UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

6f5f88ed-843d-44a5-b0d4-e56418ab0f49

Bayesovské programovanie robotov

2011

Bc. Jakub Kondela

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Bayesovské programovanie robotov

Diplomová práca

Študijný program : Aplikovaná Informatika
Študijný odbor: 9.2.9 Aplikovaná Informatika
Školiace pracovisko: Katedra aplikovanej informatiky
Školiteľ: Mgr. Pavel Petrovič, Phd.

Bratislava, 2011 **Bc. Jakub Kondela**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Jakub Kondela
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: 9.2.9. aplikovaná informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský

Názov: Bayesovské programovanie robotov

Cieľ: Hlbšie preskúmať metódu Bayesovského Programovania Robotov. Implementovať v simulácii a príp. na realnom robotovi, navrhnúť úpravy, overiť, porovnať, vyhodnotiť.

Vedúci: Mgr. Pavel Petrovič, PhD.

Dátum zadania: 13.11.2009

Dátum schválenia: 03.05.2011

za

prof. RNDr. Ivan Kalaš, CSc.
garant študijného programu

Kondela Jakub
študent

Pavel Petrovič
vedúci

Čestne prehlasujem, že som túto diplomovú prácu vypracoval samostatne len s použitím citovaných zdrojov.

.....

Poďakovanie

Ďakujem hlavne svojím rodičom, ktorí mi umožnili štúdium na vysokej škole, ďakujem aj za neustálu morálnu podporu počas štúdia od všetkých členov rodiny. Ďakujem Mgr. Pavlovi Petrovičovi, Phd za cenné rady a pripomienky, ktoré mi pomohli pri písaní tejto práce.

Abstrakt

KONDELA, Jakub: Bayesovské programovanie robotov. Diplomová práca. Univerzita Komenského v Bratislave; Fakulta matematiky, fyziky a informatiky; Katedra aplikovanej informatiky. Bratislava (2011), 50s Školiteľ : Mgr. Pavel Petrovič, PhD.

V mojej diplomovej práci som rozoberal použitie dvoch metód na učenie robotov pomocou evolúcie, použitím bayesovského programu. Mojím cieľom bolo naučiť robota správanie obchádzania prekážok. Zameral som sa na genetický algoritmus a evolučnú streatégiu CMA-ES.

Kľúčové slová:

bayesovské programovanie robotov, evolúcia, evolučné algoritmy, CMA-ES

Abstract

KONDELA, Jakub: Bayesian Robot Programming. Diploma thesis. Comenius University in Bratislava; Faculty of mathematics, physics and informatics; Departmen of Applied Informatics. Bratislava (2011), 50p Supervisor : Mgr. Pavel Petrovič, PhD.

In my thesis, I have discussed two methods of applying evolution in learning robotics using Bayesian program.My goal was to learn a robot behaviour of obstacle-avoidance task. I have focused on genetic algorithm and evolutionary strategy CMA-ES.

Keywords:

bayesian robot programming,evolution,genetic algorithm, CMA-ES

Obsah

Úvod

Prostredie v ktorom sa roboti pohybujú je dynamické a ťažko predvídateľné a samostatný robot disponuje nejakými senzormi, ktorých hodnoty sú zašumené. Realizovať nejakú akciu pomocou deterministických pravidiel spravidla vedie k chybným výsledkom. V práci sa preto zameriame na riešenie nepresnosti a nejasnosti pravdepodobnostným prístupom pomocou jednotného frameworku – Bayesovské programovanie robotov.

Riadiaci systém robota sa nerozhoduje na základe aktuálneho stavu prostredia. Stav je nahradený vierou/predpokladom (belief) o stave sveta. Tento belief sa spravidla vyjadruje rozdelením pravdepodobnosti medzi možné stavy. Belief sa priebežne neustále aktualizuje na základe nových vnemov a takto integruje novú neurčitú informáciu s predchádzajúcou. Tým získava presnejší model sveta a vyberá akcie, ktoré maximalizujú jeho účinnosť.

Bayesovký program v mojej bakalárskej práci sa zameral na učenie, ktoré bolo uskutočňované manuálnym riadením. Učenie predstavuje jednu z najdôležitejších podstát v robotike, pokiaľ vyžadujeme od robota samostatnosť. Veľká časť úspešnosti závisí od samostatného návrhu systému dizajnérom. Vo všeobecnosti sa správanie učí postupným testovaním a úpravou návrhu. Iným pohľadom je evolučný prístup, kde sa samostatná štruktúra generuje automaticky počas evolučného vývinu.

Bayesovským programom si dokážeme celý systém robota zadefinovať do pravdepodobnostných distribúcií. Cieľom je charakterizovať určité správanie robota pomocou týchto distribúcií . Jednotlivé správania je potrebné robota naučiť. V práci sa chceme sa zamerať na rozšírenie programu o nový druh učenia formou evolúcie na jednoduchom experimente.

Bayesovské usudzovanie je dnes aktívna oblasť výskumu a objavujú sa úspešne aplikácie v rôznych oblastiach

1. Inteligentné roboty

Pojem robot môžeme definovať ako preprogramovateľný a multifunkčný ovládač, ktorý je navrhnutý tak, aby dokázal hýbať materiálmi, časťami, nástrojmi alebo špeciálnymi zariadeniami pomocou rôznorodých naprogramovaných pohybov, ktoré sú schopné vykonávať rozmanité úlohy.

Takáto charakteristika robota je pre nás nepostačujúca, pokiaľ od robota nežiadame vykonávať rôzne úlohy v neznámom prostredí. Potrebujeme, aby robot svoje akcie vykonával rozumne a aby bol schopný prispôsobiť sa nepredvídaným situáciám. Preto sa v robotike zaviedol pojem „umelá inteligencia“. Na to, aby sme mohli hovoriť o umelej inteligencií robota musí byť splnených niekoľko predpokladov:

- reprezentácia znalosti z okolitého sveta
- prehľadávanie riešení
- schopnosť učiť sa
- plánovanie a riešenie problému
- inferencia (získavanie otázky z neúplnej informácie)

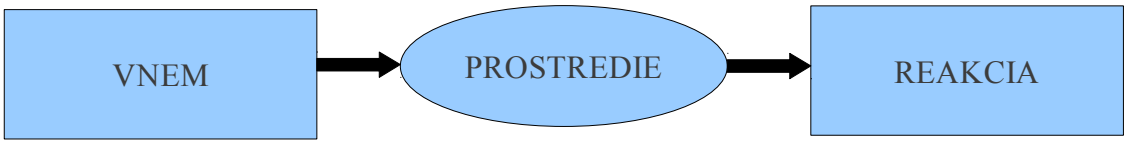
Pre nás bude inteligentný robot stroj, ktorý dokáže extrahovať informácie z prostredia v ktorom sa nachádza, vhodne ich spracovať a následne samostatne použiť akcie alebo rozhodnutia pre plnenie zadanej úlohy. Autonómnosť robota je pre nás kľúčová. Systém je autonómny, ak dokáže svoje správanie získať z vlastnej skúsenosti.

V našej práci pôjde konkrétne o mobilného reaktívneho robota, ktorý sa pohybuje v rovinnom prostredí, na ktorom je umiestnené niekoľko prekážok.

Správanie, ktoré požadujeme od robota, je naučiť ho obchádzať prekážky a pohybovať sa čo najrovnejšie.

2. Robotika založená na správaní

Správanie v robotike je výsledná reakcia agenta na podnet z prostredia (robot v blízkosti prekážky sa pohne v takom smere, aby sa tejto prekážke vyhol).



Obrázok 1. Schéma správania

Poznáme niekoľko druhov správania: správanie orientované na cieľ, preskúmavanie prostredia, sledovanie cesty, sociálne (kooperatívne) správanie. Výsledkom tejto paradigmy je celkové (globálne) správanie, ktoré vychádza z interakcie medzi základnými správaniami a prostredím. Tieto základné správania sú implementované v samostatných častiach systému.

Jednotlivé správania sú reprezentované na základe schémy vnímaj a konaj. Pri množine viacerých správaní tak máme niekoľko takýchto schém. Správania môžu prijímať informácie od rovnakých senzorov a ovplyvňovať viacero rovnakých aktuátorov. Na to, aby sme sa vyhli sporným situáciám, kde jednotlivé správania ovplyvňujú viacero rovnakých aktuátorov a svojimi akciami sa líšia (prípadne si odporujú) je potrebné rozhodovať, kedy sa majú jednotlivé správania vykonávať. Preto tu vystupuje koordinačný systém, ktorý rozhoduje kedy a ako sa majú dané správania vykonávať a teda na základe akého správania sa majú ovplyvniť aktuátory agenta.

Robotika založená na správaní väčšinou poukazuje viacej na biologický prístup. Takéto systémy častejšie robia chyby a preto opakujú svoje akcie a tým sa vylepšujú. Väčšinou sú navrhované pomocou skúšobného a testovacieho procesu, kde dizajnér systému upravuje aktuálne správania a postupne zväčšuje/ zmenšuje počet základných správaní. Daný postup sa používa, keď poznáme celkové prostredie. V takom prípade zohráva najväčšiu rolu dizajnér, pretože od jeho návrhu závisí celkový priebeh a výsledok správania systému.

Odlišný postup pri takýchto systémoch môžeme vidieť pri evolučnom prístupe. Evolučná robotika sa spolieha na automatický evolučný proces, ktorý väčšinou potrebuje väčšie množstvo skúšobných a testovacích fáz. Dizajnér tu už intuitívne nerozkladá globálne správanie, ale sa to uskutočňuje pomocou samo-organizujúceho procesu.

Cieľom mojej práce je robota naučiť správanie pomocou pravdepodobnostného modelu. Výhodou takýchto modelov je, že ku informáciám pristupujú s určitou pravdepodobnosťou. Nepresnosť, ktorou robot disponuje je získavaná z rôznych faktorov:

- Prostredie robota, ktoré podlieha fyzikálnym procesom je nepredvídateľné, premenlivé a dynamické.
- Samostatné senzory nepodávajú vždy presnú informáciu (ich hodnoty sú zašumené) a sú obmedzené svojou veľkosťou snímania, podávajú skreslenú alebo žiadnu informáciu pri náhlych zmenách prostredia (intenzívny svetelný lúč na snímaný objekt). Robot riadený kolesovými motormi pri riadení (natáčaní) podlieha motorickým šumom (zatáčanie nemusí byť presné ako sme robotovi pôvodne zadali).
- Veľkosť a stavba samostatného robota. Pri realizácií experimentu sa dizajnér musí prispôsobiť v závislosti od výpočtovej sily robota.
- Samostatný matematický model robota je len abstrakciou reálneho modelu a preto neposkytuje všetky informácie.

Riešením nepresnosti sa zaoberá pravdepodobnostná robotika. Jednotný framework pre využitie pravdepodobnostnej robotiky využíva „Bayesovské robotické programovanie“.

V práci sa chceme zamerať na prepojenie týchto dvoch paradigiem: Pravdepodobnostná robotika a evolučný prístup.

3.Bayesovské programovanie robotov

Bayesovské programovanie robotov(BPR) bolo prvýkrát predvedené v práci Bayesian Robot programming od Lebeltela a spol. [1]. V texte bol vysvetlený princíp BPR spolu s množinou rôznych experimentov na robotovi Kheppera. V bakalárskej práci sme túto metódu realizovali na reálnom robotovi, kde sme sa venovali niektorým jednoduchým experimentom s cieľom overiť metódu na inom robotickom systéme.

Cieľom tejto paradigmy je reprezentovať vnemy a akcie robota pomocou náhodných premenných a následne modelovať prehľadávací priestor na riešenie problému pomocou pravdepodobnostných distribúcií, pričom sa využívajú základné axiómy z logiky a pravdepodobnosti.

Bázou znalosti robota je abstraktný model, ktorý je reprezentovaný vzájomnou distribúciou všetkých relevantných náhodných premenných. Distribúcie pravdepodobnosti jednotlivých premenných sú získavané, buď a priori alebo počas behu samostatného programu. Kvôli neúplnosti modelu nie je vždy jednoduché prepojiť abstraktný model so samostatnými signálmi robota, ktoré sú získané zo senzorov robota. Keby sme porozumeli celému prostrediu, tak by sme mali možnosť nedostatok neúplnosti modelu eliminovať. Lenže prostredie, v ktorom sa robot nachádza je nepredvídateľné a preto je potrebné tento model prispôsobiť tak, aby bol robot schopný reagovať v prostredí, pre ktorý nie je špeciálne navrhnutý.

Bayesovský program využíva pravdepodobnostné modely a techniky a preto ponúka možné riešenie pre problémy s nejasnosťou, s ktorými sa robot vyskytuje a neúplnosťou prostredia, v ktorom sa nachádza.

Pre korektné používanie BRP potrebujeme zadefinovať tieto tri axiómy:

1. pravidlo konjunkcie

$$P(X*Y|\lambda)=P(X|\lambda)\times P(Y|X\wedge\lambda)$$
$$=P(Y|\lambda)\times P(X|X\wedge Y\wedge\lambda)$$

2. pravidlo normalizácie

$$\sum_x P(X|\lambda)=1$$

3. pravidlo marginalizácie

$$\sum_x P(X\wedge Y|\lambda)=P(Y|\lambda)$$

Pri programovaní robota , dizajnér (programátor) navrhne abstraktný model, ktorý je zložený zo základných geometrických , analytických a symbolických notácií.

3.1 Definícia Bayesovského programu

Bayesovský program pozostáva z dvoch časti:

1. popis bayesovského programu

1. špecifikácia BP
2. identifikácia

2. otázka bayesovského programu

1. fáza výberu

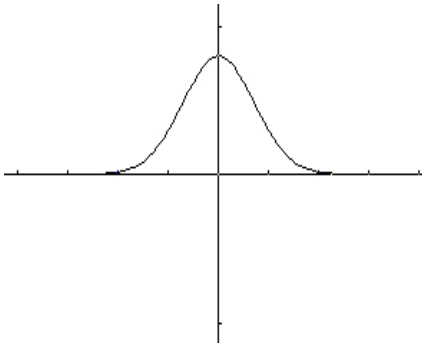
Prvou časťou *popisu bayesovského programu* je jeho špecifikácia. Špecifikáciou sa vytvára predbežná znalosť (π), predstavuje nejaké správanie. Predbežnou znalosťou sa rozumie súhrn všetkých a priori informácií o úlohe. Základnými prostriedkami programu sú premenné a ich pravdepodobnostné distribúcie.

Definovanie premenných

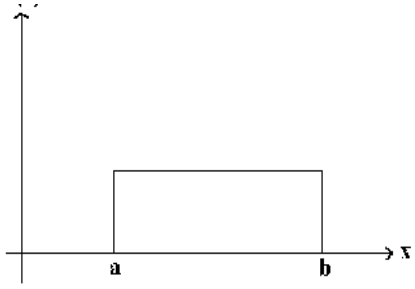
Jednotlivé výstupy senzorov sú najskôr predspracované do rozličných množín hodnôt prípadne intervalov. Jednotlivé atribúty na definovanie problému budeme definovať ako *premenné*. Premenné si rozdelíme na premenné senzorov a aktuátorov. V BP budú reprezentované ako diskrétne alebo spojité náhodné premenné.

