Evaluering av neurala nätverk för en fotbollsspelande mobil robot

(HS-IDA-EA-02-102)

Daniel Andersson (a99danan@student.his.se)

Institutionen för datavetenskap
Högskolan i Skövde, Box 408
S-54128 Skövde, SWEDEN

Examensarbete på programmet för systemprogrammering under vårterminen 2002.

Handledare: Nicklas Bergfeldt

Evaluering av neurala nätverk för en fotbollsspelande mobil robot

Examensrapport inlämnad av Daniel Andersson till Högskolan i Skövde, för Kandidatexamen (B.Sc.) vid Institutionen för Datavetenskap.

2002-06-07

Härmed intygas att allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och att inget material är inkluderat som tidigare använts för erhållande av annan examen.

Signerat: _____

Evaluering av neurala nätverk för en fotbollsspelande mobil robot

Daniel Andersson (a99danan@student.his.se)

Sammanfattning

Detta examensarbete behandlar ett experiment som Tom Smith utförde vid sitt magisterarbete, att utveckla en Kheperarobot som ska utföra en fotbollsuppgift av enklare modell. Dock koncentrerar sig detta arbete mer på en evaluering av artificiella neurala nätverk för detta problem. De olika typerna av ANN-arkitekturer som har använts till detta arbete är förutom Tom Smiths, baserade på arkitekturer från en artikel skriven av Stefano Nolfi.

De resultat som har uppnåtts visar att precis som i Stefano Nolfis artikel är det en arkitektur med ”spontan modularitet” som visar sig fungera bäst av de arkitekturer som undersökts, även till detta problem.

Nyckelord: Evolutionär robotik, Genetiska algoritmer, Artificiella neurala nätverk, Robotsimulering, Khepera, Kheperafotboll, YAKS.

Innehållsförteckning

Figurförteckning

Formelförteckning

Diagramförteckning

1. Introduktion

> Some people believe football is a matter of life and death. I'm very disappointed in that attitude. I can assure you its much, much more important than that. (Bill Shankley, manager Liverpool FC 1959-1974).

Under 1950-talet ansåg forskare att konstruera en dator som kan spela schack skulle bli en framtida milstolpe för forskning inom Artificiell Intelligens (**AI**). Den 11 Maj 1997 utspelades en match mellan IBM´s dator Deep Blue och den mänskliga regerande schackmästaren Kasparov och Deep Blue lyckades med bedriften att besegra Kasparov. Medför vinsten för Deep Blue att den ska betraktas som en intelligent maskin? Många forskare har inte den åsikten, istället började de söka efter ett nytt problemområde som ställer högre krav på en utvecklad AI. September 1993 fastslogs det officiellt på en konferens i Japan att utveckla robotar med uppgiften att kunna spela fotboll skall betraktas som den nya utmaningen inom AI-forskningen, då det enligt Østergård, Lund & Pagliarini (2000) krävs att en fotbollsspelare skall kunna:

- Arbeta i en dynamisk omgivning. En schackspelare behöver bara arbeta i en statisk omgivning, spelbrädet förändras ej under sitt eget drag.
- Arbeta med ofullständig information om sin omgivning. En schackspelare har tillgång till all information som finns på spelplanen.
- Ha någon grad av förkroppslighet. Detta krävs ej för en schackspelare då dess enda uppgift är att tänka ut sitt nästföljande drag. Till exempel så säger Pfeifer och Scheier att: ”Intelligence cannot merely exist in the form of an abstract algorithm but requires a physical instantation, a body.” (Pfeifer & Scheier, 1999).

Några forskare tror till och med att robotar kommer att kunna vinna en match mot de regerande mänskliga mästarna i fotboll innan år 2050 (Østergård, Lund & Pagliarini, 2000; Kitano, 1997).

För att konstruera en robot som skall klara av att spela fotboll krävs det att roboten har förmågan att kunna anpassa sig. Enligt Meeden & Kumar (1998) skall roboten klara av att utföra en samling grundförmågor som kan användas till en mängd olika uppgifter av varierande karaktär i dynamiska miljöer. Exempel på grundförmågor kan vara att undvika hinder, känna igen olika föremål och kunna manipulera olika föremål. Det klassiska sättet att förse robotar med dessa grundförmågor sker genom att en programmerare implementerar dessa för egen hand, detta innebär att programmeraren själv måste förutse alla tänkbara situationer som kan uppstå för roboten. Enligt Niklasson & Ziemke (1996) finns det speciellt två typer av problem då det blir för komplext för en mänsklig programmerare att klara av att programmera roboten. Den ena typen av problem är där det inte finns någon expertis som kan precisera en modell för hur problemet ifråga kan lösas och den andra är problem där det finns expertis men där sambanden mellan olika parametrar är oklara eller svåra att formalisera. Det är speciellt det senare problemet som utgör svårigheten vid skapandet av en fotbollsspelande robot.

För att komma ifrån detta problem har forskning kring robotar börjat behandla lösningar som baseras på att roboten själv evolverar fram det beteende som krävs för

att klara den önskvärda uppgiften. Den mänskliga utvecklaren behöver endast utveckla det evolutionära ramverk som krävs för roboten, medans själva robotarkitekturen evolveras fram automatiskt.

För att få ett så bra resultat som möjligt för roboten så vore det bästa enligt Meeden & Kumar (1998) att roboten evolveras i den riktiga världen. Ett stort problem med att låta roboten evolvera fram beteenden i den riktiga världen är att processen kan ta väldigt lång tid, enligt Meeden & Kumar kan en enkel simulering ta flera dagar att genomföra. Ett vanligt sätt att komma runt detta är att skapa en simuleringsmiljö för roboten som är så lik den riktiga världen som möjligt och där processen för att evolvera fram en robot kan snabbas upp.

Detta arbete kommer att utforska en gren inom evolutionär robotik som visat sig vara väldigt kraftfull (Smith 1997). Denna teknik baserar sig på idéer som framtagits inom den biologiska forskningen för djur och människors sätt att lösa problem. Denna evolutionära process kännetecknas av att använda sig av en kombination av artificiella neurala nätverk och genetiska algoritmer. Speciellt kommer det att utvärderas olika artificiella neurala nätverk för att se hur stor betydelse olika nätverksarkitekturer har för robotens förmåga att utföra en bestämd uppgift, i detta fall att spela fotboll.

2. Bakgrund

Detta kapitel ger en introduktion till evolutionär robotik, Kheperaroboten, YAKS-simulatorn och de relaterade arbetena.

I kapitel 2.1 kommer det ges en kort introduktion hur evolutionär robotik fungerar och en beskrivning av dess två stora huvudområden artificiella neurala nätverk och evolutionära algoritmer. Därefter kommer en beskrivning av hur Kheperaroboten är uppbyggd och hur den fungerar, detta tas upp i kapitel 2.2. För att kunna simulera Kheperarobotar används i detta arbete YAKS-simulatorn som kommer att beskrivas i kapitel 2.3 och tillslut kommer även de relaterade arbetena presenteras och även dess relevans för detta arbete.

2.1. Evolutionär robotik

Under många års tid har forskare försökt att konstruera robotsystem som kan utföra de sysslor som vi människor vill undvika att göra, som till exempel diska och städa. De problem som uppstår vid konstruering av dessa system är inte enkla att lösa, roboten skall kunna arbeta i en dynamisk miljö medan den utför ett icke-monotoniskt arbete. Det mål som robotforskare strävar efter är att kunna få roboten autonom, vilket innebär att den klarar av att anpassa sig efter omgivningen, klarar av att underhålla sig själv och att den skall klara av att utföra komplexa uppgifter på egen hand (Østergård, 2000).

Inom evolutionär robotik, som är en metod för att konstruera autonoma robotar, så används idéer som har framtagits genom biologisk forskning. Termen evolutionär robotik är en term som har introducerats relativt nyligen men idén med att låta en robots kontrollsystem representeras som en artificiell kromosom har funnits sedan slutet av 1980-talet (Cliff, Harvey & Husband, 1993). Metoden fungerar genom att skapa en population av robotar med slumpvis framtagna kontrollsystem. Robotarna får sedan fritt agera i miljön medan de utvärderas i hur pass bra de utför vissa bestämda uppgifter. De robotar som klarar sig bäst får vara med och skapa nästa generation av robotar. Genom att iterera denna process kommer robotarna att evolveras fram till det önskvärda beteendet (Nolfi, 1998).

Den stora fördelen med att använda evolutionär robotik till att utveckla lämpliga beteenden till robotar är att systemet utvecklas i sin helhet av en självstrukturerande process. Detta innebär att utvecklaren eller designern som mest behöver skapa det ramverk som krävs för processen, som till exempel fitnessfunktioner och ANN-strukturer (Nolfi & Floreano, 2000).

De två stora områden som behandlas inom evolutionär robotik är artificiella neurala nätverk och evolutionära algoritmer, vilka kommer att beskrivas mer utförligt i kapitel 2.1.1 – 2.1.2.

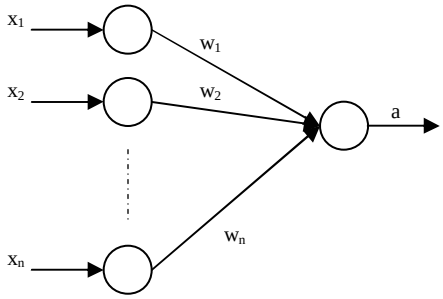
2.1.1. Artificiella neurala nätverk

Många uppgifter som involverar intelligens eller mönsterigenkänning är extremt svårt att automatisera, men dessa uppgifter förefaller som att de kan utföras väldigt enkelt av djur och människor. För att klara av dessa uppgifter använder sig djur och människor av sin hjärna och det är resultat inom forskning om hjärnan som har

resulterat i artificiella neurala nätverk (**ANN**). Det mesta av informationen om ANN är hämtad ifrån Mehrotra, Mohan & Ranka (1997).

Artificiell neuron

För att förstå hur ett ANN fungerar behövs det en matematisk förståelse för hur en enkel artificiell neuron hanteras.

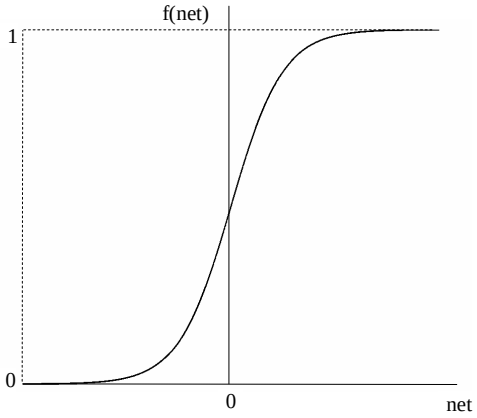


Figur 1. En generell neuron.

Varje artificiell neuron består av *n* antal inlänkar och en utlänk som kan kopplas till ett godtyckligt antal andra neuroner, alternativt så agerar utlänken som en utnod för nätverket. Beräkningar i varje nod sker i 2 steg:

1. Först summeras alla multiplikationer av inlänkar med dess vikter. Detta görs med formeln $net = \Sigma w_i * x_i$, där x_i är värdet ifrån inlänk i och w_i är vikten för inlänk i (se figur 1).
2. Det *net*-värde som beräknades i steg 1 räknas om med hjälp av en aktiveringsfunktion till ett aktiveringsvärde för utlänken ($a = f(net)$).

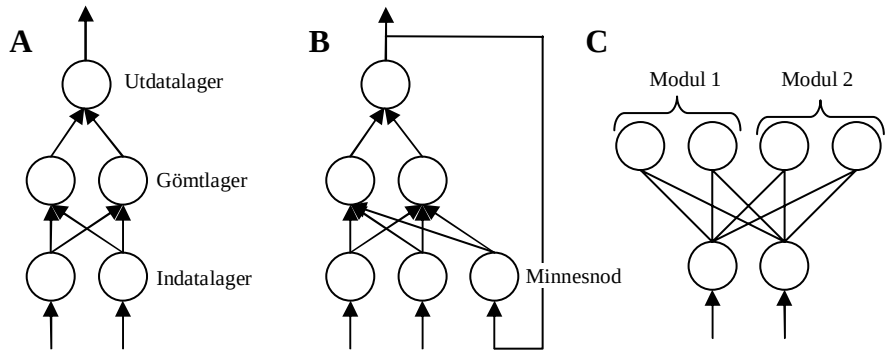
I detta arbete kommer endast en aktiveringsfunktion att användas, sigmoidfunktionen. Detta är den mest populära aktiveringsfunktionen inom ANN och då speciellt i backpropagation-algoritmer då den är kontinuerlig och deriverbar i varje punkt. Sigmoidfunktionen känns lätt igen på dess karakteristiska utseende då den är s-formad (se figur 2).



Figur 2. Sigmoidfunktion.

Artificiella Neurala Nätverk

På egen hand kan en enkel neuron uträtta väldigt lite enligt Østergård (2000). Det är när de kopplas samman i så kallade nätverksarkitekturer med flertalet noder som användbarheten ökar. De arkitekturer som kommer att användas i detta arbete baserar sig på 3 olika grundarkitekturer, framåtmatat nätverk, återkopplat nätverk och modulärt nätverk (se figur 3).



Figur 3. A: Framåtmatat nätverk, B: Återkopplat nätverk, C: Modulärt nätverk.

Det som gör ANN så fördelaktigt att använda som kontrollarkitektur i en robot är enligt Nolfi (1996) att:

- ANN är tolerant mot störningar, detta finns det gott om i en robots omgivning.
- ANN består av låg-nivå primitiver vilket är att föredra vid evolutionära metoder. Detta för att till hög grad undvika beslut av mänsklig design.
- ANN kan enkelt exploatera varierande former av inlärning under dess livstid, vilket kan snabba upp den evolutionära inlärningsprocessen.

2.1.2. Evolutionära algoritmer

Evolutionära algoritmer ger ett lite annorlunda sätt att lösa problem jämfört med de mer klassiska metoderna som till exempel olika sökalgoritmer. Evolutionära algoritmers största fördel är dess enkelhet och dess mångsidighet.

Grundprincipen för en evolutionär algoritm ser ut som följande och är hämtad ifrån Fogel & Michalewicz (2000):

```
procedure evolutionary algoritm
begin
     t<-0
     initiera P(t)
     evaluera P(t)
     while not (terminerings-villkor) do
     begin
            t<-t+1
            plocka ut P(t) ifrån P(t-1)
            förändra P(t)
            evaluera P(t)
     end
end
```

Grundprincipen fungerar så att det skapas en population av individer *P(t)* för varje iteration *t*, där varje individ representerar en potentiell lösning på problemet. Varje individ evalueras sedan för att få ett värde på hur pass bra den potentiella lösningen är för individen, ett så kallat fitnessvärde. En ny population skapas sedan vid iteration *t+1* genom att de individerna med bäst fitness plockas ut från iteration *t*. Några av individerna får sedan genomgå en viss förändring genom olika operatorer (*Se genetiska algoritmer nedan*) för att skapa nya individer. Algoritmen kommer att iterera tills ett terminerings-villkor blir uppfyllt som kan vara att en tillräckligt bra lösning är funnen eller att tiden har tagit slut.

