer from coordination of trajectories problem to pebble-motion problem and consequential solving pebble-motion problem with algorithm Push&Rotate. The advantage of this algorithm is lower time complexity. The correctness of the algorithm is shown on condition that we create an correct discretization of 2-d space which we called admissible discretization. The completeness of algorithm is guaranteed with admissible discretization with two free vertices in minimum. The algorithm was compared with prioritized planning in an environment without obstacles experimentally on the computer. The experiment confirms its completeness but costs of its trajectories are three times higher than costs in prioritized planning for 20 robots.

Prohlášení autora práce

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

V Praze dne..... .....

Podpis autora práce

Poděkování

Chtěl bych poděkovat vedoucímu mojí bakalářské práce Bc. Michalu Čápovi, Msc za jeho odborné rady, vstřícnost a velkou trpělivost během vytváření této práce.

Obsah

1 Úvod **6**

2 Pozadí problému **7**
2.1 Algoritmy pro koordinaci trajektorií v multi-robotickém týmu . . . 7
2.1.1 Centralizované multi-robotické plánování . . . 7
2.1.2 Prioritizované plánování v multi-robotických systémech . . . 7
2.2 Pebble-motion problém . . . 8
2.3 Push&Rotate . . . 8

3 Notace **10**

4 Definice problému **11**

5 Navržený algoritmus **12**
5.1 Diskretizace prostředí . . . 13
5.1.1 Požadavky na bezkoliznost v pebble-motion problému . . . 13
5.1.2 Požadavky na bezkoliznost v problému koordinace trajektorií . . . 13
5.1.3 Rozdíl diskretizace pro koordinace trajektorií a navržený přístup . . . 13
5.1.4 Přípustná diskretizace pro problém koordinace trajektorií . . . 14
5.1.5 Bezkonfliktnost hran se společným vrcholem . . . 16
5.1.6 Návrh optimálních parametrů . . . 17
5.1.7 Navržená diskretizace . . . 18
5.1.8 Analýza navrženého přístupu . . . 20
5.2 Aplikace Push&Rotate na diskretizované prostředí . . . 21
5.3 Paralelizace pohybu robotů . . . 21
5.4 Vytvoření trajektorií robotů . . . 23

6 Implementace **24**
6.1 Program pro diskretizaci prostředí . . . 24
6.1.1 Algoritmus na vyřešení problému v diskretizovaném prostředí . . . 25

7 Experimentální porovnání **26**
7.1 Kvalita řešení problému . . . 26
7.2 Rychlost řešení problému . . . 28

8 Závěr **31**

Kapitola 1

Úvod

V dnešní době se odvětví umělé inteligence snaží o vývoj továren nové generace, které usnadňují lidem práci. Jedná se například o velké sklady, kde se roboti posílají pro zboží na určitá místa, která jsou uložena v databázi. Problém koordinace trajektorií řeší to, aby se roboti nemohli srazit.
V tuto chvíli neexistuje žádný algoritmus, který by efektivně a zároveň garantovaně řešil problém koordinace trajektorií robotů. Existují metody, které mají exponenciální časovou náročnost v počtu koordinovaných robotů. Existují i metody, které mají nižší časovou náročnost, ty ale nevyřeší všechny instance. Zjdnodušenou variantou problému koordinace trajektorií je pebble-motion problém. Algoritmy řešící tento problém ovšem neumožňují současný pohyb agentů ani nám nezaručují *geometrickou bezkoliznost* (tj. těla robotů se nesmí během pohybu překrývat) v 2-d prostředí, protože tyto algoritmy jsou navrženy tak, aby nedocházelo ke *grafovým konfliktům* (tj. aby se dva roboti nikdy nenacházeli na stejném vrcholu daného grafu, případně pokud je výsledné řešení paralelizováno, aby necestovali v opačném směru po stejné hraně).
V této bakalářské práci jsem navrhl algoritmus, který řeší problém koordinace trajektorií pomocí převodu na pebble-motion problém tak, abyb byla zajištěna geometrická bezkoliznost výsledných trajektorií. Mnou navržený algoritmus funguje na základě vytvoření grafu, který zaručuje geometrickou bezkoliznost výsledných trajektorií v 2-d prostředí. Na tento graf se posléze pustí algoritmus pro řešení problému pebble-motion. Výsledek tohoto algoritmu potom změníme tak, aby se mohlo pohybovat více robotů zároveň a převedeme na n-tici trajektorií pro každého robota.
Ve druhé kapitole kapitole se dočteme o algoritmech řešících problém koordinace trajektorií a problém pebble-motion. Ve třetí a ve čtvrté kapitole se seznámíme s použitými symboly a zadefinujeme problém koordinace trajektorií, který budeme řešit. Pátá kapitola se zabývá samotným navržením algoritmu, převážně jeho omezeními, která jsou způsobena převodem problému koordinace trajektorií na pebble-motion problém. V šesté kapitole popíšeme implementaci navrženého algoritmu a seznámíme se s obsahem přiloženého CD. V sedmé kapitole experimentálně porovnáme vytvořený algoritmus s existujícím algoritmem (prioritizované plánování) pro řešení problému koordinace trajektorií. Nakonec v osmé kapitole shrneme výsledky této práce.

Kapitola 2

Pozadí problému

V této kapitole popíšeme některé známé přístupy pro koordinaci trajektorií v multirobotickém systému a zjednodušenou formulaci tohoto problému nazývaného pebble-motion problém.

2.1 Algoritmy pro koordinaci trajektorií v multi-robotickém týmu

Jedná se o problém, kdy je potřeba nalézt množinu trajektorií pro skupinu robotů. V prostředí se mohou vyskytovat překážky, se kterou nesmí mít agenti kolizi, zároveň se nesmí srazit s jiným agentem. A nyní již přikročíme k jednotlivým známým algoritmům.

2.1.1 Centralizované multi-robotické plánování

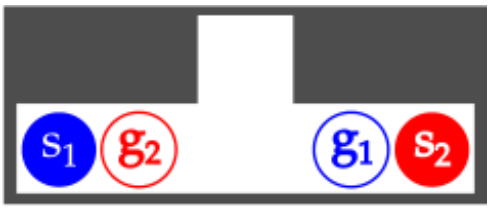
Jedním přístupem, kterým se řeší koordinace trajektorií v multi-robotickém týmu, je centralizované multi-robotické plánování [4, 5]. Tento algoritmus je založen na prohledávání kartézského součinu konfiguračních prostorů robotů:

$$J = S_1 \times S_2 \times ...S_n, \tag{2.1}$$

kde J je prostor, ve kterém se bude prohledávat, S_i je množina dosažitelných stavů agenta i a n je počet agentů. Tento algoritmus je sice kompletní a optimální, ale jeho nevýhodou je exponenciální časová složitost v počtu robotů, tudíž je pro více agentů z důvodu výpočetních nároků nepraktický.

2.1.2 Prioritizované plánování v multi-robotických systémech

Dalším algoritmem, který tento problém řeší, je prioritizované plánování [3, 6]. Prioritizovaném plánování určí každému agentovi jeho prioritu. Vezme agenta s největší prioritou a spočítá jeho cestu do cíle. Takto pokračuje se všemi agenty až k agentovi s nejnižší prioritou. Robot se přitom musí vyhýbat robotům, kteří mají svoji trajektorii již naplánovanou. Prioritizované plánování je mnohem rychlejší a tudíž i praktičtější pro více robotů než centralizované multi-robotické plánování, nicméně nemusí dojít k řešení, přestože existuje. Například ve scénáři na obrázku 2.1 prioritizované plánování nenajde řešení za žádné



Obrázek 2.1: Agent a_1 má start v bodě s_1 a cíl v bodě g_1, agent a_2 v s_1 a v g_2, tuto instanci prioritizované plánování nevyřeší

prioritizace. Pokud se naplánuje trajektorie jednoho z robotů, druhý již nenajde trajektorii, protože do jediného místa, kde se mohou vyhnout (výběžek nahoře), se již nestihne přemístit.

2.2 Pebble-motion problém

V pebble-motion problému [7] je cílem nalézt koordinovanou sekvenci pohybů pro agenty pohybující se na grafu. Tento graf je reprezentovaný vrcholy, které jsou propojené obousměrnými hranami. Na obsazených vrcholech jsou "pebbles"(do češtiny oblázky,v našem případě je reprezentují agenti). Tito agenti mají svůj start a cíl, přičemž každý agent má odlišný start, totéž platí o jejich cílech. Dále jsou tu neobsazené vrcholy, kterým se říká "blanks", tyto vrcholy umožňují pohyb agentů. Cílem je nalézt sekvenci pohybů, které přesunou agenta z jeho startu do cíle pohybem přes neobsazené vrcholy. Instance problému je vyřešena, pokud je nalezena sekvence bezkolizních pohybů, pomocí které se všichni agenti přesunou do svých cílových pozic. Dobře známou instancí tohoto problému je "15-puzzle problem". Je dokázáno, že tato instance problému má vždy řešení, nicméně nalezení řešení je v nejhorším případě NP-těžké [1]. V roce 2009 vymyslel Pavel Surynek algoritmus, který má polynomiální časovou náročnost řešení pebble-motion problému, pokud má graf dva vrcholy volné a jde o graf, který po odebrání jedné hrany bude pořád spojitý. V roce 2011 vydali Luna a Berkis algoritmus Push&Swap, o kterém tvrdili, že je kompletní. Jeho kompletnost vyvrátil článek [2], na kterém byl algoritmus zdokonalen na Push&Rotate, který vyřeší instanci pebble-motion problému v polynomiálním čase v závislosti na počtu agentů, pokud má graf alespoň 2 volné vrcholy a instance má na grafu řešení [2].

