Slovenská technická univerzita
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: Informatika


# *Robocup S - Nové stratégie*

Implementácia navrhnutého riešenia

verzia 0.1

## História zmien

| Dátum zmeny | Verzia dokumentu | Popis zmeny | Autor |
|---|---|---|---|
| 24.4.2007 | 0.1 | Založenie dokumentu a vloženie sekcií o koordinačných grafoch, pohľade hráča a úprave fuzzy pravidiel. | Juraj Staník |
| 14.5.2007 | 0.2 | Doplnenie zmien implementácie voči návrhu, brankára, opravené drobné chyby | Peter Kohaut |

## Tím


| | |
|---|---|
| Bc. Peter Kohaut | feshi@feshi.com |
| Bc. Martin Kováčik | mato.kovacik@gmail.com |
| Bc. Ladislav Lenčucha | ladislav.lencucha@gmail.com |
| Bc. Tomáš Selnekovič | tomas.selnekovic@mathsite.org |
| Bc. Juraj Somorovský | somimail@gmail.com |
| Bc. Juraj Staník | juraj.stanik@gmail.com |


Webová stránka tímu:

http://www2.dcs.elf.stuba.sk/TeamProject/2006/Team03

## Anotácia

Táto časť obsahuje krátky opis obsahu dokumentu. Súčasťou je zhrnutie cieľov, ktoré má tento dokument dosiahnuť.

## Obsah dokumentu

# 1 Úvod

Tento dokument obsahuje informácie aké navrhnuté časti sa implementovali, postup ich implementovania. Taktiež obsahuje zmeny vykonané oproti návrhu, zmeny sú zdokumentované a ak sa zmenil návrh tak aj tento nový návrh je popísaný v jednotlivých kapitolách.

# 2 Zmeny implementácie oproti návrhu

Keďže nevyšlo všetko tak ako sme zamýšľali a vzhľadom na výskumnú povahu projektu, tak niektoré časti systému sme neimplementovali prípadne implementovali v pozmenenej podobe.

## 2.1 Koordinačné grafy

Ku koordinačným grafom nie dostupných veľa materiálov, preto sa nám podarilo implementovať iba prideľovanie rolí a základ algoritmu. Taktiež nedostatok času nám nedovolil dobre otestovať existujúcu časť algoritmu. Prideľovanie rolí funguje na základe návrhu. Zo samotného algoritmu eliminácie premenných sa nám podarilo implementovať vyhodnocovanie a spracovanie jednotlivých pravidiel. Algoritmus vytvára zoznamy na základné ktorých sa dajú vylučovať premenné a tak je algoritmus pomerne blízko ukončenia. Samotný algoritmus je dobre zdokumentovaný a popísaný

## 2.2 Q-learning

na qlearning sa pripravil existujúci systém doplnením funkcií ktoré umožnia jeho prepojenie s učiacim algoritmom q-learningu. Samotný algoritmus je implementovaný v knižnici tretej strany. Kvôli nedostatku času a náročnému návrhu štruktúr (diskrétneho priestoru) použitých pri učení sa nám samotný algoritmus použiť nepodarilo.

## 2.3 Prihrávky

Oproti návrhu sa zmenil iba spôsob výberu lúča (dozadu už hráč prihrávať nebude) a spôsob výpočtu výhodnosti (viac je opísaná v konkrétnej kapitole).

## 2.4 Fuzzy regulátor

Boli upravené parametre aby hráč lepšie kopal na bránku

## 2.5 Brankár

Oproti návrhu sa upravil navyše brankár.

# 3 Koordinačné grafy

## 3.1 Úvod

Metóda rozhodovania pomocou koordinačných grafov sa používa na vysokoúrovňové rozhodovanie v kooperatívnych multiagentových systémoch, v ktorých všetci agenti majú jeden spoločný cieľ. Koordináciou sa rozumie proces, ktorý zabezpečuje, že individuálne rozhodnutia agentov vyústia v danom okamihu do realizácie optimálnej spoločnej akcie všetkých agentov.

V koordinačnom grafe, každý uzol reprezentuje agenta, každá hrana indikuje, že príslušní agenti spojení hranou musia navzájom koordinovať svoje akcie, ktoré sa chystajú vykonať. Na riešenie lokálnych koordinačných problémov medzi agentami sa používa ***algoritmus eliminácie premenných***. Tento algoritmus iteratívne rieši lokálne koordinačné problémy, výsledky riešenia týchto problémov šíri ďalej po koordinačnom grafe. Tieto čiastkové výsledky ovplyvňujú voľbu akcií a riešenie súvisiacich koordinačných problémov medzi ďalšími agentami.

Topológia grafu sa v každom okamihu aktualizuje podľa aktuálnych informácií získaných z modelu sveta hráča. Následne je použitý algoritmus eliminácie premenných na vyriešenie koordinačného problému všetkých agentov.

Uva Trilearn použitý algoritmus rozhodovania pomocou koordinačných grafov ešte zjednodušuje tak, že pridelí jednotlivým agentom role a namiesto koordinovania akcií jednotlivých agentov koordinuje akcie pre jednotlivé role. Pridelenie rolí agentom redukuje množinu dostupných akcií pre agenta a tak zmenšuje veľkosť priestoru spoločnej akcie všetkých agentov. Naviac, pri tomto procese sa s použitím modelu sveta každého agenta a istých spoločných vedomostí dostupných každému agentovi odpadá nutnosť priamej komunikácie agentov.

## 3.2 Koordinačný problém z pohľadu Nashovej rovnováhy

### 3.2.1 Koordinačný problém

Každý agent má v nejakom okamihu dostupnú nejakú množinu akcií $A_i$. Naviac, má definovanú tzv. payoff alebo výnosovú funkciu R. Spoločná akcia $A$ všetkých agentov v danom okamihu je definovaná ako kartézsky súčin všetkých týchto množín dostupných akcií jednotlivých agentov $A = A_1 \times A_2 \times \ldots \times A_n$, kde n je počet agentov v poli. Výnosová funkcia je potom definovaná ako $R_i(A) \rightarrow \square$. To znamená, že každý agent si v jednom okamihu zvolí nejakú akciu a dostane výnos (reálne číslo) založený na výstupe výnosovej funkcie, ktorá zohľadní jednotlivé akcie všetkých agentov v danom okamihu. Každý agent má výnosovú funkciu definovanú rovnako, t.j. platí $R_1 = R_2 = \ldots = R_n$.

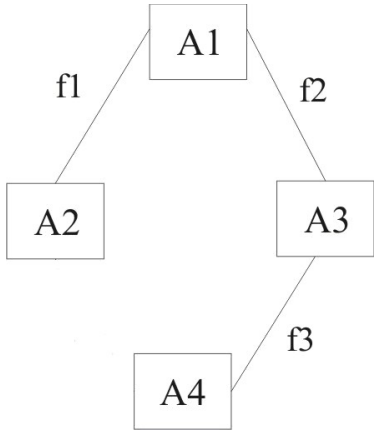
### 3.2.2 Nashova rovnováha

Problém hľadania najvýhodnejšej spoločnej akcie môže byť vyjadrený hľadaním tzv. Nashovej rovnováhy. Nashova rovnováha definuje spoločnú akciu $a^* \in A$ s takou vlastnosťou, že pre každého agenta $i$ platí, že $R_i\left(a_i^*, a_{-i}^*\right) \geq R_i\left(a_i, a_{-i}^*\right)$ pre všetky akcie $a_i \in A$, kde $a_{-i}$ je spoločná akcia pre všetkých agentov s výnimkou agenta $i$. Akcia $a^*$ je optimálna spoločná akcia a maximalizuje globálnu výnosovú funkciu. Takáto „rovnovážna“ spoločná akcia predstavuje stabilný stav z ktorého sa žiadny agent výhodne nemôže vychýliť, ak sú dané akcie ostatných agentov. Inými slovami, Nashova rovnováha nastáva, ak každý hráč urobí to, čo je pre neho najvýhodnejšie v závislosti na činnosti akéhokoľvek iného hráča, ktorý tak isto robí to, čo je pro neho najvýhodnejšie s ohľadom na činnosť ostatných hráčov. Z pohľadu jedného agenta ide o to, že agent vykoná takú akciu, ktorá je pre neho najvýhodnejšia s ohľadom na to, aby neznížil výhodnosť vykonania akcií agentov s ktorými kooperuje spoju činnosť zameranú na dosiahnutie spoločného cieľa.

Keďže počet rôznych spoločných akcií pre všetkých agentov rastie exponenciálne, z výkonnostných dôvodov nie je možné prehľadávať celý stavový priestor spoločných akcií a hľadať všetky Nashove rovnováhy. Pre zjednodušenie tohto procesu sa pridelia jednotlivým agentom role a namiesto koordinovania akcií jednotlivých agentov sa koordinujú akcie pre jednotlivé role.

## *3.3 Koordinačné grafy*

Pomocou koordinačných grafov vieme koordinovať akcie jednotlivých agentov. V koordinačných grafoch sú uzly jednotliví agenti, ktorí koordinujú svoje akcie. Hrany špecifikujú tzv. výnosové funkcie, ktoré vyjadrujú koordinačné závislosti medzi agentami. Koordinačná závislosť medzi dvoma agentami znamená, že to, akú akciu vykoná jeden agent v danom cykle závisí od zamýšľanej akcie, ktorú plánuje vykonať druhý agent. Lokálne výnosové funkcie špecifikujú aký výnos nastane, ak sa naraz splnia podmienky ňou špecifikované (napr. aký výnos nastane, ak agent A1 vykoná akciu dribluj a zároveň agent A2 vykoná akciu presuň_sa). Príspevky lokálnych výnosových funkcií sa spočítajú do výnosu celkovej spoločnej akcie agentov v danom cykle – globálneho výnosu špecifikovaného globálnou výnosovou funkciou. Vzhľadom na zložitosť hľadania optimálnej spoločnej akcie sa nekoordinujú akcie všetkých agentov navzájom, ale iba akcie podmnožín jednotlivých agentov (pridelením rolí – koordinujú sa role, nie jednotliví agenti).




*Obrázok 1: Koordinačný graf pre problém koordinácie akcií štyroch agentov*

Globálna výnosová funkcia $R(a)$ môže byť dekomponovaná na lineárnu kombináciu lokálnych výnosových funkcií, z ktorých každá vyhodnocuje spoločnú akciu menšieho počtu agentov ako pôvodná funkcia. Napríklad, predpokladajme, že máme štyroch agentov a ich globálnu výnosovú funkciu môžme dekomponovať nasledovne:

$$R(a) = f_1(a_1, a_2) + f_2(a_1, a_3) + f_3(a_3, a_4) .$$

Jednotlivé funkcie $f_i$ sú tiež výnosové funkcie, ale vyhodnocujú spoločnú akciu iba dvoch agentov a zároveň každá špecifikuje len lokálne koordinačné závislosti medzi menším počtom agentov ako globálna výnosová funkcia. Zodpovedajúci koordinačný graf je zobrazený na obrázku č. 1. Uzol v tomto grafe reprezentuje agenta $A_i$, lokálne výnosové funkcie $f_i$ špecifikujú lokálne koordinačné závislosti – agent A1 musí koordinovať svoju akciu s agentom A2 a A3. Agent A4 s A3. A3 musí koordinovať svoje akcie s A1 a A4 a nakoniec agent A2 s A1. Globálny koordinačný problém je teda nahradený istým počtom lokálnych koordinačných problémov, ktoré sa týkajú menšieho množstva agentov a ktoré je preto možné jednoduchšie vyriešiť.

### 3.3.1 Algoritmus eliminácie premenných

Na vyriešenie lokálnych koordinačných problémov sa používa algoritmus eliminácie premenných. Hlavnou myšlienkou je, že agenti sú eliminovaní jeden za druhým po vykonaní tzv. lokálneho maximalizačného kroku, pri ktorom uvažujeme všetky možné kombinácie akcií agentových susedov v koordinačnom grafe.