Inom evolutionära algoritmer finns det olika typer av algoritmer, exempel på dessa är genetiska algoritmer, genetisk programmering, evolutionär programmering och evolutionär strategi. Det som skiljer de olika algoritmerna åt är hur individerna är representerade och hur individerna förändras. I detta arbete kommer det endast att koncentreras på den genetiska algoritmen då det enligt Smith (1997) tillsammans med ANN blir en väldigt kraftfull problemlösningsmetod.

Genetiska algoritmer

Det som är utmärkande för genetiska algoritmer (**GA**) är hur individerna är representerade och hur individerna blir förändrade. De förändringsoperatorer som används inom GA är till exempel mutation och kombinering av två individer. Mutation fungerar så att individen kommer att genomgå en slumpmässig förändring i sin kromosom (*se nedan)* och en kombinering fungerar så att två föräldraindivider får producera två nya individer med material från båda föräldrarna (Koza, 1992).

Det finns enligt Whitley (1995) tre huvudområden för användning av GA tillsammans med ANN:

1. Evolvera fram vikter i en statisk ANN-arkitektur.
2. Evolvera fram topologier i ett ANN, till exempel hur många gömda noder som skall användas i nätverket.
3. För att välja ut träningsdata och tolka beteende i utdata.

I detta arbete kommer GA användas till att evolvera fram vikter i ett statiskt ANN, alltså huvudområde 1.

För att förstå hur GA fungerar behövs några av de termer som används kännas till. Dessa termer kommer ifrån Carlsson (1999):

Kromosom	Består av en sträng där variabler kodas. I detta fall lagras vikterna inom ett ANN.
Individ	En kromosom med tillhörande fitnessvärde (se nedan).
Population	En bestämd mängd individer.
Avkomma	En ny individ som ett resultat av mutation eller kombinering av två individer.
Fitness	Ett mått på hur bra en specifik individ löser problemet.
Fitnessfunktion	En funktion som avgör vilken fitness en individ skall få.

GA arbetar enligt Koza (1992) genom att först skapa en slumpartad population av ett bestämt antal individer, där varje individ har ett fitnessvärde associerat till sig ifrån en

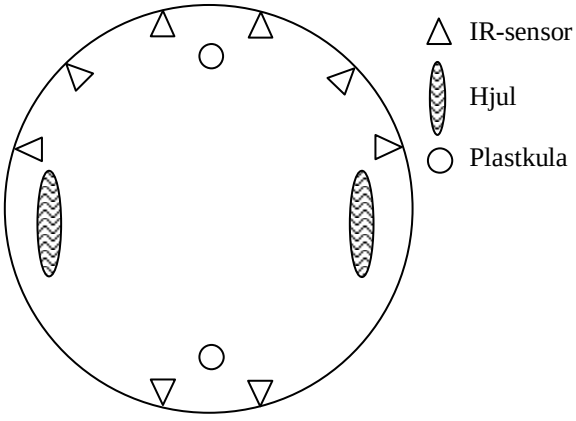
speciell fitnessfunktion. Utifrån denna population av individer skapas sedan en ny generation av individer och detta kan göras med olika tekniker. I detta arbete kommer en teknik som heter turneringsbaserat urval att användas då den enligt Mitchell (1996) är mer beräkningseffektiv och har bra stöd för parallella simuleringar vilket är att föredra då simuleringarna är en väldigt tidskrävande process. En annan orsak till att turneringsbaserat urval var den selektionsalgoritm som valdes till detta arbete är att det finns stöd för denna i YAKS-simulatorn (se kapitel 2.3) som är den robotsimulator som användes.

Denna teknik fungerar genom att två individer slumpvis väljs ur populationen. Därefter genereras ett slumptal mellan 0 och 1 och är slumptalet mindre än en fördefinierad parameter kommer individen med högst fitness att få vara med till nästa generation, annars kommer individen med lägst fitness att fortsätta till nästa generation. I turneringsbaserat urval (men också i andra tekniker) är det ofta smart att dessutom bevara de bästa individerna för nästa generation. Detta kallas elitism och garanterar att de bästa lösningarna som har skapats hittills inte går förlorade. När den nya populationen är skapad så skapas avkommor till denna genom mutation eller genom kombinering av två individer.

Genom att låta detta arbetssätt fortlöpa under ett förutbestämt antal generationer så kan individerna evolveras mot det önskvärda beteendet.

2.2. Khepera

Khepera är en robot ämnad speciellt för robotforskning och utvecklas av K-team (K-team). Kheperaroboten är cylinderformad med en diameter på 55mm, höjd på 30mm och en vikt på 70g. Som drivning har Kheperaroboten 2st likströmsmotorer som driver varsitt hjul och som stöd finns 2 plastkulor. Kheperaroboten är även utrustad med 8 stycken infraröda sensorer, 6 stycken på framsidan och 2 stycken på baksidan (se figur 4).



Figur 4. Översikt av Kheperarobotens sensor- och hjulplacering.

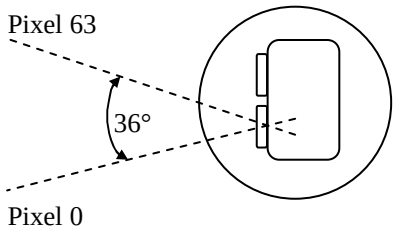
Roboten kontrolleras av en Motorola 68331 processor, har inbyggt minne för lagring av kontrollprogram och strömförsörjningen kan ske med antingen batteri eller via en seriell kabel. Används den seriella kabeln till roboten så kan en dator även utnyttjas som extra lagringskapacitet och beräkningsresurs.

Enligt Smith (1997) så är Kheperaroboten särskilt bra lämpad som en forskningsrobot då:

- Roboten används redan inom forskningen och därmed har många dokumenterade experiment utförts.
- Det finns insticksmoduler att tillgå vilket gör att roboten kan byggas på med de moduler som krävs för uppgiften. Detta gör att roboten kan anpassas enkelt för olika typer av uppgifter utan att roboten blir onödigt komplex. Exempel på moduler är griparm och kameratorn.
- Roboten är liten till storleken och därmed tillåts att experiment kan utföras på en liten yta som till exempel ett vanligt bord.

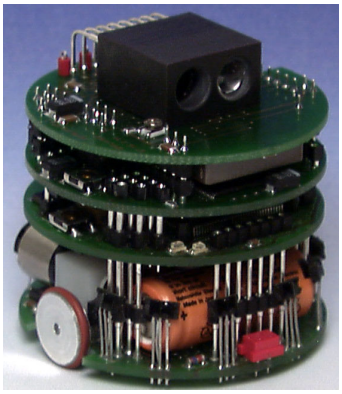
2.2.1. K213 Kameratorn

Kameratornet är en modul till Khepera som gör att robotens funktionalitet byggs på med en gråskalig kamera. Kameratornet kontrolleras av en Motorola 68HC11 processor och kommunicerar genom ett lokalt nätverk med robotens huvudprocessor. Kameran består av 2 sensorer, en bildsensor och en ljussensor.



Figur 5. Översiktsbild av kameratorn.

Bildsensorn består av en rad med 64 ljuskänsliga celler som ger en endimensionell linjär bild på 64 pixlar där varje pixel ger ett värde för gråskalan (0-255) och de 64 pixlarna kommer att ge en synvinkel på ca 36^{o} (se Figur 5). Cellerna laddar upp sig i det ljus som finns i omgivningen och beroende på hur starkt ljuset är så sker detta olika snabbt. Tiden det tar för cellerna att ladda upp sig kan variera emellan 2-300ms. Eftersom cellerna är ljuskänsliga kan de även överexponeras om de inte laddas ur i tid, för att förhindra att detta sker så finns den andra sensorn. Ljussensorn ser alltså till att bildsensorn inte överexponeras. Kameratornet är designat för att kunna lokalisera föremål i omgivningen som befinner sig emellan 5-50 cm framför roboten. Hur roboten ser ut i verkligheten med kameratornet monterat kan ses i figur 6.



Figur 6. Verklig bild av Khepera med Kameratorn.

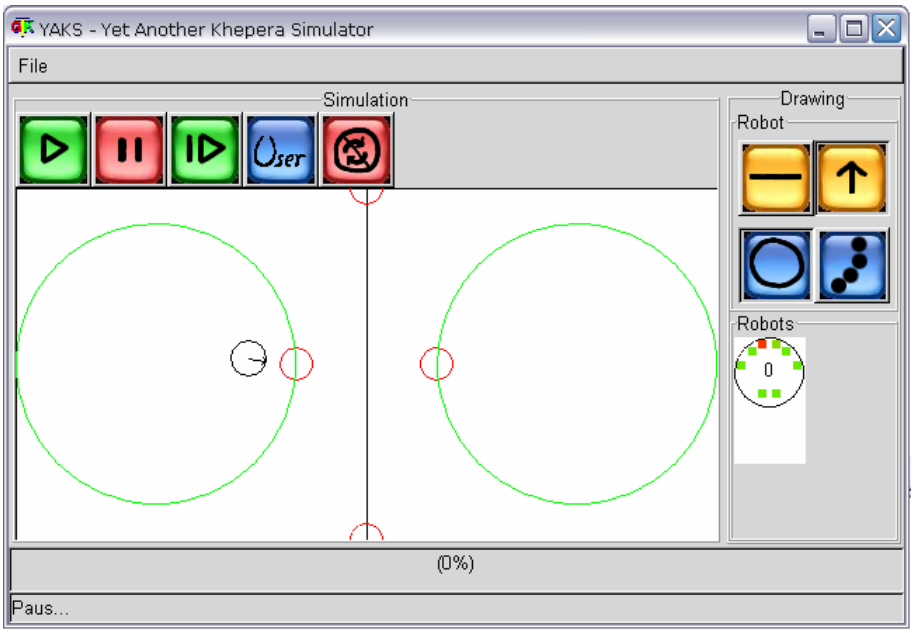
2.3. YAKS

Den Kheperasimulator som används i detta projekt är *Yet Another Khepera Simulator* (**YAKS**). YAKS är utvecklad på Högskolan i Skövde som ett led i deras forskning inom lärande robotar och robotutveckling genom användning av ANN (Carlsson & Ziemke, 2001).

YAKS utvecklades med målen att:

- Koden till robotsimulatorn, ANN´s och utvecklings/inlärningsalgoritmer skall vara helt skilda moduler. Det skall alltså gå att använda en modul utan de andra.
- Funktionaliteten i robotsimulatorn skall vara ekvivalent med den funktionalitet som finns i Kepsimsimulatorn. Detta för att det lätt skall gå att till exempel utbyta simulatormiljöer mellan dessa.
- Simulatorn skall ha tillgång till kraftfulla verktyg för 2D-visualisering och analys av robotbeteende såväl som de underliggande ANN-mekanismerna.
- Grafik och visualisering skall inte påverka tidslängden i en simuleringsprocess.
- Simulatorn skall vara portbar till flera operativsystem, åtminstone Unix, Linux och MS Windows.
- All kod och dokumentation skall finnas tillgänglig för allmänheten.

YAKS ger möjligheten att simulera ett oändligt antal Kheperarobotar i en virtuell miljö med oändlig storlek. Den virtuella miljön som robotarna simuleras i kan bestå av väggar, ljuskällor, speciella målzoner och runda hinder som kan vara stationära eller flyttbara. Än så länge kan YAKS simulera de infraröda sensorerna, ljussensorerna och en mycket förenklad funktionalitet av K213 kameratorn. Det grafiska gränssnitt som används i YAKS kan ses i figur 7 nedan och där visas några olika primitiver som kan simuleras. De gröna cirklarna är så kallade målzoner, de röda cirklarna är stationära runda hinder och den svarta cirkeln är roboten.



Figur 7. YAKS grafiska gränssnitt.

2.4. Relaterade arbeten

I huvudsak baseras detta arbete på två andra arbeten, det ena utförde Tom Smith till sin magisterexamen och behandlar i huvudsak det experiment som kommer att utföras. Det andra arbetet utförde Stefano Nolfi och behandlar en utvärdering av olika ANN-arktitekturer.

2.4.1. Adding vision to Khepera: an autonomous robot footballer

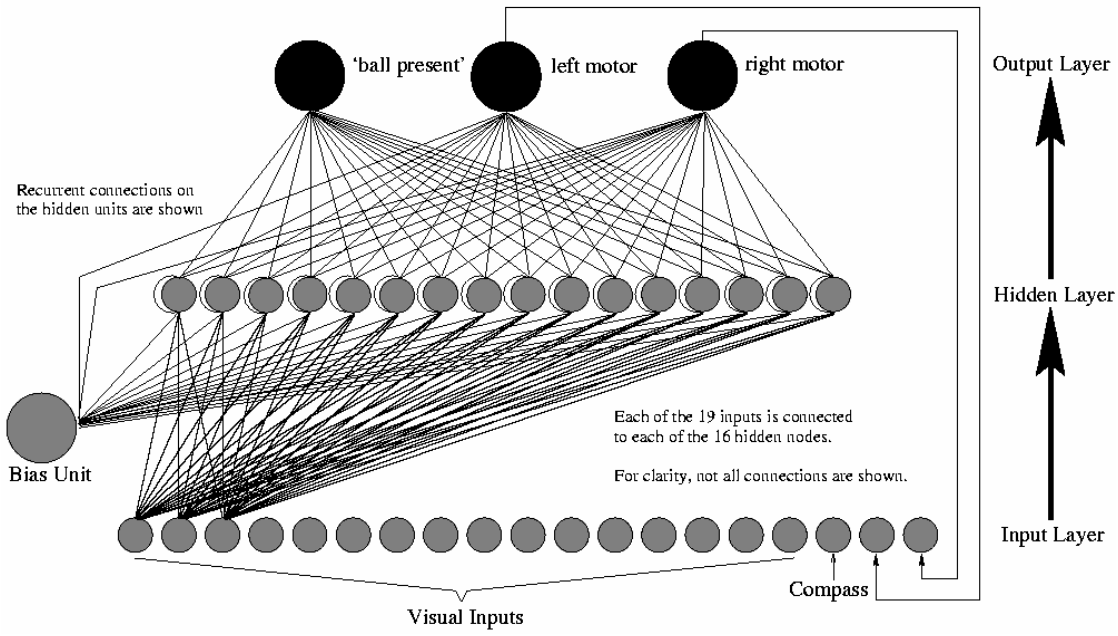
Detta arbete utförde Tom Smith som sitt examensarbete för sin magisterutbildning 1997 (Smith 1997). Smith utförde två experiment med hjälp av Kheperarobotar, varav det andra experimentet är av intresse för detta arbete.

Experimentet gick ut på att låta en fotbollsspelare, representerad av en Kheperarobot, att ensam på en fotbollsplan, utan vare sig försvarare eller målvakt mot sig, lyckas lokalisera en boll och försöka på något sätt att få denna boll i ett speciellt mål. Målet med detta experiment var att utvärdera tre olika fitnessfunktioner för den evolutionära processen till detta experiment. De fitnessfunktioner som Smith evaluerade var:

1. En funktion som endast värderar hur nära bollen var till målet.
2. En funktion som Smith kallar en utökande version där den i början endast värderar hur bra den ser bollen, sedan efter ett antal generationer så värderar den endast hur ofta roboten träffar bollen och tillslut värderar den endast hur nära bollen hamnar målet.
3. Den tredje versionen utnyttjar alla aspekter som beskrivs i den andra funktionen fast under hela processen.

Det resultat som Smith kom fram till var att det var den tredje funktionen som fungerade bäst för det evolutionära processen.