2.3 Push&Rotate

Pro vyřešení pebble-motion problému budu používat Push&Rotate algoritmus. Tento algoritmus je popsán podrobně ve článku [2], zde si popíšeme základní princip jeho fungování. Na vstupu algoritmus dostane n-tici (G, S, Φ), kde G značí libovolný neorientovaný graf, S označuje n-tici startů robotů $(s_1, \ldots, s_n)$ a Φ označuje n-tici cílů robotů $(\phi_1, \ldots, \phi_n)$. Na výstupu algoritmu je sekvence pohybů robotů. Tato sekvence je reprezentována jako n-tice $((a_1, h_1) \ldots (a_n, h_n))$, kde každý prvek (a_i, h_i) reprezentuje pohyb robota $a_i\, (i \in 1, \ldots, n)$ po hraně $h_i \in G$ v časovém kroku i. Algoritmus se skládá ze 3 základních kroků:

1. rozdělení na subgrafy
2. vyřešení problému pomocí operací push, swap, rotate

Obrázek 2.2: 15-puzzle problem

3. odmazání přebytečných pohybů

Rozdělení na subgrafy nám nalezne množinu vrcholů, do kterých se robot může přemístit pomocí operací push, swap a rotate. Už při této operaci může algoritmus dojít k tomu, že daná instance nemá řešení. Naopak se ale může stát, že je daná instance řešitelná pouze ve správném pořadí robotů. Vhodné pořadí je vypočteno na základě přiřazení robotů k subgrafům.
Druhý krok algoritmu vybírá roboty ve vypočteném pořadí. S jednotivými roboty posouvá z jejich startů do jejich cílů. Nejdříve zkusí operaci push, která se pokusí pohnout robotem na následujícím vrcholu, pokud se na něm nějaký robot nachází. Pokud je vyčištění vrcholu úspěšné posune se na něj robot, který se na tento vrchol chtěl přemístit. Pokud operace push selže, algoritmus vyzkouší operaci swap. Jejím cílem je vyměnit pozice robotů. Pokud i operace swap selže, vykoná se operace rotate. Tato operace se používá pro vyřešení problému na kružnici. Pokud danou instanci tímto způsobem vyřešíme (tzn. daná instance má řešení), tato část algoritmu nám vrátí sekvenci pohybů robotů.
V posledním kroku se snažíme sekvenci pohybů zminimalizovat, protože druhý krok nám vrátí i zbytečné pohyby tam a zpět. Pokud nebyl nějaký vrchol mezi odchodem robota z daného vrcholu a příchodem téhož robota na týž vrchol obsazen jiným robotem, můžeme sekvenci těchto dvou pohybů smazat.
V [2] bylo ukázáno, že algoritmus garantuje řešení, pokud jsou na grafu alespoň dva volné vrcholy a daná instance má na grafu G řešení. Všimněme si, že algoritmy pebble-motion neumožňují pohyb více robotů najednou, i když je to možné, pokud je dostatek volných vrcholů.

Kapitola 3

Notace

V této kapitole si uvedeme symboly, které budeme dále používat.

- $\|x, y\|$ značí euklidovskou vzálenost dvou bodů v 2-d prostoru
- $D\left(x, r\right)$ je množina všech bodů ležících v kružnici se středem x a poloměrem r
- $SV\left(\left(\langle p_1, p'_1\rangle, \ldots, \langle p_n, p'_n\rangle\right)\right)$ je funkce $\left(\mathbb{R}^2 \times \mathbb{R}^2\right)^n \rightarrow \left(\mathbb{R}^2\right)^n$, vracející počáteční vrcholy z n-tice hran definovaná jako :=

$$SV\left(\left(\langle p_1, p'_1\rangle, \ldots, \langle p_n, p'_n\rangle\right)\right) \rightarrow \left(p_1, \ldots, p_n\right)$$

- $FV\left(\left(\langle p_1, p'_1\rangle, \ldots, \langle p_n, p'_n\rangle\right)\right)$ je funkce $\left(\mathbb{R}^2 \times \mathbb{R}^2\right)^n \rightarrow \left(\mathbb{R}^2\right)^n$, vracející koncové vrcholy z n-tice hran definovaná jako :=

$$FV\left(\left(\langle p_1, p'_1\rangle, \ldots, \langle p_n, p'_n\rangle\right)\right) \rightarrow \left(p'_1, \ldots, p'_n\right)$$

- $FV\left(\left(\left(\langle p_1, p'_1\rangle, a_1\right), \ldots, \left(\langle p_n, p'_n\rangle\right), a_n\right)\right)$ je funkce $\left(\mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{N}\right)^n \rightarrow \left(\mathbb{R}^2\right)^n$, vracející koncové z vrcholy n-tice pohybů agentů po hranách definovaná jako :=

$$FV\left(\left(\left(\langle p_1, p'_1\rangle, a_1\right), \ldots, \left(\langle p_n, p'_n\rangle, a_n\right)\right)\right) \rightarrow \left(p'_1, \ldots, p'_n\right)$$

- $A\left(\left(\left(\langle p_1, p'_1\rangle, a_1\right), \ldots, \left(\langle p_n, p'_n\rangle\right), a_n\right)\right)$ je funkce $\left(\mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{N}\right)^n \rightarrow \left(\mathbb{N}\right)^n$, vracející agenty z n-tice pohybů agentů po hranách definovaná jako :=

$$FV\left(\left(\left(\langle p_1, p'_1\rangle, a_1\right), \ldots, \left(\langle p_n, p'_n\rangle, a_n\right)\right)\right) \rightarrow \left(a_1, \ldots, a_n\right)$$

- $0_{m \times n}$ je taková matice, která má počet řádků m a počet sloupců n a obsahuje samé nuly

Kapitola 4

Definice problému

Problém koordinace trajektorií lze formálně zadefinovat tímto způsobem. Mějme 2-d prostředí $E \subset \mathbb{R}^2$, kde množina E reprezentuje body, které leží ve volném prostoru mimo překážky. V tomto volném prostoru se pohybuje n agentů s radiusem r a maximální rychlostí v. Start agenta i je označen s_i, cíl agenta i je g_i. Pro zjednodušení zadefinujeme n-tici startů agentů $S = (s_1, \ldots, s_n)$ a n-tici cílů agentů $\Phi = (g_1, \ldots, g_n)$. Řešení problému Π je definováno jako n-tice trajektorií jednotlivých agentů $\Pi = \{\pi_i, \ldots, \pi_n\}$, kde $\pi_i(t)$ je funkce, která musí splňovat tyto podmínky:

1. $\pi_i(0) = s_i$ (trajektorie musí začít ve startovní pozici)
2. $\pi_i(t_{max}) = g_i$ (trajektorie musí skončit v c cílové pozici)
3. $\forall t \in \langle 0, t_{max} \rangle : D(\pi(t), r) \subset E$ (žádná část robota na žádné části trajektorie nesmí ležet mimo prostředí E)
4. $\forall i, j \in \{1, \ldots, n\}, \forall t \in \langle 0, t_{max} \rangle : \|\pi_i(t), \pi_j(t)\|_2 \geq 2r$ (v žádném okamžiku nejsou 2 agenti k sobě blíže než součet jejich poloměrů)
5. $\forall i \in \{1, \ldots, n\}, \forall t_1, t_2 \in \langle 0, t_{max} \rangle : \frac{d(\pi_i(t_1), \pi_i(t_2))}{|t_2 - t_1|} < v$, kde d je délka trajektorie mezi danými pozicemi (agent nepřekročí maximální rychlost)

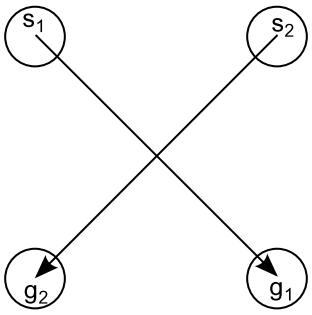
Kapitola 5

Navržený algoritmus

V této kapitole se seznámíme s algoritmem, který jsem pojmenoval Discretize-Push&Rotate-Paralize, zkráceně DPRP. Algoritmus řeší problém koordinace trajektorií pomocí převodu na pebble-motion problém. Algoritmus DPRP má na vstupu n-tici (E, r, S, Φ, v), kde E označuje prostředí pro pohyb agentů, r radius agentů, S n-tici startů agentů, Φ n-tici cílů agentů a v maximální rychlost agenta. Na výstupu je pak množina trajektorií jednotlivých agentů. Prvním krokem je diskretizace prostředí. V této části vytvoří algoritmus graf pro pohyb robotů G, n-tici hran $B = (b_i, \ldots, b_n)$, kde $SV(B) = S$ a $FV(B) = S'$, a n-tici hran $F = (f_i, \ldots, f_n)$, kde $X(F) = \Phi'$ a $Y(S) = \Phi$ ze vstupni n-tice (E, r, S, Φ). Ve druhém kroku algoritmus DPRP aplikuje algorimus Push&Rotate na vstupní graf G, počáteční vrcholy $FV(B)$ a koncové vrcholy $SV(F)$. Tento krok vrátí sekvenci pohybů M, přičemž bude pohybovat agenty z S' do Φ'. Třetím krokem bude paralelizace pohybů robotů. Na jejím vstupu bude sekvence pohybů M. Paralelizace umožní pohyb více robotů najednou a vrátí nám matici pozic agentů v závislosti na čase M' o velikosti $t \times n$, kde n je počet agentů a t je počet časových kroků potřebných k přemístění všech agentů z jejich cílů do jejich startů. Po paralelizaci dojde k přidání startů agentů S do prvního řádku matice M' a k přidání cílů agentů Φ do posledního řádku matice M'. Nakonec z matice M' za pomocí rychlosti v vytvoříme trajektorie pro jednotlivé roboty Π, tak že každou hranu projde agent a za stejný čas, který závisí na délce nejdelší hrany v grafu G a maximální rychlosti robota V. Pseudokód celého algoritmu můžeme vidět v algoritmu 1.

