NTNU
Det skapende universitet

Biologisk inspirert maskinvare

Utforskning av evolusjon for kunstig utvikling

Joacim Thomassen

Master i datateknikk
Oppgaven levert: Juni 2006
Hovedveileder: Gunnar Tufte, IDI

Norges teknisk-naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

Oppgavetekst

Biologisk inspirert maskinvare: Utforskning av evolusjon for kunstig utvikling

Kunstig utvikling av store systemer som innehar en form for oppførsel er en vanskelig oppgave. Årsaken kan sies å være at søkerommet til de fleste problemer er stort og uoversiktelig. Derfor er det vanskelig å oppnå vellykkede søk med evolusjonær utforskning. Det store søkerommet er ofte et resultat av genotypstørrelse. En stor genotyp må representere en stor fenotyp. Størrelsen på genotypen er kun del av problemet. Et større problem kan være at søkerommet gitt av representasjonen er uoversiktlig og vanskeliggjør et evolusjonært søk.

Inspirasjon fra biologisk utvikling introduserer en indirekte genotype-fenotype omforming. En mulighet er å bruke en relativt liten genotyp for å representere en stor fenotyp. Dette medfører et redusert søkerom, men en tilnærming med bruk av utvikling kan introdusere et desto vanskeligere søkelandskap.

For å utvikle store komplekse systemer, med eller uten indirekte omforming, er det nødvendig å finne en utviklingsvennlig representasjon.

Basert på den kunstige utviklingsprosessen trengs en serie eksperimenter for studering av genomrepresentasjon. Arbeidet kan innbefatte design av genomrepresentasjon det vil si den interne representasjonen i en evolusjonær algoritme, og generiske algoritmer som krysnings- og mutasjonsoperatorer. En eksperimentell tilnærming kan innbefatte implementering av en plattform for å utforske evolusjon for utvikling av emergent strukturer.

Oppgaven gitt: 20. januar 2006
Hovedveileder: Gunnar Tufte, IDI

Forord

Denne oppgaven er det skriftlige resultatet av min avsluttende hovedoppgave ved Institutt for datateknikk ved NTNU. Arbeidet er utført for Gunnar Tufte.

I oppgaven har jeg laget et program i programmeringsspråket C++. Programmet simulerer deler av et evolusjonært maskinvaresystem laget av Gunnar Tufte for kunstig utvikling i en cellulær maskin. Målet med oppgaven var å lage et program som kunne benyttes som eksperimentplattform for eksperimentell analyse av et system basert på evolusjon, for å oppnå utvikling av strukturer gjennom *emergent computation*. Jeg håper programmet kan være nyttig i videre forskning og eksperimentelle analyser. Programmet er derfor gitt ut under GNU General Public License og er tilgjengelig fra [19].

Jeg vil takke min veileder Gunnar Tufte for god oppfølging gjennom oppgaven og for å få lov til å presentere artikkelen med resultater fra programmet på GECCO konferansen 2006 i Seattle. Artikkelen er vedlagt i appendiks B.

Sammendrag

Rapporten beskriver hovedoppgaven *Utforskning av evolusjon for kunstig utvikling.*

Cellulære beregningsmaskiner og kunstig utvikling er begge nyere forskningsfelt innen datateknologi med opphav i biologisk inspirasjon og kunstig intelligens. Biologisk inspirerte metoder kan benyttes til å finne nye alternative design. Kunstig evolusjon kombinert med cellulære maskiner kan gi opphav til nye designmetoder og arkitekturer. Samtidig kan cellulære maskiners dynamiske egenskaper gi gode simuleringer av naturens komplekse systemer.

Kunstig utvikling ser ut til å kunne redusere skaleringsproblemet ved genotype-fenotype omformingen for evolusjonære systemer. Koblingen av evolusjon og kunstig utvikling byr likevel på nye utfordringer. En cellulær maskin for utvikling kan utforskes for *emergent computation* med evolusjon. Et slikt komplekst system krever forståelse for hvordan den biologiske utviklingen med genuttrykk og genregulatoriske nettverk påvirkes av evolusjonen og evolusjonsparameterene. Kunnskap om et slikt system forutsetter eksperimentell utforskning og analyse.

I oppgaven ble en plattform implementert for å utforske evolusjon for utvikling av *emergent* strukturer i en cellulær maskin. Plattformen brukes til å designe to multicellulære *emergent* strukturer, sjakkbrett- og flaggmønster. Analyseresultatene viser at evolusjonen konsekvent utnytter omgivelsene sine for å skape strukturløsninger. Det ble også utført skaleringsforsøk som tyder på at økt informasjonsmengde i systemet kan ha en positiv effekt på det evolusjonære søket.

Plattformen gjør det mulig å få en eksperimentell forståelse av hvordan parametere ved utviklings- og evolusjonsprosessene påvirker systemets *emergent computation* av strukturer.

Innhold

Forord i

Sammendrag iii

1 Innledning 1

2 Biologisk inspirasjon 5
- 2.1 Biologiske systemer . . . 5
- 2.2 Phylogeny (evolusjon) . . . 6
- 2.3 Ontogeny (utvikling) . . . 7
- 2.4 Epigenesis (læring) . . . 7

3 Evolusjon 9
- 3.1 Evolusjonære beregninger . . . 9
- 3.2 Evolusjonære Algoritmer . . . 10
  - 3.2.1 Representasjon (genom) . . . 11
  - 3.2.2 Evaluering (fitnessfunksjon) . . . 12
  - 3.2.3 Populasjon . . . 13
  - 3.2.4 Reproduksjonsseleksjon . . . 13
  - 3.2.5 Variasjonsoperatorer . . . 14
  - 3.2.6 Overlevelsesseleksjon . . . 15
  - 3.2.7 Initialisering og terminering . . . 16
- 3.3 Genetiske algoritmer . . . 16

4 Utvikling 19
- 4.1 Genuttrykk - tolkning av genotypen . . . 20
  - 4.1.1 Skalering . . . 21
  - 4.1.2 Nøytrale nett . . . 22
- 4.2 Genregulering - konstruksjon av fenotypen . . . 22
  - 4.2.1 Cellulær automata . . . 23
  - 4.2.2 Regelbasert naboskap . . . 23

4.2.3 Turingkomplett 24
4.3 Cellulær beregningsmaskin for utvikling 25

5 Plattform for eksperiment **27**
5.1 Plattformens oppbygning 28
5.1.1 Evolusjon 28
5.1.2 Utvikling 30
5.1.3 Evaluering for sjakkbrettindivid (neutral) 32
5.1.4 Evaluering for flaggindivid 33
5.2 Oppsett og bruk 38

6 Skalering og sjakkbrett **41**
6.1 Oppsett av forsøk 42
6.2 Resultat 42
6.3 Skalering av resultat 45

7 Tallforhold **47**
7.1 Oppsett av forsøk 48
7.2 Resultat 48
7.2.1 Tallforhold mellom to celletyper 49
7.2.2 Tallforhold mellom tre celletyper 55
7.2.3 Tallforhold og feilandel oppsummert 56

8 Flaggmønster **57**
8.1 Oppsett av forsøk 58
8.2 Resultat 58
8.3 Skalering av resultat 63

9 Analyse og diskusjon **73**
9.1 Skalering og sjakkbrett 73
9.2 Tallforhold eksperiment 74
9.3 Flaggmønster eksperiment 74
9.4 Systemanalyse 75

10 Videre arbeid og konklusjon **77**
10.1 Videre arbeid 77
10.2 Konklusjon 78

A Konfigurasjonsfil **79**

B Artikkel **81**

Bibliografi 94

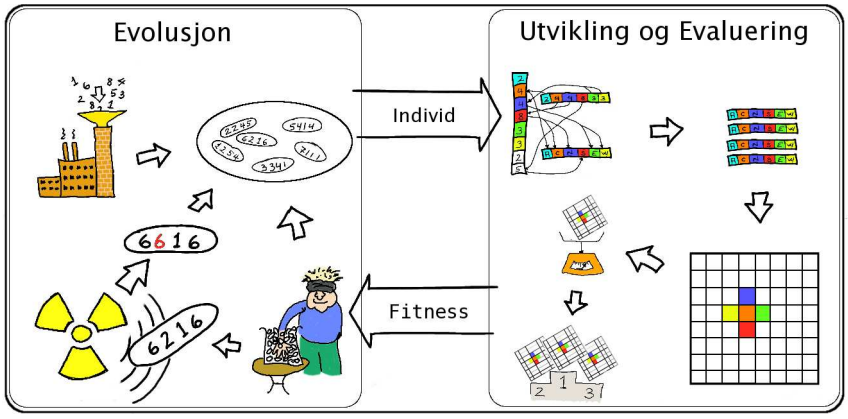
Kapittel 1

Innledning

Betydningen av relasjonen mellom genetikk og utvikling har først blitt forstått de siste 30 årene [23]. Molekylærbiologien har gjort det langt enklere å studere det genetiske grunnlaget for utvikling og genenes rolle i kontrollen av utviklingsprosessen. I hver organisme styrer genene utvikling, oppførsel og død. Genene er resultat av evolusjon og samtidig grunnlag for videre evolusjon.

Naturens mangfold er åpenbar i våre omgivelser. Det ser ut til at naturen utnytter mekanismer som evolusjon og utvikling for å skape dette mangfoldet av suksessrike organsimer. Forskning innen evolusjonær maskinvare, kunstig intelligens og kunstig liv forsøker å imitere disse mekanismene. Dette gjøres blant annet for å utvikle nye elektriske kretser og for å simulere biologske komplekse systemer.

I oppgaven implementeres en plattform for å dyrke fram digitale multicellulære organismer. Biologisk inspirerte mekanismer som evolusjon og utvikling benyttes i form av de digitale tilnærmingene evolusjonære beregninger og cellulære maskiner. Den digitale dyrkingen gjøres med *emergent computation*. De digitale multicellulære organismene representeres med et cellulært automata system tilsvarende en parallell cellulær maskin. Tilsynekomsten av nye organismer drives av de evolusjonære beregningene. Det opprinnelige systemet plattformen simulerer deler av er realisert i maskinvare (evolusjonær maskinvare). Plattformen illustrert i figur 1.1 ble utviklet for å muliggjøre eksperimentelle oppsett av evolusjons- og utviklingssystemet for å oppnå ulike *emergent* cellestrukturer. Dataene fra disse eksperimentene er tenkt å kunne gi grunnlag for videre analyse og eksperimenter. Utgangspunktet er behovet for empirisk forståelse av et komplekst system hvor evolusjonen og utviklingens relasjoner kan studeres. Dette ansees som viktig for videre forskning i [16, 20].



Figur 1.1: Prosessene på plattformen.

Det er utført tre ulike eksperimenter på plattformen. Eksperimentene er gjort underveis i implementeringen og har dannet grunnlag for videre utvikling av plattformen til slik den foreligger i dag. Resultatene og erfaringene fra analysen underveis har altså utgjort basis for hvordan plattformen fremstår i dag. Det er gjort forsøk med *emergent* sjakkmønster og skalerbarheten av dette mønsteret når fenotypestørrelsen øker. Resultatene av disse skaleringsforsøkene er presentert i artikkelen i appendiks B. Videre er det utført et eksperiment med tallforhold i et forsøk på å rettlede evolusjonen til å skape *emergent* strukturer med ønsket forhold mellom celletypene. Erfaringene fra disse forsøkene benyttes så i siste forsøk der et flaggmønster realiseres.

Wolperts *French fag* eksempel i [23] illustrerer utviklingsprosessen for utvikling av det franske trikolor flagget. Eksempelet har siden blitt demonstrert i digitale systemer i [14, 16]. Disse tidligere eksperimentene benytter systemer som imiterer biokjemiske signaler. I siste eksperiment i denne oppgaven blir eksempelet gjentatt for et tilsvarende triokolor flaggproblem, men med et enklere utviklingssystem med kun CAens enkle naboskap.

I kapittel 2 presenteres POE-modellen for inndeling av biologisk maskinvaresystemer. Det gis en oversikt over evolusjonsmekanismer i digitale systemer i kapittel 3. En tilsvarende oversikt for utvikling er gitt i kapittel 4 som avsluttes med en kort beskrivelse av et system som kombinerer utvikling og evlolusjon i en cellulær beregningsmaskin for utvikling styrt av evolusjon i avsnitt 4.3. Plattformen for å eksperimentere med et slikt system presenteres i kapittel 5. Eksperimentene for sjakkbrettmønster og skalering er presentert i kapittel 6, eksperimentene med tallforhold i kapittel 7 og eksperimentene med flaggmønsteret i kapittel 8. Anal-

yse og diskusjon av eksperimentet og systemet er gitt i kapittel 9. Oppgaven avsluttes med en betraktning av systemet for videre arbeid og konklusjon på oppgaven i kapittel 10.

Kapittel 2

Biologisk inspirasjon

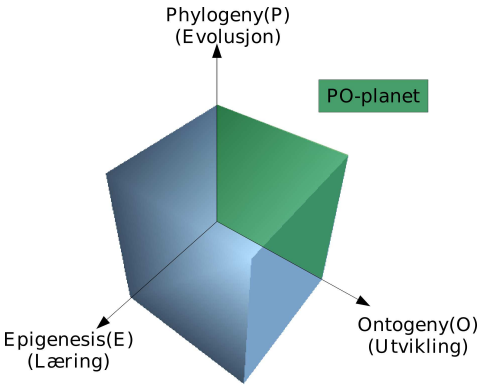
Naturens skaperevne er synlig i all dens mangfold. I håp om å kunne utnytte lignende skaperkrefter og oppnå tilsvarende robuste og suksessrike løsninger forsøker en å gjenskape naturens mekanismer i menneskeskapte systemer. Menneskene vil kanskje aldri forstå naturens helhet, men har forsått nok til å ta lærdom av og etterligne dens mekanismer.

I den senere tid har forskning innen biologi gitt oss et dypere innblikk i naturens komplekse systemer. Naturens løsninger har gradvis utviklet seg over mangfoldige år og er en grunnleggende inspirasjonskilde. Med inspirasjon fra biologien dukker det stadig opp alternative og spennende løsninger som bidrar til nye både teknologiske og biologiske oppdagelser. Det dyrkes i dag en rekke digitale individer verden over [2] takket være Darwins evolusjonslære.

Biologisk inspirerte systemer kan realiseres digitalt både i maskinvare og programvare. Programvareplattformen i denne oppgaven simulerer en maskinvareløsning og kan derfor karakteriseres i henhold til POE-modellen i figur 2.1. Plattformen simulerer maskinvare i PO-planet, siden systemet innehar mekanismer for både utvikling og evolusjon.

2.1 Biologiske systemer

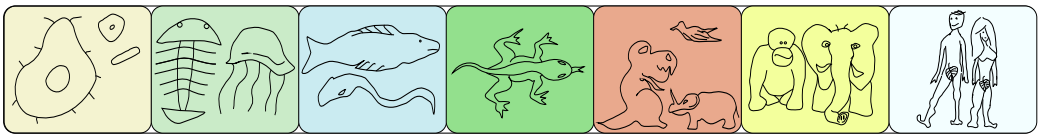
Livet på jorden kan se ut til å ha tre retninger å innordne seg i [18]. Disse inndelingene har alle utspring i en felles faktor som finnes i alle levende organismer, det genetiske programmet (genomet). Genomet er resultat av og samtidig



Figur 2.1: POE-modellen viser de biologisk inspirerte maskinvaresystemene i henhold til sin plassering langs de tre aksene pyhlogenetic (evolusjon), ontogenetic (utvikling) og epigenesis (læring) [18].

aktiv del av evolusjonen. Det styrer utviklingen, oppførsel og død ved hver organisme. Biologisk inspirerte maskinvaresystemer innehar og imiterer i ulik grad egenskaper ved levende organismer som kan innordnes i en eller flere av disse retningene: *pyhlogenetic*, *ontogenetic* og *epigenesis*. Denne modellen for inndeling av maskinvaresystemer kalles POE-modellen og er vist i figur 2.1.

2.2 Phylogeny (evolusjon)

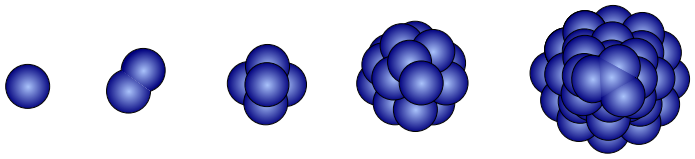


Figur 2.2: Phylogeny her illustrert gjennom perioder av livet på jorden hvor ulike arters tilsynekomst er resultat av evolusjon og endringer i omgivelsene.

Phylogeny omhandler opphav og videreføring av individgrupper, som populasjoner og arter [15]. Det innebærer endring av det genetiske materialet fra generasjon til generasjon, altså evolusjon. Artenes framvekst er avhengig av reproduksjon og videreføring av genomet uten for store variasjoner innad mellom individer av samme art. Dermed vil også avkommene tilhøre samme art uten for store endringer i arvematerialet. Nye organismer kan oppstå som følge av aseksuell reproduksjon (mutasjon) eller seksuell reproduksjon (rekombinasjon og eventuelt mutasjon). Disse to evolusjonære mekanismene er ikke deterministiske og dermed

kilder til variasjon. Uten variasjon kan ikke arten tilpasse seg endringer i sine omgivelser og utvikle seg eller gi opphav til nye arter.

2.3 Ontogeny (utvikling)



Figur 2.3: Ontogeny illustrert ved tidlig celledeling.

Ontogeny omhandler utviklingen av enkeltindividet fra det befruktede egget (genotypen) til ferdigvokst stadie (fenotypen) og død [15]. Dette gjelder for alle multicellulære organismer og begynner med at det befrukta egget (zygoten) starter sin celledelingssyklus. Denne prosessen kan sees på som hovedsaklig deterministisk da det ikke er rom for feil under kopieringen av genomet. En slik feil vil svært ofte føre til store endringer og ofte dødelige misdannelser [18].

2.4 Epigenesis (læring)

Epigenesis omhandler utviklingen som ikke kodes i arvematerialet. Det er her snakk om utveksling av erfaringer og informasjon i løpet av livssyklusen til organismene. Dette er altså systemer for læring. En finner eksempler på slike systemer blant annet i immunsystemet, nervesystemet og det endokrine systemet. Denne retningen kommer ikke til å bli fulgt videre i oppgaven da maskinvaresystemet kun vil være i PO-planet utgjort av Phylogeny og Ontogeny aksene. Læring omfatter individets adferdsendring på grunnlag av inntrykk fra sine omgivelser gjennom sansene, slik et barn lærer sitt språk ved å ape etter og forstå sine foreldre.

Retningene phylogeny og ontogeny betegnes heretter som henholdsvis evolusjon og utvikling. Realiseringen av mekanismene i evolusjons- og utviklingsprosessen utdypes i kapittel 3 og 4 før implementasjonen av disse i plattformen presenteres i kapittel 5.

Kapittel 3

Evolusjon

Naturlig seleksjon har siden starten av den menneskelige sivilisasjon blitt utnyttet for å velge ut gode avlsdyr og planter [15]. Basert på uttrykte egenskaper hos organismene, som sterke arbeidshester og gode verpehøns, har menneskenes kunstige seleksjon gitt oss arter som egner seg bedre, for eksempel husdyr og mer hardføre såkorn. Som et resultat av menneskenes stadige forsøk på å forme sine omgivelser har en i dag menneskeskapte digitale verdener. Kunstig seleksjon og andre evolusjonære mekanismer har vist seg å gi gode resultater også i formingen av disse digitale omgivelsene [2]. Evolusjonen framstår som en allsidig metode for å finne gode og alternative løsninger på mange ulike problemer. Selv om evolusjonen fortsatt ikke forstås fullt ut kan nå flere av dens tidligere mystiske egenskaper begrunnes med kunnskap fra felt som biologi, matematikk, statistikk og kunstig intelligens.

3.1 Evolusjonære beregninger

Evolusjon brukt i et datasystem går under betegnelsen evolusjonære beregninger. Metaforen evolusjonære beregninger [9], vist i tabell 3.1, innebærer et system med definerte omgivelser. I omgivelsene finnes en populasjon av individer som forsøker å overleve og reprodusere seg. Hvert individs fitness bestemmes av i hvilken grad det har nådd sitt mål i disse omgivelsene. Fitnessen viser dermed hvor stor sjanse individet har for å overleve og formere seg. Det er med andre ord en prosess for problemløsning i form av en stokastisk prøv-og-feil tilnærming, der en til en hver tid har et sett med løsningskandidater. Kvaliteten avgjør sjansen

for å bli beholdt videre og brukt som utgangspunkt for videre løsningskandidater.

Evolusjon		Problemløsing
Omgivelser	<–>	Problem
Individ	<–>	Løsningskandidat
Fitness	<–>	Kvalitet

Tabell 3.1: Basismetaforen innen evolusjonære beregninger for koblingen mellom naturlig evolusjon og problemløsning [9].

3.2 Evolusjonære Algoritmer

Det finnes flere ulike former for evolusjonære beregninger. En av disse er evolusjonære algoritmer (EA), se eksempel i algoritme 3.1. Disse brukes ofte som optimaliseringsmetoder for å finne beste løsning. Evolusjonære algoritmer er gode generelle problemløsere og kan brukes på mange ulike problemer [8, 9]. Selv om andre algoritmer kan gi en mer optimal løsning når problemet har blitt undersøkt nærmere og algoritmen tilpasset, gir EAer ofte gode løsninger innen rimelig tid uten tilsvarende problemspesifikk tilpasning. De er dermed svært nyttige i dagens situasjon der automatiske problemløsere blir desto viktigere for å holde tritt med det økende antall problemer og den økte kompleksiteten.

```
skap populasjon
evaluer individene
while STOPP-KRITERIET-IKKE-NÅDD do
  selekter individpar
  kryss individpar
  muter avkom
  evaluer avkom
  innsett avkom eller individpar i nesteGenerasjon
end while
```

Algoritme 3.1: En type evolusjonær algoritme er den genetiske algoritmen her vist på en typisk generell form.

Selv om EAer ofte kan gi løsninger på et generelt utvalg av problemer, må likevel algoritmen tilpasses hvert problem for å oppnå tilfredsstillende resultater. Hva en vil oppnå og hvordan en implementerer en EA for et gitt system er viktig. Evolusjon vil ikke gi de resultatene en ønsker hvis den ikke ledes i riktig retning

og på de rette premissene. Hvordan en representerer arvematerialet algoritmene skal gjennomsøke og gjøre sine genetiske operasjoner på vil være med og avgjøre om det er mulig å komme fram til en løsning.

Rutinene for hvordan kunstig evolusjon kan gjennomføres etterligner ulike aspekter ved den naturlige evolusjonen. Hvor naturtro disse algoritmene er varierer i stor grad av om de benyttes for å gi mer innsikt i sider ved den naturlige evolusjonen eller om de kun er et hjelpemiddel for å finne alternative eller optimale løsninger på et spesifikt problem.

