ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra řídicí techinky

Bakalářská práce

Vojtěch Pavlík

Algoritmy skupinové inteligence pro využití v multi-robotických úlohách

Vedoucí práce: **Ing. Martin Saska, Dr. rer. nat.**

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Vojtěch Pavlík**

Studijní program: Kybernetika a robotika
Obor: Systémy a řízení

Název tématu: **Algoritmy skupinové inteligence pro použití v multi-robotických úlohách**

Pokyny pro vypracování:

Cílem práce je rozšířit optimalizační algoritmus PSO pro potřeby řízení roje autonomních helikoptér. V implementovaném přístupu budou jednotlivé částice PSO reprezentovat fyzické roboty, které se budou pohybovat na základě pravidel PSO. V rámci této práce bude do těchto pravidel integrován kinematický model helikoptér a omezení dané relativní lokalizací entit robotického roje.

1. Implementujte model helikoptéry zahrnující reálná omezení jejího pohybu [1].
2. Uvažujte a popište omezení daná relativní lokalizací členů roje.
3. Navrhněte a implementujte vhodnou decentralizaci PSO respektující omezení reálného roje [2].
4. Navrhněte a implementujte pravidla PSO pro potřeby alespoň dvou konkrétních multi-robotických aplikací roje [3].
5. Přizpůsobte navrženou metodu pro potřeby pozemních robotů a ověřte její funkčnost s využitím platformy SyRoTek [4].

Seznam odborné literatury:

[1] Taeyoung Lee; Leoky, M.; McClamroch, N.H.; , Geometric tracking control of a quadrotor UAV on SE(3), Decision and Control (CDC), 2010 49th IEEE Conference on , vol., no., pp.5420-5425, 15-17 Dec. 2010

[2] J. Kennedy and R.C. Eberhart Particle swarm optimization, in Proceedings International Conference on Neural Networks IEEE, volume 4, pp. 1942–1948, 1995.

[3] J. Kennedy and Eberhart R.C., Swarm Intelligence, Morgan Kaufmann Publishers, 2001.

[4] Kulich, M. - Košnar, K. - Chudoba, J. - Faigl, J. - Přeučil, L.: On a Mobile Robotics E-learning System. In Proceedings of the Twentieth European Meeting on Cybernetics and Systems Research. Vienna: Austrian Society for Cybernetics Studies, 2010, p. 597-602. ISBN 978-3-85206-178-8.

Vedoucí: Ing. Martin Saska, Dr. rer. nat.

Platnost zadání: do konce zimního semestru 2012/2013

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 18. 1. 2012

Prohlášení

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

V Praze dne 24. 5. 2012 .....

Poděkování

Tímto bych chtěl poděkovat Ing. Martinu Saskovi, Dr. rer. nat. za vedení práce a řešení problémů, které s tím vyvstaly. Dále pak Ing. Janu Chudobovi za vstřícnost při řešení problémů s robotickou platformou SyRoTek, Mgr. Michalu Kožuchovi za pomoc se spuštěním šesti robotů SyRoTek. Dále bych rád poděkoval své rodině za její podporu a trpělivost.

Abstrakt

Tato práce se zabývá modifikací algoritmu Particle Swarm Optimization (PSO), která umožní jeho přímé použití pro řízení roje reálných robotů. V prezentovaném přístupu jsou jednotlivé roboty uvažovány jako jedinci PSO roje. Na rozdíl od virtuálních bezrozměrných entit klasického PSO je v navrhovaném reálném PSO nutné uvažovat a implementovat omezení daná robotickými roji. Konkrétně se jedná o integraci reálného modelu pohybu robotů, omezení pohybu robotů, omezení daná relativní lokalizací a komunikací mezi sousedy v roji a omezení daná velikostí robotů a jejich vzájemným ovlivňováním. Další částí je pak úprava algoritmu pro potřeby robotické platformy SyRoTek.

Abstract

This thesis covers modifications of the Particle Swarm Optimization (PSO) algorithm, which allows its direct use to control swarms of real robots. In the presented approach, individual robots are considered as particles in the PSO swarm. Contrary to a virtual swarm of dimensionless entities in the classical PSO, in engineered PSO the constraints of the real robotic swarm need to be considered and implemented. Specifically, the motion model of real robots, robot movement restrictions, restrictions given by relative locations of the robots, communication between neighbours in a swarm, the size constraints of the robots and their interaction have to be considered. The next step is the modification of the algorithm for the needs of the robotic platform SyRoTek.

Obsah

Seznam obrázků

Seznam tabulek

Seznam algoritmů

1 Úvod

1.1 Motivace

Particle Swarm Optimization algoritmus (PSO algoritmus) patří do skupiny tzv. evolučních algoritmů snažících se napodobit chování hejna živých organismů. Tento algoritmus je efektivní pro hledání optimálního řešení ve velkém počtu dimenzí. Extrém je hledán rojem, kde jsou jedinci tohoto roje bezrozměrní. Otázkou je tedy, jak se tento algoritmus bude chovat pro roboty reálných rozměrů a omezení, která s reálným modelem souvisí.

Algoritmus by do budoucna měl být použit na létajících robotech, konkrétně na kvadrikoptérách. Z modelu tohoto robotu vyvstane několik omezení, která nejsou v PSO běžně užívána. Vzhledem k tomu, že v průběhu vypracování této práce nebudou ještě kvadrikoptéry připraveny pro přidávání uživatelských programů, jako je upravený PSO algoritmus, budou testy probíhat na robotické platformě SyRoTek.

Od doby uvedení PSO algoritmu byly prezentovány modifikace algoritmu, které zlepšují jeho konvergenci a efektivitu. Většina modifikací se zabývá omezením kmitání roje okolo hledaného extrému. V navrhovaných úpravách algoritmu budou některé z těchto bezrozměrných modifikací vztaženy na fyzické roboty a parametry modifikace budou nastaveny podle reálného modelu robotu.

Další omezení, která budou implementována, souvisí s omezeným dosahem robotů. Ta mají svůj význam pro konvergenci algoritmu, ale také i pro realizovatelnost algoritmu na robotech - PSO algoritmus potřebuje pro svoji funkci znát souřadnice robotů v prostoru. Do budoucna je počítáno s relativní lokalizací robotů pomocí kamer umístěných na jednotlivých robotech. Algoritmy, které jsou pro tuto aplikaci vyvíjeny, potřebují k vyhodnocení pozic robotů data z více kamer různých robotů a je tedy nutné, aby roboty zůstávaly ve vzájemném vizuálním i komunikačním dosahu.

Aplikace takto upraveného algoritmu může být například hlídání obrazů v galerii, sběr kontaminovaných objektů či modelování signálu vysílačů. Součástí práce budou i simulace

dvou robotických aplikací, konkrétně se jedná o rozsvěcení nedostatečně osvětlené místnosti a hledání místa s největším signálem vysílačů.

1.1.1 SyRoTek

Jak již bylo zmíněno, pro testy na reálných robotech bude použita robotická platforma SyRoTek. Algoritmus pro ni bude muset být upraven - roboty SyRoTek se pohybují v planárním prostředí. Pro hledání extrému zde bude využita 2D mapa ve stupních šedi, po které budou roboty jezdit a podlahovým senzorem měřit, na jaké barvě stojí.

1.2 Přínos práce

- přechod od bezrozměrných entit v klasickém PSO na roboty reálných rozměrů a omezení
- zjištění, jak reálná velikost robotů ovlivní konvergenci PSO algoritmu
- zjištění, je-li navrhované řešení použitelné pro roje reálných robotů
- zjištění, je-li algoritmus použitelný pro relativně lokalizované roboty
- úpravy algoritmu umožňující jeho nasazení na reálné roboty
- vylepšení platformy SyRoTek

2 Teoretická příprava a simulační prostředí

2.1 Particle Swarm Optimization algoritmus

Particle Swarm Optimization algoritmus (PSO algoritmus) patří do skupiny tzv. evolučních algoritmů snažících se napodobit chování hejna živých organismů. PSO je využíváno pro hledání globálního extrému v n-dimenzionálním prostoru s využitím více částic. Při využití pouze jedné částice by bylo možno základní verzi PSO algoritmu přirovnat ke gradientnímu hledání extrému. Gradientní algoritmus ale skončí v nejbližším lokálním extrému, který nemusí být nutně globálním extrémem. Základní verze PSO algoritmu využívá více částic, které sdílí informaci o výhodnosti jejich aktuálního stavu.

Výhodou algoritmu je malá paměťová a výpočetní náročnost a schopnost poradit si s nelineárními i nespojitými funkcemi s mnoha lokálními extrémy.

2.1.1 Krátce o vzniku PSO

PSO algoritmus byl poprvé představen v roce 1995 pány Jamesem Kennedym a Russellem C. Eberhartem. V článku "Particle Swarm Optimization" [5] popisují optimalizaci nelineární funkce pomocí roje částic. V experimentech zkoumali sociální chování jedinců se snahou napodobit hejno ptáků hledajících potravu.

2.1.2 Základní verze algoritmu

Nejjednodušší verzi PSO je možno popsat dvěma rovnicemi

$$\mathbf{x}_{k+1}^i = \mathbf{x}_k^i + \mathbf{v}_{k+1}^i \tag{1}$$

$$\mathbf{v}_{k+1}^i = \mathbf{v}_k^i + c_1 r_1 (\mathbf{p}_k^i - \mathbf{x}_k^i) + c_2 r_2 (\mathbf{p}_k^g - \mathbf{x}_k^i). \tag{2}$$

Význam jednotlivých symbolů je:

- $\mathbf{x}_k^i$ pozice i-té částice
- $\mathbf{v}_k^i$ rychlost i-té částice
- $\mathbf{p}_k^i$ nejlepší zapamatovaná pozice částice (personal-best)
- $\mathbf{p}_k^g$ nejlepší zapamatovaná pozice roje (global-best)
- $c_1,\ c_2$ parametry osobní a sociální inteligence
- $r_1,\ r_2$ náhodná čísla z intervalu $< 0; 1 >$

Základní verzi algoritmu je možno popsat rovnicí (1) pro změnu polohy, kde se $\mathbf{v}_{k+1}^i$ vypočítá podle rovnice (2).

Hlavní výhodou základního algoritmu je malé množství parametrů, které je potřeba ladit. Jediné dva laditelné parametry jsou zde c_1 a c_2, které určují, jak velkou váhu dávají jednotlivé částice svojí či globální informaci o výhodnosti pozice. V literatuře [6] jsou popsána doporučení na omezení hodnoty těchto parametrů. Pokud jsou použity hodnoty vyšší než doporučené, dochází k nekontrolovatelné expanzi a částice se vzdalují od extrému. V této práci je použito nastavení prezentované v [7], konkrétně $c_1 = c_2 = 2$.

Základní algoritmus je popsán v Algoritmus 1.

Result: Find global optima

initializatize $(\mathbf{v}, \mathbf{x})$ where $\mathbf{v^i}$ is i-th particles velocity $\mathbf{x^i}$ is i-th particles position.

Constants: $NumberOfGenerations$, $NumberOfParticles$, c_1, c_2 ;

for $k = 1; k \leq NumberOfGenerations$ **do**

  for $i = 1; i \leq NumberOfParticles$ **do**

    evaulate function value f_k^i using design space coordinates $\mathbf{x}_k^i$;

    if $f_k^i \leq p_{best}^i$ **then**

      $f_{best}^i = f_k^i, \mathbf{p}_k^i = \mathbf{x}_k^i$ `// update personal best;`

    if $f_k^i \leq p_{best}^g$ **then**

      $f_{best}^g = f_k^i, \mathbf{p}_k^g = \mathbf{x}_k^i$ `// update global best;`

  for $i = 1; i \leq NumberOfParticles$ **do**

    $\mathbf{r}_1 = rand(0, 1)$;

    $\mathbf{r}_2 = rand(0, 1)$;

    $\mathbf{v}_{k+1}^i = \mathbf{v}_k^i + c_1 r_1(\mathbf{p}_k^i - \mathbf{x}_k^i) + c_2 r_2(\mathbf{p}_k^g - \mathbf{x}_k^i)$ `// update velocity of i-th particle;`

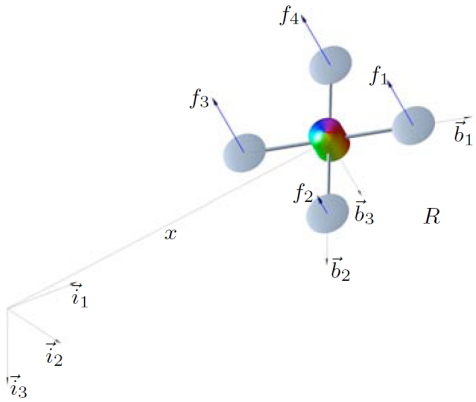
    $\mathbf{x}_{k+1}^i = \mathbf{x}_k^i + \mathbf{v}_{k+1}^i$ `// update position of i-th particle;`

Algoritmus 1: Základní PSO algoritmus

2.2 Kvadrikoptéra

Jedná se o speciální druh helikoptéry, která je poháněna čtyřmi rotory. Na Obrázku 1 je vidět příklad takovéhoto robotu.

