UNIVERZITA KONŠTANTÍNA FILOZOFA V NITRE

FAKULTA PRÍRODNÝCH VIED

# ALGORITMY NA KOOPERÁCIU AUTONÓMNYCH ROBOTOV

BAKALÁRSKA PRÁCA

Nitra 2012 Richard UHRINČAŤ

# UNIVERZITA KONŠTANTÍNA FILOZOFA V NITRE

## FAKULTA PRÍRODNÝCH VIED


# ALGORITMY NA KOOPERÁCIU AUTONÓMNYCH ROBOTOV

### BAKALÁRSKA PRÁCA

| | |
|---|---|
| Študijný program: | Aplikovaná informatika |
| Školiace pracovisko: | Katedra informatiky |
| Školiteľ: | PaedDr. Jozef Kapusta, PhD. |


Nitra 2012 Richard Uhrinčať

# ZADANIE PRÁCE

Cieľom tejto bakalárskej práce je prakticky vytvoriť algoritmus vyhľadávania pomocou metód správania sa sociálneho hmyzu, popísať možnosti a prakticky ukázať komunikáciu autonómnych robotov stavebnice Lego Mindstorms.

# ABSTRAKT

UHRINČAŤ, Richard: Algoritmy na kooperáciu autonómnych robotov. [Bakalárska práca]. Univerzita Konštantína Filozofa v Nitre. Fakulta prírodných vied. Školiteľ: PaedDr. Jozef Kapusta, PhD. Stupeň odbornej kvalifikácie: Bakalár odboru Aplikovaná informatika.. Nitra: FPV 2011. 34 s.

Cieľom tejto bakalárskej práce je popísať algoritmy pre vyhľadávanie inšpirované tzv. sociálnym hmyzom, ktoré sa dajú implementovať v dnešnej robotike. Práca sa zaoberá sociálnym hmyzom, jeho správaním, ako aj algoritmami, ktoré používa na vykonávanie rozličných úloh ako napríklad zber potravy. Práca taktiež podrobne rozoberá niektoré z týchto algoritmov používané mravcami alebo včelami a presnejšie približuje ich funkciu, zápis a implementáciu. Ďalším z cieľov tejto práce je tieto algoritmy a ich funkcie naprogramovať a potom aj prakticky ukázať za pomoci stavebnice Lego Mindstorms. Na tieto účely bol vytvorený program zvaný NXT Včely, ktorý názorne simuluje algoritmy na vyhľadávanie, komunikáciu a koordináciu včiel medonosných.

Kľúčové slová: NXT, sociálny hmyz, algoritmy, vyhľadávanie, komunikácia

## ABSTRACT

UHRINČAŤ, Richard: Algorithms for cooperation of robots. [Bachelor Thesis]. Constantine the Philosopher University in Nitra. Faculty of Natural Sciences. Supervisor: PaedDr. Jozef Kapusta, PhD. Degree of Qualification: Bachelor of Applied Informatics, FNS 2011. 34 p.

The main goal of this Bachelor Thesis is to describe search algorithms used by social insects that are used in modern robotics. The thesis focuses on social insects, its behaviour as well as algorithms it uses for various things, like foraging. The thesis also concentrates in detail on a select few of these algorithms which are used by ants or bees and in detail describes their function, mathematical formula and implementation. The secondary goal of this Thesis is to program theses algorithms and their function and then practically show them by using the Lego Mindstorms kit. For these purposes, the NXT Bees program was made, to visually simulate algorithms for foraging, communication and navigation of honey bees.



Keywords: NXT, social insects, algorithms, communication, Lego Mindstorms

# OBSAH

# ÚVOD

V súčasnej dobe narastá vo svete techniky záujem o robotiku a o možnosti jej využitia v praxi na vykonávanie rôznych úloh. Keďže sa jedná o mladý vedný odbor, nie sú podrobne preskúmané všetky jeho aspekty a jedným z nich sú aj koordinačné a komunikačné algoritmy, ktoré boli inšpirované sociálnym hmyzom.

V tejto práci popíšeme algoritmy pre vyhľadávanie inšpirované tzv. sociálnym hmyzom, ktoré sa dajú implementovať v dnešnej robotike. Ako názorný príklad implementácie týchto algoritmov zostavíme robotov zo stavebnice Lego Mindstorms s programovateľnou kockou NXT a zariadením Bluetooth, ktorý budú autonómne simulovať správanie sa sociálneho hmyzu, alebo aspoň niektoré jeho aspekty. Textová časť práce sa v jadre zaoberá koordinačnými a komunikačnými algoritmami sociálneho hmyzu, z ktorých vybrané názorne predvedieme za pomoci NXT hardwaru. V prvej sekcii priblížime NXT kocku a hardvér, ktorý tvorí celého robota a rozoberieme tematiku sociálneho hmyzu a ním implementovaných algoritmov.

V druhej sekcii bližšie popíšeme jednotlivé algoritmy, ich používanie a aj následné možnosti využitia v praxi.

Poslednou časťou celej práce je program NXT ***Včely,*** ktorý je napísaný v BricxCC a už názorne zobrazuje, ako sa roboty budú správať.

Pri programovaní týchto algoritmov sme použili všestranný open source IDE softvér na programovanie Lego kociek od autora Johna Hansena, zvaný Bricx CC (Bricx Command Center), ktorý je napísaný tak, že sa v ňom dajú písať programy pre programovateľnú RCX (a odnedávna aj pre NXT) kocku vo väčšine ňou podporovaných jazykoch. (Bricx History, 2012).

# 1 ANALÝZA SÚČASNÉHO STAVU

V dnešnej dobe sa problematika robotiky ako takej stala veľmi populárnou témou vo svete techniky a informatiky. Veľa odborníkov, ale aj žurnalistov z danej oblasti predpovedá, že už je len otázkou času, kedy vznikne prvá „skutočná“ umelá inteligencia a roboty ako také sa stanú súčasťou nášho každodenného života. Avšak ako každý iný stroj, aj roboty musia bežať pod určitým programom, teda usporiadaným zhlukom algoritmov, ktoré simulujú určité správanie, alebo činnosť. Algoritmy, ktoré toto správanie simulujú podľa živočíšneho sveta sa nazývajú aj bio-algoritmy. Inšpiráciu na tvorbu niektorých z týchto bio-algoritmov ľudia berú priamo zo sveta hmyzu, presnejšie, sociálneho hmyzu. Na prvý pohľad sa to nemusí zdať, ale sociálny hmyz ako včely, mravce, alebo termity už miléniá vo svojich kolóniách využívajú princípy, ktoré sa dnešná veda snaží zreprodukovať do svojich výtvorov. Ale kým toto je len teória a predpovede o možno nie tak vzdialenej budúcnosti, momentálne je robotika, ako vedný odbor, viac menej v plienkach. Dôvodom je nie len zložitosť danej tematiky, ale aj jej implementácia v reálnom svete. Ale ako všade na svete, ani pri robotike neplatí, že je to striktne vedný odbor určený len pre odborníkov. Dôkazom tohto tvrdenia je napríklad výtvor spoločnosti Lego a ich stavebnica Lego Mindstorms NXT.