Definovanie distribúcií

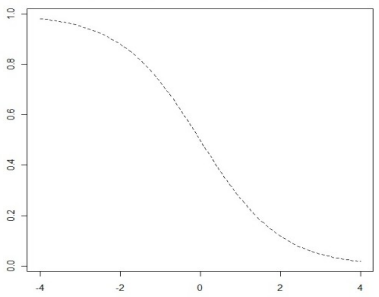
Náhodná premenná v BP je diskrétna alebo spojitá. Na reprezentáciu jednotlivých hodnôt premenných používame ich pravdepodobnostné rozdelenie. Charakterizuje s akou pravdepodobnosťou jednotlivé hodnoty nastanú. Na uchovanie jednotlivých distribúcií pravdepodobnosti v BP by sme potrebovali veľké množstvo pamäte. Preto v BP používame na reprezentáciu distribúcií parametrické formy. Parametrická forma je definovaná nejakou funkciou



Obrázok 2. Normálne rozdelenie

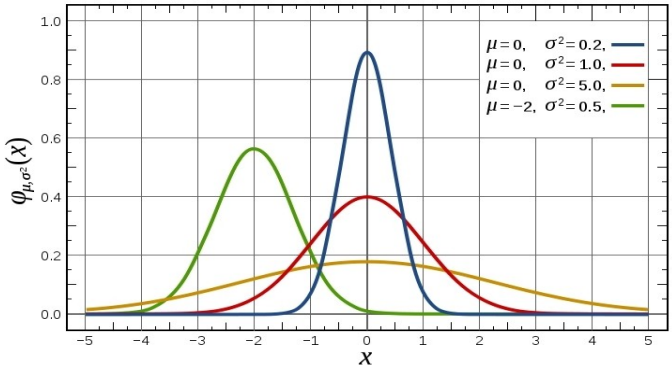


Obrázok 3. Uniformné rozdelenie



Obrázok 4. Sigmoidné rozdelenie

Výhodou týchto foriem je reprezentácia pomocou ich parametrov. V pamäti robota nám postačí uchovať si túto informáciu a tým drasticky zmenšíme veľkosť použitej pamäte ako pri explicitnej reprezentácií. Normálne rozdelenie, taktiež známe ako gaussovské, sa $\mu = E[X]$ používa v štatistike na $\delta^2 = E[(X-\mu)^2]$ reprezentovanie náhodných spojitých premenných, ktorých hodnoty sa s najväčšou pravdepodobnostnou združujú okolo jedného bodu. Tento bod sa nazýva stredná hodnota alebo medián distribúcie.Variancia tejto distribúcie definuje ako ďaleko sú všetky body vzdialené od tejto strednej hodnoty. Môžeme to chápať aj ako šírka našej distribúcie. Definuje rozptyl hodnôt distribúcie okolo strednej hodnoty.



Obrázok 5. Gaussové rozdelenie

Náhodné premenné senzorov sú nezávisle od ostatných premenných.

$$P(\text{premenné\_senzorov})$$

Distribúcie týchto premenných zadefinujeme uniformným rozdelením. Robot koná na základe toho čo vidí. Preto náhodné premenné senzorov ovplyvňujú premenné aktuátorov. Medzi týmito premennými bude podmienená závislosť.

$$P(\text{premenné\_aktuátorov} \mid \text{premenné\_senzorov})$$

Definovanie vzájomnej distribúcie

Použitím všetkých náhodných premenných definujeme vzájomnú distribúciu, ktorá tu predstavuje bázu znalosti robota. Vzájomná distribúcia predstavuje všetky možnosti prehľadávacieho priestoru. Keď si zoberieme do úvahy počet premenných a niektoré z nich sú spojité, tento prehľadávací priestor je pre nás časovo a pamäťovo neefektívny. Potrebujeme preto vhodné použiť niektoré zjednodušovacie a aproximačné techniky.

Dekompozíciou vzájomnej distribúcie využitím podmienenej nezávislosti premenných zredukujeme vo veľkej miere prehľadávací priestor.

Ďalšou značnou pomôckou je definovanie jednotlivých distribúcií pomocou parametrických funkcií. Takýmto prístupom nám na reprezentáciu postačia len určité parametre (stredná hodnota, rozptyl,...).

Identifikáciou sa vytvárajú experimentálne dáta (Δ). V tejto časti robot v závislosti od určenej paradigmy získava nové informácie. Na základe týchto hodnôt robot napĺňa parametre distribúcií.

Výsledný program môžeme charakterizovať dvojicou $<\pi, \Delta>$.

Pre lepšie znázornenie *otázky bayesovského programu* si rozdelíme premenné do dvoch tried. Premenné *hľadané* budú predstavovať množinu všetkých premenných, ktoré chceme zistiť. V našom prípade to pôjde o premenné aktuátorov. Druhou triedou bude množina *známych* premenných, ktorých hodnoty poznáme zo senzorov robota. Otázkou BP je distribúcia, ktorú môžeme zadefinovať ako $P(hľadané \mid známe)$. Inferenčným mechanizmom (použitím pravidla konjukncie a marginalizácie) dokážeme reprezentovať túto distribúciu pomocou vzájomnej distribúcie, ktorá je už aproximovaná a zjednodušená parametrickými formami.

$$P(hľadané \mid známe) \rightarrow_{inf} alfa * \sum P(hľadané \wedge známe) \rightarrow_{dek} alfa * forma_1(p_{1,} p_{2,..}) * forma_2(p_1) * ...$$

a*lfa* – normalizačná konštanta , p_i - parametre funkcie

Normalizačnú konštantu „alfa“ sme získali počas inferencií pri využití pravidla konjunkcie.

$$P(A \wedge B) = P(A) * P(A \mid B) \rightarrow P(A \mid B) = \underset{\text{normalizačná konštanta}}{1/P(A)} * \underset{\text{vzájomná distribúcia}}{P(A \wedge B)}$$

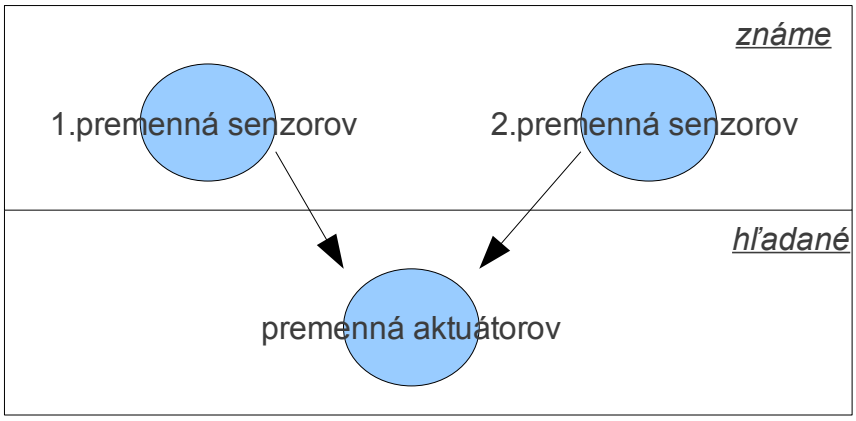
Na zodpovedanie otázky slúži fáza výberu. Pri fáze výberu robot postupuje podľa troch krokov:

1. nameraj hodnoty senzorov – hodnoty z_i zo známych premenných
2. v báze znalosti nájdi distribúciu $P(hľadané \mid známe = z_{1,} z_2)$
3. stochastickým výberom z tejto distribúcie pošli hodnotu aktuátorom

Pre lepšie znázornenie samostatného bayesovského programu nám poslúži reprezentácia bayesovského programu pomocou bayesovskej siete.

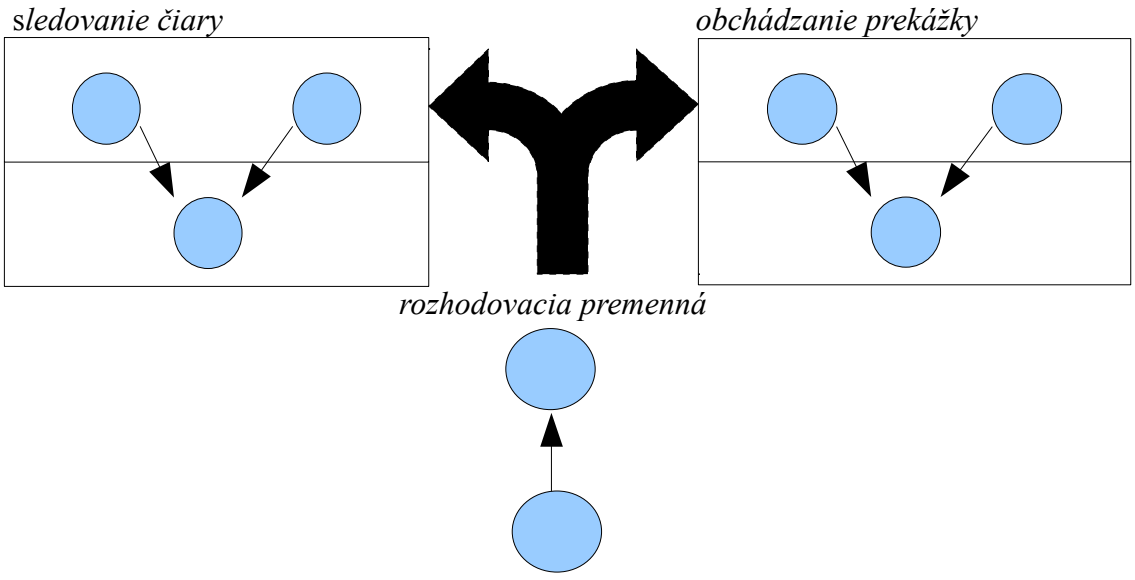
Bayesovská sieť je acyklický orientovaný graf, v ktorom vrcholy reprezentujú náhodné premenné a hranou medzi dvoma vrcholmi je znázornená podmienená závislosť medzi nimi.

Vrchol, v ktorom hrana končí je podmienený vrcholom, v ktorom hrana začína.



Obrázok 6. BP ako bayesovská sieť

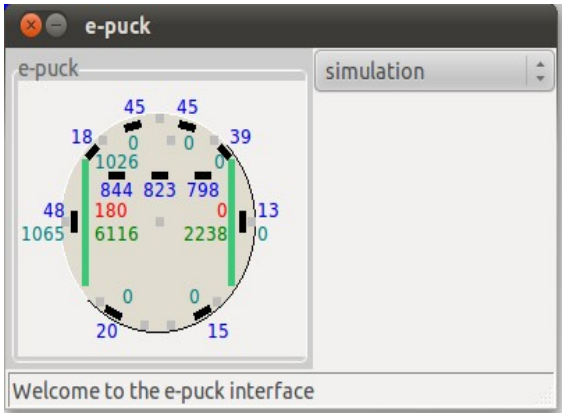
Takáto sieť al. bayesovský program reprezentuje jedno správanie. Bayesovský program je ľahko použiteľný, keď požadujeme od robota vykonávať úlohu, ktorá potrebuje využiť viacero správaní. Predpokladáme, že chceme od robota, aby sa vyhýbal prekážkam a keď nájde čiaru, aby sledoval čiaru. Keďže obidva správania ovplyvňujú tie isté aktuátory musíme si v bayesovskom programe pomôcť novou pomocnou premennou, ktorá nám bude rozlišovať kedy sa má ktoré správanie vykonať. Distribúciu tejto premennej môžeme zadefinovať sigmoidným rozdelením a jej hodnoty budu závisle od vzdialenosti k prekážke. Čím bližšie sa budeme približovať ku prekážke tým sa bude zväčšovať pravdepodobnosť na vykonanie obchádzania prekážky a na druhej strane zmenšovať pravdepodobnosť vykonania správania na sledovanie čiary a naopak.



Obrázok 7. Ilustratívna schéma pre dva správania

3.2 BP v úlohe obchádzania prekážok

Na to, aby sme úspešne dokázali objekt obísť potrebujeme vedieť ako ďaleko sme od neho vzdialený a ktorým smerom je robot naň otočený. Z výstupov senzorov vieme tento predmet lokalizovať a potrebné hodnoty vypočítať. Robot použitý v experimente je vybavený ôsmymi infračervenými senzormi a pohybuje sa dvomi krokovými motormi.



Obrázok 8. Poloha senzorov a krokových motorov. [Obdĺžniky naokolo charakterizujú polohu senzorov (4 vpredu, 2 po boku a 2 vzadu) a dlhé zelené sú dva krokové motory]

Prostredie robota je rovná plocha obkolesená stenou. Na ploche sa nachádzajú kvádrové objekty. Robot sa pohybuje tak , že mení rýchlosť otáčok na jednotlivých motoroch a tým mení svoj smer.

- **premenné senzorov**

Výpočet polohy objektu vzhľadom k robotovi definujeme hodnotou od 0 po 7. Výsledná hodnota charakterizuje, ktorý senzor dosahuje v aktuálnom čase behu programu najvyššiu hodnotu.

Premenná bude mať označenie *Pol*. V BP to bude diskrétna náhodná premenná.

$$Pol = pol_t = argmax_i(senzor[0], \ldots, senzor[i], \ldots, senzor[7])$$

Distribúciu tejto premennej zadefinujeme uniformným rozdelením.

Výpočet vzdialenosti najbližšieho objektu sa vypočíta ako najvyššia hodnota zo všetkých senzorov v aktuálnom čase behu programu, ktorá sa ešte znormalizuje konštantou. Tým dostaneme hodnoty v rozsahu od 0 po 3. Premennú nazveme *Vzd*. V BP to bude diskrétna náhodná premenná.

$$Vzd = vzd_t = floor(max(senzor[i])/625)$$

Distribúciu tejto premennej zadefinujeme uniformným rozdelením.

- **premenná aktuátorov**

Výpočet uhlovej rýchlosti závisí od aktuálnej rýchlosti obidvoch motorov. Ak sa robot pohybuje rovno, má rovnakú maximálnu hodnotu na obidvoch motoroch. Pri zatáčaní doľava (doprava) sa zmenší hodnota ľavého (pravého) motora. V experimentoch je použitých celkovo 7 hodnôt rýchlosti (jedna rovno a po tri pre zatáčanie doľava a doprava). Hodnoty sú v intervale od -3 po 3 (-3 úplne doľava,+3 úplne doprava, ...). Premennú nazveme *rýchlosť*. V BP to bude spojitá náhodná premenná. $Uhl = uhl_t = (-3,3)$

Po definovaní všetkých premenných nastáva dekompozícia vzájomnej distribúcie.

$$P(Vzd \wedge Pol \wedge Uhl) =_{dek} P(Vzd) * P(Pol) * P(Uhl \mid Vzd \wedge Pol)$$

Jednotlivé distribúcie, ktoré nám vznikli parametrizujeme.

$$P(Vzd) = uniformné\, rozdelenie = a$$

$$P(Uhl) = uniformné\, rozdelenie = a$$

$$P(Uhl \mid Vzd \wedge Pol) = normálne\, rozdelenie = (\mu_i, \delta_i)$$

Keďže distribúcie senzorových premenných sú konštantné môžeme aproximovať.

$$P(Vzd) * P(Pol) * P(Uhl \mid Vzd \wedge Pol) =_{dek} a * P(Uhl \mid Vzd \wedge Pol) =_{dek} a * forma(\mu, \delta)$$

Otázkou v tomto BP je distribúcia $P(Uhl \mid Vzd \wedge Pol)$. Na základe nameraných hodnôt premenných senzorov chceme vedieť hodnotu premennej aktuátorov.

$$P(Uhl \mid Vzd \wedge Pol) =_{inf} alfa * P(Uhl \wedge Vzd \wedge Pol) = alfa * forma(\mu, \delta)$$

Úlohou identifikácie je natrénovať parametre jednotlivých distribúcií. Proces učenia v bayesovskom programe môže vstupovať len počas identifikačnej fázy. Výhodou použitia BP spočíva v tom, že pri zmene učenia v BP samostatný program ostáva nezmenený. Mení sa len proces získavania experimentálnych dát. Jednou z možnosti je riadenie robota manuálne určitý čas, kde učiteľ riadi robota v snahe získať čo najlepšie informácie. Problémom v tomto prístupe je, že počas riadenia robota niektoré situácie nenastanú a preto parametrom týmto distribúciam nastavíme prednastavenú hodnotu. Ďalším problémom je samostatné riadenie robota človekom, ktorého riadenie nemusí byť presné. Ak požadujeme aby robot bol schopný plniť úlohy v neznámom dynamickom prostredí, takýto prístup zbieraní správnych informácií nie je najvhodnejší. V práci sa preto chceme zamerať na odlišný prístup učenia robota.