Algorithm 1 Celý algoritmus

1: **function** DPRP(E, r, $S = (s_1, \ldots, s_n)$, $\Phi = (\phi_1, \ldots, \phi_n)$,$v$)
2: $G, B = (b_1, \ldots, b_n), F = (f_1, \ldots, f_n) \leftarrow$DISCRETIZE$(E, r, S, \Phi)$
3: $M \leftarrow$PUSHROTATE$(G, FV(B), SV(F))$
4: $M' \leftarrow$PARALLELIZE$(FV(B), M)$
5: $M' \leftarrow \begin{bmatrix} S \\ M' \\ \Phi \end{bmatrix}$
6: $\Pi \leftarrow$CREATETRAJECTORIES(M', v, h_{max}) pozn. h_{max} je nejdelší hrana z G
7: return Π
8: **end function**



Obrázek 5.1: Na tomto obrázku vidíme 2 roboty kteří se pohybují ze startů s_i do cílů g_i paralelně, na tomto grafu nedojde ke grafové kolizi, nicméně kvůli špatně zvolené diskretizaci dojde ke geometrické kolizi

5.1 Diskretizace prostředí

Nyní si řekneme něco o požadavcích na bezkonfliknost v pebble-motion problému a v problému koordinace trajektorií. Umožní nám to se zamyslet nad správnou diskretizací prostoru.

5.1.1 Požadavky na bezkoliznost v pebble-motion problému

V pebble-motion problému je kolize definována na grafu. Dva roboti kolidují pokud jsou na stejném vrcholu grafu v jednom kroku nebo pokud se pohybují po jedné hraně proti sobě. Této situaci budeme říkat *grafový konflikt*.

5.1.2 Požadavky na bezkoliznost v problému koordinace trajektorií

Naopak kolize v problému koordinace trajektorií je definována tak, že v každém časovém okamžiku musí vzdálenost dvou libovolných robotů být větší než součet poloměrů jejich bezkontaktních zón. Pokud se stane že vzdálenost bude menší než součet poloměrů robotů, dojde ke *geometrickému konfliktu*.

5.1.3 Rozdíl diskretizace pro koordinace trajektorií a navržený přístup

Hlavním problémem je tedy navrhnout takovou diskretizaci, na které se nebude vyskytovat žádný geometrický konflikt, protože řešíme problém koordinace trajektorií za pomocí technik řešících pebble-motion problém a algoritmy řešící pebble-motion problém vylučují pouze grafový konflikt. Na obrázku 5.1 vidíme diskretizaci, na které je geometrická kolize, nicméně samotný algoritmus na řešení pebble-motion problému kolizi nerozpozná, protože se jedná pouze o geometrický konflikt. Nyní si zadefinujeme pojem π-bezkonfliknost, kterou budeme dále využívat.

Definice 1 (π-bezkonfliknost hran se společným vrcholem)
Dvě hrany $\boldsymbol{h_1} = \langle (x_1, y_1), (x_2, y_2) \rangle, \boldsymbol{h_2} = \langle (x_2, y_2), (x_3, y_4) \rangle$*, se společným vrcholem* $(x_2, y_2)\rangle$ *které leží na grafu G jsou bezkonfliktní pro roboty s radiusem r, pokud:=*

$$\forall \alpha \in \langle 0,1\rangle : \| \left(x_1, y_1\right) + \alpha \left[\left(x_2, y_2\right) - \left(x_1, y_1\right)\right], \left(x_2, y_2\right) + \alpha \left[\left(x_3, y_3\right) - \left(x_2, y_2\right)\right] \| \leq 2r$$

Máme 2 hrany $\boldsymbol{h_1}$,$\boldsymbol{h_2}$, každá hrana je dána počátkem (x_i, y_i) a koncem (x_{i+1}, y_{i+1}). Tyto hrany tedy můžeme považovat za orientované úsečky. Středy robotů pohybujících se po těchto usečkách tak, aby pohyb po úsečce trval jeden časový krok, musí být v každém časovém okamžiku t vzdálené alespoň $2r$.

5.1.4 Přípustná diskretizace pro problém koordinace trajektorií

Jak jsme demonstrovali na obrázku 2.1, ne všechny diskretizace jsou vhodné. Zavedeme tedy pojem přípustná diskretizace pro ty, které zachovají korektnost navrženého algoritmu vůči řešení problému koordinace trajektorií. Přípustná diskretizace pro problém koordinace trajektorií (E, r, n, S, Φ) musí splňovat tyto vlastnosti:

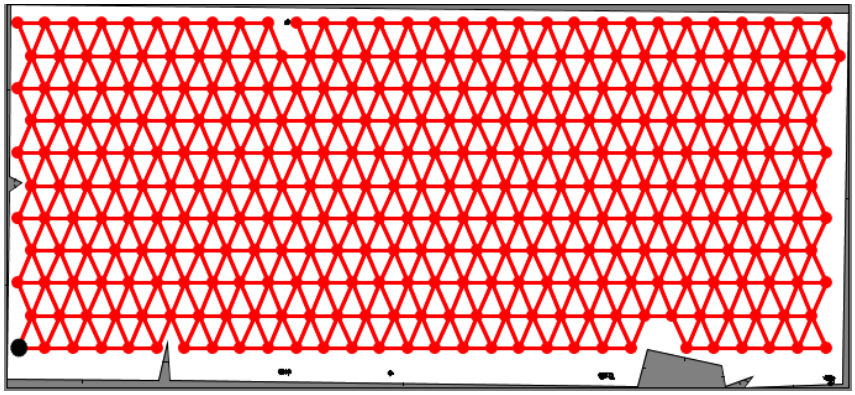
1. Každá hrana musí být dlouhá alespoň $2r$
2. Každý vrchol musí být vzdálen alespoň $2r$ od všech hran, na kterých vrchol neleží.
3. Žádné 2 hrany se nesmí protínat.
4. Každé 2 hrany se společným vrcholem musí být π-bezkonfliktní.
5. Každá startovní pozice s_i musí být přiřazena právě k jednomu vrcholu V, zároveň nesmí být více startovních pozic přiřazeno k jednomu vrcholu.
6. Každá cílová pozice g_i musí být přiřazena právě k jednomu vrcholu V, zároveň nesmí být více cílových pozic přiřazeno k jednomu vrcholu.
7. Všechny hrany a vrcholy musí mít vzdálenost od překážek alespoň $2r$

Nyní přistoupíme k jednomu druhu přípustné diskretizace, kterou jsem zvolil.

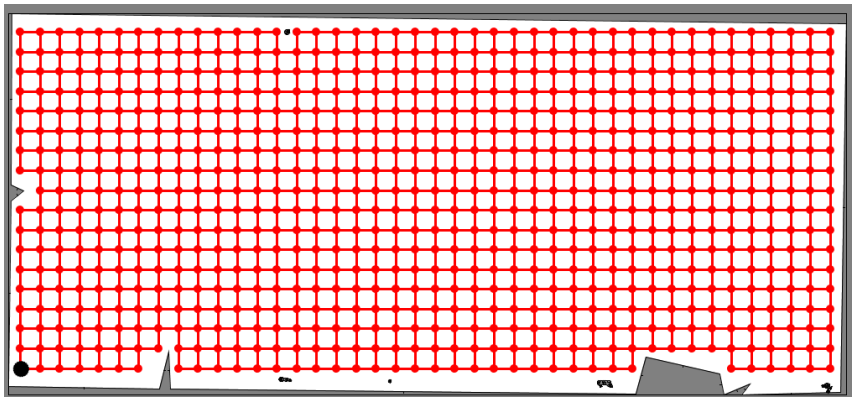
Teselace

Pro můj účel potřebuji takovou diskretizaci, která budou mít stejnou délku hran. Stejná délka hran nám totiž zajistí, že se mohou všichni roboti pohybovat maximální rychlostí, protože počítáme s tím, že robot přejde hranu za jednotkový časový úsek. Z tohoto vyplývá, že hrany musejí svírat i stejný úhel, aby mohla být délka hran maximální a zároveň, aby hrany splňovaly π-bezkonfliktnost.Tyto podmínky splňuje teselace. Teselace (česky pokrytí prostoru) je vyplnění prostoru obrazci tak, aby pokryly daný prostor bez mezer a zároveň se tyto obrazce nepřekrývali. Tato vlastnost nám zajisťuje, aby se hrany na grafu neprotínaly. Zároveň se musí jednat o tesalaci, kde hrany vycházející z jednoho vrcholu svírají stejný úhel. Vyjmenované vlastnosti splňují tyto tři teselace:

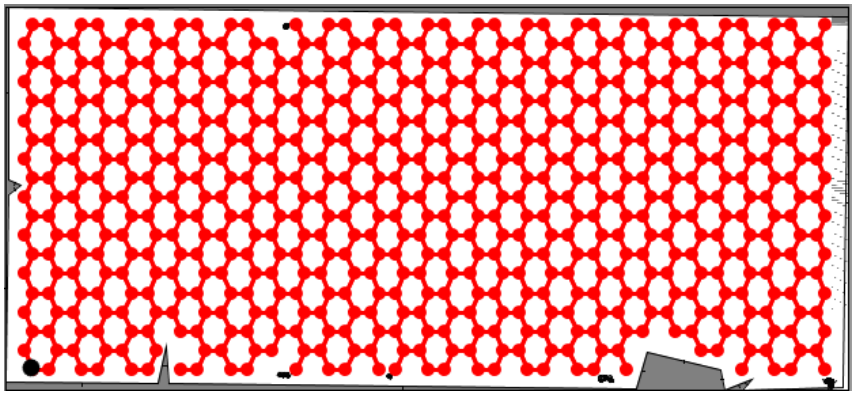
1. trojúhelníková (hrany svírají úhel 60°) viz obrázek 5.2
2. čtvercová (hrany svírají úhel 90°) viz obrázek 5.3
3. šestiúhelníková (hrany svírají úhel 120°) viz obrázek 5.4



Obrázek 5.2: Takto může vypadat jedna z variant přípustné diskretizace. Jedná se o trojúhelníkovou teselaci, která má 322 vrcholů 882 hran při radiusu robota 20 (vlevo dole)