## 1.1 BIO-ALGORITMY A ROBOTICKÉ SYSTÉMY

Väčšina práce v kooperačnej mobilnej robotike začala po uvedení novej formy kontroly robotického ovládania, ktorá fungovala na základe napodobňovania správania sa živých organizmov. Táto forma mala veľký vplyv pri výskume kooperačnej robotiky, pretože tento vzor popisujúci algoritmy, ktoré simulujú správanie, má korene v biologických inšpiráciách. Veľa výskumníkov z oboru kooperačnej robotiky považuje pozorovanie sociálnych charakteristík hmyzu a zvierat za poučné a aplikujú následné objavy a pozorovania do dizajnu viac robotových systémov.

Najznámejším využitím týchto poznatkov je používanie jednoduchých pravidiel na lokálnu kontrolu, ktoré praktizujú rôzne biologické spoločenstvá – napríklad mravce, včely a vtáky – na vývoj kooperujúcich robotických systémov s podobným správaním. Práca v tomto odvetví demonštrovala schopnosť viac robotových tímov tvoriť stáda, rozptyľovať sa, zhlukovať sa, zháňať potravu a nasledovať stopu. Aplikácia dynamiky

ekosystémov bola taktiež aplikovaná na vývoj viac robotových tímov, ktoré demonštrovali vznikajúcu kooperáciu ako výsledok sebeckého správania sa. Do istej miery, kooperácia vyšších zvierat, ako vlčích svoriek, priniesla zdokonalenia v kooperačnej kontrole. Významné štúdia v systémoch lovec-korisť sa tiež vyskytli ale iba v simuláciách. Súťaživosť vo viac robotových systémoch, taká aká je vo vyšších zvieratách, vrátane ľudí, je tiež skúmaná v doménach ako robotický futbal.

Tieto oblasti biologickej inšpirácie a ich využiteľnosť v robotických tímoch vyzerajú byť celkom dobre pochopené. Novšie, menej chápané biologické témy relevantné s týmto, zahŕňajú využitie imitácie vyšších zvierat na naučenie sa nového správania a fyzickej prepojenosti, ktorú demonštruje hmyz, ako napríklad mravce, na umožnenie kolektívnej navigácie cez zložitý terén. (Parker, 2012)

## 1.2 LEGO MINDSTORMS

V roku 1998 Lego uviedlo nový produkt zvaný: „Lego Mindstorms Robotic Invention Kit“, čo sa dá voľne preložiť ako „Vývojarský balíček robotov Lego Mindstorms“, neskôr zvaný aj RIS (Robotic Invention System). Celý produkt pozostával z približne 717 kusov vrátane LEGO kociek, motorčekov, prevodov, senzorov a „RCX Kocky“, ktorá mala v sebe zabudovaný mikroprocesor (Mindell, 2000).

V roku 2006 bol RIS nahradený Mindstorms NXT, ktorý je nielen sofistikovanejší, ale obsahuje aj viac senzorov a má zdokonalené už existujúce senzory, nehovoriac o ďalších vychytávkach, ktoré LEGO pridalo (A Review and History of the Next Generation of Robotics, NXT, 2012). V dnešnej dobe je Mindstorms jeden z najpopulárnejších produktov od Lega a to aj vďaka tomu, že je to tak hračka, ako aj veľmi sofistikovaná robotická stavebnica, ktorú dokážu využiť nielen deti na hranie, ale má aj praktickejšie využitie. Ďalším pozitívom je, že ako väčšina LEGO produktov, aj Mindstroms si zachoval všestrannú kompatibilitu so staršími produktmi od spoločnosti LEGO, ako Techinc a System čím sa možnosti využitia tejto stavebnice exponenciálne zvyšujú. Ku komerčnej sade je dostupné aj ikonografické programovacie prostredie NXT-G, ktoré svoju funkčnosť prevzalo po nástroji zvanom Labview od spoločnosti National Instruments a obohatilo ju o zjednodušené orientovanie a narábanie, čím zaručilo, že tento program je schopné používať aj dieťa na základnej škole.

## 1.3 NXT – OBSAH KOMERČNEJ SADY

Komerčne dostupná základná sada NXT obsahuje nasledujúce prvky:

- 619 LEGO Technic súčiastok (vrátane prevodov, ozubených koliesok, kociek, atď.),
- 1 x NXT programovateľná kocka,
- 1x USB kábel na prepojenie kocky s PC,
- CD s programovacím jazykom NXT-G,
- 7 káblov na prepájanie NXT kocky so senzormi,
- 2 x dotykový senzor,
- 1 x svetelný senzor,
- 1 x ultrazvukový senzor slúžiaci na meranie vzdialenosti,
- 1x svetelný senzor schopný rozlišovať intenzitu svetla ako aj farby,
- 3 x servomotor so zabudovaným senzorom otáčok

Obr. č. 1 NXT kocka so zapojenými komponentmi (LEGO, 2011)

Ku komerčne dostupnej sade sa dajú ešte voľne dokúpiť ďalšie doplnky, ako sú infračervený senzor, gyroskop, merač akcelerácie, zvukový senzor, silnejšie motory atď.

## 1.4 NXT HARDWARE TROCHU BLIŽŠIE

Hardvérová špecifikácia NXT programovateľnej kocky je nasledovná:

**Processor**: Atmel® 32-bit ARM® processor, AT91SAM7S256
- 256 KB FLASH
- 64 KB RAM
- 48 MHz

**Co-processor**: Atmel® 8-bit AVR processor, ATmega48
- 4 KB FLASH
- 512 Byte RAM
- 8 MHz

**Bluetooth**: CSR BlueCoreTM 4 v2.0 +EDR System
- Podpora Serial Port Profile (SPP)
- Interná pamäť 47 KByte RAM
- Externá pamäť 8 MBit FLASH
- 26 MHz

**USB 2.0**: vysokorýchlostný port (12 Mbit/s)

**4 vstupné porty**: 6-žilové rozhranie podporujúce digitálny aj analógový vstup
- 1 vysokorýchlostný port, IEC 61158 Type 4/EN 50170 compliant

**3 výstupné porty**: 6-žilové rozhranie podporujúce výstup so senzorov

**Display**: 100 x 64 pixel LCD čierno-biely display
- Zobrazovacia veľkosť: 26 X 40.6 mm

**Reproduktorový výstupový kanál s 8-bitovou kvalitou**
- Podporuje samplovacie rozhranie 2-16 KHz

**4-tlačidlové ovládanie**: Gumené tlačidlá

**Napájanie**: 6xAA batérie
- Odporúčané sú alkalické
- Nabíjateľná Lithium-Ion battery 1400 mAH je dostupná k sade

**Konektor**: 6-žilový štandardný RJ12 konektor s pravým usporiadaním ( LEGO® MINDSTORMS®, 2011).

# 2 BLUETOOTH KOMUNIKÁCIA

Bluetooth je IEEE štandard pre komunikačné protokoly na malé vzdialenosti (cca 10m, i keď sú známe aj s väčším dosahom). Delí sa na 3 triedy, pričom zariadenia 1. triedy sú vysoko- výkonové (100 mW) zariadenia s operačnou vzdialenosťou do 100m. Zariadenia 2. triedy (2,5 mW) majú operačnú vzdialenosť do 10m a zariadenia 3. triedy sú nízko- výkonové (1 mW) zariadenia s dosahom cca 1m. Bluetooth používa krátke rádiovlny, pričom využíva rádio technológiu FHSS („frequency-hopping spread spectrum“), čo je metóda kolísavej (skákajúcej frekvencie), ktorá rozdelí posielané dáta až do 79 pásiem, v rozpätí od 2,402 GHz do 2,480 GHz, čo je voľné univerzálne rozpätie ISM, („industrial, scientific and medical“) rádiových vĺn. Bluetooth je založený na Master-Slave hierarchii, kde master môže byť prepojený až so siedmymi slave zariadeniami. Výmena paketov je riadená hodinami master – zariadenia, ktoré tikajú raz za 312,5 μs. Jeden komunikačný slot predstavujú dve tiknutia (625 μs), kde správy môžu mať dĺžky 1, 3 a 5 slotov. Master vždy začína vysielať v čase párnych slotov a odpovede od slave - zariadenia dostane vždy v čase nepárnych slotov. (Lucistnik, 2011)

# 3 ÚVOD DO SOCIÁLNEHO HMYZU

Hmyz (*Insecta)* je najpočetnejšou a najúspešnejšou živočíšnou skupinou na Zemi, tvorí až 80% známych druhov živočíchov.