Det mest intressanta för det egna arbetet är ändå ANN-arkitekturen som användes (se figur 8) och varför den är intressant förtydligas i problembeskrivningen (se kapitel 3).

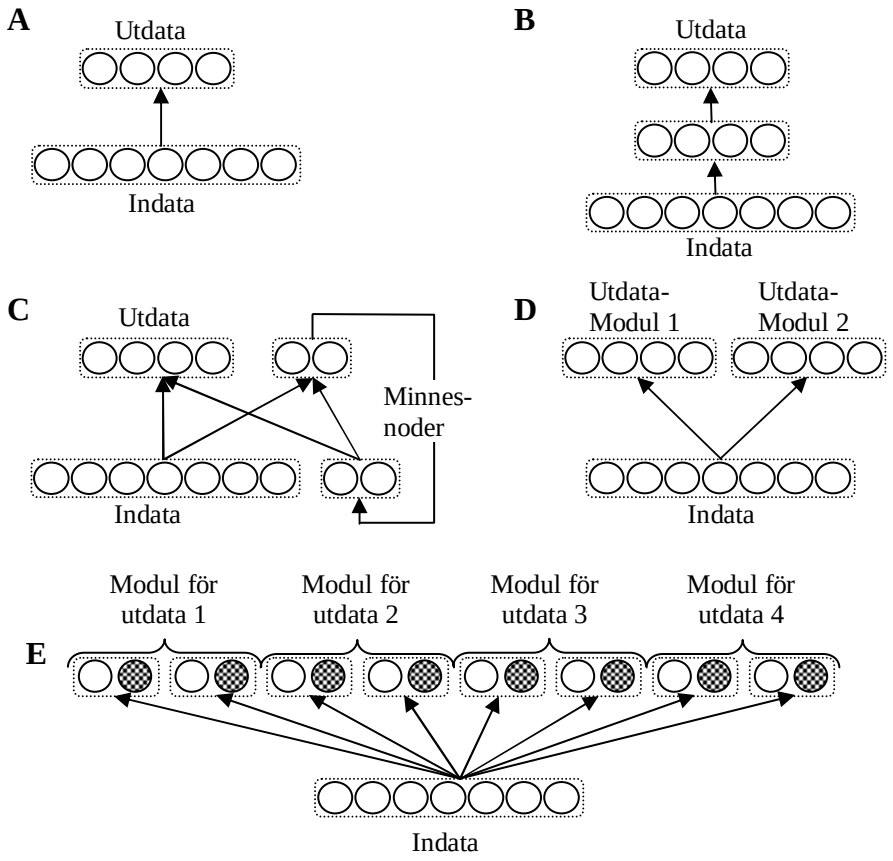


Figur 8. Smiths ANN arkitektur (Smith, 1997, s. 48).

2.4.2. Using emergent modularity to develop control systems for mobile robots

Stefano Nolfi har gjort en utvärdering av fem olika ANN-arkitekturer i sin artikel Nolfi (1996). Utvärderingen görs i ett experiment där en Kheperarobot har som uppgift att i ett område plocka upp skräp-bitar och lägga dessa utanför områdets gränser. Kheperaroboten var utrustad med en gripklo för att kunna plocka upp skräp-bitarna.

Det intressanta från detta arbete är de tre olika grundtyperna av ANN-arkitekturer som användes och evaluerades, framåtmatade nätverk (se figur 9, arkitektur A och B), återkopplande nätverk (se figur 9, arkitektur C) och modulära nätverk (se figur 9, arkitektur D och E). Det som Nolfi kom fram till var att den ANN-arkitektur som använde sig av spontan modularitet (arkitektur E i figur 9) var den klart bästa arkitekturen för att hantera komplexa uppgifter som kräver väldigt olika respons vid liknande sensorvärden.



Figur 9. Nolfis fem olika ANN-arkitekturer (efter Nolfi, 1996, s.5).

3. Problembeskrivning

De teoretiska/matematiska egenskaperna för vanliga ANN-arkitekturer är relativt väl förstådda i detta skede. Enligt Ziemke & Carlsson (1999) så säger tyvärr inte det så mycket om den praktiska tillämpningen för olika ANN-arkitekturer vid användning för olika uppgifter och hur pass bra mottaglighet arkitekturerna har för träning med olika typer av träningsalgoritmer.

Detta arbete baserar sig i huvudsak på det arbete som Smith (1997) utförde men med en liten annan infallsvinkel. Detta arbete koncentrerar sig på det ena experimentet, att evolvera fram en robot som skall kunna, ensam på en fotbollsplan, på ett eller annat sätt få en boll i ett specifikt mål. Men, istället för att utvärdera olika fitnessfunktioner, som redan Smith har utfört, skall istället en utvärdering av olika ANN-arkitekturer genomföras. I Smiths experiment lyckades roboten göra mål cirka 25 % av försöken när den individen med högst fitness efter 3000 generationer användes. Att använda sig av samma fitnessfunktion som Smith använde med lite olika ANN-arkitekturer i grunden kommer att visa hur pass stor betydelse arkitekturerna har för robotens prestanda.

Enligt Nolfi (1996) så har en ANN-arkitektur med spontan modularitet fördel vid komplexa uppgifter och det visade sig stämma i hans egna experiment. Denna information skall kunna utnyttjas även i detta experiment då komplexiteten kommer att öka för ANN-arkitekturerna då kameratornet appliceras. De olika typer av arkitekturer som utvärderas är, förutom den typ av arkitektur som används av Smith (1997), baserade från Nolfi (1996).

För att utvärdera de olika arkitekturerna kommer robotarna att evalueras i YAKS-simulatorn i en omgivning som kommer att likna den som presenteras i reglerna för Kheperafotboll (*se bilaga A*). Dessa regler skiljer sig en aning ifrån de som Smith använde till sitt arbete, Smith´s fotbollsplan hade svarta väggar med gråa målzoner istället för de gråa väggarna med svarta målzoner som används i detta arbete. Att använda sig av svarta väggar har den nackdelen enligt Smith att IR-sensorerna inte fungerar 100-procentigt, den svarta färgen har förmågan att absorbera IR-ljuset istället för att reflektera och därmed kan Kheperaroboten få felaktig information av sensorerna. Men eftersom Smith valde att inte utnyttja IR-sensorerna till sitt fotbollsexperiment, vilket han gjorde till sitt andra experiment, kommer troligtvis denna skillnad inte påverka resultatet.

Ett delproblem som uppstår är att YAKS inte har stöd för att kunna simulera en fotbollsspelande robot. För tillfället finns det bara stöd för en mycket förenklad funktionalitet av Kheperarobotens kameratorn och det saknas stöd för att kunna simulera en boll. Hur tillvägagångssättet gick till för att förbättra eller skapa dessa aspekter kan ses i bilaga B.

3.1. Avgränsningar

För att det skall vara möjligt att hinna med experimenten inom den tidsram som är utsatt så krävs det att en del avgränsningar på problemet görs. I detta fallet har valts att:

- Ingen verifiering av resultaten på en riktig Kheperarobot kommer att genomföras, detta kommer inte att genomföras då det till exempel inte finns någon tillgång till en fotbollsplan för Kheperarobotarna här på skolan.

- Smiths fitnessfunktion kommer att användas utan att någon undersökning kommer att genomföras om hur pass bra den egentligen utför sin uppgift. Detta kan vara en nackdel då fitnessfunktionen är väldigt viktig för att evolutionär robotik skall fungera på ett korrekt sätt.
- De GA-parametrar som används kommer att vara baserade på liknande arbeten, detta innebär att de troligen inte är helt optimerade för detta problem. Att optimera dessa är en väldigt tidskrävande process som det inte finns tid till i detta arbete.

3.2. Hypotes

Genom att använda det resultat som Nolfi (1996) kom fram till i sitt experiment och speciellt då Nolfis teori om spontan modularitet där han påstår att ”…the advantage of the emergent modular architecture will increase with increasing task complexity” (Nolfi, 1996, s. 13). Med detta i åtanke och med det faktum att Smith (1997) inte motiverade varför hans typ av arkitektur var bra för detta problem innebär att förbättringar på denna punkt är högst troliga.

Därmed antas det att Nolfi´s ANN-arkitektur med spontan modularitet även i detta experiment kommer att ge den bästa prestandan för roboten.

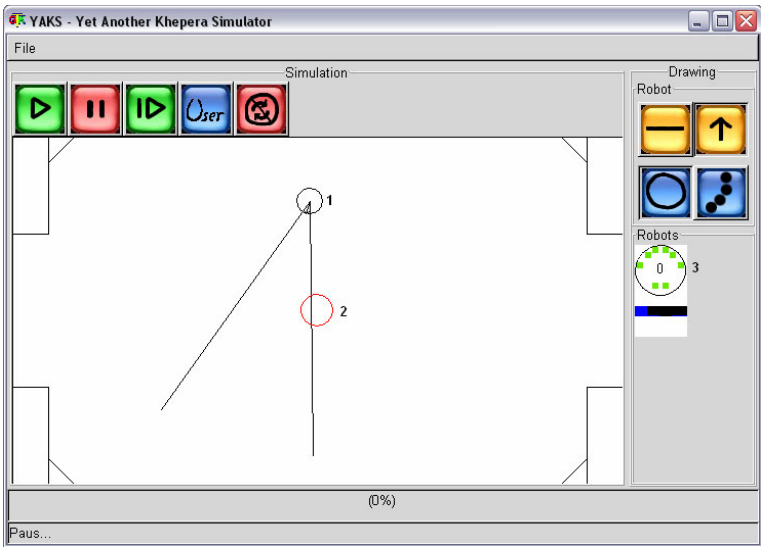
4. Metod

Ifrån problembeskrivningen (se kapitel 3) utvecklades en strategi för hur hypotesen skall verifieras. Detta kapitel ger en beskrivning av tillvägagångssättet för att genomföra dessa experiment.

Experimenten kommer enbart att genomföras i YAKS-simulatorn och hur detta genomförs kommer att diskuteras noggrant. I kapitel 4.1 kommer det att handla om hur omgivningen (i detta fall fotbollsplanen) simuleras i YAKS och vilka inställningar som genomförs gällande robotar och simulatormiljön. Därefter ges en beskrivning av de evolutionära detaljerna (kapitel 4.2-4.3) kring experimenten, speciellt vilka förändringar som sker på ANN-arkitekturerna och en detaljerad beskrivning av den fitnessfunktion som används. Till sist (kapitel 4.4) så ges även en beskrivning av hur experimenten genomförs och vilka GA parametrar som används till experimenten.

4.1. Miljön

Miljön som kommer att användas till roboten är baserad på reglerna för Kheperafotboll (se bilaga A). Denna miljö kommer att simuleras så likvärdigt som möjligt enligt reglerna i YAKS-simulatorn och slutresultatet kan ses i figur 10. Då miljön byggs upp relativt enkelt med ett scriptspråk som finns inbyggt i YAKS tas inga detaljer upp angående hur denna detalj gick tillväga.



Figur 10. Fotbollsplanen simulerad i YAKS-simulatorn.

Vad som kan urskiljas ifrån figur 10 är roboten och dess vyfält som fås genom kameran, vilket är markerad med en etta. Bollen är markerad med en tvåa och simulatorns visuella sensorindikator är markerad med en trea. En anmärkning som kan vara värd att påpeka är att ifrån gränssnittet kan inte urskiljas vilka färger olika primitiver i världen har, till exempel har väggarna som ska föreställa målen svart färg och övriga väggar har ljusgrå färg. Detta går inte att urskilja enbart genom att kolla på gränssnittet som fås genom YAKS.

Eftersom Smith (1997) hade valt att inte använda IR-sensorerna i sitt experiment så används inte dessa i detta experiment heller. Att inte använda dessa innebär att roboten enbart får navigera sig efter sin kamera, vilket resulterar i att roboten inte vet när den har kolliderat med något. Fördelen med att inte använda IR-sensorerna är att roboten och dess kontrollsystem (i detta fall den ANN-arkitektur som används) blir mindre komplex och kräver då mindre träningstid. Roboten kommer alltså endast utrustas med kameratornet som enda sensorkälla.

Att kameratornet inte exakt kan återge den vy som den ser framför sig på ett korrekt sätt vid användning på en riktig robot har Smith (1997) visat i sina experiment. Detta har då tagits i beaktning och därmed påverkas dessa sensorer med brus för att få de mer realistiska. Brusnivån som läggs till är ±5% på sensorvärdena från kameran som är ett värde baserat ifrån Smiths arbete. Om denna parameter är optimal eller inte kommer inte att inverka på detta arbete då valet har gjorts att inte verifiera resultaten på en riktig robot, skulle detta ha gjorts däremot så är valet av denna parameter desto viktigare.

4.2. ANN

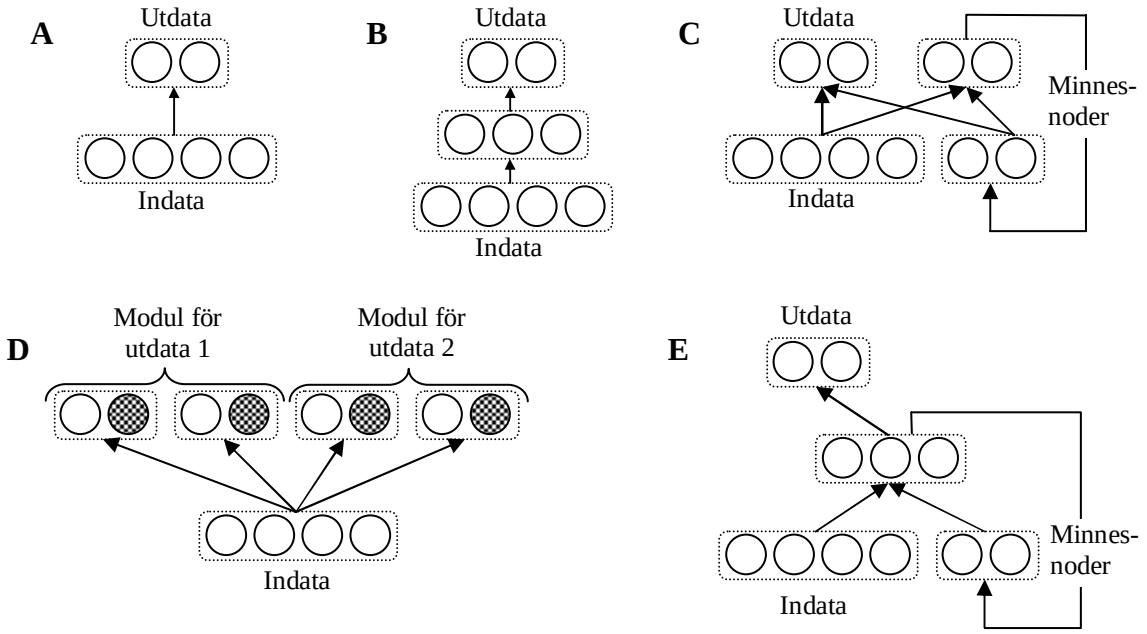
För att kunna använda Nolfis arkitekturer (Nolfi 1996) till detta experiment krävs det några förändringar.

Av Nolfis 5 olika arkitekturer är endast 4 applicerbara till detta problem. Det är Nolfis arkitektur D (se kapitel 2.4.2) som inte är applicerbar, detta då valet av vilken modul som skulle vara aktiv berodde på om roboten har ett objekt i griparmen eller inte. Något sådant enkelt samband finns inte i detta experiment, därför utvärderas inte denna arkitektur.