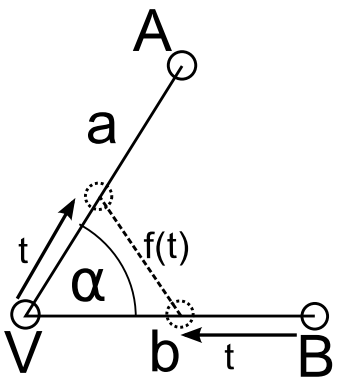
Obrázek 5.3: Takto může vypadat druhá varinta přípustné diskretizace. Jedná se o čtvercovou teselaci, která má 750 vrcholů 1435 hran při radiusu robota 20 (vlevo dole)



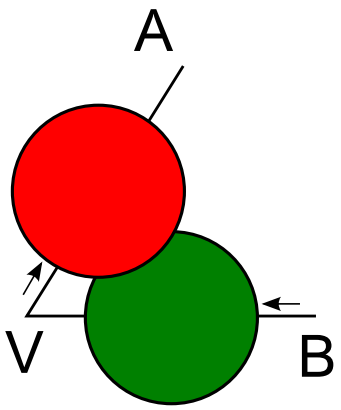
Obrázek 5.4: Takto může vypadat třetí varianta přípustné diskretizace. Jedná se o šestiúhelníkovou teselaci, která má 618 vrcholů 880 hran při radiusu robota 20 (vlevo dole)

5.1.5 Bezkonfliktnost hran se společným vrcholem

Abychom mohly použít teselaci, musíme zjistit, jaké parametry musí hrany se společným vrcholem splňovat, aby byly π-bezkonfliktní. V žádném čase t se nesmí stát, aby se roboti srazili, proto musí mít hrany určitou délku, která bude záviset na úhlu α. Úhel α je úhel, který svírají hrany a, b ve vrcholu V. Nyní budeme muset spočítat, v jaké části hran a, b je robot pohybující se po hraně a do vrcholu V a robot pohybující se po hraně b z vrcholu V (viz obrázek 5.5).



Obrázek 5.5: Jeden robot se pohybuje z vrcholu V do vrcholu A, druhý robot se současně pohybuje z vrcholu B do vrcholu V, v časovém okamžiku t mají od sebe vzdálenost $f(t)$



Obrázek 5.6: Radius robotů nesplňuje podmínku z rovnice 5.2, proto se roboti srazí

V jaké části hran se v daný okamžik roboti nachází, bude záviset na časovém okamžiku t. Jelikož robotovi s rychlostí v trvá přechod po jedné hraně jednotkový časový krok δt budeme uvažovat proměnnou t z intervalu $\langle 0, 1\rangle$. Vzájemnou vzdálenost robotů pohybujících se po těchto hranách nazveme $f(t)$. V t takovém, že jsou roboti nejblíže, musíme zajistit, že se roboti nesrazí. Tuto vzdálenost označíme jako $s = \min_{t\in\langle 0,1\rangle} f(t)$. Minimální separační vzdálenost mezi středy robotů si označíme jako $d = 2r$. Vzdálenost s musí být větší nebo stejně velká jako vzdálenost d. Nyní si spočítáme v jakém časovém okamžiku $t_{min} = \operatorname{argmin}_{t\in\langle 0,1\rangle} f(t)$ jsou robot nejblíže:

$$f^2(t) = (a - at)^2 + (bt)^2 - 2(a - at)bt\cos\alpha$$

$$f^2(t) = a^2 - 2a^2t + a^2t^2 + b^2t^2 - 2abn\cos\alpha + 2abn^2\cos\alpha$$

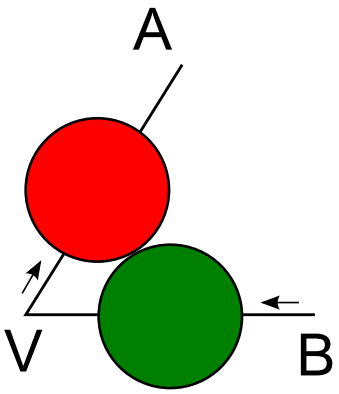
$$(f^2(t))' = -2a^2 + 2a^2n + 2b^2n - 2ab\cos\alpha + 4abn\cos\alpha$$

První derivaci položíme nule a spočítáme v jakém čase t jsou roboti nejblíže:

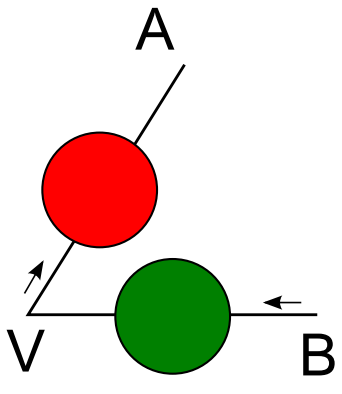
$$t_{min} = \frac{a^2 + ab\cos\alpha}{a^2 + b^2 + 2ab\cos\alpha} \tag{5.1}$$

Z tohoto vzorce si můžeme všimnout, že pokud jsou hrany a, b stejně dlouhé bude $t_{min} = \frac{1}{2}$:

$$t_{min} = \frac{a^2(1 + \cos\alpha)}{2a^2(1 + \cos\alpha)} = \frac{1}{2}$$



Obrázek 5.7: Radius robotů splňuje podmínku z rovnice 5.2, a to s rovností, proto se roboti v čase $\frac{t}{2}$ dotknou



Obrázek 5.8: Radius robotů splňuje podmínku z rovnice 5.2, a to s ostrou nerovností, proto se roboti minou

Nyní provedeme dosazení t_{min} do vzdálenosti s:

$$s^2 = \frac{a^2b^2(1-\cos^2\alpha)}{a^2+b^2+2ab\cos\alpha}$$

Z tohoto vzorce nám vyplývá důležitá závislost mezi hranami, úhlem a velikostí robotů:

$$\frac{a^2b^2(1-\cos^2\alpha)}{a^2+b^2+2ab\cos\alpha} \geq d^2 \tag{5.2}$$

Na obrázcích 5.6, 5.7, 5.8 vidíme, jestli se roboti o třech rozdíných poloměrech při stejné diskretizaci srazí nebo ne. Na obrázcích jsou hrany dlouhé 100 a úhel, který svírají je 60°. Na obrázku 5.6 dojde ke kolizi, protože jsou roboti příliš velcí, tato diskretizace je nepřípustná. Na obrázku 5.7 dojde k dotyku bezkontaktních zón, tato diskretizace je přípustná a zároveň nejlepší možná. Na obrázku 5.8 se roboti minou, tato diskretizace je přípustná, ale zároveň existuje lepší diskretizace, která má kratší hrany.

5.1.6 Návrh optimálních parametrů

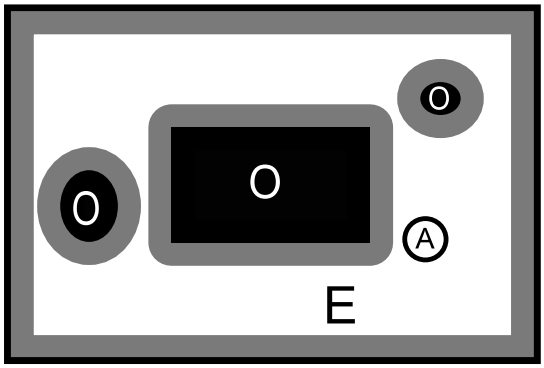
Nyní můžeme přejít k návrhu mřížky. Abychom vyplnili celý prostor stejně dlouhými hranami, můžeme si vybrat ze tří typů mřížek. Je to mřížka skládající se z rovnostranných trojúhelníků, čtverců nebo pravidelných šestiúhelníků [8]. Pro tyto možnosti spočítáme minimální velikost hrany a:

1. rovnostranný trojúhelník ($\alpha = 60°$):

$$\frac{a^4(1-0.25)}{2a^2+a^2} = d^2$$
$$a = 2d$$

2. čtverec ($\alpha = 90°$):

$$\frac{a^4(1-0)}{2a^2+0} = d^2$$
$$a = \sqrt{2}d$$



Obrázek 5.9: Na obrázku vidíme robota A, prostředí E a překážky O. Střed robota A se může pohybovat všude, kde vidíme bílou plochu, šedivou barvou je znázorněno nafouknutí překážek a zmenšení vnitřní hranice o radius robota.

3. pravidelný šestiúhelník ($\alpha = 120°$):

$$\frac{a^4(1-0.25)}{2a^2-a^2} = d^2$$

$$a = \frac{2}{\sqrt{3}}d$$

K tvorbě mřížky jsem si vybral čtverec, protože má nejlepší pokrytí prostoru (nejvíce hran a vrcholů), což je vidět na obrázcích 5.2, 5.3, 5.4. Teď již můžeme přejít k samotnému vytvoření grafu. Nejdříve musíme zajistit, aby výsledná trajektorie splňoovla podmínku 7 přípusktné diskretizace, tj. všechny hrany a vrcholy musí mít vzdálenost od překážek alespoň r. To lze zajistit zvětšením překážek o radius robota a posunutím vnější hranice prostředí dovnitř taktéž o radius robota viz obrázek 5.9.

5.1.7 Navržená diskretizace

Funkce DISCRETIZE má vstupní parametry prostředí E, radius robota r, starty robotů $S = (s_1, \ldots, s_n)$, cíle robotů $\Phi = (\phi_1, \ldots, \phi_n)$, bod p, z kterého budeme graf rozvíjet, a úhel teselace α. Jejím výsledkem bude neorientovaný graf G, vytvoření hran B pro přesunutí robotů z počátečních pozic S na počáteční vrcholy grafu S' a vytvoření hran F, pro přesunutí robotů z cílových vrcholů Φ' do cílových pozic Φ. Pseudokód navržené funkce je v algoritmu 2.