Obrázek 1: Příklad kvadrikoptéry[1]



Obrázek 2: Model kvadrikoptéry [2]

Robot se skládá ze dvou párů rotorů, které se otáčí na opačnou stranu. Tyto rotory jsou umístěny ve vrcholech čtvercového rámu. Kvadrikoptéru je možno popsat modelem [2]:

Uvažujme model ukázaný na Obrázku 2. Jedná se o systém se čtyřmi identickými rotory umístěnými ve vrcholech čtvercového rámu, které vytváří tah v rovině kolmé k tomuto rámu. Pokud zavedeme referenční vztažnou soustavu $\{\vec{i_1}, \vec{i_2}, \vec{i_3}\}$ a vztažnou soustavu spojenou s tělem kvadrikoptéry $\{\vec{b_1}, \vec{b_2}, \vec{b_3}\}$ a definujeme:

$m \in \mathbb{R}^3$	celková hmotnost
$J \in \mathbb{R}^{3\times 3}$	matice setrvačnosti vzhledem k soustavě spojené s tělem kvadroptéry
$R \in SO(3)$	rotační matice ze soustavy spojené s tělem kvadroptéry do inerciální soustavy
$\mathbf{\Omega} \in \mathbb{R}^3$	úhlová rychlost v soustavě spojené s tělem kvadrikoptéry
$\mathbf{x} \in \mathbb{R}^3$	pozice hmotného středu v inerciální soustavě
$\mathbf{v} \in \mathbb{R}^3$	rychlost hmotného středu v inerciální soustavě
$d \in \mathbb{R}$	vzdálenost středu každého rotoru od hmotného středu kvadrikoptéry v rovině $\vec{b}_1$ $\vec{b}_2$
$f_i \in \mathbb{R}$	tah generovaný i-tou vrtulí v ose $-\vec{b}_3$
$\tau_i \in \mathbb{R}$	kroutící moment generovaný i-tou vrtulí v ose $\vec{b}_3$
$f \in \mathbb{R}$	celkový tah, $f = \Sigma_{i=1}^{4} f_i$
$M \in \mathbb{R}^3$	celkový moment v soustavě spojené s tělem kvadrikoptéry

Potom je možno model popsat rovnicemi

$$\dot{x} = v, \tag{3}$$

$$m\dot{v} = mge_3 - fRe_3, \tag{4}$$

$$\dot{R} = R\hat{\Omega}, \tag{5}$$

$$J\dot{\Omega} + \Omega \times J\Omega = M. \tag{6}$$

Vzhledem k mechanickému sestavení robotu a směru otáčení vrtulí je možno vypočítat celkový tah a momenty síly v jednotlivých osách pomocí matice

$$\begin{pmatrix} f \\ M_1 \\ M_2 \\ M_3 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & -d & 0 & d \\ 0 & 0 & -d & 0 \\ -c_{\tau f} & c_{\tau f} & -c_{\tau f} & c_{\tau f} \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix}. \tag{7}$$

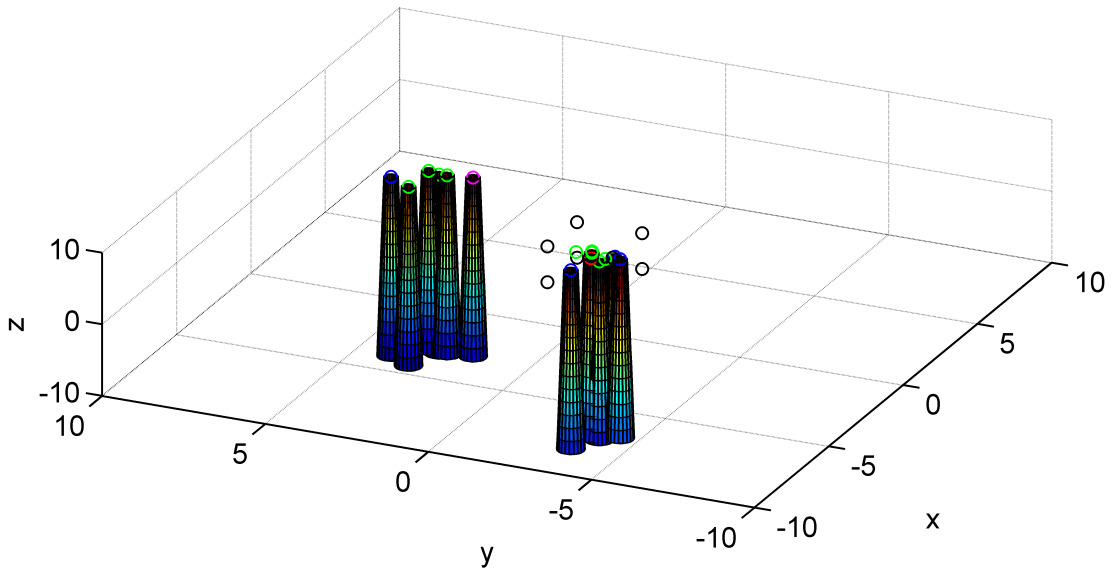
2.3 Simulační prostředí

Pro testování a vizualizaci výsledků PSO algoritmu bylo potřeba vytvořit simulační prostředí. Snímek ze simulace je vidět na Obrázku 3. Toto prostředí je upraveno pro simulaci robotické úlohy, kdy roboty hledají místo, kde je největší síla signálu vysílačů. Fitness funkce pro tuto simulaci je

$$f^i = \sum_{k=1}^{m} \sqrt{\sum_{d=1}^{3} (x_d^i - z_d^k)^2}, \tag{8}$$

kde f^i je fitness i-té částice, m je počet vysílačů, d je dimenze prostoru, x^i je pozice středu i-tého robotu a z^k je pozice k-tého vysílače. V jednoduchosti se jedná o součet vzdáleností robotu a jednotlivých vysílačů v kartézském prostoru. Pokud vycházíme z předpokladu, že síla signálu se vzdáleností od vysílače klesá, je tato vzdálenost minimalizována pro nalezení místa s největším signálem.

Význam jednotlivých bodů, které se vyskytují v legendě k simulačnímu prostředí, a jejich označení bude popsáno v kapitole 3.



Obrázek 3: Snímek ze simulačního prostředí

Legenda:

modré kroužky	pozice středu robotu
zelené kroužky	pozice personal-best
červené kroužky	pozice global-best
purpurové kroužky	pozice local-best
černé kroužky	poloha vysílače
válcové plochy	povrch tělesa, kterým je robot aproximován. Pro zlepšení viditelnosti je zobrazena jen spodní polovina tohoto tělesa.

3 Varianty PSO algoritmu

3.1 Omezení daná fyzickou realitou robotů

V knize [6] je možno najít mnoho variant algoritmu. Některé jsou založeny na tom, jak volit parametry c_1 a c_2, další pak upravují samotný algoritmus. Základní PSO algoritmus má tu nevýhodu, že částice kolem hledaného extrému kmitají ve velké vzdálenosti. Dokonce pro velké množství experimentů bylo možno vysledovat sinusový průběh tohoto kmitání.

Některé další varianty algoritmu potřebují také ladit parametry. Vzhledem k tomu, že je algoritmus použit na reálných robotech, velké množství těchto parametrů je dáno přímo modelem robotu.

3.1.1 Setrvačnost (inertia weight)

Tato modifikace algoritmu mění vliv rychlosti částice z minulé generace. Existují dvě různé verze - konstantní setrvačnost a zmenšující se setrvačnost. Reálnému modelu bude lépe vyhovovat verze s konstantní setrvačností - zde je možno si představit reálný robot. Takovémuto robotu se logicky nemění setrvačnost a je možno tuto konstantu změřit. Setrvačnost se značí $\mathbf{w}$. V původní verzi algoritmu se změní pouze rovnice pro aktualizaci rychlosti

$$\mathbf{v}_{k+1}^i = \mathbf{w}_k \mathbf{v}_k^i + c_1 r_1 (\mathbf{p}_k^i - \mathbf{x}_k^i) + c_2 r_2 (\mathbf{p}_k^g - \mathbf{x}_k^i). \tag{9}$$

Rovnice pro aktualizaci setrvačnosti je potom

$$\mathbf{w}_{k+1} = \mathbf{w}_k. \tag{10}$$

Pokud je hodnota setrvačnosti zvolena v intervalu $(0; 1)$ klesá vliv rychlosti, kterou částice měla v minulé iteraci algoritmu. Díky tomu částice více směřuje ke známým výhodným pozicím. Je-li parametr volen vyšší než 1, částice kmitají čím dál více od hledaného extrému.

Vzhledem k tomu, že se roboty pohybují v trojdimenzionálním prostoru, je i setrvačnost trojdimenzionální. Otázkou je, jestli je některá z dimenzí preferovaná a tedy jestli není vhodné použít pro různé dimenze různou setrvačnost. Vzhledem k symetrii aproximace robotu a prostředí, ve kterém se roboty pohybují, jsou dimenze x a y rovnocenné, jediná výjimka je v dimenzi z, kde působí gravitační zrychlení a na robota by případně působily i větší odporové síly, pokud by byly započteny. V uvedeném přístupu bude hodnota pro všechny prohledávané dimenze stejná.

Jak již bylo zmíněno výše, je vhodné, aby setrvačnost byla menší než 1, po několika simulacích byla zvolena hodnota $0,9$. Tato hodnota zajišťuje kmitání robotů kolem nalezeného extrému. Toto hraje velkou roli, pokud částice nalezla lokální extrém. Pokud kmitá kolem tohoto extrému, je možné, že objeví cestu k jinému, třeba i globálnímu, extrému. Dále je tím zajištěno, že rychlosti z minulých generací postupně ztrácí váhu pro určení nové rychlosti. Setrvačnost je potom

$$\mathbf{w} = \begin{pmatrix} 0.9 \\ 0.9 \\ 0.9 \end{pmatrix}. \tag{11}$$

3.1.2 Omezení rychlosti (v-max)

Další z možností, jak zajistit, aby se částice příliš nevzdalovala od hledaného extrému, je přidání omezení rychlosti. Díky tomu neroste rychlost nade všechny meze. I toto je využitelné pro reálné roboty a má to u nich nějaký fyzikální význam. Rovnice pro výpočet zůstávají stejné, pouze v algoritmu se po aktualizaci rychlosti zkontroluje, jestli nová rychlost není větší (menší) než dané maximu (minimum). Případně se rychlost nahradí touto maximální (minimální) hodnotou.

Algoritmus se změní přidáním podmínky pro v-max viz Algoritmus 2.

Pro určení hodnot v-max omezení, je možné použít model kvadrikoptéry převzatý z [2]. Tento model byl popsán v kapitole 2.2. Rychlost má v PSO význam změny polohy za

Result: Find global optima
initializatize $(\mathbf{v}, \mathbf{x})$ where $\mathbf{v^i}$ is i-th particles velocity $\mathbf{x^i}$ is i-th particles position.
constants: $NumberOfGenerations$, $NumberOfParticles$, $\mathbf{v}_{max}$, $\mathbf{w}$, c_1, c_2 ;
for $k = 1; k \leq NumberOfGenerations$ **do**
  for $i = 1; i \leq NumberOfParticles$ **do**
    evaulate function value f_k^i using design space coordinates $\mathbf{x}_k^i$;
    if $f_k^i \leq p_{best}^i$ **then**
      $f_{best}^i = f_k^i, \mathbf{p}_k^i = \mathbf{x}_k^i$ `// update personal best;`
    if $f_k^i \leq p_{best}^g$ **then**
      $f_{best}^g = f_k^i, \mathbf{p}_k^g = \mathbf{x}_k^i$ `// update global best;`
  for $i = 1; i \leq NumberOfParticles$ **do**
    $\mathbf{r}_1 = rand(0, 1)$;
    $\mathbf{r}_2 = rand(0, 1)$;
    $\mathbf{v}_{k+1}^i = \mathbf{w}_k \mathbf{v}_k^i + c_1 r_1 (\mathbf{p}_k^i - \mathbf{x}_k^i) + c_2 r_2 (\mathbf{p}_k^g - \mathbf{x}_k^i)$ `// update velocity of i-th particle;`
    if $\mathbf{v}_{k+1}^i > \mathbf{v}_{max+}$ **then**
      `// check velocity constriction;`
      $\mathbf{v}_{k+1}^i = \mathbf{v}_{max+}$
    else if $\mathbf{v}_{k+1}^i < -\mathbf{v}_{max}$ **then**
      $\mathbf{v}_{k+1}^i = -\mathbf{v}_{max}$
    $\mathbf{x}_{k+1}^i = \mathbf{x}_k^i + \mathbf{v}_{k+1}^i$ `// update position of i-th particle;`

Algoritmus 2: PSO algoritmus s využitím inertia weight a v-max

jednu generaci - jedná se tedy o průměrnou rychlost. Plánování dynamického pohybu je nad rámec této práce, a proto byl model zjednodušen:

- $\Omega = 0$ kvůli významu rychlosti v PSO algoritmu a tedy $M = 0$
- poměr maximálních rychlostí v jednotlivých směrech je stejný jako poměr maximálních zrychlení v těchto směrech.

Maximální zrychlení je možno určit pomocí matice (7) a rovnice (4). Pokud do rovnice (4) dosadíme požadované zrychlení, můžeme vypočítat požadovaný tah motorů. Ten použijeme jako vstup funkce *BoundedControl.m*, která byla dodána vedoucím práce. Ta potom na základě parametrů motorů vrátí "ořezanou"hodnotu, které je možno dosáhnout.

Pro zjištění maximálního zrychlení při všech možných úhlech natočení byl nastaven požadovaný vstup na hodnotu převyšující možnosti robotu a úhly v jednotlivých osách se postupně měnily v rozmezí $< 0, 2\pi >$. Maximální zrychlení, v jednotlivých osách je

$$\mathbf{a}_{max} = \begin{pmatrix} 29,73 \\ 29,73 \\ 39,53 \end{pmatrix}. \tag{12}$$

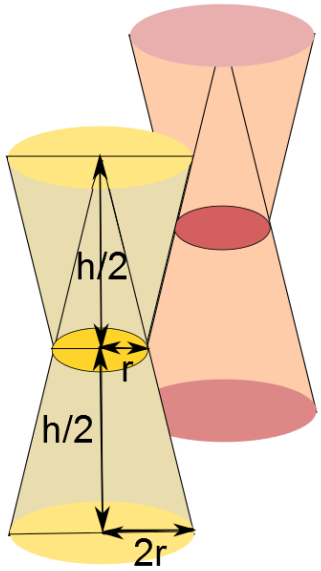
Po několika simulacích byl zvolen nejvyšší člen v-max roven $1,4$. Výsledné maximální rychlosti v jednotlivých směrech jsou

$$\mathbf{v}_{max} = \begin{pmatrix} 1,05 \\ 1,05 \\ 1,40 \end{pmatrix}. \tag{13}$$

3.1.3 Reálné rozměry částice

Dalším omezením, které je potřeba zavést, jsou reálné rozměry částic (robotů). Toto je první omezení, které není pro PSO algoritmus obvyklé. Zde se započítává nejen reálná velikost robotu, ale je nutno vzít v potaz i aerodynamické účinky jednoho robotu na druhý. Pokud by byly dva roboty nad sebou, vrchní by tahem svých motorů vytvářel vzdušné proudy, které by mohly spodní robot značně ovlivnit, a tento robot by pak spadl. Experimenty bylo zjištěno, že motory vytváří vzdušné proudy, které pod robotem tvoří těleso tvaru kužele. Proto je vhodné aproximovat kvadroptéru tělesem tvaru přesýpacích hodin s tím, že robot je umístěn v geometrickém středu těchto hodin. Průměr nejužšího místa tohoto tělesa je dán velikostí robotu, výška kuželu závisí na výkonu motorů. Na Obrázku 4 je možno vidět aproximaci dvou robotů, z nichž se jeden nachází výše než druhý, r je poloměr robotu, h je výška aproximačního tělesa.

Výše zmíněná aproximace zajistí, že se robot nedostane do kolize (ani nepřímé) s jiným robotem. Uvedený algoritmus řeší kolize následovně: Spočítá se nová poloha částice podle PSO. Jsou-li některé dvě částice v kolizi, algoritmus vypočítá novou pozici částice, která



Obrázek 4: Aproximace dvou robotů

se svým pohybem dostala do kolize. Nová pozice se nachází na plášti tělesa, kterým je aproximován nepohybující se robot. Algoritmus se potom změní přidáním podmínky pro nekolizní pozice viz Algoritmus 3.

3.2 Omezení daná relativní lokalizací robotů

Toto omezení přibližuje chování roje robotů k chování roje v přírodě. V přírodě není obvyklé, že by všichni členi roje komunikovali přímo s ostatními. To je zapříčiněno omezeným dosahem senzorických orgánů, jimiž jsou organismy schopny detekovat své okolí. V případě kvadrikoptér jde o kamerový systém. Ne vždy je možné zavést globální lokalizaci a u roje kvadrikoptér je počítáno s relativní lokalizací jednotlivých členů pomocí fůze dat z několika kamer, které jsou součástí robotů. Zde tedy mohou nastat problémy s velkou vzdáleností robotů, která ovlivní dosah vysílače robotu a hlavně na velkou vzdálenost není možné z obrazu kamery rozpoznat, zdali se jedná o robot či o něco jiného. Zde jsou tedy termíny lokalizace a komunikace ekvivalentní.