EAer må inneholde visse komponenter, prosedyrer og operatorer som vil være spesifikke for hver enkelt algoritme [9]. De viktigste komponentene er:

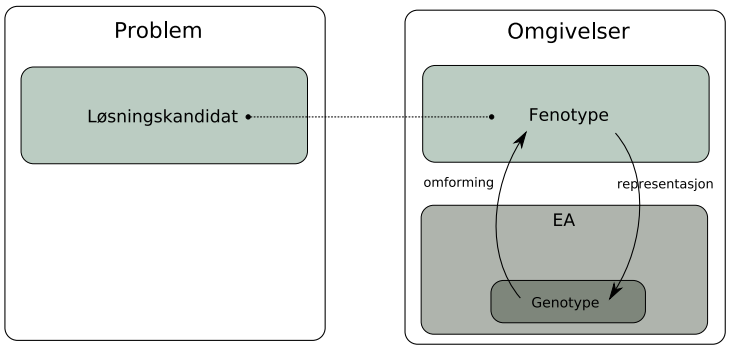
- Representasjon (defineringen av individene)
- Evaluering (fitnessfunksjonen)
- Populasjon
- Reproduksjonsseleksjon
- Variasjonsoperatorer (kryssning og mutasjon)
- Overlevelsesseleksjon (erstatning)

Hver av disse komponentene må spesifiseres for å definere en EA. De genetiske operatorene algoritmen har til rådighet er variasjonsoperatorene, reproduksjonsseleksjon og overlevelsesseleksjon. I tillegg kan individene klone seg selv. Det vil si lage en identisk kopi av sitt eget genom. For å bruke EAen er det en forutsetning at initialiseringsprosedyren og stoppkriteriet er definert.

3.2.1 Representasjon (genom)

For å kunne løse problemet i problemløsningsrommet (omgivelsene) til EAen, må en representasjon (innkoding) av løsningskandidaten etableres [9]. Løsningskandidaten kalles fenotypen og representeres som et individ i EAen kalt genotypen illustrert i figur 3.1. Det er denne innkodingen av fenotypene ned til et sett med genotyper som utgjør representasjonen.

Et eksempel på dette kan være et fargenyanseproblem der et sett med farger utgjør fenotypene. Hver farge kan da representeres med tre heltallsverdier mellom 0 og 255 for de tre basisfargene rød, grønn og blå. Den sterkeste rødfargen vil da kunne representeres som genotypen 255-0-0. Det er viktig å merke seg at fenotyperommet kan være svært ulikt genotyperommet, og at hele det evolusjonære søket foregår i genotyperommet. Etter endt evolusjon finnes løsningen



Figur 3.1: En representasjon må etableres i problemløsningsrommet til EAen.

(en god fenotype) ved å omforme den beste genotypen. Noen vanlige begreper innenfor evolusjonære beregninger er synonymer for de samme elementene i de to rommene, se tabell 3.2.

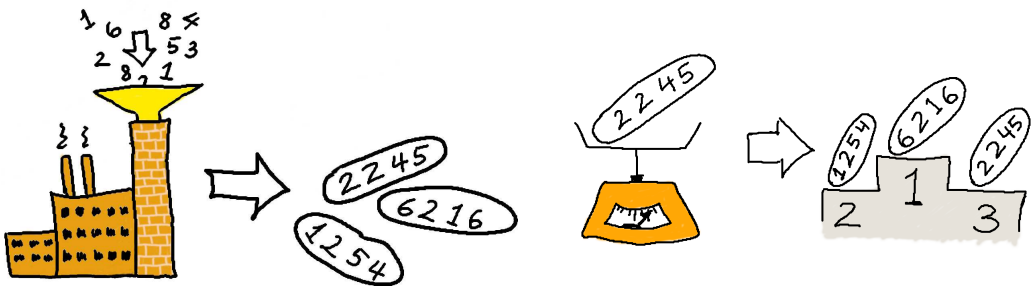
Genotyperom	Fenotyperom
genotype	fenotype
arvemateriale	løsningskandidat
individ	individ
genom	organisme
(regelsett)	

Tabell 3.2: Synonyme begreper. Regelsett er en tolkning av genomet som gjøres spesielt for dette systemets kombinasjon av evolusjon og utvikling. Regelsettet representerer individet fra genotyperommet, men tolkes og brukes i fenotyperommet av utviklingsprosessen.

3.2.2 Evaluering (fitnessfunksjon)

Rollen til fitnessfunksjonen under evalueringen er å kontrollere tilpasningskravene [9]. I fitnessfunksjonen defineres forbedring og hvilke deloppgaver som må løses for å løse problemet. Problemet en ønsker å løse avgjør hva individene skal være egnet til. I naturen ser målet ut til å være overlevelse og reproduksjon slik at arten overlever på lang sikt. I den evolusjonære algoritmen tildeles individene poeng (fitness) etter hvor godt de løser de definerte deloppgavene, illustrert i figur 3.2(b). For at problemet skal kunne løses må disse deloppgavene være definert og vektet riktig. Fitnessfunksjonen tilegner dermed et kvalitetsmål til genotypen. Kvalitetsmålet og deloppgavene er ofte basert på egenskaper ved

fenotypen. Denne tilbakemeldingen fremmer målrettet endring av genotypen og grunnlaget for videre seleksjon.



(a) (b)

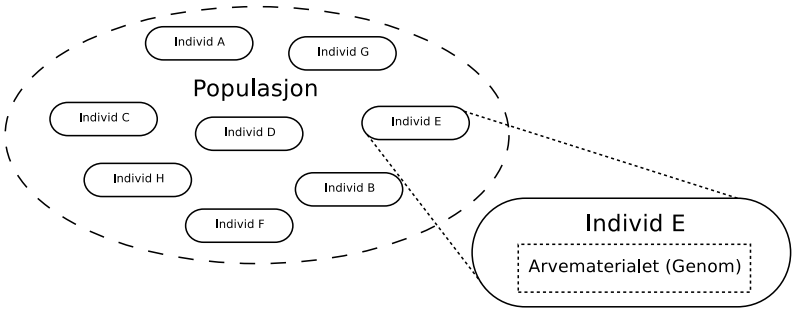
Figur 3.2: (a) En populasjon skapes. Hvert individ representeres av sitt genom (arvemateriale) laget av en tilfeldig sammensetning av gener, her med genverdier som heltall mellom 1 og 8. (b) Evaluering av individene. Hvert individ tildeles en fitnessverdi som kan benyttes for å rangere individene. En populasjon der alle individene er evaluert utgjør grunnlaget for evolusjonsprosessen.

3.2.3 Populasjon

Rollen til populasjonen er å holde på de mulige løsningskandidatene [9] illustrert i figur 3.3. Populasjonen er et sett av genotyper og utgjør arbeidsmaterialet for evolusjon. Under evolusjon endres og tilpasses populasjonen, ikke individene. Definisjonen på en populasjon kan være så enkel som å sette populasjonsstørrelsen, men den kan også uttrykke en mer avansert struktur mellom individene. I motsetning til variasjonsoperatorene som virker på en eller to foreldreindivider virker seleksjonsoperatorene på hele populasjonen. Seleksjonsoperatorene tar utgangspunkt i de løsningskandidatene som finnes i den nåværende populasjonen og kan for eksempel tillate alle bortsett fra det dårligste individet og gå videre til neste generasjon uten å bli erstattet. De fleste EAene har en konstant populasjonsstørrelse som ikke endres under det evolusjonære søket.

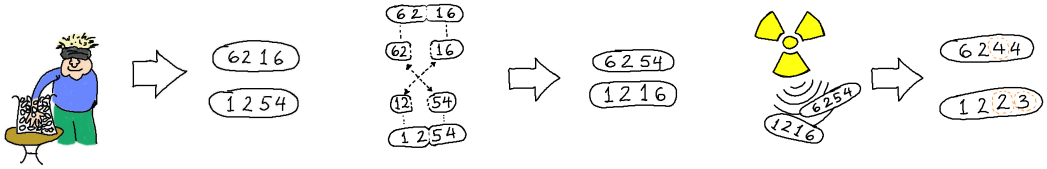
3.2.4 Reproduksjonsseleksjon

Rollen til reproduksjonsseleksjonen er å sortere ut de mest egnede individene som foreldre til neste generasjon, basert på fitness [9]. Utvalgte individer (foreldre) utsettes for variasjonsendringer for å skape avkom. Stimuleringen mot stadig bedre



Figur 3.3: En populasjon er en samling individer

fitness er resultat av reproduksjonsseleksjonen sammen med overlevelsesseleksjonen. Individer med god fitness har større sannsynlighet for å bli foreldre enn de med med dårlig fitness. Det følger av at reproduksjonsseleksjonen ofte er basert på sannsynlighet. Ofte har dårligere individer likevel en viss mulighet for å bli valgt ut for å unngå at søket setter seg fast i et lokalt optima. Et lokalt optima er en løsning som kun er best i en liten del av løsningsrommet, som for eksempel beste individ i nåværende generasjon. Et globalt optima derimot er den beste mulige løsningen totalt.



(a) Seleksjon *(b)* Krysning *(c)* Mutasjon

Figur 3.4: Seleksjon er en stokastisk utvelgelse der individer med genomet en ønsker å videreføre til neste generasjon har større sjanse for å bli valgt. Samtidig ønsker en også nye individer som er modifiserte versjoner av de selekterte. Håpet er at variasjonsoperatorene krysning og mutasjon sammen med seleksjon medfører en slik forbedring av tidligere generasjoner. Positive modifikasjoner stimuleres av tilbakemeldingene i form av fitnesspoeng fra evalueringen til seleksjonsoperatoren.

3.2.5 Variasjonsoperatorer

Rollen til variasjonsoperatorene er å skape nye individer fra gamle [9]. I fenotyperommet skapes tilsvarende nye løsningskandidater. I prøv-og-test analogien

står variasjonsoperatorene for utprøvingsdelen. Variasjonsoperatorene deles inn i to hovedgrupper basert på hvor mange foreldre de opererer på.

Mutasjon er et fellesbegrep for variasjonsoperatorer som tar en enkelt forelder og skaper et nytt avkom ved å tilføre en form for tilfeldig endring i genotypen. Hvordan denne formen for endring utføres avhenger av genotyprepresentasjonen (genomet), og eventuelle mutasjonsparametere. Antall gener i genomet som endres kan ofte varieres og disse endringspunktene kalles mutasjonspunkter. Hvilke endringer som er mulige og hvor store endringene skal være kan også varieres. Hvordan mutasjonen er implementert kan ha store konsekvenser for hvor effektiv den EAen er til å finne gode løsninger på problemet. I figur 3.4(c) vises mutasjon hos to avkom med ett og to mutasjonspunkt.

Rekombinasjon eller krysning er et fellesbegrep for variasjonsoperatorer som tar to foreldre og skaper et eller to nye avkom ved å kombinere deler av genotypen til foreldrene. En EA utfører vanligvis krysning med et tilfeldig krysningspunkt. Et eksempel på krysning med krysningspunktet plassert midt i genomet er vist i figur 3.4(b). Genene byttes på hver side av krysningpunktet slik at en får en ny kombinasjon av genomet og dermed et nytt individ. Ved krysning vil antallet avkom avhenge av om det er en EA med konstant antall individer i populasjonene og om foreldrene videreføres til neste generasjon (*steady-state EA*). Det er også vanlig å mutere avkommet ved å endre ett eller flere gener etter krysningen for igjen å få mer variasjon og kanskje mer egnede individer.

3.2.6 Overlevelsesseleksjon

Rollen til overlevelsesseleksjonen er å sortere ut individer basert på deres fitness [9]. Dette er tilsvarende reproduksjonsseleksjonen, men skjer først etter at avkommene er skapt fra de selekterte foreldrene. Da populasjonsstørrelsen ofte er konstant må det gjøres et valg for hvilke individer som skal overleve til neste generasjon. Denne seleksjonen er ofte basert på egenskaper ved individene som fitness eller alder. I motsetning til den sannsynlighetsbaserte reproduksjonsseleksjonen er overlevelsesseleksjonen ofte deterministisk. Det kan for eksempel være at en rangerer individene etter fitness og velger de beste, eller at det kun velges blant avkommene etter alder. Elitistisk overlevelsesseleksjon er eksempel på det første og sørger for at det til en hver tid beste individet ikke går tapt. Det blir alltid videreført eller klonet til neste generasjon hvis det ikke finnes et individ med tilsvarende eller bedre fitness.

3.2.7 Initialisering og terminering

De fleste EAene initialiseres enkelt ved å skape en populasjon med tilfeldige genotyper vist i figur 3.2(a). Det er også mulig å starte en populasjon med utgangspunkt i en tidligere genotype.

Terminering av det evolusjonære søket er ofte et stoppkriterium basert på et gitt antall tidsenheter i kombinasjon med en kjent optimal fitnessverdi. Ofte vil ikke søket nå den optimale fitnessverdien og uten andre stoppkriterier ville søket aldri stoppet. Det er derfor vanlig å benytte andre kriterier enten i tillegg til fitness eller i steden for. Det kan være stoppkriterier som antall generasjoner, tidsperiode og egenskaper ved populasjonen eller lignende.

3.3 Genetiske algoritmer

Genetiske algoritmer (GA) er blant de mer generelle typene av evolusjonære algoritmer og har en tilsvarende rutine som vist i algoritme 3.1. Enkle genetiske algoritmer har ofte en binær bit representasjon av genomet og krysning som den sentrale variasjonsoperatoren.

Det er vanlig å skille mellom to ulike GA modeller [9]: generativ modell og *steady-state* modellen. I den generative modellen begynner hver generasjon med en populasjon av størrelse μ. Av disse selekteres en gruppe på μ individ som foreldre. Ved bruk av variasjonsoperatorene på denne gruppen produseres et sett med $\lambda(= \mu)$ avkom. Disse blir evaluert og for hver generasjon erstattes hele populasjonen med sine avkom i neste generasjon. I *steady-state* modellen skiftes ikke hele populasjonen ut på en gang. Kun $\lambda(< \mu)$ gamle individer erstattes av λ nye individer, avkommene. Andelen av populasjonen som erstattes kalles *generative gap* og utgjør λ/μ.

Som nevnt for EAer foregår evolusjonen stegvis i generasjoner mot stoppkriteriet. Evolusjonen kan starte når populasjonens individer har fått tildelt hver sin poengsum, fitnessverdi, for hvor godt de egnet seg i henhold til kriteriene. Da er det genetiske grunnlaget med fitnessverdier på plass som utgangspunkt for å måle individene opp mot hverandre. Alle individene evalueres i hver generasjon og det plukkes ut minst et par foreldre som skal bidra til variasjon i neste generasjon av reproduksjonseleksjonen. Foreldrene kan kanskje gi opphav til nye kombinasjoner av individer som overgår de beste individene i nåværende generasjon.

I denne oppgaven brukes en genetisk algoritme hvor genotypen er representert med heltall og de genetiske operatorene er mutasjon og kloning. Det er da kun mutasjonsoperatoren som bidrar med ny variasjon i populasjonen. Evolusjonsprosessen vil da ligne mer på den som foregår hos enklere organismer i naturen hvor reproduksjonen ikke er kjønnet, men aseksuell (kloning) og mutasjonsstyrt. Individene som dyrkes i oppgaven benytter altså tilsvarende evolusjon som en av jordens mest utbredte organismer, bakterien.

Kapittel 4

Utvikling

Utvikling er hovedsaklig tilsynekomsten av organiserte strukturer fra en i utgangspunktet svært enkel gruppe av celler. -Wolpert 1998

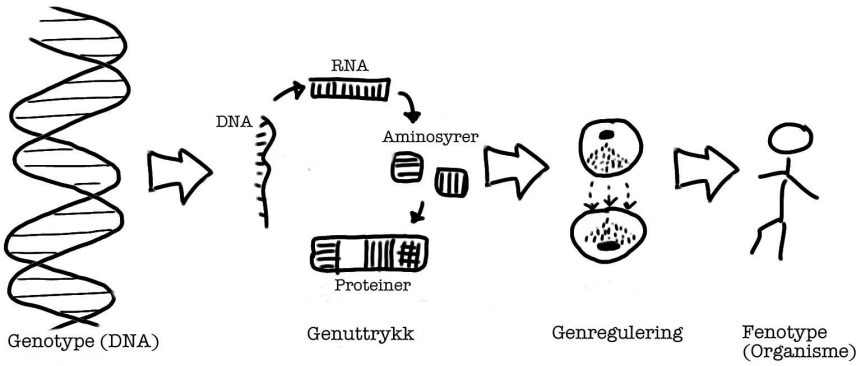
I naturen finnes et stor mangfold av systemer hvor enkle elementer med lokale vekselvirkninger gir opphav til koordinerte globale informasjonsbehandlere [17]. Noen eksempler på tilsynekomsten av slike behandlingssystemer er insektskolonier, cellestrukturer og immunsystemet. Denne *framtredende informasjonsbehandlingen* kalles **emergent computation**. Begrepet *emergent computation* refererer til tilsynekomsten av globale egenskaper for informasjonsbehandling som ikke er eksplisitt representert i systemets elementære komponenter eller komponentenes sammenkoblinger [5].

Utviklingsprosessen i naturen er resultat av samspillet mellom gener, proteiner, celler og omgivelser. Sluttproduktet fra denne prosessen er en ferdig formet organisme [13]. Den menneskelige utviklingen er prosessene en befruktet eggcelle går gjennom til den har resultert i et voksent individ. Litt forenklet kan en si at livet designes av evolusjonen og bygges av utviklingen.

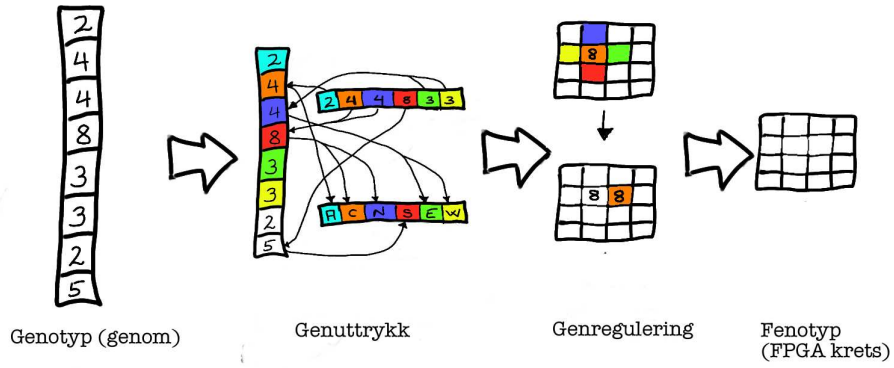
Kunstig utvikling omhandler vanligvis både tolkningen av genotypen til byggeinstruksjoner (genuttrykk) og utførelsen av disse (genregulering) fram til et ferdig utviklet individ, fenotypen [12].

Dette innebærer at genotypen ikke lenger må ha en en-til-en relasjon mot fenotypen. Utviklingen ser ut til å kunne redusere skaleringsproblemet som oppstår når fenotypens kompleksitet og størrelse skaleres opp. En illustrasjon på den digitale utviklingens imitasjon av naturens egen modell for utvikling er vist i figur

4.1.



(a) Biologisk utvikling



(b) Kunstig utvikling

Figur 4.1: Modellene for utvikling. Inspirasjonsmodellen vises i 4.1(a) og imitasjonen i 4.1(b)

4.1 Genuttrykk - tolkning av genotypen

Utgangspunktet for individets utvikling ligger i resultatet av evolusjonsprosessen, genotypen. Informasjonen kodet inn i denne representasjonen av individet har ingen verdi i seg selv hvis det ikke finnes en prosess som kan tolke innholdet og realisere individets fenotype.

Utvikling innebærer en tolkning av arvematerialet der genomet ikke representerer byggesteinene, men isteden representerer ulike byggere. Det forutsetter at det finnes forhåndsdefinerte omgivelser der byggerne vil kunne utføre en jobb som

avhenger av hvordan byggesteinene(omgivelsene) er. Denne fortolkningen skiller seg fra tidligere evolusjonære systemer uten utvikling hvor genomet ble direkte omformet til ulike byggesteiner i en en-til-en tolkning mellom genotypens genom og fenotypens byggesteiner. Når kompleksiteten til fenotypen en ønsket å utvikle økte, økte også representasjonen i genotypen tilsvarende. Dette ga etterhvert et svært stort søkerom for de evolusjonære beregningene. Det ble derfor forsket mye på hvordan kompleksiteten i fenotypen kunne økes uten at genotypen ble tilsvarende større. Igjen skulle svaret vise seg å komme fra naturen og dens robuste løsninger. Genuttrykket i våre egne celler oversetter genomet til proteiner. En kunstig imitasjon av denne utviklingsprosessen kan se ut til å være løsningen også innen evolusjonære systemer [10, 11].

4.1.1 Skalering

En av de store utfordringene innen evolusjonær maskinvare er skaleringsproblemet med genomet som følger av ønsket om å designe store komplekse kretser [10]. Hvordan kan man la evolusjonen komme fram til en løsning (fenotype) av en viss kompleksitet og størrelse uten at genomets representasjon (genotypen) vokser tilsvarende?

Hvis en skulle brukt en en-til-en omforming fra genotypen til fenotypen så ville dette raskt gi et uoverkommelig stort søkerom. Da ville hver mulige fenotype i løsningsrommet ha en tilsvarende unik genotype i søkerommet. Hvis genotypen eksempelvis skulle kunne holde alle parametrene som kreves for å bygge en båt, ville det kreve svært store mengder data per individs genom. En evolusjonær algoritme vil dermed ikke kunne finne en god løsning innen rimelig tid. I fra naturen vet en at det menneskelige DNA er langt fra å kunne uttrykke alle byggesteinene og utviklingstrinnene som skal til for å utvikle et menneske fra en befruktet eggcelle til et fullvoksent individ [23].

I alle naturens multicellulære organismer ser en tydelig hvordan genomet kan gi opphav til organismer som er langt større og mer komplekse enn hva genomet kan uttrykke med en en-til-en omforming. En kan her observere identiske byggeblokker og strukturer som settes sammen til å utgjøre en større helhet. Genomet representerer altså en oppskrift på hvordan en organisme utvikler seg istedet for å uttrykke alle de ulike egenskapene direkte. Dermed minimeres informasjonsmengden som skal til for å representere individet. Utviklingen utgjøres av fire prosesser: mønsterdannelse, celledifferensiering, morfologi og vekst. [2].

4.1.2 Nøytrale nett

En av de store utfordringene ved evolusjonære beregninger er at algoritmene konvergerer mot ikke-optimale løsninger for tidlig [3]. For å unngå å bli sittende fast i lokal optima vil variasjonsoperatorer som mutasjon og krysning være nødvendige, men hvor godt denne variasjonen utnyttes avhenger av hvordan seleksjonsoperatoren er implementert. Siden man i denne oppgaven ønsker en evolusjon som er mest mulig deterministisk, er det ikke ønskelig å velge individer med lavere egnethet. Det blir derfor svært få av de muterte individene som søket kan flytte seg til å lage nye mutasjoner fra. Det betyr at forflytningene i søkerommet kan ta for lang tid eller stoppe opp, så lenge ikke bedre løsninger genereres fra variasjonsoperatorene. Problemet kan kanskje minskes ved bruk av nøytrale nett [7].

