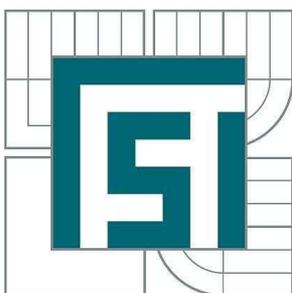


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

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 HEJNOVÝCH ALGORITMŮ

ROBOT PATH PLANNING BY MEANS OF PARTICLE SWARM ALGORITHMS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE

AUTHOR

PETR HRČKA

VEDOUCÍ PRÁCE

SUPERVISOR

RNDr. JIŘÍ DVOŘÁK, CSc.

BRNO 2010

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky

Akademický rok: 2009/2010

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

student(ka): Petr Hřčka

který/která studuje v **bakalářském studijním programu**

obor: **Aplikovaná informatika a řízení (3902R001)**

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 bakalářské práce:

Plánování cesty robotu pomocí hejnových algoritmů

v anglickém jazyce:

Robot path planning by means of particle swarm 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 kritériální funkce. Téma souvisí s řešením výzkumného záměru MSM 0021630529: Inteligentní systémy v automatizaci.

Cíle bakalářské práce:

1. Analyzovat přístupy k plánování cesty robotu.
2. Popsat problematiku hejnových algoritmů.
3. Navrhnout a implementovat systém pro plánování cesty robotu pomocí hejnových algoritmů.
4. Provést ověřovací a srovnávací experimenty.

Seznam odborné literatury:

MEYER, J.-A., FILLIAT, D. Map-Based Navigation in Mobile Robots: II. A Review of Map-Learning and Path-Planning Strategies. *Cognitive Systems Research*, vol. 4, issue 4, 2003, pp. 283-317.

QIN, Y.-Q., SUN, D.-B., LI, N., CEN, Y.-G. Path Planning for Mobile Robot Using the Particle Swarm Optimization with Mutation Operator. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*, 26-29 Aug. 2004, vol. 4, pp. 2473–2478.

LEI, K., QIU, Y., HE, Y. A Novel Path Planning for Mobile Robots Using Modified Particle Swarm Optimizer. In *Proceedings of the 1st International Symposium on Systems and Control in Aerospace and Astronautics ISSCAA 2006*, 19-21 Jan. 2006, 4 pp.

Vedoucí bakalářské práce: RNDr. Jiří Dvořák, CSc.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2009/2010.

V Brně, dne

L.S.

Ing. Jan Roupec, Ph.D.
Ředitel ústavu

prof. RNDr. Miroslav Doupovec, CSc.
Děkan fakulty

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení:

Bytem:

Narozen/a (datum a místo):

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta strojního inženýrství

se sídlem Technická 2896/2, 616 69 Brno

jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....

(dále jen „nabyvatel“)

Čl. 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):

- disertační práce
 - diplomová práce
 - bakalářská práce
 - jiná práce, jejíž druh je specifikován jako
- (dále jen VŠKP nebo dílo)

Název VŠKP: _____

Vedoucí/ školitel VŠKP: _____

Ústav: _____

Datum obhajoby VŠKP: _____

VŠKP odevzdal autor nabyvateli v*:

- tištěné formě – počet exemplářů
- elektronické formě – počet exemplářů

* hodící se zaškrtněte

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracování díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2

Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
 - ihned po uzavření této smlouvy
 - 1 rok po uzavření této smlouvy
 - 3 roky po uzavření této smlouvy
 - 5 let po uzavření této smlouvy
 - 10 let po uzavření této smlouvy(z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3

Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísní a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabyvá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne:

.....
Nabyvatel

.....
Autor

ABSTRAKT

Tato práce se zabývá plánováním cesty robotu pomocí hejnového algoritmu. V první části je popsán samotný PSO algoritmus a přístupy k pracovnímu prostoru robotu pro nasazení PSO. V druhé části jsou srovnávány různé přístupy k plánování cesty robotu na vytvořené simulaci v jazyku C#.

ABSTRACT

This paper describes robot path planning by means of particle swarm algorithms. The first part describes the PSO algorithm and the approaches to the workspace of the robot for deployment of the PSO. The second part compares various approaches to robot path planning on the created simulation in C# .

KLÍČOVÁ SLOVA

Robot, plánování cesty, optimalizace rojem částic, MAKLINK, Dijkstrův algoritmus, mutační operátor.

KEYWORDS

Robot, path planning, particle swarm optimization, MAKLINK graph, Dijkstra's algorithm, mutation operator.

Obsah

Zadání závěrečné práce.....	3
Licenční smlouva	5
Abstrakt	7
Seznam použitých symbolů	11
Úvod	13
1 Rozbor	15
1.1 Rojová inteligence	15
1.1.1 Historie a vývoj algoritmu	15
1.1.2 Popis základních prvků PSO.....	16
1.1.3 Popis algoritmu PSO.....	16
1.1.4 Mutace	17
1.2 Realizace pracovního prostoru robotu pro simulaci	18
1.3 Graf viditelnosti a Dijkstrův algoritmus	20
2 Plánování cesty robotu pomocí PSO.....	21
2.1 PSO a horizontální členění pracovního prostoru	21
2.1.1 Původní přístup	21
2.1.2 Upravený přístup.....	22
2.2 PSO + MAKLINK	23
2.3 Ukončení běhu PSO	23
3 Experimenty a simulace	25
3.1 Srovnávací experimenty	25
3.1.1 Konvexní překážky	25
3.1.2 Mutační operátor a jeho parametry	25
3.1.3 Nekonvexní překážky	26
3.2 Popis simulačního programu	27
3.2.1 Tvorba bludiště	28
3.2.2 Tvorba MAKLINK grafu.....	28
3.2.3 Spuštění algoritmů	28
3.2.4 Podrobnější nastavení PSO	28
3.2.5 Diagnostické funkce	29
Závěr	31
Seznam použité literatury.....	33

SEZNAM PUŽITÝCH SYMBOLŮ

PSO	Optimalizace pomocí roje částic (particle swarm optimization, PSO).
t	Iterace.
T	Počet iterací PSO algoritmu.
D	Dimenze.
c1,c2	Učící parametr.
r1,r2	Náhodný parametr.
w	Váhový parametr.
wmin	Minimální váhový parametr.
wmax	Maximální váhový parametr.
P	Počet částic roje.
i	Označení i-té částice roje.
\vec{x}_i	Vektor polohy částice.
\vec{v}_i	Vektor rychlosti částice.
\vec{p}_i	Vektor lokální paměti částice.
\vec{p}_g	Vektor globální paměti roje (globální částice).
L	Pomyslná svislice v pracovním prostoru robotu, procházející vrcholem překážky.
XMIN,XMAX	Meze velikosti složky vektoru polohy částice.
VMIN,VMAX	Meze velikosti složky vektoru rychlosti částice
LogjamStep	Doba po kterou se \vec{p}_g nezměnilo nebo jen velmi málo.
SwarmDist	Vzdálenost mezi všemi částicemi a aktuální optimální hodnotou.
BorderDist	Hraniční hodnota změny \vec{p}_g .
S, T	Počáteční a koncový bod cesty robotu (Start ,Target).
F	Finální optimalizovaná funkce.
f	Část optimalizované funkce.
H	Penalizační funkce.
λ	Velikost penalizace.
T1	Parametr mutace.
P_d	Střed spojnice v MAKLINK grafu, jehož polohu optimalizujeme.
h_d	Parametr určující polohu bodu P_d .
K	Množina obsahující všechny nevertikální hranice překážek (k_{11}, k_{12}) ... (k_{D-1n}, k_{Dn}) mezi svislicemi L.
η	Náhodná proměnná z intervalu [0,1].