Result: Find global optima
initializatize $(\mathbf{v}, \mathbf{x})$ where $\mathbf{v^i}$ is i-th particles velocity $\mathbf{x^i}$ is i-th particles position.
Constants: $NumberOfGenerations$, $NumberOfParticles$, $\mathbf{v}_{max}$, $\mathbf{w}$, c_1, c_2 ;
for $k = 1; k \leq NumberOfGenerations$ **do**
  for $i = 1; i \leq NumberOfParticles$ **do**
    evaulate function value f_k^i using design space coordinates $\mathbf{x}_k^i$;
    if $f_k^i \leq p_{best}^i$ **then**
      $f_{best}^i = f_k^i, \mathbf{p}_k^i = \mathbf{x}_k^i$ `// update personal best;`
    if $f_k^i \leq p_{best}^g$ **then**
      $f_{best}^g = f_k^i, \mathbf{p}_k^g = \mathbf{x}_k^i$ `// update global best;`
  for $i = 1; i \leq NumberOfParticles$ **do**
    $\mathbf{r}_1 = rand(0, 1)$;
    $\mathbf{r}_2 = rand(0, 1)$;
    $\mathbf{v}_{k+1}^i = \mathbf{w}_k \mathbf{v}_k^i + c_1 r_1 (\mathbf{p}_k^i - \mathbf{x}_k^i) + c_2 r_2 (\mathbf{p}_k^g - \mathbf{x}_k^i)$ `// update velocity of i-th particle;`
    while *all restrictions are not satisfied* **do**
      if $\mathbf{v}_{k+1}^i > \mathbf{v}_{max+}$ **then**
        `// check velocity constriction;`
        $\mathbf{v}_{k+1}^i = \mathbf{v}_{max+}$
      else if $\mathbf{v}_{k+1}^i < \mathbf{v}_{max-}$ **then**
        $\mathbf{v}_{k+1}^i = \mathbf{v}_{max-}$
      $\mathbf{x}_{k+1}^i = \mathbf{x}_k^i + \mathbf{v}_{k+1}^i$ `// update position of i-th particle;`
      for $j = 1; j \leq numberOfParticles$ **do**
        if *detected collision between particles i and j,* $i \neq j$ **then**
          $\mathbf{x}_{k+1}^i = \mathbf{x}_{k+1}^j + (x_{off}, y_{off}, 0)'$ `// where` x_{off} `and` y_{off} `are coordinates of random point on surface of hourglass of j-th robot according to geometric center of j-th robot;`
          $\mathbf{v}_{k+1}^i = \mathbf{x}_{k+1}^i - \mathbf{x}_k^i$

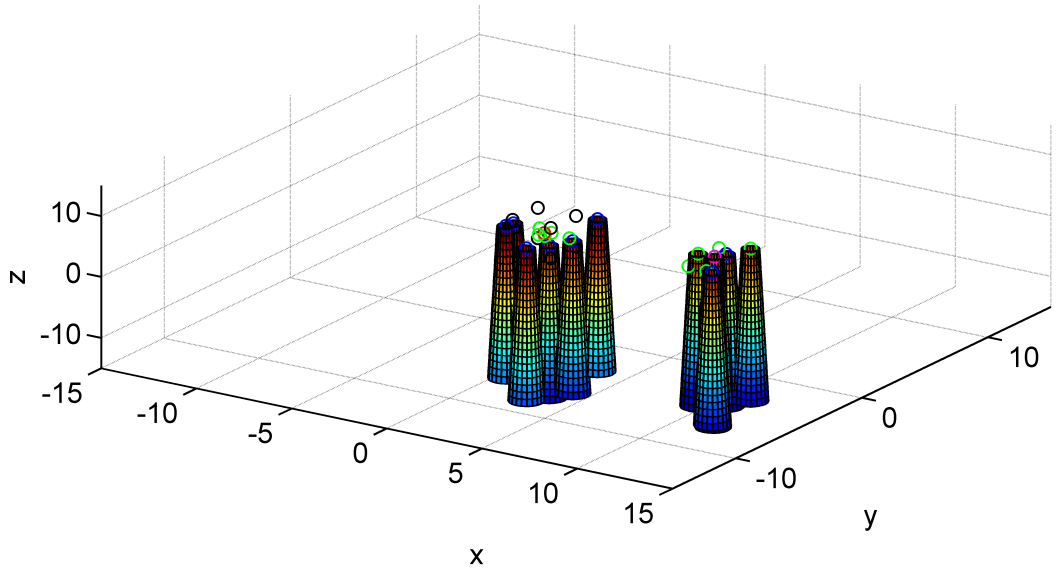
Algoritmus 3: PSO algoritmus s využitím inertia weight, v-max a řešením kolizí

Jako verze PSO algoritmu má toto omezení navíc vliv i pro hledání extrému funkce s mnoha lokálními extrémy. Globální informace směruje roj do lokálního extrému. Pokud částice komunikují jen s omezeným počtem dalších, tato částečná informace pak zajistí, že roj prohledává větší prostor, za cenu toho, že se výjimečně může rozpadnout na podskupiny.

3.2.1 Local-best (lbest)

Toto omezení využívá omezeného dosahu komunikace/lokalizace v prostoru. Každá částice komunikuje pouze s částicemi v dosahu. V rámci celé skupiny částic tak vznikají menší roje, které prohledávají různé části prostoru. Pokud se tyto malé roje přiblíží na komunikační/lokalizační dosah, spojí se do jednoho roje.

Algoritmus v podstatě zůstává stejný, navíc je nutno vyhodnotit, které částice jsou vzájemně v dosahu. Global-best se zamění za local-best. Zatímco global-best byl vždy pouze jeden, bodů local-best může být více. Jejich počet závisí na počtu částic a na lokalizačním dosahu a vzniká emergentě. V extrémním případě je local-best pro každou částici stejný jako její personal-best. Potom však PSO ztrácí význam. Dvě skupiny robotů, které mají dva local-best je možno vidět na Obrázku 5, popis simulačního prostředí viz kapitola 2.3. Global-best je vykreslen pouze pro informaci, v této variantě algoritmu ho částice nevyužívají a jeho pozice je shodná s jedním z bodů local-best.



Obrázek 5: Ukázka výstupu algoritmu s dosud uvedenými omezeními.

3.2.2 Udržení komunikace

Roje a hejna, která je možno nalézt v přírodě, udržují komunikaci s několika blízkými sousedy. Tato vazba je důležitá pro stabilizaci robotického roje. Toto je další z omezení daných reálným nasazením robotů, které se v PSO obvykle nevyskytuje a studie jeho vlivu na chování PSO se jeví jako perspektivní.

Jak už bylo uvedeno v předchozí podkapitole 3.2.1, může se stát, že se částice navzájem tolik vzdálí, že ztratí spojení s ostatními a tudíž personal-best a local-best některé z částic by potom byly totožné. Takto osamocená částice by potom uvízla v lokálním extrému a dál by nepomáhala roji s hledáním globálního extrému. Mohlo by se také stát, že každá částice bude kmitat kolem svého lokálního extrému a globální extrém nebude nikdy nalezen.

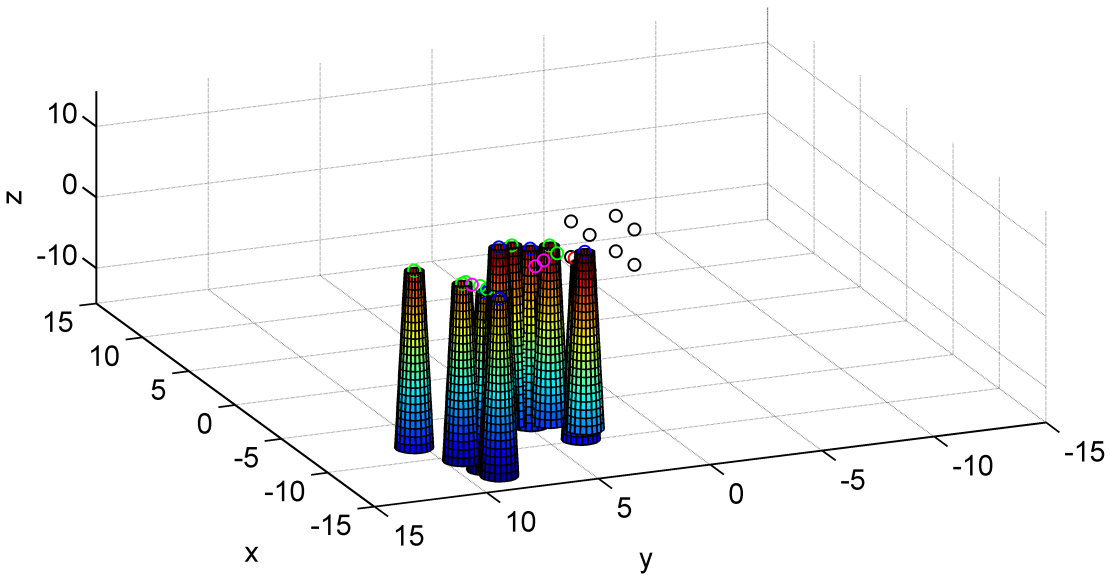
Vhodné je tedy zavést omezení, které by drželo částice ve skupinách o daném minimálním počtu jedinců. Řešení uvedené v Algoritmus 4 využívá dynamické omezení rychlosti. Pokud se aktualizací polohy jedné částice zmenší počet sousedů některé z částic pod danou mez, zmenší se rychlost pohybující se částice. Parametr pro zmenšování rychlosti byl zvolen 0.9. To zajistí, že rychlost neklesá zbytečně rychle a hledání extrému je tím tedy ovlivněno co nejméně.

Toto omezení má některé zajímavé důsledky. Pokud jsou roje rozděleny na skupiny a některý člen jednoho roje má "lepší"informaci z druhého roje, přitáhne celý svůj roj k tomuto druhému roji. Na Obrázku 6 je vidět počátek této situace.

Průběh třiceti generací finálního algoritmu je vidět na Obrázku 7. Na části 7a) je vidět rozmístění robotů v nulté generaci. Jsou zde dva shluky o šesti robotech a každý shluk má svůj local-best. Stávající pozice středů robotů jsou pro všechny jejich personal-best pozicí. Na části 7b) je vidět, že se roboty přiblížily první skupině na dosah a že už některý z druhé skupiny má local-best informaci od prvního skupiny. Na obrázku 7c) je vidět, že se skupiny ještě úplně nespojily, ale už více robotů z druhé skupiny komunikuje s roboty z první skupiny. Dále je zde vidět, že druhá skupina byla přivedena některými roboty k první skupině. Na obrázku 7d) se již obě skupiny spojily v jednu. Na části 7e) je vidět výsledná

Result: Find global optima
initializatize $(\mathbf{v}, \mathbf{x})$ where $\mathbf{v^i}$ is i-th particles velocity $\mathbf{x^i}$ is i-th particles position.
Constants: $NumberOfGenerations$, $NumberOfParticles$, $\mathbf{v}_{max}$, $\mathbf{w}$, c_1, c_2 and $RequiredNeighbors$;
for $k = 1; k \leq NumberOfGenerations$ **do**
  for $i = 1; i \leq NumberOfParticles$ **do**
    evaulate function value f_k^i using design space coordinates $\mathbf{x}_k^i$;
    if $f_k^i \leq p_{best}^i$ **then**
      $f_{best}^i = f_k^i, \mathbf{p}_k^i = \mathbf{x}_k^i$ `// update personal best;`
    if $f_k^i \leq p_{best}^l$ **then**
      $f_{best}^l = f_k^i, \mathbf{p}_k^l = \mathbf{x}_k^i$ `// update local best;`
  for $i = 1; i \leq NumberOfParticles$ **do**
    $\mathbf{r}_1 = rand(0, 1)$;
    $\mathbf{r}_2 = rand(0, 1)$;
    $\mathbf{v}_{k+1}^i = \mathbf{w}_k \mathbf{v}_k^i + c_1 r_1 (\mathbf{p}_k^i - \mathbf{x}_k^i) + c_2 r_2 (\mathbf{p}_k^l - \mathbf{x}_k^i)$ `// update velocity of i-th particle;`
    restore $\mathbf{v}_{max}$ to its original value;
    while *all restrictions are not satisfied* **do**
      if $\mathbf{v}_{k+1}^i > \mathbf{v}_{max}$ **then**
        `// check velocity constriction;`
        $\mathbf{v}_{k+1}^i = \mathbf{v}_{max}$
      else if $\mathbf{v}_{k+1}^i < -\mathbf{v}_{max}$ **then**
        $\mathbf{v}_{k+1}^i = -\mathbf{v}_{max}$
      $\mathbf{x}_{k+1}^i = \mathbf{x}_k^i + \mathbf{v}_{k+1}^i$ `// update position of i-th particle;`
      for $j = 1; j \leq numberOfParticles$ **do**
        if *detected collision between particles i and j, $i \neq j$* **then**
          $\mathbf{x}_{k+1}^i = \mathbf{x}_{k+1}^j + (x_{off}, y_{off}, 0)'$ `// where` x_{off} `and` y_{off} `are coordinates of random point on surface of hourglass of j-th robot according to geometric center of j-th robot;`
          $\mathbf{v}_{k+1}^i = \mathbf{x}_{k+1}^i - \mathbf{x}_k^i$
      count all neighbors ($numberOfNeighbors$) of i-th particle using $\mathbf{x}_k$ for other particles , $\mathbf{x}_{k+1}^i$ for i-th particle and range constant ;
      if $numberOfNeighbors < RequiredNeighbors$ **then**
        $\mathbf{v}_{max} = 0.9\mathbf{v}_{max}$

Algoritmus 4: PSO algoritmus s využitím inertia weight, v-max, řešením kolizí, local-best a udržením se ve skupině