Algoritmus pracuje nasledovne:

1. Vyberieme náhodne agenta z koordinačného grafu na elimináciu. Tento agent zozbiera výnosové funkcie od svojich susedov.
2. Agent optimalizuje svoje rozhodnutie akú akciu vykoná v nasledujúcom kroku tak, že uváži všetky možné kombinácie akcií svojich susedov, tieto kombinácie akcií spolu so svojou akciou ohodnocuje výnosovou funkciou. Vyberie si takú akciu, ktorá maximalizuje výnosovú funkciu. Následne výsledok – správu o tom, akú akciu vykoná vo forme konštrukcie novej výnosovej funkcie odošle späť svojim susedom.
3. Agent je eliminovaný z grafu.
4. Proces pokračuje bodom 1 dovtedy, pokiaľ nie sú z grafu eliminovaní všetci agenti. Ak už sú všetci agenti eliminovaní, pokračuje sa bodom č. 5.
5. Posledný agent vyberie najoptimálnejšiu akciu, ktorá maximalizuje výnosovú funkciu pre celú skupinu agentov v grafe.
6. Vykoná sa prechod grafom v reverznom poradí eliminácie agentov z grafu, aby každý agent vedel určiť optimálnu akciu podľa zvolených akcií svojich susedov.

***Príklad:***

Obrázok č. 1 zobrazuje situáciu potreby koordinácie štyroch agentov: A1, A2, A3 a A4. Eliminujme postupne všetkých agentov z grafu. Proces eliminácie začneme napríklad výberom agenta A1. Tento agent zozbiera výsledky výnosových funkcií svojich susedov $f_2(a_1,a_3)$ a $f_1(a_1,a_2)$ (špecifikujú koordinačnú závislosť medzi agentami A1 a A3 a a medzi agentami A1 a A2). Následne sa A1 snaží vybrať takú akciu $a_1$ ktorá maximalizuje zloženú výnosovú funkciu $f_1 + f_2$. Tento princíp môžme zapísať nasledovne:

$$\max_a R(a) = \max_{a_2,a_3,a_4}\left\{ f_3(a_3,a_4) + \max_{a_1}\left[ f_1(a_1,a_2) + f_2(a_1,a_3)\right]\right\}.$$

Agent A1 teraz vytvorí novú výnosovú funkciu $f_4(a_2,a_3) = \max_{a_1}\left[ f_1(a_1,a_2) + f_2(a_1,a_3)\right]$, ktorá vráti hodnotu zodpovedajúcu výberu najlepšej akcie $a_1$, ktorá ju maximalizuje. Po výbere takejto akcie je problém zjednodušený a platí, že:

$$\max_a R(a) = \max_{a_2,a_3,a_4}\left[ f_3(a_3,a_4) + f_4(a_2,a_3)\right].$$

Rovnaký proces aplikujeme ďalej napr. na agenta A2. V tomto prípade, jedine výnosová funkcia $f_4$ záleží od akcie agenta A2. Potom: $f_5(a_3) = \max_{a_2} f_4(a_2, a_3)$. Následne platí, že:

$$\max_a R(a) = \max_{a_3, a_4} \left[ f_3(a_3, a_4) + f_5(a_3) \right].$$

Následne, z koordinačného grafu eliminujeme agenta A3 nahradením funkcií $f_3$ a $f_5$ funkciou $f_6(a_4) = \max_{a_3} \left[ f_3(a_3, a_4) + f_5(a_3) \right]$. Potom sa problém hľadania globálneho maxima výnosovej funkcie redukuje na:

$$\max_a R(a) = \max_{a_4} f_6(a_4).$$

Agent A4 si teraz vyberie optimálnu akciu, ktorá maximalizuje výnosovú funkciu. Táto akcia bude slúžiť ako konštanta vo výnosových funkciách jeho susedov. Následne sa vykoná prechod grafom v reverznom poradí výberu agentov na elimináciu a každý agent si zvolí takú akciu z tých akcií, ktoré má dostupné, ktorá maximalizuje výnosovú funkciu.

Tento princíp môžeme zapísať v pseudo kóde nasledovne:

For each agent in parallel
  $F = \{f_1, \ldots, f_k\}$.
  For each agent $i = 1, 2, \ldots, n$
    Find all $f_j(a_{-i}, a_i) \in F$ that involve $a_i$.
    Compute $B_i(a_{-i}) = \arg\max_{a_i} \sum_j f_j(a_{-i}, a_i)$.
    Compute $f_{k+i}(a_{-i}) = \max_{a_i} \sum_j f_j(a_{-i}, a_i)$.
    Remove all $f_j(a_{-i}, a_i)$ from $F$ and add $f_{k+i}(a_{-i})$ in $F$.
  End
  For each agent $i = n, n-1, \ldots, 1$
    Choose $a_i^* \in B_i(a_{-i}^*)$ based on a fixed ordering of actions.
  End
End

*Obr. 1: Algoritmus eliminácie premenných*

Pomocou algoritmu eliminácie agenti vyberú také akcie, ktoré predstavujú Nashovu rovnováhu a tak vyberú najlepšiu možnú spoločnú akciu. Výnosové funkcie, ktoré hodnotia spoločné akcie agentov a špecifikujú koordinačné závislosti sú definované ako hodnotové pravidlá, ktoré špecifikujú ako vybraná akcia agenta závisí na aktuálnom kontexte špecifikovaného jednak definovanou stratégiou tímu a jednak akciami ostatných agentov. Ukážme teraz ako je reprezentovaná výnosová funkcia. Výnosová funkcia je špecifikovaná tzv. kontextom, ktorý určuje, kedy výnosová funkcia poskytne hodnotu výnosu v závislosti od daného kontextu. Pokiaľ kontext nie je splnený, výnos je nulový.

### 3.3.2 Kontext

Kontext je definovaný ako vetné pravidlo nad stavovými premennými z modelu sveta a vybranými akciami susedov agenta.

Formálne, nech A1, A2, ..., AN je skupina agentov. Každý agent AJ si musí vybrať nejakú akciu $a_j$ z množiny dostupných akcií $A_j$, teda $a_j \in A_j$. Tento výber akcií všetkých agentov vyústi v jednom cykle do vykonania jednej spoločnej akcie $a \in A = A_1 \times \ldots \times A_n$. Nech $X$ je množina stavových premenných z modelu sveta agenta. Potom kontext $c$ je prvok z množiny všetkých možných kombinácií stavových premenných a akcií agentov: $c \in C \subseteq X \cup A$.

### 3.3.3 Hodnotové pravidlá

Výnosové funkcie sú reprezentované prostredníctvom tzv. hodnotových pravidiel. Hodnotové pravidlo $\langle p; c : v \rangle \in P$ je funkcia $p : C \rightarrow \square$, taká, že $p(x,a) = v$, ak $c = (x,a)$ je konzistentný s aktuálnym kontextom. V opačnom prípade $v = 0$. V konkrétnej situácií, iba tie hodnotové pravidlá prispievajú ku globálnemu výnosu špecifikovaného globálnou výnosovou funkciou, ktoré sú konzistentné s aktuálnym kontextom. Teda platí: $R(a) = \sum_{i=1}^{m} p_i(x,a)$, kde $m$ je celkový počet hodnotových pravidiel, $x$ sú aktuálne stavy premenných a $a$ je akcia (aj spoločná) agentov.

***Príklad:***

Uvažujme situáciu, že dvaja agenti musia koordinovať svoje akcie pri prechode cez jedny dvere. Evidentne, je nevýhodné, ak by sa cez dvere rozhodli prejsť obaja naraz. Potom pravidlo má tvar:

$$< \quad p_1; \quad su\_pred\_dverami\_agenti(1,2) \square$$
$$a_1 = prejdi\_cez\_dvere \square$$
$$a_2 = prejdi\_cez\_dvere \qquad :- \ 50 >$$

Ak je aktuálny kontext, t.j. aktuálny stav a zvolené akcie agentov konzistentný s kontextom definovaným v pravidle, potom sa pravidlo aplikuje a k zodpovedajúcemu kontextu špecifikuje príslušnú hodnotu výnosu.

Vo všeobecnosti:

- v koordinačnom grafe sa konektivita medzi agentami mení dynamicky podľa aktuálneho stavu sveta,
- koordinačné grafy medzi jednotlivými agentami musia byť riedke kvôli problému exponenciálneho rastu stavového priestoru spoločnej akcie,
- musia byť aplikovateľné na situácie, v ktorých komunikácia nie je možná alebo vykonanie komunikácie je príliš komplikované.

### 3.3.4 Role agentov

Každý agent v tíme musí mať pridelenú rolu. Rola predstavuje abstraktnú špecifikáciu istej množiny dostupných akcií, ktoré agent môže vykonať v danej situácií v závislosti od pridelenej roly. Každá rola špecifikuje rôzne množiny akcií.

Rovnaká rola môže byť pridelená aj viacerým agentom, ale každý agent má pridelenú iba jednu rolu. Každá rola m má priradený potenciál (kladné reálne číslo) $r_{im}$, čo je odhad ako je agent i vhodný na rolu m v závislosti od aktuálneho stavu sveta. Pre každú rolu sú pri určovaní potenciálu vzhľadom k danej role dôležité rôzne atribúty. Napr. pre rolu „útočník“ môžu byť dôležité napr. čas za ktorý je schopný hráč dobehnúť k lopte alebo globálna pozícia hráča na ihrisku, prípadne kombinácia týchto dvoch parametrov a podobne. Dôležité je, že hráč v každom okamihu vypočítava tieto potenciály pre každú rolu, následne si ju zvolí podľa toho, ku ktorej má najbližšie hodnotu vypočítaného potenciálu.
Týmto spôsobom si každý agent priradí rolu. Potenciály jednotlivých rolí a kritériá pri vypočítavaní týchto potenciálov je všeobecná vedomosť všetkých agentov. Brankár má rolu priradenú automaticky.

V tíme Uva Trilearn 2003 sa rozlišujú štyri typy rolí:

- **active –** očakáva sa, že hráč vykoná akciu s loptou,
  - **passer** – hráč môže kopnúť do lopty,
  - **interceptor** – hráč nemôže kopnúť do lopty,
- **receiver –** očakáva sa, že hráč dostane prihrávku,
- **passive –** hráč sa presúva na svoju strategickú pozíciu,
- **brankár**.

Rola ***active*** je najdôležitejšia a od agenta, ktorý túto rolu dostane sa očakáva, že s vysokou pravdepodobnosťou vykoná nejakú akciu s loptou. Rola ***receiver*** znamená, že od agenta, ktorý dostane pridelenú túto rolu sa očakáva, že pravdepodobne dostane nahrávku od hráča s rolou ***active***.

Role sú zoradené v prioritnom zozname podľa dôležitosti. V tíme Uva Trilearn 2003 bol zoznam rolí, ktorý bol k dispozícií určený nasledujúcou množinou:

$$M = \left\{ \begin{matrix} active, receiver, receiver, passive, passive, \\ passive, passive, passive, passive, passive \end{matrix} \right\}$$

V každom cykle sa si agenti podľa potenciálu vyberú príslušnú rolu. Nemalo by sa stávať, že viac agentov dostane rolu ***active/passer***. Rola ***active/interceptor*** je určená pre hráča, ktorý sa chystá zastaviť loptu, t.j. prebrať loptu od súpera. Od hráča s rolou a***ctive/passer*** je očakávané, že bude nahrávať hráčom s rolou ***receiver,*** prípadne že bude strieľať na bránku. Rola ***brankár*** je priradená automaticky agentovi, ktorý bude vykonávať funkciu brankára tímu.