4.Učenie robota

Učenie robota môžeme voľne interpretovať ako problém umelej inteligencie (AI problem). Rieši problém vložených systémov, kde žiak, ktorého učíme sa nachádza v neznámom dynamickom systéme. Učenie je založené na tom, že riadiaci systém môže byť učený pomocou nekompletných dát z prostredia pomocou už nameraných dát, ktoré sa získali počas trénovacej fázy. Z týchto poznatkov je systém schopný generalizovať získané vlastnosti v nových situáciach. Robot musí riešiť neustále problémy s neznámou informáciou, s premenlivým (dynamickým) prostredím a samostatným problémom naučiť sa danú schopnosť (správanie).

Niektoré obmedzenia robota:

- chybné senzory
- nedeterministické akcie
- reaktívnosť systému
- inkrementálnosť
- čas na učenie

Základnou motiváciu tohto prístupu je, že by mal produkovať lepšie výsledky ako prístupy založené na explicitnom dizajne. Pri učení sa využívajú rôzne algoritmy: algoritmus spätného šírenia (Rumelhart, 1986), učenie odmenou a trestom (Barto, 1995), samo organizujúce siete (Kohonen, 1982), evolučný algoritmus (Holland, 1975).

Tieto algoritmy sú postavené na troch základných učiacich paradigmách: učenie s učiteľom, učenie bez učiteľa a učenie odmenou a trestom.

4.1 Typy učenia

Učenie s učiteľom

Učenie s učiteľom sa najviac odlišuje od ostatných tým, že pre daný problém existuje určitá množina trénovacích dát pozostávajúca zo vstupných a výstupných hodnôt. Na základe týchto dát dokážu metódy vytvoriť odpoveď na riešenie problému charakterizovaný trénovacími dátami. Cieľom je navrhnúť takú metodiku, aby natrénovaný systém z trénovacích dát dokázal správne vyhodnotiť výsledky z rozdielnych dát ako trénovacie. Preto je potrebné naučiť učiaci algoritmus generalizovať z trénovacích dát na neznáme. Experiment bayesovského programu, ktorý som realizoval v bakalárskej práci sa najbližšie približoval práve k tejto paradigme. V práci som na jednoduchom experimente obchádzania prekážky dokázal schopnosť generalizácie. Robot po natrénovaní (identifikačná fáza) bol schopný objekt na ploche obchádzať, aj keď sa s objektom počas behu manipulovalo, prípadne sa zmenil samostatný objekt. Generalizácia patrí medzi výhody Bayesovského programu.

Pre danú trénovaciu množinu $\{(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)\}$ učiaci algoritmus hľadá takú funkciu *f*, ktorá dokáže urobiť transformáciu $x \rightarrow y$ s minimálnou chybou.

Učenie bez učiteľa

Trénovacie dáta sú zložené len zo vstupných hodnôt. Cieľom je nájsť nejakú skrytú štruktúru v týchto dátach. Keďže nemáme výstupné hodnoty v trénovacích dátach, nedokážeme nájsť takú transformačnú funkciu na danú množinu. V tejto metodike sa vo väčšine zo vstupných dát vyextrahujú určité kľúčové vlastnosti. V týchto metódach sa využívajú prevažne štatistické výpočty.

Učenie odmenou a trestom

Cieľom je nájsť najlepšie možné správanie v danej úlohe, aby sme maximalizovali výkon agenta. Agentovi sú poskytnuté informácie o všetkých možných akciách a taktiež agent dostane informáciu o hodnotení vykonanej akcie (zisk), ale nie o tom, aká akcia mala byť správna. Robot je na základe tohto hodnotenia (odmena alebo trest) schopný naučiť sa dané správanie.

Učenie evolúciou môžeme zaradiť medzi hybridný typ učenia bez učiteľa a odmenou a trestom

5.Učenie robotov pomocou evolúcie

Umelá evolúcia vychádza z Darwinovskej evolúcie. Darwinová evolučná teória si zakladá na troch základných zložkách: prirodzený výber, náhodný genetický drift a reprodukčný proces.

Hlavná myšlienka je založená na tom, že na jednotlivé prípustné riešenia problému sa pozeráme ako na živé organizmy v umelom prostredí. Tieto organizmy majú priamo definované svoje možnosti pôsobenia v tomto prostredí. Majú schopnosť prežiť a schopnosť reprodukcie, kde realizácia týchto akcií závisí od ohodnotenie týchto organizmov, ktoré hovorí ako dobré sú tieto prípustné riešenia. Proces hľadania najlepšieho riešenia spočíva vo výbere nejakej počiatočnej populácií organizmov a v následnej simulácií ich vývoja prostredníctvom evolučných mechanizmov (prirodzený výber, mutácia). Ako sa táto populácia od generácie ku generácií vyvíja, zlé riešenia máju tendenciu vymierať a dobré riešenia sa medzi sebou navzájom krížia a produkujú ešte lepšie riešenia.

Na to, aby sme dokázali robota úspešne naučiť určité správanie pomocou evolúcie musíme brať do úvahy niektoré problémy, s ktorými sa stretávame:

- mechanická robustnosť – indivídua v prvotných generáciách môžu produkovať také správania, ktoré môžu poškodiť samostatný hardvér robota, ako napríklad prílišná rýchlosť robota, ktorá spôsobí kolíziu so stenou alebo nejakými objektami alebo neustále tlačenie robota do steny
- výdrž batérie – trvanie evolučných experimentov trvá dlhšie ako priemerná výdrž batérie, preto je potrebné nájsť iný prístup prísunu elektrickej energie do robota, prípadne obísť tento problém inou cestou
- analýza – ovládací systém robota môže byť veľmi komplexný čo spôsobuje problém navrhnúť takýto systém pomocou jednoduchých správaní
- časová efektivita – evolúcia môže trvať niekoľko hodín, prípadne aj dní v závislosti od zložitosti experimentu
- návrh účelovej funkcie – výber rôznych kritérií pri zostavovaní účelovej funkcie bude mať veľký vplyv na evolučný proces, dosahovať úspešnosť účelovej funkcie v neznámom prostredí je taktiež obtiažne

Ako vidíme realizácia evolúcie na reálnom robotovi je hlavne časovo náročná, pričom robot sa môže značne poškodiť, čo môže zapríčiniť nedokončenie experimentu. Preto je účinné použitie simulácie robota v nejakom simulátore. Aj keď evolúcia na reálnom robotovi je prijateľná, sériové vyhodnotenie indivíduí pri reálnom robotovi by trvalo podstatne dlhšie. Použitie paralelného prístupu a zabezpečenie väčšieho množstva identických robotov by malo za výsledok zrýchlenie celého experimentu. Pri využívaní viacerých robotov pomocou paralelizmu na evolúciu je dôležité si uvedomiť, že roboty, ktoré majú rovnakého výrobcu, sa ich výsledky stále môžu líšiť.

Jedným z prístupov, je evolúciu realizovať v simulátore a potom úspešných jedincov použiť v reálnom prostredí na reálnom robotovi. Takéto riešenie má podľa Brooksa ďalší problém „Bez pravidelnej validácií na reálnom robotovi, je veľké nebezpečenstvo, že celé úsilie vynaložené na riešenie problému jednoducho nebude spolupracovať s reálnym fyzickým robotom a že je reálne nebezpečenstvo (takmer isté), že program, ktorý dobre pracoval v simulátore, nakoniec bude úplne neúspešný na reálnom robotovi A to kvôli rozdielnosti vnímania a ovládania v reálnom prostredí. Teda je veľmi ťažko simulovať aktuálnu dynamiku reálneho sveta.“[1] .

Ale zase na druhej strane v niekoľkých experimentoch bolo ukázané, že robot bol schopný úspešne zvládnuť daný problém v reálnom prostredí. Pričom robot bol natrénovaný evolúciou v simulátore. Preto existuje určitá trieda interakcií robot - prostredie, pri ktorých je schopný sa robot naučiť v simulácií.

5.1 Evolučné algoritmy

Evolučná algoritmy patria medzi moderné prístupy na riešenia problémov v ťažkých situáciach, kde deterministický prístup zlyháva z dôvodu veľkej časovej alebo pamäťovej náročnosti. Sú efektívne v prípadoch kde nepotrebujeme dosiahnuť len optimálne riešenia, ale si vystačíme aj s kvalitným sub-optimálným riešením. Patria medzi základné prostriedky modernej numerickej matematiky pre riešenie zložitých optimalizačných problémov. Používajú sa vtedy, keď hľadáme také globálne minimum, ktoré je obklopené veľkým množstvom lokálnych miním.

Môžeme ich chápať ako abstrakcia a formalizácia Darwinovej evolučnej teórie. Poskytujú univerzálny algoritmus pre simuláciu evolúcie, kde je potrebné len modifikovať spôsob určenia účelovej funkcie.

1 [2]

Techniky evolučných algoritmov:

1. Genetický algoritmus
2. Genetické programovanie
3. Evolučné programovanie
4. Evolučné stratégie
5. Neuroevolúcia

Všetky sú založené na koncepte simulácie evolúcie jednotlivých štruktúr pomocou procesov ako sú selekcia, mutácia a reprodukcia . V práci som sa zameral na genetické algoritmy a evolučné stratégie.

5.2 Genetický algoritmus

Genetický algoritmus je prehľadávacia technika, ktorá vychádza z prirodzenej evolúcie organizmov. Táto technika je založená na heuristike, ktorá generuje prípustné riešenia v danom prehľadávacom probléme. Tento algoritmus po prvýkrát predstavil Holland v roku 1975. Patrí medzi najpopulárnejšie algoritmy v evolučných výpočtoch v informatike.

Pracujú s určitou populáciou jedincov, ktorí sa jednotne nazývajú populácia. Populácia je zložená z chromozómov, čo sú lineárne reťazce symbolov. Umelý chromozóm sa najčastejšie reprezentuje binárnym reťazcom, vektorom reálnych čísiel alebo binárny reťazec s použitím gray-ovím kódovania. Samostatné reťazce sa môže reprezentovať nielen binárnou sústavou ale aj terárnou, atď. Reťazec symbolov je genotypom chromozómu.

binárny genotyp s gray kódovaním

V robotike binárna reprezentácia na charakteristiku uhlovej rýchlosti by malo za dôsledok, že len malá zmena v bite by zmenila veľký skok, ktoré by mohlo viesť ku novej vlastnosti. Preto sa používa gray kód. Je to binárny numerický systém, kde sa dva nasledujúce čísla líšia len v jednom bite.

genotyp ako reálny vektor

Hodnoty genotypu sú reálne čísla. Kódovanie reálnymi číslami nám zabezpečí požadovanú presnosť.

Populácia chromozómov je zakódované aktuálne riešenie optimalizačného problému, každý takýto chromozóm je ohodnotený svojou účelovou funkciou. Chromozómy, ktoré kvalitne riešia problém sú ohodnotené vyššou hodnotou účelovej funkcie. Na to, čí daný chromozóm bude vstupovať do procesu reprodukcie taktiež závisí od veľkosti účelovej funkcie. So zväčšujúcim fitness sa zväčšuje aj pravdepodobnosť, že chromozóm sa bude reprodukovať. V reprodukcií je zahrnutý aj náhodný proces mutácie jednotlivých častí. V reťazci sa náhodné vybratý symbol zamení za iný náhodne vybratý symbol.

V umelej evolúcií sa používa účelová funkcia na ohodnotenie výkonu jednotlivých jedincov a následný výber najlepších z nich. Celkový priebeh evolúcie závisí práve od návrhu a formy tejto funkcie.

Účelová funkcia pre autonómne roboty väčšinou zahŕňa také premenné a konštanty, ktoré hodnotia celkový výkon robota vzhľadom k očakávanému správaniu. Tieto premenné sa ťažko hľadajú, pretože nepoznáme celkové správanie robota. Je dôležité vedieť ako kombinovať tieto premenné vo funkcií. Premenné, ktoré ohodnocujú celkovo správanie môžu byť v skorých generáciách neschopné ohodnotiť primitívne správanie robota a preto môžu dosahovať nulovú hodnotu.

Tieto problémy spojené s návrhom fitness funkcie sa dajú reprezentovať pomocou trojrozmerného fitness priestoru:

1. Funkcionálna - preferujúc správanie

   Tieto fitness funkcie preferujú buď špecifickú funkcionalitu systému alebo celkové výstupné správanie robota.

   Príklad: Robot, ktorého chceme naučiť kráčať pomocou neurónovej siete. Tak preferujeme buď jeho frekvenciu z neurónov alebo naučíme časti robota, ktoré korešpondujú so vzorkou chodenia.

2. Explicitná - Implicit

   Definuje počet premenných a konštánt vo funkcií, či už je to veľký počet (explicitná) alebo malý (implicitná).

   príklad: Implicitná potrebuje len vedieť, či robot má ešte energiu a v explicitnej by funkcia obsahovala aj napríklad či už je robot v nabíjacej ploche, jeho rýchlosť, aktivita určitých neurónov, ...

3. Externá – Interná

Funkcie, ktorých premenné a konštanty sa počítajú z informácií z robota (interná) alebo z informácií zo samostatného prostredia (externá).

príklad : Absolútna vzdialenosť robota k cieľu nemôže byť braná do úvahy ako vstup pre reálneho robota. Naopak pri realizácií experimentu v simulátore si s touto informáciou môžeme pomôcť.

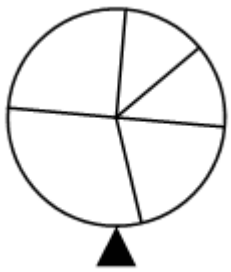
Samostatný proces genetického algoritmu:

1. vytvor štartovaciu populáciu (niekoľko náhodne vygenerovaných chromozómov)
2. ohodnoť každý chromozóm účelovou funkciou
3. SELEKCIA : vyber pár chromozómov použitím jednej z metód selekcie
4. REPRODUKCIA: použi kríženie na vybratý pár a mutáciu na jednotlivcov
5. vytvor novú populáciu s časťou pôvodných chromozómov a novovytvorených chromozómov opakuj od bodu 2 až keď nenastane nejaké ukončovacie kritérium

Na začiatku každej novej generácie nastáva selekcia (výber) jedincov do reprodukcie v závislosti od ich fitness hodnoty.