Obrázek 6: Větší množství local-best bodů

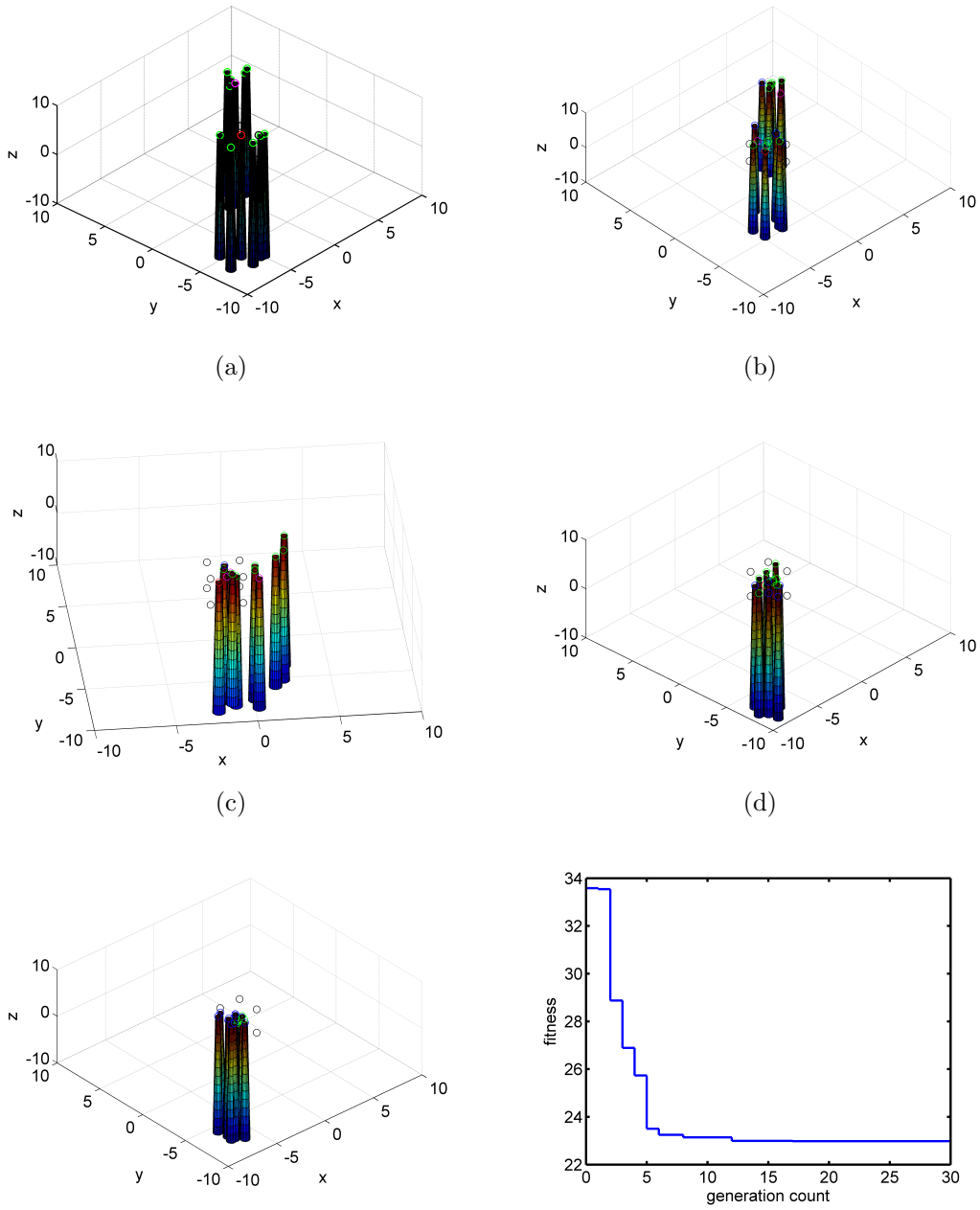
pozice roje. Na obrázku 7f) je potom uveden průběh fitness funkce.

3.3 Ladění parametrů

Poslední uvedená verze algoritmu uvedená v předchozí podkapitole obsahuje několik laditelných parametrů. Jedná se o parametry

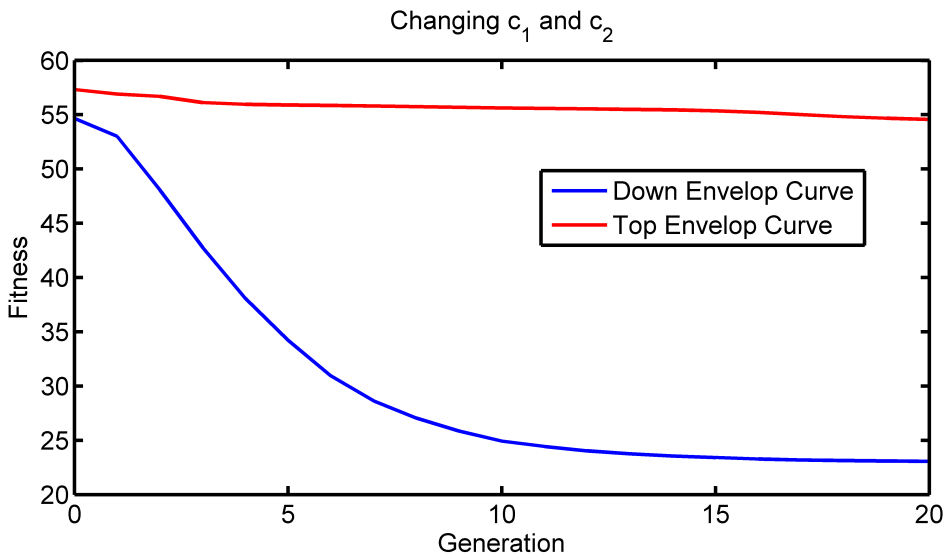
- c_1
- c_2
- $v - max$
- *inertia weight*,

jejichž vliv na chování roje bude prezentován v této sekci.



(a) (b) (c) (d) (e) (f)

Obrázek 7: Ukázka průběhu celého algoritmu

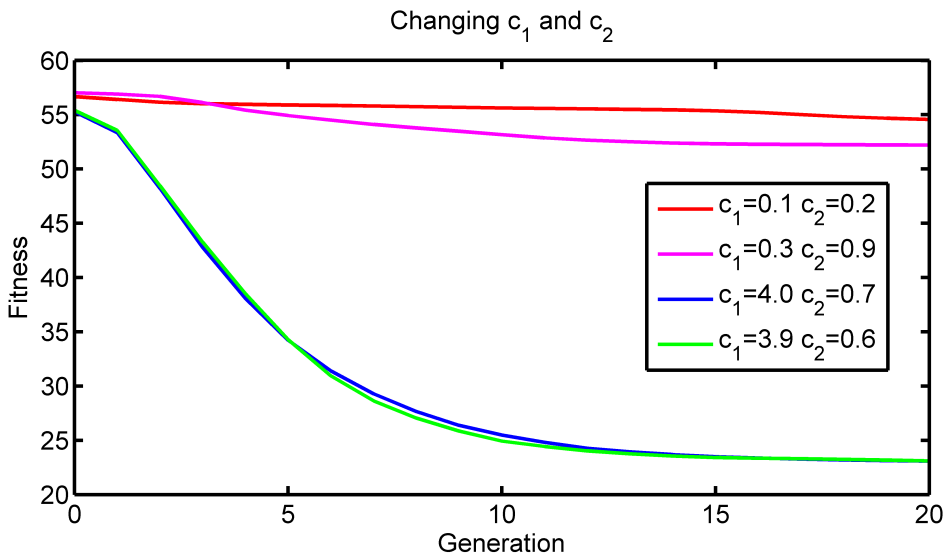


Obrázek 8: Nejhorší a nejlepší výsledky pro změnu parametrů c_1 a c_2

3.3.1 Ladění parametrů inteligence

Jedná se o první dva uvedené laditelné parametry. Podle literatury [5] je vhodné tyto parametry volit menší než 4. Pokud by byly vyšší a nebyla by zavedena další omezení, mohlo by se stát, že by částice kmitaly ve zvětšující se vzdálenosti kolem hledaného extrému. Ladění probíhalo na finálním algoritmu uvedeném v podkapitole 3.2.2. Parametry c_1 a c_2 se postupně měnily v rozmezí $< 0.1; 40 >$ s krokem o 0.1. Pro každou kombinaci parametrů c_1 a c_2 bylo provedeno 100 pokusů. V každé generaci byla zaznamenána hodnota fitness funkce a pro 100 pokusů byla zprůměrována. Test probíhal po 20 generací s využitím dvanácti robotů, které se měly pohybovat ve skupinách o nejméně šesti členech. Na Obrázku 8 jsou vidět obalové křivky všech simulací - zobrazit všech 1600 simulací by nemělo smysl. Na Obrázku 9 jsou zobrazena data pro dvě nejlepší a dvě nejhorší kombinace parametrů c_1 a c_2. Data jednotlivých fitness funkcí pro kombinace parametrů je možno nalézt na CD v souboru *matlab/data/test_c1_c2.mat*.

Hodnoty byly určeny: $c_1 = 3,9$ a $c_2 = 0.6$.

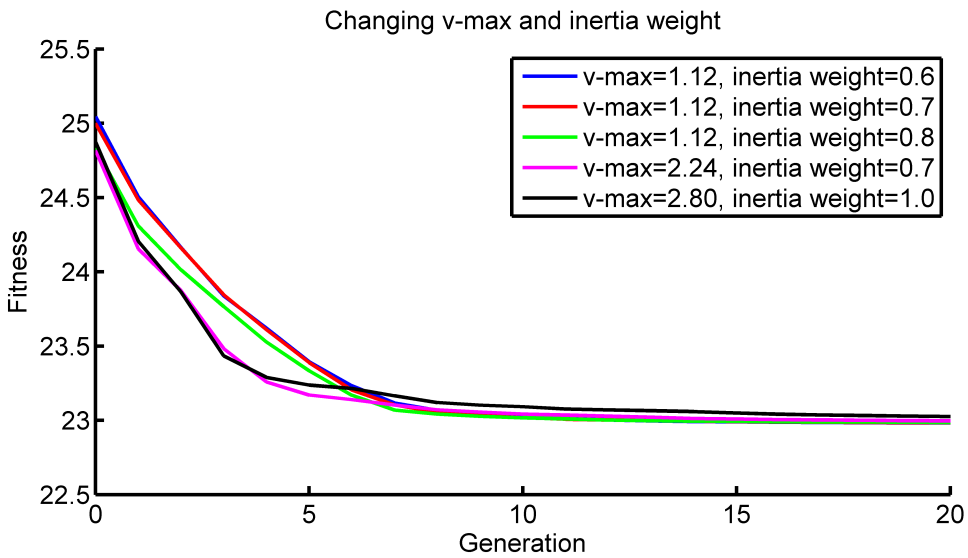


Obrázek 9: Dvě nejlepší a nejhorší fitness pro dané parametry c_1 a c_2

3.3.2 Násobící konstanta pro v-max a inertia-weight

Na Obrázku 10 jsou vidět průběhy fitness pro měnící se parametry $v{-}max$ a *inertia weight*. Pro každou kombinaci parametrů $v - max$ a *inertia weight* bylo provedeno 100 pokusů. V každé generaci byla zaznamenána hodnota fitness funkce a pro 100 pokusů byla zprůměrována. Test probíhal po 20 generací s využitím dvanácti robotů, které se měly pohybovat ve skupinách o nejméně šesti členech. Nastavení poměrů jednotlivých složek parametrických vektorů uvedené v kapitolách 3.1.1 a 3.1.2 bylo zachováno. V legendě obrázku je uvedena pouze hodnota největší složky tohoto vektoru. Data jednotlivých fitness funkcí je možno nalézt na CD v souboru *matlab/data/test_vmax_inertia.mat*.

Dále je na Obrázku 10 vidět, že pro vyšší hodnotu $v - max$ roj rychleji konvergoval k hledané hodnotě v několika prvních generacích. Hledanou hodnotu roj ale lépe nalezl pro menší parametr $v - max$. Hodnoty těchto parametrů, které byly zvoleny v předchozích kapitolách, tedy odpovídaly nejvíce situaci, kdy požadujeme rychlejší konvergenci roje, ale není nutné najít přesně hledanou hodnotu. Vzhledem k reálné velikosti robotů a delší době, než proběhne generace, je vhodné použít toto nastavení, konkrétně: $v - max =$



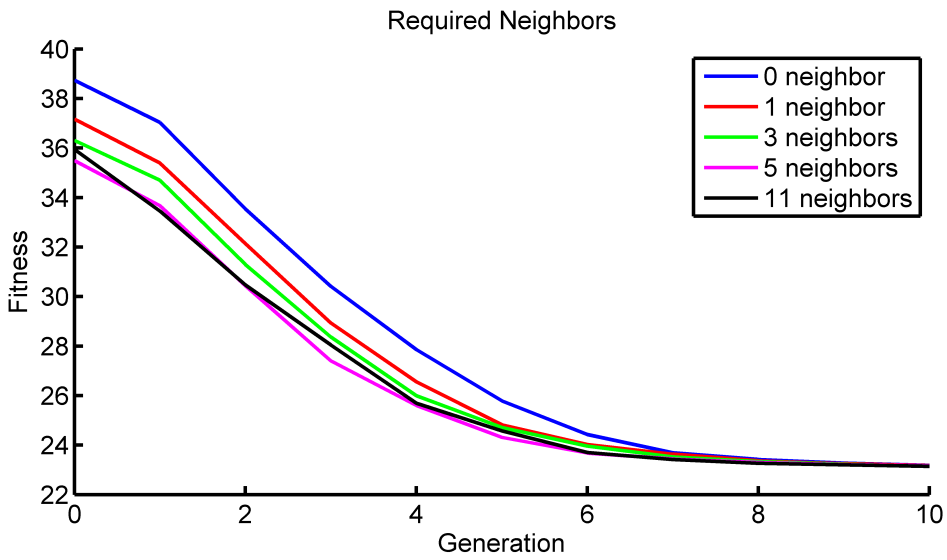
Obrázek 10: Čtyři fitness pro dané parametry $v - max$ a $inertia\ weight$

$(1,68; 1,68; 2,24)'$ a $inertia\ weight = (0,7; 0,7; 0,70)'$

3.3.3 Srovnání

V algoritmu se mimo přímo laditelných konstant objevují i další konstanty, jejichž volba může značně ovlivnit konvergenci roje. Především jde o konstanty, které určují dosah robotů a počet sousedů, se kterými má být robot v dosahu. Simulace byla opět spuštěna 100x a zprůměrována, počet robotů v simulaci byl 12. Simulace byla spuštěna po 40 generací.

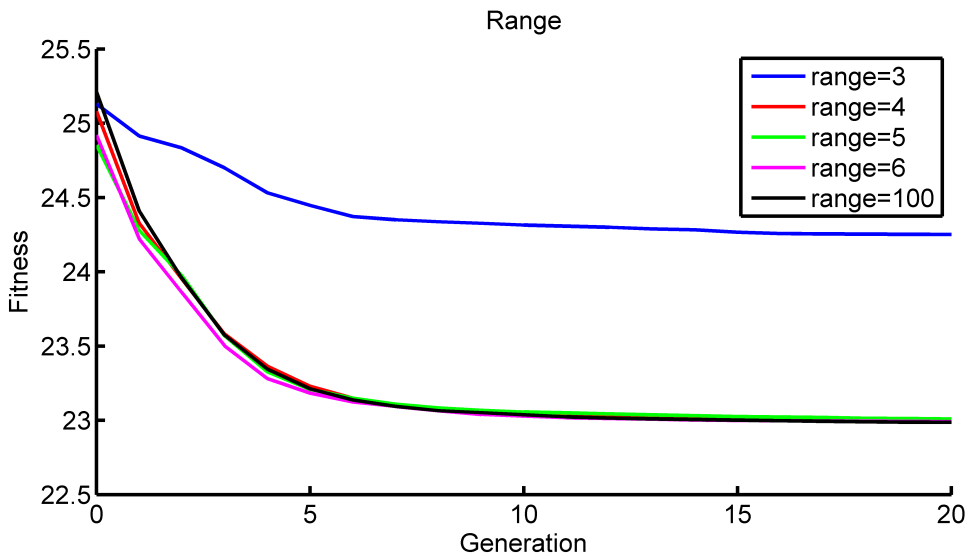
Graf pro zvyšující se počet sousedů je uveden na Obrázku 11. Graf popisuje, kolik sousedů měl mít jeden robot v dosahu, jednalo se tedy o skupiny o počtu 1, 2, 4, 6 a 12 členů. Posunutí v počáteční hodnotě fitness pro jednotlivé počty sousedů je dáno rozmísťovací funkcí v prostoru, která inicializuje roboty v prostoru po skupinách o předepsaném počtu jedinců. Tyto skupiny jsou od sebe vzdáleny tak, aby nebyly navzájem v dosahu. Pro měnící se počet sousedů je vidět, že lépe fungovaly podskupiny o větším počtu jedinců. Například pro jednoho a pět sousedů začíná fitness funkce na obdobné hodnotě, pro pět sousedů ale rychleji konverguje k hledané hodnotě.