#### 3.3.4.1 Spôsob prideľovania rolí agentom

Potenciál $r_{i,active}$ pre aktívneho hráča je rovný $\frac{1}{t_i}$, kde $t_i > 0$ je predpokladaný čas za ktorý sa hráč $i$ z aktuálnej pozície dostane k lopte. V tíme Uva Trilearn používajú na určenie tohto času upravenú Newtonovu metódu. Táto metóda nájde najmenší koreň (rovný najmenšiemu možnému času dobehnutia k lopte) funkcie ktorá reprezentuje rozdiel medzi prejdenou vzdialenosťou lopty a pohyb hráča $i$. Ak spĺňa hráč potenciál na to aby mohol byť považovaný za active, to či sa následne pridelí rola ***passer*** alebo rola ***interceptor*** sa rozhodne na základe vzdialenosti hráča od lopty. Ak je hráč dostatočne blízko aby mohol kopnúť do lopty, dostane rolu ***passer***, v opačnom prípade ***interceptor***.

Potenciál pre rolu receiver je určený na základe relatívnej vzdialenosti $d_{i,b}$ k lopte a relatívnej vzdialenosti k oponentovej bránke $d_{i,g}$ pre hráča $i$:

$$r_{i,receiver} = \begin{cases} \frac{1}{\max(1, d_{i,g})} + 1 & ,ak\ d_{i,b} < k \\ \frac{1}{\max(1, d_{i,g})} & ,inak \end{cases}$$

Táto funkcia hovorí, že ak má dostať hráč rolu ***receiver***, preferujú sa tí hráči, ktorí sú bližšie k súperovej bránke. Naviac, ak sa hráč nachádza do 28 metrov od súperovej bránky (k=28), potom dostane dodatočný bonus k potenciálu +1 bod.

Hráči, ktorým nebola pridelená ani rola ***receiver*** ani žiadna z rolí ***active***, potom dostanú automaticky rolu ***passive***.

Štruktúra koordinačného grafu závisí od aktuálneho rozdelenia rolí (pridelenie konkrétnej role závisí od aktuálneho stavu sveta) a mení sa v každom cykle. Agent, ktorý má pridelenú pasívnu rolu sa v grafe neuvažuje a vykonáva jedinú akciu – vykonáva pohyb ku svojej strategickej pozícií.

### 3.3.5 Podmienky aplikácie koordinačných grafov pri probléme koordinácie nekomunikujúcich agentov

Potenciál $r_{i,passive}$ pre rolu pasívneho hráča je konštantná, t.j. všetci zostávajúci agenti dostanú túto rolu pridelenú.
Pri probléme koordinácie nekomunikujúcich agentov musí každý agent obdržať od susedných agentov príslušné výnosové funkcie, vypočítať optimálnu stratégiu vo forme vytvorenia novej výnosovej funkcie a túto funkciu oznámiť späť svojim susedom. Podobne, pri reverznom procese, každý agent musí oznámiť svoje rozhodnutie, akú akciu vykoná jeho susedom za účelom dosiahnutia optimálnej spoločnej akcie všetkých agentov, ktorá spĺňa Nashovu rovnováhu a teda je v danej situácií najvýhodnejšia.

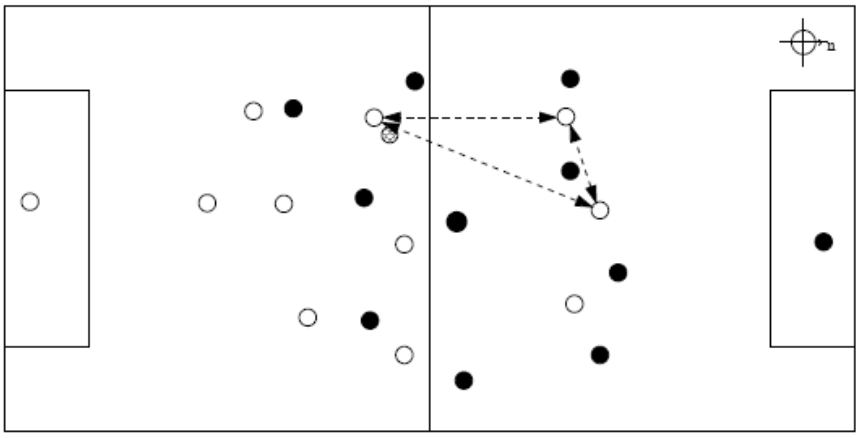
Aby sme teda mohli použiť koordinačné grafy pri probléme koordinácie nekomunikujúcich agentov, musia byť splnené nasledujúce podmienky:

1. výnosové funkcie agenta i sú známou vedomosťou všetkých ostatných agentov dostupných z agenta i v koordinačnom grafe,
2. každý agent dokáže spočítať jeho potenciál $r_{im}$ pre všetkých agentov i v koordinačnom grafe,
3. poradie rolí a k nim zodpovedajúce potenciály je známa vedomosť všetkých agentov,
4. všetci agenti v koordinačnom grafe musia byť schopní predvídať hodnoty stavových premenných svojich susedov

V hre Robocup je často nemožné splniť podmienku číslo 4 kvôli tomu, že každý agent nemá absolútne rovnaké informácie o prostredí ako ostatní agenti. V tíme Uva Trilearn 2005 platí, že ak sa agenti majú zhodnúť na distribúcií rolí vzhľadom na aktuálny kontext, potom dohoda alebo ich výber je založený hlavne na stavoch, ktoré sú dobre pozorovateľné všetkými agentami alebo ktoré sa dajú vhodne aproximovať. Výber môže byť založený napríklad na základe aktuálnych pozícií spoluhráčov, súperových hráčov a pozícií lopty.

### 3.3.6 Príklad aplikácie koordinačných grafov

Predstavme si na ihrisku nižšie uvedenú situáciu. Agent s loptou má pridelenú rolu ***active***, jeho spoluhráči spojení šípkou majú pridelenú rolu ***receiver***. Ostatní hráči majú pridelenú rolu ***passive***, t.j. nie sú koordinovaní s tými hráčmi, ktorí majú pridelenú inú rolu ako ***passive***.



*Obrázok 2: Situácia s jedným hráčom s rolou active a dvomi hráčmi s rolou receiver*

Každý z agentov v poli má k dispozícií nasledujúce akcie:

- $passTo(i, dir)$: pass the ball to a position with a fixed distance from agent $i$ in the direction $dir \in D = \{center, n, nw, w, sw, s, se, e, ne\}$. The direction parameter specifies a direction relative to the receiving agent. 'North' is always directed toward the opponent goal and 'center' corresponds to a pass directly to the current agent position,
- $moveTo(dir)$: move in the direction $dir \in D$,
- $dribble(dir)$: move with the ball in direction $dir \in D$,
- $score$: shoot to the best spot in the opponent goal (22),
- $clearBall$: shoot the ball hard between the opponent defenders to the opponent side,
- $moveToStratPos$: move to the agent's strategic position based on its home position and the position of the ball which serves as an attraction point.

Každý agent má k dispozícií nasledujúce informácie o stave sveta:

- $is\text{-}pass\text{-}blocked(i, j, dir)$ indicates whether a pass from agent $i$ to agent $j$ is blocked by an opponent or not. The actual position to which is passed is the position at a small fixed distance from agent $j$ in direction $dir$. A pass is blocked when there is at least one opponent located within a cone from the passing player to this position.
- $is\text{-}empty\text{-}space(i, dir)$, indicates that there are no opponents within a small circle in the specified direction $dir$ of agent $i$.
- $is\text{-}in\text{-}front\text{-}of\text{-}goal(i)$ indicates whether agent $i$ is located in front of the opponent goal.

Tieto akcie spolu so stavovými informáciami tvoria kontext. Použitím týchto akcií a stavových premenných môžeme prostredníctvom definície výnosových pravidiel (definujú príspevok ku globálnemu výnosu pre špecifický kontext). Výnosové pravidlá sú špecifikované pre každého hráča i a sú definované prostredníctvom uvedených stavových premenných a akcií.

Celková stratégia tímu je definovaná nasledovne:

$$
\begin{aligned}
&\langle p_1^{interc.} \;;\; \text{intercept} : 10\rangle \\
&\langle p_2^{passer} \;;\; \text{has-role-receiver}(j) \;\wedge \\
&\quad \neg\text{isPassBlocked}(i,j,dir) \;\wedge \\
&\quad a_i = \text{passTo}(j,dir) \;\wedge \\
&\quad a_j = \text{moveTo}(dir) : u(j,dir) \in [5,7]\rangle \;\forall j \neq i \\
&\langle p_3^{passer} \;;\; \text{is-empty-space}(i,\text{n}) \;\wedge \\
&\quad a_i = \text{dribble}(\text{n}) : 2\rangle \\
&\langle p_4^{passer} \;;\; a_i = \text{clearBall} : 0.1\rangle \\
&\langle p_5^{passer} \;;\; \text{is-in-front-of-goal}(i) \;\wedge \\
&\quad \text{is-ball-kickable}(i) \;\wedge \\
&\quad a_i = \text{score} : 10\rangle \\
&\langle p_6^{receiver} \;;\; \text{has-role-interceptor}(j) \;\wedge \\
&\quad \neg\text{isPassBlocked}(j,i,dir) \;\wedge \\
&\quad a_j = \text{intercept} \;\wedge \\
&\quad a_i = \text{moveTo}(dir) : u(i,dir) \in [5,7]\rangle \quad \forall j \neq i \\
&\langle p_7^{receiver} \;;\; \text{has-role-receiver}(k) \;\wedge \\
&\quad \neg\text{isPassBlocked}(k,i,dir) \;\wedge \\
&\quad a_j = \text{passTo}(k,dir2) \;\wedge \\
&\quad a_k = \text{moveTo}(dir2) \;\wedge \\
&\quad a_i = \text{moveTo}(dir) : u(i,dir) \in [5,7]\rangle \;\forall j,k \neq i \\
&\langle p_8^{receiver} \;;\; \text{moveToStratPos} : 1\rangle \\
&\langle p_9^{passive} \;;\; \text{moveToStratPos} : 1\rangle
\end{aligned}
$$

Tieto výnosové funkcie špecifikujú pre každú rolu kontext a zodpovedajúci výnos – príspevok ku globálnemu výnosu spoločnej akcie koordinovanej koordinačným grafom. Ak sú tieto všetky pravidlá inicializované, celkový počet výnosových pravidiel je 204. Tieto pravidlá sú všeobecnou vedomosťou všetkých agentov.

Ak sú už roly priradené, potom každý agent vie, ktoré výnosové pravidlá (funkcie) má v danom cykle uvažovať. Uvážia sa sa aktuálne stavové premenné a zistí sa, ktoré pravidlá sú výnosové pravidlá sú aplikovateľné. Aplikovateľné pravidlá v danom cykle špecifikujú koordinačné závislosti medzi agentami, t.j. je možné vytvoriť koordinačný graf.