ÚVOD

Přesto, že existuje plno osvědčených metod, které se v praxi využívají pro řízení robotu, neznamená to, že by se nemohly najít nové nebo upravit stávající metody.

Tato práce se zabývá nasazením metody optimalizace pomocí roje částic (Particle Swarm Optimization; dále jen PSO), na řízení robotu. Tato metoda spadá mezi tzv. rojovou inteligenci (Swarm Intelligence). Tato skupina algoritmů se inspiruje chováním skupin živočichů (patří sem např. mravenčí algoritmy). Samotná rojová inteligence spadá mezi genetické algoritmy, neuronové sítě a jiné metody umělé inteligence. Za posledních 6 let vzniklo mnoho způsobů aplikování PSO na řízení robotu. V některých pracích se srovnává PSO s genetickými algoritmy. V této práci však srovnáváme různé přístupy PSO k řízení robotu mezi sebou, inspirované pracemi [1],[2] a s jedním ze základních algoritmů pro nalezení nejkratší cesty v ohodnoceném grafu Dijkstrovým algoritmem. Tyto dvě metody jsme vybrali proto, že jejich výsledky jsou podobné s grafem viditelnosti a tak je můžeme lépe porovnat mezi sebou.

Cílem praktické části je vytvořit simulaci pro řízení robotu nejlepším možným způsobem za pomoci PSO.

1 Rozbor

Pro plánování cesty robotu se v dnešní době používá mnoho algoritmů a přístupových technik. Mezi reprezentaci prostředí z hlediska topologie můžeme zahrnout grafy viditelnosti, grafy tečen nebo Voronoiovy diagramy. Prostor lze reprezentovat také metricky, to znamená, že prostředí je rozděleno do množiny buněk (polí), nesoucí informaci zda jsou či nejsou průchozí.

Metody pro plánování cest mohou být lokální (robot se nachází v neznámém prostředí) nebo globální, kdy máme dostatečné množství informací o prostředí. Mezi globální metody plánování cesty robotu řadíme:

- Plánování cesty robotu pomocí map cest
Na topologicky reprezentovaném prostředí, se nejčastěji pomocí pokročilých algoritmu (Dijkstrův algoritmus, algoritmus A*) hledají cesty grafem.
- Plánování cesty metodou pravděpodobnostní cestovní mapy
Tento algoritmus využívá pravděpodobnostní cestovní mapy (PRM, probabilistic roadmaps).
- Plánování cesty rychle mapujícími náhodnými stromy
Speciální případ metody pravděpodobnostní cestovní mapy. Rychle mapující náhodné stromy (RRT, rapidly-exploring random trees).
- Plánování cesty metodami rozkladu prostředí do buněk
Použijeme zde opět grafové algoritmy. Vrcholy grafu jsou volné buňky a hranice buněk představují hrany grafu.
- Plánování cesty metodou pole potenciálu
Tato metoda využívá teorii elektrického pole.
- Plánování cesty genetickými algoritmy
Použití algoritmů inspirovaných evolucí.
- Metody založené na rojové inteligence
Využití mravenčích algoritmů (ACO, ant colony optimization) nebo optimalizace pomocí roje částic (PSO).

Pro srovnání použitelnosti nasazení PSO na problém plánování cesty robotu jsem se rozhodl pro Dijkstrův algoritmus a graf viditelnosti.

1.1 Rojová inteligence

Podobně jako je celulární automat inspirován principem DNA nebo genetické algoritmy evolucí, tak je rojová inteligence založena na chování živočichů jednoho druhu ve skupině.

1.1.1 Historie a vývoj algoritmu

PSO je založen na stochastické optimalizační metodě vyvinuté psychologem J. Kennedym a elektroinženýrem R. Eberhartem v roce 1995. Tato metoda je inspirována modelem chování hejna ptáků, vytvořeným zoologem Frankem Heppnerem v roce 1990.

Heppner sledoval zpomalené záběry hejna ptáků a zjistil, že se od hejna oddělí jeden či dva jedinci a ostatní je pak následují. Vytvořil model, kde jsou ptáci přitahováni k hnízdišti. Tím se odlišoval od ostatních modelů popisujících pohyb hejna. Pohyb

definoval dvěma pravidly. Každý jedinec se snaží dostat co nejbližší k ostatním a zároveň se snaží do žádného z nich nenarazit. Pokud se pták dostal k hnízdišti, touha přistát byla větší, než zůstat v hejnu. Pokud jedinec najde hnízdiště a odtrhne se, pak se k tomuto jedinci postupně přidávají i ostatní, dokud zde nepřistane celé hejno.

1.1.2 Popis základních prvků PSO

PSO algoritmus simuluje sociální chování hejna ptáků, reprezentované pohybem částic v n -rozměrném prostoru. Každá částice v roji představuje možné řešení optimalizačního problému a je reprezentována vektorem polohy (1) a rychlosti (2) v rozměrném prostoru. Počáteční populace v roji je náhodně vygenerována a v průběhu výpočtu se mění, podobně jako v genetickém algoritmu. Po každém cyklu se aktualizuje poloha a rychlost každé částice, na základě jejich individuální a sociální zkušenosti s jistou danou mírou váhy k této transformaci. Zkušeností rozumíme nejlepší polohu částice v prostoru (3), tzv. lokální paměť, a u sociální zkušenosti je to nejlepší poloha ze všech částic (4), nazývaná taky globální paměť (globální částice).

$$\vec{x}_i = \{x_{i1}, x_{i2}, \dots, x_{iD}\} \quad (1)$$

$$\vec{v}_i = \{v_{i1}, v_{i2}, \dots, v_{iD}\} \quad (2)$$

$$\vec{p}_i = \{p_{i1}, p_{i2}, \dots, p_{iD}\} \quad (3)$$

$$\vec{p}_g = \{g_1, g_2, \dots, g_D\} \quad (4)$$

D představuje počet neznámých parametrů v optimalizované funkci. Podle vzorců (5), (6) se aktualizuje poloha roje v každé iteraci t . Iterací rozumíme jeden cyklus z konečného počtu T cyklů potřebných k nalezení optimálního výsledku.

$$\vec{v}_i(t+1) = \underbrace{w \times \vec{v}_i(t)}_{\text{Váhová}} + \underbrace{c_1 \times r_1 \times (\vec{p}_i(t) - x_i(t))}_{\text{individuální}} + \underbrace{c_2 r_2 \times (\vec{p}_g(t) - \vec{x}_i(t))}_{\text{skupinová složka}} \quad (5)$$

Váhová, individuální a skupinová složka.

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \quad (6)$$

kde $i=1,2,\dots,P$ (P je počet částic v roji); $t=1,2,\dots,T$ (T je maximální počet cyklů); c_1, c_2 parametry učení v rozsahu 0..2 (nejčastěji bývají oba parametry rovny 2); r_1, r_2 jsou náhodná čísla z intervalu $[0,1]$ a $w(t)$ je váha omezující vliv předchozí rychlosti částice na novou daná vzorcem:

$$w(t) = w_{\min} - \frac{w_{\max} - w_{\min}}{T} * t \quad (7)$$

kde w_{\min} a w_{\max} je počáteční a koncová váha.