Jednou z najpoužívanejších metód je tzv. „**točiaca ruleta**“. Metóda je založená na kvázináhodnom výbere dvoch chromozómov v závislosti od ich *fitness* funkcie. Chromozómy, ktoré majú väčšiu fitness hodnotu, majú väčšiu pravdepodobnosť, že budú vybrané. Každá časť tejto rulety predstavuje jedného jedinca a veľkosť každej časti je proporčný k veľkosti *fitness* funkcie každého jedinca. Keď máme jednotlivý *fenotyp* jedinca x_i a jeho *fitness* $f_i = \Omega(x_i)$ tak veľkosť časti F_i je proporčne normalizovaná celkovým fitness populácie s p jednotlivcami

$$pi = \frac{f_i}{\sum_n f_i}$$



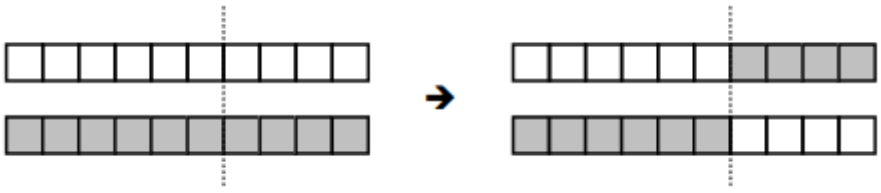
Obrázok 9. točiaca ruleta

Druhým prístupom pri výbere je pomocou **hodnosti**. Populácia je usporiadaná podľa hodnoty účelovej funkcie od najväčšej po najmenšiu. Chromozómy s najmenšou účelovou funkciou majú najnižšiu hodnosť a najlepšie majú najvyššiu hodnosť. Výber chromozómov je proporčne závislý od hodnosti a nie od hodnoty účelovej funkcie. Medzi ďalšie modifikácie patrí aj orezanie vybranej množiny jedincov podľa hodnosti. Jedinci sa usporiadajú podľa hodnoty účelovej funkcie , ale do procesu reprodukcie postupuje len niekoľko prvých jedincov.

Medzi ďalšie metódy patrí aj výber jedincov **turnajom**. Metóda je založená na vzájomnom turnaji vybraných dvoch jedincov. Jedinec je zložený z víťazného jedinca v turnaju. Turnaj pozostáva z generovania náhodného čísla *r* v rozpätí 0 až 1. Ak je *r* menšie ako nejaké prednastavené číslo *,* tak sa potomok generuje z jedinca, ktorého fitness hodnota je väčšia. Tieto dva jedince sú potom naspäť vložené do populácie. Výhody tejto metódy spočívajú v tom, že nie je potrebné vypočítať globálnu fitness hodnotu ako to je pri metóde s ruletou alebo pri metódach s hodnosťami. A preto je výhodná pri paralelnej implementácií.

Po výbere jedincov nastáva proces *reprodukcie* medzi týmito jedincami. Potom ako sme vytvorili novú populáciu výberom, nový potomci sú náhodne spárovaní, krížení a mutovaní.

Kríženie dvoch jedincov nastáva vždy s určitou prednastavenou pravdepodobnosťou. Náhodne sa vyberie bod kríženia pozdĺž celého chromozómu. Miesto za týmto bodom si jedince navzájom vymenia. Takýto prístup sa nazýva *jednobodovým krížením. Takýmto* postupom vznikajú noví dvaja jedinci a starí zaniknú. Existuje aj *mnohobodové kríženie*, kde kríženie pozostáva z viacerých bodov na chromozóme.



Obrázok 10. Jednobodové kríženie jedincov

Mutácia nastáva po aplikovaní kríženia na dvoch chromozómoch s určitou pravdepodobnosťou. Pre každú časť nového chromozómu môže nastať prepnutie hodnoty na inú s touto pravdepodobnosťou. Vo väčšine sa táto pravdepodobnosť nastavuje veľmi nízka a preto mutácia nastáva vo veľmi ojedinelých prípadoch. Pri veľkých hodnotách pravdepodobnosti by sa prehľadávanie zmenilo na lokálne prehľadávanie. V binárnej reprezentácií je to zmena O na 1. V ostatných reprezentáciách je to náhodné zvolený symbol z daného jazyka. Pri vektore reálnych čísiel sa buď náhodne zvolí číslo z daného rozsahu alebo sa zväčší o malú hodnotu vybratú z normálneho rozdelenia so strednou hodnotou 0. Cieľom mutácie je začleniť do evolúcie rozmanitosť. Algoritmus sa môže touto technikou vyhnúť lokálnym minimami a vzniká tým rôznorodosť populácie. Rôznorodosťou zabránime spomaľovaniu alebo úplnému zastaveniu evolúcie.

Obrázok 11. Mutácia jedincov

Po aplikovaní základných genetických operátorov (mutácia , kríženie) vzniknú takto noví jedinci, ktorých počet je rovnakí ako predchádzajúca generácia. Vo väčšine metód genetických algoritmov sa používa *elitizmus*, kde najlepší jedinec sa dostáva do novej populácie bez zmeny. Takýto úsečný jedinec sa nestratí a bude prispievať k tvorbe nových jedincov s lepším fitness.

Cyklus evolúcie sa môže zastaviť keď určitý jedinec v populácií dosahuje požadovanú fitness alebo bez významnej zmeny aktuálnej fitness funkcie alebo prekročením dohodnutého času evolúcie. Keďže skoro všetky postupy pri evolučnom algoritme sú stochastické, nastáva opakovanie evolučného procesu niekoľko krát začínajúcej z rôznych počiatočných podmienok. Pritom sa výsledné evolúcie štatistický vyhodnotia (priemer, priemerná odchýlka, …).

5.2.1 GA v úlohe obchádzania prekážok a ísť čo najrovnejšie

Na naučenie správania je potrebné definovať genotyp chromozómu (jedinca) a navrhnúť účelovú funkciu.

návrh genotypu

Genotyp jedinca je reprezentovaný ako dvojrozmerné pole. Prvý rozmer je definovaný premennou *vzdialenosť* a druhý rozmer premennou *poloha*. Takto dostávame dvojrozmerné pole v rozmeroch $4x8$. Čo je celkovo 32 polí, ktoré charakterizujú všetky možnosti polohy a vzdialenosti najbližšieho predmetu k robotovi v našom priestore. Z definície nášho problému platí, že pre každú možnosť existuje pravé jedno správne riešenie uhlovej rýchlosti robota, takže každé pole môže nadobúdať jednu z hodnôt -3 až 3.

Je použitá binárna reprezentácia. Preto bolo potrebné jednotlivé hodnoty prekonvertovať. Na binárny zápis siedmych hodnôt sú potrebné tri bity($2^3=8$). Keďže jedno číslo je voľné bola hodnota 0 (rovný pohyb) použitá dvakrát. Počas mutácie zmena jedného bitu môže výslednú hodnotu posunúť o dosť značnú mieru (000 (hodnota: -3) -> 100 (hodnota 0)), čo je pravý opak podstaty mutácie. V genetických algoritmoch sa preto často využíva gray kódovanie genotypu jedinca. Pre gray kód platí, že dva susediace prvky sa líšia maximálne o jeden bit.

000	-3
001	-2
011	-1
010	0
110	0
111	+1
101	+2
100	+3

Tabuľka 1: Gray kód

100	**100**	**101**	**100**	**100**	**000**	**001**	**100**
100	**100**	**100**	**100**	**011**	**101**	**101**	**000**
010	**100**	**111**	**110**	**110**	**101**	**100**	**110**
111	**011**	**011**	**010**	**011**	**000**	**110**	**001**

Tabuľka 2: Genotyp jedného z jedincov v generácií

návrh účelovej funkcie

Na úspešné zvládnutie úlohy robota je potrebné navrhnúť takú účelovú funkciu, ktorá bude najlepšie hodnotiť správanie robota. Náš robot je reaktívny ovládací systém, ktorý nemá žiadne vnútorné stavy, nevie čo sa odohráva v celom prostredí. Je závislý len od svojich senzorov a jeho jediná reakcia na prostredie je pomocou krokového motora. Keďže realizujeme experiment v simulátore je možnosť použitia určitého dozorcu celého prostredia, ktorý by mohol komunikovať s robotom a posielať mu dôležite informácie o zmenách, pozíciach objektov, vlastných súradniciach, atď. V práci som sa ale zameral na robota, ktorý nenadväzuje na žiadne takéto spojenie s dozorcom. A takýto robot by mohol byť úspešný aj v reálnom prostredí s reálnym robotom. A ako bolo spomenuté v statí [2]. Cieľom návrhu účelových funkcií v robotike pre reaktívne roboty je orientovať sa predovšetkým na návrh takých funkcií, ktoré budú preferovať správanie,implicitné a interné. Preto úlohu dozorcu v tomto prípade eliminujeme. Samozrejme sa takto odoberáme o dôležité informácie, ale zase takýmto spôsobom sa snažíme o maximálne využitia len dostupných dát, ktorými robot disponuje. Robot s takouto funkciu je nezávislý od vonkajších systémov.

Pri realizácií návrhu bolo dôležite nezabúdať na podstatu problému a z tohto problému navrhovať deduktívne jednotlivé časti účelovej funkcie.

Problém: Naučiť robota obchádzať prekážky a pritom by sa mal pohybovať čo najrovnejšie.

Robot, ktorý má obchádzať prekážky musí vedieť, že keď je pred ním prekážka je nutné zmeniť smer a zase na druhej strane pokiaľ sa dá nech robí taká akcie aby jeho pohyb bol čo najrovnejší. Za takéto správanie musí byť patrične ohodnotený. Na základe hodnôt predných senzorov vieme zistiť ako blízko je prekážka. Na to, aby úspešne prekážku obchádzal bude najlepšie ohodnotený ak je od nej vzdialený čo najviac. Na to, aby robot šiel čo najrovnejšie potrebujeme vedieť jeho aktuálnu rýchlosť na oboch krokových motoroch. Ak je táto rýchlosť rovnaká, robota ohodnotíme najvyššou hodnotou. V prípade ak sa rozdiel medzi rýchlosťami motorov zväčšuje bude hodnota nižšia. Od robota ešte očakávame, že sa bude pohybovať čo najrýchlejšie, teda čím väčší je súčet rýchlosti jeho motorov tým je ohodnotený vyššou hodnotou.

Takto sme sa dostali k trom zložkám účelovej funkcie. Každú zložku treba ešte vhodne normalizovať .Takáto účelová funkcia je striktne interná, keďže robot pracuje len so svojimi informáciami. Celkový návrh účelovej funkcie bol inšpirovaný z[2].

2 Incremental Evolution of Neural Controller for Robust Obstacle-Avoidance in Khepera ::J.Chavas

"isť čo najrýchlejšie"

$$f_1 = 0{,}5 + (motor[1] + motor[2])/4 * maximalnaRychlost$$

"isť čo najrovnejšie"

$$f_2 = 1 - |(motor[1] - motor[2])|/2 * maximalnaRychlost$$

"vyhýbať sa prekážkam"

$$f_3 1 - \sum_i predneSenzory[i]/4 * maximalnaHodnotaSenzora$$

Každý jedinec štartuje z rovnakého pred pripraveného miesta. Účelová funkcia v čase t je= $f_t = f_1 * f_2 * f_3$. Evaluácia jedinca trvá niekoľko sekúnd, výsledná hodnota účelovej funkcie je sumou všetkých f_t normalizovaná ich počtom. $F = (\sum f_t)/|\mathrm{f}|$ .Po ukončení sa evaluácia opakuje s novým jedincom.

Pre beh GA potrebujeme nastaviť potrebné parametre algoritmu: veľkosť populácie, pravdepodobnosť kríženia a mutácie.

Ukončovacou podmienkou bude skonvergovanie priemernej fitness populácie.

Úlohu obchádzania pri takejto reprezentácií genotypu môžeme naučiť pomocou samostatného GA, použitím vhodným knižníc.

Takýto prístup si vieme ale aj jednoducho predstaviť vložený v bayesovskom programe. Premenné reprezentujeme rovnako len z rozdielom, že premenná senzorov bude diskrétna a jej distribúciu si zadefinujeme formou tabuľky.

Tabuľka bude totožná s genotypom aktuálneho jedinca. Pre každú dvojicu hodnôt senzorových premenných bude existovať práve jedna hodnota premennej aktuátorov. Táto hodnota bude s pravdepodobnosťou 100%.

Identifikácia tabuľky bude prebiehať pomocou GA. Pri testovaní úlohy počas fázy výberu sa s nameraných hodnôt explicitne zadá robotovi hodnota z tabuľky. Fáza výberu sa stáva deterministickou a tým neberieme do úvahy aktuálnu dynamiku prostredia, prípadne zašumené hodnoty senzorov. Robot sa môže dostať do singulárných bodov, v ktorých sa môže zablokovať.

Preto je potrebné zmeniť reprezentáciu a prístup učenia. Evolučné stratégie pracujú s reálnymi vektormi jedincov a pracuje s náhodnosťou selekcie a reprodukcie formou mnohorozmernej distribúcie.

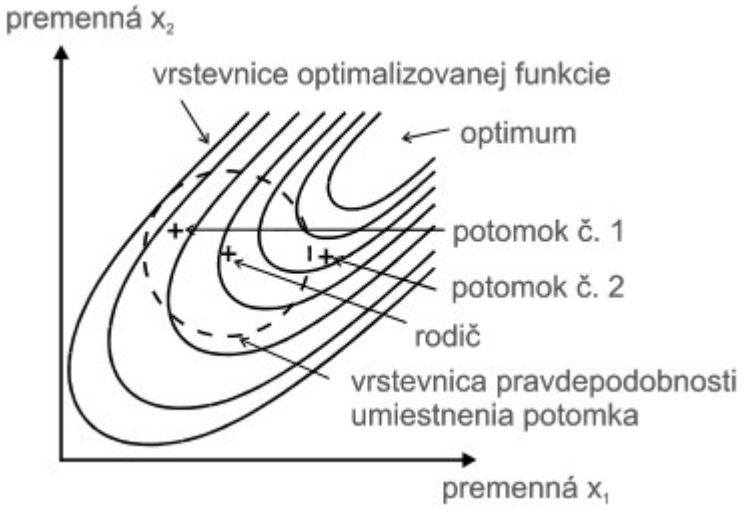
5.3 Evolučné stratégie

Technika evolučných stratégií bola prvýkrát predstavená v roku 1960 a vylepšovaná v sedemdesiatych rokoch Ingom Recherbergom a Hans-Paul Schwefelom. Vychádzajú tiež z všeobecných predstáv prirodzeného výberu pričom o dosť nejasnejšie ako genetický algoritmus. Je to stochastická optimalizačná technika založená na adaptácií a mutácií. Nie je založená na binárnej reprezentácií ako genetický algoritmus ale na vektore reálnych čísiel. Prvé evolučné stratégie využívali len jeden rekombinačný operátor – mutáciu a v populácií sa nachádzal len jeden jedinec. Pričom sa uchovávali súčasne dvaja jedinci rodič a potomok. Je to dvojčlenná evolučná stratégia (1 + 1) - ES.