Hmyz sa podľa spôsobu života delí na solitárny a sociálny. V prípade solitárneho hmyzu jedince žijú osamote, združujú sa len v období párenia, alebo pri zdrojoch potravy. Navzájom si nepomáhajú a hlbšie nekomunikujú.

Za sociálny hmyz sa považuje hmyz, ktorý sa združuje v kolóniách – v spoločnom hniezde, kde sú všetci členovia potomkami kráľovnej. Dochádza tu k deľbe práce, výchove nových jedincov, spoločnému získavaniu potravy. Tento hmyz patrí medzi indivíduá vyššieho rádu, čo znamená, že príslušníci toho istého druhu majú rôznu anatómiu – sú prispôsobení na špecifické úlohy a teda zadelení do kást. Rozdeľujeme tri hlavné typy kást; kráľovnú, samce, robotnice a niekoľko vedľajších napríklad vojaci. Štruktúru a súdržnosť kolónie, podobne ako vzájomnú komunikáciu medzi jedincami všetkých kást zabezpečujú hmatové a chemické podnety (feromóny). Medzi sociálny hmyz patria mravce, termity, včely a osy (Husár a Sámel,2010).

## CIEĽ BAKALÁRSKEJ PRÁCE

Cieľom práce je prakticky vytvoriť algoritmus hľadania pomocou metód správania sa sociálneho hmyzu, popísať možnosti a prakticky ukázať komunikáciu autonómnych robotov stavebnice Lego Mindstorms.

# 2 METÓDY (POSTUP) RIEŠENIA PROBLÉMU

## 2.1 KOMPLEXNOSŤ A SEBA-ORGANIZÁCIA

Fyzika, biológia, počítačové, alebo humanitné vedy poskytujú veľa príkladov na systémy, kedy globálne správanie je výsledkom interakcie homogénnych a heterogénnych entít. Naším účelom je namodelovať a analyzovať sociálne správanie inšpirované týmito prírodnými systémami. K tomu všetkému chceme aj zvýrazniť esenciálnu rolu priestoru a ako sú priestorové interakcie katalyzátorom vznikajúcich vlastností, ktoré sa objavujú pri formovaní týchto komplexných systémov.

Seba-organizácia, alebo vlastná organizácia je pojem často používaný v prírodných systémoch. Táto práca popisuje niektoré z týchto systémov, presnejšie tie z hmyzích spoločenstiev, ako sú včely, mravce, alebo termity.

## 2.2 Zhluková inteligencia (SWARM INTELLIGENCE)

Algoritmy zhlukovej inteligencie (Geraldo a Jing, 2012; Bonabeau et al.,1999; Kennedy a Eberhart, 2001) majú korene vo viacerých aspektoch vývoja počítačových vied: od evolúcie sieťových architektúr, ktoré viedli k decentralizovanému počítaniu, až po evolúciu softwareového inžinierstva, ktoré viedlo k rozloženej forme umelej inteligencie.

Zhluková inteligencia je založená na očakávanej interakcii viacerých decentralizovaných riešiteľov. Pri tomto prístupe nie sú žiadni hlavní (master) riešitelia, čím sa problém delí na menšie časti a následne sa zozbierajú pod-riešenia každého sluhu (slave). Inými slovami, pri algoritmoch zhlukovej inteligencie je výpočet decentralizovaný a globálne riešenie vzniká z distribuovaných interakcií riešiteľov.

Najznámejšie metódy zhlukovej inteligencie sú „časticová zhluková optimalizácia“ (PSO – Particle swarm optimization) (Kennedy a Eberhart, 1995) a „mravčie systémy“ (Bonabeau et al., 1999). V PSO sa rozdelení riešitelia pohybujú s virtuálnymi časticami. Tieto častice sa kolektívne pohybujú vo výslednom priestore. Kolektívne rozloženie týchto častíc je inšpirované systémom C.Reynolds Boidsa (Reynolds, 1987), ktorý sa snaží simulovať kolektívne správanie rybích škôl, alebo vtáčích kŕdľov. Virtuálne častice PSO sa medzi sebou vzájomne ovplyvňujú výmenou informácií, aby našli najlepšie riešenie nad výsledným priestorom.

V mravčích systémoch sú rozloženými riešiteľmi mravce pohybujúce sa v danom prostredí. Vzniknuté riešenia sú archivované cez koncept, zvaný stigmergia (nepriama komunikácia medzi jednotlivcami v kolóniách sociálneho hmyzu.). Tento koncept je založený na systéme nepriamej komunikácie, pre ktorú, ako výpomoc, slúži okolie. Je inšpirovaný správaním mravcov, ktoré na zemi zanechávajú feromónovú stopu, aby mohli komunikovať so svojimi kolegami.

# 2.3 MRAVČIE SYSTÉMY

### 2.3.1 Seba-organizácia v prírodných mravčích systémoch

Už roky prirodzené mravčie správanie fascinovalo výskumníkov komplexných systémov svojou kapacitou na seba-organizáciu vo viacerých aspektoch.

Výskumníci z oboru počítačových vied používajú správanie mravčích kolónií na vývoj bio-inšpirovaných metód a algoritmov. Na dosiahnutie tohto cieľa je potrebných len pár esenciálnych aspektov mravčieho správania, ktoré vedú k seba-organizačným procesom, ktoré musia byť pochopené a následne popísané matematickými vzorcami.

Vytváranie aplikácií takýchto umelých mravčích koloniálnych systémov začína byť aplikované v dnešnej dobe, presnejšie pre problémy v okruhu optimalizácie grafov, alokácií úloh, alebo klastrovania.

### 2.3.2 Optimalizácia mravčej kolónie

Optimalizácia mravčej kolónie (Bonabeau et al, 1999) bola pôvodne navrhnutá na riešenie známeho problému Putujúceho obchodníka. Tento problém spočíva v nájdení najkratšej cesty, ktorá spája $N$ prepojených miest vo váženom grafe, pričom do každého mesta je povolený len jeden vstup. Problém je popísaný grafom kde uzli $C_i$ sú mestá a vážené okraje $D_{ij}$ sú vzdialenosti medzi mestami. $T_{ij}(t)$ sú feromónové kvantity vylúčené mravcami na okraji (i,j).

Keď je mravec v meste *i,* vyráta si, aká je pravdepodobnosť, že pôjde do ešte nenavštíveného mesta *j* za pomoci vzorca:

$$P_{ij}^{k}(t) = \begin{cases} \dfrac{(T_{ij}(t))^{\alpha}\left(\dfrac{1}{D_{ij}}\right)^{\beta}}{\sum\limits_{l \in J_k(t)} (T_{il}(t))^{\alpha}\left(\dfrac{1}{D_{il}}\right)^{\beta}} & if \quad j \in J_k(t) \\ 0 & if \quad j \notin J_k(t) \end{cases}$$

Kde:

- $J_k$ je množina miest, ktoré ešte mravec $k$ nenavštívil;
- $\alpha$ a $\beta$ umožňujú kontrolu relatívnej dôležitosti 2 kvantít ovplyvňujúcich mravčie správanie: množstvo feromónovej stopy ($T_{ij}$) a vzdialenosť ($D_{ij}$).