Nolfi använde i sina experiment sju stycken in-noder för sina arkitekturer, sex för de främre IR-sensorerna och en för griparmssensorn. I detta arbete kommer inte IR-sensorerna eller griparmssensorn att användas men däremot så kommer det att användas 16 stycken in-noder för kameratornet. Valet av 16 noder för kameratornet är anpassat efter Smiths arkitektur som han använde sig av och eftersom detta arbete evaluerar prestanda mellan olika arkitekturer gör att det är den relativa prestandan mellan dessa som är det intressanta. Om detta är det optimala antalet noder för kameratornet kan diskuteras men det är inget krav i detta arbete. Alltså kommer det att användas 16 stycken in-noder för indata till det artificiella neurala nätverket (se figur 10).

Vad gäller utdata använde Nolfi fyra ut-noder till sina arkitekturer, två för motorerna och två för griparmsfunktionaliteten. I detta arbete kommer inte griparmen användas och därför behövs det bara två utnoder för utdatan, en för varje motor. När det gäller antalet gömda noder i arkitektur B och E (se Figur 11) så används det 16 stycken noder, samma antal som Smith använde i sitt experiment och med samma motivering som vid valet av antalet in-noder.

Till Smiths arkitektur (se kap 2.4.1) har det endast gjorts två smärre förändringar. Någon kompass kommer inte att användas då robotarna inom Khepera-fotboll inte är utrustade med sådana. Därmed tas den in-noden bort. Den andra förändringen är ut-noden ”ball present”, varför denna skall användas har inte någon logisk förklaring hittats hittills. Därför har valet gjorts att plocka bort denna.



Figur 11. De modifierade ANN-arkitekturerna, A-D Nolfis, E Smiths.

Alltså kommer 5 stycken olika arkitekturer att testas:

A. En enkel framåtmatad arkitektur med 16 in-noder och 2 ut-noder, anpassad efter Nolfis arkitektur A.
B. En framåtmatad arkitektur med ett gömt lager. Arkitekturen består av 16 in-noder, 16 gömda noder och 2 utnoder. Anpassad efter Nolfis arkitektur B.
C. En återkopplande arkitektur med 16 in-noder, 2 minnes-noder och 2 ut-noder. Anpassad efter Nolfis arkitektur C.
D. En spontan modulär arkitektur med 16 in-noder och 2 olika moduler, en för varje motor. Anpassad efter Nolfis arkitektur E.
E. En återkopplande arkitektur med gömt lager. Arkitekturen består av 16 in-noder, 16 gömda noder och 2 ut-noder. Anpassad efter Smiths arkitektur.

4.3. Fitnessfunktion

I experimentet kommer det användas en liknande fitnessfunktion som Smith (1997) använde sig av. Denna används då Smith provade några olika funktioner och kom fram till att denna fungerade tillfredställande (se formel 1). Att använda en redan testad funktion ger den fördelen att fel som härrör till fitnessfunktionen förhoppningsvis undviks.

$$fitness = \frac{TidKollaPåBollen}{TotalTid} + f(NärmstaAvståndTillBoll) + \\ + \sum s(TräffPåBoll) + g(AvståndBollTillMål)$$

Formel 1. Smiths fitnessfunktion.

Denna funktion värderar robotens prestanda genom 4 punkter:

TidKollaPåBollen/TotalTid: Ett värde som baserar sig på hur lång tid roboten har visuell kontakt med bollen under sin livstid. Detta värde ligger emellan 0 - 1 poäng.

f(NärmstaAvståndTillBoll): En funktion som returnerar ett linjärt avstånd mellan bollen och roboten med övre/nedre gränsvärden. Funktionen returnerar ett värde mellan 1-15 poäng och 1 returneras om robotens avstånd till bollen aldrig är kortare än 15 cm, annars returneras (15 – närmsta avstånd till bollen i cm under hela körningen).

Σs(TräffPåBoll): En funktion som returnerar ett värde baserat på antalet kollisioner mellan boll och robot. Funktionen returnerar ett värde mellan 0-300 poäng. För varje kollision med bollen tilldelas 15 poäng till robotens fitness med en övre begränsning på 20 kollisioner.

g(AvståndBollTillMål): En funktion som returnerar ett värde baserat på avståndet mellan bollen och målet. Värdet som fås baseras på formeln (((avstånd mellan mål och mittpunkt – avstånd mellan boll och mål)/avstånd mellan mål och mittpunkt)*3000)+315 hamnar då mellan 0 som är den undre gränsen och 3315 poäng. Detta innebär att även om roboten bara lyckas putta bollen någon ynka cm mot målet så rankas det högre än de andra delfunktionerna.

Om roboten lyckas på något sätt att få bollen i rätt mål ges roboten automatiskt en fitness på 10 000 och körningen avbryts. Om bollen hamnar i fel mål händer ingenting, detta eftersom den automatiskt får en relativt låg fitness när bollen är på fel planhalva.

Fitnessvärdet kommer att variera emellan 0-10 000 poäng beroende på hur pass bra roboten presterar. Att de olika del-funktionerna inte bidrar lika mycket till fitnessen har att göra med hur viktig funktionen anses. Att se bollen ska inte ge lika hög fitness som att flytta bollen mot målet, detta eftersom att få bollen att röra sig mot målet har större sannolikhet att resultera i ett mål än att endast se bollen.

Om roboten kolliderar med en vägg så avbryts körningen, dock får roboten behålla det aktuella fitnessvärdet den lyckats få ihop.

4.4. Tillvägagångssätt

I grund och botten krävs det två olika typer av simuleringar för att kunna evaluera skillnader mellan olika typer av ANN-arkitekturer. En typ av simulering som utvecklar roboten och en typ av simulering som testar prestandan på den bäst utvecklade roboten. I detta arbete har de olika typerna kallats för evolutionär simulering samt resulterande simulering och de kommer att beskrivas utförligt här nedan.

1. **Evolutionär simulering.** Simulering av den evolutionära processen för utvecklingen av den fotbollsspelande roboten.

2. **Resulterande simulering.** Simulering för slutgiltiga tester av den bästa individens förmåga att klara av uppgiften, i detta fall att spela fotboll.

4.4.1. Evolutionär simulering

Grundsyftet med den evolutionära simuleringen är alltså att träna/utveckla den fobollsspelande roboten. Till den evolutionära processen finns det ett flertal parametrar som kan påverka det slutgiltiga resultatet, att optimera dessa till detta arbete är en allt för tidskrävande process. Istället kommer parametervärdena att väljas efter tidigare liknande arbeten. Att inte tid har lagts ner till att optimera dessa är att huvudsyftet med detta arbete är att evaluera olika ANN-arkitekturer och de kommer att evalueras relativt mot varandra med samma förutsättningar och därmed är det inte helt nödvändigt att dessa parametrar är optimerade för detta problem.

Till den evolutionära processen finns det 5 parametrar som påverkar resultaten (*population*, *generation*, *epok*, *elit* och *mutation*, se kapitel 2.1.2 för detaljer) och dessa kommer att gås igenom en efter en.

Population Denna parameter anger antalet individer vid varje generation, om denna parameter är för liten innebär det en mindre spridning på lösningarna och därmed är chansen mindre att en bra lösning hittas. Är dock denna parameter för stor innebär det främst att simuleringstiden blir onödigt lång. Mitchell (1996) anser att en population mellan 50-100 individer är att rekommendera till de flesta problem och då Smith (1997) använde till sitt experiment en population på 100 individer används denna storlek även till detta experiment. Alltså används en population på 100 individer.

Generation Det antal generationer som utvecklingen av roboten skall fortgå är en parameter som påverkar hur länge sökningen skall pågå. Om denna parameter är för liten innebär det att det troligen finns bättre lösningar som inte har fått tid att evolveras fram ännu och om den är för stor innebär det precis som för populationsstorleken att simuleringstiden blir onödigt lång. Smith (1997) använder i sitt arbete 3000 generationer för den evolutionära processen men resultaten ifrån hans fitnessutveckling visar tecken på att roboten kan utvecklas ytterliggare och därmed ökas denna parameter rejält. I detta arbete utvecklas individerna under 10 000 generationer.

Epok En epok är en beteckning på hur lång tid som individen får agera i miljön och samla på sig fittnesspoäng. Får individen bara agera en kort tid innebär det att individen kommer att få svårt att hinna med att prestera nåt vettigt och används en lång tid innebär det även i detta fall att simuleringstiden blir onödigt lång. Smith (1997) låter individen få agera i 100 sekunder under sina försök men detta är en relativt lång tid för detta problem och denna tid kan utnyttjas till annat. Då den uppgift som individen ställs mot kan betraktas som liknande det straffslag som beskrivs i reglerna för Kheperafotboll (se bilaga A) och där får roboten 30 sekunder på sig att försöka göra mål. En epoktid på cirka 30 sekunder kommer alltså att användas.

Elit Denna parameter bestämmer hur många av de bästa individerna vid varje generation som direkt får gå vidare till nästa generation. De övriga som får vara med och skapa nästa generation väljs ut genom det turneringsbaserade urvalet som beskrivs i kapitel 2.1.2. Då Smith (1997) använde sig av en annan typ av selektionsalgoritm så kan inte hans

arbete ligga till grund för denna parameter, istället baserar vi i detta arbete denna parameter ifrån en mer generell rekommendation av Mitchell (1996). Mitchell rekommenderar att det används cirka 10 individer till den så kallade eliten vid en populationsstorlek på 100 individer. Om denna parameter sätts för lågt innebär det att förloras lösningar som har visat potential i en population och om det används en för hög parameter innebär det att processen kan lättare fastna vid lokala optima då urvalet till nästa generation lätt begränsas till de bästa individerna för stunden.

Mutation Hur stora förändringar som sker på ANN-arkitekturernas vikter mellan generationerna justeras av denna parameter. Muteringen av vikterna har som uppgift att ge förändringar på individernas beteenden mellan generationerna och därmed få fram nya lösningar som kommer att utvärderas med hjälp av fitnessfunktionen. Då Smith (1997) inte använde sig av gaussisk mutation och inte heller har Mitchell (1996) någon rekommendation gällande denna parameter har ett godtyckligt val gjorts. I detta arbete används en gaussisk fördelning med standardavvikelse 1**.**

När de flesta parametrar har valts till denna typ av simulering finns det två detaljer kvar som kan påverka det slutgiltiga resultatet i stor grad. Den första detaljen är hur individernas startposition skall hanteras. Innan simuleringarna hade påbörjats hade 3 olika sätt att hantera detta identifierats. Det första sättet baserar sig på hur straffar hanteras i reglerna för Kheperafotboll (se bilaga A) och då placeras roboten 10 cm rakt bakom bollen riktad mot målet, vilket innebär att den har raka vägen fram till målet. Det andra sättet är att slumpvis ta fram en startposition någonstans på den egna planhalvan. Detta innebär att roboten får svårare att ”bara” härma ett visst rörelsemönster men nackdelen är fortfarande att roboten får så gott som hela tiden bollen mellan sig självt och målet. Därför har det valts, precis samma sätt som Smith (1997) använde, att till detta arbete använda en startposition som är slumpvis framtagen någonstans på planen förutom då i något mål, på bollen eller på någon vägg. Roboten är även vid starten riktad mot det mål som den skall försöka att forcera bollen emot.

Den andra detaljen är hur många försök som varje individ få på sig. Används bara ett försök innebär det att slumpen spelar en stor roll, det vill säga att individerna kommer att bli olika favoriserade beroende på var startpositionen hamnar. Smith (1997) använde i sitt arbete endast ett försök vilket då anses vara för lågt då slumpen kommer att inverka i allt för stor grad, därmed ökas denna parameter till två försök. Varför den inte ökas ytterligare beror på den tidskomplexitet som denna parameter för med sig, genom att öka från ett försök till två försök så dubbleras simuleringstiden.

Varje simulering kommer alltså att gå till på följande sätt:

1. Vid den första generationen individer tilldelas vikterna slumpmässigt i ANN-arkitekturen, vilket innebär att individerna kommer att få olika beteenden.
2. Varje individ placeras slumpmässigt ut varsomhelst på planen fast inte på någon vägg, i något mål eller på bollen. Individen börjar alltid med att riktas emot det vänstra målet, detta för att den skall veta var den skall göra mål.

3. Varje individ får sedan 2*30 sekunder på sig att försöka få bollen i det vänstra målet, hur pass bra den lyckas med det utvärderas med hjälp av fitnessfunktionen.
4. Efter att alla individer har fått agera kommer det att skapas en ny generation av individer enligt turneringsbaserat urval med elitism och detta innebär att de 10 bästa individerna förflyttas direkt till nästa generation medan de övriga 90 får tävla om det.
5. I den nya generationen kommer det att ske en mutation på vikterna i ANN-strukturen innan de får påbörja sina försök och i detta arbete har det valts att använda en gaussisk fördelning med standardavvikelse 1.
6. Punkterna 2-5 kommer sedan att itereras under 10 000 generationer.

Allt som allt kommer det att genomföras 50 simuleringar, 10 simuleringar per arkitektur.

4.4.2. Resulterande simulering

Den resulterande simuleringen har som uppgift att testa prestandan på den bästa evolverade roboten i den 10 000: e generationen vid varje simulering. Till denna simulering har 3 aspekter som är intressanta för detta arbete identifierats, den första aspekten är hur bra individen egentligen är på att utföra den tänkta uppgiften. Alltså att utföra ett ordentligt test av individens förmåga att ifrån olika positioner göra mål. Den andra aspekten är hur roboten löser uppgiften, alltså att med visuell övervakning undersöka hur roboten angriper problemet. Den sista aspekten som är av intresse har dock inte någon direkt koppling till prestandan av roboten, det som undersöks i denna aspekt är relaterat till det spontana modulära nätverket och hur fördelningen sker mellan modulerna. Det kommer nu att ges en mer utförlig beskrivning av de 3 aspekterna som i detta arbete kallas prestandatest, beteendetest och modultest.

Prestandatest
Prestandatestet kommer alltså att göra ett utförligt test på hur bra den bästa individen ifrån den 10 000: e generationen verkligen har utvecklats. Testet är baserat ifrån Smith (1997) där han gjorde ett liknande test för att utvärdera robotens prestanda och som riktvisning kan då ges att Smiths bästa robot gjorde mål cirka 25% av försöken.

Testet går till så att roboten med högst fitnesspoäng ifrån den 10 000: e generationen vid varje evolutionär simulering får 10 000 försök att göra mål på fotbollsplanen. Startpositionerna hanteras likadant som under den evolutionära simuleringen, alltså att den slumpas ut varsomhelst på planen förutom i något mål, på bollen eller på någon vägg. Det enda som räknas är om det blir mål eller inte och det innebär att resultatet är antalet försök som resulterar i mål, alltså ett värde mellan 0 till 10 000. Allt som allt kommer det även här att genomföras 50 simuleringar, 10 simuleringar per arkitektur.

Beteendetest
Beteendetestet görs för att undersöka hur roboten löser sin uppgift och är baserad ifrån ett liknande test som Smith (1997) utförde. Genom en visuell övervakning av robotens försök kommer beteenden ifrån roboten att kunna urskiljas. Detta kommer att undersökas under prestandatestet så det är samma förutsättningar som gäller angående startpositioner och dylikt, dock kommer inte alla försök ifrån prestandatestet att undersökas då det skulle ta några månader att genomföra om detta gjordes.