Základom evolučnej stratégie je jedinec *x*, ktorý je reprezentovaný ako vektor *n* reálnych čísiel

$$X = (x_1, x_2, x_3, x_4, x_5, x_6, \ldots, x_n)$$

Nové riešenie x' vznikne mutovaním pôvodného jedinca *x* kde $N(0, \delta)$ je vektor nezávislých náhodných čísiel s nulovou strednou hodnotou a odchýlkou delta. Hodnoty vzdialenejšie od nuly sa budú vyskytovať s menšou pravdepodobnosťou.



Obrázok 12. Výber dvoch jedincov v ES

Neprerušovaná kružnica okolo rodiča predstavuje okolie pravdepodobných potomkov. Nový jedinec v populácií, potomok, sa vždy vyberá lepší ako rodič. V tomto prípade je lepši potomok č.2, ďalšia generácia bude vyberať potomka z jeho okolia. Takže akceptovanie nového jedinca je

striktne deterministické. Pre riešenie x' musí platiť $f(x') > f(x)$. Ak sa náhodou vyberie horší potomok proces výberu potomka sa opakuje. Takýto prístup je podobný s gradientovou metódou alebo horolezeckým algoritmom.

Evolučné stratégie ďalej rozdeľujeme podľa počtu rodičov, potomkov. Typy evolučných stratégií:

- (1 + 1)-ES : základná technika evolučných stratégií, jeden rodič a jeden potomok, výber sa uskutočňuje s rodičov aj potomkov
- (1 + lambda) – ES : jeden rodič a lambda potomkov, výber sa uskutočňuje s rodičov aj potomkov
- (1 , lambda) – ES : jeden rodič a lambda rodičov pričom výber sa uskutočňuje len z potomkov
- (μ/ρ, λ)- ES - : μ je počet rodičov a z toho je p jedincov, ktorí sa rekombinujú a lambda rodičov, Výber sa uskutočňuje len z potomkov.

Algoritmus evolučných stratégií:

1. nastav populáciu rodičov podľa charakteristiky ES
2. vytvor lambda potomkov tvoriaci novú populáciu, kde každí jedinec tejto novej populácie je tvorený ako:
    1. vyber náhodné p rodičov z populácie P, ak je p = μ potom vyber všetkých jedincov
    2. prekombinuj p vybraných jedincov a vytvor indivíduá x
    3. uprav vnútorné parametre
3. vyber novú populáciu rodičov
4. choď na bod 2 ak nieje dosiahnuté zastavovacie kritérium

5.4 CMA-ES

CMA -ES patrí medzi evolučné stratégie . Je založená na kovariančnej adaptívnej metóde. V tejto stratégií je nový jedinec vyberaný z mnohorozmernej normálnej distribúcie. Vzájomné závislosti medzi premennými sú vyjadrené v kovariančnej matici. Kovariančná adaptívna metóda spočíva vtom, že sa každou generáciou aktualizuje kovariančná matica distribúcie.

Táto metóda sa využíva hlavne na riešenie nelineárnych optimalizačných problémov. Zaužívané prehľadávacie metódy ako kvázi-newtononová metóda alebo gradientová metóda v týchto prípadoch pochybia kvôli nekonvexnosti a nerovnosti prehľadávacieho priestoru. Medzi výhody metódy CMA-ES patrí, že kvôli aktualizovaní kovariančnej matice sa mnohorozmerná distribúcia dokáže prispôsobiť k ťažko kalibrovanému alebo neseparovateľnému problému. Malá veľkosť populácie vo väčšine vedie k rýchlejšej konvergencií a veľká veľkosť populácie zase napomáha k vyhýbaniu sa lokálnych extrémom. Použitím krokového parametru nenastáva predčasná konvergencia populácie.

Keďže jedinec sa vyberá náhodne s mnohorozmerného normálneho rozdelenia pravdepodobnosti a v tejto metóde sa využíva kovariančná matica. V nasledujúcej časti sa zameriam na definovanie týchto pojmov.

Mnohorozmerná normálna distribúcia je generalizovaná jednorozmerná normálna distribúcia Má vlastnosti presne také aké by sme od nej očakávali : je symetrická a maximum je centrovaný okolo jedného bodu.

Mnohorozmerná normálna distribúcia má dva parametre a pritom každý tento parameter obsahuje určité množstvo informácií. Prvý parameter μ obsahuje informáciu o lokácií centra distribúcie a druhý parameter určuje *kovariančná matica,* rozptyl dát a teda šírku tejto distribúcie centrovanej okolo jedného bodu.

<u>stredná hodnota</u>

$$\mu = |E|X_1|, E|X_2|, \ldots, E|X_k||$$

kovariančná matica pre jednu náhodnú premennú

$$\sum = |Cov|X, X^t|| = E[(x_i - \mu)(x_i - \mu)^t] = \sum_X [(x_i - \mu)(x_j - \mu)] \quad i = 1,2,\ldots,k \; ; \; j = 1,2,\ldots,k$$

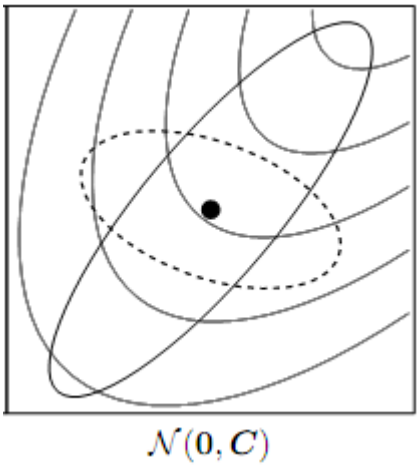
Mnohorozmerná normálne rozdelenie pravdepodobnosti k-rozmerného náhodného vektora

$X = (x_1, x_2, x_3, \ldots, x_k)$ sa označuje ako $X \sim N(\mu, \sum)$

Náhodný vektor X má mnohorozmerné normálne rozdelenie pravdepodobnosti ak spĺňa podmienku, že pre každú lineárnu kombináciu vektorov Y platí že sú normálne distribuované a taktiež platí, že pre konštantný vektor *a* platí že to je jednorozmerné normálne rozdelenie.

$$\Sigma = \begin{bmatrix} \mathrm{E}[(X_1 - \mu_1)(X_1 - \mu_1)] & \mathrm{E}[(X_1 - \mu_1)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_1 - \mu_1)(X_n - \mu_n)] \\ \mathrm{E}[(X_2 - \mu_2)(X_1 - \mu_1)] & \mathrm{E}[(X_2 - \mu_2)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_2 - \mu_2)(X_n - \mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}[(X_n - \mu_n)(X_1 - \mu_1)] & \mathrm{E}[(X_n - \mu_n)(X_2 - \mu_2)] & \cdots & \mathrm{E}[(X_n - \mu_n)(X_n - \mu_n)] \end{bmatrix}.$$

Obrázok 13. Maticová reprezentácia kovariančnej matice



Obrázok 14. Geometrická interpretácia kovariančnej matice je hyperelipsa

Algoritmus CMA-ES

1. nastav lambda
2. nastav počiatočné premenné
3. vyber vzorku lambda jedincov a ohodnoť ich účelovou funkciou
4. utrieď jedincov v populácií
5. ulož pôvodnú strednú hodnotu mnohorozmernej distribúcie *m*
6. aktualizuj *m* k lepšiemu riešeniu
7. aktualizuj kovariančnú maticu
8. aktualizuj veľkosť kroku
9. pokiaľ nieje dosiahnuté zastavovacie kritérium choď na 3
10. vráť *m* alebo x_1 (najlepší jedinec)

Algoritmus spočíva v tom, že sa vyberú nové prípustné riešenia, tie sa potom vzhľadom k ich účelovej funkcie pre usporiadajú a následne sa aktualizujú vnútorné premenné metódy. Algoritmus $(\mu/\mu_w,\lambda)$ patrí medzi najpoužívanejší. Pri výbere μ najlepších jedincov λ rodičov sa používa rekombinačný váhový faktor μ_w .V nasledujúcej časti sa budeme venovať jednotlivým postupom výpočtu algoritmu. Podrobnejšie informácie algoritmu CMA-ES sú dostupné v literatúre The CMA: Evolution Strategy Tutorial od N.Hansena .

1.vzorkovanie

Výber nových jedincov do populácie je generovaný náhodným výberom z n-rozmernej normálnej distribúcie

$$x_k^{(g+1)} \sim m^g + \sigma^g N(0, C^g) \; pre \; k = 1, \ldots, \lambda$$

$N(0, C^g)$ - mnohorozmerné normálne náhodné rozdelenie pravdepodobnosti s nulovou strednou hodnotou a kovariančnou maticou C^g , môžeme si ju prepísať na tvar :

$$N(0, C^g) = (x_i^{(g+1)} - m^g)/\sigma^g = y_i^g \quad pre \; i = 1, \ldots, \lambda$$

$x_k^{(g+1)}$ k - tý potomok (jedinec) z generácie g+1

$m^{(g)}$ – aktuálna stredná hodnota mnohorozmernej distribúcie generácie g

$\sigma^{(g)}$ – veľkosť kroku v generácií g, využíva sa pri výbere jedincov a adaptácií parametrov ako normalizačná konštanta

$C^{(g)}$ – kovariančná matica v generácií g

$\lambda > 2$ je veľkosť populácie, jej hodnota sa väčšinou nastavuje v závislosti od rozmeru distribúcue.

$$\lambda = 4 + 3 * \log N \quad [\text{ P.1 }]$$

N- rozmer distribúcie

Proces vzorkovania je formou mutácie ako pri genetických algoritmoch. Podobný prístup sa používa aj pri genetických algoritmoch s genotypom reálneho vektora. Distribúcia $N(0, C^{(g)})$ tu slúži ako šum .

2.výber jedincov zo vzorky

V závislosti od definície evolučnej stratégie závisí koľko jedincov zo vzorky bude slúžiť na aktualizovanie parametrov stratégie. Počet vybraných jedincov určuje číslo μ . Prednastavená hodnota parametra je rovná polovičke počtu vzorky.

$$\mu = \lambda / 2 \quad [\text{ P.2 }]$$

Všetci jedinci vzorky sa ohodnotia účelovou funkciou $f(x)$ (snažíme sa ju minimalizovať) . Po ich usporiadaní od najlepšieho po najhoršieho si zvolíme prvých μ .

$$f(x_{1:\lambda}^{(g+1)}) \leqslant f(x_{2:\lambda}^{(g+1)}) \leqslant \ldots \leqslant f(x_{\lambda:\lambda}^{(g+1)})$$

$x_{(i:\lambda)}^{(g+1)}$ i - tý najlepší jedinec z x_1 , ... , x_λ

Po výbere nastáva aktualizácia strednej hodnoty, kovariančnej matice a σ -kroku v novej generácií.

3.aktualizácia strednej hodnoty

Nová stredná hodnota v novej generácií g+1 sa vypočíta ako vážený priemer vybraných μ bodov zo vzorky $x_{1,} \ldots . x_n$ spoločne s rekombinačným faktorom.

$$m^{(g+1)} = \sum_{i=1}^{\mu} w_i x_{i:\lambda}^{(g+1)}$$

w_i je pozitívny váhový koeficient, ktorý slúži ako prostriedok rekombinácie. Z definície vyplýva, že najlepšie jedince sú najviac zvýhodnené (najmenej zmutované). Ak nechceme mutovať najlepších jedincov všetky hodnoty w_i nastavíme na rovnakú hodnotu. Výsledný $m^{(g+1)}$ bude priemernou hodnotou všetkých μ_i .

$$\sum_{i=1}^{\mu} w_i = 1, \quad w_1 \geqslant w_2 \geqslant \ldots \geqslant w_\mu \geqslant 0$$

4.aktualizácia kovariančnej matice

Medzi ďalšie postupy patrí aktualizovanie kovariančnej matice. Cieľom je upraviť kovariančnú maticu v závislosti od novovybraných μ jedincov a strednej hodnoty z predchádzajúcej generácie.

$$C_\mu^{(g+1)} = E[(x_i^{(g+1)} - m^g)(x_i^{(g+1)} - m^g)^t]$$

Použitím len vybraných μ bodov na výpočet každej kovariančnej matice by sme stratili dôležité informácie o predchádzajúcich generáciách. Hodnota μ môže byť veľmi malá a tým sa stráca spoľahlivosť predpovede. Preto sa v metóde používa priemerná kovariančná matica z niekoľkých predchádzajúcich generácií. Pri takejto reprezentácií majú všetky kovariančné matice z generácií rovnakú váhu. Zadefinovaním väčšej váhy pre novšie matice sa bude určovať "učiacim parametrom" 0 < c_μ < 1. Týmto parametrom vieme nastaviť ako ďaleko do minulosti chceme zájsť. V metóde CMA-ES je tento proces nazývaný ako aktualizácia o hodnosť μ [3]

$$C^{(g+1)} = (1 - c_\mu) C^{(g)} + c_\mu C_\mu^{(g+1)}$$

Ďalším krokom je aktualizovanie kovariančnej matice o maticu s hodnosť jedna, ktorá je

3 rank μ update

reprezentovaná evolučnou cesty p_c .V tejto fáze deformujeme kovariančnú maticu v smere evolučnej cesty. Evolučná cesta je cesta stredných hodnôt, ktorú stratégia vykoná počas behu niekoľkých generácií. Na výpočet cesty sa používa suma za sebou nasledujúcich stredných hodnôt v generáciách. Pri výpočte cesty sa definuje jej „učiací parameter“ c_c .

$$p_c^{(g+1)}=(1-c_c)\,p_c^{(g)}+\sqrt{c_c(2-c_c)\mu_{eff}}\,\frac{m^{(g+1)}-m(g)}{\sigma^{(g)}}$$

Efektivita μ_{eff} : $\mu_{eff}=(\sum_{i=1}^{\mu} w_i^2)^{-1}$

Použitím týchto dvoch prístupov získame výslednú kovariančnú maticu:

$$C^{(g+1)}=(1-c_1-c_\mu)\,C^{(g)}+c_1\underbrace{p_c^{(g+1)}p_c^{(g+1)T}}_{\text{aktualizácia } 1-\text{hodnosti}}+c_\mu\underbrace{C_\mu^g}_{\text{aktualizácia } \mu-\text{hodnosti}}$$

Maticu je ešte doplnená o ďalší „učiaci parameter“ c_1

5.aktualizácia kroku σ

Po aplikovaní potrebných parametrov, kde sa využíva krokový parameter σ , nasleduje jeho aktualizácia. Pri konštantnej hodnote σ metóda môže pravdepodobne skončít v lokálnom minime. Musíme ju porovnať z *očakávanou* cestou(pri náhodnom výbere $x_{(i:\lambda)}$).

Ak evolučná cesta je väčšia ako *očakávaná* tak sa σ zmenší. Ak evolučná cesta je menšia ako *očakávaná* tak sa σ zväčší.