Keď mravec nájde dobré riešenie na cyklenie medzi mestami, vylúči na všetky okraje cyklu feromóny, proporcionálne obrátené k dĺžke cyklu ($L_k$):

$$\delta T_{ij}^{k}(t) = \begin{cases} \dfrac{Q}{L_k} \\ 0 \end{cases}$$

- ak (i,j) je okraj cyklu niekde inde;

- Q je konštanta

Na každom okraji (i,j) je feromónová stopa upravená od kroku t po krok t+1, pridaním všetkých príspevkov každého mravca k predošlému množstvu:

$$T_{ij}(t+1) = \rho T_{ij}(t) + {}^{m}\Sigma_{k=1}\, \delta T^{k}{}_{ij}(t)$$

Kde $\rho$ je faktor vyparovania, ktorý umožňuje, že niektoré prvotné cesty/riešenia môžu byť nahradené inými/lepšími.

### 2.3.3 Mravčie klastrovanie

Cieľom klastrovania je klasifikácia rôznych vzorov a skupín, bez dohľadu nejakej kontrolnej entity. Klastrovanie je v podstate zoraďovanie určitej sady dát do skupín, rozdelených podľa jednej, alebo viacerých vlastností daných dát. (Mohamed a Sivakumar, 2010).

Mravce a iný sociálny hmyz rieši distribúciu klastrov kooperatívne. Už od začiatku agregácie nejakého objektu sa vytvárajú okolo neho malé klastre, ktoré konajú nezávisle na základe stigmergie. Daný mechanizmus zvyšuje počet klastrov ako plynie čas.

Tento algoritmus zabezpečuje, aby sa veľké množstvo autonómnych mravcov, pohybujúcich sa náhodným smerom v prostredí s objektmi, dokázalo zoskupiť do klastra.

V každom kroku sa mravec nachádza v jednom z troch stavov:

1. Mravec sa pohybuje bez toho, aby niečo niesol a na nič nenarazí. V tomto prípade sa bude naďalej pohybovať náhodne.

2. Mravec sa pohybuje bez toho, aby niečo niesol a narazí na objekt. Mravec môže teraz tento objekt vziať. Pravdepodobnosť, že mravec daný objekt zoberie je:

$$P_p = \left(\frac{k_1}{k_1 + f}\right)^2$$

- f je hodnota korešpondujúca k počtu vnímaných objektov v mravcovom okolí
- $k_1$ ja vchod do mraveniska, čo robí z toho pravdepodobnosť medzi 0 a 1, podľa mravcovej relatívnej pozície s $k_1$

3. Mravec sa hýbe a nesie objekt. Mravec môže daný objekt nechať na zemi. Pravdepodobnosť, že mravec nechá nesený objekt na zemi je:

$$P_d = \left(\frac{f}{k_2 + f}\right)^2$$

-$k_2$ je ďalší vchod (Bertelle et al, 2010).

# 2.4 VČELIE ALGORITMY

### 2.4.1 Úvod do včelích algoritmov

V tejto časti si priblížime nový výpočtový algoritmus inšpirovaný sociálnym správaním včiel medonosných. Pozostáva z náborovej a navigačnej stratégie, pričom náborová stratégia slúži na odovzdávanie už nadobudnutých vyhľadávacích skúseností ostatným členom kolónie, zatiaľ čo navigačná stratégia slúži na navigáciu neznámym prostredím. Tento algoritmus nepoužíva feromóny ani na nábor, ani na navigáciu, ako je tomu u mravčích algoritmov. V mravčích algoritmoch chvíľu trvá, kým sa sformujú cesty, ale vďaka povahe včelieho správania tento čas dokážeme skrátiť. Tento algoritmus aplikujeme v doméne hľadania potravy a porovnávame ho empiricky s algoritmom založeným na feromónoch ako je „*optimalizácia mravčej kolónie*“.

V skratke, mravce vytvárajú feromónovú stopu na cestách, ktorými sa uberajú. Využitím týchto stôp sú schopný navigovať smerom k hniezdu, alebo potrave. Na nábor

používajú mravce nepriamu stratégiu akumulácie týchto stôp. Ak je stopa dosť silná, naláka viac mravcov a tie ju využijú na to, aby sa dostali na miesto určenia. Toto je známe aj ako autokatalytický proces; čím viac mravce idú po nejakej stope, tým viac sa pre nich stáva atraktívnejšou. Najpreferovanejšie sú kratšie cesty.

Algoritmy založené na ne-feromónovej báze sú inšpirované (hlavne) včelím správaním a nepoužívajú feromóny na navigáciu v neznámom prostredí. Namiesto toho používajú na navigáciu stratégiu zvanú „cestná integrácia“ (**PI** – Path Integration). Včely sú schopné opakovane vysielať svoju momentálnu pozíciu na základe ich predošlej trajektórie. To má za následok, že sa na svoju začiatočnú pozíciu dokážu vrátiť priamou cestou a nie vracať sa po ceste, ktorou išli (Lambrinos et al, 2000; M¨uller a Wehner, 1988). Na nábor včely používajú priamu metódu tanca v úli. Tento tanec obsahuje vzdialenosť a smer k cieľovej destinácii (von Frisch, 1967).

I keď sa včelie a mravčie spôsoby hľadania potravy výrazne líšia, oba druhy dokážu túto úlohu efektívne riešiť.

Algoritmy na ne-feromónovej báze sú ale o dosť menej skúmané a ich výskum začal len nedávno. Napríklad, (Nakrani a Tovey, 2004; Chong et al, 2006; Teodorovic a M. Dell’ Orco., 2006) všetky doterajšie algoritmy inšpirované včelami, ktoré poskytujú riešenia na riešenie rôznych problémov implementovaním včelieho náborového správania. V Lambrinos et al., (2000) sú navigačné vlastnosti včiel preskúmané a aplikované na robotovi. Avšak tieto algoritmy využívajú len jeden aspekt včelieho správania, navigáciu, alebo nábor. Ako také sú to stále otvorené a dôležité otázky. Po prvé, náborové a navigačné algoritmy sú momentálne študované len separátne; spojitý algoritmus je nepreskúmaná zem. Po druhé, keďže spojitý algoritmus ešte neexistuje, porovnávajúce štúdia ešte neboli podniknuté. V práci sa chceme pozrieť, či náš ne-feromónový algoritmus na zber potravy je lepší, ako feromónový. Presnejšie, či cesty vznikajú rýchlejšie s ne-feromónovým algoritmom. Porovnávajúce štúdium by sa muselo zamerať na efektívnosť, náväznosť a adaptabilitu týchto algoritmov.

Táto práca zahŕňa oba problémy. Najprv máme nový algoritmus na ne-feromónovej báze, ktorý implementuje obe včelie stratégie, aj navigačnú aj náborovú. Po druhé sme si vytvorili virtuálne prostredie nazvané „BeeHive“ (včelí úľ), v ktorom sa dajú algoritmy na zber potravy kompletne porovnať. Využívajúc BeeHive sme schopný porovnať feromónové a ne-feromónové algorimy.