Syftet med detta test är att se hur roboten löser problemet med att starta ifrån olika startpositioner och speciellt då den startar någonstans på motståndarens planhalva. Svårigheten med att starta ifrån motståndarens planhalva är att roboten hamnar mellan bollen och det mål som den skall försöka få bollen i. Detta innebär att roboten på något sätt måste ta sig runt bollen för att ha en chans att göra mål och det beteende som den uppvisar för detta fall är det som är av intresse.

Modultest

I Nolfi (1996) gjordes flera intressanta test av det spontana modulära nätverket och tanken med att även göra ett modultest i detta arbete är att se hur väl det stämmer överens med de resultat Nolfi kom fram till. Modultestet görs alltså inte för att evaluera de olika arkitekturerna utan för att undersöka hur Nolfis arkitektur med spontan modularitet utnyttjar sina moduler. Använder arkitekturen till exempel endast två olika moduler så är det ingen skillnad på denna arkitektur och Nolfis framåtmatade arkitektur. Detta test kommer inte att vara lika utförligt som Nolfis men kommer ändå att ge en riktvisning hur modulerna används.

Testet är väldigt enkelt, under prestandatestet så skapas det en logg över vilken modul som är aktiv i varje tidssteg och därmed kan modulutnyttjandet enkelt fås fram genom en sammanställning av denna logg.

5. Resultat

Detta kapitel innehåller resultaten som har genererats ifrån simuleringarna beskrivna i kapitel 4 och dessa kommer att representeras i några olika diagram.

5.1. Evolutionär simulering

Diagram 1 visar den bästa individens utveckling under simuleringarna för de olika arkitekturerna. Detta ger en indikation på om det är någon ANN-arkitektur som lär sig fortare än de andra eller om det är någon ANN-arkitektur som lyckas att få robotarna att prestera bättre än de andra. En riktlinje som kan ges är att en fitness på 20 000 indikerar att bägge försöken har resulterat i mål.

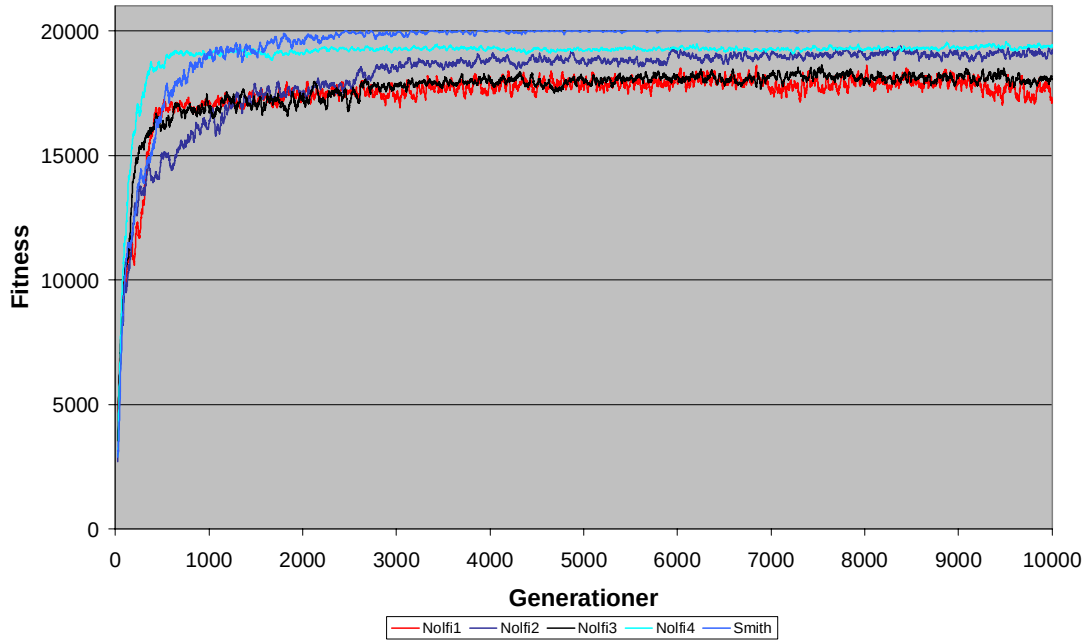


Diagram 1. De olika arkitekturernas fitnessutveckling.

Något som visade sig vara intressant för valideringen av Diagram 1 ovan är hur spridningen såg ut mellan de olika simuleringarna för sig. Varför detta är av intresse tas upp under analysen men en presentation av de olika kommer att visas nedan.

Allt som allt genomfördes det alltså 10 simuleringar per arkitektur och detta medför då att det är 10 kurvor per arkitektur som visas. Här nedan presenteras då 5 diagram ifrån de enskilda simuleringarna för arkitekturerna och de är då:

Diagram 2 Visar hur spridningen var mellan de enskilda simuleringarna för Nolfis arkitektur 1, arkitekturen med det enkla framåtmatade nätverket.

Diagram 3 Visar spridningen mellan simuleringarna för Nolfis arkitektur 2, arkitekturen med det gömda lagret.

Diagram 4 Visar spridningen mellan simuleringarna för Nolfis arkitektur 3, arkitekturen med återkoppling.

Diagram 5 Visar spridningen mellan simuleringarna för Nolfis arkitektur 4, arkitekturen med spontan modularitet.

Diagram 6 Visar spridningen mellan simuleringarna för Smiths arkitektur.

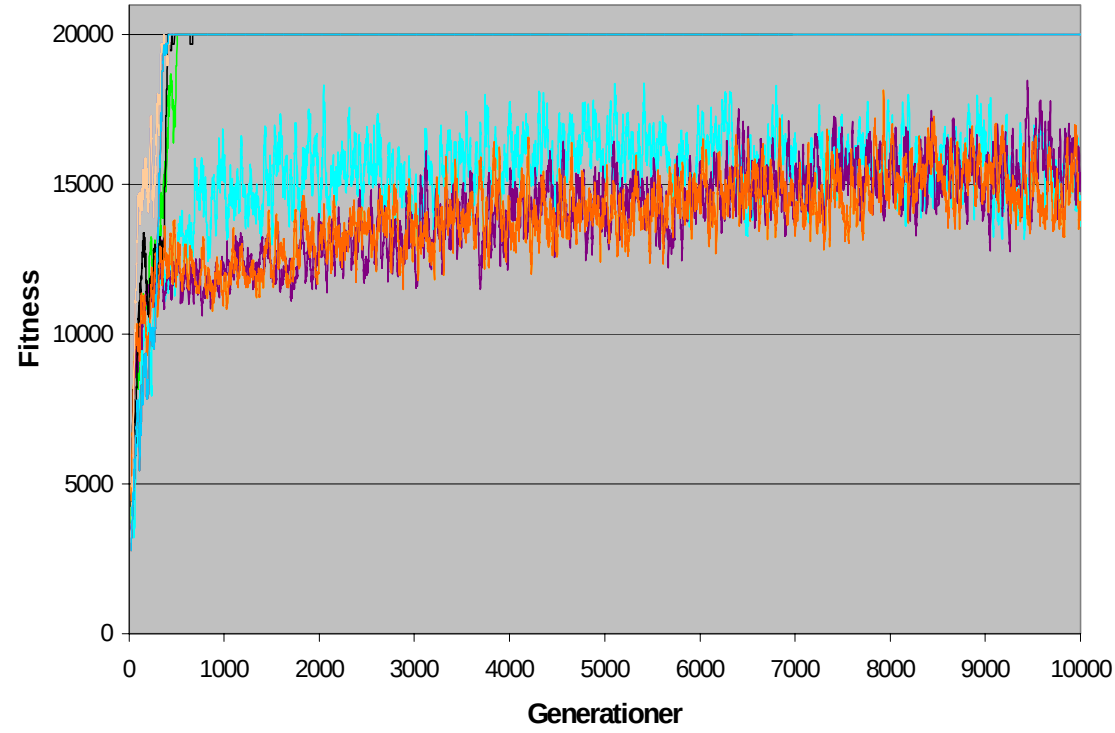


Diagram 2. Fitnessutvecklingen för de enskilda simuleringarna för Nolfi1.

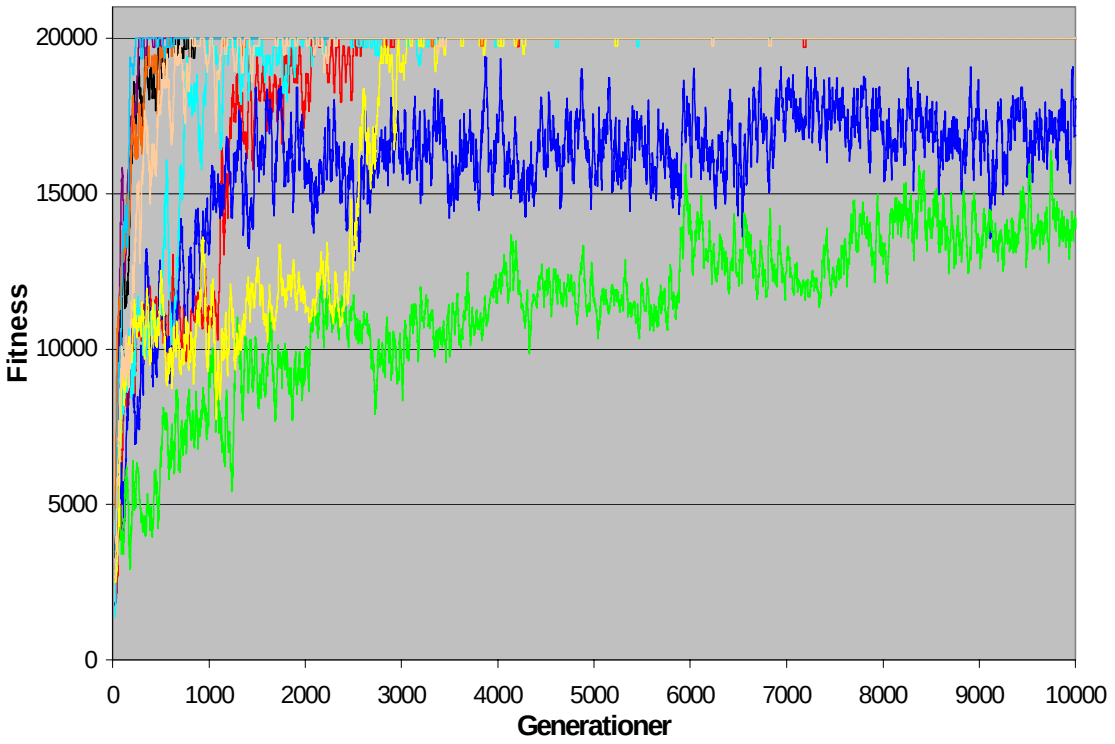


Diagram 3. Fitnessutvecklingen för de enskilda simuleringarna för Nolfi2.

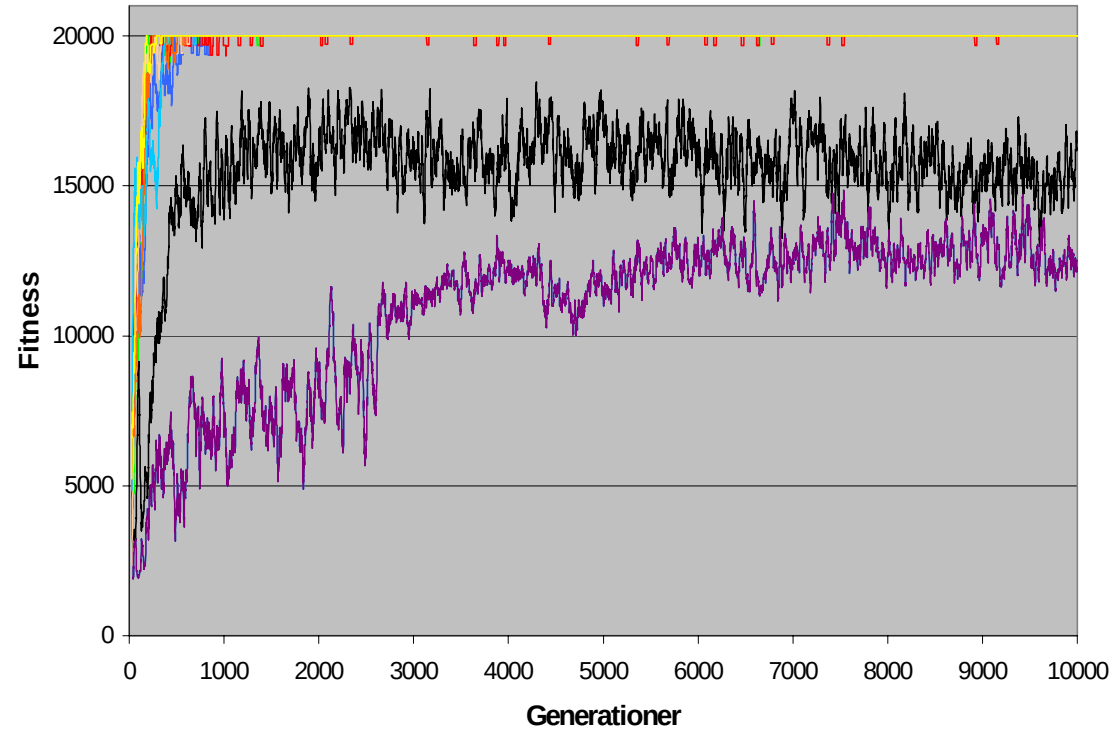


Diagram 4. Fitnessutvecklingen för de enskilda simuleringarna för Nolfi3.

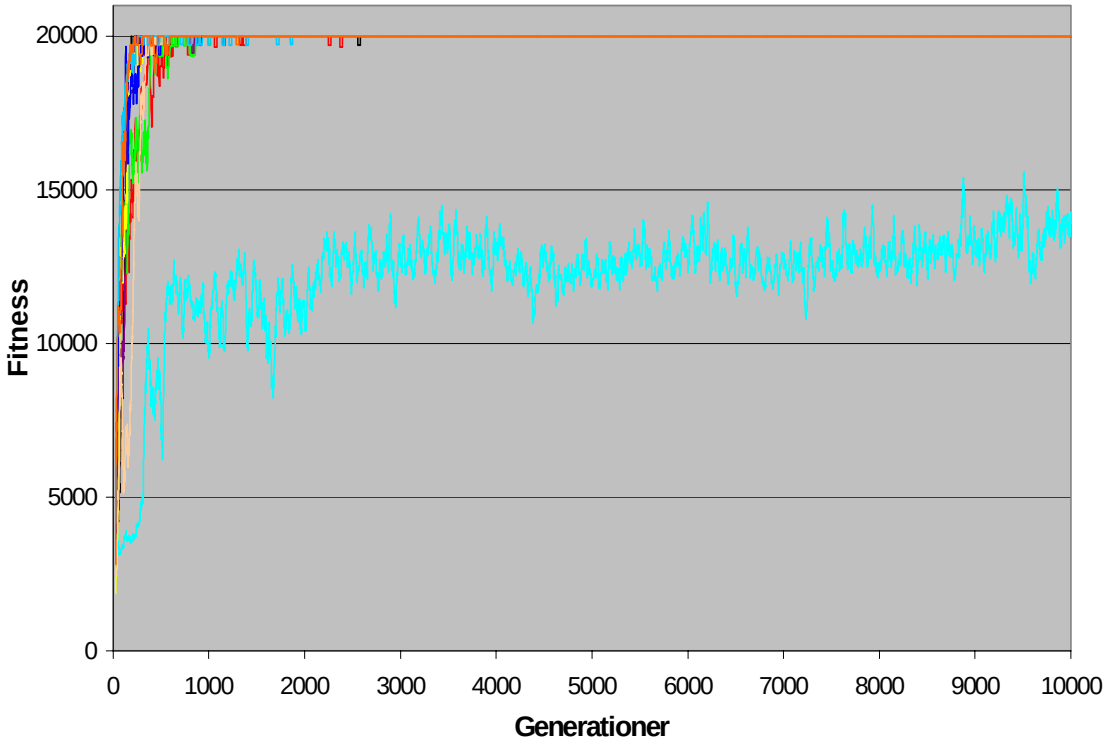


Diagram 5. Fitnessutvecklingen för de enskilda simuleringarna för Nolfi4.