1.1.3 Popis algoritmu PSO

1. Určení počátečních parametrů. Počet částic roje P , počet iterací T , vektor omezující prostor řešení $[XMIN, XMAX]$ a vektor omezující velikost rychlosti částice $[VMIN, VMAX]$. Konstanty c_1, c_2 , váhový parametr w a počet dimenzí D .

2. Inicializace roje. Náhodné vygenerování počáteční pozice částic a jejich rychlosti pro celou populaci čítající P částic, v rozsahu vektorů $[XMIN, XMAX]$, $[VMIN, VMAX]$ o

velikosti D . Lokální paměť každé částice je rovna vektoru pozice. Nakonec stanovíme globální paměť \vec{p}_g jako polohu částice s nejlepší hodnotou kritériální funkce ze všech \vec{p}_i .

3. Změna poloh částic dle rovnice (5) a jejich přizpůsobení na interval $[XMIN, XMAX]$.

4. Změna rychlostí částic dle rovnice (6) a jejich přizpůsobení na interval $[VMIN, VMAX]$.

5. Výpočet aktuální hodnoty kritériální funkce pro každou částici. Jestli je menší než hodnota \vec{p}_i , nastav $\vec{p}_i = \vec{x}_i$. A nakonec aktualizace \vec{p}_g .

6. Přejít na krok 3 dokud se neprovede daný počet iterací nebo kritériální funkce nedosáhne přijatelné hodnoty.

1.1.4 Mutace

Nejčastějším nedostatkem PSO je, že vývoj algoritmu konverguje do lokálního minima. To nastane tehdy, když jsou částice nasměrovány na lokální minimum a hledané globální minimum je mimo dosah roje. Tento problém se dá řešit mutací, podobně jako v GA. Nastíníme zde dvě řešení tohoto problému.

1. Tento přístup na konci každého cyklu algoritmu změní globální paměť roje. Obsahuje pouze jeden volitelný parametr $T1$ z intervalu $[0.5T, T]$. V literatuře je doporučena hodnota tři čtvrtiny T .

Pseudokód mutace:

```
IF t ≤ T1 THEN  $\vec{p}_{gk} = \vec{p}_{gk} * (1 + 0,5\eta)$ 
ELSE  $\vec{p}_{gk} = \vec{p}_{gk} * (1 - 0,5\eta)$ 
```

kde \vec{p}_{gk} je k-tá složka vektoru \vec{p}_g , η je náhodná proměnná z intervalu $[0, 1]$.

2. Další možností je sledování změny hodnoty \vec{p}_g . Pokud je hodnota \vec{p}_g několikrát za sebou stejná a rozptyl částic vzhledem k \vec{p}_g je menší nebo roven krajní hodnotě, provede se znovu vygenerování populace (náhodné nastavení vektoru polohy a rychlosti všech částic).

```
While k < MaxIterationTime & fitnessValue > přijatelná mez
```

```
IF (fitnessValue (k) == fitnessValue (k-1))
```

```
LogjamStep = LogjamStep + 1
```

```
END
```

```
IF LogjamStep ≥ MaxStep
```

```
IF SwarmDist < BorderDist
```

```
    Znovu vygenerování pozice a rychlosti populace
```

```
    LogjamStep = 0
```

```
ELSE
```

```
    LogjamStep = LogjamStep + 1
```

```
END
```

```
END
```

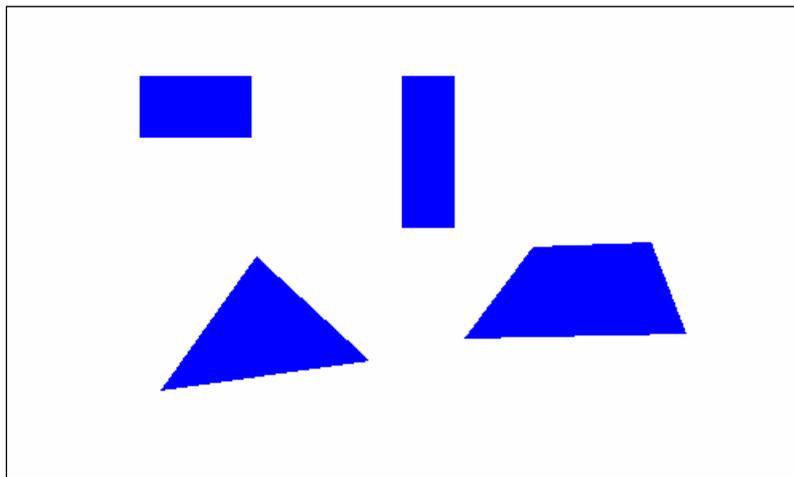
```
    Aktualizace částic podle vzorců (5),(6)
```

```
END
```

kde k je iterace, $LogjamStep$ je doba, po kterou se \bar{p}_g nezměnilo nebo jen velmi málo. $MaxStep$ je hraniční hodnota kdy se částice nezměnila, $SwarmDist$ je vzdálenost mezi všemi částicemi a aktuální optimální hodnotou, což může být součet vzdáleností nebo průměrná vzdálenost od \bar{p}_g . $BorderDist$ je hraniční hodnota změny \bar{p}_g .

1.2 Realizace pracovního prostoru robotu pro simulaci

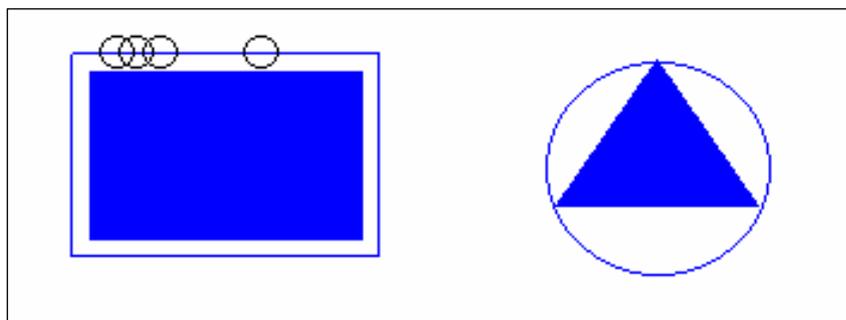
Pracovní prostředí je realizováno 2D prostorem, na kterém jsou známé překážky zobrazeny jako jednoduché konvexní útvary, modelované skupinami vrcholů.



Obr. 1 Pracovní prostor.

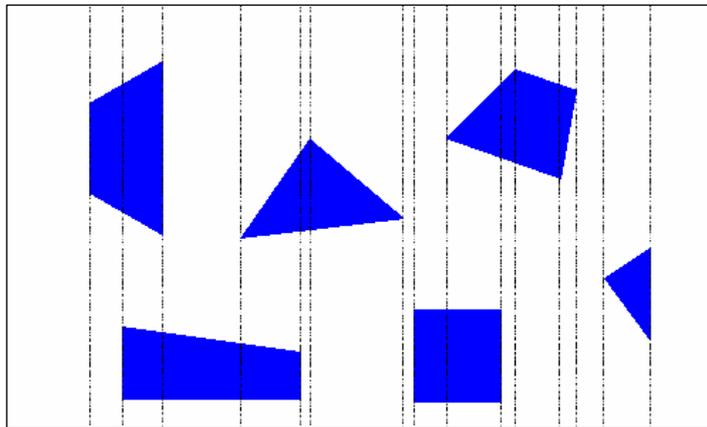
Hledáme nejkratší nekolizní cestu z počáteční pozice S (start) do koncového bodu T (target). Robot je chápán jako bod (pixel) a cesta robotu je pak znázorněna navazujícími úsečkami.