Uvažujme situáciu na obrázku č. . V danej situácií uvažujeme desiatich agentov, z ktorých 1 agent má priradenú rolu **brankár**, 6 hráčov má priradenú rolu **passive**, dvaja hráči majú priradenú rolu

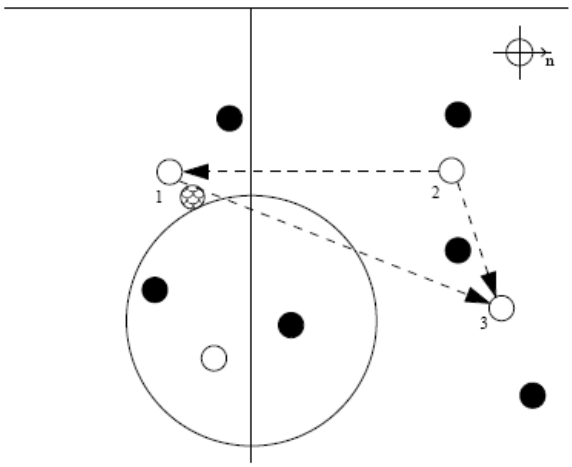
receiver a jeden hráč rolu **active** (podúlohu **passer)**. Uvažujme, že jediné platné stavové premenné sú:

$$\neg\, isPassBlocked(1,2,s) \text{ a } \neg\, isPassBlocked(2,3,nw).$$

potom pre jednotlivé role platia nasledujúce výnosové pravidlá:

$$\begin{aligned}
A_1 : \langle p_2^{passer} \quad ; \quad & a_1 = \mathrm{passTo}(2,\mathrm{s}) \quad \wedge \\
& a_2 = \mathrm{moveTo}(\mathrm{s}) : 6\rangle \\
\langle p_3^{passer} \quad ; \quad & a_1 = \mathrm{dribble}(\mathrm{n}) : 2\rangle \\
\langle p_4^{passer} \quad ; \quad & a_1 = \mathrm{clearBall} : 0.1\rangle \\
A_2 : \langle p_8^{receiver} \quad ; \quad & a_2 = \mathrm{moveToStratPos} : 1\rangle \\
A_3 : \langle p_7^{receiver} \quad ; \quad & a_1 = \mathrm{passTo}(2, dir) \quad \wedge \\
& a_2 = \mathrm{moveTo}(dir) \quad \wedge \\
& a_3 = \mathrm{moveTo}(\mathrm{nw}) : 5\rangle \ \forall dir \in D \\
\langle p_8^{receiver} \quad ; \quad & a_3 = \mathrm{moveToStratPos} : 1\rangle
\end{aligned}$$

Ak zohľadníme aplikovateľné výnosové pravidlá a role jednotlivých agentov, môžeme zostaviť koordinačný graf, ktorý koordinuje akcie troch agentov – jedného s rolou **active/passer** a dvoch s rolou **receiver** (obrázok č. 3). Orientované hrany v grafe špecifikujú od akého agenta (pri konci orientovanej hrany – rodič) závisia akcie potomka (pri začiatku orientovanej hrany – child).




*Obrázok 3: Koordinačný graf po uvážení stavových premenných modelu sveta. Passer (agent 1) sa rozhodne nahrať prvému hráčovi s rolou receiver (agent 2), zatiaľčo druhý receiver (agent 3) sa presunie na dobrú pozíciu na nahrávku z pohľadu prvého hráča s rolou receiver.*

Ak už máme koordinačné závislosti vyjadrené, môžeme aplikovať algoritmus eliminácie premenných. Každý agent je postupne eliminovaný z grafu, pričom maximalizuje svoj lokálny výnos zvolením takej akcie, ktorá maximalizuje lokálnu výnosovú funkciu.

Eliminujme z grafu najskôr napr. agenta – A1. V tomto prípade, agent zoberie od svojich susedov výnosové pravidlá (všetky tie pravidlá v ktorých figuruje jeho akcia). To, ktoré stavové premenné sú platné už poznáme. A1 teraz vyskúša všetky možné aplikovateľné kombinácie akcií svojich susedov, pričom pre každú kombináciu skúsi všetky možné kombinácie svojej akcie $a_1$ a zistí, aká jeho akcia maximalizuje danú lokálnu výnosovú funkciu. V našom prípade dostaneme nasledujúce tri nové výnosové pravidlá:

$$\begin{aligned}
&\langle p_{10}^{passer} \quad ; \quad a_2 = \text{moveTo(s)} \quad \wedge \\
&\qquad\qquad\quad a_3 = \text{moveTo(nw)} : 11\rangle \\
&\langle p_{11}^{passer} \quad ; \quad a_2 = \text{moveTo(s)} \quad \wedge \\
&\qquad\qquad\quad a_3 = \neg\text{moveTo(nw)} : 6\rangle \\
&\langle p_{12}^{passer} \quad ; \quad a_2 = \neg\text{moveTo(s)} : 2\rangle
\end{aligned}$$

Výnosové pravidlo $p_{10}^{passer}$ bolo vytvorené kombináciou pravidiel $p_2^{passer}$ a $p_7^{receiver}$ ak obaja agenti A2 a A3 vykonajú predpísané akcie. Ak A3 vykoná inú akciu a agent A2 sa presunie smerom na juh ako je stanovené v pravidle $p_{11}^{passer}$. Ak agent A2 tiež vykoná inú akciu, jediná zostávajúca akcia, ktorú A1 môže vykonať je driblovanie s výnosom 2. Po tom ako A2 a A3 si zvolili svoju stratégiu, agent A1 vykoná akciu passTo(2,s), agent A2 vykoná akciu moveTo(s) aby zachytil prihrávku a agent A3 vykoná akciu moveTo(nw) aby zachytil prípadnú budúcu nahrávku od agenta A2.

# 4 Implementácia rozhodovania pomocou koordinačných grafov

Problém hľadania optimálnej spoločnej akcie pomocou metódy koordinačných grafov pozostával z viacerých častí úloh:

- realizovať pridelenie rolí hráčom,
- navrhnúť vhodné dátové štruktúry na realizáciu výnosových funkcii,
- realizovať tímovú stratégiu pomocou výnosových pravidiel,
- implementovať viaceré chýbajúce predikáty vyskytujúce sa vo výnosových funkciách,
- implementovať algoritmus eliminácie premenných.

Počas doby riešenia projektu sa nám úspešne podarilo implementovať algoritmus prideľovania rolí pomocou upravenej Newtonovej metódy. Naviac sme vytvorili framework pre riešenie problému hľadania optimálnej spoločnej akcie prostredníctvom metódy koordinačných grafov. Stratégiu tímu sme špecifikovali prostredníctvom deviatich všeobecných výnosových funkcii (pozri kap. č. 3.6) použitý v tíme Uva Trilearn 2003.

## 4.1 Reprezentácia rolí hráčov

Role hráčov sme v našej implementácií reprezentovali pomocou vymenovaného typu:

```
enum RoleCG { ACTIVE_PASSER=0, ACTIVE_INTERCEPTOR, RECEIVER, PASSIVE, GOALIE,
NO_ROLE };
```

## 4.2 Reprezentácia všeobecnej akcie

Aby sme mohli implementovať výnosové pravidlá, bolo potrebné nájsť vhodnú reprezentáciu všeobecnej akcie hráča. Vo všeobecnosti, ľubovoľnú akciu hráča schopností nižnej úrovne je možné reprezentovať prostredníctvom dátovej štruktúry umožňujúcu špecifikovať *typ akcie*, *všeobecný smer* a *priradenie spoluhráča*, ktorého sa potenciálne akcia môže týkať. Parameter typ akcie je povinný, ostatné parametre sú voliteľné.

V algoritme eliminácie premenných nie je možné uvažovať a kombinovať všetky možné smery. Preto bolo nutné definovať definovať konkrétnu množinu možných smerov:

- N – north, smer od hráča k bodu OBJECT_GOAL_R
- NE – north east, smer od hráča k bodu OBJECT_FLAG_R_B
- E – east, smer od hráča k bodu OBJECT_FLAG_C_B
- SE – south east, smer od hráča k bodu OBJECT_FLAG_L_B
- S – south, smer od hráča k bodu OBJECT_GOAL_L
- SW – south west, smer od hráča k bodu OBJECT_FLAG_L_T
- W – west, smer od hráča k bodu OBJECT_FLAG_C_T
- NW – north west, smer od hráča k bodu OBJECT_FLAG_R_T

- NO_DIRECTION – nedefinovaný smer

Implementácia bola realizovaná vymenovaným typom DirCG:

```
typedef enum { N, NW, W, SW, S, SE, E, NE, NO_DIRECTION } DirCG;
```

Na to, aby sme mohli transformovať uvedené všeobecné smery na uhol vzťahujúci sa k aktuálnej polohe hráča sme implementovali transformačnú funkciu:

```
double transformGlobalDirectionToAngleCG(DirCG globalDirection);
```

V našej implementácií je možné akcií priradiť nasledujúce typy:

- INTERCEPT – akcia zachytenia lopty,
- LEADING_PASS_TO – nábežná prihrávka, tzv. leading pass v špecifikovanom smere,
- MOVE_TO – akcia presunu hráča daným smerom,
- DRIBBLE – akcia driblovania daným smerom pre implicitne definovanú dĺžku dráhy,
- CLEAR_BALL,
- SCORE – akcia kopnutia lopty na bránku,
- MOVE_TO_STRAT_POS – presun hráča na pozíciu špecifikovanú aktuálnou formáciou,
- NO_ACTION – implicitná hodnota všeobecnej akcie

Implementácia bola realizovaná vymenovaným typom PlayerActionTypeCG:

```
typedef enum { INTERCEPT=0, LEADING_PASS_TO, MOVE_TO, DRIBBLE, CLEAR_BALL,
SCORE, MOVE_TO_STRAT_POS, NO_ACTION } PlayerActionTypeCG;
```

Všeobecná akcia sa ešte môže týkať nejakého spoluhráča toho hráča, ktorý akciu vykoná. Spoluhráč v našej implementácií je reprezentovaný číslom. Toto číslo je možné získať pomocou transformačnej funkcie:

```
int getPlayerIndexRole(ObjectT o);
```

Z čísla – indexu hráča je možné pôvodný objekt hráča získať pomocou inverznej funkcie:

```
ObjectT getPlayerTypeRole(int index);
```

Všeobecnú akciu reprezentujeme pomocou triedy **PlayerActionCG**. Táto trieda umožňuje priradiť typ akcie, smer akcie a spoluhráča, ktorého sa akcia môže týkať.

```
class PlayerActionCG
{
      ...
      settery/gettery
      ...

      // typ akcie
      PlayerActionTypeCG actionType;

      // v prípade ak je to PASS_TO je určený smer nahrávky
```

```
        DirCG direction;

        // index spoluhráča, ktorého sa akcia týka
        int teammateIndex;
};
```

## 4.3 Reprezentácia výnosových pravidiel

Heuristické výnosové funkcie (pravidlá) sme implementovali ako obyčajné privátne metódy triedy Player. Aby sme mohli implementovať všetky výnosové funkcie, bolo treba ešte implementovať viacero pomocných funkcií a predikátov rozširujúce schopnosti hráča strednej úrovne:

- **isPassBlocked –** predikát, ktorý rozhodne, či je možné vykonať nábežnú prihrávku na spoluhráča daným smerom, bez toho, aby ju zachytil akýkoľvek protihráč.
  Implementácia:
  - `bool isPassBlockedCG(int playerIndex, int teammateIndex, DirCG passDirection);`

- **isEmptySpace –** predikát, ktorý rozhodne, či je možné vykonať driblovanie daným smerom
  Implementácia:
  - `bool isEmptySpace(int playerIndex, DirCG direction);`

- **uCG –** heuristická funkcia, ktorá priradí výhodnosť – reálne číslo z intervalu <5,7> nábežnej prihrávky daným smerom a k zadanému spoluhráčovi.
  Implementácia:
  - `double uCG(int teammateIndex, DirCG passDirection);`

- **isInFrontOfGoalCG –** predikát, ktorý vyhodnotí, či sa hráč nachádza pred bránkou súpera vo výhodnej polohe na realizáciu streľby na bránku.
  Implementácia:
  - `isInFrontOfGoalCG(VecPosition pos);`

## 4.4 Prideľovanie rolí hráčom

V každom cykle hráčom sú hráčom priradené role pomocou metódy `getPlayerRoles()` spôsobom opísaným v predchádzajúcej časti. Na určovanie najkratšieho možného času dobehnutia hráča k lopte používame upravenú Newtonovu metódu. Podrobný analytický opis tejto metódy je možné nájsť v odbornom článku Stone P. , McAllester. D. : „An Architecture for Action Selection in Robotic Soccer“. Všetky možné role, ktoré je možné priradiť hráčom sú implementačne špecifikované vymenovaným typom:

```
typedef enum { ACTIVE_PASSER=0, ACTIVE_INTERCEPTOR, RECEIVER, PASSIVE, GOALIE,
NO_ROLE } RoleCG;
```

## 4.5 Algoritmus eliminácie premenných

Hľadanie optimálnej spoločnej akcie spĺňajúcich Nashovu rovnováhu s ohľadom na heuristiku špecifikovanú výnosovými funkciami realizujeme pomocou algoritmu eliminácie premenných. Uvedený algoritmus je realizovaný metódou `getOptimalCommonAction()`. Počas letného semestra sa nám podarilo implementovať prvú časť algoritmu (dopredný prechod - vyriešenie

všetkých lokálnych koordinačných závislostí), druhá časť (spätný prechod – nájdenie globálnej optimálnej spoločnej akcie) nebola zatiaľ dokončená. Celkovo sme tak úspešne implementovali prideľovanie rolí pomocou rozšírenej Newtonovej metóde, našli sme spôsob ako reprezentovať výnosové pravidlá, definovali sme kompletnú tímovú stratégiu a logické rozhodovanie, doplnili množstvo chýbajúcich predikátov, našli vhodný spôsob ako reprezentovať všeobecné akcie. Všetky funkcie sú dostupné v implementácií. Aby bolo možné uvedený prístup vysokoúrovňového rozhodovania nasadiť, je nutné ešte dokončiť druhú časť algoritmu eliminácie premenných.