Nøytralitet i søkerommet er også en metode for å unngå at søket blir sittende fast i lokale optima. Nøytrale nett i søkerommet gjør at evolusjonsalgoritmen beveger seg gjennom søkerommet av løsningskandidater uten å måtte gå til dårligere løsninger for å få komme ut av et lokalt optima. Istedet drifter evolusjonen mellom likeverdige eller bedre løsninger. Dette forutsetter en genotyp-fenotyp omforming som har mye overlapping.

Teorien om nøytral evolusjon antyder at de fleste mutasjoner ikke gir endringer i fenotypen. Det vil si at omformingen fra genotyp til fenotyp inneholder redundans slik at mange mutasjoner ikke vil gi noen merkbar effekt på fenotypen. Dette kan resultere i sett av genotyper som er koblet sammen av enkeltpunktmutasjoner som alle omformes til den samme fenotypen og går under betegnelsen nøytrale nett. En populasjon kan vandre langs disse nettverkene, eventuelt støte på fenotyper med høyere egnethet, hvorpå sjansen for å bli fanget i en lokalt optimalt område av genotyprommet minskes.

4.2 Genregulering - konstruksjon av fenotypen

Hos pattedyrene starter utvikling med det befrukta egget, zygoten, som begynner å dele seg raskt. Allerede tidlig skjer det en inndeling av cellesamlingen i to poler. Den ene delen kommer til å utgjøre det utvendige vevet av kroppen og den andre det indre av kroppen [15].

Hvilke underliggende mekanismer gir en utvikling som gjør det mulig for en enkelt celle å gi opphav til en multicellulær organisme? Innen biologisk utvikling er det

fem hovedprosesser: celledeling, mønsterdannelse, morfogenese, celledifferensiering og vekst [13].

Under celledelingen deles det befrukta egget, zygoten, uten at cellene vokser. Det resulterer i en celleklump kalt blastocyst. Under mønsterdannelse organiserer de ulike celleaktivitetene seg i midlertidige plasseringer innad i organismen. Det sikrer at videre utvikling skjer med en viss strukturell inndeling. Morfogenesen er endringene i organismens form som skjer gjennom cellebevegelse og interne endringer i cellene. Celledifferensiering gjør at cellene får ulik struktur og funksjonalitet i forhold til hverandre og bestemmes av intracellulære signaler mellom cellene. Vekst kan skyldes celler som formerer seg, øker i størrelse og opptak av ekstracellulær materialer.

Denne utviklingen på FPGAen foregår med Sblock celler. I stedet for proteinsignaler under genregulering med ekstracellulære signaler, får Sblockcellene en tilsvarende oppførsel med et fireveis von Neumann-naboskap og seg selv. Hvis nye celletyper dukker opp i naboskapet til ei celle kan dette trigge andre regler, og genuttrykket for cella endres dermed av omkringliggende celler slik det skjer i naturens organismer. Den cellulære automatavirkemåten til FPGAen gjør denne spesielt egnet til slike massive interaksjoner.

4.2.1 Cellulær automata

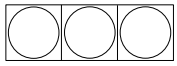
Cellulære automata (CA) ble introdusert av Ulam og von Neumann i 1940 [17]. En CA er et dynamisk system der tid og rom er diskrete verdier. CAen består av en matrise med celler. Hver celle kan være i en av et endelig antall mulige tilstander. En identisk regel finnes lokalt i alle cellene og oppdaterer tilstanden for hvert diskrete tidssteg. Tilstanden til cella i neste steg er gitt av dens nåværende naboskap. CAer ble introdusert som et formelt rammeverk for å utforske komplekse systemer.

4.2.2 Regelbasert naboskap

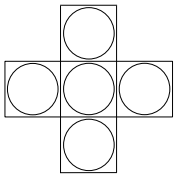
Den cellulære matrisen er n-dimensjonal [17]. Fokus i oppgaven vil være en todimensjonal matrise siden systemet er basert på en slik løsning. Cellene med hver sine identiske regler tilsvarer i prinsippet en endelig tilstandsmaskin. Reglene utgjør en regeltabell med et innslag for hver mulige nye tilstand og dens betingelser til naboskapets tilstand for å aktiveres. Denne regeltabellen kan sammenlignes

med overgangsfunksjonen [eng:transient function] i en tilstandsmaskin. Det cullulære naboskapet til en celle er de omkringliggende cellene i tillegg til cellen selv. Naboskapet vil være ulikt avhengig av antall dimensjoner matrisa er definert i, men det er også ulike representasjoner for to-dimensjoner som vist i figur 4.2.

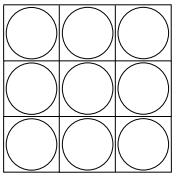
I en matrise med endelig utstrekning brukes ofte sømløse kanter som gir matrisen periodisk romlig (her:arealmessig) utstrekning. Det vil si at kantradene og kantkolonnene i den cellulære matrisen kobles sømløst til motstående kantrad og kantkolonne. Resultatet er at en-dimensjonale cellulære matriser har form som en sirkulære vektor og to-dimensjonale cellulære matriser har form som en toroid.



(a) 1D matrise inneholder 3 naboceller, cella selv og to naboer.



(b) 2D matrise inneholder 5 naboceller, cella selv og 4 naboer.



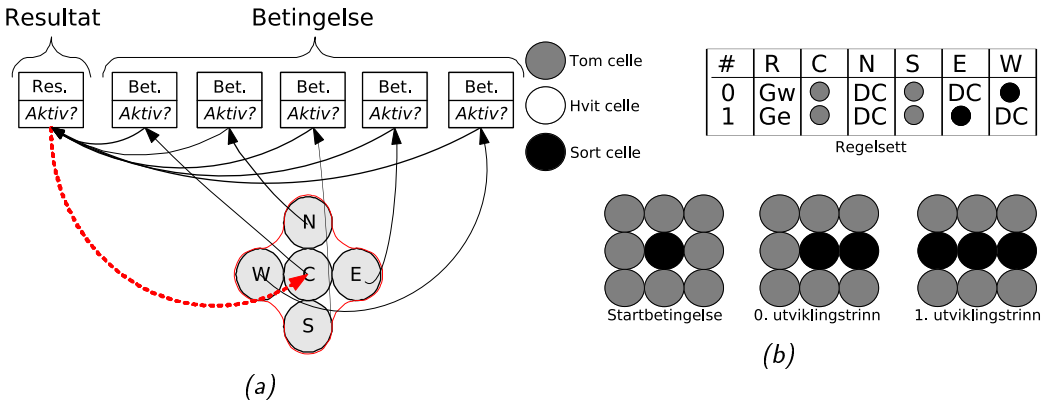
(c) 2D matrise med diagonale naboer inneholder 9 naboceller, cella selv og 8 naboer.

Figur 4.2: Cellulære naboskapsmodeller for en- og to-dimensjonale matriser.

En to-dimensjonal CA med naboskap med 5 celler der hver celle kan ha tilstandene tom, sort og hvit kan for eksempel ha regelen *grow from west* hvis du selv er tom, nabocellen sør for deg er tom, nabocellen vest for deg er sort og nabocellene nord og øst kan være hva som helst, vist i regelsettet i figur 4.3(b). I tillegg inneholder regelsettet en *grow from east*, men regel 0 har høyest prioritet. Et slik regelsett vil medføre tilstandsendringene vist i 0. og 1. utviklingstrinn i samme figuren. Figur 4.3(a) viser sammenhengen mellom hver enkelt celle og reglene i regelsettet. Samtlige betingelser må matche og aktiveres for at resultatet skal kunne slå til.

4.2.3 Turingkomplett

CAer maskiner har tilsvarende universell beregningskapasitet som en universell Turingmaskin [4]. Denne beregningsegenskapen koblet med de dynamiske egenskapene skiller allikevel CAen fra en standard Turingmaskin. CA modellen er både generell og enkel [17]. Universelle beregninger er avhengig av denne generaliteten. De enkle basiskomponentene (cellene) har en generell form for lokale



Figur 4.3: Eksempel på regelaktivitet og regelprioritet. I (b) er det kun regel 0 som blir brukt i begge utviklingstrinnene, da den har høyere prioritet enn regel 1.

vekselvirkninger og ikke spesialiserte handlinger seg imellom. Denne enkelheten gjør at utvekslingen av informasjon mellom basiskomponentene er moderate i forhold til en Turingmaskin. CA modellen er blant de enkleste og minst komplekse generelle modellen av tilstandsmaskiner. CAer som parallelle cellulære maskiner har tre viktige egenskaper: massiv parallellitet, lokale cellulære vekselvirkninger og enkle basiskomponenter (cellene) [17].

4.3 Cellulær beregningsmaskin for utvikling

Det er mulig å realisere en parallell cellulær beregningsmaskin i dagens maskinvareteknologi [20]. Den cellulære beregningsmaskinen utnytter prinsippet om *emergence* av en global oppførsel fra lokale cellers vekselvirkning, og muliggjør en massivt parallell arkitektur. Den massive parallelliteten er vanskelig å utnytte med klassiske design- og programmeringsmetoder. Resultatet av å gå over fra klassiske metoder til adaptive metoder [18] er at designeren ikke lenger kan spesifisere alle detaljene ved systemet [17]. EA kan være en adaptiv løsning, men kan også introdusere nye begrensinger i designprosessen. EAer er ofte ressurskrevende og med en direkte omforming mellom genotype til fenotype medfører de ofte vanskeligheter ved skalering av problemet [6].

En løsning for å unngå ressurskrevende bruk av EAen er å følge naturens eksempel og redusere søkerommet for genotypen. I naturen gjøres dette ved å la omformingen fra genotype til fenotype gå via en utviklingsprosess. En kunstig utviklingsmetode tilsvarende zygoten som utvikler seg til en multicellulær organ-

isme i naturen [23] kan benyttes i EAer. Dette kan øke skalerbarheten av problemet [1] og muliggjøre bruk av cellulære beregningsmaskiner i kombinasjon med evolusjonære metoder for mer komplekse fenotypeproblemer. I denne cellulære beregningsmaskinen kombineres utvikling og evolusjon i en kompleks prosess inspirert fra naturens systemer. I et slikt system er det stort behov for å kunne eksperimentere med parametere og komponenter for å danne seg en eksperimentell forståelse av systemet for videre utbedring og funksjonalitetssøken.

Kapittel 5

Plattform for eksperiment

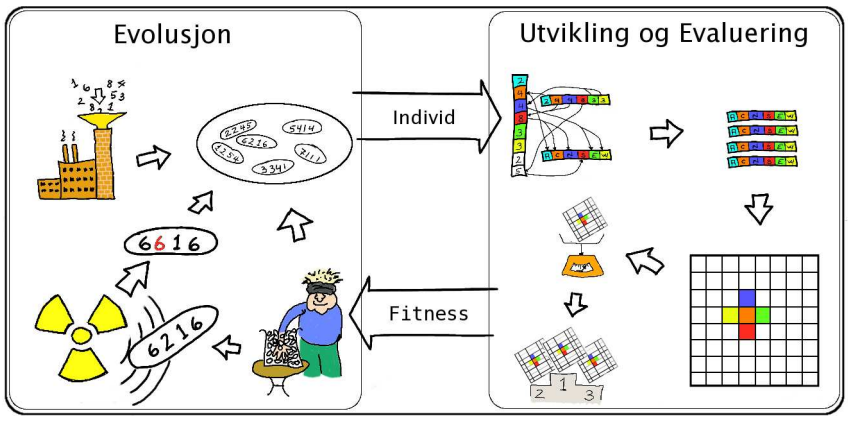
I dette kapitelet presenteres plattformens oppbygning og hvordan teorien ble brukt i implementeringen av programmet. Plattformen ble implementert for å muliggjøre eksperimentell utforskning av oppsett og data gjennom evolusjon og utvikling av cellestrukturer. Den er basert på den cellulære beregningsmaskinen presentert i kapittel 4.3 og simulerer den strukturelle delen av utviklingssystemet. Det er mulig å gjøre eksperimenter med tre ulike arter og to ulike strukturer uten å endre programkoden. For disse artene og strukturene, vist i tabell 5.1, gir plattformen mulighet til å utforske ulike parameterverdier for evolusjons- og utviklingsprosessen. Resultatet av hvert eksperiment logges til en loggfil som danner grunnlaget for analyse og videre eksperimenter. Loggfilen inneholder blant annet det strukturelle resultatet og data fra prosessene.

Arter (Individgrupper)	Simple	Neutral (sjakk)	Flag
Struktur	sjakkbrett	sjakkbrett	flagg
Omforming (genotyp-fenotyp)	ikke-nøytral (1-*)	nøytral (*-*)	nøytral (*-*)
Evolusjonsmetode	randmut-3	randmut-3	randmut-3
Evolusjonens stoppkriterie	antall generasjoner	antall generasjoner	antall generasjoner
Utviklingens stoppkriterie	antall utviklingstrinn	antall utviklingstrinn	antall utviklingstrinn
Mulige celletyper	Z, B, W, Bl, R, Gr	Z, B, W, Bl, R, Gr	Z, B, W, Bl, R, Gr
Oppstartsstruktur	enkel celle	enkel celle	enkel celle
Celletype ved oppstart (posisjon)	black (0,1)	black (0,1)	green (0,1)

Tabell 5.1: Artene og strukturvariantene implementert i plattformen. Tabellen viser egenskapene ved disse som er forhåndsdefinerte. (satt i programkoden).

5.1 Plattformens oppbygning

Plattformen består i hovedsak av de tre prosessene evolusjon, utvikling og evaluering illustrert i figur 5.1. Evolusjonsprosessen starter med å produsere en populasjon av individer med tilfeldige heltallsgenom. Individene blir så sendt over til utviklingsprosessen. Der tolkes hvert genom til et regelsett. Regelsettet gjelder for alle cellene i den cellulære matrisen og avgjør hvordan cellene utvikler seg i et gitt antall utviklingstrinn. Den ferdige organismens struktur evalueres og individet tildeles et fitnesspoeng etter hvor godt strukturen oppfyller delkriteriene i fitnessfunksjonen. Det evaluerte individet sendes tilbake til evolusjonen hvor en seleksjon basert på fitnessverdien velger ut et individ som klones til neste genereasjon og er opphav til resten av neste generasajons muterte avkom. Klonet og de muterte avkommene sendes så over for utvikling og evaluering og slik fortsetter det i et gitt antall generasjoner. Løsningen er individet i siste generasjon med best fitnessverdi. Disse prosessene er beskrevet nærmere i 5.1.1, 5.1.2, 5.1.3 og 5.1.4.



Figur 5.1: Prosessene på plattformen.

5.1.1 Evolusjon

Evolusjonsprosessen benytter den genetiske algoritmen vist i algoritme 5.1 med elitistisk overlevelsesseleksjon. Antall individer i populasjonen og genomets størrelse er gitt i oppsettet. Evolusjonsprosessen initieres ved å skape et genom til hvert individ, *skap populasjon*. Hvert gen tildeles et tilfeldig heltall mellom 0 og N, genverdien. N er $genomstørrelsen - 1$, siden genverdiene starter på 0.

```
skap populasjon
while < MAKS-ANTALL-GENERASJONER do
  evaluer individene
  selekter individ
  klon individ
  muter avkom
  innsett avkom og klon i nesteGenerasjon
end while
```

Algoritme 5.1: Genetisk algoritme

Evolusjonen utføres så på populasjonen til antall evolverte generasjoner har nådd maks antall generasjoner spesifisert i oppsettet. For hver generasjon evalueres alle individene ved å la hvert individ utvikle sitt genom. Genotypen utvikles da til en ferdigutviklet organisme, fenotypen. Hvert individ har et gitt antall utviklingstrinn som settes i oppsettet. Når individet er ferdigutviklet kan fenotypen evalueres med fitnessfunksjonen. Hvert individ rangeres av denne forhåndsdefinerte fitnessfunksjonen. Det er her evolusjonsprosessen får tilbakemelding på hvor god fitness individet hadde. Den resulterende fitnessen vil være avgjørende i neste steg i algoritmen, seleksjonen.

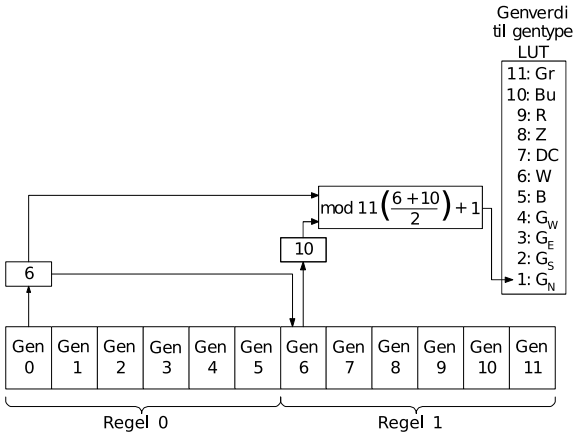
Under seleksjonen er det individet med best fitness som blir videreført til neste generasjon. Andre individer blir kun selektert hvis de har bedre fitness eller jevngod fitness med det hittil beste individet. Ved å tillate selektering av andre individ med jevngod fitness kan evolusjonen vandre mellom alternative løsninger i henhold til teorien om nøytrale nett beskrevet i avsnitt 4.1.2.

Algoritmen bruker kun kloning og mutasjon som genetiske operatorer. Mutasjonsoperatoren (randmut-3) kan endre en, to eller tre mutasjonspunkter i genomet. Disse mutasjonspunktene velges tilfeldig. Genet i et mutasjonspunkt tilordnes en ny tilfeldig verdi som også kan tilsvare opprinnelig verdi, altså ingen endring. Kloningen av et individ resulterer alltid i en identisk kopi av individets genom, og brukes her for å lage kloner av det beste individet.

Populasjonsstørrelsen er konstant. Det beste individet som ble plukket ut under seleksjonen gir opphav til neste generasjon gjennom sitt klon. De resterende individene blir fjernet og erstattes i neste generasjon av det beste individets avkom. Disse avkommene er mutasjoner av det beste invidiet. Klonen og avkommene utgjør neste generasjon.

5.1.2 Utvikling

For at evolusjonsprosessen skal kunne evaluere individene, må altså hver genotype utvikles til sin fenotype. Dette gjøres ved først å tolke genomet til regelsett som vist i figur 5.2. Mulige gentyper i reglene er vist i tabell 5.3. Genotypen som nå er representert ved sitt regelsett bestemmer adferden til hver enkelt celle i den cellulære matrisen. Den to-dimensjonale cellulære matrisa inneholder et gitt antall tomme celler hvor hver enkelt celle selv sjekker om naboskapet matcher betingelsesdelen i en av regelen sine for hvert utviklingstrinn. I den cellulære matrisen er det kun celletypene, vist i tabell 5.2 av gentypene som kan forekomme. Siden det er forbudt å *grow* tomme celler inn i tomme celler vil en cellulær matrise med kun tomme celler aldri utvikles uansett hvor mange utviklingstrinn som tillates. Før utviklingen startes settes derfor en enkelt celle til en gitt celletype, for eksempel grønn. Plasseringen av den initielle cellen er irrelevant da den cellulære matrisen har sømløse kanter. I første utviklingstrinn vil alle cellene gå gjennom de identiske reglene og sjekke om de har et naboskap som aktiviserer alle betingelsene i en regel. Det kan finnes flere enn en regel som matcher naboskapet, reglene sjekkes derfor alltid i prioritert rekkefølge. Ved første match i regelsettet i et utviklingstrinn vil kun resultatet av denne regelen utføres, selv om det finnes flere potensielt matchende celler i regelsettet.



Figur 5.2: Tolkning av genom til regelsett. Snittverdien fra genverdiene av hvert gen og tilhørende adresserte gen tolkes sekvensielt til en gentype i regelsettet. De mulige gentypene er definert i oppslagstabellen (LUT, eng: look-up-table) og omgjøringen gjøres med modulo operatoren. Tolkningen er irreversibel. Eksempelet viser tolkning av første gentype i første regel for et individ med 2 regler.

I motsetning til naturens celledeling der en enkelt celle deler seg og blir til to er det her de tomme nabocellen som trekker startcellen utover med *growth* regler som

vist i 4.3(b) i avsnitt 4.2.2. I tillegg til *growth*reglene er det *change* regler som endrer celletypen på ikke tomme celler hvis naboskapet matcher en *change*regel. Denne deterministiske prosessen fortsetter til siste utviklingstrinn. Uansett om det fortsatt finnes regelaktivitet eller ikke så er organismen ferdig utviklet umiddelbart i siste utviklingstrinn. Det er dette siste utviklingstrinnet som blir evaluert av fenotypen.

I motsetning til naturlig utvikling vil to identiske individ (genom) utvikles likt forutsatt at omgivelsene er identiske. Mulig i dette systemet da omgivelsene kan holdes statiske.

Type	Navn	Grafisk
Gr	Green	
Bu	Blue	
R	Red	
B	Black	
W	White	
Z	Empty	

Tabell 5.2: Mulige celletyper i den cellulære matrisen

Type	Navn	Grafisk
Gr	Green	
Bu	Blue	
R	Red	
B	Black	
W	White	
Z	Empty	
G_N	Grow North	G_N
G_S	Grow South	G_S
G_E	Grow East	G_E
G_W	Grow West	G_W
DC	Dont Care	DC

Tabell 5.3: Mulige gentyper i reglene

Hver regel merkes som gyldig eller ugyldig. Utvikling utføres kun for individer med minst en gyldig regel i sitt regelsett.

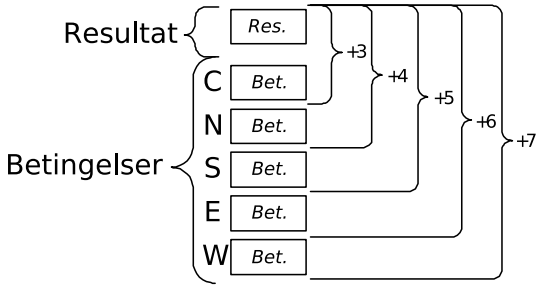
Under utvikling sjekkes hver celle og dens naboskap mot individets regelsett for å se om det er en match mellom regelens betingelser (naboskapet) og naboskapet rundt den aktuelle cella. Kun reglene som da er merket som gyldige blir vurdert.

Regler som matcher, men som endrer celletypen til det samme som den var er også gyldige regler. Disse ønskes rapportert selv om det ikke skjer noen reell endring i fenotypen da de kan tenkes å ha en stabiliserende effekt på sluttindividet, da de står og trigger og bevarer mønsteret. De kan også tenkes å være med å hindre andre regler å trigge og hindre utbredelsen av et mønster. Men tom-i-tom *growth*regler blir ikke rapportert.

Etter at ikke reelle match er filtrert vekk matches samtlige celler med hvert sitt naboskap mot individets regelsett for å se om en av reglene trigges ved oppfylte naboskapbetingelser. Den første regelen i det prioriterte regelsettet som eventuelt trigger blir utført og gir en ny celletype til den aktuelle cellen. Dette skjer enten ved *growth*, der typen til en nabocelle i fenotypen vokser inn i den aktuelle cellen, eller ved *change*regler, der den aktuelle cellens type endres til celletypen spesifisert i regelen.