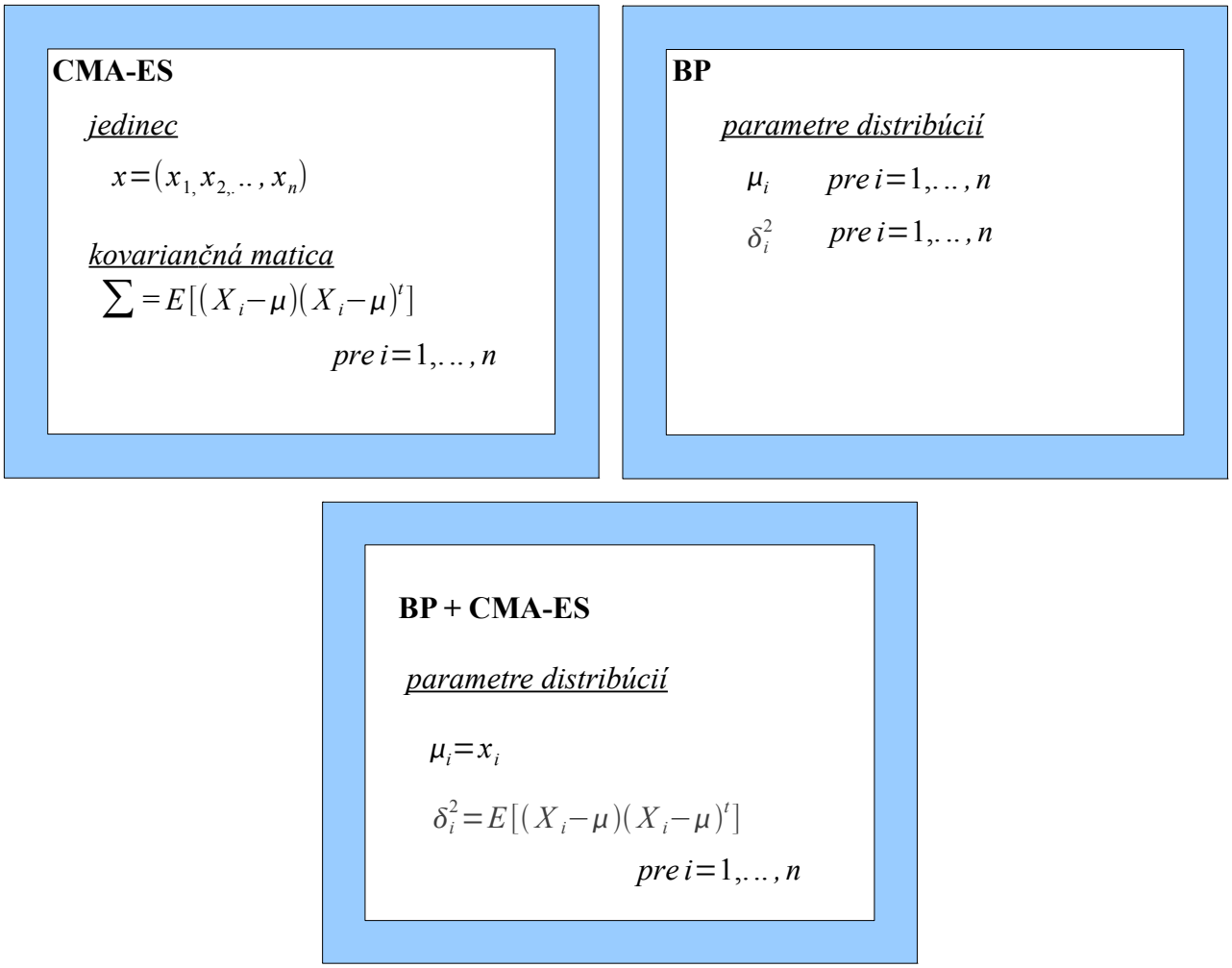
5.4.1 CMA-ES na trénovanie distribúcií Bayesovského programu

Evolučná stratégia CMA-ES využíva mnohorozmernú náhodnú distribúciu ako prehľadávací priestor k nájdeniu optimálneho riešenia (prípadne sup-optimálneho). Postup pri návrhu realizácií bayesovského programu je totožný s pôvodným frameworkom. CMA-ES použijeme na natrénovanie distribúcií v BP. Algoritmus CMA-ES pracuje s jedincami reprezentovanými n-rozmerným vektorom reálnych čísiel. Tieto hodnoty reálnych čísel definujú n stredných hodnôt jednorozmerných distribúcií s gaussovým rozdelením.

Distribúcie, ktoré vstupujú do evolúcie a potrebujeme ich natrénovať sú tie, ktoré po

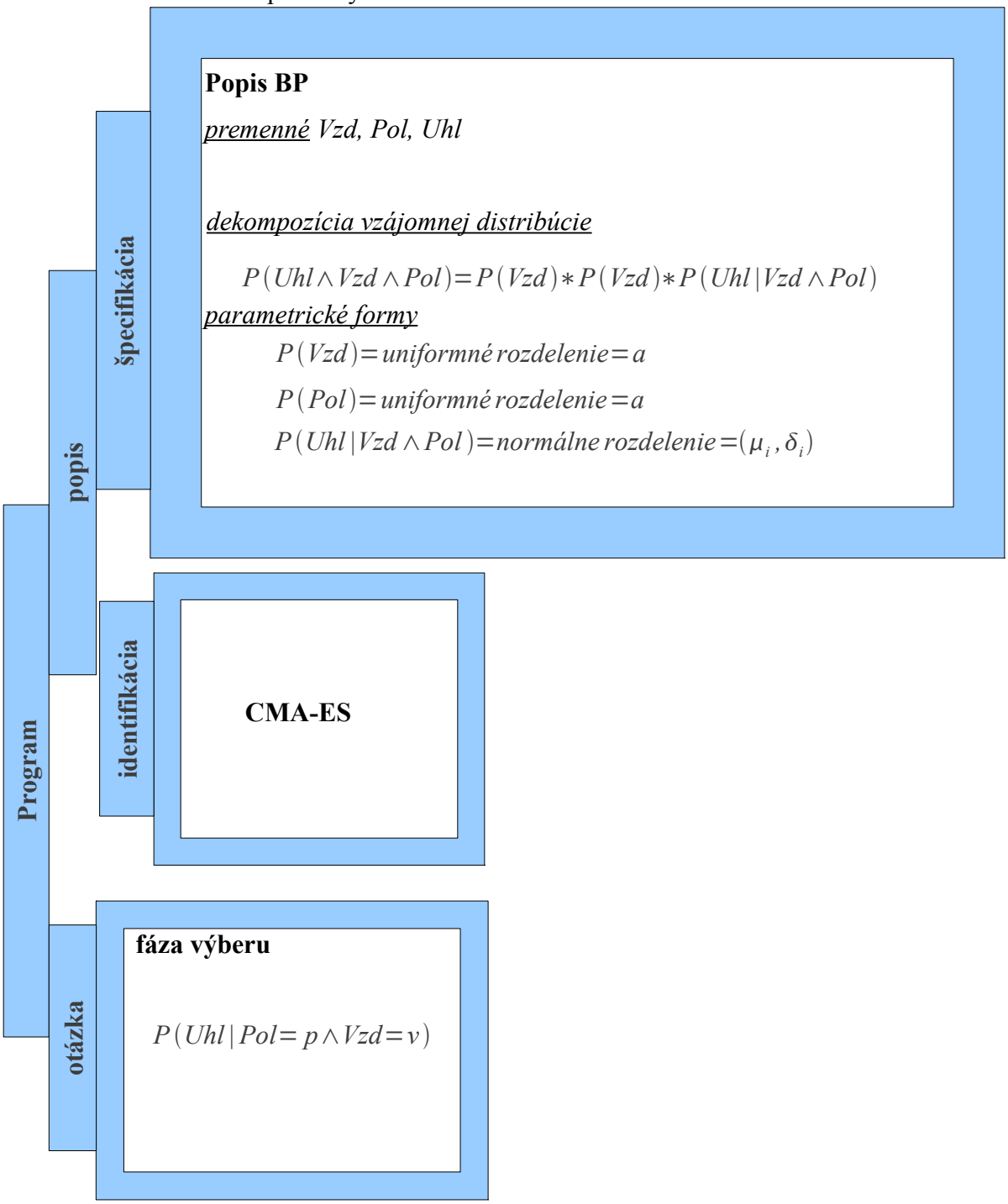
dekompozícií nemajú a priori definovanú distribúciu a ich parametrická forma je gaussovským rozdelením. Na úspešné natrénovanie týchto distribúcií potrebujeme „uhádnuť“ dvojicu parametrov.

Stredné hodnoty týchto distribúcií získame z reprezentácie jedinca. Keďže tieto distribúcie sú parametrizované gaussovským rozdelením pre celistvosť potrebujeme vedieť nielen ich strednú hodnotu μ ale aj rozptyl δ^2 . Ten dokážeme získať z kovariančnej matice mnohorozmernej normálnej distribúcie. Z definície kovariančnej matice platí, že na jej diagonále sa nachádza práve n rozptylov.



Obrázok 15. Parametre distribúcií pomocou CMA-ES

Definícia BP bude identická s pôvodným BP.



Obrázok 16. Bayesovský program + CMA-ES

V tomto bayesovskom programe už nepotrebujeme experimentálne dáta. Keďže učenie je

evolvované. Účelová funkcia bola použitá taka istá ako v experimente s GA, keďže šlo o totožné zadanie úlohy. Len evaluácia prebiehala inač, keďže sme už súčasťou bayesovského programu. O identifikáciu voľných premenných sa nám postará evolučná metóda CMA-ES.

Jedinec je reprezentovaný 32-rozmerným vektorom reálnych čísiel.

Pred štartom CMA-ES nastavíme parametre metódy. Podľa prednastavených parametrov definujeme $\lambda = 4 + 3 * log32 = 8$ [P.1] . $\mu = 8/2 = 4$ [P.2]

Na začiatku identifikačnej fázy sa spusti evolučná stratégia CMA-ES. Na začiatku evolúcie nastavíme náhodnú hodnotu strednej hodnoty a kovariančnej matice .

Po selekcií λ rodičov, spustíme náš program na evaluácia.

evaluácia jedinca

1. z CMA-ES máme jedinca definovaného 32 rozmerným vektorom U a kovariančnú maticu C
2. z kovariančnej matice získame z jej diagonály 32 rozmerný vektor D
3. zadefinujeme všetky parametre gaussovskej distribúcie z parametrami (μ_i, δ_i^2) pomocou vektora U a D
4. zavoláme metódu na počítanie účelovej funkcie a ohodnotíme výsledné správanie jedinca

Každého jedinca položíme na prednastavené miesto a spustíme ho s konštantnou rýchlosťou. Robot na základe svojich hodnôt senzorov a naučených distribúcií pošle hodnotu aktuátorom a ten svoj pohyb vykoná. Toto sa vykonáva určitý časový úsek.

metóda pre účelovú funkciu

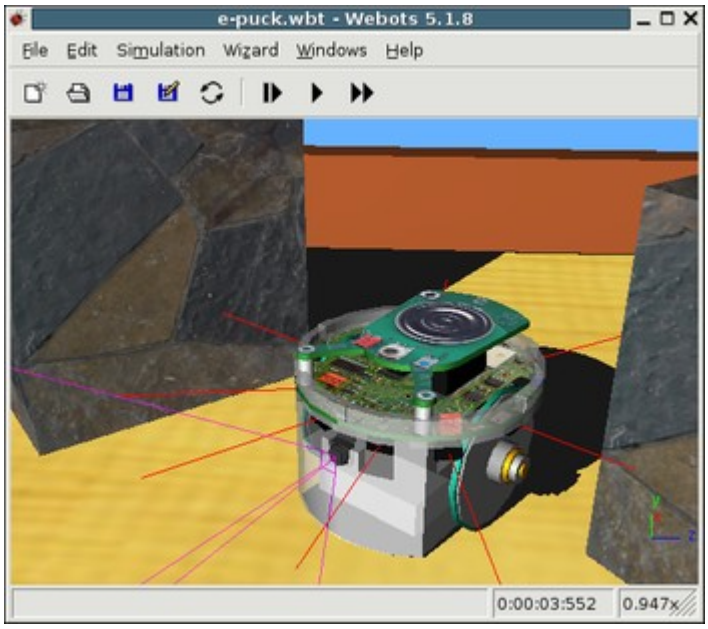
1. nameraj aktuálne hodnoty premenných $Vzd = v_i$ a $Pol = p_i$
2. z distribúcie $P(Uhl \mid Vzd = v_i \wedge Pol = p_i)$ sa stochastický vyberie hodnota premennej $Uhl = u_i$
3. hodnota u_i sa pošle robotovi a ten vykoná akciu

Po ukončení evalvácií všetkých λ jedincov, metóda CMA-ES pokračuje vo svojom výpočte. Vyberie sa najlepších μ jedincov a nastane adaptácia strednej hodnoty a kovariančnej matice mnohorozmernej distribúcie.

Po adaptácií parametrov metódy nastáva výber jedincov do novej generácie a proces sa opakuje. Ukončenie procesu nastáva po určitom počte generácií.

6.Výsledky experimentov

V tejto práci realizujeme všetky experimenty v simulátore Webots. Simulátor Webots je profesionálny robotický simulačný softvér. Simulácia je realizovaná vo virtuálnom 3D prostredí s emuláciou fyzikálnych procesov. V 3D virtuálnom prostredí užívateľ dokáže pridávať základné útvary ako kocka, stena, guľa,... a rôzne ich upravovať. Samozrejmosťou sú aj aktívne mobilné roboty s rozdielnymi možnosťami pohybu(kolesové, kráčajúce alebo lietajúce). Každý takýto robot je vybavený rôznymi senzormi (kamera, dotykové, infračervené,...) a aktuátormi (serva, kolesá).
V experimentoch sme používali robotický systém E-PUCK.