# 5 Pohľad hráča

V hráčovi bola vylepšená schopnosť obzerania sa. Pôvodne hráč zameriaval svoj pohľad stále na loptu pod uhlom 90°. To však spôsobovalo, že hráč nemal často krát dobrý prehľad o dianí vedľa a za hráčom. Jeho model sveta bol často neaktuálny a z toho vyplývali aj zlé rozhodnutia, ktoré robil. Bolo preto nevyhnutné vykonať nasledovné úpravy:

- Upravovať veľkosť zorného uhla hráča podľa potreby.
- Zabezpečiť, aby si hráč neustálim otáčaním hlavy získal čo najlepší prehľad o svojom okolí a nestratil pritom loptu zo svojho dohľadu.

## 5.1 Zorný uhol hráča

Zorný uhol hráča je upravovaný v závislosti od vzdialenosti medzi hráčom a loptou. Pri väčšom zornom uhle vidí hráč väčšiu časť ihriska, ale informáciu o stave na ihrisku dostáva menej často. Hráč, ktorý je ďaleko od lopty ani nepotrebuje častú vizuálnu informáciu. Je skôr preňho dôležitejšie, aby vnímal čo najväčšiu časť ihriska pre prípad, že by sa situácia rýchlo zmenila a on by sa ocitol pri lopte. V takomto prípade by mal možnosť reagovať na základe informácie pozorovanej v minulosti bez potreby obzerania sa.

Zorný uhol hráča v závislosti od vzdialenosti od lopty je preto upravovaný nasledovne:

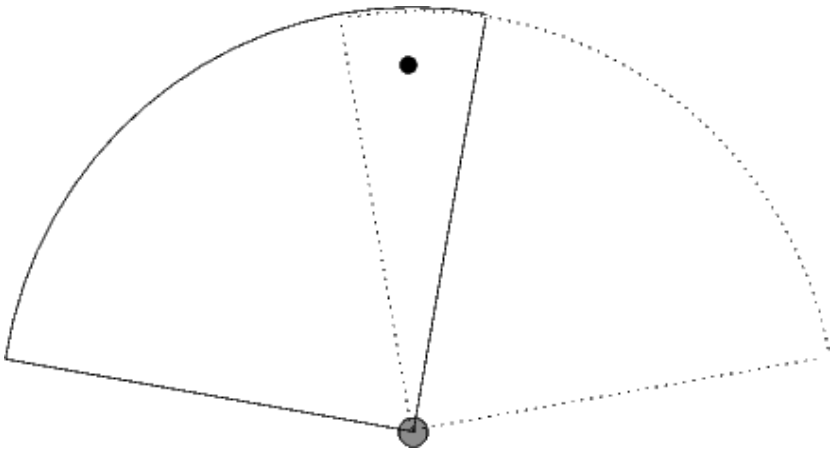
*Tabuľka 1: Závislosť zorného uhla hráča od vzdialenosti lopty*

| Vzdialenosť hráča od lopty | Zorný uhol hráča |
|---|---|
| d <= 1 | 45° |
| 1 < d <= 20 | 90° |
| d > 20 | 180° |

## 5.2 Obzeranie

Aby bola predstava hráča o svete ešte presnejšia, hráč otáča hlavou do strán tak, aby maximalizoval informačný zisk a pritom nikdy nestratil zo svojho dohľadu loptu.

Situácia je znázornená na obrázku . Hráč v závislosti od zorného uhla otáča hlavu doľava a doprava o určitý uhol vzhľadom na loptu. Frekvencia otáčania hlavy je tiež závislá na veľkosti zorného uhla, keďže pri rôznom zornom uhle je ja rôzna frekvencia vizuálnych vnemov. Tieto závislosti sú uvedené v tabuľke .



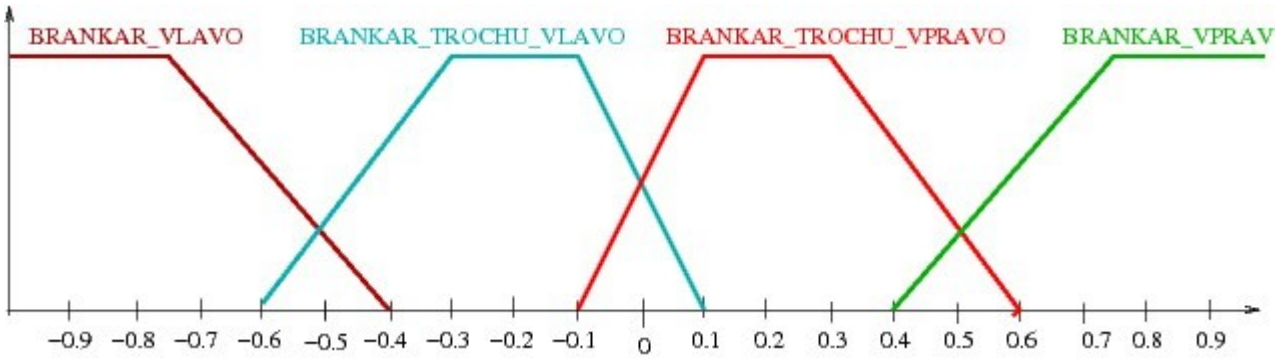
*Obrázok 4: Kmitanie pohľadu hráča*

*Tabuľka 2: Veľkosť otočenia krku a frekvencia v závislosti od veľkosti zorného uhla.*

| **Veľkosť zorného uhla.** | **Veľkosť uhlu otočenia vzhľadom na pozíciu lopty.** | **Perióda prijímania vizuálneho vnemu (ms).** | **Počet cyklov, za ktoré hráč isto obdrží vizuálny vnem - perióda otáčania.** |
|---|---|---|---|
| 45° | 15° | 75 | 1 |
| 90° | 30° | 150 | 2 |
| 180° | 70° | 300 | 3 |

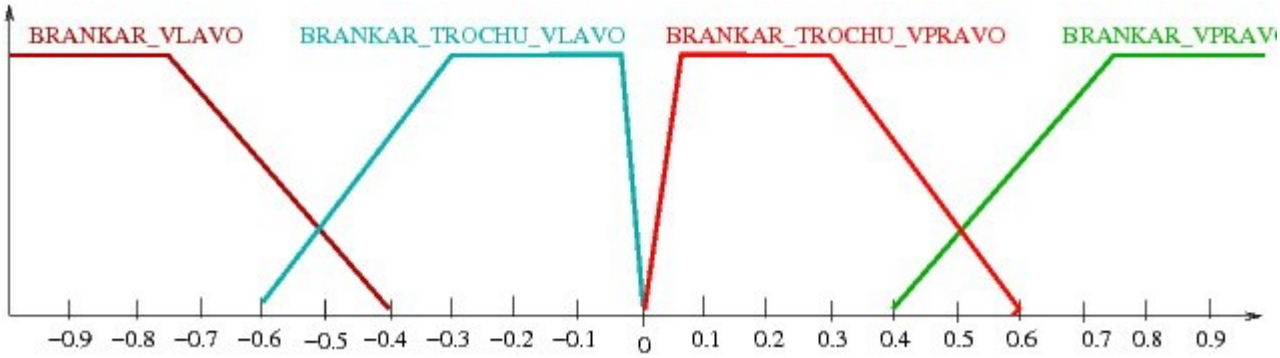
# 6 Úprava pravidiel pre fuzzy regulátor

Oproti prototypu boli upravené hodnoty pravidiel pre fuzzy regulátor, ktoré sa používali na výber najvhodnejšieho smeru na kopnutie lopty. Počas zápasov sa totiž ukázalo, že hráč nie vždy vyhodnotil správne pozíciu brankára, a preto niekedy volil kopnutie buď priamo naňho, alebo cez okolie v jeho tesnej blízkosti. Toto správanie hráča bolo ovplyvnené definíciou charakteristickej funkcie pre fuzzy množiny v lingvistickej premennej IN_Y_GOALIE. Konkrétne išlo o charakteristické funkcie fuzzy množín BRANKAR_TROCHU_VLAVO a BRANKAR_TROCHU_VPRAVO, ktorých priebehy sa na intervale <-1, 1> pretínali (obr.5).




*Obrázok 5: Pôvodné priebehy charakteristických funkcií pre fuzzy množiny jazykovej premennej IN_Y_GOALIE*

V situácií, keď bol brankár len o čosi málo vychýlený na jednu stranu, bral fuzzy regulátor do úvahy viacero pravidiel (viď. Prototyp, tab. 3), ktoré sa svojimi účinkami navzájom veľmi zoslabili, čoho výsledkom bolo kopanie na bránku v smere na brankára alebo do okolia jeho tesnej blízkosti.




*Obrázok 6: Nové priebehy charakteristických funkcií pre fuzzy množiny jazykovej premennej IN_Y_GOALIE*

Tento problém sme sa pokúsili odstrániť predefinovaním priebehov charakteristických funkcií dvoch spomínaných množín. Začiatočný bod charakteristickej funkcie pre fuzzy množinu BRANKAR_TROCHU_VPRAVO a koncový bod charakteristickej funkcie pre fuzzy množinu BRANKAR_TROCHU_VLAVO bol posunutý do bodu 0. Ďalej boli body zlomu pre tieto funkcie presunuté z bodu -1 pre prvú a z bodu 1 pre druhú charakteristickú funkciu presunuté do bodu -0,03

pre prvú a 0,03 pre druhú. Nové priebehy týchto dvoch charakteristických funkcií sú zobrazené na obrázku 6. Tým sa teda zväčšil vplyv týchto množín v situácií, keď je brankár len mierne vychýlený zo svojej stredovej polohy, a pravidlá rozhodujúce sa na základe týchto množín sa už navzájom nerušia.

# 7 Prihrávky

## 7.1 Prehľad

Počas testovania prihrávania boli postupne vyskúšané a vylepšené viaceré „teórie“ pravidiel, podľa ktorých sa hráč rozhoduje komu a kde prihrať.