### 2.4.2 Biologický profil

Správanie sa včiel medonosných pri zbere potravy sa delí na dve časti: náborové správanie a navigačné správanie. Aby nalákali členov kolónie na zber potravy, včely medonosné informujú svojich spolu-robotníkov o vzdialenosti a smere tejto potravy trasúcim sa tancom na vertikálnych hrebeňoch úľa (von Frisch, 1967). Tento tanec (včelí jazyk) pozostáva zo série meniacich sa ľavo a pravo točivých slučiek prerušovaných časťami, kedy včela vrtí svojím zadočkom zo strany na stranu. Trvanie vrtiacej fázy určuje vzdialenosť k potrave a uhol medzi slnkom a osou vrtenia na vertikálnom hrebeni reprezentuje azimutálny uhol medzi slnkom a smerom, ktorým sa má regrút vydať, aby našiel potravu (von Frisch, 1967; Michelsen et al, 1992; Dyer, 2002). Tento oznam o zdroji potravy môže byť prevzatý aj inými členmi kolónie. Rozhodovací mechanizmus podľa ktorého včely preberajú tieto oznamované lokácie potravy nie je úplne známy. Predpokladá sa, že nábor medzi včelami je vždy funkciou kvality zdroja potravy (Camazine a Sneyd., 1991).

Ako už bolo spomenuté, rôzne druhy sociálneho hmyzu využívajú ne-feromónovú navigáciu, ktorá zväčša pozostáva z „integrácie cesty“ (PI – Path Integration), čo je opakované aktualizovanie vektora integrovaním všetkých smerových uhlov a všetkých prejdených vzdialeností (Lambrinos et al, 2000). PI vektor reprezentuje hmyzie vedomosti o smere a vzdialenosti k cieľu. Na zostavenie PI vektora hmyz nepoužíva matematické sčítavanie vektorov, ako ľudia, ale vyrátateľne jednoduchý odhad (M¨uller a Wehner, 1988). Využitím tohto odhadu je hmyz schopný sa priamo vrátiť k cieľovej destinácii. Presnejšie, ak nie je cesta blokovaná, hmyz dokáže tento problém riešiť optimálne. Avšak, keď je na ceste prekážka, hmyz sa musí spoliehať na iné stratégie, ako je napríklad navigácia podľa orientačných bodov (Collett et al., 2003; Collett T. a Collett M., 2004). Samozrejme, že včely vedia lietať, a teda sa môžu rozhodnúť danú prekážku jednoducho obletieť, ale aj keď je cesta neblokovaná, včely využívajú navigáciu podľa orientačných bodov na minimalizovanie chýb PI vektora. Orientačné body delia celú cestu na segmenty a každý orientačný bod má vlastný PI vektor. Ďalej v práci sa budeme na PI vektory smerujúce naspäť do hniezda odkazovať pojmom „domáce vektory“ (Home Vector- **HV**). PI sa využíva ako pri objavovaní, tak aj pri zužitkovávaní. Počas objavovania, hmyz neustále aktualizuje svoj HV, avšak toto nie je súčasťou objavovacej stratégie. Počas zužitkovania, hmyz aktualizuje HV aj PI vektor indikujúci potravu, a využívajú tieto vektory ako navádzanie k cieľovej lokácií.

### 2.4.3 Modelovanie včelieho správania

Ako bolo naznačené už skôr, včelie správanie pri zbere potravy pozostáva z dvoch typov správania: náborové a navigačné. V kontraste s inými modelmi (Nakrani a Tovey, 2004; Chong et al., 2006; Teodorovic a Dell' Orco, 2006), náš model spája oba.

Náborové správanie je namodelované analogicky so včelím tancom. Umelé včely zdieľajú informácie, keď sú spolu v úli (domovská lokácia každej z umelých včiel). Vždy, keď je umelá včela v úli, hľadá hocijakú iný umelú včelu, čo má už nejaké skúsenosti s hľadaním. Keď ju nájde, rozhodne sa, či zužitkuje tieto nadobudnuté skúsenosti, pričom zužitkovanie znamená skopírovanie PI vektora (smer získaný tancom). Avšak umelá včela sa môže aj rozhodnúť využiť vlastné skúsenosti, ak také už nadobudla. Biologické včely využívajú (ešte) neobjavený mechanizmus na adoptovanie poskytovaných PI vektorov, ktorý pravdepodobne súvisí s kvalitou zdroja (Camazine a Sneyd, 1991), avšak náš model neberie do úvahy kvalitu zdroja, keďže tento parameter nemal žiadny dopad na naše porovnávania. Presnejšie, agenti preferujú PI vektory, ktoré indikujú zdroj potravy na trase kratšej od úľa, ako ich momentálny vektor.

Navigačné správanie v modeli buď využíva predošlé skúsenosti, alebo necháva umelé včely, aby sa pohybovali náhodným smerom. Využívanie predošlých skúseností je riadené PI vektorom, ktorý umelé včely môžu získať „nasledovaním" tanca v úli, alebo z ich vlastných, predošlých skúseností.

Algoritmus 1, ktorý veľmi blízko pripomína ACO (Dorigo a St¨utzle, 2000), zahŕňa aj navigačné aj náborové správanie a pozostáva z 3 funkcií.

**Algoritmus 1** Algoritmus na ne-feromónovej báze.

```
1: SpavujVčeliuAktivitu()
2: VýpočítajVektory()
3: DaemonickéSprávanie() (voľitelné)[1]
```

[1] Posledná funkcia, DaemonickéSprávanie(), môže byť použitá na implementáciu centralizovaných akcií, ktoré nemôžu byť prevedené jedným agentom, ako napríklad zber globálnych dát, ktoré môžu byť použité pri rozhodovaní či sa oplatí nechať agenta tancovať. V tejto práci sa toto správanie nepoužíva.

**Algoritmus 2** Vnútorné zmeny agenta

1: **if** Stav is Ostaň Doma **then**
2: **if** Vektor existuje **then**
3: Využívanie
4: **end if**
5: **else if** Agent neni Doma **then**
6: **if** Agent má potravu **then**
7: NesiePotravu
8: **else if** Záleží na šanci **then**
9: ChoďDomov, Explorácia or Využívanie
10: **end if**
11: **else if** Využi preferované **AND** stav is Doma **then**
12: **if** Vektor existuje **then**
13: Využívanie
14: **else**
15: Explorácia
16: **end if**
17: **else if** Preferencia Ostať Doma AND stav is Doma **then**
18: **if** Vektor existuje **then**
19: Využívanie
20: **else**
21: OstaňDoma
22: **end if**
23: **else**
24: Explorácia
25: **end if**

Po prvé, *SpavujVčeliuAktivitu()* jedná s aktivitami agentov, založenými na ich vnútornom stave. Každý agent má šesť vnútorných stavov, pričom v každom stave sa vykonáva špecifické správanie.

Stav agenta „Doma“ indikuje, že agent sa nachádza v úli. Pokiaľ je v tomto stave, agent sa rozhoduje, do akého nového stavu pôjde. Stavové zmeny sa dejú podľa Algoritmu 2.

Stav agenta „Ostaň Doma“ taktiež indikuje, že agent je v úli. Avšak pokiaľ je v tomto stave, ostane v úli a opustí ho, iba ak by mal dostupné už nejaké predošlé exploračné skúsenosti. V tom prípade agent opustí úľ, aby využil tieto nadobudnuté skúsenosti.