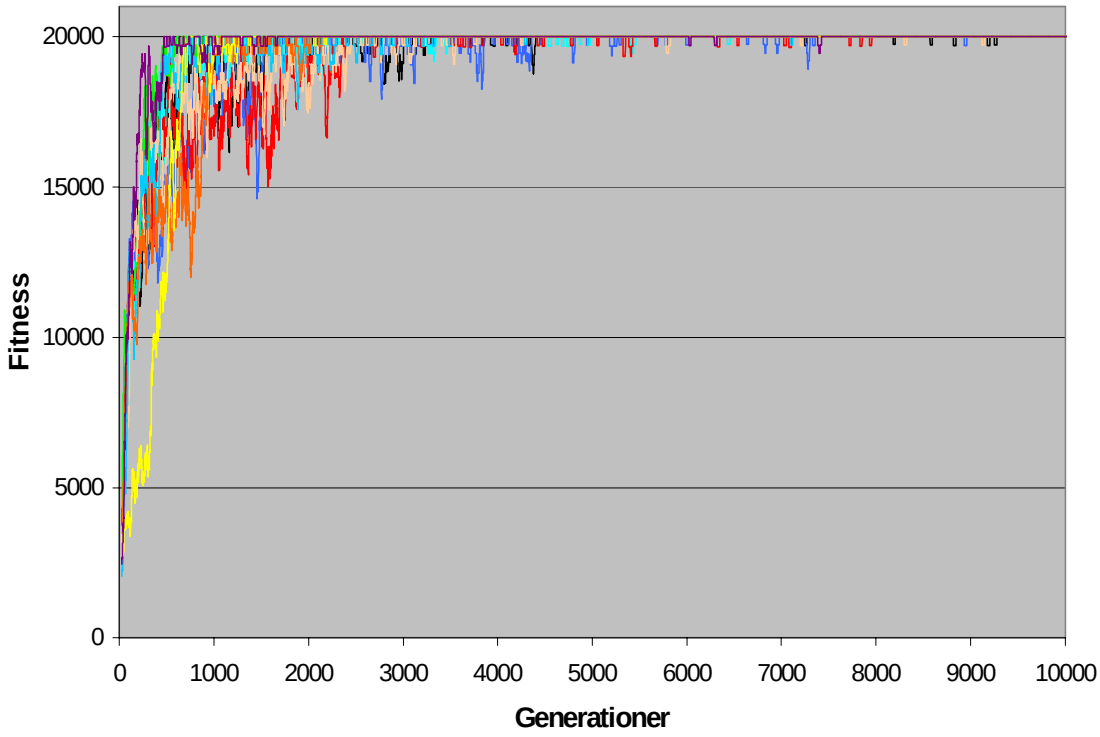


Diagram 6. Fitnessutvecklingen för Smiths arkitektur.

5.2. Resulterande simulering

Prestandatest

Diagram 7 visar hur pass väl de olika ANN-arkitekturerna utför sin uppgift i att försöka få bollen i målet. Detta diagram baserar sig på resultaten ifrån prestandatestet (se kapitel 4.4) där den bästa individen ifrån varje simulering får 10 000 försök på sig att göra mål.

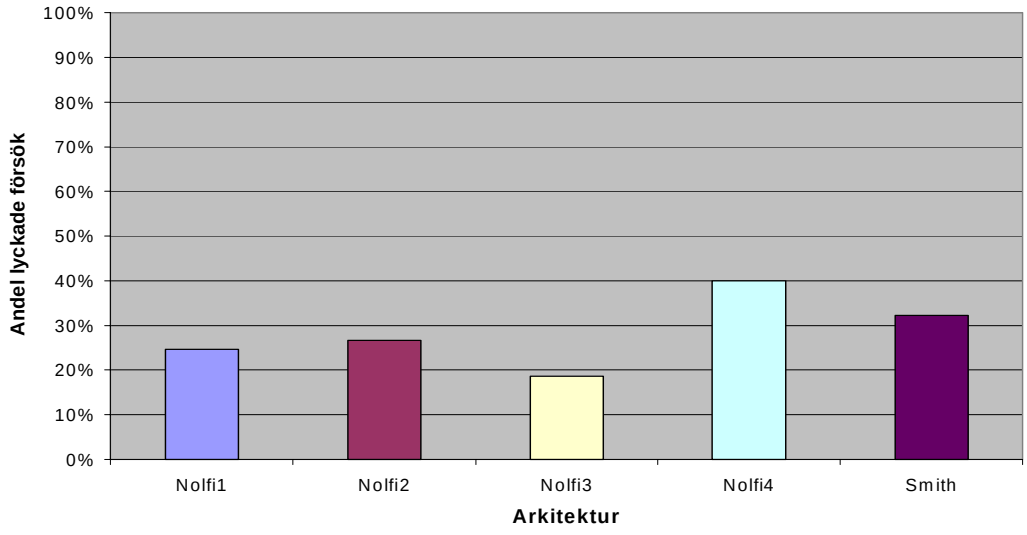
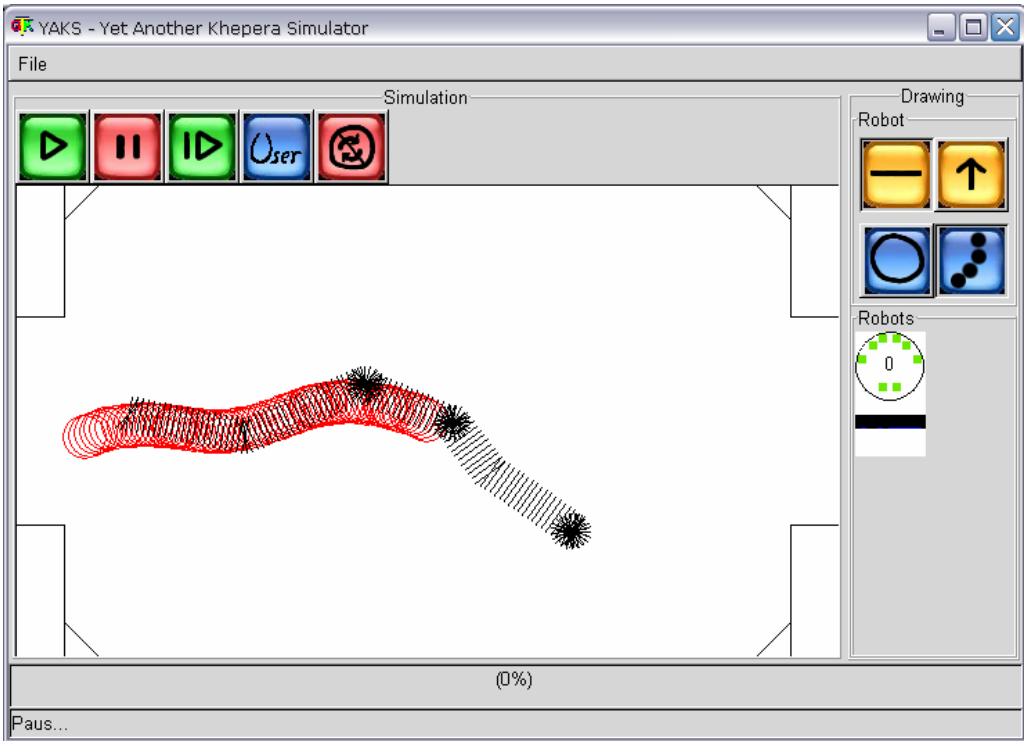


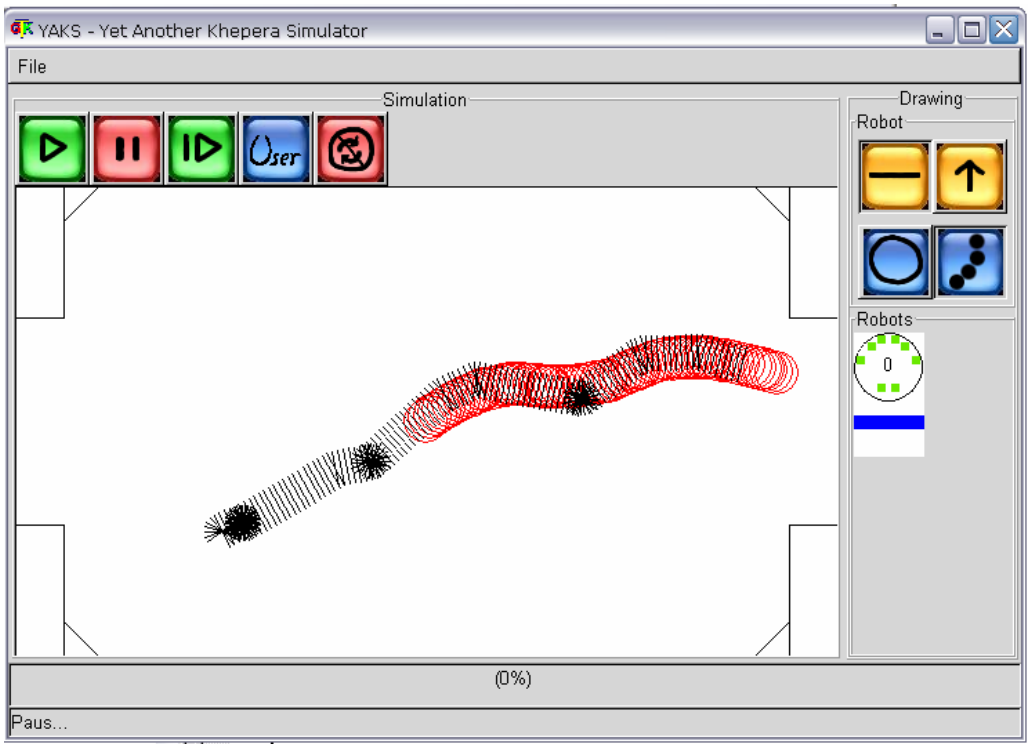
Diagram 7. Andel försök som resulterade i mål.

Beteendetest

Detta test ska undersöka ifall roboten uppvisar ett sådant beteende så den vet åt vilket håll den skall försöka forcera bollen. Genom att roboten får olika startpositioner undersöks det vilket rörelsemönster roboten utför. Figur 12 visar rörelsemönstret för roboten när den startade ifrån bollens högra sida och Figur 13 visar rörelsemönstret för roboten när den startade ifrån bollens vänstra sida. Det kan påpekas dock att de röda fälten är bollens rörelsemönster och de svarta fälten är robotens rörelsemönster.



Figur 12. Robotens rörelsemönster vid startposition på den högra planhalvan.



Figur 13. Robotens rörelsemönster vid startposition på den vänstra planhalvan.

Modultest

Diagram 3 visar hur Nolfis arkitektur med spontan modularitet (se kapitel 4.2) fördelar utnyttjandet av sina olika moduler. Testet gjordes genom att kontrollera hur stor andel varje modul hade kontrollen under 4 olika simuleringar på 10 000 epoker vardera. Det som kan påpekas är att modul 1 och modul 2 konkurrerar om den vänstra motorn och modul 3 och modul 4 konkurrerar om den högra.

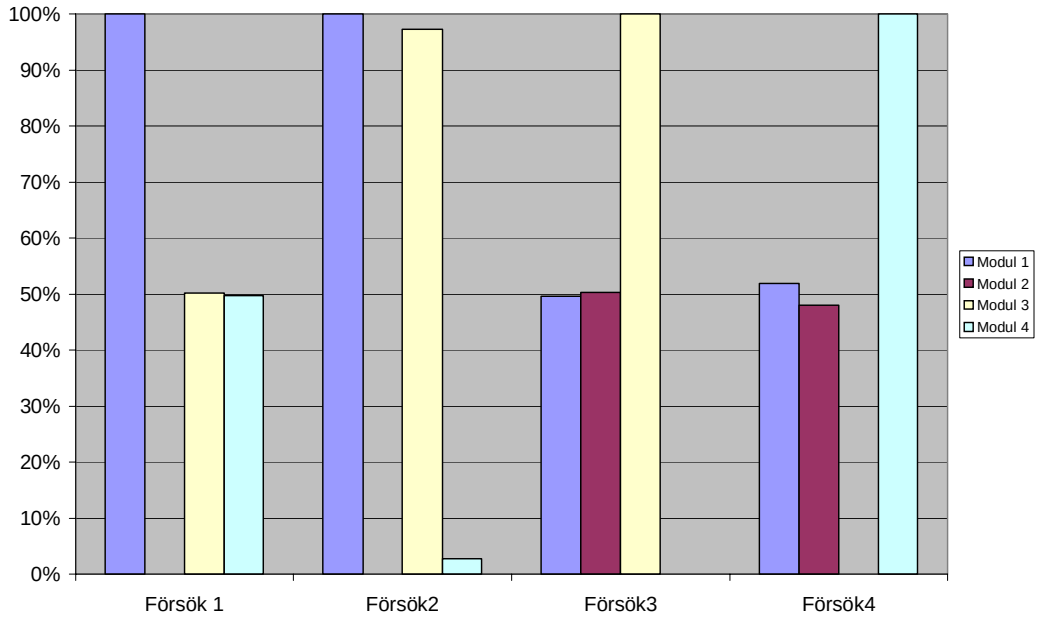


Diagram 8. Modulutnyttjande av Nolfis arkitektur med spontan modularitet.

6. Analys

I detta kapitel kommer arkitekturerna att evalueras med hjälp av de resultat som bildades ifrån de olika simuleringstyperna som presenterades i kapitel 4.4.

6.1. Evolutionär simulering

Diagram 1 (se kapitel 5.1) visar att Smiths arkitektur uppvisar en förvånansvärd snabb utveckling och hög stabilitet hos den bästa individen, bäst av alla arkitekturer faktiskt. Detta resultat hade inte förväntats då Smiths arkitektur är en blandning mellan Nolfis arkitektur med gömda noder (Nolfi2) och Nolfis arkitektur med återkoppling (Nolfi3), dessa två arkitekturer uppvisar inte alls samma tecken i detta arbete eller i Nolfis artikel (Nolfi 1996).

En annan intressant företeelse som kan urskiljas ifrån Diagram 1 är att det gömda lagret kräver en längre träningstid, både Smiths arkitektur och Nolfis arkitektur med gömt lager (Nolfi2) uppvisar samma typ av lite segare utveckling. Detta kan ses genom att de kräver fler generationer på sig innan fitnessutvecklingen börjar plana ut.