5.1.3 Evaluering for sjakkbrettindivid (neutral)

Tilbakemeldingen fra fitnessfunksjonen til EAen er avhengig av at genomene inneholder minst en lovlig regel. Før de har nådd dette fitnessnivået har individene ikke anledning til å utvikle seg. Evalueringen av det siste utviklingstrinnet i utviklingsprosessen er altså avhengig både av prosesseringen av genomet (reglene) i cellene og den *emerging* (tilsynekommende) fenotypen. For å kunne forsyne utviklingsprosessen med genom som inneholder gyldige regler benyttes en regelfitness. Regelfitnessfunksjonen gir poeng for gyldige gentyper i reglene. Så lenge individene ikke inneholder noen gyldige regler er det kun regelfitnessen som blir beregnet. Den tyngre utviklingsprosessen og etterfølgende evalueringen av fenotypen gjøres kun på kvalifiserte individ. Dette sparer systemet for unødvendige beregninger i starten når evolusjonsprosessen leter fram individer med gyldige regler.



Figur 5.3: Fitnesspoeng basert på riktig sammensetning av gentyper i regelen.

Regelfitnessfunksjonen er basert på tre egenskaper ved hver regel i genomet. Først tildeles poeng for hver gentype uavhengig av regelens sammensetning. Det forutsetter at gentypen er gyldig, altså at det ikke er en DC i *resultat* og ingen veksttypeinnslag i *betingelse* delen. For det andre tildeles en trinnvis økende poengsum basert på hvor korrekt regelen er komponert, altså om den følger vekst- og endringsreglenes restriksjoner. Den tredje egenskapen det tildeles poeng for er gyldige regler.

Poengene for de tre egenskapene er vektet gjennom prøving og feiling for å gi en god balanse. Hver riktige gentype gir 2 poeng. Utregningen for korrekt komponerte regler er vist i figur 5.3. Hvis *resultat*-delen og *center*-betingelsen er korrekt for en endrings- eller vekstregel gis 3 poeng. Hvis *resultat*, *center* og *north* er korrekt gis det 4 poeng til osv. Totalt gis 25 poeng hvis alle gentypinnslagene er korrekte. Hver regel som følger alle restriksjonene gis 20 poeng. Det er verdt å merke seg at fitnessverdien for regler avhenger av genomstørrelsen. Det må taes hensyn til under vektingen mot fenotypfitness.

Genom med gyldige regler utvikles og tildeles en fitnessverdi basert på antall korrekte celletyper i den ferdigutviklede fenotypen, altså siste utviklingstrinn. Hver korrekte celletype, enten svart eller hvit, tildeles 50 poeng.

```
for CELLE(0,0)→CELLE(N,N) I FERDIG-UTVIKLET-FENOTYPE
do
  if celletype = (like & hvit) then
    fitness = fitness + 50
  else
    if celletype = (odde & svart) then
      fitness = fitness + 50
    end if
  end if
end for
```

Algoritme 5.2: Pseudokoden for sjakkbrettindividenes fitnessfunksjon for evaluering av den ferdigutviklede fenotypen.

Denne algoritmen vil stimulere til utbredelse av celletypene hvit og svart mot et brett hvor innføring av svarte celletyper kun gir uttelling hvis de plasseres i odde celler og hvite kun i like celler. Dette gjør at det kun finnes en perfekt løsning for evolusjonsprosessen. I virkeligheten finnes det to løsninger på sjakkbrettmønsteret. Ved å velge kun den ene forenkles fitnessfunksjonen. Da plattformen likevel fant perfekte resultater i mer enn halvparten av forsøkene ble denne fitnessfunksjonen beholdt.

5.1.4 Evaluering for flaggindivid

Målet var å evolvere fram et genom som ga et *emerging* flaggmønster etter endt utvikling. Fenotypens strukturelle egenskaper ved endt utvikling var hovedkriteriet for evalueringen. I tillegg ble en regelfitness tilsvarende den for sjakkbretteval-ueringene i avsnitt 5.1.3.

For å stimulere evolusjonen til å gi et slikt mønster måtte evalueringsfunksjonen jobbes fram eksperimentelt. Da sjakkbrettløsningen hadde et svært ulikt mønster, kunne ikke en tilsvarende fitnessfunksjon benyttes for et flaggmønster. Kun fitness for proliferasjon (cellevekst), antall celler med ønsket celletype, ble beholdt. Sjakkbrettmønsteret hadde kun to mulige løsningskandidater, der kun en ble utnyttet i fitnessfunksjonen. Flagget har langt flere både løsningsmuligheter og fitnessdeler.

Fitnessfunksjonen består av følgende deler:

- Regelfitness (genotypefitness)
- Proliferasjonsfitness (utbredelse)
- Tallforholdsfitness (celle- og typeforhold)
- Identisk-sum (flere like typer i samme rad)
- Koblingsfitness (kobling av rader med lik typedominans)

For å la evolusjonen jobbe mest mulig fritt, stor frihet og mange løsningskandidater, ble det fokusert på å finne gode fitnessmål som ikke begrenset mer enn nødvendig. Et trikolor flagg vil som et generelt tilfelle kunne defineres ved riktig forhold mellom antallet av tre ulike celletyper. Med dette som utgangspunkt ble første forsøk på fitnessfunksjon definert til å stimulere til et gitt tallforhold mellom i første omgang to celletyper, vist i algoritme 5.3.

Tallforholdsfitnessen for 3 celletyper ble konstruert med utgangspunkt i resultatene fra tallforholdseksperimentene med 2 celletyper. Algoritmen for tallforholdfitness med 3 celletyper er vist i 5.4. Det var opp til evolusjonen selv å velge hvilken celletype som skulle utgjøre den størst andelen både for 2 og 3 celletyper.

Dette ga gode forhold mellom celletypene, men lignet ikke på et flaggmønster. Tallforholdsfitnessen ble derfor kombinert med identisk-sumfitness vist i algoritme 5.5 som ga poeng avhengig av størst antall like celler i en rad og koblingsfitness vist i algoritme 5.6 som ga poeng for kobling mellom rader av lik celletypedominans.

Under eksperimenteringen og utviklingen av fitnessfunksjonene viste det seg at mønstrene ofte hadde en kaotisk men forholdsmessig riktig inndeling av de 3 celletypene for hver rad. Tallforholdfitnessens vekting ser ut til å ha tvunget hver endring mot andre fitnessvektinger til å også holde på det riktige tallforholdet underveis. Det var samtidig en nødvendighet med en viss avhengighet i poengberegningen fra koblingsfitnessen og identisk-sum til tallforholdsfitnessen. Fravær av denne avhengigheten ga langt dårligere løsninger. Det kaotiske mønster gjorde det antagelig enklere å beholde et godt tallforhold fra individ til individ under evolusjon samtidig som celleutbredelsen økte.

Utbredelsefitnessen kan ha gjort det vanskelig for evolusjon å endre mønster da poengverdien per ikke tomme celle en periode var høyere enn tilsvarende endring i tallforholdet og de to andre kriteriene. Det hjalp å senke verdien per enkelt celle til mindre enn poenget fra andre trinnvise fitnesspoeng

Et typisk trekk under balanseringen av de ulike fitnessfunksjonene var at vekt-

ing av identisk-sum og koblingsfitness ga fenotyper som oppfylte disse kriteriene, men med dårlig tallforhold. Ofte kunne det da være kun en celletype på brettet. Det store problemet var å balansere identisk-sum og koblingsfitness mot tallforholsfitnessen.

```
R_T = 3/7
for CELLE(0,0)→CELLE(N,N) I FERDIG-UTVIKLET-FENOTYPE
do
  if blacks > whites then
    R = whites/blacks
    fitness = fitness + (1 − |R_T − R|) ∗ (blacks + whites) ∗ 25)
  end if
  if whites >= blacks then
    R = blacks/whites
    fitness = fitness + (1 − |R_T − R|) ∗ (blacks + whites) ∗ 25)
  end if
end for
```

Algoritme 5.3: Pseudokoden for flaggindividenes fitnessfunksjon for tallforhold med 2 celletyper, black og white. Kun tallforholdet mellom typene.

Da det endelige flagget skulle bestå av tre celletyper ble tallforholdfitnessfunksjonen videreutviklet, som vist i algoritme 5.4.

Ingen av disse algoritmene er forsøkt optimalisert utover første suksess med å nå målet. Dette er de fitnessfunksjonene som gjennom eksperimentell prøving og feiling har vist seg å fungere.

Evolusjonen har vist seg gang på gang å kunne finne og utnytte de enkleste løsningene for å oppnå sitt mål (høyest mulig fitness ved å føye den implementerte fitnessfunksjonen enklest og best mulig). Kombinasjonen av utbredelsesfitness og tallforholdfitness ga dermed ingen trikolor flagg som løsning.

For å diktere evolusjonen mot et løsningsmønster som et triokolor flagg, ble det forsøkt med ulike kombinasjoner og tillegg i fitnessfunksjonene. Den endelige fitnessfunksjonen er vist i algoritme 5.4.

$R_T^{min} = 0.3$
$R_T^{med} = 0.3$
$R_T^{max} = 0.4$
for CELLE(0,0)→CELLE(N,N) I FERDIG-UTVIKLET-FENOTYPE **do**
  $R^{min} = min/phenotype\_size$
  $R^{med} = med/phenotype\_size$
  $R^{max} = max/phenotype\_size$
  $D^{min} = \frac{|R_T^{min} - R^{min}|}{N^{min}}$ $\{N^{min} = maksavvik,\ normaliserer\ til\ [0-1]\}$
  $D^{med} = \frac{|R_T^{med} - R^{med}|}{N^{med}}$ $\{N^{med} = maksavvik,\ normaliserer\ til\ [0-1]\}$
  $D^{max} = \frac{|R_T^{max} - R^{max}|}{N^{max}}$ $\{N^{max} = maksavvik,\ normaliserer\ til\ [0-1]\}$
  $R_T^a = \frac{R_T^{min}}{R_T^{max}}$
  $R_T^b = \frac{R_T^{med}}{R_T^{max} + R_T^{max}}$
  $R^a = min/max$
  $R^b = med/(min + max)$
  $D^a = \frac{|R_T^a - R^a|}{N^a} \{N^a = maksavvik, normaliserertil[0-1]\}$
  $D^b = \frac{|R_T^b - R^b|}{N^b} \{N^b = maksavvik, normaliserertil[0-1]\}$
  $F^{min} = (1 - D^{min}) * 1200$
  $F^{med} = (1 - D^{med}) * 1200$
  $F^{max} = (1 - D^{max}) * 1200$
  $F^a = (1 - D^a) * \frac{F^{min} + F^{max}}{2}$
  $F^b = (1 - D^b) * F^{med}$
  if $(fitness > MAX - PROLIFERATIONFITNESS)$ **then**
    $fitness = fitness + F^a + F^b$
  end if
end for

Algoritme 5.4: Pseudokoden for flaggindividenes fitnessfunksjon for tallforhold med 3 celletyper, red, blue og green. Tallforholdet mellom typene og tallforholdet mellom faktiske celler uttrykt i fenotypen.

for RAD(0)→RAD(N) I FERDIG-UTVIKLET-FENOTYPE **do**
  $fitness = fitness + (dominante - celler * 10)$
end for

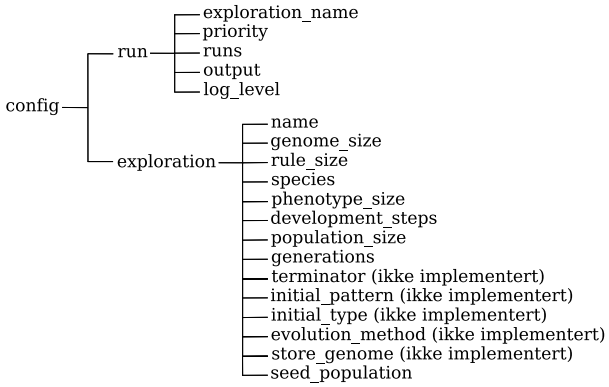
Algoritme 5.5: Pseudokoden for flaggindividenes identisk-sum-fitness. Fitnesspoeng gis for hver rad. Poenget i hver rad er gitt som multiplum av antallet celler av radens dominante celletype.

```
makskoblinger = rader/3
for RAD(0)→RAD(N) I FERDIG-UTVIKLET-FENOTYPE do
  if radtype! = EMPTY then
    if koblinger = 0 then
      radtype = RAD
      koblinger = 1
    else
      if (koblinger > 0)&(kobling mellom identiske rader)&(koblinger <
      makskoblinger then
        koblinger = koblinger + 1
        if identisk − sum − fitness > (maks_identisk−sum−fitness)/2 then
          fitness = fitness + (F^min + F^med + F^max)/10
        end if
      else
        koblinger = 1
        radtype = RAD
      end if
    end if
  else
    koblinger = 0
    radtype = RAD
  end if
end for
```

Algoritme 5.6: Pseudokoden for flaggindividenes koblingsfitness. Fitnesspoeng gis for hver kobling mellom rader med lik celltypedominans. Det er kobingene som gir poeng, og flere påfølgende koblinger vil dermed gi bedre uttelling enn mange enkeltkobinger mellom et mindre sett av like rader.

5.2 Oppsett og bruk

Plattformen er designet for å muliggjøre eksperimentering med parameterverdiene for de tre artene, vist i tabell 5.1. Parametrene definerer egenskaper ved evolusjons- og utviklingsprosessen for hvert eksperiment. Et eksperimentoppsett er et fullstendig sett med definerte parametere og settes i en konfigurasjonsfil. For å utføre et forsøk må et gyldig eksperimentoppsett være definert. Nødvendige oppføringer med tilhørende parametere for konfigurasjonsfilen er vist i figur 5.4.



Figur 5.4: Obligatoriske elementer i konfigurasjonsfila, ofte kalt exploreed.conf

Eksperimentet settes opp med en *exploration* oppføring i konfigurasjonsfilen. Det er mulig å ha flere eksperimentoppsett i samme konfigurasjonsfil. Det er dermed nødvendig å gi hver *exploration* oppføring et unikt navn med parameteren *name*. Antallet gener i genomet settes i *genome_size* og må være et positivt multiplum av regelstørrelsen. Regelstørrelsen settes som et positivt heltall i *rule_size*, men må være satt til 6 med nåværende plattform. Antallet regler i individets regelsett er $genome\_size/6$ og følger av plattformens tolkning fra genom til regelsett vist i figur 5.2. Arten settes med parameteren *species* til en av plattformens tre definerte arter vist i tabell 5.1. Plattformen benytter dermed evalueringsprosessen for den aktuelle arten beskrevet i 5.1.3 eller 5.1.4. Fenotypen er alltid en kvadratisk cellestruktur og antallet celler i en rad settes med *phenotype_size*. Verdien gis som et positivt heltall der 8 tilsvarer en organisme med 64 celler. Antall utviklingstrinn sattes med *development_steps* og populasjonsstørrelsen med *population_size*. Antall generasjoner populasjonen skal gjennomgå settes med parameteren *generations* og er termineringskriteriet for evolusjonen. *Terminator* brukes ikke, men er tiltenkt å muliggjøre valg mellom for eksempel generasjoner og fitnessverdi som termineringskriterie. *Initial_pattern* er tiltenkt å muliggjøre valg mellom ulike forhåndsdefinerte oppstartsmønster i fenotypen der *initial_type*

settes til ønsket celletype. Plattformen har kun en evolusjonsmetode, men med parameteren *evolution_method* vil valg av forhåndsdefinerte varianter av GAen muliggjøres. *Store_genome* brukes ikke, men er ment som en *yes* eller *no* verdi for å avgjøre om genomet til løsningsindividet skal lagres i en egen fil i tillegg til i loggfila. Dette for eventuell bruk som seed (frø, bio.plantemetafor) underveis eller i senere forsøk der dette er mer hensiktsmessig enn å lete det fram i loggfila. Parameteren *seed_population* kan benyttes for å starte opp et forsøk med genomet fra et tidligere resultat. Den initielle populasjonen opprettes da ikke tilfeldig, men i steden ut i fra dette *seedet*. *seed_population* er vanligvis *no* og populasjonen initialiseres med tilfeldige individ (genom). Ved å sette verdien til *yes* vil plattformen isteden forvente et eksternt individ angitt med **-s [tekstfil_med_genom]** ved oppstart. **[tekstfil_med_genom]** må inneholde genomet (seedet) i første tekstlinje.

Et forsøk (*run*) settes opp med en *run* oppføring i konfigurasjonsfila. Det er mulig å ha flere forsøksoppsett i samme konfigurasjonsfil. Rekkefølgen disse utføres i avgjøres av *priority* parameteren. *Priority* angis med et positivt heltall. Forsøket med lavest *priority*verdi starter først. Antall repetisjoner av hvert forsøk angis som et positivt heltall for *runs* parameteren. Alle repetisjoner i et forsøk utføres før neste forsøk i prioriteringslist starter. *Run* oppføringen må kobles mot *exploration* oppføringen som skal benyttes. *Exploration_name* må settes til det unike navnet til en eksisterende *exploration* oppføring. *Output* parameteren må settes til verdien 5 for rapportering av loggdata fra utviklingsprosessen for løsningen. Alle andre positive heltallsverdier gir kun standardtype loggdata. *Log_level* angir mengden loggdata som rapporteres, men er ikke implementert. Denne verdien settes med et positivt heltall.

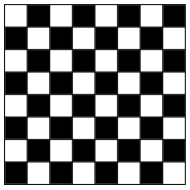
Plattformen gir ikke perfekte løsninger i hvert forsøk. Det er derfor vanlig å sette opp 10 forsøk av gangen for å være sikker på å få et perfekt resultat. Det gjøres enkelt ved å endre *run* parameteren *runs* til 10.

Konfigurasjonsfila for sjakkbrettforsøket er vedlagt i appendiks A.

Kapittel 6

Skalering og sjakkbrett

I dette kapittelet presenteres realiseringen av sjakkbrettmønsteret, vist i figur 6.1 med tilhørende skaleringseksperimenter. Realiseringen av sjakkbrettmønsteret ble delvis utført for å undersøke hvor godt EAen virket og delvis for å etablere en retningslinje for implementering av plattformen. Strukturproblemer som sjakkbrettet danner grunnlaget for videre eksperimentell analyse og plattformutvikling av utviklings- og evolusjonssystemet.



Figur 6.1: Ønsket mønster.

Et sjakkbrettmønster har enkle koblinger mellom celletypene, men da alle cellene oppdateres i parallell vil organiseringen av et slik mønster likevel være en kompleks *emergent computation*. Den enkle lokale oppførselen og de enkle lokale bindingene mellom strukturkomponentene i naboskapet for et sjakkbrettmønster er i seg selv ikke nok til å frambringe en global struktur. Dette skyldes samtidigheten i utviklingsprosessen der hvert naboskap må forholde seg til andre naboskap i en kompleks parallell oppdateringsprosess. Fitnessfunksjonene for å oppnå dette mønsteret er vist i algoritme 5.2.

6.1 Oppsett av forsøk

Eksperimentet settes opp som vist i figur 6.2 og utføres med de 3 celletypene vist i tabell 6.1. Genomet inneholder 30 gener som i kombinasjon med 6 gentyper i reglene gir et regelsett med 5 regler. Fenotypen er alltid kvadratisk og størrelsen 8 tilsvarer da en organisme med 64 celler. Antall utviklingstrinn for alle skaleringsforsøkene satt til 100 for å gi større frihet for evolusjonen. Flertallet av individene var ferdig utviklet før 20 utviklingstrinn. Resultateksempelet presentert i 6.3 er utført med tilsvarende oppsett som i 6.2, men med *development_steps* satt til 20. Populasjonsstørrelsen var 5 for alle forsøkene. Forsøkene terminerer alltid for et maksimalt antall generasjoner som i dette eksperimentet er satt til 3000. Arten er satt til *neutral* siden et sjakkbrettmønster er målet for eksperimentet. Evolusjonen benytter dermed evalueringsprosessen for sjakkbrettindivider beskrevet i 5.1.3.

```
============================================================
#. 8x8-100-5-30-neutral
============================================================
genome_size:          30    spicies:             neutral
rule size:             6    terminator:          generations
phenotype size:        8    initial pattern:     single cell
development steps:   100    initial type:        black
population size:       5    evolution method:    randmut-3
generations:        3000    store genome:        false
                            seed population:     false
------------------------------------------------------------
```

Figur 6.2: Sjakkbretteksperimentet ble utført med dette oppsettet

Type	Navn	Grafisk
B	Sort	●
W	Hvit	○
Z	Tom	◍

Tabell 6.1: Sjakkbretteksperimentene kunne benytte disse celletypene.

6.2 Resultat

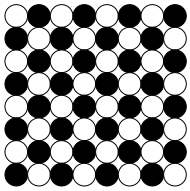
Genotype- og fenotyperesultatet av et suksessfullt forsøk er vist i figur 6.3. Individet er det beste i populasjonen etter 3000 generasjoner med oppsettet vist i 6.2.

Genomet for utviklingsprosessen er vist i figur 6.3(a) og inneholder 3 *growth* regler og 2 *change* regler. Etter endt utvikling oppnås en perfekt *emergent* sjakkbrettstruktur vist i 6.3(b). Figur 6.4 viser de første 12 utviklingstrinnene til individet. Individet utvikles fra en enkelt sort celle til en multicellulær organisme med 64 celler som uttrykker det ønskede sjakkbrettmønsteret. Eksempelet viser en perfekt løsning. Fenotypen har en stabil struktur ved utviklingstrinn 11. Det skjer altså ingen endring i fenotypen fra utviklingstrinn 11 til trinn 19.

Organismen i figur 6.3 endte på 3200 i fitness for sin strukturelle komposisjon (fenotypefitness) og regelfitness på 286 hvor 100 poeng var for de 5 korrekte reglene. Individet ble totalt tildelt en fitnessverdi på 3486.

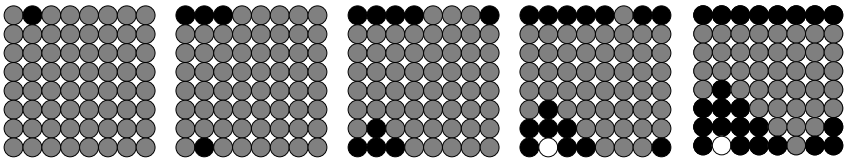
#	R	C	N	S	E	W
0	Gw	●	DC	●	●	DC
1	Ge	●	DC	●	DC	DC
2	Gs	●	DC	DC	DC	DC
3	○	○	○	●	●	DC
4	○	●	●	●	●	●

(a) Genotypen.

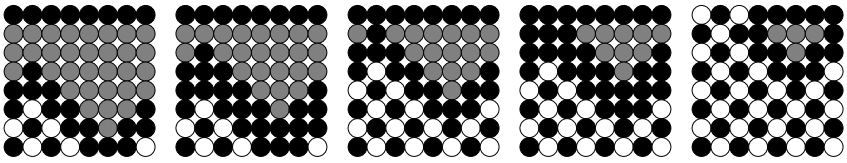


(b) Fenotypen.