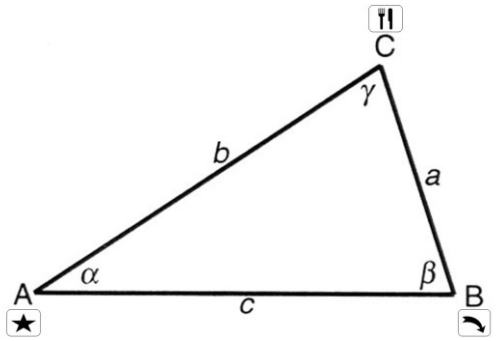
Stav „Využívanie“ indikuje, že agent využíva už dopredu nadobudnuté exploračné skúsenosti. Tieto skúsenosti sú reprezentované PI vektorom indikujúcim zdroj potravy. Agent sa rozhodne, do ktorého článku úľa sa presunie aby sa zhodoval s PI vektorom indikujúcim zdroj potravy.

Stav agenta „Explorácia“ indikuje, že agent báda po svojom okolí po zdroji potravy.

Stav „ChoďDomov“ indikuje, že agent smeruje naspäť domov bez toho, aby niesol potravu. Domov sa dostane nasledovaním HV. Od momentu, kedy agent začne svoju cestu za hľadaním potravy sa HV neustále počíta pre každého agenta.

Stav „NesiePotravu“ znamená, že agent našiel potravu a že ju nesie naspäť domov. Cesta naspäť je určená tým istým HV vektorom ako u agenta v stave „ChoďDomov“.

Po druhé, *VýpočítajVektory()* je používané na administratívne účely a ráta PI vektor pre každého agenta, t. j. domáci vektor a pravdepodobne PI vektor indikujúci zdroj potravy. Náš model používa presný výpočet PI vektora, ktorý vylučuje smerové a vzdialenostné chyby, ktoré sa v prírodnom vektore určite vyskytujú (M¨uller a Wehner, 1988; Collett T. a Collett M., 2004). Avšak, funguje veľmi podobne.




**Forma 1**. Hviezdička reprezentuje úľ, šípka reprezentuje začiatočný bod agenta a príbor reprezentuje agentov cieľ.

Nový PI vektor je vypočítaný berúc ohľad na ten starý. Predpokladajme, že vo forme 1, *A* je referenčný bod, *B* je štartovná pozícia agenta a *C* je agentova destinácia. V tom prípade by *a* bola vzdialenosť, *b* by bola nová navádzacia vzdialenosť a *c* by bola stará navádzacia vzdialenosť. $\beta$ je uhol, o ktorý sa treba otočiť aby sa dostal agent do *C* a $\alpha$ je uhol používaný na úpravy starého navádzacieho uhla. Aby sme vypočítali novú navádzaciu vzdialenosť, sme použili kosínusové pravidlo a prepísali ho na:

$$b = \sqrt{a^2 + c^2 - 2ac \times \cos\beta} \quad \text{(Rovnica 1)}$$

Použitím rovnice 1 teraz môžeme vypočítať $\alpha$ (uhol na úpravu starého navádzacieho uhla) znovu použitím kosínusového pravidla.

$$\alpha = \arccos\left(\frac{a^2-b^2-c^2}{-2bc}\right) \qquad \text{(Rovnica 2)}$$

Hodnoty získané v rovnici 1 a 2 sú použité na vytvorenie nového PI vektora. PI vektor teda pozostáva z 2 hodnôt, jedna indikuje smer a druhá indikuje vzdialenosť. Tieto hodnoty sú uložené v premennej pre každého agenta.

Hlavnou črtou včelieho správania je, že prirodzene zostavuje priame, optimálne cesty medzi začiatočným bodom (úľom) a destináciou (zdroj potravy). Niekto by mohol tvrdiť, že toto správanie je prirodzenou formou tvorby možností v „Markovových rozhodovacích procesoch“ (**MDP** – Markov Decision Process). Možnosti sú priebehy akcií v MDP , ktorého výsledky sú stavové prechody predĺženej a variabilnej doby trvania (Sutton et al., 1999). Takéto akcie sa preukázali ako veľmi výhodné pri zrýchľovaní učenia a plánovania, zabezpečujúc robustnosť a umožňujúc integráciu predošlých vedomostí do systémov umelej inteligencie (Iba, 1989). Možnosť je špecifikovaná sadou stavov, v ktorých môže byť možnosť inicializovaná vnútornými zásadami a podmienkami jej ukončenia. Ak je špecifikovaná inicializačná sada a podmienka ukončenia, tak môžu byť použité tradičné vystužovacie učebné metódy na naučenie sa vnútorných zásad danej možnosti. Vo včelej navigácií, základné akcie pozostávajú z pohybovania sa v rôznych smeroch nad uzlami v MDP (svet potravy). Zásady možnosti sú reprezentované PI vektorom (umelého) hmyzu, kde začiatočným bodom je úľ a podmienka ukončenia je lokácia zdroja potravy (Lemmens et al., 2007).

# 3 VÝSLEDKY RIEŠENIA A ICH ZHODNOTENIE

## 3.1 NXT VČELY

V tejto časti uvedieme a bližšie popíšeme programy ktoré sme vytvorili na názorné zobrazenie v práci popísaných algoritmov. Rozlišujeme dva hlavné programy: NXT Master a NXT Slave, pričom každý je ešte samostatne diferencovaný, závisiac od činnosti, ktorú má daná časť programu reprezentovať.

### 3.1.1 NXT Master

Názov „Master“ vychádza z Bluetooth štandardu a reprezentuje dominantnú kocku, ktorá vysiela údaje kocke „Slave“ a príma len potvrdenia o komunikácií s danou kockou. V našej simulácií Master kocka reprezentuje včelu v stave „Explorácia“, čo znamená, že robot nemá žiadne predošlé vedomosti o lokácii hľadaného objektu. Po spustení programu sa robot začne pohybovať po náhodnej trase a popritom bude hľadať potravu. Časť programu zodpovedajúca za tento pohyb je nasledovná:

**Pohyb**

```
#include "protocol.h" //Bluetooth komunikačný protokol
#define NAP 10
    mutex moveMutex; //token
#define turn_around \  //makro (otáča včelu)
        OnRev(OUT_C, 75); Wait(780); OnFwd(OUT_AC, 75); Wait(1000);  //otočenie (navráť do úľa)
#define next_destination(t) \  //makro na výber novej destinácie
    OnRev(OUT_C, 75); Wait(t); R++; //vyber nového smeru a zvyšovanie poctu jeho zmien

int P = 730;   //"uhol" otočenia
int R = 0;   //celkový počet zmien smeru

task move()    //vykonáva pohyb dokedy nedostane podnet na jeho ukončenie
{
    while (true)
    {
    Acquire(moveMutex);
    OnFwd(OUT_AC, 75); Wait(1000);
    turn_around;
```

```
    OnRev(OUT_C, 75); Wait(780);
    Off(OUT_AC); Wait(2000);
    next_destination(P);
    Release(moveMutex);
    }
}
```

Ďalšou časťou programu je komunikácia medzi včelími robotmi. Táto časť programu je bohužiaľ vierohodnosťou dosť odlišná od skutočných algoritmov implementovaných včelami kvôli softvérovým a hardvérovým obmedzeniam samotných NXT kociek. Ale keďže nevieme vierohodne zreprodukovať včelí tanec s dostupnými zdrojmi, tak jeho úlohu a účel aspoň približne znázorníme. Komunikačná časť Master programu funguje na báze Bluetooth komunikácie a keď robot dostane požadovaný podnet, tak začne vysielať získané informácie, v tomto prípade lokáciu potravy reprezentovanej nejakým objektom. V prípade, že robot dostane podnet na komunikovanie, tak pošle túto informáciu Slave kocke. Toto vysielanie nahrádza včelí tanec, ktorý v sebe obsahuje dané informácie.