Aby nedošlo ke kolizi reálného robotu při projíždění těsně podél překážky, zvětšíme rozměry překážek minimálně o polovinu délky největšího rozměru robotu nebo se vytvoří nad překážkou kružnice opsaná (viz obr.2). Se zvětšováním lze počítat již při sestavování mapy.



Obr. 2 Způsoby zvětšení překážek.

Abychom mohli na tento model aplikovat PSO, musíme jej rozčlenit. Nejjednodušším způsobem je protnout každý vrchol vertikální přímkou (je možno použít kružnice, elipsy, spirály a šikmé přímky).

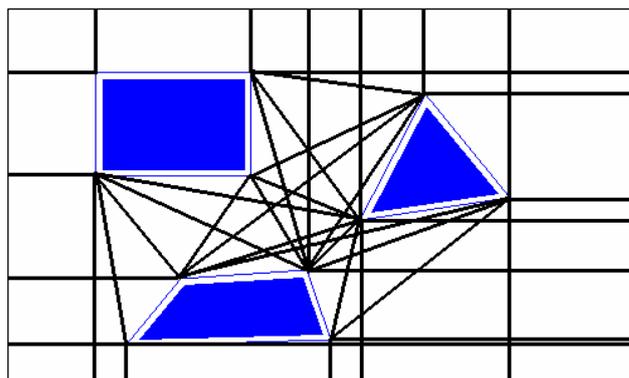


Obr. 3 Vertikální členění prostoru.

Další možností je vytvoření MAKLINK grafu [2],[3], pro který platí pravidla: každá MAKLINK čára je spojnicí mezi dvěma vrcholy nebo kolmou spojnicí vrcholu a hranicí mapy. Žádná čára nesmí protínat překážku.

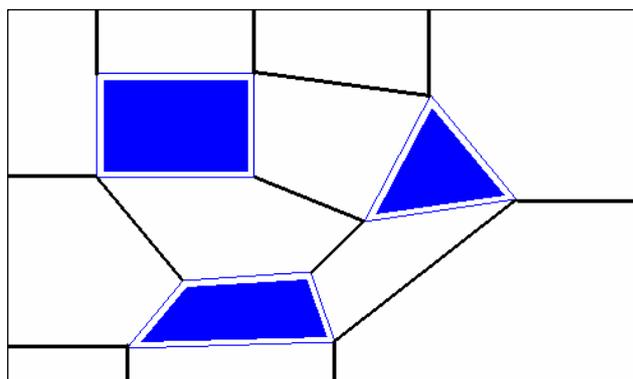
Postup pro vytvoření MAKLINK grafu:

1. Vytvořte všechny spojnice, které spojují vrcholy s ostatními a kolmé spojnice vrcholů s hranicí mapy (obr. 4).



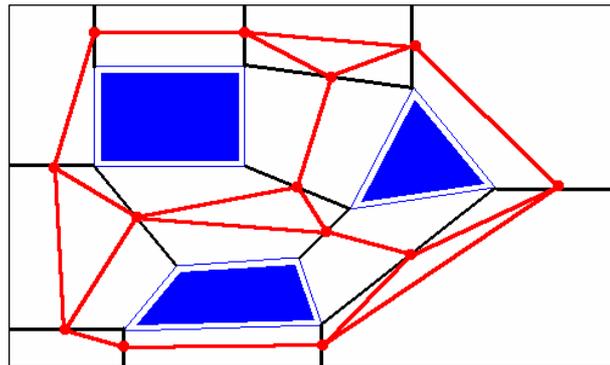
Obr. 4 Příprava MAKLINK grafu.

2. Smažte všechna přebytečná spojení tak, aby ze zbylých spojnic a okrajů překážek vznikly konvexní oblasti s největší rozlohou (obr. 5).



Obr. 5 Maximální konvexní prostory mezi překážkami.

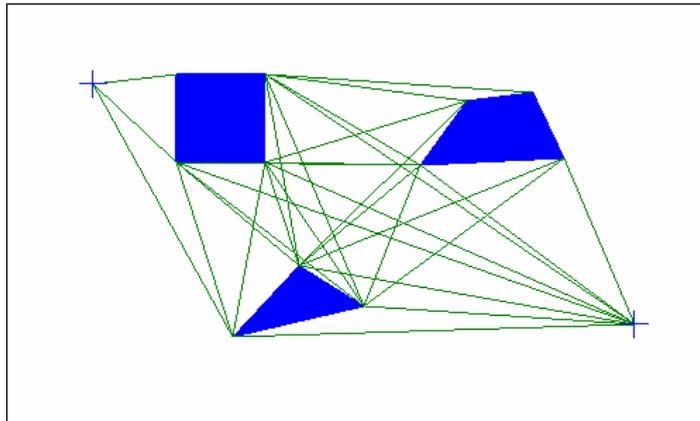
3. V každé konvexní oblasti propojte všechny středy spojnic, aby vznikla spojením všech oblastí síť (obr. 6).



Obr. 6 MAKLINK graf.

1.3 Graf viditelnosti a Dijkstrův algoritmus

V grafu viditelnosti představuje každý vrchol překážky, počáteční a koncový bod vrchol grafu. Spojnice mezi těmito vrcholy, které neprotínají žádnou překážku jsou hrany grafu.



Obr. 7 Graf viditelnosti.

Před aplikaci Dijkstrova algoritmu [5] ohodnotíme každou hranu její délkou. Pro každou hranu musí existovat hrana duplicitní, opačné orientace. Nejprve se počátečnímu vrcholu přiřadí hodnota 0, všem ostatním maximální hodnota. Poté procházíme seznam hran a zjišťujeme, zda hodnota přiřazená k výstupnímu uzlu hrany je nejvýše o délku hrany větší než hodnota přiřazená výstupnímu uzlu. Dokud dochází k opravám, opakujeme průchod seznamem hran. K určení cesty provedeme algoritmus zpětně od koncového bodu tak, aby platilo, že hodnota přiřazená koncovému uzlu hrany minus hodnota přiřazená počátečnímu uzlu je rovná délce hrany.

2 Plánování cesty robotu pomocí PSO

2.1 PSO a horizontální členění pracovního prostoru

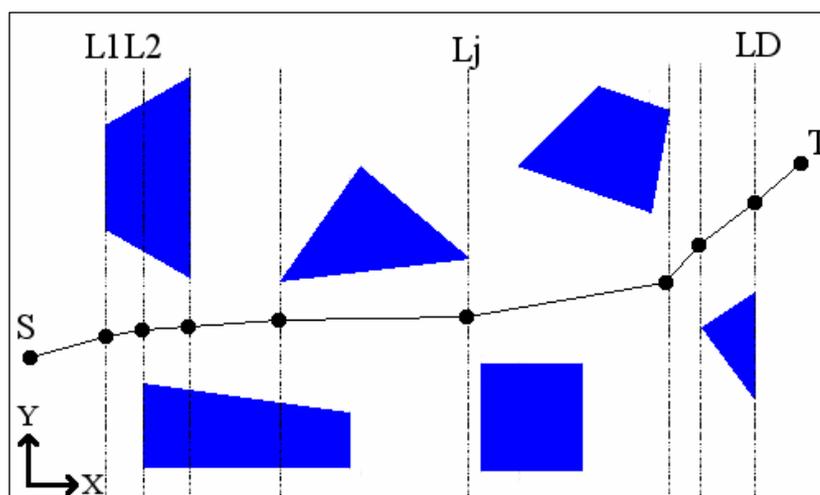
Tato část je inspirovaná článkem [1].