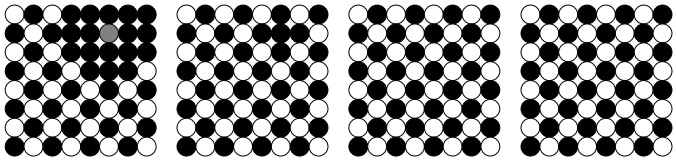
Figur 6.3: Det resulterende individet etter utvikling i et av forsøkene. (a) Resulterende regelsett etter tolkningen av genomet. Regel 0 har høyest prioritet. (b) Resulterende struktur etter utvikling av fenotypen.



(a) Oppstart *(b)* Trinn 0 *(c)* Trinn 1 *(d)* Trinn 2 *(e)* Trinn 3



(f) Trinn 4 *(g)* Trinn 5 *(h)* Trinn 6 *(i)* Trinn 7 *(j)* Trinn 8

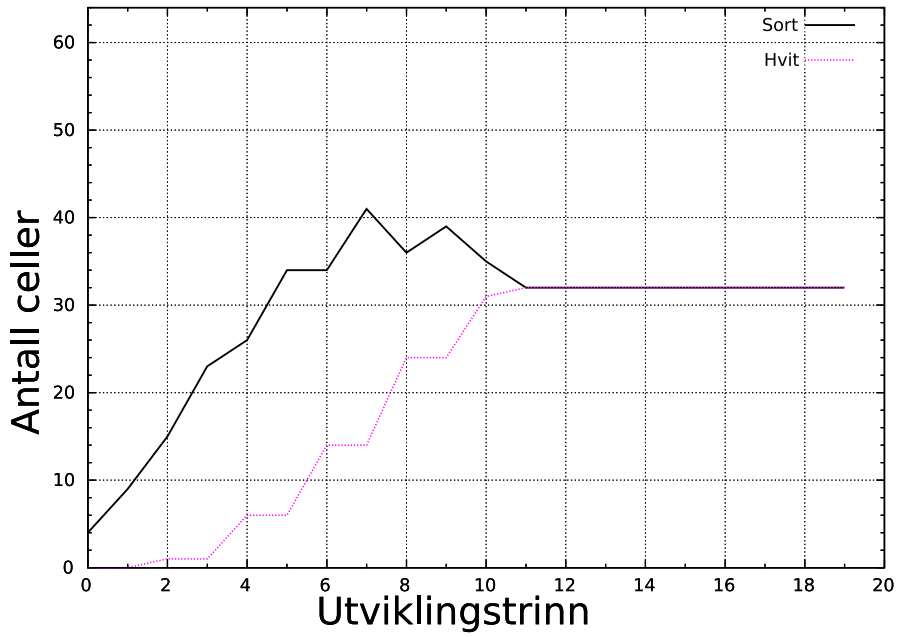


(k) Trinn 9 *(l)* Trinn 10 *(m)* Trinn 11 *(n)* Trinn 19

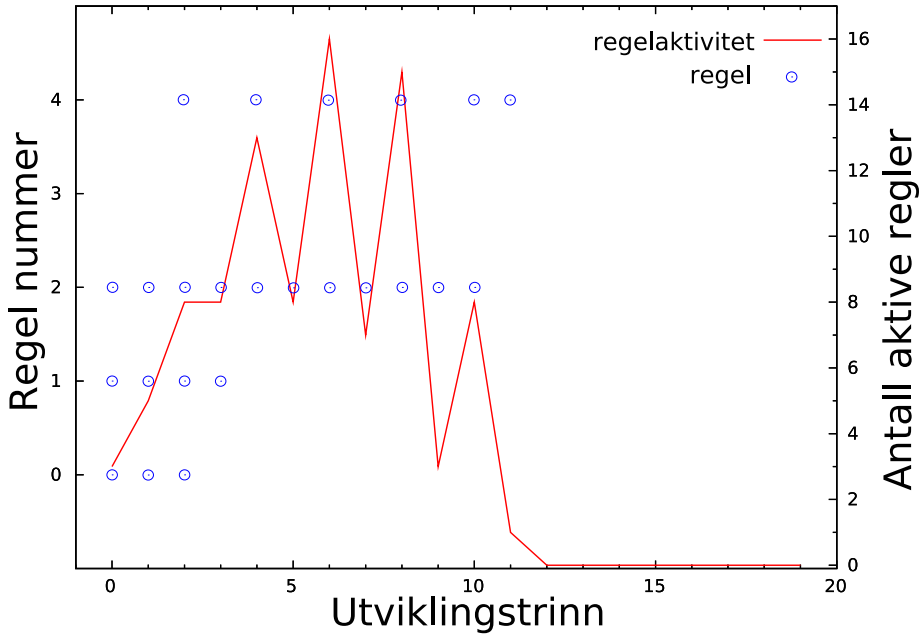
Figur 6.4: Utviklingstrinn for endelig individ. Mønsteret er stabilt fra trinn 11 til siste trinn, trinn 19.

Figur 6.5 viser proliferasjonen (utbredelsen) av celler i fenotypen under utviklingen i figur 6.4. Proliferasjonen av hver enkelt celletype er vist med en egen graf. Graffargen assosiert med hver celletype er gitt øverst i diagrammet. Antallet celler er angitt i den venstre Y-aksen.

I figur 6.6 er genaktiveringsmønsteret vist for genotypen i figur 6.3(a) med genomstørrelse på 5 regler som gir perfekt løsning i siste utviklingstrinn. Figur 6.4



Figur 6.5: Proliferasjonen av celler vist for de uttrykte celletypene i fenotypen under utviklingen i 6.4. Proliferasjonen for hver enkelt celletype er vist med en egen farget graf. Graffargen assosiert med hver celletype er gitt øverst i diagrammet. Antallet celler er angitt i den venstre Y-aksen.



Figur 6.6: Regelaktiviteten under utvikling. Reglene fra 0 til 4 er plassert på den venstre Y-aksen. Regel 0 har høyest prioritet. Hver markering (o) i plottet indikerer at regelen ble aktivert i gitt utviklingstrinn. Den høyre Y-aksen viser antallet celler med aktiv regel i gitt utviklingstrinn. Antallet aktive regler er likt antallet aktive celler og vises i den plotta grafen.

viser utviklingen av fenotypen. Plottet i figur 6.6 viser genaktivieringen (aktive regler) og det totale antallet aktive regler i organismen for hvert utviklingstrinn. Reglene fra 0 til 4 er plassert på den venstre Y-aksen. Hver markering (o) i plottet indikerer at regelen ble aktivert i dette utviklingstrinnet. Den høyre Y-aksen viser antallet celler i organismen med en aktiv regel i det gitte utviklingstrinnet. Antallet aktive regler for hvert utviklingstrinn vises i den plotta grafen.

Plottet har et økende antall aktive regler fra første utviklingstrinn til et maksimum på 16 i utviklingstrinn 6. Fra utviklingstrinn 6 synker antallet aktive regler ned til 0 i utviklingstrinn 12. Fenotypen har da kommet fram til en stabil struktur og har ikke genregulatorisk aktivitet (aktive regler) etter trinn 11.

6.3 Skalering av resultat

Hvilke cellulære strukturer det er mulig å bygge er avhengig av antall tilgjengelige celletyper og antall tilgjengelige instruksjoner for konstruksjon [22]. Nødvendig antall celletyper for generelle beregninger i cellulære maskiner ble undersøkt av Sipper [17]. Inspirert av disse resultatene ble eksperimenter med skalering av tilgjengelig informasjon utført [21]. Forsøkene ble gjort for å undersøke hvordan skaleringen påvirket evolusjonen av genome brukt til utvikling.

For å undersøke hvilken innvirkning fenotypestørrelsen har på resultatet av EAen ble antall celler tilgjengelige for utviklingsprosessen økt. Det ble gjort fire ulike forsøk med fenotypstørrelser med 16, 64, 256 og 1024 celler. Oppsettet er som i figur 6.2, men *phenotype_size* varieres (4, 8, 16, 32) i de fire forsøkene.

Fenotypens struktur ved start, vist i figur 6.4(a), ble satt til å være en enkelt celle av typen sort og resten som tomme celletyper. Antallet utviklingstrinn (*development_steps*) ble satt til 100. Populasjonsstørrelsen er 5, og maksimalt antall generasjoner er 3000.

Regler	Størrelse	SR
5	16	55%
5	64	64%
5	256	59%
5	1024	61%

Tabell 6.2: Resultat med 5 regler: fenotypestørrelsen skalert fra 16 til 1024 celler.

Regler	Størrelse	SR
4	16	33%
4	64	41%
4	256	34%
4	1024	44%

Tabell 6.3: Resultat med 4 regler: fenotypestørrelsen skalert fra 16 til 1024 celler.

Regler	Størrelse	SR
6	16	68%
6	64	70%
6	256	73%
6	1024	66%

Tabell 6.4: Resultat med 6 regler: fenotypestørrelsen skalert fra 16 til 1024 celler.

Tabell 6.2 viser resultatet fra de fire forsøkene når fenotypestørrelsen økes. I tabellen viser *Regler* antall mulige regler i genomet. *Størrelse* er det maksimale antallet celler tilgjengelige for utvikling, organismens maksimale størrelse. Suksessraten (*SR*) viser prosentandelen av perfekte løsninger funnet ved å repetere hvert forsøk 100 ganger.

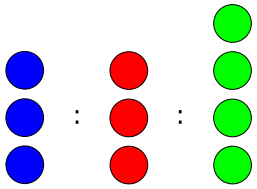
I tillegg til de fire forsøkene i tabell 6.2 ble det gjort to tilsvarende forsøk med fenotypskalering der størrelsen på genotypen ble endret. I tabell 6.3 er antall regler redusert til 5. Fenotypestørrelsen økes som før fra 16 til 1024 celler.

I det siste eksperimentet vist i tabell 6.4 økes genotypestørrelsen til 6 regler, altså *genome_size* lik 36. Igjen blir fenotypstørrelsen skalert opp fra 16 til 1024 celler i fire forsøk.

Kapittel 7

Tallforhold

I dette kapittelet presenteres eksperimenter med tallforholdet mellom celletyper i fenotypen, illustrert i figur 7.1. Eksperimentet ble utført som et første steg i søken etter en fitnessfunksjon for en flaggstruktur. En fitness basert på tallforhold setter få begrensninger for evolusjonen og er en generell tilnærming som derfor kan fungere godt sammen med andre fitnesskriterier.



Figur 7.1: Eksperimenter med tallforhold. Ønsket tallforhold settes i fitnessfunksjonen for å undersøke om fordelingen fremkommer i fenotypen, her illustrert for tallforholdet 3:3:4 mellom 3 celletyper.

Beregningene som skal til for å holde orden på det totale antallet av hver celletype er ikke en egnskap som enkeltcellene eller naboskapene deres innehar. Det er ingen globale variable som kan telle celletypene i løpet av utviklingen. Tilbakemeldingen for hvor godt tallforholdet mellom celletypene er blir først gitt i siste utviklingstrinn, og det er altså generasjonene av populasjoner som må finne fram til et stadig bedre tallforhold samtidig som cellerproliferasjonen foregår. Samtidigheten i utviklingsprosessen og fremveksten av en global inndeling fra den enkle lokale oppførselen tilsier at dette er en *emergent computation*. Fitnessfunksjonene for å oppnå denne inndelingen er vist i algoritme 5.3 for 2 uttrykte celltyper og i algoritme 5.4 for 3 uttrykte celletyper.

7.1 Oppsett av forsøk

Eksperimentet settes opp som vist i figur 7.2 og utføres med 2 og 3 mulige celletyper i tillegg til tomme celler vist i tabell 7.1. Genomet inneholder 48 gener som i kombinasjon med 6 gentyper i reglene gir et regelsett med 8 regler. Fenotypen er alltid kvadratisk og størrelsen 16 tilsvarer da en organisme med 256 celler. Antall utviklingstrinn er satt til 100 og populasjonsstørrelsen til 5. Forsøkene terminerer alltid for et maksimalt antall generasjoner som i dette eksperimentet er satt til 3000. Arten er satt til *flag* siden dette var første steg på veien mot en ny artsdefinisjon i plattformen. Evolusjonen benytter tallforholdsalgoritmer tilsvarende algoritme 5.3 og 5.4 i avsnitt 5.1.4. Tallforholdsalgoritmene i avsnitt 5.1.4 er resultat av disse eksperimentene. Det er ikke spesifisert hvilken celletype som skal ha hvilken andel, noe som gir evolusjonene større frihet.

```
============================================================
#. 16x16-100-5-48-flag
============================================================
genome_size:          48    spicies:             flag
rule size:             6    terminator:          generations
phenotype size:       16    initial pattern:     single cell
development steps:   100    initial type:        black
population size:       5    evolution method:    randmuta-3
generations:        3000    store genome:        false
                            seed population:     false
------------------------------------------------------------
```

Figur 7.2: Tallforholdseksperimentene ble utført med dette oppsettet

Type	Navn	Grafisk
Gr	Grønn	
Bu	Blå	
R	Rød	
B	Sort	
W	Hvit	
Z	Tom	

Tabell 7.1: Tallforholdseksperimentene kunne benytte disse celletypene.

7.2 Resultat

Genotype- og fenotyperesultatet av et suksessfullt forsøk med tallforhold 3:3:4 er vist i figur 7.3. Individet er det beste i populasjonen etter 3000 generasjoner med oppsettet vist i figur 7.2.

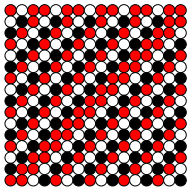
Genomet for utviklingsprosessen er vist i figur 7.3(a) og inneholder 2 *growth* regler og 6 *change* regler. Etter endt utvikling oppnås en *emergent* struktur som vist i 7.3(b). Tallforholdet i strukturen er 75 sorte, 80 hvite og 101 røde mot den ideelle tilnærmingen 77:77:102 for 256 celler. Det tilsvarer en feilandel på $2,3\%$, altså 6 celletyper av 256 er feil. Figur 7.4 viser et utvalg av de 100 utviklingstrinnene til individet. Individet utvikles fra en enkelt sort celle til en multicellulær organisme med 256 celler som uttrykker det ønskede tallforholdet. Eksempelet viser en nærmest perfekt løsning. Fenotypen har en stabil struktur

ved utviklingstrinn 47. Det skjer altså ingen endring i fenotypen fra utviklingstrinn 47 til trinn 99.

Organismen i figur 7.3 endte på 12846 i fitness for sin strukturelle komposisjon (fenotypefitness) og regelfitness på 457 hvor 160 poeng var for de 8 korrekte reglene. Individet ble totalt tildelt en fitnessverdi på 13303.

#	R	C	N	S	E	W
0	Gs	●	DC	●	DC	DC
1	○	●	●	●	●	●
2	●	●	DC	●	DC	●
3	●	●	●	DC	DC	○
4	Gw	●	DC	DC	DC	DC
5	●	○	○	●	●	DC
6	●	●	DC	DC	●	●
7	●	●	●	●	○	●

(a) Genotypen



(b) Fenotypen

Figur 7.3: Det resulterende individet etter utvikling i et av forsøkene. (a) Resulterende regelsett etter tolkningen av genomet. Regel 0 har høyest prioritet. (b) Resulterende struktur etter utvikling av fenotypen.

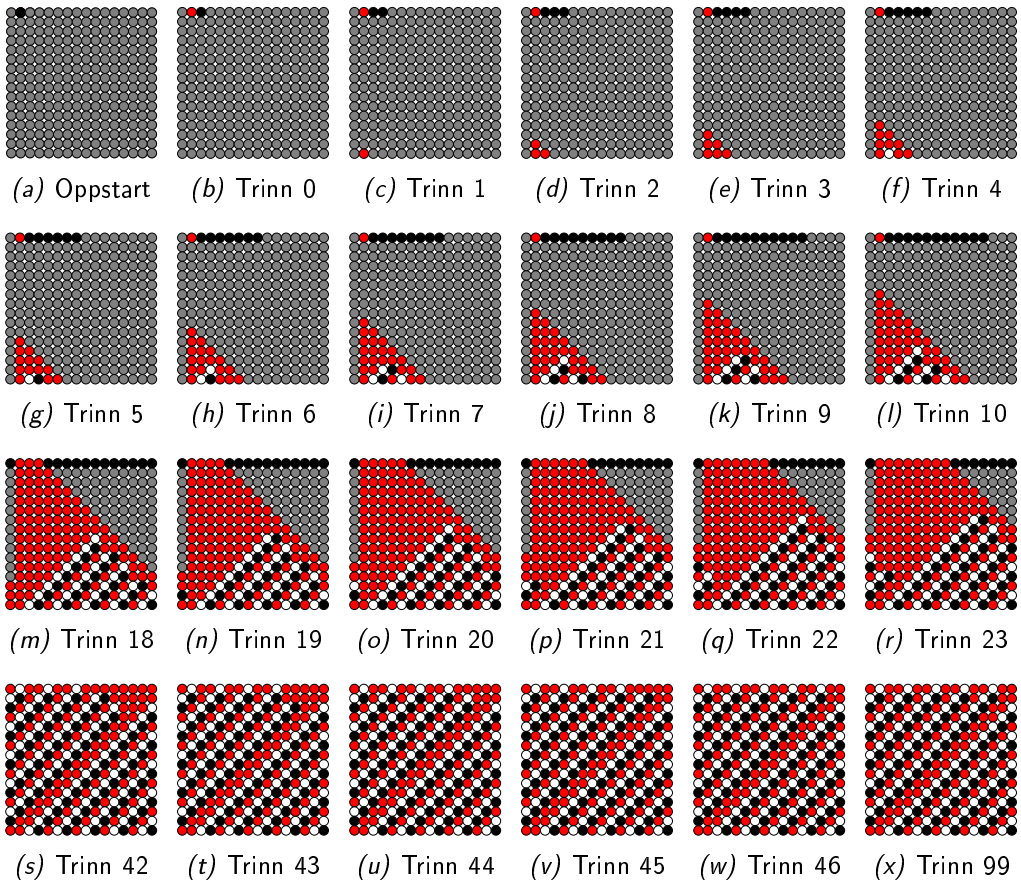
Figur 7.7(a) viser proliferasjonen av celler i fenotypen under utviklingen i figur 7.4. Proliferasjonen av hver enkelt celletype er vist med en egen graf. Graffargen assosiert med hver celletype er gitt øverst i diagrammet. Antallet celler er angitt i den venstre Y-aksen.

I figur 7.7(b) er genaktiveringsmønsteret vist for genotypen i figur 7.3(a) med genomstørrelse på 8 regler som gir god løsning i siste utviklingstrinn. Figur 7.4 viser utviklingen av fenotypen. Plottet i figur 7.7(b) viser genaktivieringen (aktive regler) og det totale antallet aktive regler i organismen for hvert utviklingstrinn. Reglene fra 0 til 7 er plassert på den venstre Y-aksen. Hver markering (o) i plottet indikerer at regelen ble aktivert i dette utviklingstrinnet. Den høyre Y-aksen viser antallet celler i organismen med en aktiv regel i det gitte utviklingstrinnet. Antallet aktive regler for hvert utviklingstrinn vises i den plotta grafen.

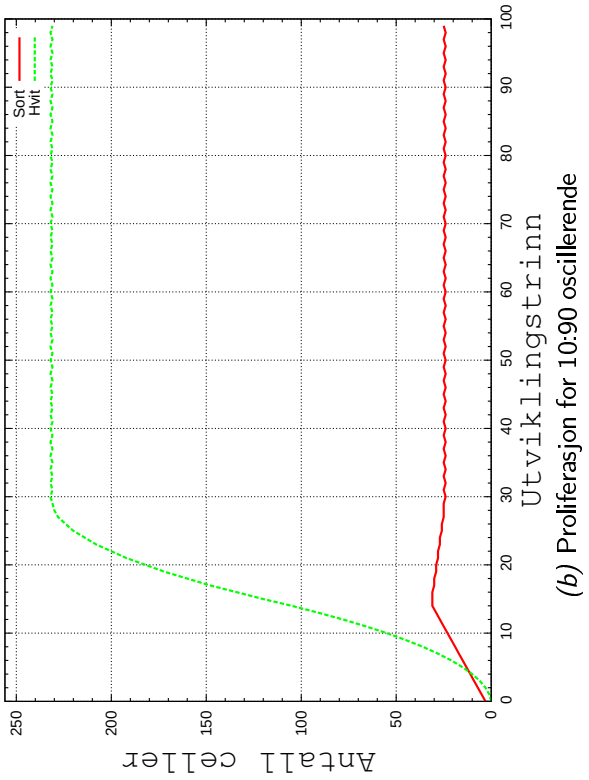
Plottet har et økende antall aktive regler fra første utviklingstrinn til et maksimum på 20 i utviklingstrinn 16. Fra utviklingstrinn 17 synker antallet aktive regler ned til 0 i utviklingstrinn 47. Fenotypen har da kommet fram til en stabil struktur og har ikke genregulatorisk aktivitet (aktive regler) etter trinn 47.

7.2.1 Tallforhold mellom to celletyper

Fire ulike resultatet av eksperimentene med 2 celletyper er vist i figur 7.5. I samtlige eksperimenter er oppsettet som vist i figur 7.2.

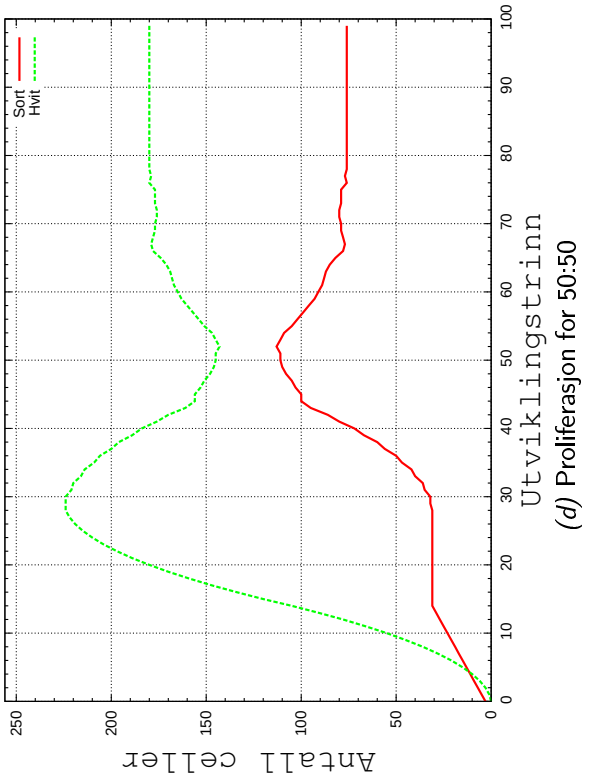


Figur 7.4: Utviklingstrinn for endelig individ. Mønsteret er stabilt fra trinn 46 til siste trinn, trinn 99.