**Komunikácia**

```
sub commain(int q)    //komunikačná podtrieda
{
     string a;
    // -- ráta počet iterácii a počet prijatých správ:
     int i = 1, n = 0;

    // -- inicializácia display-a:
     TextOut(0, LCD_LINE1, "        Master", true);
     TextOut(0, LCD_LINE2, "A: *");
     TextOut(0, LCD_LINE8, "rcvd msgs:");

    // -- kontrola BlueTooth spojenia (musí sa ustanoviť ako Master):
    // -- ak nie je žiadny Master, nastane chybový výpis a program sa ukončí
     mastercheck();

    // -- hlavný cyklus:
```

```
  for(;;)
  {
// -- získaj správu od slave-a:
    m = receivefromslave();

// -- prázdny string znamená "žiadna sprava", preto kontrolujeme jeho dĺžku:
    if(StrLen(m)>0)
     {
// -- zvýšime počet "prijatých sprav" a zobrazíme ho na display-i:
     n++;
     TextOut(0, LCD_LINE7, "              ");
     TextOut(0, LCD_LINE7, m);
     NumOut(66, LCD_LINE8, n);
     }
// -- q znižujeme o 1, pretože robot používa funkciu "Next Destination" jeden krát naviac pred tým ako sa "odprace z cesty"
    a = NumToStr((q-1)*P);

// -- samotný obsah správy, ktorú pošleme slave-ovi cez Bluetooth
     m = StrCat(a);

// -- výpis čísla správy, ktorú ideme poslať:
    NumOut(0, LCD_LINE1, i);

// -- výpis hodnôt:
    TextOut(18, LCD_LINE2, a);

// -- pošli správu:
    sendtoslave(m);
// -- zdriemni si na moment...
    Wait(NAP);
// -- ...ale nezaspi!
    ResetSleepTimer();
// -- aktualizuj počítadlo a pokračuj v cyklení:
    i++;
  }
}
```

Poslednou časťou Master programu je kontrola senzorov a hlavná trieda zodpovedná za chod celého programu. Senzorová časť reprezentuje včelie zmysly. Keď robotická včela dostane podnet na senzor, tak dá podnet pohybovej a komunikačnej časti podprogramu a ten po návrate domov začne vysielať; inak sa nič nedeje a včela hľadá ďalej. V reálnom svete to reprezentuje stav, keď bádajúca včela narazí na zdroj potravy. Zapamätá si jeho lokáciu a vráti sa domov túto informáciu odovzdať ostatným.

**Kontrola Senzorov a Hlavná Trieda**

```
task check_sensors()  //kontrola senzorov (podnet na ukončenie pohybu)
{
      while (true)
      {
      // -- kontrola či senzor niečo zaznamenal
      if (SENSOR_1 == 1)
         {
      Acquire(moveMutex);
      OnRev(OUT_AC, 75); Wait(500);
      Off(OUT_AC); Wait(5000);
      // -- ak bol podaný podnet, pošli ho slave-ovi
      commain(R);
      Release(moveMutex);
         }
      }
}

task main() //hlavná trieda
{
Precedes(move,check_sensors); //spustenie všetkých tried
SetSensorTouch(IN_1)[2]; //nastavenie senzoru na port 1, v tomto prípade dotykový
}
```

[2] Dotykový senzor sme si zvolil kvôli tomu, že má presne definované stavy kedy je stlačený (1) alebo nie (0). Pri iných senzoroch, ako napríklad svetelnom, sa tieto hodnoty pohybujú rôzne, závisiac či už od kalibrácie senzoru alebo od samotného povrchu ktorý sníma, ale prakticky by sme mohli použiť ktorýkoľvek senzor a pokiaľ by sa splnila podmienka na jeho aktiváciu by bol výsledok vždy rovnaký.

### 3.1.2 NXT Slave

Názov taktiež vychádza z Bluetooth štandardu, ale v tomto prípade reprezentuje submisívnu kocku, ktorá správy len prijíma a vysiela len potvrdenia o komunikácií. V simulácii táto kocka reprezentuje včelu v stave „Doma“. Pokiaľ nedostane informáciu od inej včely, ostáva doma a nič nerobí. Kvôli účelom demonštrácie sme robo-včele odobrali možnosť „Ostaň Doma“ z ponuky rozhodovacích procesov, takže vždy, keď sa Master včela vráti do úľa a začne „tancovať“, tak Slave včela si tieto informácie prisvojí za vlastné a bude konať podľa nich. Podobne, ako Master včela, aj Slave včela má triedu na pohyb, ale u nej je to len podtrieda, lebo pokiaľ nedostane informácie od Master včely, tak len čaká na mieste.

**Pohyb**

```
#include "protocol.h"    //inicializácia Bluetooth komunikácie
#define NAP 10

#define turn_around \
OnRev(OUT_C, 75); Wait(780); OnFwd(OUT_AC, 75); Wait(1000);

mutex moveMutex;

sub move(int r)  //robot sa začne hýbať až keď dostane od mastra spravu kam ma ísť.
{
      Acquire(moveMutex);
//-- na začiatku programu nie je ešte poslaná správa od Mastra, takže "r" by sa rovnalo 0 a robot by
zbytočne urobil 1 pohybový cyklus hore-dole
      while(r>0){
      OnRev(OUT_C, 75); Wait(r);
      OnFwd(OUT_AC, 75); Wait(1000)[3];
      turn_around;
      Off(OUT_AC); Wait(10000);
      Release(moveMutex);
      }
}
```

[3] Všetky hodnoty pre parameter Wait() vo všetkých častiach oboch programov, sú simulačné a ich hodnota nemá nijaký vplyv na celkovú funkčnosť programov. Zvolené hodnoty slúžia len k zjednodušeniu praktickej demonštrácie.

Ďalšou a zároveň aj poslednou triedou je komunikačná trieda, ktorá je zároveň aj hlavnou triedou. V tomto prípade reprezentuje prijímanie informácie z tanca inej včely a následné využitie danej informácie na navigáciu k cieľu. Bohužiaľ, ani tu nie sú skutočné algoritmy dôveryhodne znázornené.