2.1.1 Původní přístup

Pro následné experimenty jej nazveme PSO1. Na rozdíl od PSO algoritmu má PSO1 pozměněn výpočet váhového parametru.

$$w(t) = \frac{1}{1 + \exp\left(0.015 * \left(t - \frac{T}{3}\right)\right)} \quad (8)$$

Rozdělíme prostor tak, že v každém vrcholu překážky povedeme svislici L , na které se budou pohybovat příslušné polohové složky vektoru částice. Tyto svislice utvoříme pouze mezi body S, T . Ke každému L přiřadíme parametry y_{jmin}, y_{jmax} . Tyto parametry reprezentují počáteční a koncové body překážky na svislici. Před vytvořením populace částic je zapotřebí svislice seřadit.



Obr. 8 Členění prostoru a zobrazení jedné částice (cesty robotu).

Složky vektoru polohy částice \vec{x}_i jsou měnící se y souřadnice na svislici L .

Předpokládejme, že souřadnice bodů S, T jsou $[x_0, y_0]$ a $[x_{D+1}, y_{D+1}]$. Délka dráhy robotu je pak dána vzorcem (9)

$$ST(\vec{x}_i) = \sum_{j=0}^D \sqrt{(x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2} \quad (9)$$

kde x_l, y_l až x_D, y_D jsou souřadnice polohy složek vektoru \vec{x}_i na mapě pracovního prostoru robotu. To samé platí pro funkci (10). Zde se přidává k délce ST velikost rozptylu v ose y .

$$f(\vec{x}_i) = ST(\vec{x}_i) + \sum_{j=0}^D \sqrt{(y_{j+1} - y_j)^2} \quad (10)$$

Aby se předcházelo kolizním stavům, k funkci $f(\vec{x}_i)$ se přidává penalizační funkce (11). Ta zajistí, že když se částice dostane v některé dimenzi mezi body y_{jmin} , y_{jmax} , tak vznikne penalizace λ (doporučená hodnota z [1] je $\lambda = 120$).

$$H(\vec{x}_i) = \sum_{j=1}^D \lambda \vec{x}_{ij} u_j(\vec{x}_i) \quad (11)$$

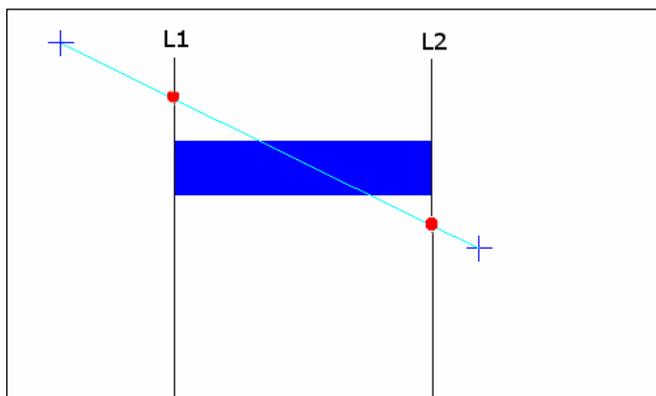
$$\text{IF } y_{jmin} \leq \vec{x}_{ij} \leq y_{jmax} \text{ THEN } u_j(\vec{x}_i) = 1 \text{ ELSE } u_j(\vec{x}_i) = 0$$

Finální kritériální funkce (12) se skládá z penalizační části a délky částice obohacené o její rozptyl v ose y.

$$F(\vec{x}_i) = f(\vec{x}_i) + H(\vec{x}_i) \quad (12)$$

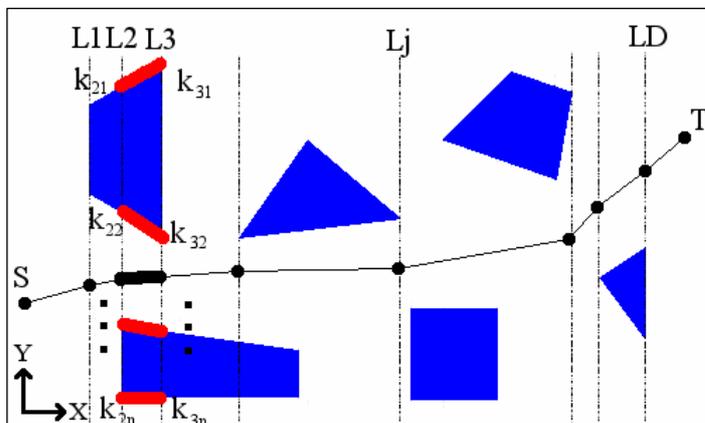
2.1.2 Upravený přístup

Tento přístup označíme PSO2. V předchozím řešení docházelo ke kolizím, kvůli nedostatečným podmínkám. Uvedu příklad (viz obr. 9). Oba červené body na spojnicích L1,L2 nejsou v prostoru překážky, avšak dochází ke kolizi.



Obr. 9 Nedokonalost původního algoritmu.

Tento problém jsme řešili přidáním další penalizační podmínky. Nejprve jsme vytvořili množinu K . K obsahuje všechny nevertikální hranice překážek $(k_{11}, k_{12}) \dots (k_{D-1n}, k_{Dn})$ mezi svislicemi L . Poté zkontrolujeme kolizi příslušné části částice s odpovídající částí K . Následně se provede penalizace o λ obdobně jako ve vzorci (11).



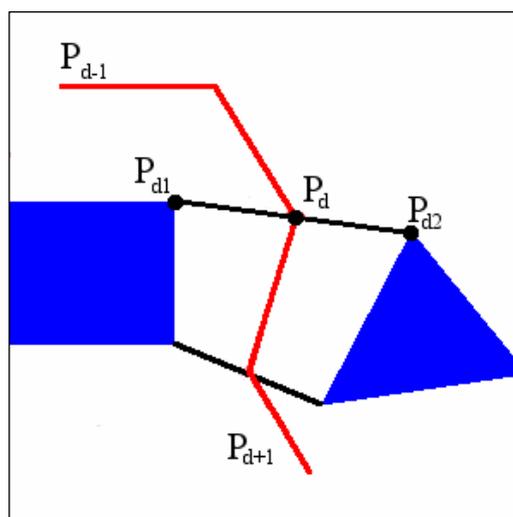
Obr. 10 Znárodnění množiny K mezi $L2$ a $L3$.