### 7.1.1 Prvá verzia prihrávok

V 1. verzii hráča boli nahrávky implementované nasledovným algoritmom:

1. Nájdi hráča, ktorý je najbližšie ku stredu súperovej bránky.
2. Ak taký nie je, nájdi hráča, ktorý je najbližšie ku tebe.
3. Ak ani taký nie je, tak kop na bránku
4. Vypočítaj počet cyklov *PC1* ako najkratší čas potrebný na prejdenie lopty ku vybranému hráčovi.
5. Vypočítaj pozíciu *POS1*, kde bude hráč za počet cyklov *PC1* pomocou predikčnej funkcie
6. Nahraj hráčovi na pozíciu *POS1*

Pri testovaní sa ukázalo, že prediktívne funkcie sú príliš „optimistické“ a príliš predkopávajú hráčovi. Dochádzalo tak k strate lopty z týchto dôvodov:

- hráč nestihol so svojou únavou dobehnúť za loptou
- hráč stihol zmeniť smer behu a teda lopta šla úplne „mimo“
- hráč pre príliš veľkú vzdialenosť stratil záujem o loptu

### 7.1.2 Druhá verzia prihrávok

Z dôvodov uvedených v predchádzajúcej časti sa prihrávanie hráča upravilo spriemerovaním aktuálnej a budúcej pozície. Došlo tak ku:

- kompenzácii nepresnosti v modele sveta
- kompenzácii nevedomosti o unavenosti hráča
- zvýšeniu záujmu hráča o loptu.

Váhovanie sa realizuje s využitím počtu cyklov *PC1*, kde pôvodné súradnice reprezentované ako vektor *[X,Y]* sa násobia počtom cyklov a spočítajú s pozíciou *POS1* reprezentovanou ako vektor. Výsledné súradnice sa normujú počtom *cyklov + 1*.

Napriek jednoduchosti prihrávok a neuvažovaniu súpera pri nahrávaní podával uvedený algoritmus dostatočne dobré výsledky.

## 7.2 Prihrávky pomocou okolia hráčov

### 7.2.1 Princíp popisu okolia hráča

Pri prihrávkach s využitím okolia hráčov ide o jednoduchý model, kde sa na základe vzdialenosti určuje „osobný priestor hráča“ a je popísaný pomocou kruhu:

- všetko vo vnútri kruhu je oblasť, kde je hráč schopný loptu zachytiť
- všetko mimo kruhu je oblasť, kde hráča netreba uvažovať

Ak zvolíme „lúče“, ktoré reprezentujú smer budúcej prihrávky lopty v určitom rozostupe *RL* v 360° rozsahu a prehľadáme priesečníky kružníc (ktoré reprezentujú uvedené kruhy), dostávame tak zoznam hráčov, ktorých pri danej nahrávke musíme uvažovať.

Problémy, ktoré treba vyriešiť sú:

- výber najvhodnejšieho lúču
- výpočet sily, ktorou loptu kopnúť
- polomer kruhu, ktorý použiť

### 7.2.2 Prvá verzia nahrávok

V prvej verzii nahrávok sa polomer kruhu počíta v lineárnej závislosti od vzdialenosti hráča ako

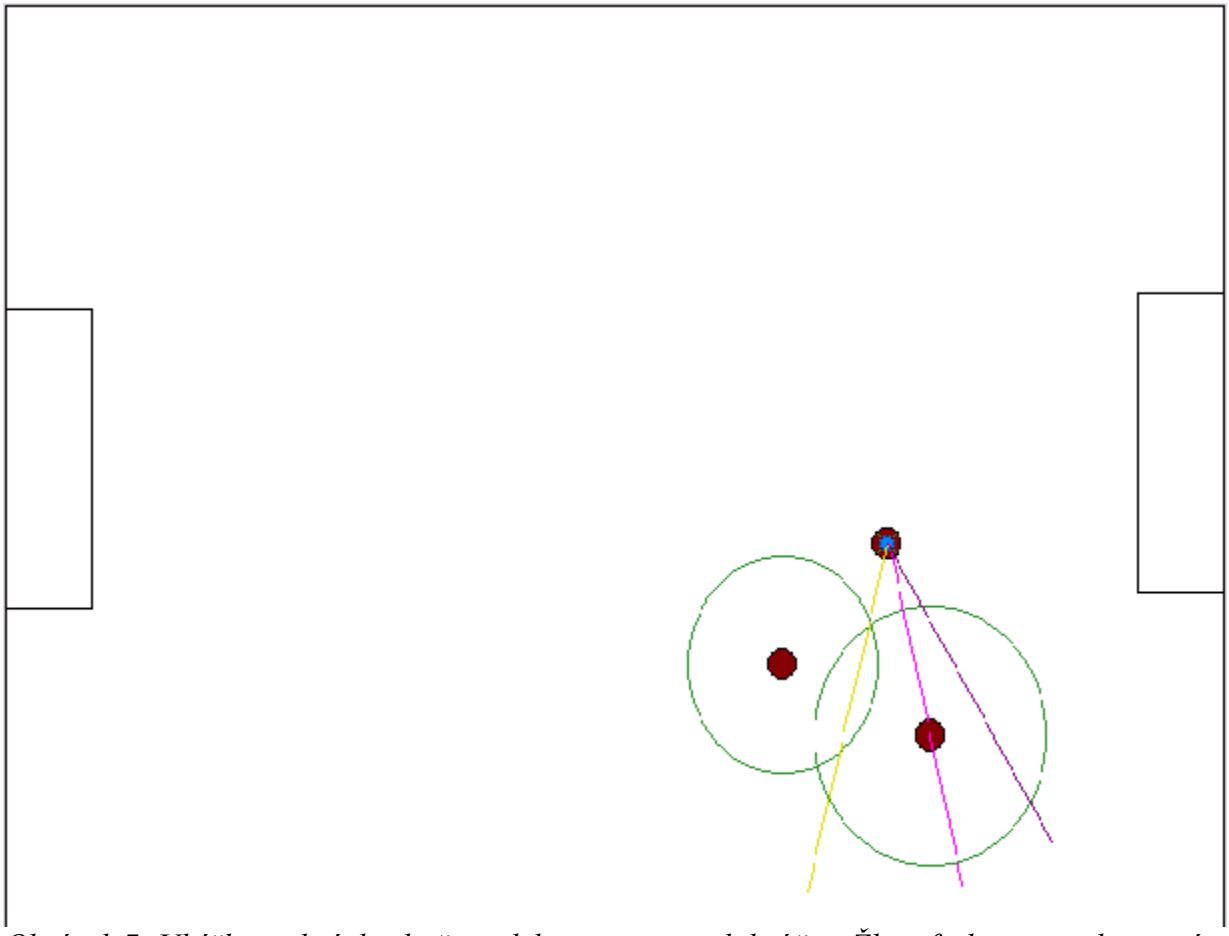
$$r = vzdialenosť / 3$$

Výber najlepšieho lúča prebieha pomocou fitnes funkcie, ktorá sumuje ohodnotenie cez všetkých hráčov, ktorí sa uvažujú. Spoluhráči sa počítajú ako prírastok, protihráči naopak znižujú celkové fitnes lúča. Do úvahy sa berie:

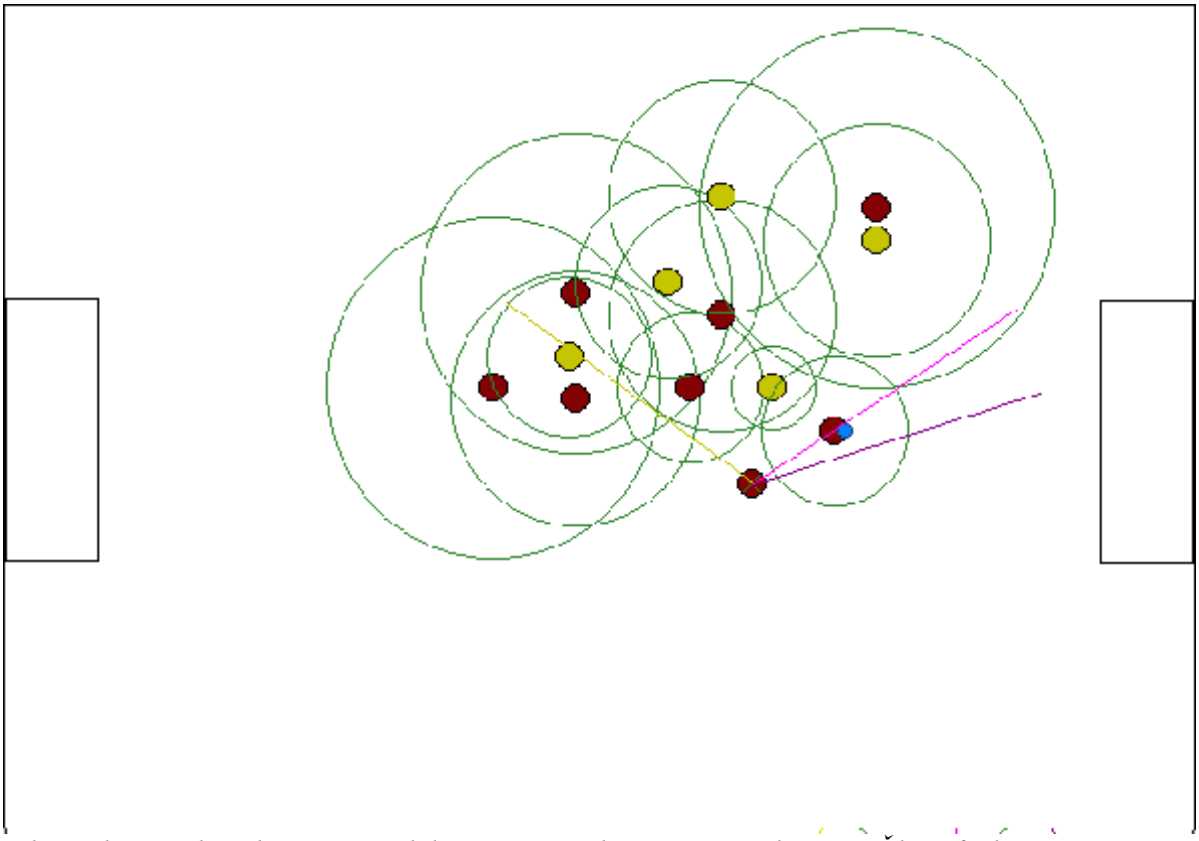
- *vzdialenosť hráča.* Bližšia je *K* krát lepšia, kde *K* je číslo v rozsahu 1-10
- *vzdialenosť hráča od súperovej bránky.* Bližšie je *L* krát lepšie, kde *L* je číslo v rozsahu 1-3
- *vzdialenosť lúča od hráča*. Bližšie je lepšie, t.j. z dvoch rovnocenných lúčov vedľa seba sa vyberie ten bližší (znižuje riziko straty lopty)

Táto verzia nahrávok (v poradí už tretia) podáva lepšie výsledky ako predchádzajúce, pretože berie do úvahy aj súperov. Napriek tomu má niekoľko nepríjemných vlastností:

1. Hráč nahráva tam, kde je zvýšená koncentrácia spoluhráčov, čo má za následok prihrávanie útočníkov stredopoliarom alebo obrancom späť ku vlastnej bránke (viď obrázok 2.1)
2. Hráč uprednostní aj prihrávku priamo na súpera, pokiaľ je v jeho okolí dostatočný počet spoluhráčov (obrázok 2.2)



*Obrázok 7: Ukážka prihrávky do časti, kde je viacej spoluhráčov. Žltou farbou je realizovaná prihrávka, ružovou je dobrá nahrávka, fialovou je najideálnejšia nahrávka*



*Obrázok 8: Nahrávka v smere, kde je zvýšená koncentrácia hráčov. Žltou farbou je zvýraznený smer nahrávky, ružovou ideálny smer, fialovou najideálnejší*