Obrázek 11: Zvyšující se počet sousedů

Na Obrázku 12 pro zvyšující se dosah robotů s požadovanými šesti sousedy. Je vidět, že pokud byl dosah příliš malý, roj konvergoval velmi pomalu, to je dáno tím, že se roboty od sebe nesměly vzdálit. Pro komunikační dosah roven 6 roj dokonce lépe konvergoval než pro "neomezený" dosah, zde reprezentovaný hodnotou 100. To je částečně dáno tím, že v nulté generaci měl roj pro nejvyšší dosah průměrnou fitness vyšší než roje s menším dosahem.

Na Obrázku 13 pro měnící se velikost robotů je vidět velmi zajímavý jev. Předpoklad byl, že PSO bez omezení na velikost robotů bude rychleji konvergovat k hledané hodnotě. Zde je ale vidět, že nejlépe roj konvergoval pro roboty o poloměru $0,4m$. To může být dáno hodnotou parametrů c_1 a c_2, které byly laděny právě pro tento poloměr, a dále může být lepší konvergence podmíněna způsobem řešení kolizí. Přestože větší poloměr dosahoval lepší konvergence, časová náročnost algoritmu značně rostla. Pro nulový poloměr robotů trvalo 100 opakování experimentu 32s, pro poloměr $0,4$ $160s$ a pro největší poloměr celých $1296s$.



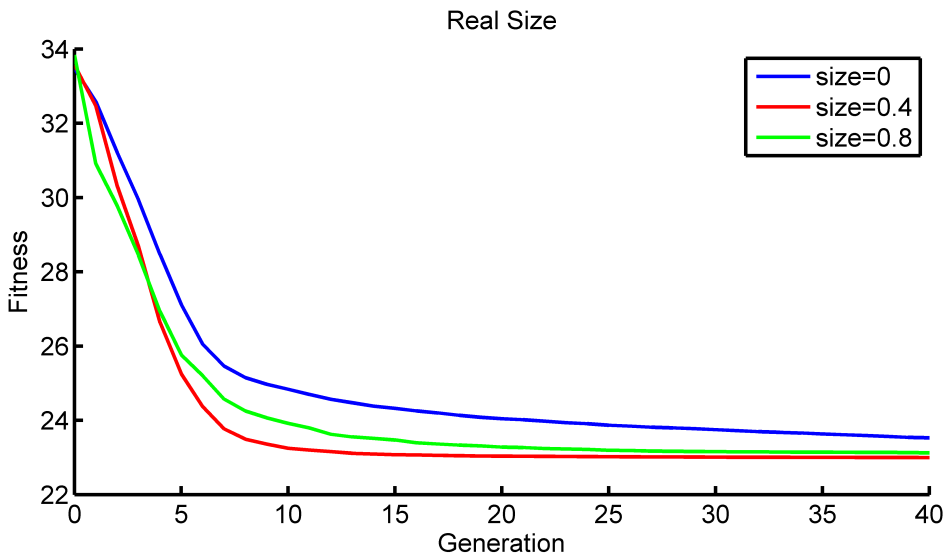
Obrázek 12: Zvyšující se dosah

3.4 Zajímavosti chování roje

Ve verzi algoritmu, která umožňovala, aby částice opouštěly roj, se takto odloučená částice v následující generaci většinou vrátila zpět. To mohlo být způsobeno tím, že její personal-best byl v blízkosti menšího roje. Dále se zde projevovalo vytěsňování robotů ze středu roje na jeho okraj. Protože se nové pozice robotů počítají postupně, byl mnohdy robot ze středu vytlačen ven, aby byla zaručena nekolizní pozice robotů.

Ve chvíli, kdy se jedna podskupina přiblíží na dosah jiné podskupině, se může objevit několik local-best pozic. To je dáno tím, že každá částice z právě přiblíženého roje má v dosahu jiné částice z druhého roje, které mohou mít lepší personal-best.

Pro skupiny robotů s využitím všech výše zmíněných omezení docházelo k situaci, že se menší skupiny robotů k sobě přiblížily a pouze některý z robotů z druhého roje věděl o lepší informaci, kterou měl první roj. Tento robot pak dovedl svoji skupinu k prvnímu roji. Toto mohlo vzniknout díky podmínce, že všechny roboty musí mít daný počet sousedů.



Obrázek 13: Závislost fitness na velikosti robotů

4 Robotické simulace

4.1 Hledání nejvyššího signálu vysílačů

V prostoru je několik vysílačů, úkolem robotů je najít místo, kde je jejich signál nejvyšší.

4.1.1 Motivace

Roj robotů poletující v okolí reálných vysílačů může vizualizovat dosah signálu svým pohybem. Zajímavou aplikací by mohlo být čištění kontaminovaných oblastí - nejvíce zamořené "kousky" by mohly roboty selektivně sbírat podle změřené kontaminace.

4.1.2 Realizace

Tato úloha byla základní úlohou, na které bylo testováno chování reálného roje v rámci této práce. Úpravy a přizpůsobení byly popsány v kapitole 3. Na Obrázku 7 v kapitole 3.2.2 je vidět průběh hledání místa s nejvyšším signálem. Simulační prostředí bylo popsáno v kapitole 2.3.

4.2 Rozsvícení místnosti

Úkolem roje je najít v místnosti osvětlené několika zdroji světla nejtmavší bod. Tam jeden člen roje zůstane a začne svítit - vytvoří tak další zdroj světla. Roboty takto rozsvítí celou místnost.

4.2.1 Motivace

V reálu se může jednat třeba o pokrytí daného území přijímači radiového signálu. Ve velmi složitých a materiálně nehomogenních budovách by mohl být roj použit pro nalezení míst, kam je vhodné umístit další wifi access-point pro zlepšení pokrytí signálem.

4.2.2 Realizace

Do původního algoritmu nebylo potřeba radikálně zasahovat. Byla vytvořena úloha, ve které se snižuje počet jedinců a zvyšuje se počet zdrojů v měřené funkci. Dále bylo pomocí podmínky zajištěno, že roboty neopouští předepsanou oblast - místnost. Pro jednoduchost se jedná o válcovou místnost o rozměrech $polomer = 10$, $vyska = 10$. Fitness funkce se počítá obdobně jako v kapitole 2.3. Přičemž hodnota této fitness funkce

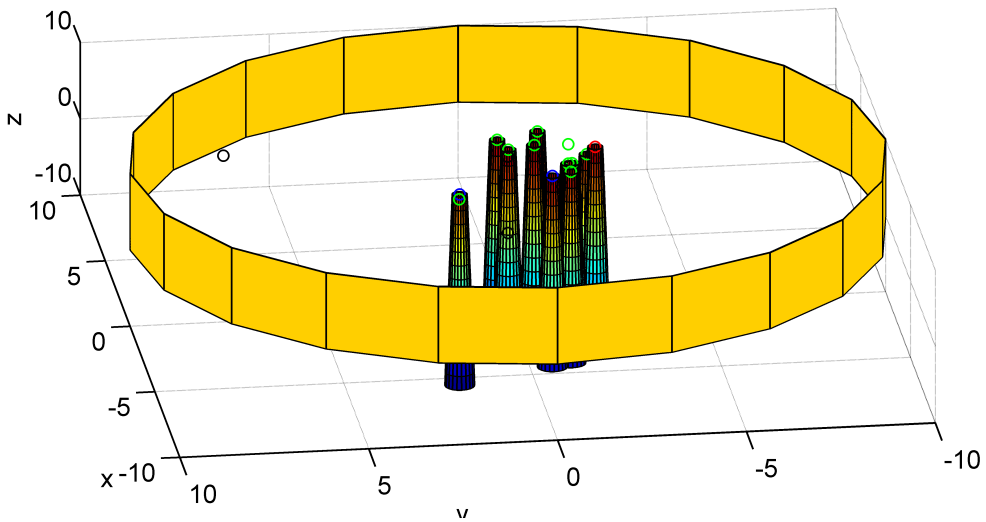
$$f^i = \sum_{k=1}^{m} \sqrt{\sum_{d=1}^{3} (x_d^i - z_d^k)^2}, \tag{14}$$

je maximalizována.

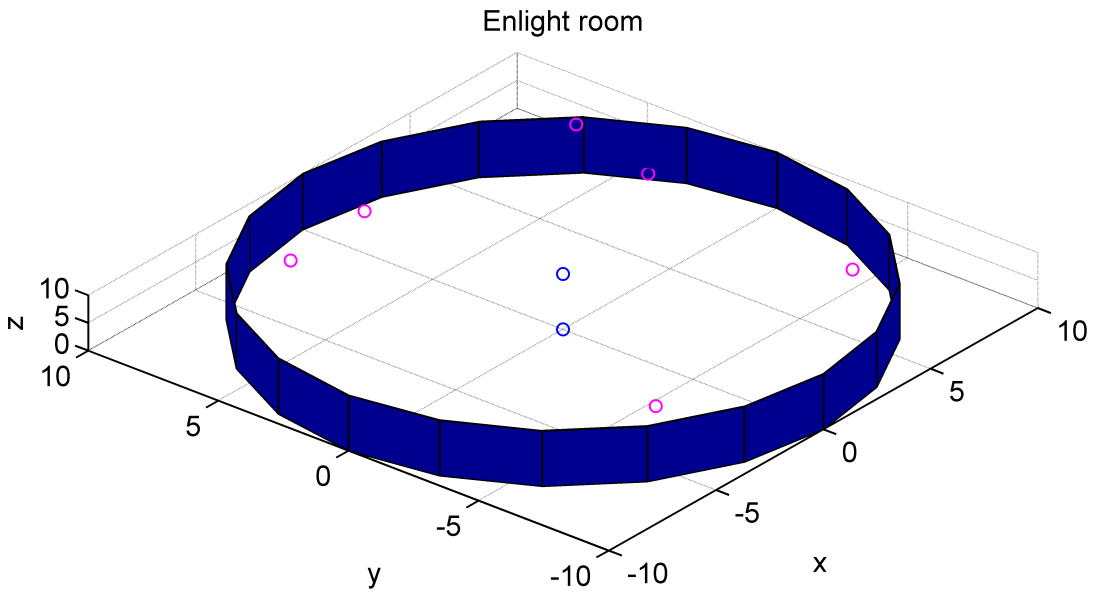
Na Obrázku 14 je vidět ukázka roje hledajícího místo s minimálním osvětlením. V místnosti jsou od počátku umístěny dva zdroje světla na souřadnicích $\{0; 0; 0\}$ a $\{0; 0; 10\}$. V simulaci jsou všechny zdroje světla zobrazeny pomocí černých kroužků. Je vidět že mimo zmíněných původních zdrojů, je v místnosti přítomen další zdroj (svítící robot) a to blízko stěny místnosti. Ta je symbolizována žlutým válcem. Podlaha ani strop nejsou zobrazeny pro zlepšení viditelnosti roje.

Na Obrázku 15 je vidět výsledek pokusu, který byl spuštěn pro 12 robotů, kde každý robot měl udržovat komunikaci minimálně s pěti dalšími. Algoritmus byl postupně spuštěn 6x, aby byla výše zmíněná podmínka pro komunikační dosah splnitelná. Vždy po deseti generacích byl algoritmus zastaven a od roje se odpojil jeden robot, který se stal novým zdrojem světla. Následně byl roj inicializován kolem středu místnosti, aby hledal nový extrém. V místnosti jsou originální zdroje světla symbolizované modrými kroužky a purpurovými kroužky jsou zobrazeny pozice zdrojů světla vzniklé z robotů.

Roj robotů se choval podle očekávání. Pokud v jednom běhu algoritmu prohledával jednu stranu místnosti, v dalším běhu byla prohledávána druhá strana.



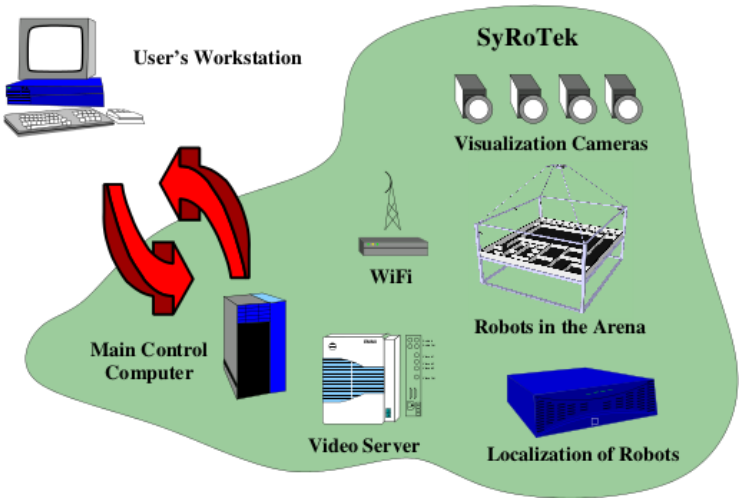
Obrázek 14: Snímek ze simulačního prostředí pro rozsvícení místnosti



Obrázek 15: Nalezená místa a osvětlení

5 SyRoTek

SyRoTek (Systém pro robotickou tele-výuku) umožňuje vzdáleně (přes internet) ovládat multi-robotickou platformu s dynamickými překážkami. Se SyRoTkem je možno vyvíjet algoritmus a rovnou ho testovat na reálných robotech. Schéma systému je ukázáno na Obrázku (16).



Obrázek 16: Přehled systému SyRoTek [3]

Systém se skládá z arény, kde se mohou roboty pohybovat, dále pak ze samotných robotů, které jsou vybaveny různými senzory. Lokalizaci robotů zajišťuje kamera spolu se speciálním softwarem. Spojení robotů se serverem je realizováno pomocí WiFi.

Aréna má dvě části, kde se pohybují roboty - dokovací stanice, kde se roboty dobíjí, a plochu pro experimenty s roboty. Každý robot má na vrchní části umístěn identifikační obrazec, podle kterého systém identifikuje jednotlivé roboty, určuje jejich polohu a natočení. Systém má pro každý robot interní číslo (1-12).