Algorithm 2 Diskretizace

1: **function** DISCRETIZE(E, r, S, Φ, p, α)
2: $G \leftarrow$ CREATEGRID(E, r, p, α)
3: $B \leftarrow$ ASSIGNAGENTS(G, S)
4: $F \leftarrow$ revert(ASSIGNAGENTS(G, Φ)) pozn. funkce revevert otočí hranu do směru pohybu robota
5: return G, B, F
6: **end function**

Vlastní řešení vytvoření mřížky

V tomto odstavci si popíšeme pseudokód funkce CREATEGRID, která vytvoří graf G, který splňuje pravidla přípustné diskretizace. Graf G bude obsahovat množinu hran a množinu vrcholů. Při vytváření mřížky dostaneme bod $p \in E$, ze kterého se bude mřížka rozvíjet. Dále musíme dostat úhel teselace α, který bude buď 60° pro trojúhelníkovou teselaci, 90° pro čtvercovou, nebo 120° pro šestiúhelníkovou. Nyní budeme bod p rozvíjet do všech směrů daných úhlem teselace α, čímž vytvoříme mřížku. Zadefinujeme si dvě množiny: a) openlist OL, b) closedlist CL. Dokud bude v OL alespoň jeden bod, tak libovolný bod vezmeme a smažeme z OL. Dále zjistíme, jestli daný vrcholů můžeme přidat do grafu (zda-li není na překážce). Pokud jsme ho mohli přidat a jeho předchůdce jsme také přidali do grafu, zjistíme, jestli můžeme přidat i hranu. Následně přidáme aktuální bod do CL. Posledním krokem cyklu je expanze aktuálního bodu. Směry expanze jsou závislé na úhlu teselace τ, kterou jsme zvolili na začátku.

Algorithm 3 Vytvoření mřížky

1: **function** CREATEGRID(E, r, p, α)
2: $G \leftarrow (\{\}, \{\})$
3: $OL \leftarrow \{p\}$
4: $CL \leftarrow \{\}$
5: **while** $|OL| > 0$ **do**
6: $v \leftarrow$ libovolný prvek z OL
7: $OL \leftarrow OL \setminus \{l\}$
8: Do grafu G přidáme vrchol v pokud splňuje podmínku $D(v, r) \subset E$. Do grafu přidáme hranu h spojující vrchol v a vrchol v', z kterého se vrchol v vytvořil pomocí expanze, pokud byly oba vrcholy přidány do grafu a hrana je vzdálena od překážek alespoň o r.
9: $CL \leftarrow CL \cup \{l\}$
10: Do OL přidáme vrcholy, které splňují podmínky teselace (případné hrany vedoucí do těchto vrcholů budou svírat úhel α a budou mít délku závislou na α a r).
11: **end while**
12: return G
13: **end function**

Přiřazení startů a cílů k jednotlivým vrcholům

V této části si popíšeme algoritmus 4, který popisuje funkci ASSIGNROBOTS, která vytvoří hrany tak, že přiřadí body z množiny P k vrcholům na grafu G a tím vytvoří množinu oboustranných hran H. Přiřazení popíši na startech, s cíli je to obdobné. K danému startu p najdeme nejbližší vrchol v a přiřadíme ho k němu za podmínky, že je neobsazený. Neobsazený znamená, že k tomuto vrcholu v není přiřazen žádný start robota, to znamená, že v množině H není žádná hrana, která má cíl ve vrcholu v. Pokud taková hrana existuje, hledáme další nejbližší vrcholy. V tu chvíli musí ale platit, že při pohybu na nově vytvořené hraně nedojde ke kolizi. Všechny hrany v množině H musí splňovat pravidla 2, 3, 7 přípustné diskretizace. Pokud tyto pravidla hrana $\langle p, v \rangle$ splňuje, přidáme ji do množiny H.

Algorithm 4 Přiřazení agentů k vrcholům na grafu

```
1: function assignAgents(G, P)
2:     H ← {}
3:     while |P| > |H| do
4:         p ← P
5:         v ← nejbližší přípustný bod grafu G k bodu p
6:         H ← H ∪ ⟨p, v⟩
7:     end while
8:     return H
9: end function
```

5.1.8 Analýza navrženého přístupu

V této kapitole si zanalyzujeme že navržený přístup (aplikace Push&Rotate na přípustnou diskretizaci prostředí a následnou paralelizaci) a ukážeme, že nalezené řešení splňuje bezkoliznost pro problém koordinace trajektorií.

Věta 1 *Pokud (G, B, F) je přípustná diskretizace pro problém $P = (E, r, S, \Phi, v)$, M je řešením pebble-motion problému $(G, Y(B), X(F))$, pak platí, že řešení M převedéné na trajektorie je řešením problému koordinac trajektorií.*

Důkaz 1 *Důkaz provedeme sporem. Předpokládejme, že existuje časový krok i, během kterého se dva roboti a_i, a_j pohybující se po hranách h_i, h_j srazí:*

1. *Roboti stojí na vrcholech grafu v_1, v_2.*
   Pokud oba roboti stojí na vrcholech postačí podmínky 1 a 2 přípustné diskretizace. Pokud roboti stojí na koncích jedné hrany použijeme podmínku 1 (minimální délka hrany musí být dlouhá alespoň $2r$). Pokud leží na jiných hranách použijeme podmínku 2, protože vrchol je součást hrany a hrana musí být od vrcholu, který na ní neleží vzdálený alespoň $2r$. Roboti se nesrazí.

2. *Jeden robot stojí v_1 a jeden se pohybuje po h_2.*

   Robot se pohybuje na vrchol, kde je stojící robot.
   Toto se nemůže stát kvůli podmínce pebble-motion, kdy se robot vždy pohybuje na prázdné místo, viz kapitola 2.2. Roboti se nesrazí.

   Robot se pohybuje z vrcholu, kde je stojící robot.
   Toto se nemůže stát kvůli tomu, že se robot nemohou pohnout na stejné místo, viz předchozí problém a zároveň nemohou mít stejné počáteční vrcholy viz podmínka 5 přípustné diskretizace. Roboti se nesrazí.

   Robot se pohybuje po jiné hraně než stojící robot.
   Jelikož musí být hrana od vrcholu, který na ní neleží vzdálena alespoň $2r$ podle podmínky 2 přípustná diskretizace, je i tento předpoklad splněn. Roboti se nesrazí.

3. *Oba roboti se pohybují.*

   Roboti se pohybují proti sobě po stejné hraně.
   Tato situace nemůže nastat, protože ji neumožňuje pebble-motion problém, kdy se roboti mohou pohybovat jen na prázdné vrcholy. Roboti se nesrazí.

Roboti se pohybují po hranách, které nemají společný vrchol.
Podle podmínky 3 přípustné diskretizace se nesmí hrany protínat a podle podmínek 2 a 3 musí být hrany od sebe vzdáleny alespoň r. Roboti se nesrazí.

První robot se pohybuje na startovní pozici druhého robota a ten se zároveň pohybuje na jinou pozici než je startovní pozice prvního robota, viz obrázek 5.5.
Jelikož musí být tyto hrany bezkonfliktní podle podmínky 4 přípustné diskretizace, roboti se nesrazí.

4. *Žádný robot se nesmí srazit s překážkou. Tento předpoklad je splněn podmínkou 7 přípustné diskretizaci. Robot nikdy nenarazí do překážky.*

Jde o spor, protože se roboti nikdy navzájem nemohou srazit a ani se nemohou srazit s překážkou, proto věta 1 platí.

5.2 Aplikace Push&Rotate na diskretizované prostředí

Na vstupu algoritmu Push&Rotate je graf G, který reprezentuje prostředí E, dále posloupnost startovních vrcholů $S' = Y(B)$ a posloupnost cílových vrcholů $\Phi' = X(F)$. Na výstupu je sekvence pohybů $M = ((a_1, h_1) \dots (a_n, h_n))$, kde každý prvek (a_i, h_i) reprezentuje pohyb agenta a_i $(i \in 1, \dots, n)$ po hraně $h_i \in G$ v časovém kroku i.

5.3 Paralelizace pohybu robotů

Paralelizace pohybů robotů nám umožní pohybovat s agenty současně, což se při řešení pebble-motion problému neděje. Paralizace funguje tak, že sekvenci pohybů M převede na matici pozic robotů M' o velikosti $t \times n$, kde t je počet časových kroků pro přesun všech robotů ze startovních pozic na cílové pozice a n je počet robotů. Nyní si popíšeme průběh funkce PARALLELIZE popsanou v algoritmu 5. Zadefinujme si dvourozměrné pole M'. Jeho první dimenze je čas, druhá dimenze jsou agenti. Je to tedy matice pozic, kde každý prvek odpovídá určitému agentovi a času. Samotná paraleliza probíhá v hlaví smyčce while, která běží dokud nejsou zpracovány všechny pohyby sekvence M. Algoritmus využívá množinu U, která reprezentuje všechny pohyby, které se staly před daným pohybem m_i v neparalelizované sekvenci pohybů. Další proměnnou je pole Δ, která udržuje informaci o tom, zda se daný robot pokusil pohnout. Hodnota 1 v tomto poli symbolizuje, že se o to robot pokusil, hodnota 0, že se zatím o pohyb agent nepokusil. V cyklu while z řádky 9 se pokoušíme o pohyb více robotů najednou, pokud je umožněn. To znamená, že pokud žádný pohyb jiného agenta nesměřoval do vrcholu, kterého chce dosáhnout agent z daného pohybu m_i, pohneme s ním do tohoto vrcholu. Zároveň tento pohyb odmažeme z posloupnosti pohybů. Koncový vrchol každého pohybu, přidáme do U, pokud tam již není. Pokud jsme se pokusili pohnout se všemi agenty nebo již není žádný nevyzkoušený pohyb z M (ukončovací podmínka cyklu), agenti j, kteří mají $P_{t,j} = 0$, zůstávají na místě, tudíž algoritmus opíše jejich pozici z času $t - 1$.