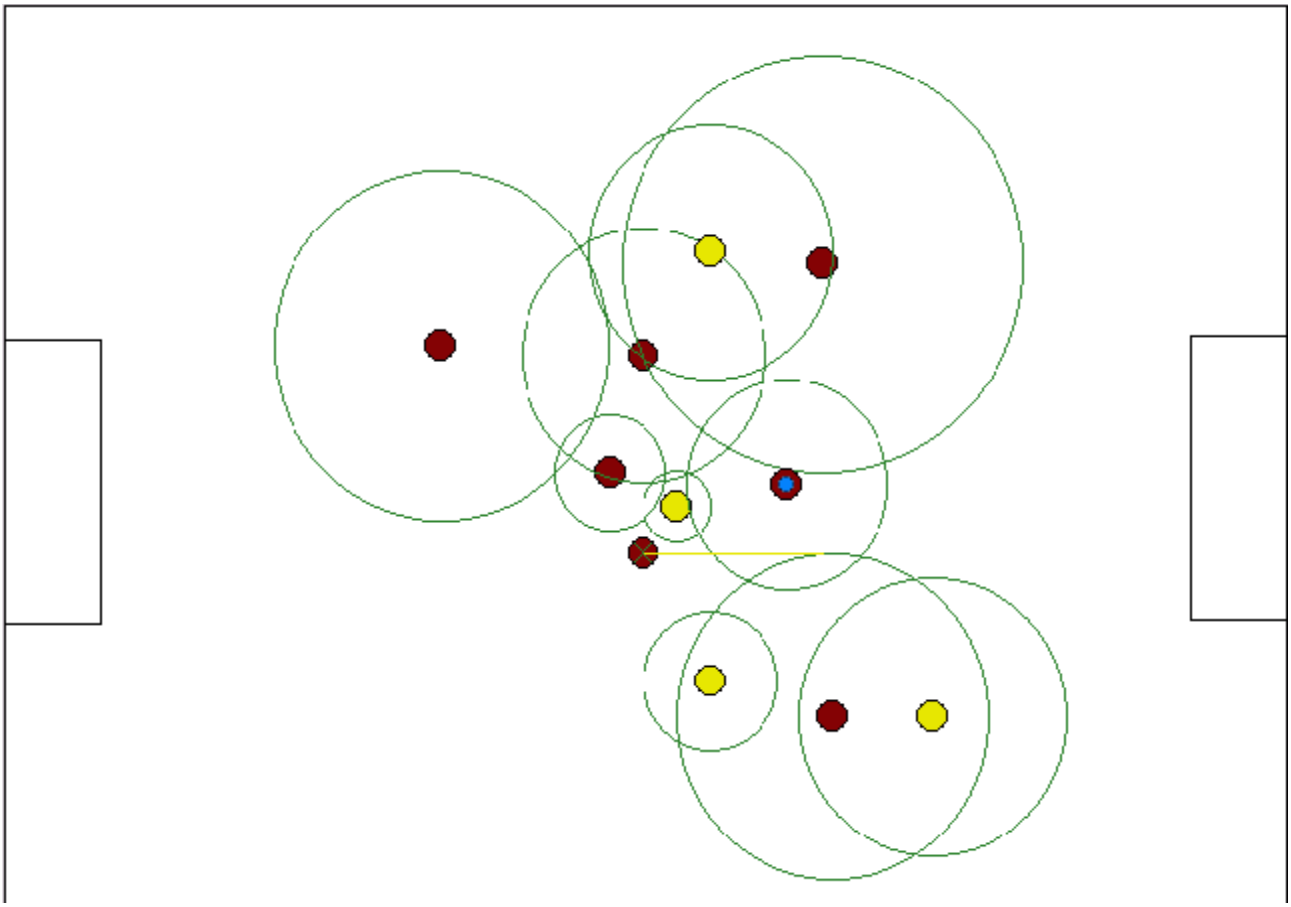
### 7.2.3 Druhá verzia nahrávok

Pri analýze prvej verzie nahrávok sa došlo k záveru, že nie je vhodné ohodnocovať lúč kvôli hráčom v jeho okolí, ale na základe najlepšieho hráča, s tým, že v závislosti od rastúcej vzdialenosti k súperovej bránke sa bude zvyšovať ohodnotenie lúčov smerujúcim k súperovej bránke. Takto by nemalo dojsť k nahrávke súperovi pred vlastnou bránkou.

Algoritmus prechodu cez lúč je nasledovný:

1. Nájdi najlepšie ohodnoteného spoluhráča, ktorý je v smere prihrávky
2. Nájdi najlepšie ohodnoteného protihráča, ktorý je v smere prihrávky, ale takého, že nie je ďalej ako najlepší spoluhráč (lúč má určitú dĺžku a je zbytočné rátať protihráčov, ktorí „nezavadzajú“)
3. Pre spoluhráča pripočítaj bonus, alebo potrestanie za nahrávku dopredu/dozadu

Na obrázku 2.3 je vidno typicky realizovanú nahrávku. Hráč už neuprednostňuje zvýšenú koncentráciu, ale jednotlivého hráča (v čom hráči zvýšenej koncentrácie zlyhávajú kvôli súperovi v ich smere)



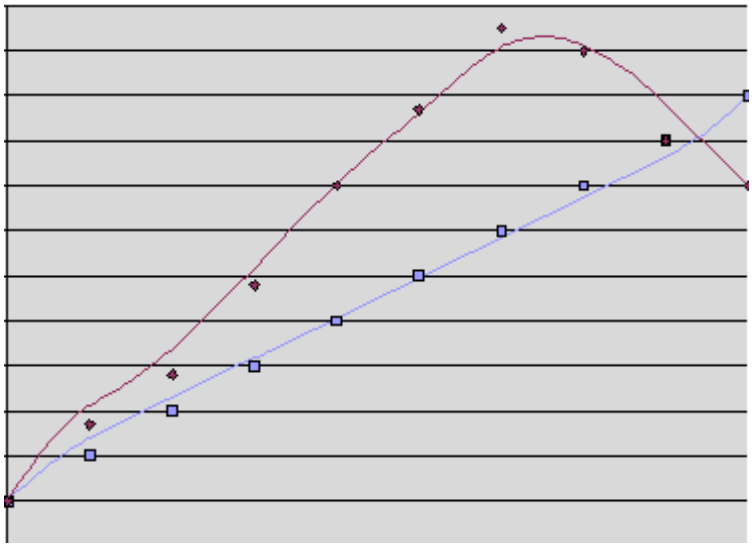
*Obrázok 9: Typická nahrávka hráča s prihrávkami 4. verzie – uprednostňuje nahrávku k súperovej bránke*

Fitnes funkcia pre lúč je daná týmto vzorcom:

Pre spoluhráča sa fitnes obohacuje ešte o zložku vzdialenosti od bránky:

Pre protihráča sa fitnes iba násobí (predpokladá sa, že súper je lepší):

Na obrázkoch 2.1 a 2.2 sú kreslené smery lúčov. Smer ružovej farby – ideálny smer - zodpovedá aktuálne dosahovaným výsledkom. Na to aby sa dosiahol smer fialovej farby – najideálnejší smer, ktorý predkopáva - je potrebné upraviť vzorec na výpočet *FITNESS* tak, aby menovateľ nebol lineárne závislý od vzdialenosti hráča a lúču, ale aby mal tvar krivky posunutej v závislosti od vzdialenosti hráča (ktorá je súčasne aj v čitateli zlomku). Ukážka je na obrázku 2.4.



*Obrázok 10: Ukážka obídenia linearity zapracovaním nelineárneho prírastku v závislosti od vzdialenosti lúču a hráča*

Vrchol krivky (lokálne maximum) by sa posúval v smere osi *X* v prípade, že hráč je ďalej (treba viacej predkopnúť) a proti smeru osi *X* v prípade, že je hráč bližšie.

## 7.3 Finálna podoba Prihrávok

Na základe výsledkov a pozorovaní priebehu zápasov s viacerými tímami (Stjupit dox, FIIT Media, Squirell squadron) boli prihrávky doladené tak, aby sa dali ľahšie konfigurovať na konkrétne tím, resp. aby sa dalo vytvoriť rozhranie na učenie nastavenia koeficientov. Jednotlivé funkcie majú nasledovný tvar:

Fitnes funkcia pre lúč:

Zložka pre spoluhráča:

Zložka pre protihráča:

Všetky funkcie sú vo výsledku osekávané na interval (0, 12). Výsledné správanie je také, že hráč preferuje dlhšie nahrávky po určitú hranicu, ktorú zabezpečuje funkcia *Nelinearita*, ktorá spôsobí, že od vzdialenosti 20 sa fitnes za vzdialenosť mierne znižuje. Výsledná nahrávka takisto nikdy nie je presne na hráča, ale silne závisí od vzdialenosti 2 hráčov - čím sú hráči ďalej, tým sa viac výsledný smer posúva na okraj kruhu.

## 7.4 Rozhranie pre koordinačné grafy

Z dôvodu potreby ohodnotenia prihrávky pri koordinačných grafoch bolo potrebné vytvoriť podporné funkcie na určenie hodnoty lúča. V koordinačných grafoch funkcia *u* na nahrávku závisí od hráča, ktorému má smerovať a smeru (uhlu) lopty. Po refactoringu kódu sa dá priamo využívať.

Bol stanovený interval hodnôt, ktoré môže fitnes funkcia vrátiť v rozmedzí 0-12. Pri volaniach sa využívajú funkcie na konvertovanie relatívnych smerov () na absolútne, pričom lopta prejde v danom smere, resp. lopta je prihrateľná spoluhráčovi, ak fitnes funkcia pre daný smer a hráča má min. hodnotu *FC*.

# 8 Driblovanie - aplikácia Q-learningu

Vo finálnej implementácií nie je zahrnutý q-learning pre driblovanie. To bolo spôsobené časovou tiesňou pri finalizácií riešenia. Predpokladom v prvej časti projektu bolo využitie knižnice RIL. Keďže napokon q-learning nebol implementovaný, podpora bola odstránená, aby bola zlepšená prehľadnosť zdrojových kódov a zjednodušenie kompilačného procesu.

Aktuálna implementácia driblovania využíva pevné rozhodovacie pravidlá a neobsahuje adaptívne prvky.

Pre potreby aplikácie q-learningu bola pridaná podpora na lepšiu identifikáciu stavového priestoru a podpora pre neskoré vyhodnocovanie akcie. Podrobnosti sú opísané v časti kapitoly opisujúcej implementáciu uvedených vlastností.

## 8.1 Metóda Q-learning

Metóda q-learning vychádza z predpokladu, že hráč sa pohybuje v stavovom priestore. Prechody medzi jednotlivými stavmi sú označované akcie.

Jednotlivé stavy v priestore je možné ohodnotiť. Cieľom agenta využívajúceho tento stavový priestor je prechádzanie medzi stavmi, ktoré majú čo najlepšie ohodnotenie. Problémom býva často krát asociácia akcia-stav. Akciu nie je možné jednoznačne prepojiť s prechodom do určeného stavu. Toto je dané charakterom stavového priestoru, v ktorom je metóda q-learningu aplikovaná.

V prípade driblovania sme predpokladali využitie q-learningu na zlepšenie práce s loptou pri rôznych rýchlostiach. Stavový priestor je futbalový zápas. Akcie sú príkazy, ktoré môže hráč vykonať. Výstupy týchto príkazov sú ovplyvnené pravdepodobnostným modelom futbalu, ale aj rušením zo strany servera. Iným rušením je aj prítomnosť iných hráčov, ktorý daný stav z pohľadu agenta nedeterministicky ovplyvňujú.

Q-learning v prípade driblovania prispôsobí správanie hráča tak, aby zohľadňoval pravdepodobnostný model futbalu, svoj vlastný stav a správanie sa okolitých hráčov.

Predpokladom pre aplikáciu q-learningu je existencia spôsobu ako ohodnotiť prechod do nového stavu. Hodnota je vyjadrená premennou, ktorá hovorí odmenu v čase, kde čas je počet stavov, ktoré už agent prekonal. Táto premenná je označovaná  .

Výber akcií prebieha na základe nasledujúceho algoritmu:

```
start: vyber akciu a prejdi na dalsi stav, t=t+1
     odmena = 
     aktualizuj odhad - uprav q-funkciu
goto start
```

Výber akcie prebieha na základe maximálneho ohodnotenia vyberanej akcie q-funkciou. V prípade vykonania dostatočného počtu prechodov sa hodnota generovaná q-funkciou bude približovať reálnemu ohodnoteniu nasledujúceho stavu.