5.1 Player/Stage

Systém SyRoTek je postaven na softwarovém rozhraní Player-Stage, které umožňuje stejný přístup jak k reálným robotům, tak k simulátoru Stage. Dále je pak možno spouštět program pro roboty na vlastním počítači či serveru SyRoTek.

Player poskytuje síťové rozhraní pro různé roboty a jejich senzory. Architektura klient/server pak umožňuje napsat ovládací program v jakémkoliv programovacím jazyce a spustit ho z počítače připojeného k internetu. [8]

Stage simuluje skupinu robotů, kteří se pohybují a "vnímají" v dvojdimenzionálním prostoru. [8]

5.2 Přístup k robotům pro studenty

Studenti mohou k robotům přistupovat přes internet. Nejdříve je potřeba vytvořit rezervaci pro danou robotickou úlohu na internetových stránkách [3]. Systém potom studentům připraví na daný čas požadovaný počet robotů s danými senzory a rozestaví je na výchozí pozice v aréně. Dále student získá na časový slot (rezervaci) práva dávat robotům příkazy. Arénu je možno zarezervovat nejdéle na čtyři hodiny. Aby byly roboty nabité pro další rezervaci, je vždy po rezervaci vyhrazen čekací slot stejně dlouhý, jako byla samotná rezervace. Během tohoto čekání není možno jezdit s roboty.

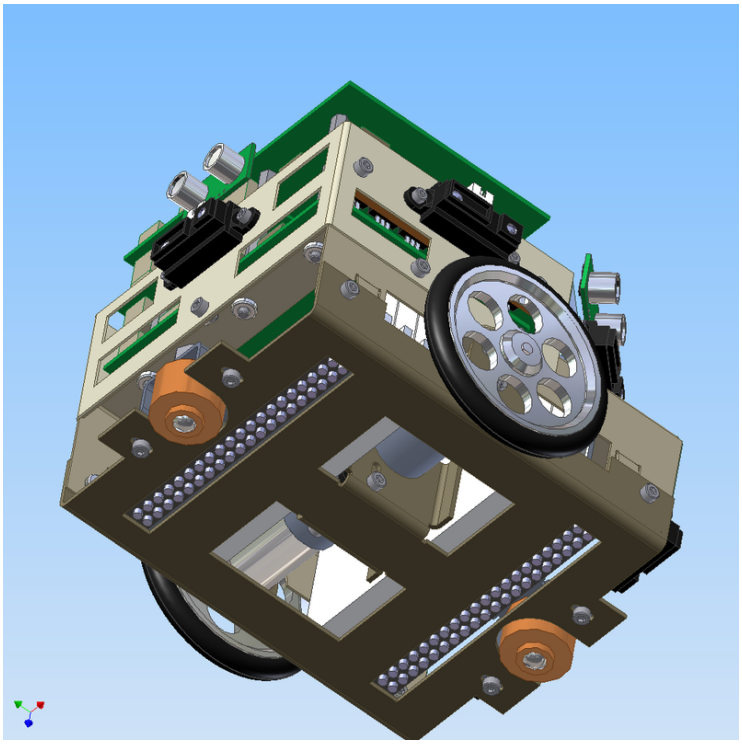
V rezervované úloze jsou roboty reprezentovány čísly ("one", "two", "three"...) bez ohledu na to, jaké je interní číslo reálného robotu v systému. Není tedy možné určit, jaký reálný robot (1-12) bude "one" apod.

5.3 Podlahový senzor

Roboty mají nainstalované různé senzory, přičemž pro tuto práci je důležitý *floor:ir* senzor, který je složen z lišty a vyhodnocovacích obvodů. Na liště je 6 dvojic smd LED

diod, z nichž jedna svítí a druhá snímá intenzitu odraženého světla. Dále se na okrajích lišty nachází dvě osvětlovací diody. Umístění dvou senzorů je vidět na Obrázku 17.

Na robotu se nacházejí dvě tyto lišty, interně pojmenované *floor front* a *floor rear*. Senzor načítá data s periodou 500ms, přičemž tu je možno zmenšit, ale potom může robot přestat fungovat. Se senzorem se zatím na SyRoTku nepracovalo a není tedy kalibrovaný a občas nefunguje správně. Otestování senzoru bylo součástí této bakalářské práce.



Obrázek 17: Floor Senzor [4]

5.3.1 Testování senzoru na robotech

Po prvních pokusech bylo zjištěno, že spuštění senzoru u některých robotů způsobuje resetování celého robotu. Tvůrci systému tento jev přikládají kolizi na sběrnici. Některý z mikroprocesorů robotu je připojen na stejné sběrnici, jako je i senzorová lišta. Příchozí data ze senzoru nejsou ze sběrnice vyčítána dostatečně rychle a tak se sběrnice zahltí.

Mikroprocesor detekuje tuto kolizi a restartuje se. Restart tohoto jediného mikroprocesoru však zaviní restart všech součástí robota. Robot potom chvíli nereaguje na jakékoliv příkazy, a pokud se znovu zapne, nepřijímá příkazy od normálního uživatele (studenta). To je způsobeno nastavením práv pro přístup k robotům, která student restartem robotu ztratí.

Z pokusů však vychází, že některé roboty, přestože by měly být navrženy ekvivalentně, jsou odolnější vůči kolizím a během celého časového slotu fungovaly bezchybně. V tabulce 1. je uvedeno, na kterých robotech byl senzor testován a jak pokus dopadl.

číslo robotu	1	2	3	4	5	6	7	8	9	10	11	12
senzor floor:front	X	F	OK	S	X	H	OK	OK	X	H	H	F

Tabulka 1: Testování floor senzoru na robotech

Legenda k tabulce 1:

X netestováno

OK senzor fungoval bez problémů

F robot se restartoval

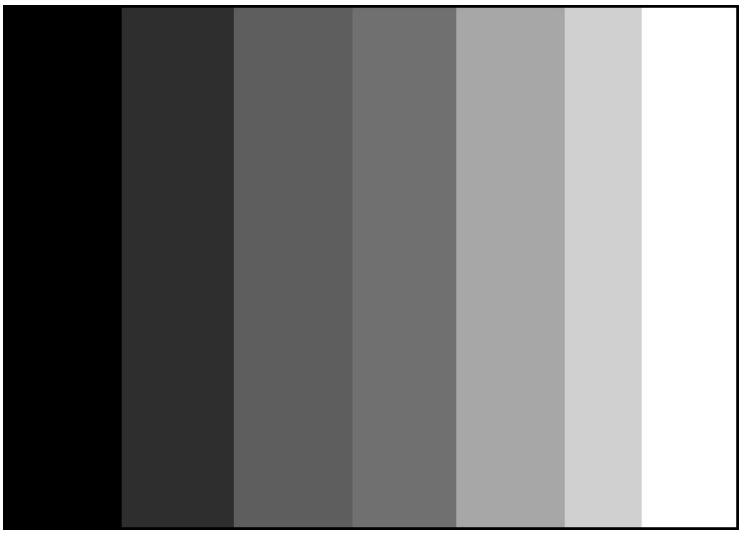
H k restartu robota docházelo výjimečně

S senzor nereagoval

5.3.2 Měření intenzity odraženého světla

Pro potřebu měření funkce navrženého algoritmu byl testován *floor:front* senzor. Pro kalibraci a testování podlahových senzorů byl využit testovací papír rozměru A4 s vytištěnou kalibrační stupnicí viz Obrázek 18. Jednotlivé barvy testovacího obrázku odpovídají html notaci (po řadě od černé k bílé) 000000, 2e2e2e, 5e5e5e, 707070, a7a7a7, d0d0d0, ffffff. Takové hodnoty byly zvoleny, aby na papíře vznikly pruhy 3-4cm široké a jednotlivé stupně šedi byly od sebe rovnoměrně vzdáleny. První pokusy ukázaly, že se hodnota, kterou senzor

v programu vrací, mění v rozmezí $(0, 5)$ jednotek. Čím je povrch tmavší, tím větší hodnotu senzor vrací. Ukázalo se také, že některé ze senzorů na liště jsou v saturaci. Buď ukazovaly hodnotu kolem $0,15$ nebo $4,995$ nezávisle na tom, jestli byl snímaný povrch černý nebo bílý.



Obrázek 18: Testovací obrázek pro floor senzor

Pro měření charakteristiky byly vybrány roboty 3 a 8. Robot byl umístěn na testovací vzor tak, aby byl senzor přímo nad černým pruhem. Systémovým příkazem *syr_control.rb* bylo pak robotem posouváno o 1cm směrem k bílému pruhu, až se senzor ocitl nad povrchem arény. Naměřené hodnoty jsou uvedeny v Tabulce 2 pro robot 3. a v Tabulce 3 pro robot 8.

Data byla zobrazena do grafu a z nich bylo určeno, o jakou barvu se jedná. Musely se vyloučit případy, kdy robot naměřil přechod mezi dvěma různými testovacími pruhy. Hodnoty pro jednotlivé barvy jsou vykresleny v Tabulce 4 pro robot č. 3. a v Tabulce 5 pro robot č. 8. Na Obrázku 19 a Obrázku 20 jsou vykresleny grafy pro postupně se měnící stupně šedi. U robotu 8. je vidět, že pro dva senzory se křivky lišily pouze o nějaký offset. Dále je zde vidět, že některé ze senzorů na liště byly v saturaci.

	senzor					
poloha	0	1	2	3	4	5
0,44	0,25	0,27	1,13	0,47	0,13	0,13
0,43	0,24	0,27	1,10	0,46	0,14	0,14
0,42	0,24	0,27	1,10	0,46	0,14	0,14
0,41	0,25	0,26	1,14	0,48	0,14	0,14
0,40	0,25	0,26	1,14	0,48	0,14	0,14
0,39	0,24	0,27	1,08	0,45	0,13	0,13
0,38	0,24	0,27	1,03	0,40	0,13	0,13
0,37	0,24	0,27	1,03	0,40	0,13	0,13
0,36	0,23	0,26	0,91	0,35	0,14	0,14
0,35	0,23	0,25	0,73	0,31	0,13	0,14
0,34	0,23	0,25	0,73	0,31	0,13	0,14
0,33	0,24	0,26	0,76	0,31	0,13	0,14
0,32	0,24	0,27	0,62	0,29	0,13	0,14
0,31	0,24	0,27	0,62	0,29	0,13	0,14
0,30	0,25	0,25	0,45	0,28	0,13	0,13
0,29	0,25	0,25	0,45	0,28	0,13	0,13
0,28	0,22	0,25	0,33	0,28	0,13	0,13
0,27	0,22	0,25	0,33	0,28	0,13	0,13
0,26	0,22	0,25	0,33	0,28	0,13	0,13
0,25	0,22	0,24	0,26	0,25	0,13	0,13
0,24	0,22	0,24	0,26	0,25	0,13	0,13
0,23	0,22	0,23	0,25	0,23	0,13	0,13
0,22	0,22	0,23	0,25	0,23	0,13	0,13
0,21	0,22	0,23	0,25	0,23	0,13	0,13
0,20	0,22	0,23	0,25	0,23	0,13	0,13

Tabulka 2: floor:front senzor na robotu 3

	senzor					
poloha	0	1	2	3	4	5
0,44	2,64	2,51	2,37	2,43	0,22	0,18
0,43	2,64	2,51	2,37	2,43	0,22	0,18
0,42	2,66	2,53	2,36	2,44	0,24	0,20
0,41	2,66	2,53	2,36	2,44	0,24	0,20
0,40	2,66	2,53	2,36	2,44	0,24	0,20
0,39	2,66	2,53	2,36	2,44	0,24	0,20
0,38	2,65	2,50	2,34	2,41	0,22	0,19
0,37	2,65	2,50	2,34	2,41	0,22	0,19
0,36	2,57	2,42	2,26	2,34	0,21	0,19
0,35	2,57	2,42	2,26	2,34	0,21	0,19
0,34	2,57	2,42	2,26	2,34	0,21	0,19
0,33	2,57	2,42	2,26	2,34	0,21	0,19
0,32	2,55	2,40	2,22	2,20	0,21	0,19
0,31	2,55	2,40	2,22	2,20	0,21	0,19
0,30	2,50	2,34	2,15	2,21	0,19	0,16
0,29	2,50	2,34	2,15	2,21	0,19	0,16
0,28	2,43	2,21	2,02	2,05	0,19	0,17
0,27	2,26	2,00	1,77	1,79	0,18	0,17
0,26	2,26	2,00	1,77	1,79	0,18	0,17
0,25	2,24	1,95	1,73	1,78	0,17	0,17
0,24	1,95	1,62	1,35	1,32	0,16	0,16
0,23	1,95	1,62	1,35	1,32	0,16	0,16
0,22	1,50	1,20	1,02	1,05	0,16	0,13
0,21	1,50	1,20	1,02	1,05	0,16	0,13
0,20	1,20	0,83	0,64	0,65	0,15	0,14

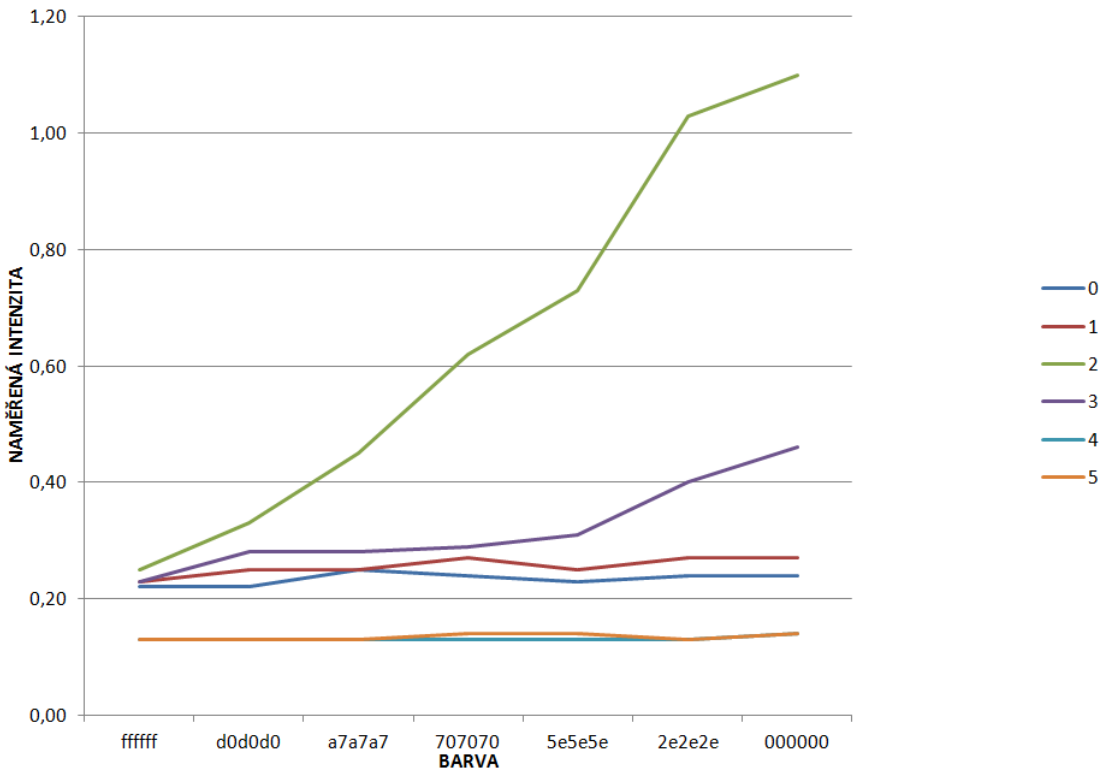
Tabulka 3: floor:front senzor na robotu 8

	senzor					
barva	0	1	2	3	4	5
ffffff	0,22	0,23	0,25	0,23	0,13	0,13
d0d0d0	0,22	0,25	0,33	0,28	0,13	0,13
a7a7a7	0,25	0,25	0,45	0,28	0,13	0,13
707070	0,24	0,27	0,62	0,29	0,13	0,14
5e5e5e	0,23	0,25	0,73	0,31	0,13	0,14
2e2e2e	0,24	0,27	1,03	0,40	0,13	0,13
000000	0,24	0,27	1,10	0,46	0,14	0,14

Tabulka 4: Vztah mezi barvou a měřenou intenzitou pro robot 3.

	senzor					
barva	0	1	2	3	4	5
ffffff	1,20	0,83	0,64	0,65	0,15	0,14
d0d0d0	1,50	1,20	1,02	1,05	0,16	0,13
a7a7a7	1,95	1,62	1,35	1,32	0,16	0,16
707070	2,26	2,00	1,77	1,79	0,18	0,17
5e5e5e	2,50	2,34	2,15	2,21	0,19	0,16
2e2e2e	2,57	2,42	2,26	2,34	0,21	0,19
000000	2,66	2,53	2,36	2,44	0,24	0,20

Tabulka 5: Vztah mezi barvou a měřenou intenzitou pro robot 8.