(a) Proliferasjon for 10:90

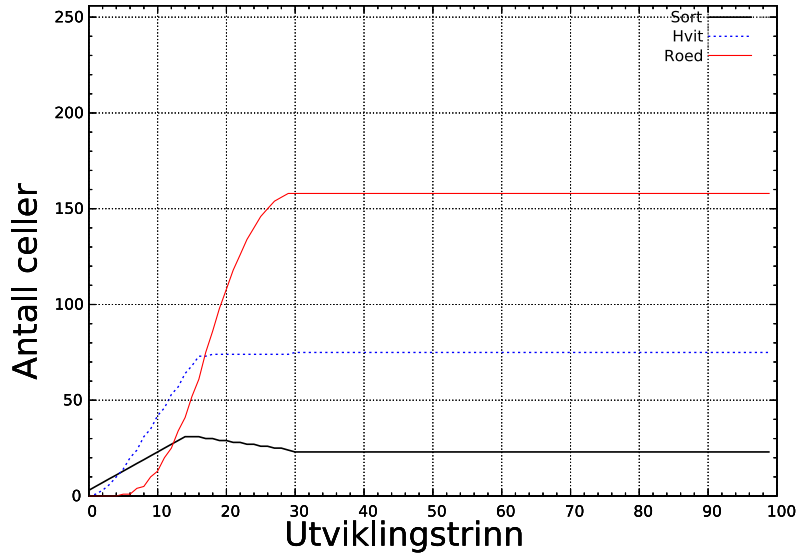
(b) Proliferasjon for 10:90 oscillerende



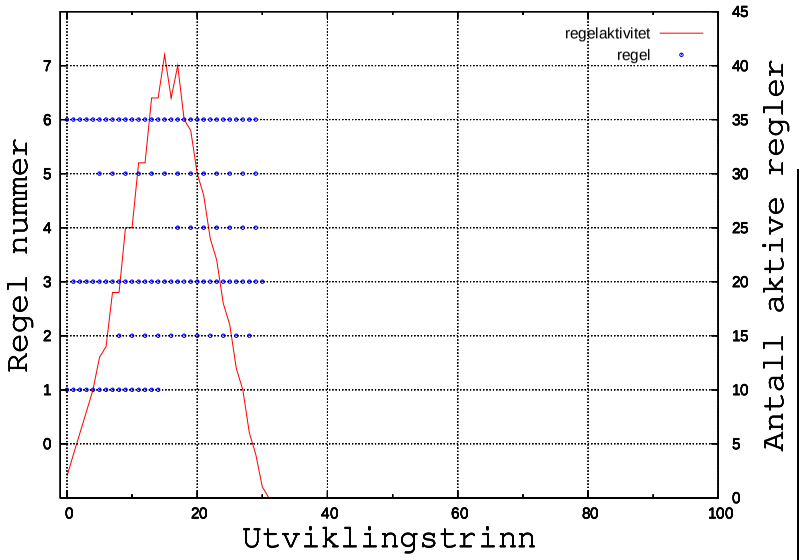
(c) Proliferasjon for 30:70

(d) Proliferasjon for 50:50

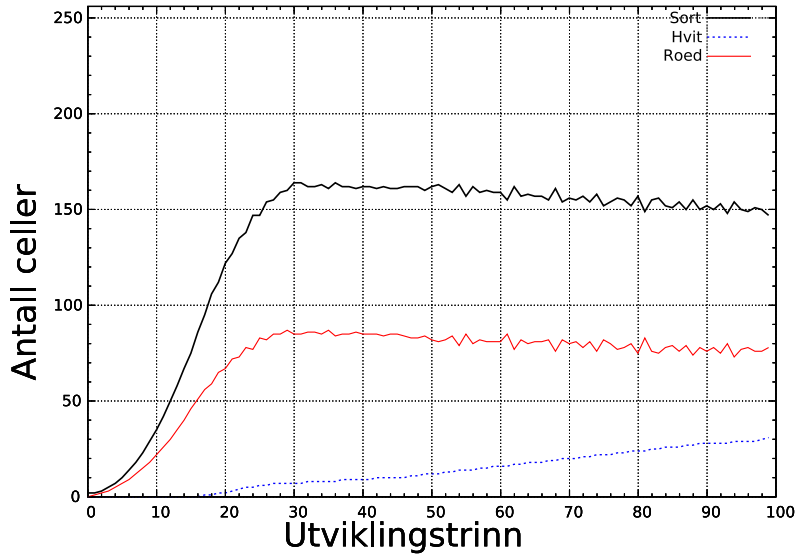
Figur 7.5: Eksperimenter der ulike tallforholdet mellom svarte og hvite celletyper gir maksimalt fitnesspoeng.



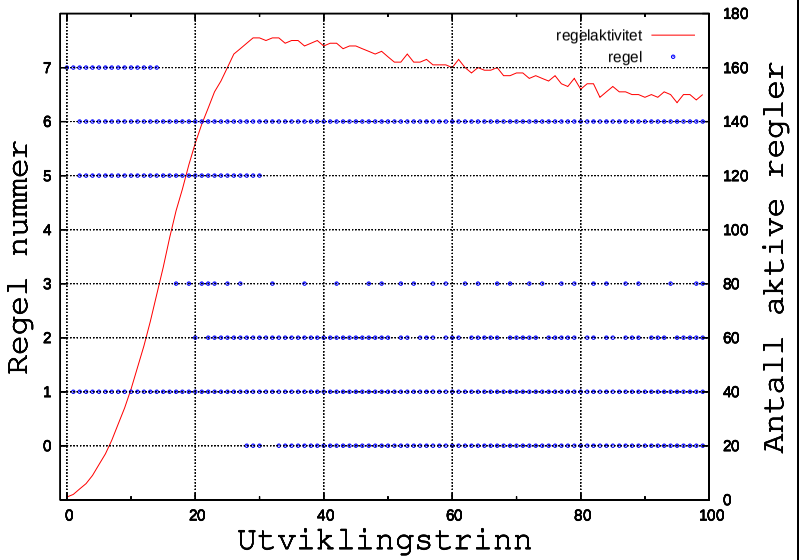
(a) Proliferasjon for 1:3:6 statisk



(b) Regelaktivitet for 1:3:6 statisk

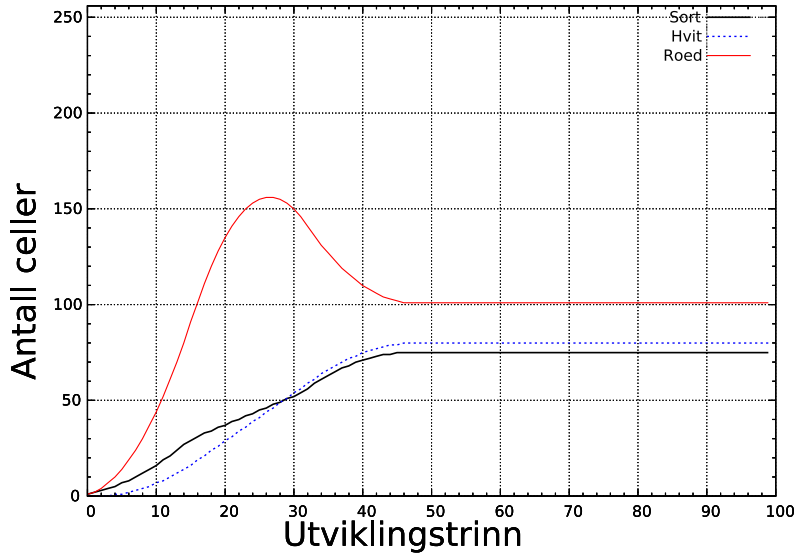


(c) Proliferasjon for 1:3:6 oscillerende

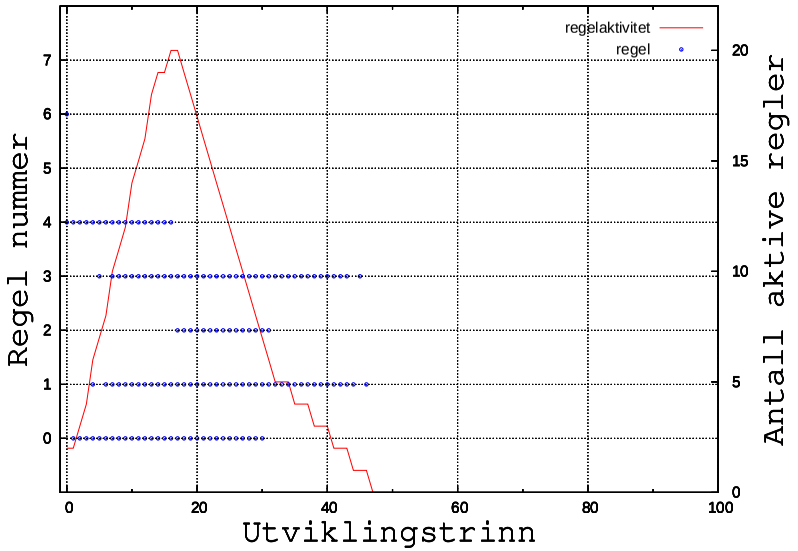


(d) Regelaktivitet for 1:3:6 oscillerende

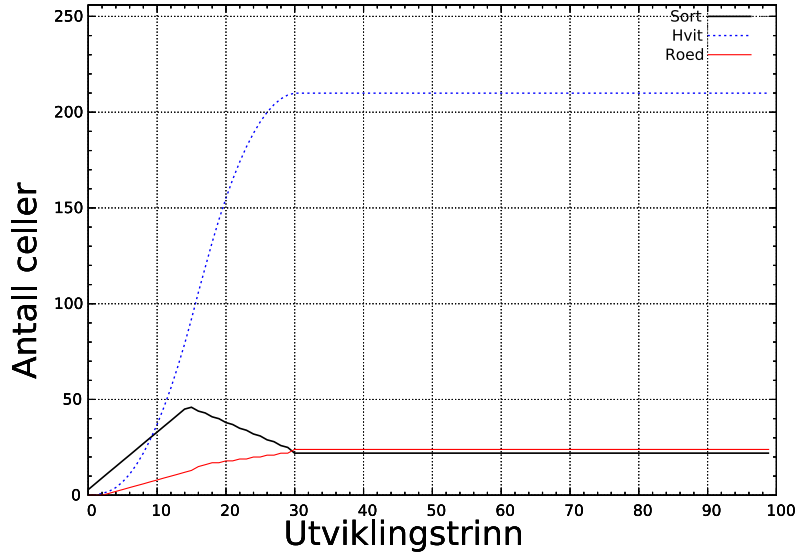
Figur 7.6: Eksperiment der tallforholdet 1:3:6 mellom svarte, hvite og røde celletyper gir maksimalt fitnesspoeng med tilhørende regelaktivitetsplot. (a) og (b) viser et resultat som oppnår stabilt mønster og avslutta regelaktivitet innen 100 utviklingstrinn. (c) og (d) viser et resultat med oscillerende mønster og høy regelaktivitet etter 100 utviklingstrinn. Begge løsningene gir en god fordeling av celletypene i henhold til ønsket tallforhold på 1:3:6.



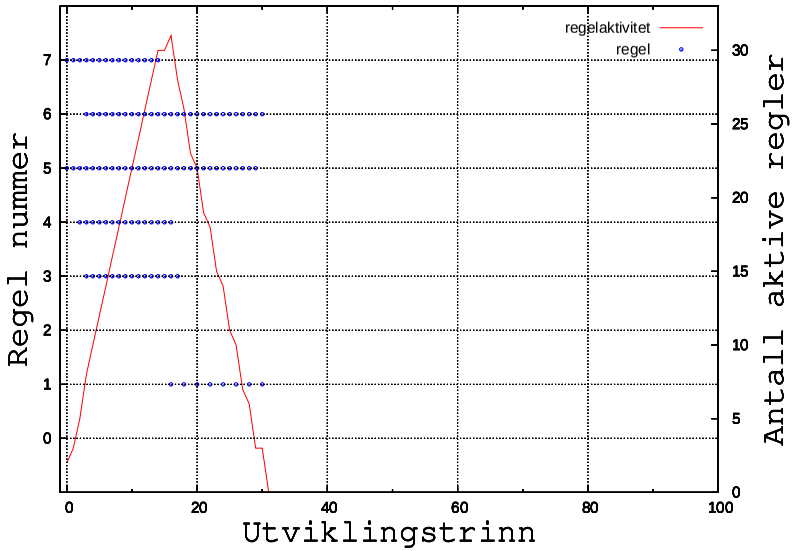
(a) Proliferasjon for 3:3:4



(b) Regelaktivitet for 3:3:4



(c) Proliferasjon for 1:1:8



(d) Regelaktivitet for 1:1:8

Figur 7.7: Eksperiment der ulike tallforholdet mellom svarte, hvite og røde celletyper gir maksimalt fitnesspoeng med tilhørende regel-aktivitetsplot.

Figur 7.5(a) viser proliferasjonen av celler i en fenotype med tallforhold satt til 1:9. Proliferasjonen av hver enkelt celletype er vist med en egen graf. Graffargen assosiert med hver celletype er gitt øverst i diagrammet. Antallet celler er angitt i den venstre Y-aksen. Plottet viser en jevn økning i antallet av både sorte og hvite celler. Begge celletypene når sin celleandel samtidig etter 30 utviklingstrinn. Det skjer ingen strukturell endring fra utviklingstrinn 32 til trinn 99. Fenotypen har da kommet fram til en stabil struktur. Tallforholdet i strukturen er 24 sorte og 232 hvite mot det tilnærmede ideelle 26:230. Det tilsvarer en feilandel på $1,6\%$, altså 4 celletyper av 256 er feil.

Proliferasjonen for en alternativ løsning med tallforhold 1:9 er vist i figur 7.5(b). Plottet viser også her en nesten jevn økning i antallet av både sorte og hvite celler, og at begge celletypene når sin celleandel samtidig etter 30 utviklingstrinn. Derimot ender fenotypen ikke opp med en statisk struktur, men isteden en dynamisk og periodisk oscillerende struktur. Fenotypen er regelmessig og jevnt oscillerende fra utviklingstrinn 30 til trinn 99. Tallforholdet i strukturen oscillerer mellom 24 sorte og 232 hvite og 25 sorte og 231 hvite. Sammenlignet med den ideelle tilnærmingen på 26:230, tilsvarer dette en feilandel på $0,8\%$ eller $1,6\%$, altså 2 eller 4 celletyper av 256 er feil.

I figur 7.5(c) vises proliferasjonen av celler i et forsøk med tallforhold satt til 3:7. Plottet viser en jevn økning i antallet av både sorte og hvite celler. Begge celletypene når sin celleandel samtidig ved utviklingstrinn 30. Det skjer ingen strukturell endring fra utviklingstrinn 30 til trinn 99. Fenotypen har da kommet fram til en stabil struktur. Tallforholdet i strukturen er 74 sorte og 182 hvite mot den ideelle tilnærmingen på 77:179. Det tilsvarer en feilandel på $2,3\%$, altså 4 celletyper av 256 er feil.

Figur 7.5(d) viser proliferasjonen av celler i et forsøk med tallforhold satt til 5:5 (1:1). Plottet viser en kraftig økning i antallet hvite celler som rett før utviklingstrinn 30 snur og synker i takt med det økende antallet sorte celler. Begge celletypene når ved utviklingstrinn 52 verdier (115 sorte og 145 hvite) som ville gitt et godt resultat. Derimot øker forskjellen i andelene og ved utviklingstrinn 72 har de stabilisert seg på 78 sorte og 180 hvite celler. Det skjer ingen strukturell endring fra utviklingstrinn 78 til trinn 99. Fenotypen har da kommet fram til en stabil struktur. Det ideelle tilnærmede tallforholdet for 256 celler er 128:128. Løsningen har en feilandel på $40,6\%$, altså 104 celletyper av 256 er feil.

7.2.2 Tallforhold mellom tre celletyper

To ulike resultatet av eksperimentene med 3 celletyper og tallforholdet 1:3:6 er vist i figur 7.6. I samtlige eksperimenter er oppsettet som vist i figur 7.2.

Figur 7.6(a) viser proliferasjonen av celler i fenotypen under utviklingen når tallforhodet er 1:3:6. Proliferasjonen av hver enkelt celletype er vist med en egen graf. Graffargen assosiert med hver celletype er gitt øverst i diagrammet. Antallet celler er angitt i den venstre Y-aksen.

I figur 7.6(b) er genaktiveringsmønsteret vist for individet i figur 7.6(a) med genomstørrelse på 8 regler som gir en god løsning i siste utviklingstrinn. Plottet i figur 7.6(b) viser genaktivieringen og det totale antallet aktive regler i organismen for hvert utviklingstrinn. Reglene fra 0 til 7 er plassert på den venstre Y-aksen. Hver markering (o) i plottet indikerer at regelen ble aktivert i dette utviklingstrinnet. Den høyre Y-aksen viser antallet celler i organismen med en aktiv regel i det gitte utviklingstrinnet. Antallet aktive regler for hvert utviklingstrinn vises i den plotta grafen.

Plottet har et økende antall aktive regler fra første utviklingstrinn til et maksimum på 41 i utviklingstrinn 15. Fra utviklingstrinn 15 synker antallet aktive regler ned til 0 i utviklingstrinn 31. Fenotypen har da kommet fram til en stabil struktur og har ikke genregulatorisk aktivitet etter trinn 31.

Tallforholdet i strukturen er 23 sorte, 75 hvite og 158 røde mot den ideelle tilnærmingen på 26:77:153. Det tilsvarer en omtrentlig feilandel på $3,9\%$, altså 10 celletyper av 256 er feil.

Proliferasjonsdiagrammet for et alternativt resultat er vist i figur 7.6(c). Tallforholdet er også her 1:3:6. I figur 7.6(d) er genaktiveringsmønsteret vist for individet i figur 7.6(c).

Plottet har et økende antall aktive regler fra første utviklingstrinn til et maksimum på 171 i utviklingstrinn 29. Fra utviklingstrinn 31 synker antallet aktive regler ned til 150 i utviklingstrinn 99. Mønsteret stabiliseres altså ikke av den genregulatorisk aktiviteten, men avslutter utviklingen umiddelbart i trinn 99 siden individet har en begrenset utviklingssyklus på 100 trinn. Individet ender ikke opp med en statisk struktur, men en dynamisk og periodisk oscillerende struktur. Fenotypen er regelmessig og jevnt oscillerende fra utviklingstrinn 52 til trinn 99. Tallforholdet i strukturen oscillerer rundt forholdet 31 hvite, 78 røde og 147 sorte som var forholdet i trinn 99. Den ideelle tilnærmingen var 26:77:153 så i siste

trinn hadde individet en feilandel på $4,7\%$, tilsvarende 12 celletyper feil av 256 celler.

Figur 7.7(c) viser proliferasjonen av celler i et forsøk med tallforhodet 1:1:8. Tilhørende genaktiveringsmønsteret er vist i figur 7.7(d).

Plottet har et økende antall aktive regler fra første utviklingstrinn til et maksimum på 31 i utviklingstrinn 16. Fra utviklingstrinn 16 synker antallet aktive regler ned til 0 i utviklingstrinn 31. Fenotypen har da kommet fram til en stabil struktur og har fra utviklingstrinn 31 til 99 verken genregulatorisk aktivitet eller endring i fenotypestrukturen.

Tallforholdet i strukturen er 22 sorte, 210 hvite og 24 røde mot den ideelle tilnærmingen på 26:26:204. Det tilsvarer en feilandel på $4,7\%$, altså 12 celletyper av 256 er feil.

7.2.3 Tallforhold og feilandel oppsummert

Feilandelene for alle tallforholseksperimentene er vist i tabell 7.2 og tabell 7.3.

Tallforhold	Feilandel
$1:9$	1,6%
$1:9$	1,6%
$3:7$	2,3%
$1:1$	40,6%

Tabell 7.2: Oppsummering av feilandelen i tallforholdene med 2 celletyper.

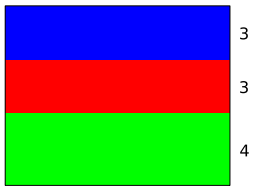
Tallforhold	Feilandel
$1:1:8$	4,7%
$1:3:6$	3,9%
$1:3:6$	4,7%
$3:3:4$	2,3%

Tabell 7.3: Oppsummering av feilandelen i tallforholdene med 3 celletyper.

Kapittel 8

Flaggmønster

I dette kapittelet presenteres realiseringen av flaggmønsteret, vist i figur 8.1 med tilhørende skaleringseksperimenter. Realiseringen av flaggmønsteret ble delvis utført for å gi plattformen en utfordring som ennå ikke hadde blitt løst og delvis for å katalysere et økt behov for forbedringer i plattformen. Strukturproblemet krevde eksperimentell utarbeiding av en egnet fitnessfunksjon basert på analyse av loggdata fra plattformen.



Figur 8.1: Ønsket mønster. Forholdet 3-3-4 benyttes siden dette ga godt resultat i tallforholdseksperimentet. Celletype(farge) og posisjon til hver andel bestemmes av evolusjonen.

Et trikolor flaggmønster har kompliserte koblinger mellom celletypene, og alle cellene oppdateres i parallell. Organiseringen av et slik mønster er en kompleks *emergent computation*. Den enkle lokale oppførselen og de enkle lokale bindingene mellom strukturkomponentene i naboskapet for et flaggmønster er i seg selv ikke nok til å frambringe en global struktur. Dette skyldes samtidigheten i utviklingsprosessen der hvert naboskap må forholde seg til andre naboskap i en kompleks parallell oppdateringsprosess. Fitnessfunksjonene for å oppnå dette mønsteret er vist i algoritme 5.3.

8.1 Oppsett av forsøk

Eksperimentet settes opp som vist i figur 8.2 og utføres med de 6 celletypene vist i tabell 8.1. Genomet inneholder 96 gener som i kombinasjon med 6 gentyper i reglene gir et regelsett med 16 regler. Fenotypen er alltid kvadratisk og størrelsen 16 tilsvarer da en organisme med 256 celler. Antall utviklingstrinn er satt til 100 og populasjonsstørrelsen til 10. Forsøkene terminerer alltid for et maksimalt antall generasjoner som i dette eksperimentet er satt til 10000. Arten er satt til *flag* siden et flaggmønster er målet for eksperimentet. Evolusjonen benytter dermed evalueringsprosessen for flaggindivider beskrevet i 5.1.4.

```
============================================================
#. 16x16-100-10-96-flag
============================================================
genome_size:           96    spicies:              flag
rule size:              6    terminator:           generations
phenotype size:        16    initial pattern:      single cell
development steps:    100    initial type:         green
population size:       10    evolution method:     randmut-3
generations:        10000    store genome:         false
                             seed population:      false
------------------------------------------------------------
```

Figur 8.2: Flaggeksperimentet ble utført med dette oppsettet.

Type	Navn	Grafisk
Gr	Green	
Bu	Blue	
R	Red	
B	Black	
W	White	
Z	Empty	

Tabell 8.1: Flaggeksperimentet kunne benytte disse celletypene.

8.2 Resultat

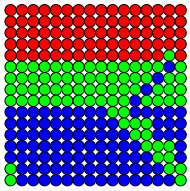
Genotype- og fenotyperesultatet av et suksessfullt forsøk er vist i figur 8.3. Individet er det beste i populasjonen etter 10000 generasjoner med oppsettet vist i 8.2.