2.2 PSO + MAKLINK

Nejprve je zapotřebí přidat do MAKLINK sítě body S, T . Tímto vytvoříme graf, na který můžeme aplikovat Dijkstrův algoritmus. Vytvoříme tak nejkratší cestu MAKLINK grafem $P_0, P_1, P_2, \dots, P_D, P_{D+1}$, kde P_0 je počáteční bod a P_{D+1} je koncový bod trasy robotu. P_d ($d=1, 2, \dots, D$) je střed spojnice, tak jako na obr. 11. Úkolem optimalizace je nastavit pozici P_d tak, aby se zkrátila cesta. Poloha bodu P_d je na úsečce P_{d1}, P_{d2} dána rovnicí:

$$P_d = P_{d1} + (P_{d2} - P_{d1}) \times h_d, \quad (13)$$

$$h_d \in [0, 1]$$



Obr. 11 Označení bodů na spojnici.

Částice je pak dána skupinou po sobě jdoucích parametrů h_d . Tento parametr posouvá bod P_d na příslušné úsečce. Je-li $h_d=0$, pak je $P_d = P_{d1}$, je-li $h_d=1$, pak je $P_d = P_{d2}$ a je-li $h_d=0,5$, pak P_d leží mezi body P_{d1}, P_{d2} .

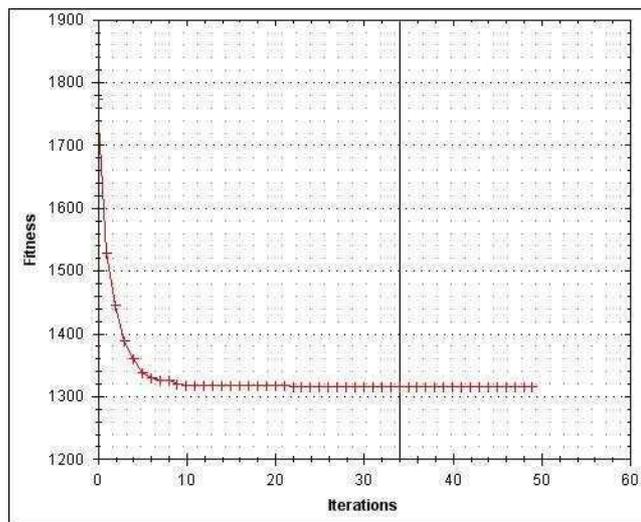
$$\vec{x}_i = \{h_{i1}, h_{i2}, \dots, h_{iD}\} \quad (14)$$

Optimalizovaná funkce je pak dána vzorcem (15). Je to součet vzdáleností mezi všemi body P_d .

$$f(\vec{x}_i) = \sum_{d=0}^D \text{vzdálenost}_{\{P_d(h_{id}), P_{d+1}(h_{id+1})\}} \quad (15)$$

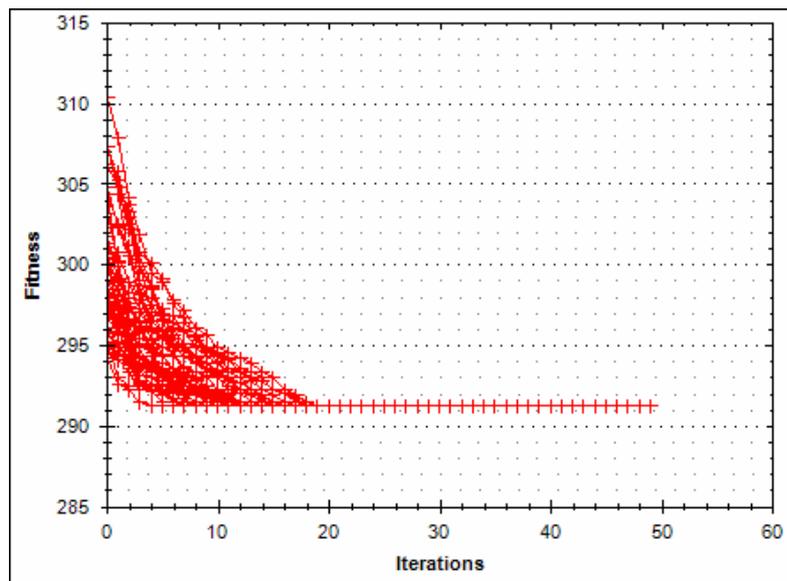
2.3 Ukončení běhu PSO

Důležitou součástí algoritmu je jeho ukončení. Extrémně dlouhý počet běhů zbytečně prodlužuje dobu trvání algoritmu a malý počet cyklů zapříčiní nedostatečnou konvergenci. Jednou z možností je sledování průběhu kriteriální funkce. Když se funkce nezmění několikrát za sebou, algoritmus se ukončí. T se nastaví na vysokou hodnotu. Ukončení se dá nastavit procentuálně z P . Na obr. 12 je znázorněno ukončení po 20% stálosti funkce (34. iterace z 50). Je vidět že další průběh byl zbytečný.



Obr. 12 Znáznornění 20% stálosti funkce.

Další možností je spustit vícekrát algoritmus mezi dvěma nejvzdálenějšími místy v pracovním prostoru robotu (obr. 13). Tak nastavíme maximální předběžný počet T , v tomto případě 20, které můžeme ještě zkrátit předchozím způsobem.



Obr. 13 Vícenásobné spuštění algoritmu k nastavení T .

algoritmus	D	Dráha [px]	Čas [ms]	T	P	Úspěšnost[%]
PSO1	7	319	44,22	60	100	45%
PSO2	7	331	45,76	50	100	88%
MAKLINK	3	327	2,23	20	30	100%

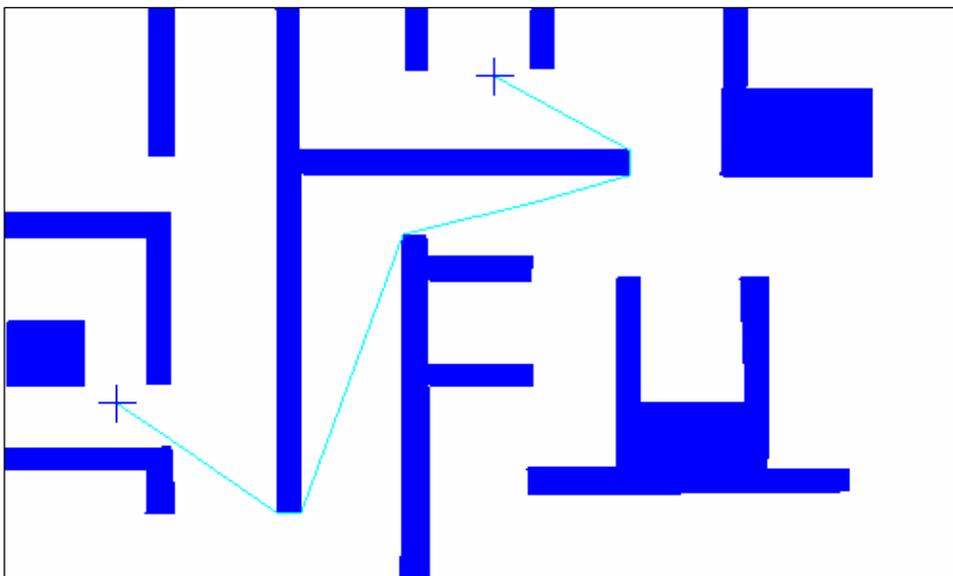
Změna oproti předchozím výsledkům je jen minimální. Lépe funguje, když je cesta více horizontální.