**Komunikácia**

```
task main()
{
  string r, m, tmp;
  int i = 0, j;

 // -- inicializácia robota ako "slave"
  slavecheck();

 // -- nachystanie desplay-a:
  TextOut(0, LCD_LINE1, "sending:");
  TextOut(0, LCD_LINE4, "receiving:");

  for(;;)
  {
 // -- získaj string od iného robota (mastra):
   r = receivefrommaster();

 // -- dĺžka správy (0 znamená žiadna správa) lenght of message string (zero means "no message received"):
   j = StrLen(r);

 // -- vypíš na obrazovku len ak je práva:
   if(j!=0)
   {
    TextOut(0, LCD_LINE5, "              ");
    TextOut(0, LCD_LINE5, r);
   }

 // -- vytvor správu na poslanie nasäť (a vypíš ju na display):
```

```
        tmp = NumToStr(i);
        m = StrCat("msg", tmp);
        TextOut(54, LCD_LINE1, m);

      // -- choď na požadované miesto:
        move(StrToNum(r));

      // -- pošli správu mastrovy:
        sendtomaster(m);

        Wait(NAP);

        ResetSleepTimer();

      // -- aktualizuj počítadlo a pokračuj v cyklení:
        i++;
       }
}
```

# ZÁVER

V práci bol vytvorený program NXT Včely, ktorý bol vytvorený za cieľom demonštrácie funkcie algoritmov využívaných sociálnym hmyzom. Aj keď sa teoreticky podarilo dopodrobna popísať rôzne algoritmy, využívané či už mravcami, alebo včelami a vysvetliť princípy na akých tieto algoritmy fungujú, alebo ako vyzerá ich všeobecný matematický zápis, praktickú realizáciu sťažovali hardvérové a programové obmedzenia. NXT robotom sa podarilo napodobniť niektoré črty správania sa sociálneho hmyzu, a výsledný program NXT Včely symbolicky demonštruje, ako asi dané algoritmy môžu vyzerať v reálnom svete. Kódová časť programu sa zameriava viac na všeobecnú funkčnosť a využiteľnosť, napríklad na demonštráciu pri výuke ,ako na vierohodnosť a zhodu so skutočnými algoritmami. Dôvodom je fakt, že skutočné algoritmy závisia aj od vecí, ktoré proste Lego Mindstorms nedokáže zreprodukovať.

# ZOZNAM BIBLIOGRAFICKÝCH ODKAZOV


BERTELLE, C., DUTOT, A. a GHNEMAT, R. *Modeling Self-Organizing Systems with social insects algorithms. In* BRAIN. Broad Research in Artificial Intelligence and Neuroscience, ISSN 2067-3957, Volume 1, July 2010 [online]. [cit 2012-3-2] Dostupné na internete: <http://www.edusoft.ro/brain/index.php/brain/article/view/99>

Bricx CC History. [online]. [cit. 2011-10-16]. Dostupné na internete: <http://texteditors.org/cgi-bin/wiki.pl/wiki.pl?Bricx_CC>

BONABEAU, E.; DORIGO, M. a G. THERAULAZ (1999) *Swam Intelligence, from natural to artificial systems*, "Santa Fe Institute Studies in the Sciences of Complexity" series, Oxford University Press.

CAMAZINE S. a SNEYD J. *A model of collective nectar source by honey bees: selforganization through simple rules.* Journal of Theoretical Biology, 149:547–571, 1991.

CHONG C., LOW M.-H., SIVAKUMAR A.a GAY K. *A bee colony optimization algorithm to job shop scheduling*. *In* In Proceedings of the 2006 Winter Simulation Conference, pages 1954–1961, Monterey, CA USA, 2006.

COLLETT P., GRAHAM T.S. a DURIER V. *Route learning by insects*. Current Opinion in Neurobiology, 13(6):718–725, 2003.

COLLETT T. a COLLETT M. *How do insects represent familiar terrain*. Journal of Physiology, 98:259–264, 2004.

DORIGO M. a ST¨UTZLE T. *The ant colony optimization metaheuristic: algorithms, applications, and advances*. Technical report, Universit Libre de Bruxelles, 2000.

DYER F. *When it pays to waggle*. Nature, 419:885–886, 2002.

GERALDO B. a JING W.. *Swarm intelligence*. In *Proceedings Seventh Annual Meeting of the Robotics Society of Japan*, pages 425–428, 1989.

HUSÁR P. a SÁMEL P. *Hmyz* [online]. [cit. 2011-10-16]. Dostupné na internete: <www.gjgt.sk/digitalna_studovna/biologia/2010/94_**hmyz**.doc>

IBA G. *A heuristic approach to the discovery of macro-operators*. Machine Learning, 3:285–317, 1989.

KENNEDY J. a EBERHART R. *Particle swarm optimization*. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

KENNEDY J. a EBERHART R. C. *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001.

LAMBRINOS D., M¨OLLER R., LABHART T., PFEIFER R., a WEHNER R. *A mobile robot employing insect strategies for navigation*. Robotics and Autonomous Systems, 30(1-2):39–64, 2000.

LEGO® MINDSTORMS®, 2011 *NXT Hardware Developer Kit* [online]. [cit. 2011-10-18] Dostupné na internete:

<http://www.csd.uoc.gr/~hy325/reading/lego_nxt_hw_dev_kit.pdf>

LEGO MINDSTORMS NXT*, A Review and History of the Next Generation of Robotics, NXT* [online]. [cit. 2011-10-16]. Dostupné na internete: <http://www.educationaltoyskidslove.com/review-history-lego-mindstorms-nxt.html>

LEMMENS N., de JONG S., TUYLS K. a NOWE A. *A Bee Algorithm for Multi-Agent Systems: Recruitment and navigation combined.* In *Proceedings of ALAG 2007, an AAMAS 2007 workshop*, Honolulu, Hawai, 2007.

LUCISTNIK, P.2011. *FreeBSD Handbook* [online]. [cit 2011-10-18] Dostupné na internete: < http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/network-bluetooth.html>

MICHELSEN A., ANDERSEN B., STORM J., KIRCHNER W., a LINDAUER M. *How honeybees perceive communication dances, studied by means of a mechanical model*. Behavioral Ecology and Sociobiology, 30(3-4):143–150, 1992.

MINDELL, D. 2000. *LEGO Mindstorms, The Structure of an Engineering (R)evolution* [online]. [cit. 2011-10-16]. Dostupné na internete: <http://web.mit.edu/6.933/www/Fall2000/LegoMindstorms.pdf>

MOHAMED, J. O.A. a SIVAKUMAR, R. *Ant-based Clustering Algorithms: A Brief Survey. In* International Journal of Computer Theory and Engineering, Vol. 2,

No. 5, October, 2010 793-8201 [online]. [cit 2012-2-20] Dostupné na internete: <http://www.ijcte.org/papers/242-G730.pdf>

M¨ULLER M. a WEHNER R. *Path integration in desert ants*, Cataglyphis Fortis. Proceedings of the National Academy of Sciences, 85(14):5287–5290, 1988.

NAKRANI S. a TOVEY C. *On honey bees and dynamic server allocation in internet hosting centers*. Adaptive Behaviour, 12:223–240, 2004.

PARKER, E. L. *Current State of the Art in Distributed Autonomous Mobile Robotics.* [online]. [cit 2011-12-11] Dostupné na internete: <http://www.cs.cmu.edu/afs/cs.cmu.edu/Web/People/motionplanning/papers/sbp_papers/integrated1/parker_aut_robots.pdf>

REYNOLDS C. W., *Flocks, herds, and schools: A distributed behavioral model*. *Computer Graphics (SIGGRAPH '87 Conference Proceedings)*, 21(4):25–34, 1987.

SUTTON R., PRECUP S., a SINGH S. *Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning*. Artificial Intelligence, 112:181–211, 1999.

TEODOROVIC D. a DELL' ORCO M. *Bee colony optimization: A cooperative learning approach to complex transportation problems.* In Proceedings of the 16th Mini – EURO Conference and 10th Meeting of EWGT, 2006.

von FRISCH K. *The dance language and orientation of bees*. Harvard University Press, Cambridge, Massachusetts, 1967.

# ZOZNAM PRÍLOH

Príloha A – CD nosič

CD obsahuje:

- Elektronická podoba bakalárskej práce „Algoritmy na kooperáciu autonómnych robotov“
- Program NXT Včely
- Program Bricx CC