Genomet for utviklingsprosessen er vist i figur 8.3(a) og inneholder 7 *growth* regler og 9 *change* regler. Etter endt utvikling oppnås en nesten perfekt *emergent* sjakkbrettstruktur vist i 8.3(b). Figur 8.4 viser et utdrag av de 100 utviklingstrinnene til individet. Individet utvikles fra en enkelt grønn celle til en multicellulær organisme med 256 celler som nesten uttrykker det ønskede flaggmønsteret. Eksempelet viser en nesten perfekt løsning. Fenotypen har ikke en stabil struktur ved utviklingstrinn 99.

Organismen i figur 8.3 endte på 7676 i fitness for sin strukturelle komposisjon (fenotypefitness) hvor 2183 poeng var for tallforholdet mellom celletypene og 1873 poeng for radkoblingene. Regelfitnessen var på 1823 hvor 320 poeng var for de 16 korrekte reglene. Individet ble totalt tildelt en fitnessverdi på 9499.

#	R	C	N	S	E	W
0	●	●	DC	DC	DC	●
1	●	●	●	●	DC	●
2	●	●	●	●	●	●
3	Ge	●	●	●	DC	●
4	●	○	●	●	●	●
5	Gs	●	DC	●	DC	DC
6	●	●	●	DC	●	●
7	●	●	●	●	DC	●
8	●	●	●	●	DC	●
9	●	●	DC	DC	●	●
10	Ge	●	DC	●	DC	●
11	Gw	●	DC	DC	DC	●
12	Gw	●	DC	DC	●	●
13	Gw	●	●	●	DC	DC
14	●	●	●	DC	●	DC
15	Ge	●	DC	DC	●	●

(a) Genotypen.



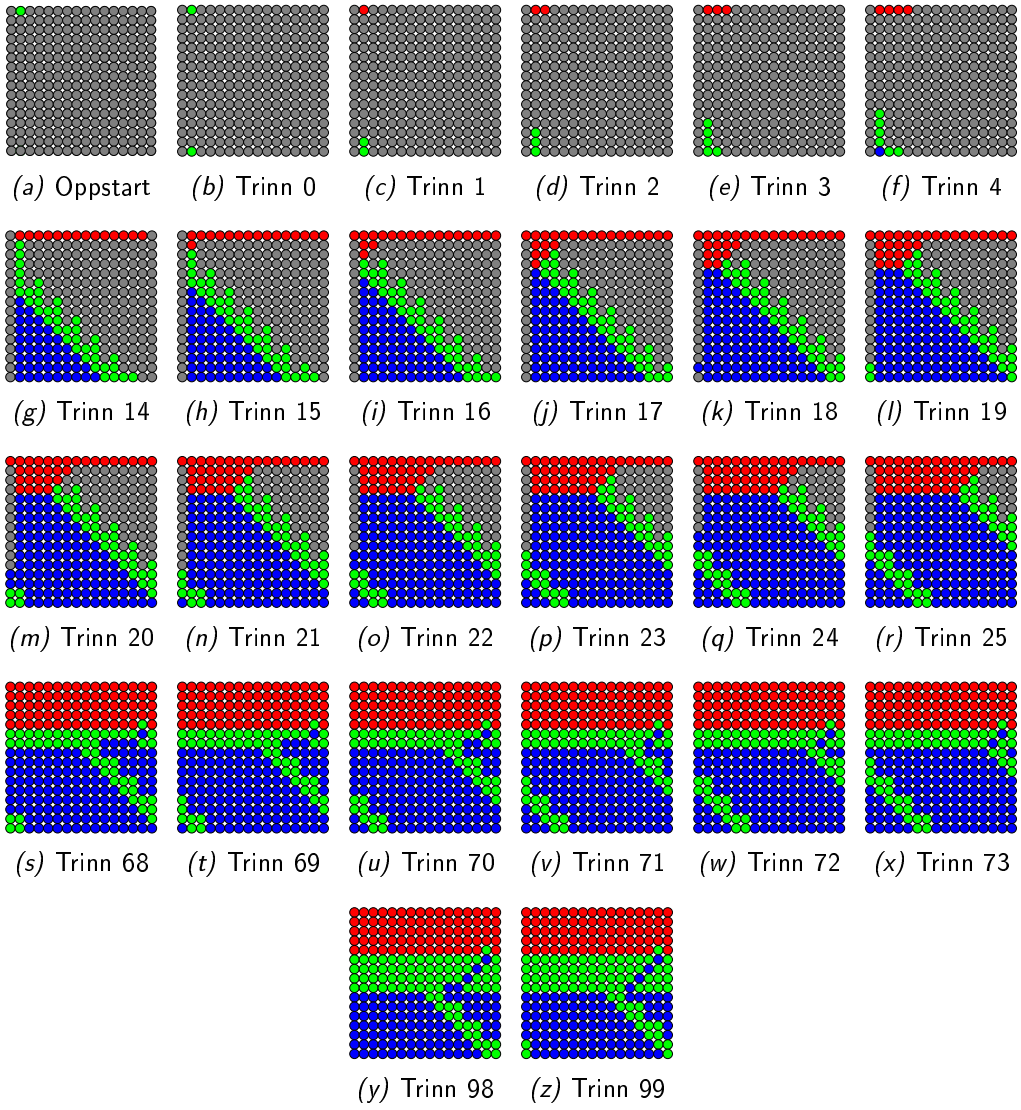
(b) Fenotypen.

Figur 8.3: Det resulterende individet etter utvikling i et av forsøkene. (a) Resulterende regelsett etter tolkningen av genomet. Regel 0 har høyest prioritet. (b) Resulterende struktur etter utvikling av fenotypen.

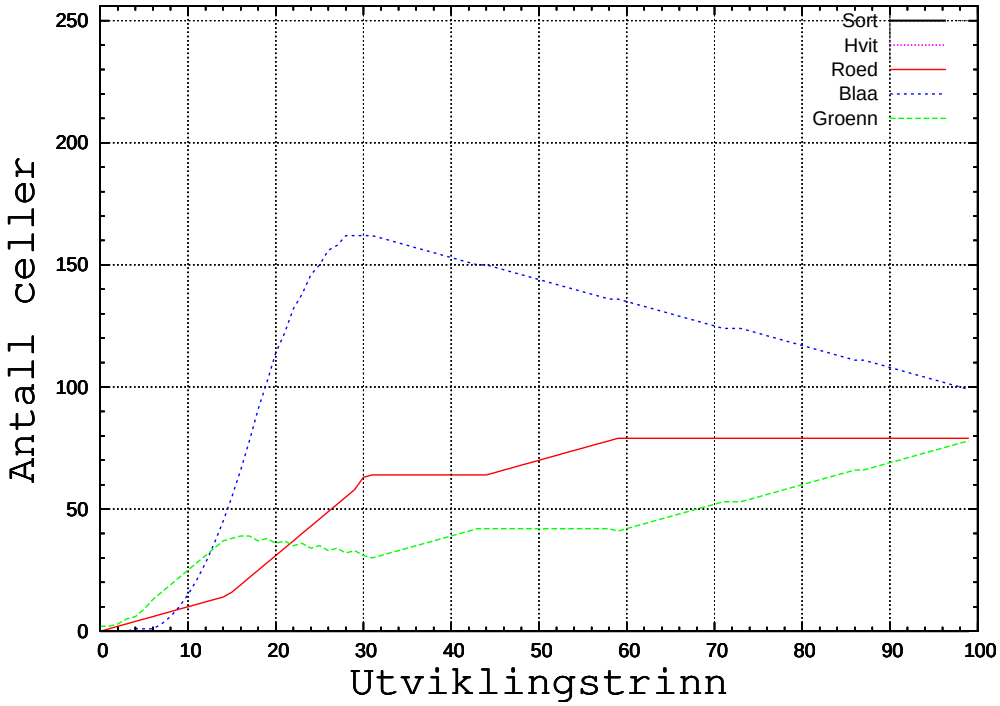
Figur 8.5 viser proliferasjonen (utbredelsen) av celler i fenotypen under utviklingen i figur 8.4. Proliferasjonen av hver enkelt celletype er vist med en egen graf. Graffargen assosiert med hver celletype er gitt øverst i diagrammet. Antallet celler er angitt i den venstre Y-aksen.

I figur 8.6 er genaktiveringsmønsteret vist for genotypen i figur 8.3(a) med genomstørrelse på 16 regler som gir nesten perfekt løsning i siste utviklingstrinn. Figur 8.4 viser utviklingen av fenotypen. Plottet i figur 8.6 viser genaktivieringen (aktive regler) og det totale antallet aktive regler i organismen for hvert utviklingstrinn. Reglene fra 0 til 15 er plassert på den venstre Y-aksen. Hver markering (o) i plottet indikerer at regelen ble aktivert i dette utviklingstrinnet. Den høyre Y-aksen viser antallet celler i organismen med en aktiv regel i det gitte utviklingstrinnet. Antallet aktive regler for hvert utviklingstrinn vises i den plotta grafen.

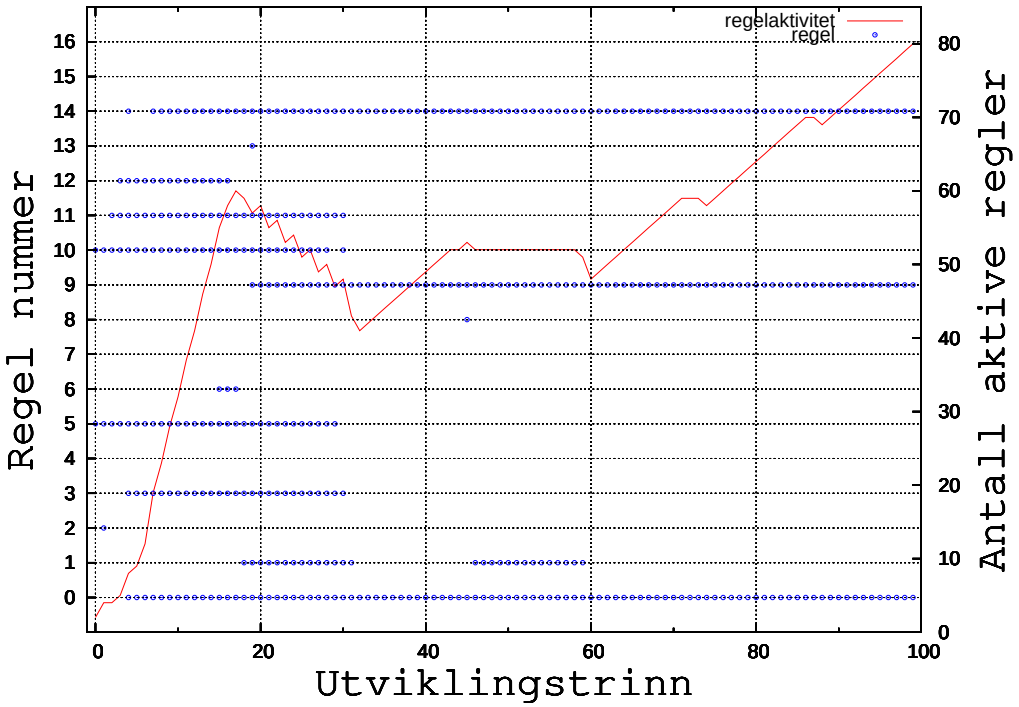
Plottet har et økende antall aktive regler fra første utviklingstrinn til et maksimum på 80 i utviklingstrinn 99. Fenotypen kommer fram til en struktur som fortsatt er i sterk endring ved endt utvikling i trinn 99. Mønsteret stabiliseres altså ikke av den genregulatorisk aktiviteten (aktive regler), men avslutter utviklingen umiddelbart i trinn 99 siden individet har en begrenset utviklingssyklus på 100 trinn.



Figur 8.4: Et utvalg av de 100 utviklingstrinnene til individet i figur 8.3. En animasjon av utviklingen kan sees under Flagg 1 i [19].



Figur 8.5: Proliferasjonen av celler vist for de uttrykte celletypene i fenotypen under utviklingen i 8.4. Proliferasjonen for hver enkelt celletype er vist med en egen farget graf. Graffargen assosiert med hver celletype er gitt øverst i diagrammet. Antallet celler er angitt i den venstre Y-aksen.



Figur 8.6: Regelaktiviteten under utvikling. Reglene fra 0 til 15 er plassert på den venstre Y-aksen. Regel 0 har høyest prioritet. Hver markering (o) i plottet indikerer at regelen ble aktivert i gitt utviklingstrinn. Den høyre Y-aksen viser antallet celler med aktiv regel i gitt utviklingstrinn. Antallet aktive regler er likt antallet aktive celler og vises i den plotta grafen.

8.3 Skalering av resultat

Hvordan evolusjonen utnytter parameterene i systemet for å oppnå gode løsninger er interessant å undersøke. Spesielt vil løsningens avhengighet til fenotypestørrelsen være interessant. Skalerbarheten til løsningen er avhengig av regelsettets generalitet og i hvor stor grad de avhenger av parametere som fenotypestørrelsen og antall utviklingstrinn for å oppnå flaggmønsteret.

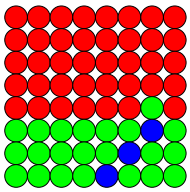
Forsøkene er gjort med det samme genomet og dermed det samme regelsettet som vist i figur 8.3(a). Det første eksperimentet gjøres ved å utvikle individet på en skalert fenotype av 64 celler. Det andre eksperimentet gjøres med 1024 celler.

Oppsettet er som i figur 8.2, men det er kun utviklingsprosessen for individet med genomet fra 8.3(a) som utføres. *Population_size* og *generations* er derfor begge satt til verdien 1, og *seed_population* til *yes* som forklart i 5.2.

Fenotyperesultatet er vist i figur 8.7 og 8.11.

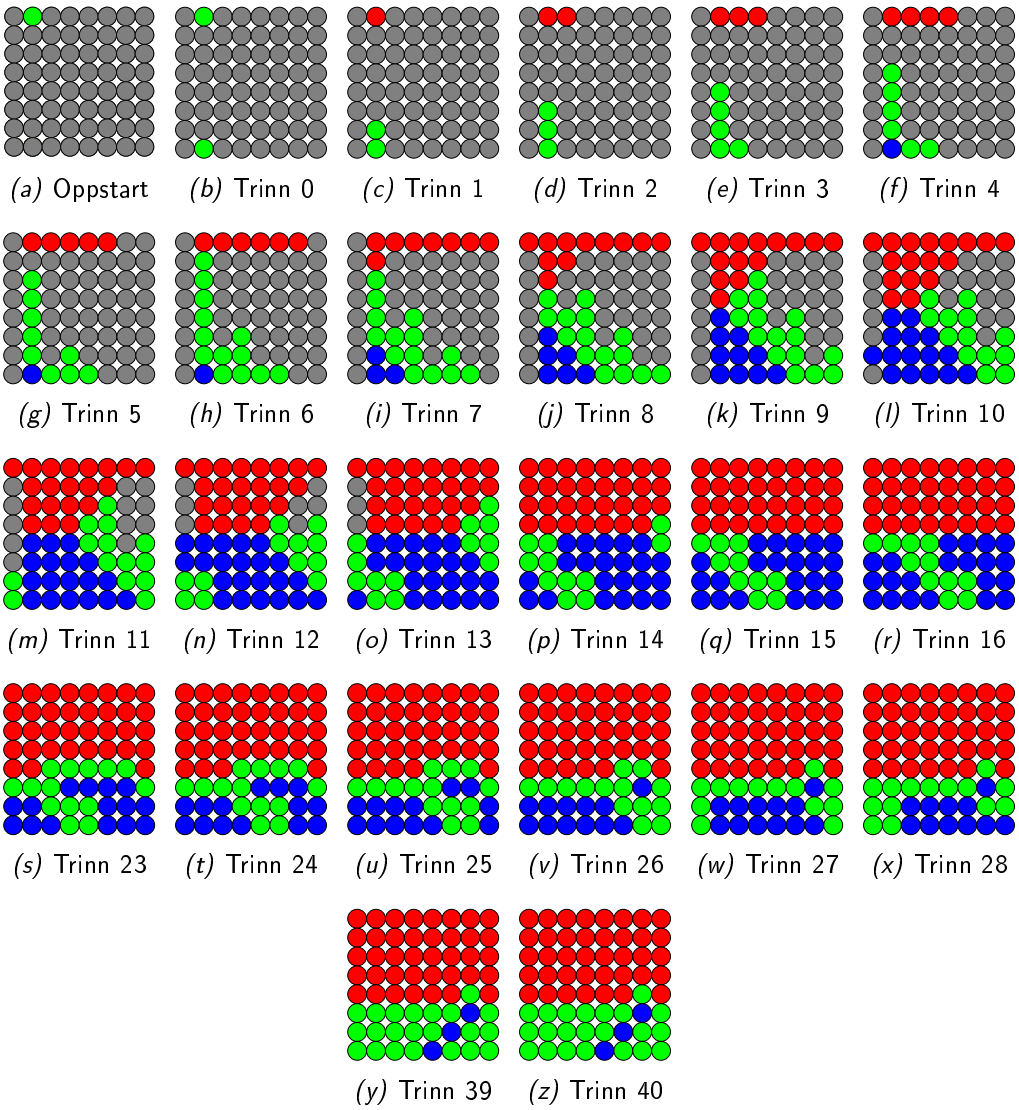
Figur 8.8 viser et utvalg av de 100 utviklingstrinnene til individet utviklet i en 64 cellers fenotype. Den multicellulære organismen har ikke tilsvarende struktur som tilsvarende genom utviklet i en 256 cellers fenotype. Fra utviklingstrinn 38 og til 99 skjer det ingen uttrykte strukturelle endringer i fenotypen.

Figur 8.12 viser et utvalg av de 100 utviklingstrinnene til individet utviklet i en 1024 cellers fenotype. Den multicellulære organismen har ikke tilsvarende struktur som tilsvarende genom utviklet i en 256 cellers fenotype. Fenotypestrukturen er fortsatt i endring ved utviklingstrinn 99.

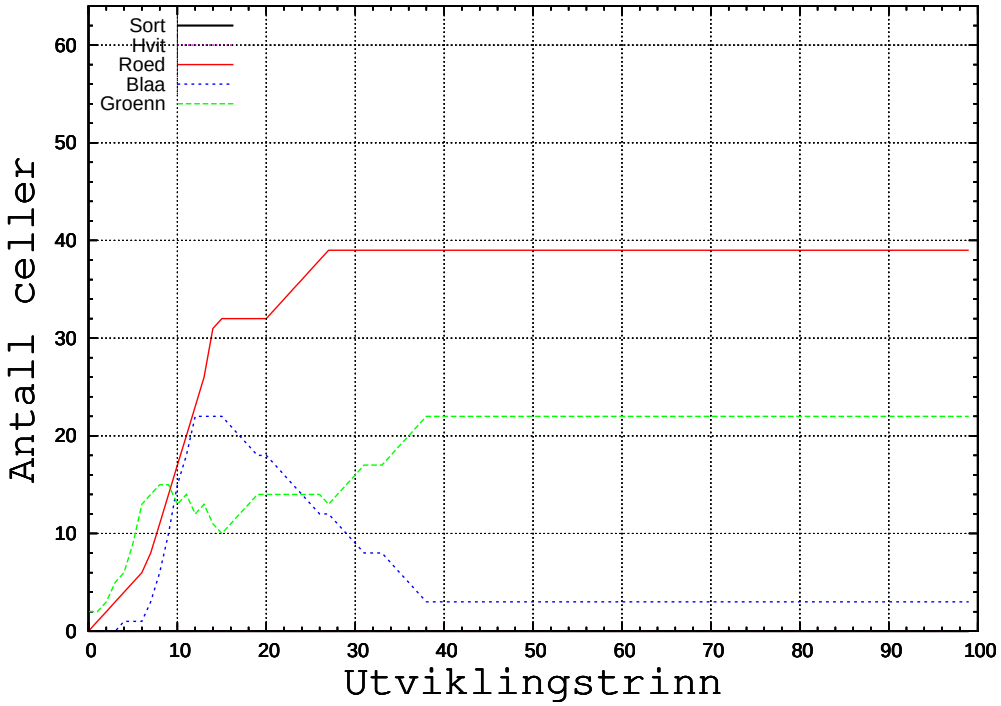


Figur 8.7: Det resulterende individet etter utvikling. Fenotypen er skalert til 8 ganger 8 celler. Individets regelsett fra tolkningen av genomet er identisk med genotypen i figur 8.3(a).

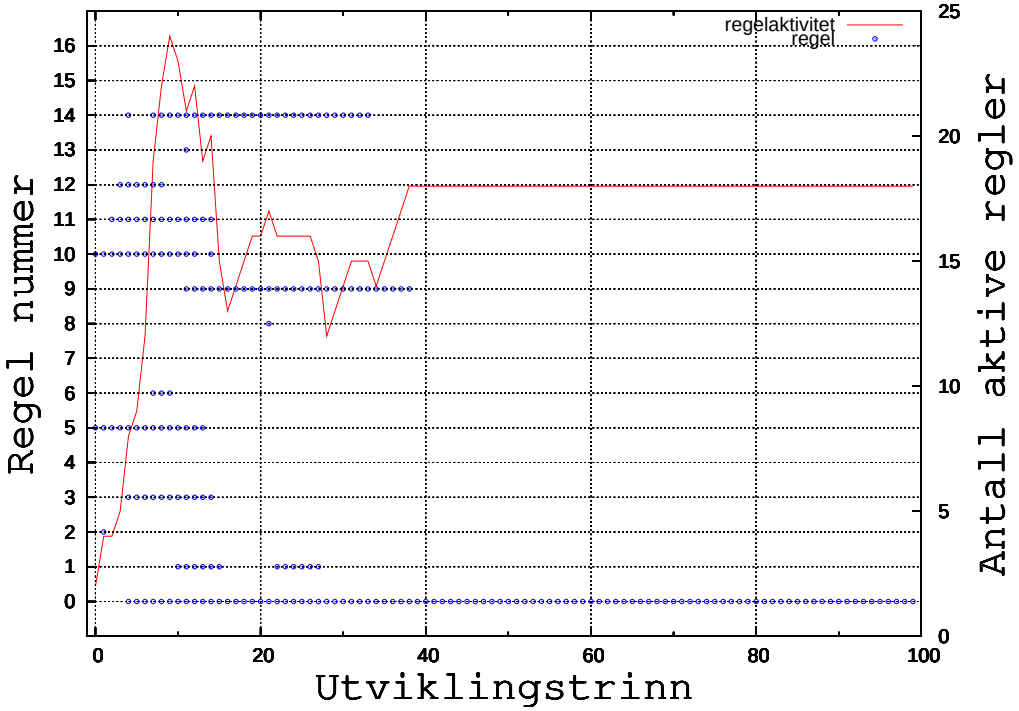
Figur 8.9 viser proliferasjonen (utbredelsen) av celler i fenotypen under utviklingen i figur 8.8. Proliferasjonen av hver enkelt celletype er vist med en egen graf. Graffargen assosiert med hver celletype er gitt øverst i diagrammet. Antallet celler er angitt i den venstre Y-aksen.