Pozorováním chování druhé mutace jsme zjistili, že nasazení druhé metody funguje, když je počet cyklů větší a tím se zvětšuje doba výpočtu. Parametry mutace záleží hodně na ostatních parametrech PSO a samotné členitosti prostoru.

Možnost použití těchto typů mutace je velice individuální. PSO ve spojitosti s MAKLINK grafem nepotřebuje mutaci.

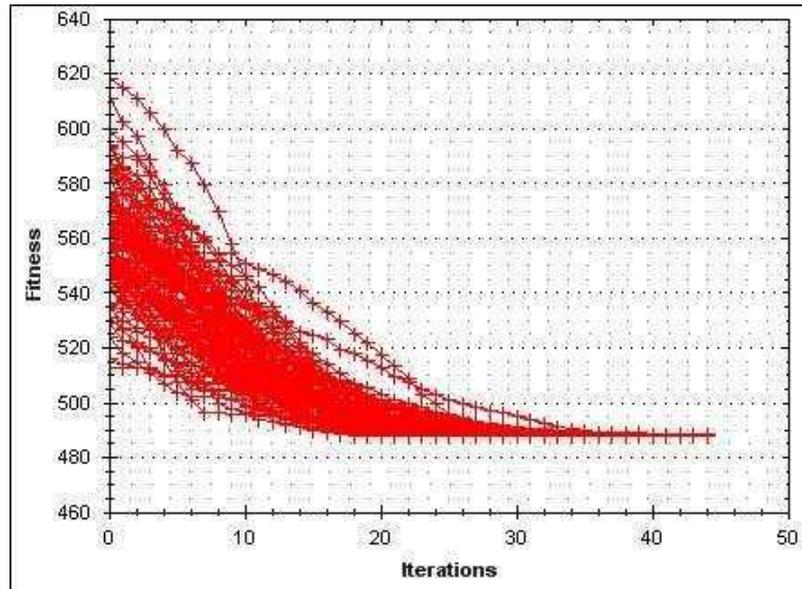
3.1.3 Nekonvexní překážky

V reálném světě neexistují pouze překážky znázornitelné konvexním útvarem. Lze znázornit nekonvexní překážku spojením několika konvexních překážek. V reálném světě jsou například takovéto překážky zdi budov. Pro použití PSO s vertikálním členěním je tento prostor až příliš složitý a úspěšnost by nebyla 100%. Budeme zde porovnávat pouze MAKLINK graf a Dijkstrův algoritmus.



Obr. 15 Cesta robotu mezi nekonvexními překážkami.

Oba algoritmy jsme spustili 100 krát při nastavení PSO: $S[59,209]$; $T[256:35]$; $w[0.4,0.9]$; $c1=c2=2$. Je zde vidět další výhoda MAKLINK grafu, tj. že robot se může vracet zpět ve všech osách a není tedy pouze funkcí osy x.



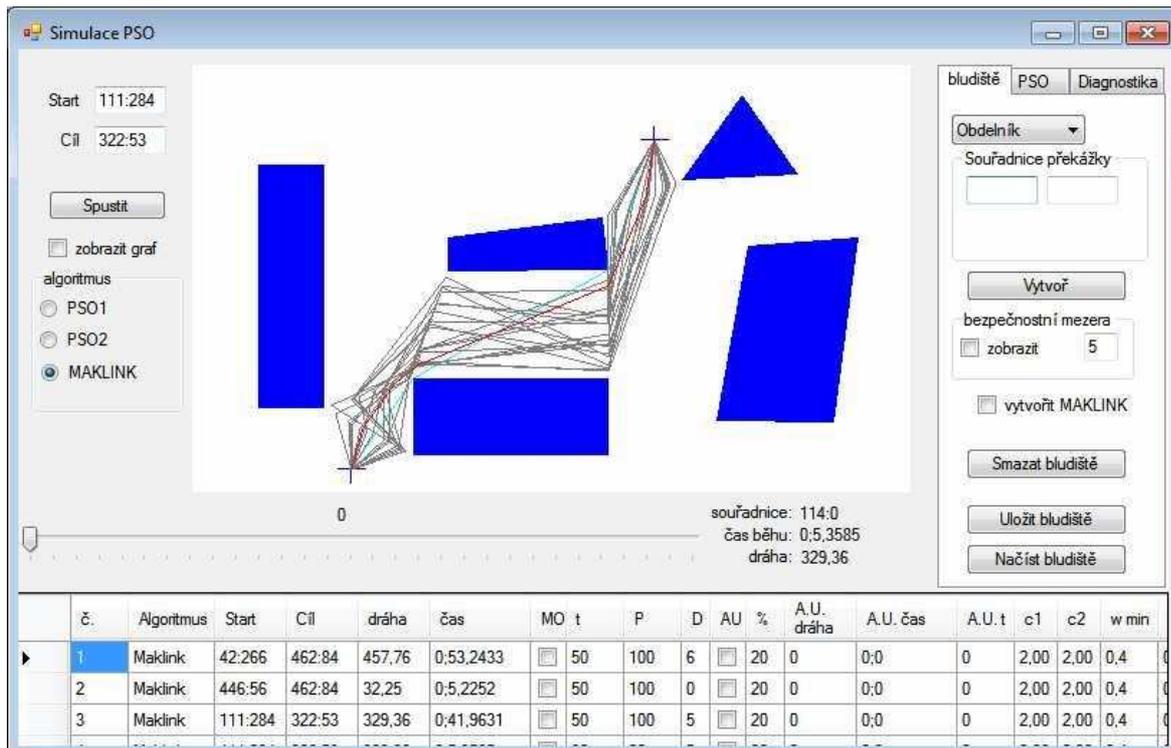
Obr. 16 Souhrnný graf vývoje optimalizované funkce.

MAKLINK graf s použitím PSO měl průměrný čas výpočtu 13,67ms při 45 cyklech a 40 částicích. Částice měla 8 dimenzí. Protože se složitostí pracovního prostoru robotu roste počet hran grafu viditelnosti a tím i doba výpočtu Dijkstrova algoritmu, je jeho průměrný čas 85 ms. Délka dráhy je v obou případech totožná.

3.2 Popis simulačního programu

Okno programu se dá rozdělit na několik částí. Vlevo se nachází pole pro souřadnice počáteční a koncové pozice, tlačítko pro spuštění simulace a výběr druhu algoritmu. Je zde také možnost zobrazení vývoje optimalizované funkce. Vpravo jsou záložky pro tvorbu bludiště, podrobné nastavení PSO algoritmu a diagnostická část zahrnující opakované spuštění vybraného algoritmu a znázornění pohybu částic, ovládané prvkem trackbar, v prostřední části obrazovky. Nahoře uprostřed se nachází prostor pro samotnou simulaci. Tabulka ve spodní části se automaticky plní výsledky předchozích simulací. Na začátku bývá skryta, roztažením okna směrem dolů se nám celá zobrazí. Zápis souřadnic do kolonek lze provádět výběrem příslušné kolonky a kliknutím v prostoru simulace.

V programu byla použita open source .NET knihovna ZedGraph s licencí Library General Public License a napsaná v jazyce C#. Je specializována na dvojrozměrnou vizualizaci dat. Tuto knihovnu jsme použili pro zobrazení grafu vývoje optimalizované funkce.



Obr. 17 Obrazovka simulace.