Obrázok 17. Model robota e-puck v simulátore Webots

prevzate z: http://en.wikibooks.org/wiki/Cyberbotics%27_Robot_Curriculum/E-puck_and_Webots

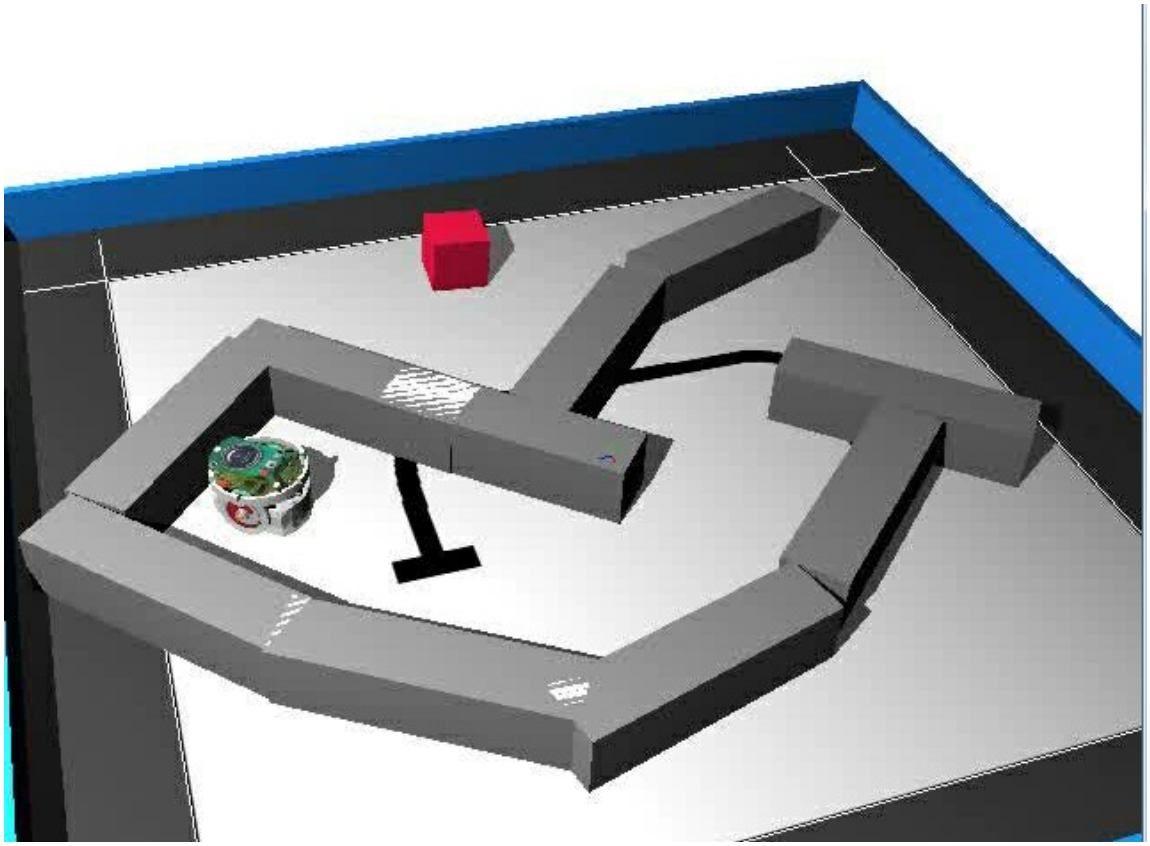
Robot E-puck bol navrhnutý Dr. Francesco Montana a Michael Bonani v roku 2006. Slúži predovšetkým na vzdelávacie účely, ale využíva sa aj vo vedeckom smere. Pre svoj jednoduchý dizajn a robustnosť systému patrí medzi obľúbené roboty.

Je vybavený 8 infračervenými senzormi, ktorými dokáže zachytiť objekt do vzdialenosti 6 centimetrov, 3 senzormi na podvozku, ktoré slúžia na sledovanie čiary a VGA kamerou. Pohybuje sa maximálne 6cm/sekundu. Jeho priemer je 70 milimetrov a je poháňaný dvomi krokovými motormi.

V simulátore bežia všetky akcie robota v reálnom čase, pričom pri behu dlhých evolučných

výpočtoch je možnosť zapnúť zrýchlený režim, kde v závislosti od aktívnych procesov (textový výstup,zapnutá kamera, atď) sa dá dosiahnuť až 100-200 násobné zrýchlenie času.

Robot je položený v priestore obkolesený kvádrovými objektami.



Obrázok 18. Prostredie robota

Simulátor poskytuje samostatné programovacie prostredie. Program bol napísaný v jazyku C/C++. Hlavnou triedou programu je *MyRobot*.

```
//actuators
void setspeed(int vr);

//methods
void ga();
void cmaes();

//bayes program
void compute_variables();
double draw_question();
double rand_normal(double mean, double stdvev);

float compute(float **gen);
void run();
```

- *void MyRobot::setspeed* – v závislosti od hodnoty premennej aktuátorov (vr) nastavíme rýchlosti dvom krokovým motorom. V CMA-ES je hodnota reálne číslo, preto ju zaokruhlíme na *integer*.
- *MyRobot:: ga()* - spustíme genetický algoritmus.
- *MyRobot::cmaes()* - spustíme evolučnú stratégiu CMA.
- *MyRobot::compute_variables()* - vypočítame hodnoty senzorových premenných z aktuálnych nameraných hodnôt senzorov.
- *MyRobot:: draw_question()* - z hodnôt senzorových premenných vyberieme hodnotu pre premennú aktuátorov. V závislosti od typu použitej metódy vyberieme hodnotu pomocou distribúcie z *rand_normal* (CMA-ES), alebo pomocou tabuľky(GA).
- *MyRobot::rand_normal* – z gaussovskej distribúcie vyberáme pseudo-náhodne použitím polárnej formy Box-Mullerovej transformácie.
- *MyRobot::compute* – v tejto metóde ohodnocujeme aktuálneho jedinca.
- *MyRobot::run* - podľa zvolenej metódy sa zavolá *ga()* alebo *cmaes()*.

Štruktúra *Rrobot* definuje potrebné parametre Bayesovského programu.

```
#define COUNT1 4
#define COUNT2 8
#define COUNT3 5

struct Params
{
  double mean;
  double variance;
};

struct Rrobot
{
  int         Premenna1;
  int         Premenna2;
  int              Vrot;
  struct Params      distr[COUNT1][COUNT2];
  int                 tab [COUNT1][COUNT2];
};
```

- *int Variable1* (hodnota premennej Poloha), *int Variable2* (hodnota premennej Vzdialenosť), *int Vrot*(zaokrúhlená hodnota premennej Uhlová rýchlosť)
- *struct Params distr[COUNT1][COUNT2]* – definovanie 32- gaussovských distribúcií pomocou ich parametrov (μ – *double mean*, δ^2 – *double variance*)
- *int tab[COUNT1][COUNT2]- tabuľka pre GA*

Pri prvom spustení sa zavolá metóda *run()* a podľa zvolenej techniky (GA al. CMA-ES) sa začína evolučný proces. Pre každého jedinca sa opakuje počítanie fitness v metóde *compute()* určitý

časový úsek. V experimentoch sme si nastavili 1500 krokov (1 krok = 320ms). Za tento čas je robot schopný dostať sa na koniec chodby pri správnom riadení.

6.1 Genetický algoritmus

Na implementáciu genetického algoritmu som použil knižnicu GALib[4] vyvíjanú Matthew Wallom s prostriedkami a finančnou podporou MIT[5]. Najnovšia verzia 2.4.7[6], ktorú som aj používal, bola naposledy aktualizovaná v roku 2007.Knižnica je napísaná v jazyku C++ .

Genetický algoritmus má svoje interné parametre, ktoré musíme pred behom programu nastaviť:

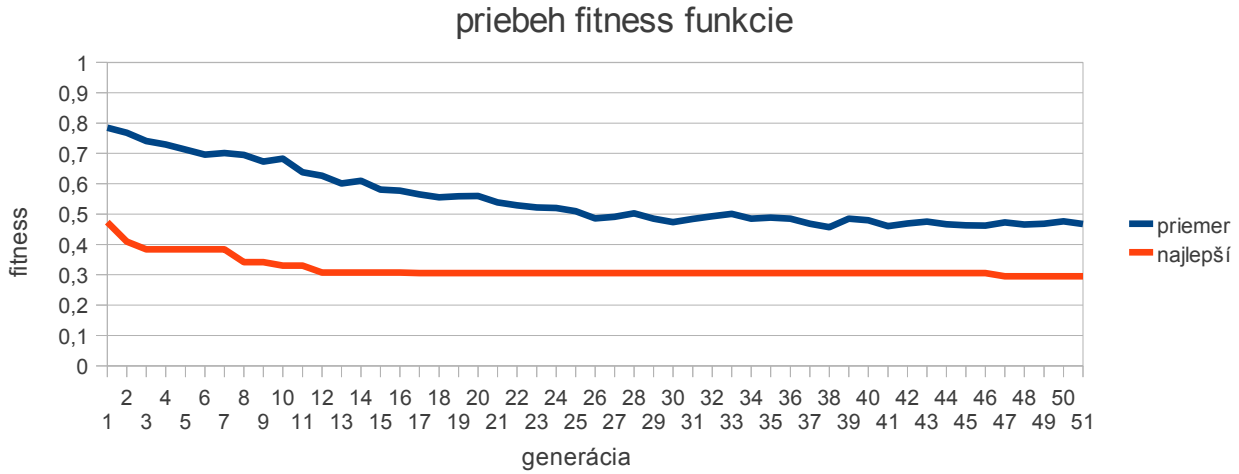
pjed - populácia jedincov v každej generácií, *pmut* - pravdepodobnosť mutácie

pkriz- pravdepodobnosť kríženia.

Zastavovacie kritériom GA bude počet generácií – *pgen.*

Po každej generácií sa do textového súbora ukladajú informácie o najlepšej fitness jedinca a priemerná fitness generácie. Po ukončení evolúcie sa najlepší jedinec uloží. Pri ďalšom spustení je možne načítať jeho genóm.

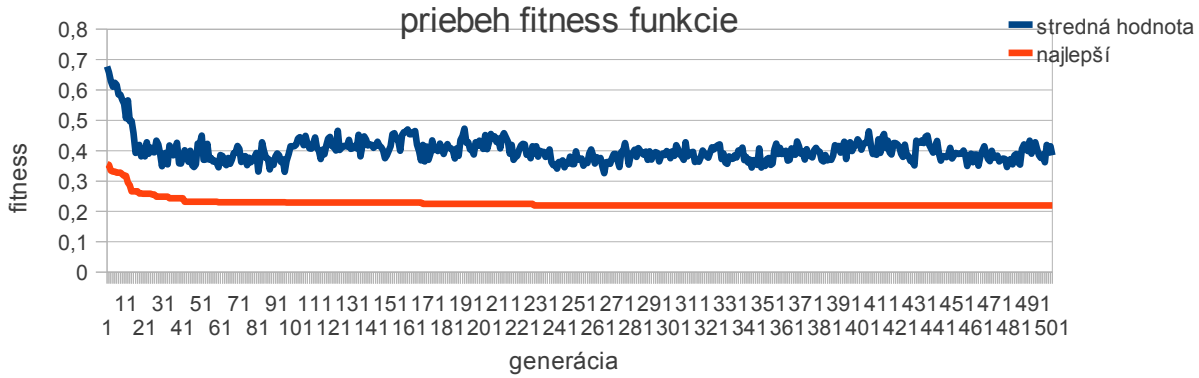
Takmer vo všetkých príkladoch,ktoré sme používali, najvýraznejšie zmeny nastávali v prvých 50-100 generácií, pokiaľ sme nenastavovali extrémne hodnoty pravdepodobnosti mutácie a kríženia. Príkladom môže byť GA s populáciou 50 a počtom 100 jedincov na populáciu.



Obrázok 19. Grafický priebeh fitness v GA

4 A C++ Library of Genetic Algorithm Components
5 Massachusetts Institute of Technology
6 http://lancet.mit.edu/ga/dist/

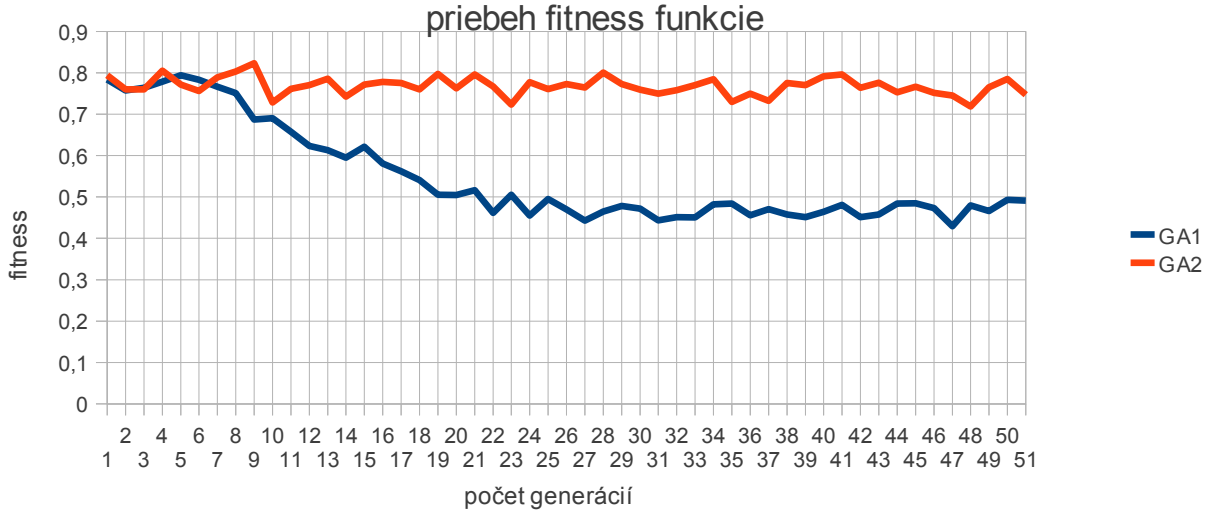
Pri dlhých behoch genetického algoritmu priemerná fitness skonvergovala a neočakávali sme od nej nejaké výrazne zmeny a výsledné správanie najlepšieho jedinca bolo dostačujúce. Ako vidieť aj na priloženom videu č.1. Parametre takého GA: *pjed: 50, pgen: 500, pmut: 0.002%,pkriz: 0.8%*



Obrázok 20. Grafický priebeh fitness v GA

Pravdepodobnosť mutácie vplýva značnou mierou pri obchádzaní prekážky robota. Vysoká pravdepodobnosť prispieva k častej zmene hodnôt premennej aktuátora a preto priemerná fitness stagnuje už hneď v prvých generáciach. Noví jedinci tak vznikajú často náhodne bez možnosti vytvorenia si štruktúry pre jednotlivé možnosti obchádzania. V situácií, kde robot má pred sebou prekážku na ľavej strane sa musí naučiť otáčať motormi doprava určitý časový úsek. Na vykonanie takéhoto pohybu potrebujeme naučiť väčšiu množinu dvojíc hodnôt senzorových premenných.

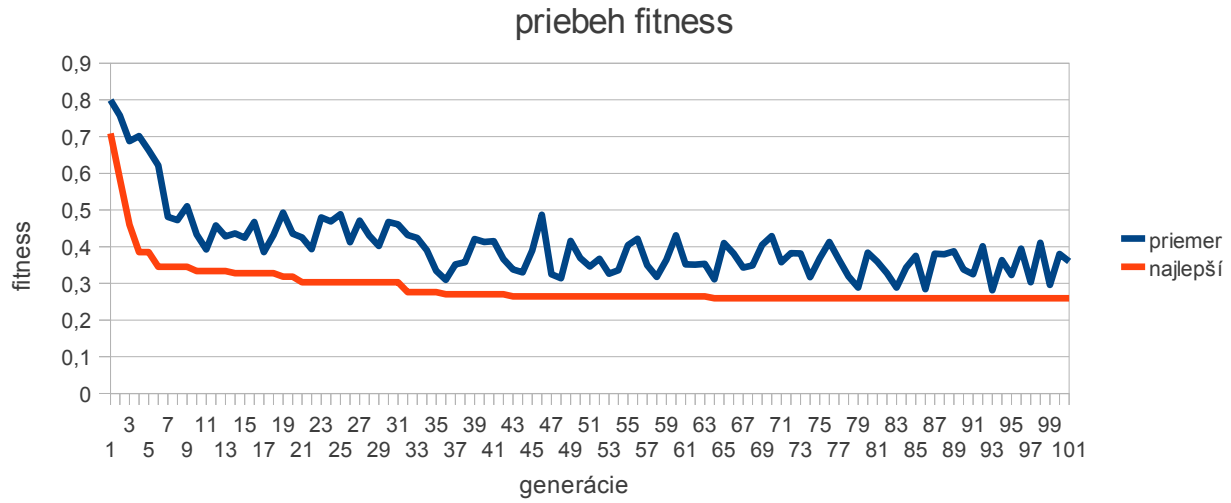
Pre porovnanie sme použili dva genetické algoritmy s rovnakým počtom jedincov nastavený na 20 a počet generácií 50. Mutácia bola nastavená v jednom prípade(GA2) na 0.002% a v druhom na 0,8% (GA1).