Hur mycket ska dessa resultat bidra till slutsatserna då? Vid undersökning av de enskilda simuleringarnas utveckling, som finns som Diagram 2 – Diagram 6 i kapitel 5.1, visar till exempel att Nolfis arkitektur med spontan modularitet (Nolfi4) presterar mycket bra över lag förutom vid ett tillfälle då en lösning inte har hittats som klarar av att göra mål på bägge försöken. Eftersom det är så pass stor skillnad i fitnessvärde mellan att göra mål på ett försök och göra mål på två försök resulterar detta i att det enskilda resultatet påverkar det sammanlagda resultatet i stor utsträckning. Om det bortses ifrån den enda simulering som inte lyckades så uppvisar Nolfis arkitektur med spontan modularitet en både snabbare och stabilare utveckling på den bästa individen än vad Smiths arkitektur visar fast detta går inte att se ifrån det sammanlagda resultatet. Att ett enskilt resultat påverkar så stort beror på att det bara utförs 10 körningar, hade det utförts fler simuleringar per arkitektur hade också ett statistiskt bättre resultat erhållits.

Frågan är dock hur pass generell är lösningen efter 10 000 generationers utveckling, denna fråga skall besvaras under kapitel 6.2.

6.2. Resulterande Simulering

Prestandatest

Diagram 7 (se kapitel 5.2) visar att Nolfis arkitektur med spontan modularitet (Nolfi4) lyckas bättre än de övriga med cirka 40% av försöken som resulterar i mål. Smith (1997) utförde även han detta test och då lyckades hans arkitektur göra mål cirka 25% av sina försök och detta stämmer ganska väl med det resultat som har genererats i detta arbete (strax över 30%). Den skillnad som har uppstått kan bero på en mängd saker, till exempel används inte samma simulator, arkitekturen har tränats under en längre tid och det genomförs bara 10 körningar.

Att roboten lyckas med att göra mål cirka 40% av försöken kan ur mänsklig synpunkt anses vara väldigt lågt. Detta då att ensam på planen utan vare sig försvarare och målvakt lyckas med att göra mål anses inte vara en svår uppgift. Varför inte roboten

presterar bättre skall undersökas genom att kontrollera hur den beter sig under sina försök, detta undersöks i beteendetestet nedan.

Beteendetest
Det som ville visas med detta test är hur roboten beter sig ifrån olika startpositioner på fotbollsplanen, speciellt då när den startar ifrån den vänstra planhalvan. I figur 12 (se kapitel 5.3) visas rörelsemönstret för roboten när den får starta på den högra planhalvan. Att startpositionen är på den högra planhalvan innebär (om roboten inte får starta relativt närma mittlinjen) att bollen är mellan roboten och det vänstra målet vilket resulterar i att roboten i stor grad endast behöver lokalisera bollen och forcera den framåt för att den skall lyckas.

För att roboten skall lyckas att göra mål i rätt bur när den startar ifrån den vänstra planhalvan krävs det för roboten att på något sätt ta sig runt bollen för att kunna forcera denna mot det vänstra målet. I figur 13 (se kapitel 5.3) visas rörelsemönstret för roboten när den startar ifrån den vänstra planhalvan. Ifrån denna bild kan urskiljas att roboten inte visar några tecken på att försöka ta sig runt bollen utan roboten lokaliserar bollen med kameratornet och forcerar bollen framåt. Detta beteende har observerats på alla testkörningar som utfördes för att testa beteenden.

Att roboten beter sig på detta sätt ger därmed svar på varför arkitekturerna presterar så lågt som de gör. Eftersom roboten får en slumpvis framtagen startposition så innebär det att cirka 50% av försöken får den starta på den vänstra sidan av planen och om den inte klarar av att komma runt bollen så innebär det att så gott som alla dessa försök kommer att misslyckas.

Varför klarar roboten inte av att komma runt bollen? Den största orsaken till detta är att fitnessfunktionen enbart ger högre fitnessvärde om bollen flyttas mot det vänstra målet, det som behövs också är att roboten bestraffas när bollen flyttas åt det högra målet. Att detta inte finns innebär att robotarna som får starta på den vänstra planhalvan betraktas lika ”dåliga” och därmed går det inte att urskilja de bättre individerna mot de sämre.

En annan orsak som bidrar till detta problem är att robotarna får en slumpmässig vald startposition under sina försök. När det som i detta arbete används en population på 100 individer per generation så är det högst troligt att åtminstone någon av dessa 100 individer får bägge sina försök på den egna planhalvan. Även om antalet epoker ökas kommer detta problem att bestå i stor utsträckning (om inte antalet ökas drastiskt) och att öka antalet epoker gör även att simuleringstiden blir väldigt lidande. Ett sätt att lösa detta problem är att startpositionerna riktas för individerna. Med detta menas att varje individ kommer att få till exempel en startposition på den vänstra planhalvan och en startposition på den högra planhalvan. Detta innebär att alla individer kommer att få väldigt snarlika förutsättningar och en mer generell lösning kommer att krävas för att lyckas på bägge försöken.

Modultest
Det här testet gjordes inte för att evaluera de olika arkitekturerna utan mest för att undersöka hur Nolfis arkitektur med spontan modularitet utnyttjar de olika modulerna. Diagram 8 (se kapitel 5.4) visar resultaten ifrån detta test och visar hur stor andel varje modul hade kontrollen. Tanken är som sagt att endast undersöka om arkitekturen med spontan modularitet utnyttjar fler moduler än 2 och därmed visar att det är skillnad mellan denna arkitektur och ett vanligt framåtmatat nätverk.

Resultaten visar att 3 av 4 simuleringar använder fler än 2 moduler för att kontrollera roboten och detta kan då betraktas som att det utvecklas en skillnad mellan dessa två arkitekturer. En detalj som är intressant är att simulering 2 i Diagram 8 som nära inpå endast använde sig av två moduler är även den simulering som inte lyckades med att evolvera fram en lösning som klarade av att göra mål på bägge försöken (se Diagram 5 och den kurvan med låga fitnessutvecklingen). Detta kan tydas som att det inte är tillräckligt med endast 2 moduler för att styra roboten då resultatet blev så pass dåligt, dock krävs det fler tester för att verifiera detta.

7. Slutsats

I detta kapitel kommer slutsatser att dras ifrån analyserna av resultaten som gjordes i kapitel 6. Hypotesen som ställdes i kapitel 3.2 kommer även att verifieras utifrån de analyser som gjordes.

Det första som analyserades var fitnessutveckling av den bästa individen för de olika arkitekturerna, att dra några konkreta slutsatser ifrån detta resultat kan inte göras då det endast har utförts 10 simuleringar per arkitektur. Att det var för lite med 10 simuleringar visades när en undersökning gjordes av de enskilda simuleringarna. Det intressanta ifrån dessa är att Nolfis arkitektur med spontan modularitet presterar mycket bra överlag förutom under en simulering och denna simulering påverkade i så pass stor grad det sammanlagda resultatet att några ordentliga slutsatser inte kan dras. För att kunna göra detta skulle det krävas att fler simuleringar genomförs.

Det intressanta är dock när de bästa individerna ifrån den 10 000: e generationen får tillfället att testa sina fotbollsspelartalanger. Resultaten ifrån dessa simuleringar jämfördes mellan de olika arkitekturerna där Nolfis arkitektur med spontan modularitet uppvisade de bästa resultaten med mål vid cirka 40% av försöken. Jämförs detta resultat med det resultat som Smiths arkitektur, som har legat till grund för detta arbete, så lyckades denna göra mål strax över 30 % av sina försök. Alltså har en förbättring skett med nästan 10 procentenheter genom att använda en annan typ av arkitektur.

Den fråga som dock kvarstår, varför bara 40% som resulterar i mål? Vid undersökningen av robotarnas beteende visade det sig att de endast utförde ett relativt primitivt beteende. Att den endast uppvisar ett primitivt beteende har visat sig då roboten i så gott som alla fall bara kan göra mål ifrån den egna planhalvan. Att den får starta ifrån den egna planhalvan innebär att roboten i de flesta fall har bollen mellan sig själv och det mål som bollen slutligen helst skall hamna i. Får dock roboten starta ifrån en annan position har inga tecken visats på att den kan på något sätt förstå att den måste lösa problemet på ett annat sätt än att bara köra bollen framför sig.

Att roboten inte uppvisar ett mer utvecklat beteende har två troliga orsaker hittats. Den första orsaken är att fitnessfunktionen är felaktig, den har en förmåga att favorisera individer som får starta på den egna planhalvan. Det som behövs läggas till är någon typ av bestraffning för de individer som ”dribblar” bollen mot det egna målet.

Den andra orsaken är att det krävs riktade startpositioner och med riktade startpositioner menas att varje individ skall få en startposition ifrån olika zoner på spelplanen, till exempel ett försök på den egna planhalvan och ett försök på motståndarens planhalva. Fördelen med att använda riktade startpositioner mot slumpmässigt valda som användes i detta arbete är att alla individer får mer likvärdiga förutsättningar då alla får liknande startpositioner.

7.1. Hypotesprövning

Det här projektets hypotes antog att Nolfis ANN-arkitektur med spontan modularitet skulle förbättra prestandan för roboten, i detta fall göra mål på en fotbollsplan. Ifrån kapitel 5 så har arkitekturerna evaluerats i två olika anseenden, det ena undersökte de

olika arkitekturerna under träning och det andra undersökte arkitekturernas förmåga, efter träning, att utföra sin uppgift. Under träningsdelen så uppvisade inte Nolfis arkitektur med spontan modularitet någon häpnadsväckande prestanda men det som resultatet baserar sig mest på är arkitekturens förmåga att utföra sin uppgift och i detta anseende så presterar individerna som använder Nolfis arkitektur med spontan modularitet klart bättre än de övriga. Detta medför att hypotesen är besannad.

8. Diskussion

Målet med detta arbete har varit att få en inblick i hur det kan skapas ett kontrollsystem för robotar genom evolution. Detta är ett ämne som har visat sig vara aktuellt under senare tid och är under frammarsch. I tidigt skede visade det sig att mycket forskning behandlade olika typer av lösningar för fotbollsspelande robotar och som gammal fotbollsspelare så hakades denna trend på.

Hur väl lyckades arbetet med att skapa ett kontrollsystem för en fotbollsspelande robot? Resultaten visar på att robotarna delvis klarar av sin uppgift men att betrakta robotarna som fotbollsspelare är svårt att försvara. Att robotarna uppvisar att de kan lokalisera bollen med hjälp av kameratornet och föra bollen framför sig råder det som helst inga tvivel även om detta beteende inte är tillräckligt. Det som saknas är förmågan att kunna förstå att det vid vissa tillfällen inte är tillräckligt att enbart föra bollen framför sig.

Varför de uppvisar ett såpass primitivt beteende har försök gjorts att hitta orsaker till. Analyserna av resultaten visar att detta troligen beror på två orsaker. Det första är att fitnessfunktionen inte har visat sig fungera tillräcklig bra för att roboten skall evolveras mot ett mer utvecklat beteende och den andra orsaken är att startpositionerna för roboten skulle ha hanterats på ett annorlunda sätt. Frågan är dock om dessa två aspekter räcker för att kunna evolvera fram en bättre robot.

En tanke som även har uppstått är hur pass bra hårdvara det krävs för att kunna få fram det beteende som saknas i detta experiment. En Kheperarobot som används i detta experiment är framtagen som en forskningsrobot och det innebär att den har skapats för att vara så pass enkel som möjligt och det kameratorn som kan monteras är också av enklare modell för att den lättare skall kunna simuleras. Frågan lyder då om det bara räcker med den enkla modellen av kameratornet som endast kan uppfatta en platt bild av verkligheten?

Inga tester har utförts på en riktig robot vilket medför att många aspekter är otestade. Hur pass väl simulerade kameran och bollen är mot verkligheten kan ej några svar ges då inga tester har genomförts för att kontrollera detta. Vid skapandet av det simulerade kameratornet har det försökts att följa manualens beskrivning så gott som möjligt och förhoppningsvis skall den simulerade varianten stämma väl överens med den riktiga.

8.1. Framtida Arbete

Efter genomförandet av detta arbete har det funnits flera intressanta aspekter att jobba vidare med:

- Den första aspekten är att verifiera hur pass bra det simulerade kameratornet hanteras mot ett riktigt kameratorn och även hur pass bra simulerad är bollen mot en riktig. Inga tester har utförts utan manualer och videofilmer har lagt grunden för dessa.
- Detta arbete använde sig bara av en robot på planen. På riktig Kheperafotboll sker matcherna 1 mot 1 och detta innebär att tid kommer in som en viktig faktor. Detta arbete tar inte hänsyn till att lösningen skall prestera snabbt vilket kommer att krävas. Detta medför också att andra tekniker kan användas för att evolvera fram robotar, en intressant teknik som har påträffats under detta

arbete är Co-evolution. Co-evolution fungerar så att även under den evolutionära processen kommer två individer och mötas på planen och tanken med detta är att varje individ måste sträva efter att vara bättre än sin motståndare och därmed hjälper de varandra att utvecklas. Østergård (2000) har gjort ett magisterarbete om detta inom Kheperafotboll och det han kom fram till var att genom denna metod så skapades en mer robust lösning än genom en vanlig evolutionär simulering som gjordes i detta arbete.

- Det användes enbart statiska ANN-arkitekturer till robotarna och en undersökning om det antal noder som används till kameran, gömda lager och även hur tillägg av moduler till arkitekturen med spontan modularitet skulle påverka resultaten skulle vara av intresse. Det finns även möjligheter att låta arkitekturen evolveras fram på samma sätt som vikterna evolverades fram i detta arbete och detta skulle kunna ge väldigt bra riktlinjer för hur många noder som krävs.

Referenser

Barone, L. & While, L. (1997) Evolving computer opponents to play a game of simplified poker. Proceedings of the University of Western Australia Department of Computer Science 1996 Department Research Conference.

Bourke, P. (1989) *Intersection point of two lines (2D).*Tillgänglig på Internet: http://astronomy.swin.edu.au/~pbourke/geometry/lineline2d/ [Hämtad 2002.03.22].

Bourke, P. (1992) *Intersection of a line and a sphere (circle).* Tillgänglig på Internet: http://astronomy.swin.edu.au/~pbourke/geometry/sphereline/ [Hämtad 2002.03.22].

Carlsson, J. (1999) Evolution av modulära neuronnät för styrning av en mobil robot. Examensarbete för programmet för systemprogrammering. Institutionen för datavetenskap, Högskolan i Skövde.

Carlsson, J. & Ziemke, T. & Bodén, M. (1999) An experimental comparison of weight evolution in neural Control architectures for a 'Garbage-collecting' Khepera robot. *Löffler, Mondada & Rückert (eds.) Experiments with the Mini-Robot Khepera - Proceedings of the 1st International Khepera Workshop. Paderborn, Germany: HNI-Verlagsschriftenreihe.*.

Carlsson, J. & Ziemke, T. (2001) YAKS- Yet Another Khepera Simulator. Rückert, Sitte & Witkowski (Eds.) Autonomous Minirobots for Research and Entertainment – Proceedings of the 5th International Heinz Nixdorf Symposium, Paderborn, Germany: HNIVerlagsschriftenreihe.