3.2.1 Tvorba bludiště

Bludiště, lze vytvořit výběrem typu překážky v horní části pravého panelu a zadáním souřadnic vrcholů. Lze nastavit i zvětšení překážky (bezpečnostní mezeru), ta je přednastavena na hodnotu 5pixelů. Bludiště lze načíst nebo uložit do souboru příponou .fff. Již hotová bludiště jsou součástí této práce. Vše lze vytvořit také z kontextového menu v prostoru simulace. V tomto menu lze také snímat obrazovku a uložit ji jako obrázek nebo načíst pozadí, podle kterého můžeme vytvořit překážky v simulaci.

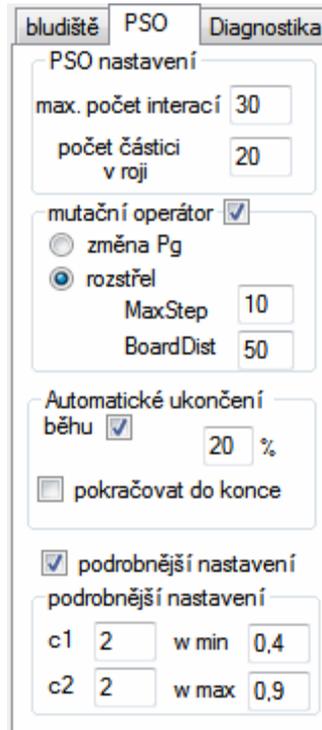
3.2.2 Tvorba MAKLINK grafu

Nejprve je zapotřebí zapnout mód tvorby grafu přes checkbox „vytvoř MAKLINK“. Zapnutím tohoto módu můžeme zjistit, zda graf již existuje. Poté kliknutím v blízkosti vrcholů vytvoříme úsečku, těmito úsečkami rozdělíme prostor na maximální konvexní prostory. Pro vytvoření kolmé úsečky spojující vrchol a hranici simulace vybereme požadovaný vrchol a myši se přiblížíme k hranici simulace. Když hrana změní barvu na růžovou, můžeme jí kliknutím vybrat. MAKLINK graf se také dá uložit nebo načíst ze souboru, kde jsou uloženy překážky.

3.2.3 Spuštění algoritmů

Pokud jsou nastaveny všechny potřebné parametry, spustíme vybraný algoritmus tlačítkem start v levé části okna.

3.2.4 Podrobnější nastavení PSO



Obr. 18 Podrobnější nastavení algoritmu PSO.

Na tomto panelu se dají modifikovat následující parametry PSO:

- Počet iterací T .
- Počet částic v roji P .
- Zapnutí mutačního operátoru.
- Zvolení jednoho z druhů mutace.
- Nastavení automatického ukončení.
- Možnost zobrazit pouze hodnoty automatického ukončení a nechat proběhnout dokonce celý algoritmus.
- A nakonec možnost změnit váhové a učící parametry.

3.2.5 Diagnostické funkce

Na panelu „Diagnostika“ lze nastavit počet opakování zvoleného algoritmu a dále pak zapnout zobrazení vývoje částic ovládaných trackbarem, který se po zvolení této možnosti zobrazí. Částice mají šedou barvu a globální částice má červenou barvu. Lze zapnout jen zobrazování globální částice. Je zapotřebí po této volbě sledovaný algoritmus spustit.

Ve spodní části se zobrazují předchozí zaznamenané parametry a výsledné hodnoty předchozích již spuštěných algoritmů. Kliknutím na sloupce dráha a čas se zobrazí průměrná hodnota. Přejetím myši nad řádkem se zobrazí v simulaci navržená cesta, která odpovídá příslušnému řádku. V kontextovém menu lze celou tabulku smazat.

Závěr

Cílem této práce bylo analyzovat přístupy k plánování cesty robotu a popsat problematiku hejnových algoritmů. Nejprve jsme kompletně prostudovali algoritmus PSO. Jeho výhodou je práce přímo s hodnotami optimalizované funkce. Nemusíme převádět hodnoty do Grayova kódu, jako je tomu u genetického algoritmu. Nevýhodou je konvergence do lokálního minima. V experimentech jsme se snažili tomuto předejít mutačními operátory. Došli jsme k závěru, že použití těchto operátorů je velice individuální. To platí i pro samotné parametry mutace.

Porovnávali jsme tři možnosti přístupu k aplikování řízení robotu pomocí PSO. První byl inspirován článkem [1], druhý byla jeho obměna přidáním vlastní penalizační funkce. To se ukázalo jako velké zlepšení, ale jeho účinnost nebyla 100%. Poslední metoda byla inspirována článkem [2], spojení MAKLINK grafu a PSO. Tato metoda se v experimentech projevila jako nejúspěšnější. Předčila i srovnávací graf viditelnosti spojený s Dijkstrovým algoritmem. Výhodou MAKLINK grafu je jeho 100% účinnost, malá rychlost (daná malou populací a malým počtem cyklů i ve složitém prostředí). Není nutné používat mutace.

V práci je popsáno ovládání vytvořeného simulačního programu, který je na přiloženém CD. Simulaci lze testovat výše zmíněné přístupy k řízení robotu, za pomoci algoritmu PSO, v dvojrozměrném prostředí. Lze vytvořit vlastní mapu pracovního prostoru nebo použít již hotové mapy. V simulaci můžeme také sledovat pohyb částic při hledání cesty a tak lépe pochopit práci tohoto algoritmu a lépe nastavit jeho parametry.

Díky velké rychlosti výpočtu by bylo možné nasadit tento model řešení do dynamického prostředí a upravit přístup tak, aby výslednou cestu tvořila křivka. Pomocí PSO se dá řídit i skupina robotů, kteří hledají nějaký cíl. V budoucnosti je možno se pokusit nasadit tento model řízení na reálného robotu.

Seznam použité literatury

- [1] LEI, K., QIU, Y., HE, Y. A Novel Path Planning for Mobile Robots Using Modified Particle Swarm Optimizer. *In Proceedings of the 1st International Symposium on Systems and Control in Aerospace and Astronautics ISSCAA 2006*, 19-21 Jan. 2006, 4 pp.
- [2] QIN, Y.-Q., SUN, D.-B., LI, N., CEN, Y.-G. Path Planning for Mobile Robot Using the Particle Swarm Optimization with Mutation Operator. *In Proceedings of 2004 International Conference on Machine Learning and Cybernetics*, 26-29 Aug. 2004, vol. 4, pp. 2473–2478.
- [3] KHAMIS, A. <akhamis@pami.uwaterloo.ca> Swarm Inteligence in Robotics. [PDF prezentace]. PAMI Reserch Group - University of Waterloo 2009 [cit. 27.3.2010]. Dostupné z <<http://129.97.86.45:88/redmine/attachments/11/SI-Robotics.pdf>>
- [4] DLOUHÝ, M. Exaktní plánování [online]. 11.7.2003 [cit. 8.3.2010]. Dostupné z <<http://robotika.cz/guide/exactplan/cs>>
- [5] PROKOP, J. *Algoritmy v jazyku C a C++: Praktický průvodce*. 1. vyd. Praha: Grada Publishing, 2009. 160 s. ISBN 978-80-247-2751-6.
- [6] MEYER, J.-A., FILLIAT, D. Map-Based Navigation in Mobile Robots: *II. A Review of Map-Learning and Path-Planning Strategies*. *Cognitive Systems Research*, vol. 4, issue 4, 2003, pp. 283-317.