Algorithm 5 Paralelizace

function PARALLELIZE(M, S')
  $M' \leftarrow 0_{t \times n}$
  $M' \leftarrow \begin{bmatrix} S' \\ M' \end{bmatrix}$
  $t \leftarrow 1$
  while $|M| > 0$ **do**
    $i \leftarrow 0$
    $U \leftarrow \{\}$
    $\Delta \leftarrow 0_{1 \times n}$
    while $i < |M|$ **do**
      if $\Delta_{1,A(m_i)} = 0$ **then**
        $\Delta_{1,A(m_i)} = 1$
        if $FV(m_i) \notin U$ **then**
          $M'_{t,A(m_i)} \leftarrow FV(m_i)$
          $M \leftarrow M \setminus \{m_i\}$
          $i \leftarrow i - 1$
        end if
      end if
      if $FV(m_i) \notin U$ **then**
        $U \leftarrow U \cup FV(m_i)$
      end if
      $i \leftarrow i + 1$
      if $\Delta = 1_{1,n}$ **then**
        break
      end if
    end while
    for $j = 0; j < n; j \leftarrow j + 1$ **do**
      if $M'_{t,j} = 0$ **then**
        $M'_{t,j} \leftarrow M'_{t-1,j}$
      end if
    end for
    $t \leftarrow t + 1$
  end while
  return M'
end function

5.4 Vytvoření trajektorií robotů

V této sekci popíšeme poslední krok algoritmu DPRP. Jedná se o konečný převod sekvece pohybů robotů na trajektorie. Vstupem této části algoritmu je zparalelizovaná matice M' pohybů agentů, maximální rychlost agenta v a velikosti nejdelší hrany h_{max}. Naším úkolem je vytvořit z ní trajektorie pro každého robota. Jelikož ve sloupci nám roste čas t, hrany po kterých se budou roboti pohybovat jsou dány bodz v časech t_i a t_{i+1}. Pohyb po takto vytvořené hraně bude trvat jeden časový krok $\delta t = \frac{h_{max}}{v}$. Všichni roboti, kteří pojedou po kratších hranách, pojedou pomaleji. Tímto je pebble-motion problém zpět převeden na problém koordinace trajektorií a můžeme ho tedy porovnat s jinými algoritmy, které řeší problém koordinace trajektorií.

Algorithm 6 Vytvoření trajektorií

1: **function** CREATETRAJECTORIE(M', v, h_{max})
2: $\Pi \leftarrow$ prázdná n-tice trajektorií (π_1, pi_n)
3: **for** $i = 1; i \leq n; i \leftarrow i + 1$ **do** pozn. n je počet robotů
4: Vezmeme i-tý sloupec matice M', jeho prvky jsou body, které tvoří trajektorii robota. Přechod robota mezi jednotlivými body trvá čas $\delta t = \frac{h_max}{v}$, což nám vytvoří trajektorii i-tého robota π_i
5: **end for**
6: return Π'
7: **end function**

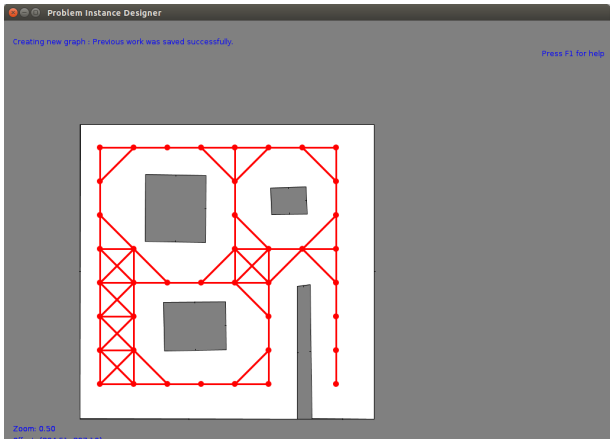
Kapitola 6

Implementace

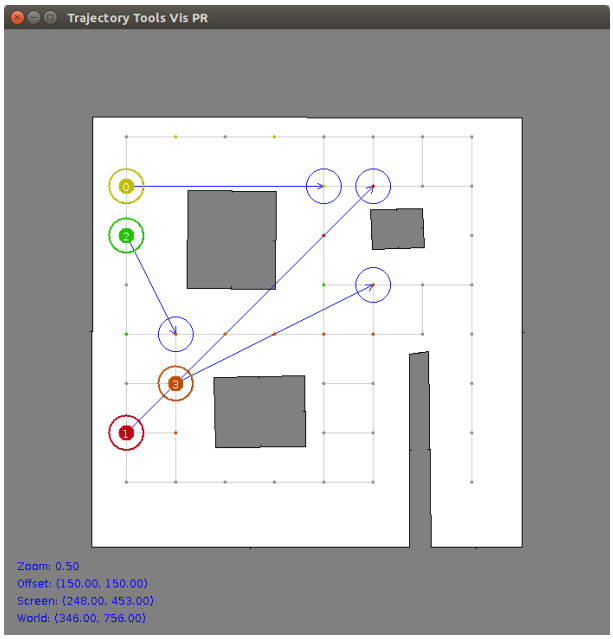
Algoritmus popsaný v předchozí kapitole byl naimplementován v jazyce Java. Naimplementováno bylo prostředí pro tvorbu přípustné diskretizace i samotný algoritmus. Zdrojové kódy naleznete na přiloženém CD.

6.1 Program pro diskretizaci prostředí

Pro účel diskretizace prostředí jsem použil již existující třídu ProblemDesigner z knihovny deconflictiontools [10]. Tuto třídu jsem rozšířil o funkcionalitu umožňující diskretizaci prostředí, která vytvoří graf pro danou instanci. Hlavní spouštěcí třídou je ProblemInstanceDesigner. V programu při stisknutí klávesy x přejdeme do režimu tvorby grafu. V tuto chvíli máme tři možnosti nastavení tvorby gridu. Když stiskneme klávesu c bude čtvercový, při stisku klávesy t trojúhelníkový, při stisku klávesy h bude šestiúhelníkový. Po tomto nastavení klikneme do prostředí levým tlačítkem myši a z tohoto bodu se vytvoří graf se zadanými parametry. Protože jsem testoval algoritmus s algoritmem prioritizovaného plánování, který nemusí dodržovat námi zadefinovanou přípustnou diskretizaci, můžeme vytvářet i obecné diskretizace. Pro tento účel jsem použil klávesu f, která vytvoří graf s osmiokolím, a klávesu r, která vytvoří graf s šestnáctiokolím. Vytvořený graf můžeme do výstupní instance uložit pomocí mezerníku a následným stisknutím klávesy s. Grafickou ukázku nástroje pro vytváření diskretizací vidíte na obrázku 6.1.



Obrázek 6.1: Na obrázku vidíme čtvercovou "nepřípustnou"diskretizaci s osmiokolím



Obrázek 6.2: Na obrázku vidíme počáteční krok ($t = 0$) vizualizace navrženého algoritmu

6.1.1 Algoritmus na vyřešení problému v diskretizovaném prostředí

Implementace algoritmu DPRP zabrala většinu času stráveného nad mojí bakalářskou prací. Samotnou implementaci algoritmu Push&Rotate jsem rozšířil o paralelizaci pohybů agentů a následný převod na trajektorie. Celý algoritmus je naimplementovaný ve třídě PR, která je spouštěna hlavní třídou PRSolver. Třída PRSolver dědí vlastnosti z třídy Solver, která se nachází v projektu deconflictiontools [10]. Na obrázku 6.2 vidíme ukázku vizualizace pomocí třídy Solver. Jak program spustit se dozvíte na přiloženém CD v souboru readme.txt. Na CD můžete také nalézt soubor s bakalářskou prací (bakalarka.pdf) a zdrojové kódy algoritmy ve složce vytvořeného projektu PushRotate.

Kapitola 7

Experimentální porovnání

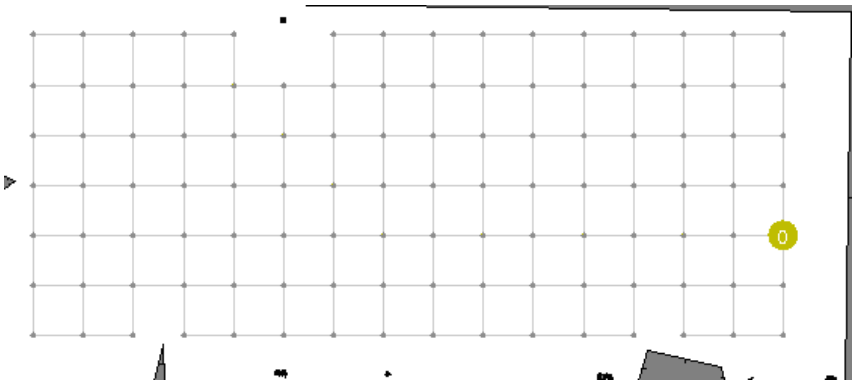
V této kapitole si popíšeme výsledky experimentálního porovnání. Algoritmus jsem otestoval na prázdném prostředí. V prázdném prostředí jsem vytvořil 80 náhodných instancí problému koordinace trajektorií tak, že jsem přidal starty a cíle robotů do náhodných bodů tohoto prostředí. Testoval jsem algoritmus pro tyto počty robotů: 1, 2, 4, 3, 5, 10, 15, 20. Pro každý počet robotů jsem vytvořil náhodně 10 různých instancí. Roboti měli radius $r = 50$. Na každé instanci jsem spustil tyto algoritmy:

1. DPRP: Discretize-Push&Rotate-Paralize pro přípustnou diskretizaci, viz obrázek 7.1
2. PP4: Prioritizované plánování pro přípustnou diskretizaci s timestepem 1, viz obrázek 7.1
3. PP8: Prioritizované plánování pro nepřípustnou diskretizaci s osmiokolím s timestepem 1, viz obrázek 7.2
4. PP16: Prioritizované plánování pro nepřípustnou diskretizaci s šestnáctiokolím s timestepem 1, viz obrázek 7.3
5. PP4t: Prioritizované plánování pro přípustnou diskretizaci s timestepem 72 viz obrázek, 7.1
6. PP8t: Prioritizované plánování pro nepřípustnou diskretizaci s osmiokolím s timestepem 72 viz obrázek, 7.2
7. PP16t: Prioritizované plánování pro nepřípustnou diskretizaci s šestnáctiokolím s timestepem 36, viz obrázek 7.3

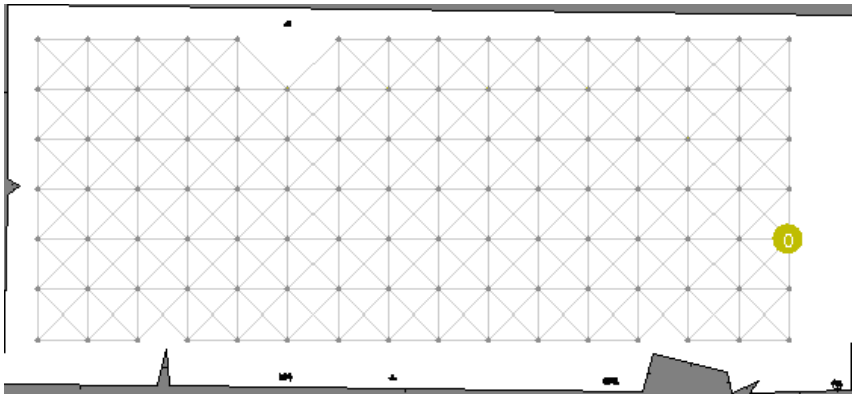
Timestep nám určuje krok diskretizace časové dimenze. Algoritmus DPRP byl úspěšný na všech testovaných instancích. Na grafech uvidíme i rozsahy ceny trajektorií (časové náročnosti), které jsou spočítány jako průměrné odchylky pro daný počet robotů.

7.1 Kvalita řešení problému

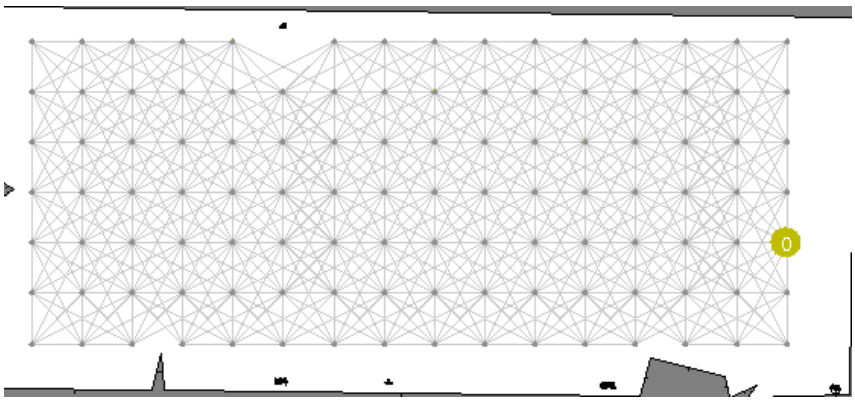
Kvalitu řešení definujme jako průměrnou cenu nalezeného řešení pro instance s různým počtem robotů. Cena trajektorií je určena tak, že se počítá čas agenta strávený mimo cílovou pozici. Na obrázcích 7.5, 7.6 vidíme, že cena trajektorií pro algoritmus DPRP roste rychleji v závislosti na počtu agentů oproti algoritmu na prioritizované plánování.



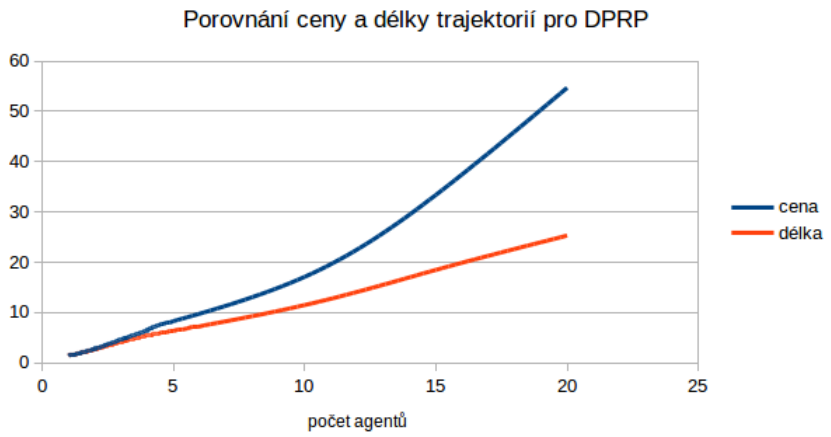
Obrázek 7.1: Na tomto obrázku vidíte přípustnou diskretizaci v porovnání s velikostí robota



Obrázek 7.2: Na tomto obrázku vidíte nepřípustnou diskretizaci s osmiokolím v porovnání s velikostí robota



Obrázek 7.3: Na tomto obrázku vidíte nepřípustnou diskretizaci s šestnáctiokolím v porovnání s velikostí robota

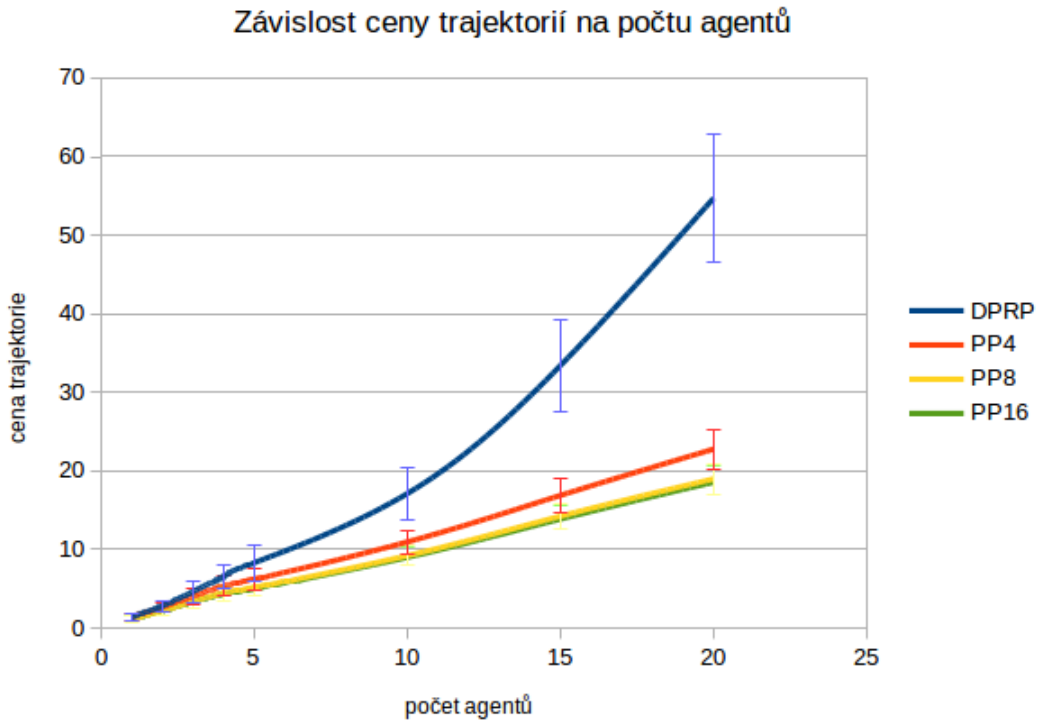


Obrázek 7.4: Na grafu vidíme rozdíl mezi délkou trajektorie a cenou pro algoritmus DPRP

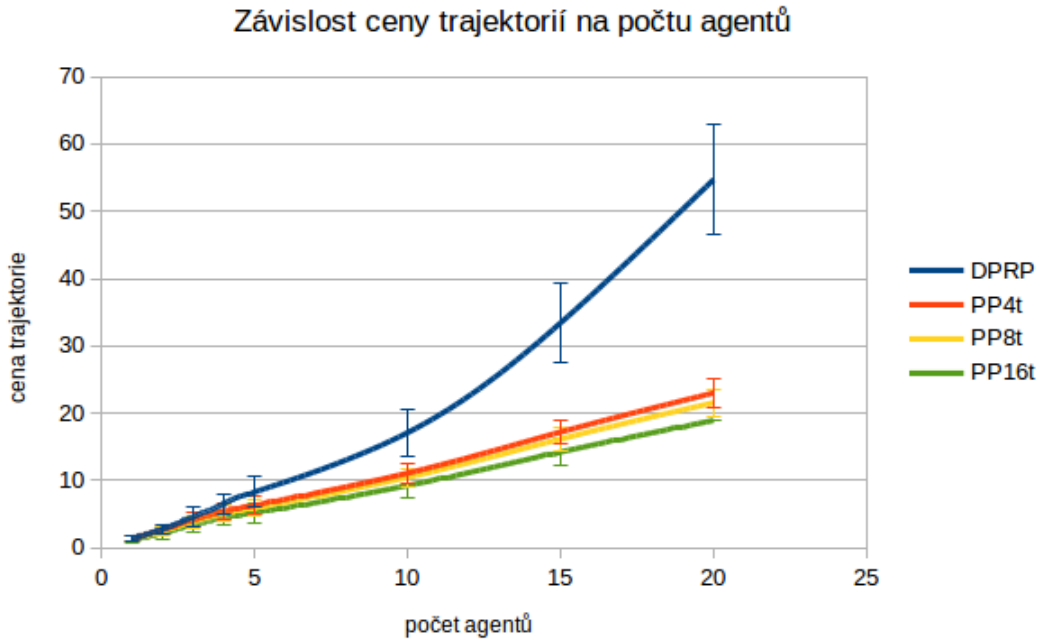
Jelikož při paralelizaci pohybů více robotů v problému pebble-motion musíme často čekat než daným vrcholem projde jiný agent (ten, který byl v sekvenci pohybů řešen dříve), cena trajektorie rychle roste. Na obrázku 7.4 si můžeme všimnout velkého rozdílu mezi cenou a délkou trajektorií algoritmu DPRP. Protože se roboti pohybují jednotkovou rychlostí jsou tyto dvě veličiny srovnatelné.