Cliff, D. Harvey, I. & Husbands, P. (1993) Explorations in evolutionary robotics. *Adaptive behaviour*, 2, 73-110.

Fogel, D. B. & Michalewicz, Z. (2000) How to solve it, modern heuristics. Springer-Verlag Berlin Heidelberg.

K-team. Mobile robotics for research, education and industrial applications. Tillgänglig på Internet: http://www.k-team.com [Hämtad 2002.02.10].

Kitano, H. (1997) Robocup -97: Robot soccer world cup 1. Springer-Verlag Berlin.

Koza, J. R. (1992) Genetic Programming: On programming on computers by means of natural selection. MIT Press.

Lund, H. H. (1999) *Rules for Khepera robot football*. Tillgänglig på Internet: http://www.mip.sdu.dk/~hhl/robotfootballrules.html [Hämtad 2002.02.10].

Meeden, L. A. & Kumar, D. (1998) Trends in evolutionary robotics. *Soft Computing for intelligent robotic systems*, 215-233.

Mehrotra, K. Mohan, C. K. & Ranka, S. (1997) Elements of artificial neural networks. MIT Press.

Mitchell, M. (1996) An introduction to genetic algorithms. MIT Press.

Niklasson, L. & Ziemke, T. (1996). Självlärande datorer, utopi eller verklighet? *Ung Forskning*.

Nolfi, S. (1996) Using emergent modularity to develop control systems for mobile robots. *Adaptive behaviour*, 5, 343-363.

Nolfi, S. (1998) Evolutionary robotics: Exploiting the full power of self-organization. *Connection Science*, 10, 167-183.

Nolfi, S. & Floreano, D. (2000) Evolutionary robotics. MIT Press.

Smith, T. (1997) Adding vision to Khepera: an autonomous robot footballer. Master thesis. School of cognitive and computing sciences, University of Sussex.

Ziemke, T. (2001) Are Robots Embodied?. Balkenius, Zlatev, Brezeal, Dautenhahn & Kozima (Eds.). *Proceedings of the First International Workshop on*

Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems (pp. 75-93). Lund University Cognitive Studies, Vol. 85, Lund, Sweden

Østergård, E. Lund, H. H. & Pagliarini, L (2000) *Co-evolutionary robot soccer show.* Tillgänglig på Internet: http://www.legolab.daimi.au.dk/cerss/ [Hämtad 2002.02.10].

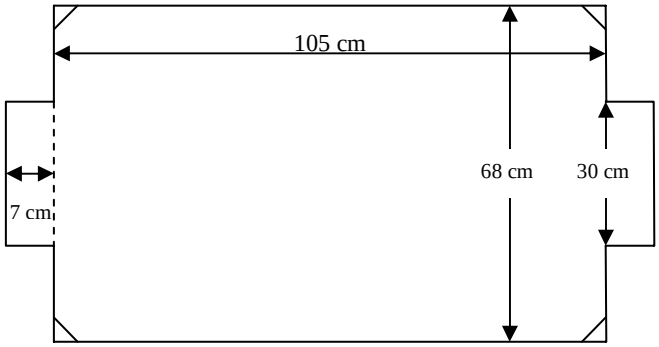
Østergård, E. (2000) Evolving complex robot behaviour. Master thesis. Department of computer science, University of Aarhus.

Bilaga A
Regler för Kheperafotboll

Dessa regler är de som användes i Danska mästerskapet i robot-fotboll, Lund (1999).

Planen

Golvet: Grönt, 105cm långt och 68cm brett med fasade hörn.
Väggar: Gråa, 20cm höga.
Mål: Svarta, 30cm breda, 7cm djupa och 15cm höga.
Ljus: Starkt, enhetligt ljus från ovan. Till exempel 2*500 watt halogen. Ljuset skall ge möjligheten för en Kheperarobot utrustad med ett K213 kamera torn att kunna särskilja de tre färgerna svart, grå och bollens färg (vit eller gul).



Figur 14. Översikt på fotbollsplanen.

Boll

En vit eller gul tennisboll.

Spelare

Kheperarobotar utrustade med kameratorn. Endast en spelare på plan per lag vid varje match. Varje spelare skall bära en svart- och vitrandig dräkt med ränderna i vertikal längdriktning. Dräkten skall även ha ett hål för kameran. De två spelarna ska även kunna urskiljas från ovan genom olika färger och även varje robots framåtriktning skall märkas ut med en pil (se figur 15).

Figur 15. Verklig bild av fotbollsplan under en match.

Matchstart

Bollen placeras vid planens centrum medan spelarna placeras ut på symmetriska positioner på planen. Varje spelare kommer att kunna placeras varsomhelst på planen fast den är alltid riktad mot motspelarens mål. När domaren blåser i visselpipan får roboten startas genom aktivering av de bakre IR-sensorerna.

Match

En match består av 5 rundor av max 4 min. Varje runda stoppas tidigare om någon robot gör ett mål (mer än halva bollen över mållinjen) eller om inte bollen har rört sig på 30 sekunder. Om matchen blir oavgjord så avgörs den på straffslag.

Straffslag

Bollen placeras på mittpunkten med roboten placerad 10 cm bakom denna. Roboten är riktad mot bollen och den får 30 sekunder på sig att försöka göra mål. Rundor om ett försök per spelare görs tills någon har vunnit.

Bilaga B

Tillägg av funktionalitet till YAKS

För att klara av att utföra de experiment som utfördes i detta arbete var det tvunget att lägga till funktionalitet till YAKS. Det som saknades i YAKS var en utvecklad simulering av kameratornet och även möjligheten att simulera en fotboll. Denna bilaga är tänkt att förklara hur tillvägagångssättet såg ut för att skapa dessa två aspekter. Det som tas upp här är endast en 2-dimensionell lösning men att göra om detta till 3-dimensioner skall inte innebära några större problem.

Kameratornet

Kameratornet till Kheperaroboten är relativt enkelt uppbyggt (se kapitel 2.2.1). Grundidén är att roboten ser en linjär bild framför sig med på 1*64 pixlar med cirka 36º vinkel, där varje pixel får ett gråskalevärde (0-255) lagrat. Lösningen bygger på idéer hämtade ifrån ”raycasting” principer med att ljusstrålar skickas ut från en ljuskälla som bryts mot föremål i världen. På liknande sätt hanteras kameratornet genom att varje pixel skapar en linje med en viss vinkel som skickas ut i världen.

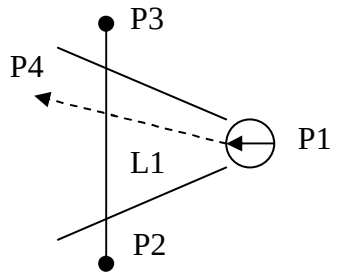
Grundprincipen ser ut som följande:

```
for(varje kamerapixel)
     for(varje primitiv i världen)
          if(pixeln ”ser” primitiven och primitiven är närmare än
          övriga primitiver)
                  Tilldela pixeln primitivens färg
```

Då YAKS endast hanterar två olika sorters geometriska figurer (cirklar och linjer) så krävs det endast två typfall, linje/linje skärning och linje/cirkel skärning för att kontrollera om pixeln ”ser” primitiven.

Linjer

I YAKS är det bara för tillfäller väggar som representeras av linjer.



Figur 16. Enkel vy över roboten och en vägg.

För att kontrollera om kameratornet ser en vägg så skapas en linje *L1* ifrån kameratornets pixel *P1* till kameratornets maximala vylängd (i vanliga fall 500mm) med den vinkeln pixeln har *P4* (se figur 16).

För att beräkna punkten *P4* används Pytagoras sats, då en punkt representeras av en (x,y) koordinat beräknar vi dessa för sig.

- x4 = x1 + (Kamerans maximala vylängd + cos α)
- y4 = y1 + (Kamerans maximala vylängd + sin α)

Där α representerar linjen *L1´*s vinkel.

För att sedan beräkna om linjen mellan punkterna *P1* och *P4* och linjen mellan punkterna *P2* och *P3* skärs används en formel som Paul Bourke har tagit fram, Bourke (1989).

Linjernas ekvationer ser ut som följande:

$$PA = P1 + ua(P4 - P1)$$
$$PB = P2 + ub(P3 - P2)$$

Formel 2. Linjernas ekvation.

Genom att lösa ekvationen *PA = PB* får man reda på om linjerna skär varandra, dock leder detta till att två okända variabler (*ua* och *ub*) bildas.

Dessa kan dock lösas ut:

$$ua = \frac{(x4 - x3)(y1 - y3) - (y4 - y3)(x1 - x3)}{(y4 - y3)(x2 - x1) - (x4 - x3)(y2 - y1)}$$
$$ub = \frac{(x2 - x1)(y1 - y3) - (y2 - y1)(x1 - x3)}{(y4 - y3)(x2 - x1) - (x4 - x3)(y2 - y1)}$$

Formel 3. Lösning av variabler.

Genom att lösa dessa ekvationer kan sedan skärningspunkten räknas fram:

$$x = x1 + ua(x4 - x1)$$
$$y = y1 + ub(y4 - y1)$$

Formel 4. Skärningspunkt.

I formel 3 kan noteras att:

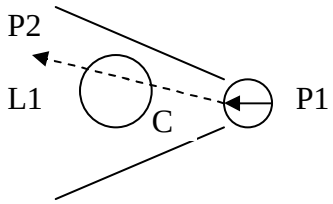
- Bägge nämnarna är likadana.
- Om nämnaren är lika med noll så är linjerna parallella.
- Linjerna skärs om både *ua* och *ub* ligger mellan 0 och 1.

Genom att tillämpa dessa formler kan det relativt enkelt fås fram om linjerna skär varandra och om de gör detta kan man tilldela kameratornets pixel den färg som väggen har. Eftersom man även kan räkna fram skärningspunkten kan även väggarna

ha den möjligheten att vara flerfärgade, detta är dock upp till väggens funktionalitet att då returnera rätt färgvärde.

Cirkel

I YAKS är de flesta primitiverna representerade av cirklar, till exempel robotar, den nyskapade bollen, flyttbara och icke flyttbara föremål.



Figur 17. Enkel vy över roboten och ett runt föremål.

För att kontrollera om kameratornet ser ett runt föremål så skapas en linje *L1*, precis som vid väggarna, ifrån kameratornets pixel *P1* till kameratornets maximala vylängd (i vanliga fall 500mm) med den vinkeln pixeln har till en punkt *P2* (se figur 17).

För att beräkna punkten *P2* används pytagoras sats, då en punkt representeras av en (x,y) koordinat beräknar vi dessa för sig.

- x2 = x1 + (Kamerans maximala vylängd + cos α)
- y2 = y1 + (Kamerans maximala vylängd + sin α)

Där α representerar linjen *L1´*s vinkel.

För att sedan beräkna om linjen *L1* och cirkeln *C* skär varandra används en formel som Paul Bourke har tagit fram, Bourke (1992).

Skärningspunktens ekvationen kan representeras som följande:

$$x = x1 + u(x2 - x1)$$
$$y = y1 + u(y2 - y1)$$

Formel 5. Skärningspunktens ekvation.

Genom att representera cirkeln med följande ekvation: (där *(x,y)* representerar skärningspunten, *(x3, y3)* representerar cirkelns mittpunkt och *r* representerar cirkelns radie)

$$(x - x3)^2 + (y - y3)^2 = r^2$$

Formel 6. Cirkelns ekvation.

Genom att byta ut *x* och *y* i formel 5 med ekvationerna ifrån formel 4 fås en kvadratisk ekvation $au^2 + bu + c = 0$ där:

$$a = (x2 - x1)^2 + (y2 - y1)^2$$
$$b = 2((x2 - x1)(x1 - x3) + (y2 - y1)(y1 - y3))$$
$$c = x3^2 + y3^2 + x1^2 + y1^2 - 2(x3x1 + y3y1) - r^2$$

Formel 7. Parameterlösning av den kvadratiska ekvationen.

Om Linjen skär med cirkeln kontrolleras genom att beräkna:

$$b^2 - 4 * a * c$$

Formel 8. Kontroll ekvation.

Om resultatet ifrån formel 8:

- är mindre än noll så sker ingen skärning.
- är lika med noll så är linjen en tangent till cirkeln och skär i en punkt.
- är större än noll så sker en skärning i 2 punkter.

Dock räknar dessa formler med en oändlig linje, detta gör att om det sker skärningar mellan linjen och cirkeln så görs det en efterkontroll som kontrollerar att skärningspunkten ligger mellan ändpunkterna på ursprungslinjen, annars får roboten både ögon i nacken och ett oändligt långt siktfält.

Bollen

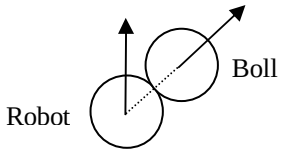
För att roboten skall kunna spela fotboll krävs det att en boll kan simuleras i YAKS, vilket det inte gör i dagsläget. Därför har en sådan skapats.

Eftersom det redan finns stöd för ett runt flyttbart objekt i YAKS så utnyttjas detta till att skapa grunden för bollen. Det som skiljer detta objekt och den tänkta bollen är att det existerande objektet är skapat för att Kheperaroboten skall kunna flytta runt det med sin griparmsmodul, detta innebär att den inte kan rulla och studsa mot andra föremål vilket krävs för en boll. Alla aspekter för att skapa en boll kommer inte att gås igenom i detta arbete, detta på grund av att den boll som skapats till YAKS inte är speciellt avancerad.

Bollen är anpassad så att den enda energikällan som kan få bollen i rörelse är själva Kheperaroboten, det är alltså bara roboten som kan få bollen att rulla. När bollen är i rörelse kan den enbart studsa mot väggar för tillfället, en förbättrad kollisions och reflektionshanterare är på gång men denna blev inte färdig innan det var tvunget att påbörja simuleringarna.

Kollision med robot

När roboten kolliderar med bollen kommer alltså bollen att hamna i rörelse.

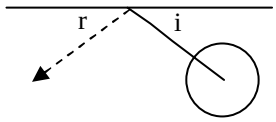


Figur 18. Kollision mellan boll och robot.

För att beräkna när kollision sker mellan robot och boll är relativt enkelt. Genom att beräkna avståndet mellan centrum på bollen och centrum på roboten och om detta avstånd är mindre än robotens radie + bollens radie så har det skett en kollision. För att veta vilken vinkel bollen ska rulla iväg åt dras en tänkt linje mellan centrum på roboten och centrum på bollen, vinkeln denna linje får är den vinkel som bollen skall rulla iväg åt (se figur 18). En detalj som inte hanteras speciellt bra är rörelseenergin som överförs till bollen ifrån roboten baseras enbart på hastigheten ifrån roboten, bollen får alltså inte lägre rörelseenergi om roboten snuddar vid bollen än om den kör rakt in i den. Troligtvis har denna detalj inte så stor inverkan på resultaten då det inte handlar om några stora krafter som roboten påverkar bollen med (bollen är större och tyngre än vad själva roboten är).

Kollision med vägg

Denna detalj har gjorts väldigt enkel, ifrån fysikens lagar så är infallsvinkeln lika med reflektionsvinkeln och detta är det faktum som används här (se figur 19).



Figur 19. Bollens reflektionsvinkel.

Detta är alltså den enda primitiv i världen som bollen kan studsa emot, när det gäller roboten så är det roboten som kolliderar med bollen inte vice versa.