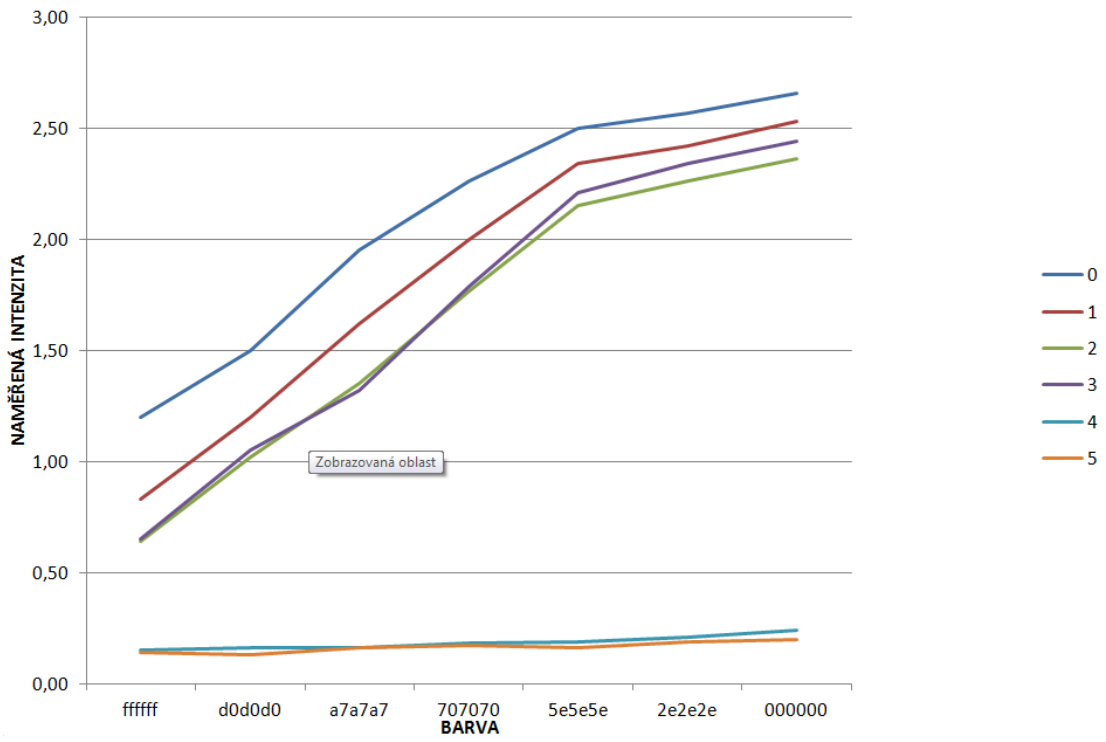
Obrázek 19: Vztah barvy a měřené intenzity pro robot 3.

5.3.3 Použití senzoru

Z měření vychází, že je nutno zasáhnout do hardwaru robotů a vyřešit saturaci některých senzorů. Dále pak bude potřeba udělat kalibraci senzorů, aby se hodnota naměřená jedním senzorem nelišila od hodnoty naměřené jiným.

Vzhledem k tomu, že není možné určit, který reálný robot (1-12) bude připraven na rezervovanou úlohu pod daným pseudonymem ("one", "two", ...), je obtížné provést kalibraci v uživatelském programu. Zmíněné hardwarové zásahy i kalibrace na úrovni firmwaru přesahují náplň této BP a jejich realizace je plánována vývojáři systému SyRoTek v další etapě vývoje.

Z výše uvedených důvodů nakonec není senzor v úloze použit a "měřená funkce"je načítána ze souboru. Nicméně výsledky této práce se staly podnětem pro inicializaci dalšího



Obrázek 20: Vztah barvy a měřené intenzity pro robot 8.

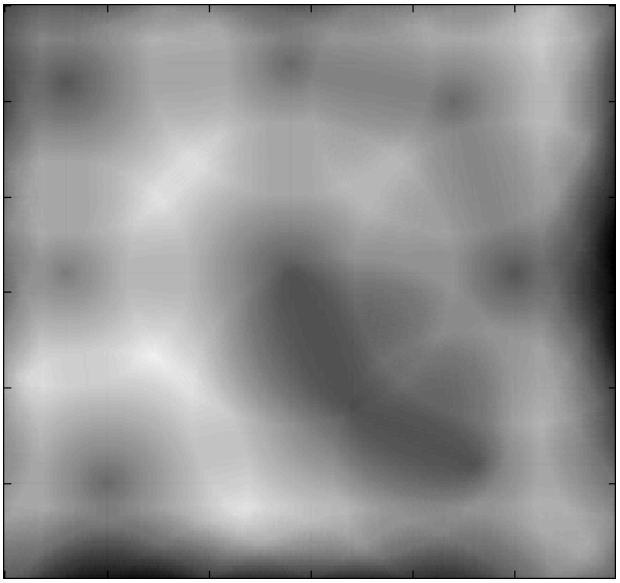
vývoje senzorového vybavení robotů platformy SyRoTek a budou použity k pozdější kalibraci senzorů.

5.4 Implementace PSO na šesti robotech SyRoTek

Jedná se o přizpůsobení algoritmu popsaného v kapitole 3.1.3 pro použití na reálných robotech systému SyRoTek. Dále zde musela být zavedena omezení daná přístupem k systému SyRoTek a jiný model robotu. Oproti zmíněné verzi algoritmu byl zaveden jiný model robotu a kolizní pozice byly řešeny tak, aby ani při pohybu robotů nedocházelo ke kolizím. Roboty se proto pohybovaly jeden po druhém.

Jako základ byla použita úloha *IMR_DANCE_6*[3] a kód zveřejněný vývojáři systému pro spuštění programu na šesti robotech. Měřená funkce je uvedena na Obrázku 21. Mapa byla generována pomocí funkce v Matlabu *getEnvironment.m*, kterou poskytl vedoucí

bakalářské práce. V implementované úloze roboty hledají co nejsvětlejší místo na 2D mapě (Obrázek 21) s využitím PSO algoritmu.



Obrázek 21: 2D mapa ve stupních šedi

5.4.1 Modifikace algoritmu

Program byl nejdříve z Matlabu přepsán do C++ a zde odladěn. Poté byl kód dále upravován přímo pro roboty SyRoTek. Algoritmus byl přizpůsoben pro použití na autonomních pozemních robotech pracujících v planárním prostředí. Algoritmus použitý na platformě SyRoTek nepoužívá rozšíření "local-best". Jedním z důvodů je, že pro úlohu bylo k dispozici jen málo robotů. Zachována jsou omezení "v-max", "inertia weight"a reálné rozměry robotů.

Dále bylo potřeba nastavit jiné parametry pro fyzikální model robotů. Pro kolize byl robot reprezentován válcem o poloměru $r = 0.16$, což zajistilo, že se k sobě roboty příliš nepřiblížily. Parametr v-max byl zvolen tak, aby nedocházelo v jednotlivých krocích ke kolizi.

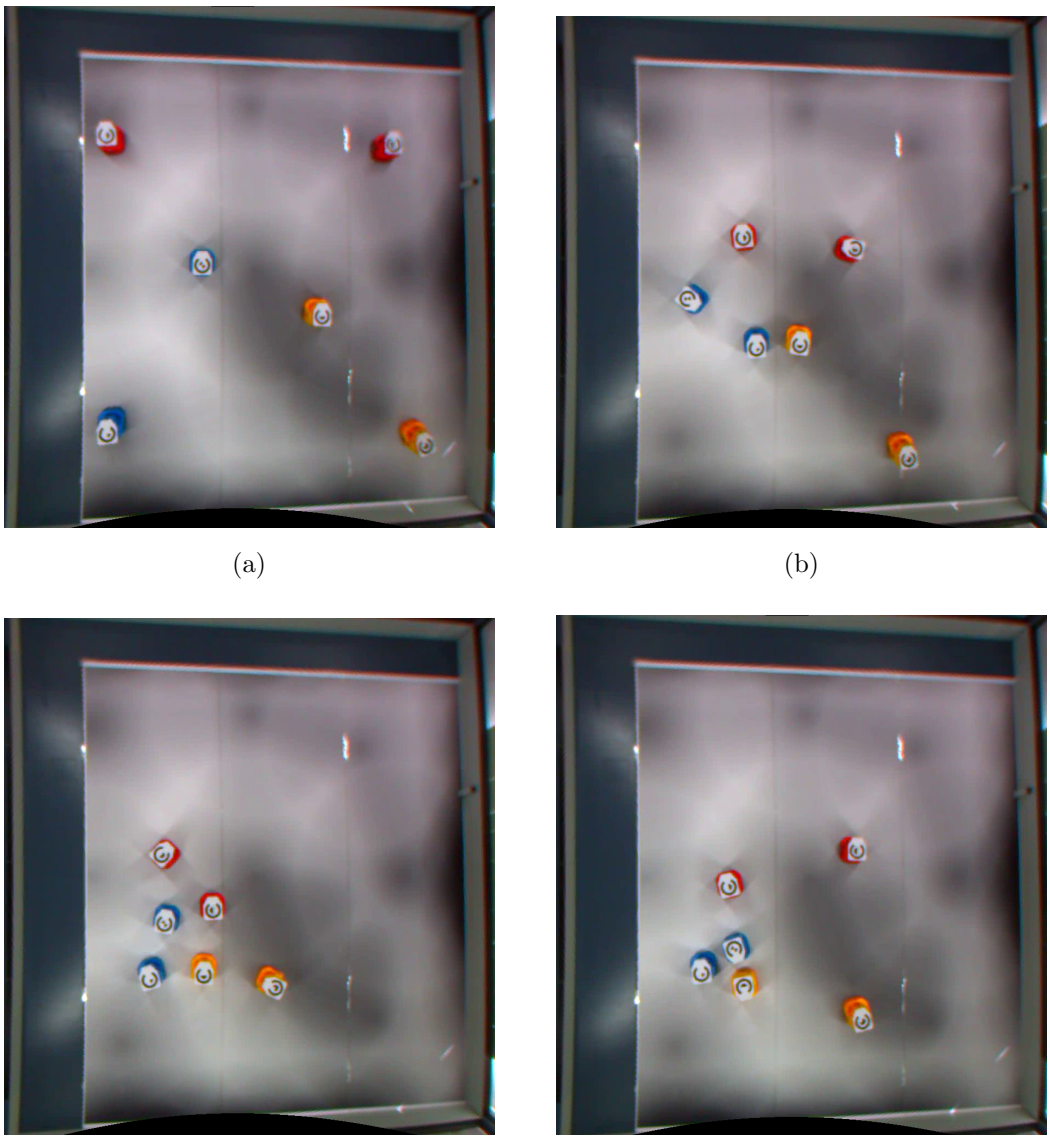
5.4.2 Spuštění programu na reálných robotech

Program byl nejprve testován v simulátoru Stage, poté byl přenesen na platformu SyRoTek. Tam se projevily rozdíly mezi simulátorem a reálnými roboty. Především šlo o rychlost pohybu, která ve Stage byla mnohem vyšší. Dále se pak objevily problémy s nekompatibilními příkazy rozhraní Player - simulátor Stage měl implementované všechny dostupné příkazy, platforma SyRoTek však ne. Z polohových příkazů byl implementován pouze ten, který nastaví požadovanou úhlovou a dopřednou rychlost. Tento problém byl v průběhu testování opraven a potřebný příkaz byl do platformy doimplementován.

Testování na reálných robotech doprovázely různé technické problémy způsobené nedokončeností platformy. Ať už se jednalo o defekty na jednotlivých robotech či výše zmíněné neimplementované funkce. Pro podporu dalšího zdokonalení systému a odladění problémů, které se objevily v souvislosti s použitím více robotů najednou, byla vytvořena zpráva pro tvůrce platformy SyRoTek, kterou je možno najít v kapitole *B Příloha - Zpráva SyRoTek*.

Vlastní experiment byl spuštěn po dobu dvaceti generací PSO algoritmu. Průběh testování je vidět na Obrázku 22. Počáteční rozestavení robtů je vidět na Obrázku 22a). Ve skupině obrázků jsou vidět některé zajímavé situace. V průběhu pokusu se roboty dostaly do poměrně těsné skupiny kolem globálního maxima 22c). Po dvaceti generacích se však dva roboty od skupiny vzdálily 22d) . To bylo dáno jejich informací personal-best a náhodnými parametry r_1 a r_2, které v dané generaci zapříčinily, že informace personal-best měla mnohem větší váhu, než global-best.