Obrázok 21. Grafický priebeh fitness v GA

Pravdepodobnosť kríženie sa nastavuje vysoká, keďže požadujeme od evolúcie aby z generácie na generáciu nastávali časté prepojenie dvoch jedincov. Predovšetkým pri ranných fázach, keďže tu jedinci často narážajú do stien alebo sa zasekávajú do rohov a majú naučené len niektoré možnosti pohybu. Vysoká pravdepodobnosť spôsobuje v neskorých generáciách pomalejší priebeh konvergencie, keďže jednotliví jedinci majú lepšiu fitness.

Vhodnými nastaveniami parametrov dokážeme evolúciu vykonať úspešne. Preto počet generácií nastavíme na 100, pravdepodobnosť mutácie na 0.002% a pravdepodobnosť kríženia na 0.8%. Pri definovaní malého počtu jedincov(8) v generácií je evolúcia úspešná ako vidieť v grafe. Počet selekcií je 800.



Obrázok 22. Grafický priebeh fitness v GA

V nameraných experimentoch sme dokázali naučiť správanie pomocou základného evolučného algoritmu.

V prílohe sa nachádzajú videa ďalších najlepších jedincov s GA.

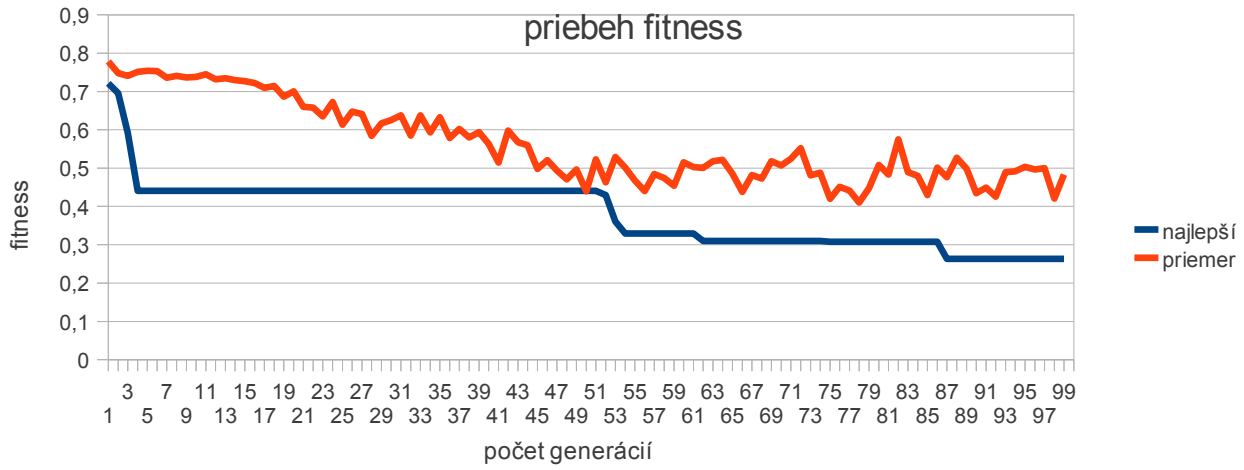
	video č.2	video č.3
počet gen	50	150
počet jed	100	50
p. mutácie	0,002	0,002
p kríženia	0,08	0,08

Tabuľka 3. Videá s GA

6.2 CMA-ES v Bayesovskom programe

Na implementáciu CMA-ES som použil knižnicu Eskit[7] , ktorá je pod licenciou MIT License. Najnovšia verzia bola aktualizovaná 25.augusta 2010. Knižnica je napísaná v jazyku C.

Metóda CMA-ES má parametre, ktoré pred jej spúšťaním potrebujeme nastaviť. Autori tejto metódy odporúčajú ponechať parametre prednastavené. V prvej úlohe som preto ponechal $\lambda=8$ a $\mu=4$. Počet všetkých selekcií bolo $100*8=800$.



Obrázok 23. Grafický priebeh fitness v CMA-ES

Ako vidieť z grafu porovnaním s GA(s "dobrými parametrami") na obrázku č.19 obidve metódy dokázali úspešne zvladnuť úlohu s podobnými parametrami.

Algoritmus dokázal úspešne naevolvovať parametre distribúcie:

stredná hodnota

vzdialenosť\poloha	0	1	2	3	4	5	6	7
0	-0,925649	-2,45669	-1,53628	1,70762	2,25599	-0,762734	0,504252	-1.2595
1	-3,61811	-1,22672	-2,97271	-1,68062	3,47278	3,4142	3,3229	-2.1678
2	-2,62859	-2,55171	-3,61062	-1,29844	-2,59521	-2,21608	-2,79299	2.41579
3	-0,641222	-0,716355	-1,45965	0,679125	-1,98857	1,78617	3,20426	-0.802193

Tabuľka 4. Parametre distribúcie = stredné hodnoty

variancia

vzdialenosť\poloha	0	1	2	3	4	5	6	7
0	0.901963	0,860117	0,898724	0,899161	0,902789	0,950197	0,929109	0,933373
1	1.1228	0,951681	0,911756	0,905825	0,988585	0,917765	0,916434	0,892435
2	0,915901	0,895834	1,18026	0,914882	0,953039	0,969761	1,06229	1,01741
3	1,07305	1,00069	0,960344	0,988954	0,959717	0,89896	0,982426	0,958152

Tabuľka 5. Parametre distribúcie = variancie

7 http://www.marmakoide.org/code/eskit/releases

Ako vidíme variancia dosahovala veľmi malé hodnoty, takže jedinec bol skoro vo všetkých situáciach "istý" akú akciu ma vykonať. Kovariančná matica mala na začiatku prednastavené nulové hodnoty.

Inicializovaním vlastných hodnôt na začiatku experimentu pri dlhých behoh malo za následok tieto variancie vyraznejšie pomeniť. V prípade keď sme nastavili diagonálu kovariančnej matice na hodnoty 2. Dostali sme výsledky, ktore sa líšili veľmi minimálne.

variancia								
vzdialenost\poloha	0	1	2	3	4	5	6	7
0	1.3931	1.28505	1.28589	1.36566	1.56066	1.42385	1.7037	1.48305
1	2.03568	2.09798	2.0888	1.87468	1.77872	1.85409	2.287	2.59686
2	2.01453	1.60378	1.28344	1.67566	1.56115	1.56169	1.82886	2.54135
3	1.60497	1.69591	1.96013	1.49376	1.52544	1.56431	1.7896	2.33545

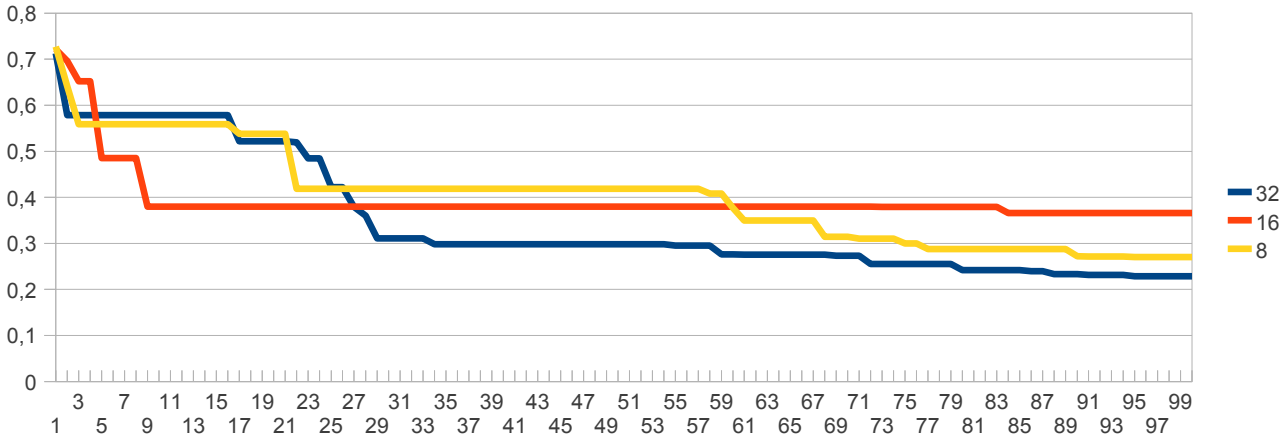
Tabuľka 6. Parametre distribúcie = variancie pri inicializovani 2

V prípade nastavení hodnotami 3 sme dostavali výsledky, ktoré zaznamenali zmenu ale rozdelenie týchto hodnôt bolo skôr náhodne Priebeh tohto algoritmu je znázornený na videu č.4.

variancia								
vzdialenost\poloha	0	1	2	3	4	5	6	7
0	1.96769	5.01797	4.36645	0.967323	3.3929	1.07258	1.32866	1.5297
1	2.48877	0.952711	1.1572	0.461629	5.07735	3.03233	1.04929	2.34949
2	1.40261	1.2236	0.639011	1.58131	2.16415	4.54318	4.07559	1.16236
3	0.67676	2.03892	1.87255	1.90802	1.13172	1.88373	1.84778	1.44453

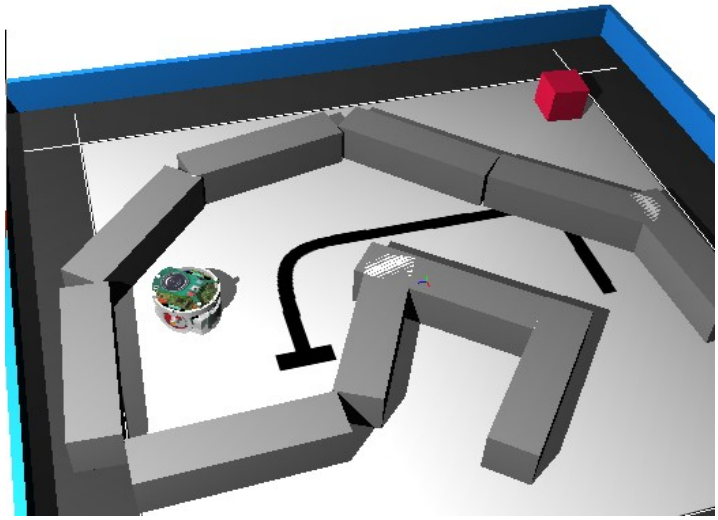
Tabuľka 7. Parametre distribúcie = variancie pri inicializovani 3

Selekcia v CMA-ES je náhodným výberom. Do selekcie v GA vstupujú všetci jedinci a preto ich priemerná hodnota je viacej reprezentatívna ako v prípade CMA-ES.Preto som v ďalšej časti realizoval 3 experimenty s rozdielnými λ (32, 16, 8) hodnotami počas behu 100 generácií, Hodnoty na grafe reprezentujú najlepších jedincov.



Obrázok 24. Grafický priebeh fitness CMAES s tromi najlepšími jedincami

Medzi veľkosťami populácie nieje zaznamený veľký rozdiel v priebehu fitness najlepšieho jedinca.V práci sme ďalej použili naučené správanie z predchádzajúceho prostrediana v novom inom prostredí.



Obrázok 25. Nové prostredie robota

Robot dokázal toto nove prostredie zvládnuť čo môžeme vidieť aj na priloženom videu č.5.

7.Záver

V úvode práce som sa venoval základnými definiciami inteligentného robota. Robot v tejto práci predstavoval reaktívny systém, ktorý nekomunikuje s dozorcom .Jeho aktuatory reagujú len na základe hodnôt vlastných senzorov. V práci som sa zameral na pravdepodobnostný model riadený Bayesovským programom.

Bayesovský program je definovaný náhodnými premennymi a distribúciami. Pomocou nich vieme reprezentovať vnemy a akcie robota.Distribúcie parametrizujeme známymi funkciami. Cieľom tejto paradigmy je voľné parametre natrénovať

V mojej bakalarskej práci som realizoval jednoduchý experiment bayesovského programu na obchádzanie prekážky a tlačenie objektu. Správanie, ktoré sa robot naučil získal z experimentálných dát pomocou riadenia robota joystickom určitý časový úsek.

Počas experimentu bolo potrebné niekoľkokrát robota premiestniť a znova učiť. Preto sme v tejto práci predstavili rozšírenie bayesovského programu o evolúciu. Evolúcia nám slúžila na natrénovanie potrebných parametrov.Úspešnosť celeho experimentu v evolučných algoritmoch závisí od definovania účelovej funkcie a definovania ich vnútorných parametrov.

V prvej úlohe sme použili základný genetický algoritmus s binárnou reprezentáciou na obchádzanie prekážky a pričom sa mal robot pohybovať čo najrovnejšie. Binárna reprezentácia mala za následok odstranenie pravdepodobnostných distribúcií, keďže pre jednotlivé výstupy v programe existovalo práve jedno riešenie.

Plne využitie Bayesovského programu s evolúciou nám poskytuje evolučná stratégia CMA-ES., ktorého jedinec je vektor reálných čisiel a pracuje s mnohorzmerným normálnym rozdelením pravdepodobnosti.

Experimenty Bayesovského programu boli uskutočňované v simulátore, kde som jednotlivé metódy porovnal a výsledky zdokumentoval. Použité algoritmy boli úspešné a tým som overil ich funkčnosť pri takomto zadaní úlohy. V prípade metódy CMA-ES sme dokázali generalizovať svoje naučene informácie na novom prostredí.

Literatúra

[1]LEBELTEL a spol. : Bayesian Robot Programming.
Vydavateľstvo:Kluwer Academic Publishers , 2004.

[2]ARKIN, RONALD C.: Behavior–Based Robotics
Vydavateľstvo: The MIT Press , 1998.

[3]HANSEN, N.: The CMA Evolution Strategy: A Tutorial[online]
Dostupné na internete: <http://www.bionik.tu-berlin.de/user/niko/cmatutorial.pdf>

[4]KVASNIČKA, V.- POSPÍCHAL, J.- TIŇO, P.: Evolučné Algoritmy.
Vydavateľstvo STU: Bratislava 2000.

[5]KOIKE, C. : Obstacle Avoidance and Proscriptive Bayesian Programming[online]
Dostupné na internete: <*hal.archives-ouvertes.fr/hal-00019258/en/*>2006

[6]NOLFI, S.- FLOREANO, D. : Evolutionary Robotics.
Vydavateľstvo :Massachusetts Institute of Technology ISBN 0-262-147070-5

[7]CHAVAS, J. - CORNE, CH. - HORVAI, P. - KODJABACHIAN, J. - MEYER J.A. :
Incremental Evolution of Neural Network Controller for Robust Obstacle-Avoidance in Khepera
Dostupné na internete:<*www.springerlink.com/index/6252727122646656.pdf*>

[8]PETROVIC, P. : Pravdepodobnostne riadenie mobilneho robota trénovane evolucnym algoritmom, Kognicia a umely zivot, Praha, 2008.

[9]THRUN, S.- BURGARD, W.-FOX, D. : Probabilistic robotics
Vydavateľstvo: The MIT Press , 2005.