Učenie je realizované na základe vzorca:

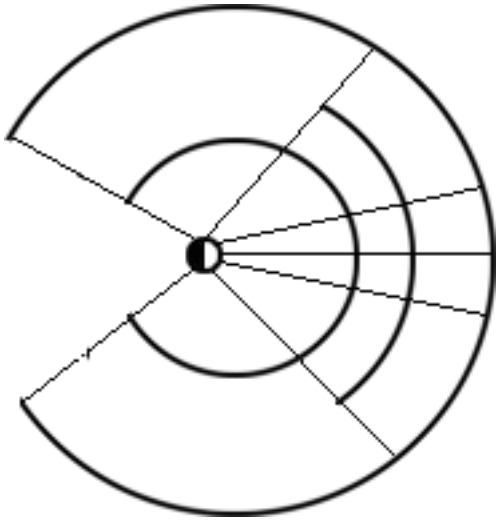
Toto učenie sa vykonáva v *t+1* kroku. Konštanty a ovplyvňujú rýchlosť učenia.

## 8.2 Návrh aplikácie Q-Learningu pre driblovanie

Stavový priestor je tvorený premennými:

- Stamina
- Aktuálna rýchlosť hráča a lopty
- Výskyt protihráčov v sektoroch
- Výskyt spoluhráčov v sektoroch
- Výskyt lopty v sektoroch

Sektory okolo hráča sú plochy okolo hráča, kde sa môže lopta nachádzať, sú znázornené na nasledujúcom obrázku



*Obrázok 11: Navrhovaná segmentácia priestoru okolo hráča*

Výstupná akcia mala popisovať rýchlosť a smer kopnutia lopty.

Vyhodnocovanie stavu je v prostredí Robocup pomerne zložité. Problematická je identifikácia vhodného stavu. Nie je možné brať stav ako ekvivalent k novému cyklu. Je to dané aj tým, že hráč môže vyhodnotiť v ďalšom cykle nevhodnosť driblovania a použiť úplne inú akciu – kopnutie na bránu, prihrávku a i. Ďalším problémom je, že hráč neprichádza do kontaktu s loptou každý cyklus.

Navrhovaný spôsob vyhodnocovania predpokladal zaznamenávanie stavu v cykloch a určenie medzných miest, ktoré sú využité na vyhodnotenie stavu. Návrh funkcie bol založený na vyhodnocovaní nasledovných premenných:

- *Určenie presnosti driblovania.* Toto je stanovené, či hráč dobehol hru po predikovanom počte cyklov a lopta sa nachádzala na mieste, ktoré hráč predikoval.

- *Stanovenie, či hráč ovláda loptu po predikovanom počte cyklov*

Vyhodnotenie stavu vyžaduje neskoré vyhodnocovanie. Podpora pre neskoré vyhodnocovanie stavu bola pridaná do aktuálnej verzie hráča.

# 9 Driblovanie

Vylepšené bolo v hráčovi taktiež driblovanie. Boli doplnené rôzne smery a tri typy driblovania, medzi ktorými sa hráč môže rozhodovať. Jeho rozhodnutie medzi daným driblovaním sa robí v metóde *handleWithBall()* na základe troch funkcií: *canDribbleFast(), canDribbleSlow() a canDribbleWithBall()*. Táto je volaná iba vtedy, ak je kladne vyhodnotená funkcia *canDribbleWithBall()*. V ďalších podkapitolách sú tieto funkcie spoločne s funkciou *isFreeCone()* podrobnejšie opísané.

## 9.1 Funkcia canDribbleFast()

Funkcia rozhoduje, či môže hráč driblovať najrýchlejším spôsobom. Ak áno, vracia sa aj smer jeho driblovania. Na jej kladné vyhodnotenie musí hráč spĺňať tieto podmienky:

- Musí mať danú hodnotu energie a danú rýchlosť, aby bol schopný za loptou po jej predkopnutí bežať.
- Musí sa nachádzať na súperovej polovici ihriska. Rýchle driblovanie je totiž nebezpečné a môže viesť k strate lopty.
- Musí byť otočený smerom na súperovu bránu, lebo toto driblovanie neposkytuje možnosť rýchleho otočenia sa na opačnú stranu a driblovanie na vlastnú bránu nemá veľký význam a môže byť nebezpečné.
- V jeho okolí sa nesmú nachádzať protihráči.

## 9.2 Funkcia canDribbleSlow()

Táto metóda podobne ako predchádzajúca rozhoduje o možnosti driblovania daným typom. Týmto typom je pomalé driblovanie, ktoré síce neposkytuje najväčsiu kontrolu nad loptou alebo otáčanie sa s ňou, ale je dostatočne razantné a nie tak nebezpečné ako rýchle driblovanie. Na povolenie tohto driblovania musí hráč zase spĺňať niektoré podmienky:

- Musí mať danú hodnotu energie a danú rýchlosť, aby bol schopný za loptou po jej predkopnutí bežať.
- Musí byť otočený smerom na súperovu bránu, lebo toto driblovanie taktiež neposkytuje možnosť rýchleho otočenia sa na opačnú stranu. Driblovanie na vlastnú bránu nemá veľký význam a môže byť nebezpečné.
- V jeho okolí sa nesmú nachádzať protihráči.

## 9.3 Funkcia canDribbleWithBall()

Táto funkcia rozhoduje o tom, či môže hráč driblovať najpomalším spôsobom. Tento je používaný pri driblovaní dozadu, driblovaní na vlastnej polke, pri otáčaní alebo rozbiehaní. Preto je tento typ u hráča aj najčastejšie využívaný. Funkcia sa okrem toho využíva aj na rozhodovanie o spustení funkcie *handleWithBall()*, čím sa rozhodne o driblovaní hráča vôbec. Na kladné vyhodnotenie sú

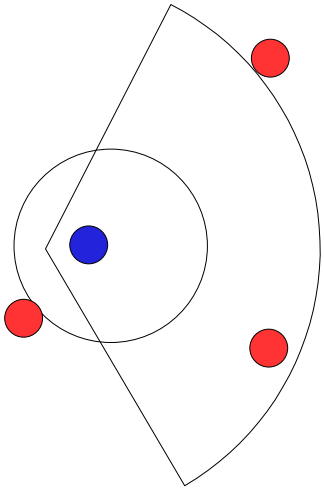
podobne ako pri predchádzajúcich funkciách potrebné dané podmienky:

- Hráč musí mať danú hodnotu energie (nie však rýchlosť, ktorú práve nadobúda týmto driblovaním), aby bol schopný za loptou po jej predkopnutí bežať.
- V jeho okolí sa nesmú nachádzať protihráči.

Výstupom funkcie je podobne ako pri predošlých metódach číslo, ktoré hovorí, či môže hráč driblovať. Ak áno, udáva aj smer jeho driblovania. Môže hráčovi povedať, aby dribloval rovno, viac doľava alebo doprava. Okrem týchto troch smerov, ktoré sú používané aj v predošlých funkciách, udáva dodatočne *handleWithBall()* i priamy smer ťahania na bránu.

## 9.4 Funkcia isFreeCone()

Na zisťovanie toho, či sa v hráčovom okolí nenachádzajú žiadne prekážky, sa využíva funkcia *isFreeCone()*. Táto ako vstup berie dva parametre – smer hráča, ktorým dribluje a polomer kruhového výseku. Výstupom je rozhodnutie *true* alebo *false* na základe toho, či sa v daných výsekoch nachádza prekážka.



*Obrázok 12: Driblovanie hráča s loptou.*

Funkcia ako prvé vypočíta polohu kruhu, ktorý sa nachádza pred hráčom a spočíta v ňom protihráčov. Ak sa v ňom nejakí nachádzajú, vracia okamžite funkcia hodnotu *false*. V opačnom prípade sa pokračuje vo vyhodnocovaní. Je vypočítaná poloha kruhového výseku (na základe vstupných parametrov) a zisťuje sa, či sa v ňom nachádzajú protihráči alebo postranné čiary. Ak tu nebol nájdený ani jeden z týchto objektov, funkcia vracia *true*, čím je driblovanie povolené. Náčrt tohto procesu je zobrazený na obrázku 12.

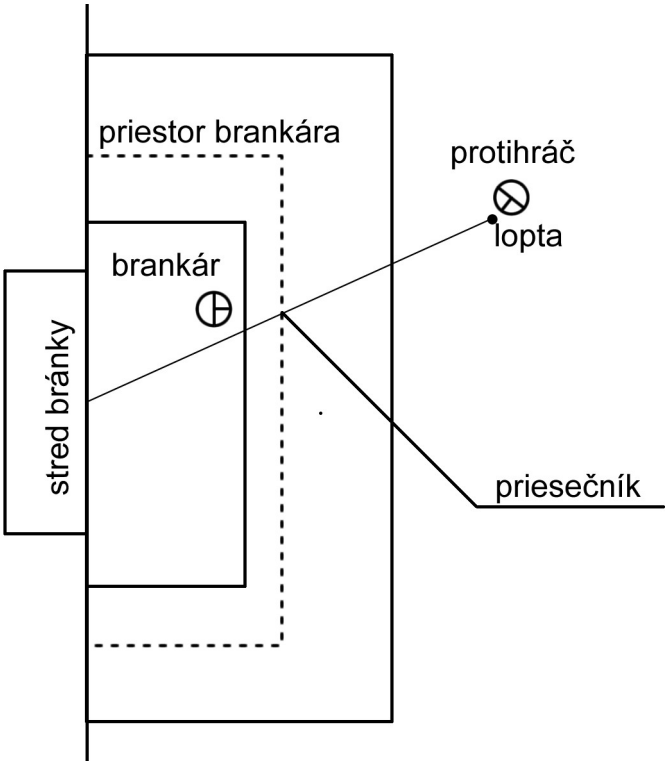
# 10 Brankár

V pôvodnej verzii UvA Trilearn bol zahrnutý brankár ktorý spĺňal základnú funkcionalitu. Bola však implementovaná dostatočná a upravená existujúca.

V existujúcej verzii bol pohyb brankára obmedzený obdĺžnikom okolo bránky (je medzi bránkoviskom a jedenástkou). Brankár nemôže vybehnúť mimo tento obdĺžnik okrem jediného prípadu a to je vybehnutie po lopte ktoré bolo pridané nami.

Brankár vybehne po lopte z bránky ak je lopta najviac 4 metre od neho je bližšie ako 1,5 metra od protihráča a letí pomalšie ako 1 metre za sekundu. Teda lopta je predkopávaná pre spoluhráčom.

Ďalej bol upravený výpočet kde by mal stáť brankár. Výpočet prebieha v každom cykle. X-sová súradnica pozície sa počíta pomocou všeobecnej funkcie hráčov na výpočet strategickej pozície (závisí od zvolenej formácie a absolútnej pozície lopty v hracom poli). Y-sová súradnica pozície bránkara sa počíta sa rovná priesečníku priamky lopta-stred bránky a priamky ktorá je prednou čiarou obdĺžnika ktorý obmedzuje pohyb brankára. Situácia je znázornená na Obrázok 13: Umiestnenie brankára




*Obrázok 13: Umiestnenie brankára*

Brankár si celý čas drží natočenie tela smerom ako keby letela lopta smerom na bránku. Ale uhoľ natočenia hlavy (a teda pohľadu) je vždy nasmerovaný na loptu. Pohľad sa rozširuje a zužuje podobne ako pri hráčoch (na základe vzdialenosti).

Brankárovi tiež bola pridaná schopnosť vrhnúť sa po lopte. Ak je lopta v catchable area (dva metre pred brankárom) brankár sa po nej hodí.

# Prílohy

## *A. Bibliografia*

## *B. Zoznam tabuliek*

## *C. Zoznam obrázkov*