Experiment byl opakován desetkrát pro jedno rozestavení (konfiguraci) robotů v aréně na globálních souřadnicích $\{0.5, 0.5\}, \{0.7, 2.6\}, \{1.4, 1.2\}, \{1.7, 2.0\}, \{2.5, 0.7\}, \{2.5, 2.6\}$ viz Obrázek 22a). Pro druhou konfiguraci $\{0.3, 0.3\}, \{0.5, 2.4\}, \{1.2, 1.0\}, \{1.5, 1.8\}, \{2.3, 0.5\}$, $\{2.3, 2.4\}$ byl test spuštěn pouze třikrát. Výsledné hodnoty fitness jsou zaznamenány v Tabulce 6. Nejvyšší možná hodnota měřené funkce byla 60, nejnižší potom 0. Pro první uvedenou konfiguraci robotů v prostoru při spuštění je použito označení běhů čísly, pro druhou pak písmeny. Hodnoty uvedené v tabulce nedosahují vždy hodnoty 60, to je zapříčiněno



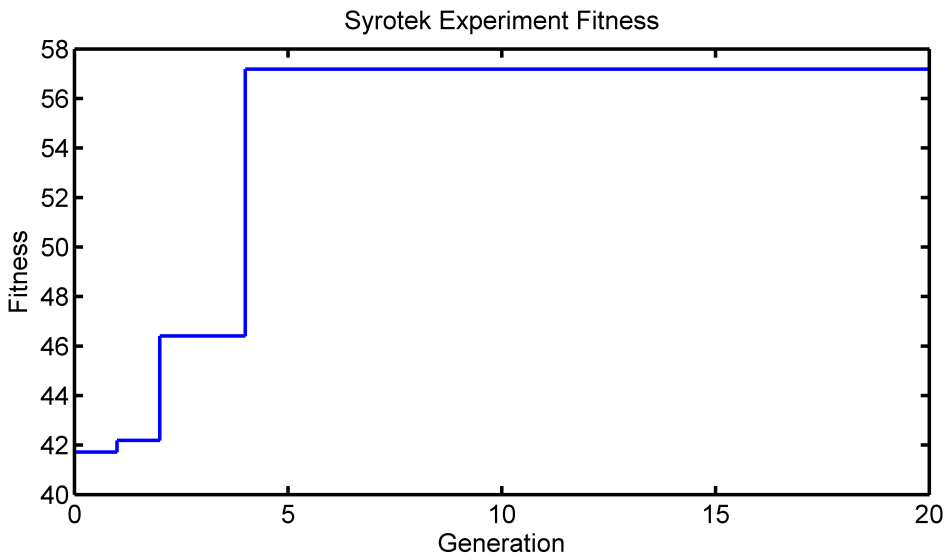
(a) (b) (c) (d)

Obrázek 22: Ukázka průběhu celého algoritmu

tím, že jsou roboty reálně velké a proto se stávalo, že robot stál nad hledaným extrémem. Ve všech případech roboty našly globální extrém. Oblast tohoto extrému měla rozměr několika cm^2 a proto byla dvakrát měřená hodnota rovna 60. Na Obrázku 23 je vidět průběh fitness pro jeden běh v první konfiguraci robotů v prostoru. Fitness je neklesající, protože šlo o hledání globálního maxima.

běh	fitness po dvaceti generacích
1	57,1875
2	58,1250
3	58,1250
4	60,0000
5	57,6562
6	58,1250
7	59,0625
8	57,6562
9	57,6562
10	59,5312
a	57,1875
b	57,6562
c	60,0000

Tabulka 6: Výsledné hodnoty fitness po dvaceti generacích pro testování na systému SyRoTek



Obrázek 23: Průběh fitness pro reálné roboty

6 Závěr

Cílem této práce byla úprava PSO algoritmu pro použití na reálných robotech. Algoritmus byl upraven přidáním podmínek, které zlepšují konvergenci a dále byla přidána omezení na pohyb robotů. Vzhledem k tomu, že je v PSO několik laditelných parametrů, byly tyto parametry laděny pro roboty reálných velikostí. V simulacích pro různě velké roboty algoritmus nejlépe konvergoval pro velikost robotu, která byla zadána při ladění těchto parametrů. Do nastavení některých konstant byl integrován model robotů. Do budoucna bude tento model integrován do dynamického plánování pohybu robotů mezi jednotlivými generacemi PSO algoritmu.

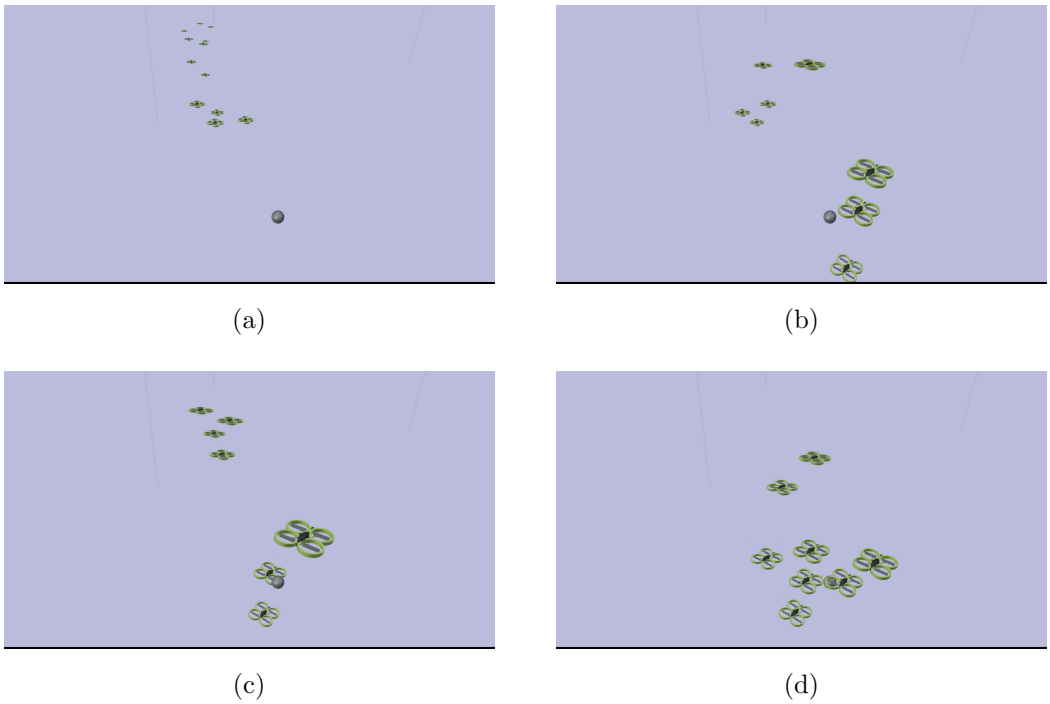
Mezi laděnými parametry se nachází parametr, který ovlivňuje, jakou měrou jsou roboty přitahovány k dosud nalezenému nejlepšímu místu. Ukázalo se, že pro roboty reálných rozměrů je vhodné tento parametr volit velmi malý. Důvodem je, že reálná velikost robotů neumožňuje jejich velké vzájemné přiblížení a algoritmus, který řeší kolize mezi roboty, pak negativně ovlivňuje konvergenci algoritmu k hledané hodnotě.

6.1 Relativní lokalizace robotů

Vzdálenost a dosah robotů hrají roli nejen při jejich lokalizaci, ale také ovlivňují konvergenci PSO algoritmu. Ukázalo se, že je vhodné, aby byly jednotlivé členy roje v dosahu co nejvíce ostatních. Algoritmus dobře konvergoval už pro skupiny o čtyřech členech.

Parametr dosahu jednotlivých robotů značně ovlivňoval konvergenci algoritmu. Pokud byl tento parametr zvolen příliš malý, musely být roboty v těsnějších skupinách, kde snáze docházelo ke kolizím.

Na Obrázku 24 je vidět několik snímků ze simulace výsledného algoritmu. Pro jednoduchost je z měřené funkce v prostoru zobrazen pouze bod, kde tato funkce nabývá globálního extrému. Na části 24a) je vidět počáteční stav, kdy jsou roboty vzdáleny od hledaného extrému a jsou rozděleny do tří skupin. Na částech 24b) a 24c) se dvě skupiny přibližují a



Obrázek 24: Ukázka průběhu celého algoritmu

na části 24d) je vidět, že se dvě malé skupiny robotů spojily do větší skupiny. V robotických simulacích se roj choval podle očekávání. Při postupném rozsvěcování místnosti prohledával roj vždy jinou oblast než v předchozím běhu.

6.2 SyRoTek

PSO algoritmus byl upraven pro potřeby robotické platformy SyRoTek. Vzhledem k technickým problémům se senzory, které jsou popsány v kapitole 5.3.3, nebyly tyto senzory využity pro měření funkce v prostoru. Místo toho byla mapa virtuálně načtena do každého robotu. Pro vizualizaci výsledků byla do arény s roboty umístěna mapa s vyobrazenou funkcí. Díky tomu bylo možno konstatovat, že se algoritmus choval podle očekávání.

Reference

[1] http://www.bit-tech.net. , 2012.

[2] N.H. Taeyoung Lee; Leoky, M.; McClamroch. Geometric tracking control of a quadrotor UAV on SE(3). In *Decision and Control (CDC), 2010 49th IEEE Conference on , vol., no., pp.5420-5425*, 2010.

[3] http://syrotek.felk.cvut.cz. , 2012.

[4] http://lynx1.felk.cvut.cz/syrotek. , 2012.

[5] J. Kennedy and R.C. Eberhart. Particle swarm optimization. In *Proceedings International Conference on Neural Networks IEEE*, pages 1942–1948, 1995.

[6] J. Kennedy and R.C. Eberhart. Swarm Intelligence. , 2001.

[7] L. Preucil L. Lhotska M. Saska, M. Macas. Robot Path Planning using Particle Swarm Optimization of Ferguson Splines. In *IEEE ETFA 2006 Proceedings*, 2006.

[8] http://playerstage.sourceforge.net. , 2012.

A Příloha CD

A.1 Obsah CD

```
/
├── bp ..... Bakalářská práce v pdf formátu
├── matlab/
│   ├── code/ ..... skripty a funkce pro Matlab
│   │   ├── PSO/
│   │   │   ├── algorithmPSO3D ..... hlavní spustitelný skript
│   │   │   ├── drawCylinder ..... pomocný vykreslovací skript
│   │   │   ├── drawPSO3D ..... funkce pro vykreslení PSO
│   │   │   ├── checkVelocity ..... funkce pro kontrolu rychlosti
│   │   │   ├── inRangePositionv2 ..... funkce pro kontrolu vzdálenosti robotů
│   │   │   ├── measure3DMultipointLinear ..... funkce pro výpočet fitness funkce
│   │   │   └── noColicionPosition ..... funkce pro řešení kolizních pozic
│   │   ├── sviceni/ ..... obdobně jako předchozí složka, ale ↓
│   │   │   └── sviceni ..... spustitelný skript
│   │   ├── getEnvironment ..... skript pro vytvoření 2D mapy
│   │   └── BoundedControl ..... skript pro úpravu vstupu kvadrikoptéry
│   └── data/ ..... data ze simulací - soubory s příponou mat
│       ├── test_c1_c2
│       ├── test_vmax_inertia
├── movies/ ..... soubory s příponou avi
│   ├── SyRoTek/
│   │   ├── malfunction ..... video, kde je vidět ztráta lokalizace robotu
│   │   └── syrotek2 ..... výslendý algoritmus na platformě SyRoTek
│   └── animation/
│       ├── noNeighbor ..... roboty nemusí hlídat sousedy a odlétávají
│       └── final ..... simulace výsledného algoritmu
└── SyRoTek_code/ ..... obsahuje dodaný template a ↓
    └── robot.cc ..... upravovaný soubor
```

B Příloha - Zpráva SyRoTek

Práce s platformou SyRoTek

V rámci BP jsem upravoval particle swarm optimization algoritmus pro potřeby robotické platformy SyRoTek. Aby algoritmus dobře fungoval, je potřeba více robotů, kteří mezi sebou sdílí informace. Dále by pak mohl být využit podlahový senzor, který roboty mají. Cílem této zprávy je vylepšení systému SyRoTek.

Programování v simulátoru Stage

V dodané linuxové distribuci nám scházel nějaký defaultní internetový prohlížeč. Dále by bylo vhodné dodávat rovnou s distribucí i root heslo - ve virtuálním stroji je potřeba nastavit mnoho věcí přes rozlišení obrazovky až ke změně fontů, aby byly čitelné.

Za největší nevýhodu dodané distribuce považuji nutnost ji spouštět ve virtuálním stroji nebo jako liveCD. Pokud počítač, na kterém je spouštěn virtualizovaný stroj, nemá hardwarovou virtualizaci, je i psaní kódu pomalé nemluvě o simulátoru Stage. Na druhou stranu nainstalovat požadované softwarové vybavení na vlastní distribuci není pro neznalce OS Linux snadné a vyžaduje úpravy knihoven apod.

V průběhu testování se stalo, že některé funkce, které fungovaly ve Stage, nebyly implementovány na platformě SyRoTek. Doporučuji vytvořit pro studenty seznam funkcí, které fungují jak ve Stage, tak na reálných robotech. Dále jsem zjistil při použití funkce *GoTo(x,y,yaw)*, že implementace této funkce na platformě SyRoTek je mnohem sofistikovanější, než ta, co je ve Stage. Ve Stage robot neplánuje nekolizní trajektorii a jede po přímce ze stávajícího bodu do daného bodu.

Práce s hardwarem

Značnou pomocí by byl jednoduchý spustitelný template vytvořený ke každé úloze/typu úlohy. Jde především o nastavení jednotlivých rozhraní pro potřebné senzory, a spuštění úlohy pro více robotů. Případně by se hodilo vytvořit tutoriál, jak napsat program pro Stage a jak ho potom přenést na platformu SyRoTek a spustit ho tam.

Velkou nevýhodou jsou i čekací sloty po rezervaci. Ty sice zajišťují, že se roboty, co zrovna byly použity, dobijí, ale zbytečně tak čekají i roboty, které se celou dobu rezervace dobíjely. Kdyby se podařilo nastavit systém tak, aby bral v potaz, že se některé roboty dobíjely, mohla by se doba použitelnosti systému zvýšit. Pokud by mělo systém používat více studentů, byl by tento problém citelnější.

Pro některé úlohy bude nejspíše důležité i na jakém robotu jsou spuštěny - musí se zohlednit hardwarové tolerance senzorů, které má každý robot jiné. Bylo by vhodné pokud by uživatel (student) mohl požádat systém o poslání určitého robotu (1-12) pod daným aliasem ("one","two", ...). Systém by potom buď provedl požadované, nebo napsal důvod, proč daný robot nepošle (vybití baterií, servis, nefunkčnost apod.).

Během rezervace se občas stalo, že robot přestal reagovat, nebo že se na rezervaci připravilo málo robotů. Tento problém by mohl řešit i student, pokud by mohl poslat systému příkaz o vyměnění robotu "one, two apod.". Toto by šlo spojit s výše uvedeným posláním daného robotu.

V průběhu testování byly zprovozněny kamery a bylo tak možno ovládat roboty i z domova. S vizualizací pouze z lokalizačního systému jsme si netroufli vzdáleně jezdit - občas se stalo, že postavení jednotlivých robotů z vizualizace lokalizačního systému arény se lišilo od reálu. Například systém o jednom robotu tvrdil, že není v aréně apod.

Shrnutí

Hlavní pomocí systému bylo, že jsme nemuseli řešit zprovoznění většího počtu robotů a jejich vzájemnou komunikaci. Pokud bychom museli toto řešit, nezbyl by čas na vlastní bakalářskou práci. Celková idea projektu SyRoTek se mi líbí, nicméně je ještě potřeba doladit nějaké detaily, aby systém fungoval bezchybně a nevyžadoval častý zásah ze strany jeho správce.