7.2 Rychlost řešení problému

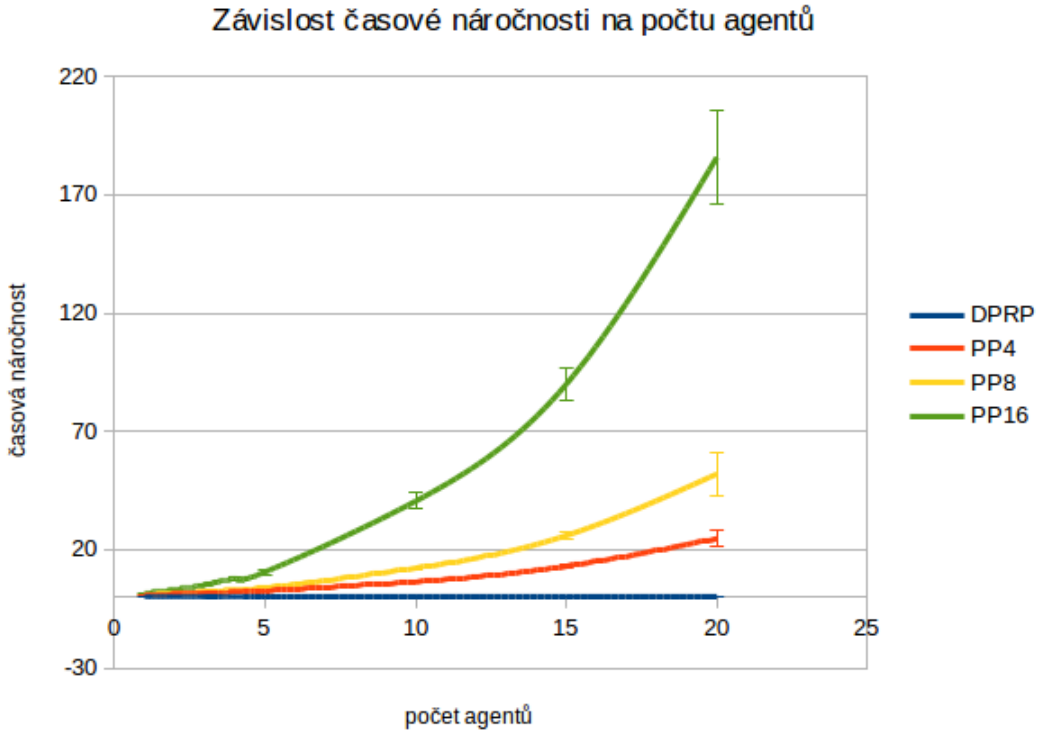
Na obrázcích 7.7, 7.8 si všimněme, že algoritmus DPRP pracuje téměř konstantní rychlostí. Jeho časová náročnost je tedy extrémně menší, což je dané polynomiální časovou náročností algoritmu Push&Rotate [2].



Obrázek 7.5: Na obrázku vidíme graf závislosti ceny trajektorií na počtu agentů s timestepem 1

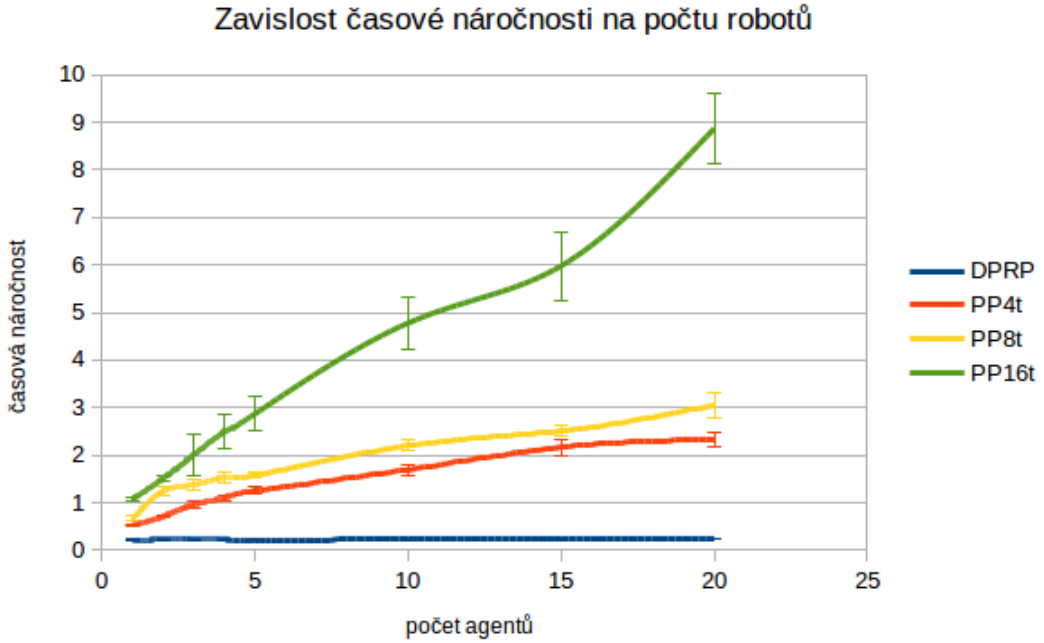


Obrázek 7.6: Na grafu závislosti ceny trajektorií na počtu agentů



Obrázek 7.7: Na obrázku vidíme graf závislosti časové náročnosti na počtu agentů s timestepem 1

.



Obrázek 7.8: Na obrázku vidíme graf závislosti časové náročnosti na počtu agentů s timestepem 29 kromě PP16, kde je timestep 15

Kapitola 8

Závěr

Efektivní a korektní řešení problému koordinace trajektorií robotů může zlepšit vývoj továren nové generace, kde budou roboti hlavní pomocnou silou. Algoritmy, které řeší problém koordinace trajektorií mají exponenciální časovou náročnost nebo nejsou korektní.
V této bakalářské práci jsem navrhl algoritmus Discretize-Push&Rotate-Paralize DPRP, který funguje na základě převodu problému koordinace trajektorií na problém pebble-motion. Problém pebble-motion je následně vyřešen algoritmem Push&Rotate. Nakonec algoritmus pohyby zparalelizuje a vrátí trajektorie jednotlivých robotů (zpětné převedení do problému koordinace trajektorií). Podrobněji se o navrženém algoritmu můžeme dočíst v kapitole 5, kde je i důkaz jeho korektnosti. Pro důkaz korektnosti jsem zavedl pojem přípustná diskretizace, která nám zajistí geometrickou bezkoliznost (tj. těla robotů se nesmí během pohybu překrývat) pohybu robotů v 2-d prostředí, protože pebble-motion problém řeší jen grafovou bezkoliznost (tj. aby se dva roboti nikdy nenacházeli na stejném vrcholu daného grafu, případně pokud je výsledné řešení paralelizováno, aby necestovali v opačném směru po stejné hraně), tudíž by mohl být algoritmus při "nepřípustné"diskretizaci nekorektní. V kapitole 6 jsem popsal implementaci navrženého algoritmu. V kapitole 7 jsem experimenálně porovnal kvalitu a časovou náročnost ve srovnání algoritmem prioritizovaného porovnání, který řeší problem koordinace trajektorií. Nyní přejděme k výsledkům experimentů.
V experimentu se ukázalo že algoritmus DPRP má až třikrát větší cenu trajektorií (doba, po kterou není agent ve své cílové pozici) než prioritizované plánování. Zároveň se ale ukázalo, že tento špatný výsledek je způsoben dlouhým čekáním robotů na místě, protože délka jednotlivých trajektorií je až více než dvakrát menší než jejich cena. V experimentu se také ukázalo, že algoritmus je nesrovnatelně rychlejší než prioritizované plánování, v čemž spočívá jeho hlavní výhoda.

Literatura

[1] Pat Mähler: *Polynomial Algorithms for Multi-Agent Path Planning Problems. Master thesis, University of Basel, 2012.*

[2] Boris de Wilde, Adriaan W. ter Mors, Cees Witteveen: *Push and Rotate: a Complete Multi-agent Pathfinding Algorithm. Journal of Artificial Intelligence Research, 2014, 443-492.*

[3] Michal Čáp, Peter Novák, Alexander Kleiner, Martin Selecký: *Prioritized Planning Algorithms for Trajectory Coordination of Multiple Mobile Robots. IEEE transactions on automation science and engineering (in review). 2014*

[4] Glenn Wagner, Howie Choset: *Subdimensional expansion for multirobot path planning", Artificial Intelligence, Volume 219. February 201. Pages 1-24.*

[5] Trevor Scott Standley: *Finding optimal solutions to cooperative pathfinding problems. In Maria Fox and David Poole, editors, AAAI. AAAIPress, 2010*

[6] Michael Erdmann and Tomas Lozano-Pérez: *On multiple moving objects. Algorithmica, 2:1419–1424, 1987*

[7] Adi Botea, Pavel Surynek: *Multi-Agent Path Finding on Biconnected Directed Graphs. Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI 2015), Austin, TX, USA, pp. 2024-2030, AAAI Press, 2015.*

[8] W. W. Rouse Ball, H. S. M. Coxeter: *Mathematical Recreations and Essays (Dover Recreational Math), 2010*

[9] Steven M. La Valle *Chapter 7.2 – Multiple Robots of Steven M. La Valle: Planning Algorithms. Cambridge University Press, 2006*

[10] *Projekt deconflictiontools*
`jones.felk.cvut.cz/redmine/projects/deconflictiontools/`