Figur 8.8: Et utvalg av de 100 utviklingstrinnene til et individ med identisk genotype som i 8.3(a). Fenotypen er skalert ned til en 8 ganger 8 cellulær matrise for å undersøke reglenes avhengighet til fenotypestørrelsen.



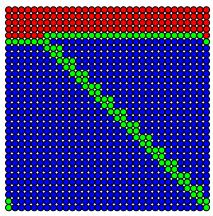
Figur 8.9: Proliferasjonen av celler vist for de uttrykte celletypene i fenotypen under utviklingen i 8.8. Proliferasjonen for hver enkelt celletype er vist med en egen farget graf. Graffargen assosiert med hver celletype er gitt øverst i diagrammet. Antallet celler er angitt i den venstre Y-aksen.



Figur 8.10: Regelaktiviteten under utvikling. Reglene fra 0 til 15 er plassert på den venstre Y-aksen. Regel 0 har høyest prioritet. Hver markering (o) i plottet indikerer at regelen ble aktivert i gitt utviklingstrinn. Den høyre Y-aksen viser antallet celler med aktiv regel i gitt utviklingstrinn. Antallet aktive regler er likt antallet aktive celler og vises i den plotta grafen.

I figur 8.10 er genaktiveringsmønsteret vist for genotypen i figur 8.3(a) med genomstørrelse på 16 regler som ikke gir en god løsning i siste utviklingstrinn. Figur 8.8 viser utviklingen av fenotypen. Plottet i figur 8.10 viser genaktivieringen (aktive regler) og det totale antallet aktive regler i organismen for hvert utviklingstrinn. Reglene fra 0 til 15 er plassert på den venstre Y-aksen. Hver markering (o) i plottet indikerer at regelen ble aktivert i dette utviklingstrinnet. Den høyre Y-aksen viser antallet celler i organismen med en aktiv regel i det gitte utviklingstrinnet. Antallet aktive regler for hvert utviklingstrinn vises i den plotta grafen.

Plottet har et økende antall aktive regler fra første utviklingstrinn til et maksimum på 24 i utviklingstrinn 9. Fra utviklingstrinn 9 synker antallet aktive regler og stabiliserer seg til 18 i utviklingstrinn 38. Fenotypen har da kommet fram til en stabil struktur, men har fortsatt genregulatorisk aktivitet (aktive regler) fra trinn 38 og helt til siste trinn, trinn 99. De aktive reglene i denne perioden uttrykker endringer som bevarer strukturen oppnådd i trinn 38.

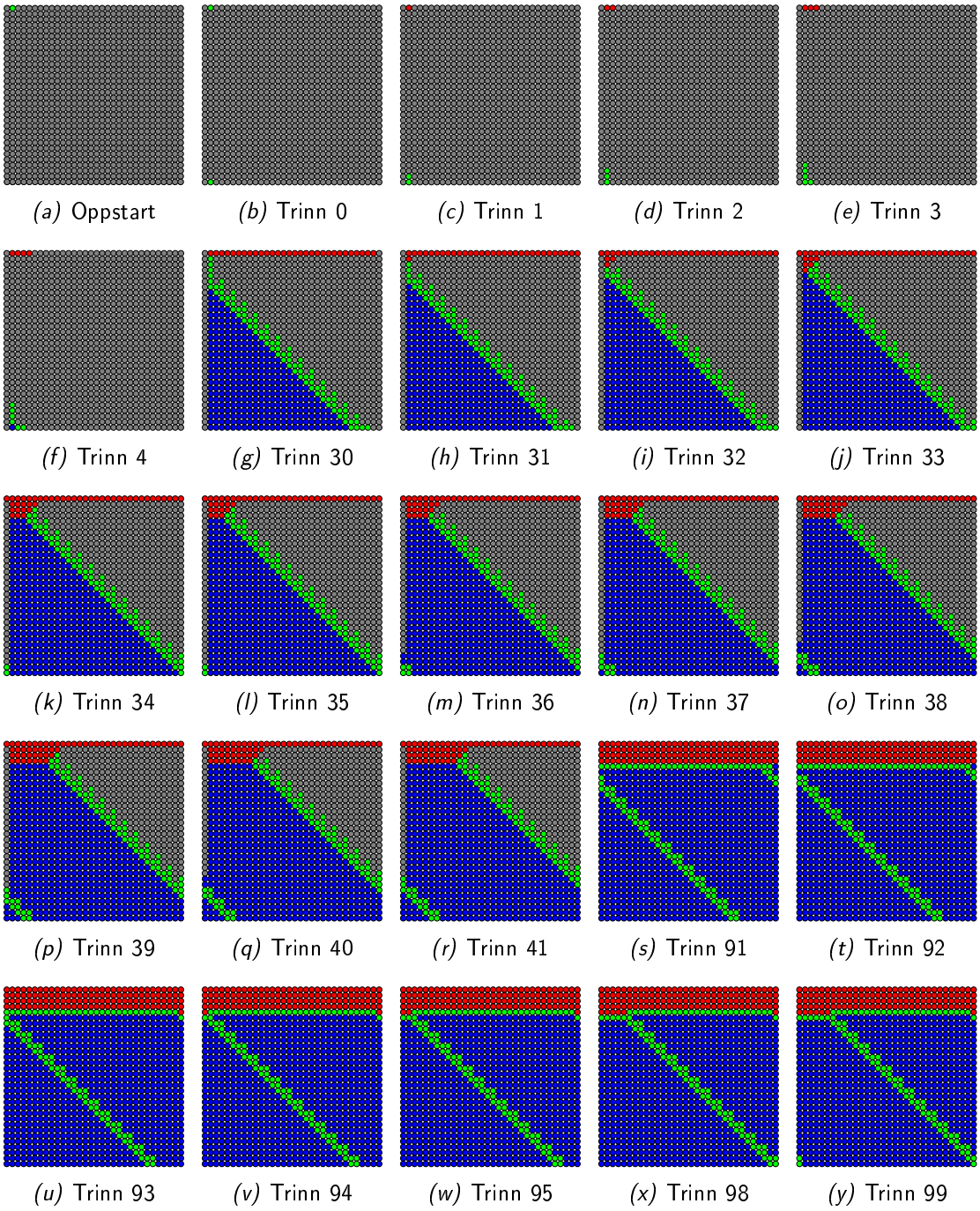


Figur 8.11: Det resulterende individet etter utvikling. Fenotypen er skalert til 16 ganger 16 celler. Individets regelsett fra tolkningen av genomet er identisk med genotypen i figur 8.3(a).

Figur 8.13 viser proliferasjonen (utbredelsen) av celler i fenotypen under utviklingen i figur 8.12. Proliferasjonen av hver enkelt celletype er vist med en egen graf. Graffargen assosiert med hver celletype er gitt øverst i diagrammet. Antallet celler er angitt i den venstre Y-aksen.

I figur 8.14 er genaktiveringsmønsteret vist for genotypen i figur 8.3(a) med genomstørrelse på 16 regler som ikke gir en god løsning i siste utviklingstrinn. Figur 8.12 viser utviklingen av fenotypen. Plottet i figur 8.14 viser genaktivieringen (aktive regler) og det totale antallet aktive regler i organismen for hvert utviklingstrinn. Reglene fra 0 til 15 er plassert på den venstre Y-aksen. Hver markering (o) i plottet indikerer at regelen ble aktivert i dette utviklingstrinnet. Den høyre Y-aksen viser antallet celler i organismen med en aktiv regel i det gitte utviklingstrinnet. Antallet aktive regler for hvert utviklingstrinn vises i den plotta grafen.

Plottet har et økende antall aktive regler fra første utviklingstrinn til et maksimum



(a) Oppstart *(b)* Trinn 0 *(c)* Trinn 1 *(d)* Trinn 2 *(e)* Trinn 3

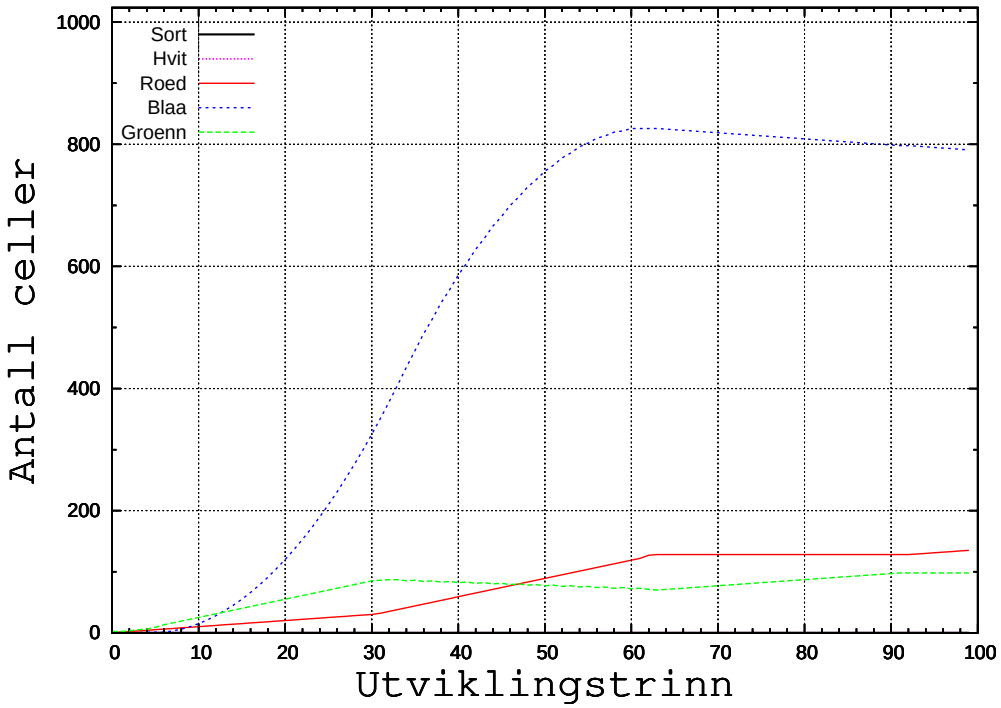
(f) Trinn 4 *(g)* Trinn 30 *(h)* Trinn 31 *(i)* Trinn 32 *(j)* Trinn 33

(k) Trinn 34 *(l)* Trinn 35 *(m)* Trinn 36 *(n)* Trinn 37 *(o)* Trinn 38

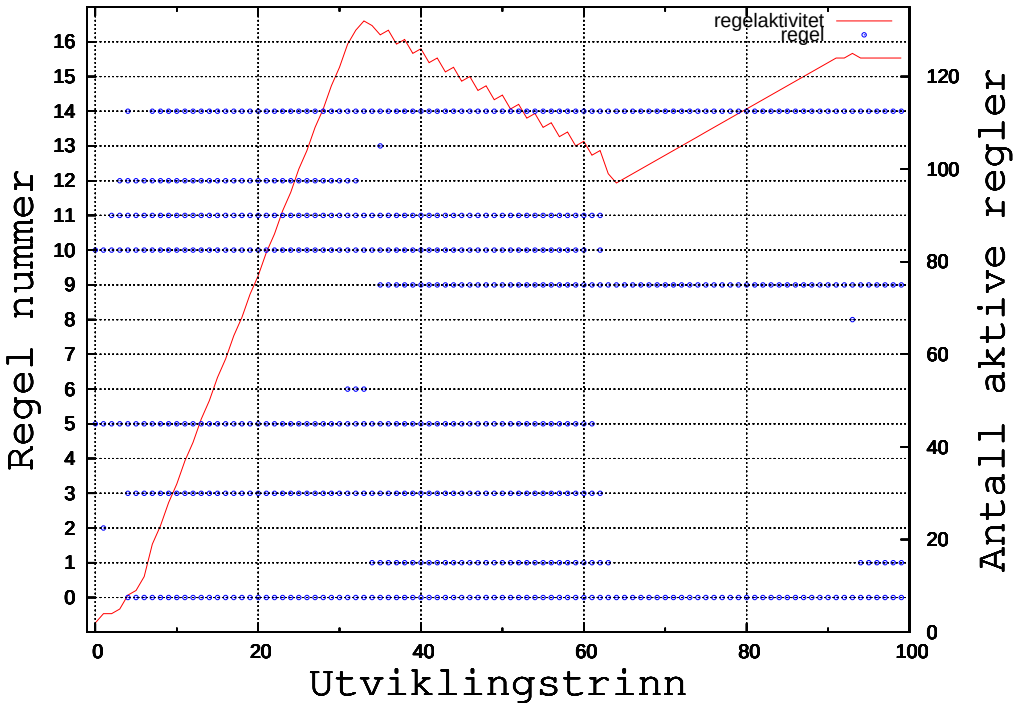
(p) Trinn 39 *(q)* Trinn 40 *(r)* Trinn 41 *(s)* Trinn 91 *(t)* Trinn 92

(u) Trinn 93 *(v)* Trinn 94 *(w)* Trinn 95 *(x)* Trinn 98 *(y)* Trinn 99

Figur 8.12: Et utvalg av de 100 utviklingstrinnene til et individ med identisk genotype som i 8.3(a). Fenotypen er skalert opp til en 16 ganger 16 cellulær matrise for å undersøke reglenes avhengighet til fenotypestørrelsen.



Figur 8.13: Proliferasjonen av celler vist for de uttrykte celletypene i fenotypen under utviklingen i 8.12. Proliferasjonen for hver enkelt celletype er vist med en egen farget graf. Graffargen assosiert med hver celletype er gitt øverst i diagrammet. Antallet celler er angitt i den venstre Y-aksen.



Figur 8.14: Regelaktiviteten under utvikling. Reglene fra 0 til 15 er plassert på den venstre Y-aksen. Regel 0 har høyest prioritet. Hver markering (o) i plottet indikerer at regelen ble aktivert i gitt utviklingstrinn. Den høyre Y-aksen viser antallet celler med aktiv regel i gitt utviklingstrinn. Antallet aktive regler er likt antallet aktive celler og vises i den plotta grafen.

på 131 i utviklingstrinn 33. Fra utviklingstrinn 33 faller antallet aktive regler før det øker igjen opp til det stabiliserer seg på 123 i utviklingstrinn 99. Fenotypen kommer fram til en struktur som fortsatt er i endring ved endt utvikling i trinn 99. Mønsteret stabiliseres altså ikke av den genregulatorisk aktiviteten (aktive regler), men avslutter utviklingen umiddelbart i trinn 99 siden individet har en begrenset utviklingssyklus på 100 trinn.

Kapittel 9

Analyse og diskusjon

I dette kapittelet presenteres analyse og diskusjon av resultatene fra kapittel 6, 7 og 8. Kapittelet avsluttes med en oppsummert analyse og diskusjon av systemet.

9.1 Skalering og sjakkbrett

Resultatet av skaleringsforsøkene viser at den evolusjonære algoritmen delvis opprettholder suksessraten og i noen tilfeller også har fordel av det økte antallet celler i fenotypen. En klar tendens kan ikke sees ut fra statistikken, men resultatet tenderer mer til en positivt eller nøytral endring av SR ved økt antall tilgjengelige celler.

Skaleringen av genotypen viser derimot en tydelig tendens oppsummert i tabell 9.1. Suksessraten øker i samtlige fire forsøk for de tre ulike genotypestørrelsene. Gjennomsnittlig SR øker fra 38% med 4 regler til 60% med 5 regler og til slutt 69% med 6 regler.

Regler	SR(gj.snitt)
4	38%
5	60%
6	69%

Tabell 9.1: Endringen i gjennomsnittlig suksessrate ved skalering av genotypen. Tabellen er en oppsummering av tabell 6.3, tabell 6.2 og tabell 6.4

9.2 Tallforhold eksperiment

Resultatene av tallforholdseksperimentet med 2 celletyper vist i tabell 7.2 indikerer bedre løsninger når andelene i tallforholdet har størst differanse. Den svært høye feilandelen ved 1:1 tallforholdet ble bekreftet gjennom flere repeterte forsøk. Resultatene viste seg å ha tilsvarende vanskeligheter med å stabilisere seg rundt tallforholdet 1:1 med den implementerte fitnessfunksjonen i algoritme 5.3.

Tallforholdsfitnessen for 3 celletyper ble konstruert med utgangspunkt i resultatene fra tallforholdseksperimentene med 2 celletyper. Algoritmen vist i 5.4 kan aldri ha tallforhold med mindre differanse mellom andelene enn i forholdet 2:3.

Det var opp til evolusjonen selv å velge hvilken celletype som skulle utgjøre den størst andelen både for 2 og 3 celletyper. I eksperimentene med 2 celletyper, vist i figur 7.5, har samtlige individer størst andel av hvite celletyper. Tilsvarende tendens finnes ikke i eksperimentene med 3 celletyper. Det er en mulighet for at den initielle sorte celletypen har hatt innvirkning på resultatet.

Det ble i tillegg utført forsøk med 5 celletyper, der kun 3 ble benyttet i fitnessfunksjonen. Dette ga dårligere SR og skyldes antagelig at det økte antallet gentyper bidrar med ugunstige gentyper under tolkning av genom til regler.

9.3 Flaggmønster eksperiment

Et godt flaggmønster ($< 10\%$ feilandel) ble funnet av plattformen i omtrent 2 av 30 forsøk. De gode løsningene utnyttet faktorer som fenotypestørrelsen og antall utviklingstrinn konsekvent for å oppnå best mulig flaggstruktur i siste utviklingstrinn.

Skaleringsforsøket viste tydelig hvor bundet reglene er til fenotypstørrelsen. Evolusjonen kan kanskje klare å produsere et regelsett som vil gi riktig resultat ved ubegrenset antall utviklingstrinn. Resultatene tyder på at det vil være vanskelig å oppnå med evolusjon og utvikling i en fenotype med fast størrelse.

9.4 Systemanalyse

Forsøkene presentert i oppgaven er alle eksperimentelle tilnærminger for plattformimplementasjonen. Det er allikevel mulig å trekke noen konklusjoner.

Resultatene tyder blant annet på at økt informasjonsmengde i systemet kan ha en positiv effekt på det evolusjonære søket. Dette ble observert ved å øke antall gener i genotypen og antall celler i fenotypen. Skaleringsresultatene med sjakkbrettmønsteret viste en tydelig tendens til at økt genotypestørrelse var gunstig for suksessraten.

I flaggeksperimentet ble det også tydelig hvor bunden evolusjonen var av sine omgivelser i form av parametrene fenotypestørrelse og antall utviklingstrinn. Evolusjonen hadde helt tydelig utnyttet fenotypestørrelsen for å klare å lage flaggmønsteret i løpet av 100 utviklingstrinn. I skaleringsforsøkene der individene ble plassert i andre omgivelser for utvikling, ga tilsvarende genotyper ulike uttrykte fenotyper avhengig av omgivelsen, tilgjengelige antall celler.

Eksperimentene viser mangfoldet av løsningsalternativer som evolusjon i et utviklingssystem gir. Løsningsalternativer som oppfyller de samme strukturelle kravene er representert i et variert utvalg av uttrykte fenotyper.

Utfordringen med å balansere fitnessfunskjonen for flaggmønsteret viser hvordan behovet for å definere gode fitnessfunksjoner for å løse stadig mer komplekse problemer øker. Eksperimentresultatene underveis tyder på at evolusjonen stort sett velger enkleste utvei for å oppfylle kriteriene i fitnessfunksjonen. Det blir dermed viktig å kunne finne gode generelle metoder for å definere fitnesskravene opp mot hvert problem en ønsker å løse.

Kapittel 10

Videre arbeid og konklusjon

10.1 Videre arbeid

Arbeidet utført i oppgaven er forskningsarbeid og både fokus og delmål har endret seg etter resultatene underveis. Mulige retninger for videre arbeid med plattformen nevnes her.

Undersøkelse av effektiviteten i det evolusjonære systemet vil være av interesse både med hensyn på økt suksessrate og som eksperimentell utforskning av hvordan nøytrale nett benyttes. Et større antall forsøk med arten *simple* for å undersøke SRen opp mot for eksempel arten *neutral* kan være et første steg. Resultatene kan gi grunnlag for utbedringer av implementasjonen eller for videre eksperimenter med alternative løsninger for nøytrale nett.

Eksperimentet med flaggstruktur viste tydelig behovet av å kunne definere og balansere en god fitnessfunksjon. Konsekvensene av kriteriene i fitnessfunksjonen burde visualiseres. En tydeliggjøring av sammenhengen mellom kriterienes vekting i fitnessfunksjonen og parametrene i systemet kan gi et bedre grunnlag for å definere fintessfunksjoner for mer komplekse løsninger.

Eksperimentet med flaggmønsteret med økt antall generasjoner og genotypestørrelse kombinert med en mindre suksessrate tydeliggjør behovet for økt beregningskapasitet. En utvidelse av plattformen mot maskinvare slik at de tyngste beregningene kan sendes direkte til maskinvareplattformen for utvikling kan være en løsning. Det vil da være viktig å utrede hvilke eksperimenter eller deler av eksperimenter det er ønskelig å simulere i programvare. Det vil typisk være deler

hvor informasjonen under forsøket vanskelig lar seg hente ut fra maskinvareplattformen. En utvidelse av plattformen som kan være aktuell med en slik løsning er implementasjon av funksjonalitet i utviklingssystemet.

10.2 Konklusjon

I oppgaven ble en plattform for å eksperimentere med evolusjon for et utviklingssystem implementert i C++ programmet Exploreed. Gjennom en rekke eksperimenter ble plattformens funksjonalitet forbedret og utvidet i takt med de eksperimentelle oppgavene. Resultatet er eksperimentene presentert i denne hovedoppgaven og plattformens kildekode vedlagt digitalt.

Plattformen muliggjør omfattende eksperimentell utforskning av evolusjon for utvikling. Det er mulig å eksperimentere med en rekke systemparametere for å få en eksperimentell forståelse av hvordan disse påvirker systemet.

Nytten av plattformen er vist i eksperimentene i denne oppgaven. Resultatene viser at evolusjonen klarer å skape multicellulære *emergent* strukturer gjennom *emergent computation* i en cellulære beregningsmaskin.

Plattformen har muliggjort *emergent computation* av et sjakkbrettmønster og et trikolor flaggmønster. Resultatene fra disse prosessene er presentert i visuelle grafer og animasjoner fra forsøkenes loggdata. Disse resultatene gir grunnlag for analyse og videre eksperimentelt arbeid med plattformen.

Resultatene tyder blant annet på at økt informasjonsmengde i systemet kan ha en positiv effekt på det evolusjonære søket. Dette ble observert ved økning av antall gener i genotypen og antall celler i fenotypen.

I flaggeksperimentet ble det også tydelig hvor bunden evolusjonen var av sine omgivelser som parametrene fenotypestørrelse og antall utviklingstrinn.

Eksperimentene med tallforhold viser mangfoldet av løsningsalternativer som oppfyller de samme strukturelle kravene.

Hovedmålet med oppgaven var å lage et program som kunne benyttes i ulike evolusjonære parametere og metoders innvirkning på kunstig utvikling. Programmet har gjennom resultatene vist seg som en god plattform for å utvikle cellulære strukturer.

Tillegg A

Konfigurasjonsfil

sjakkoppsett.conf

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Setup and configuration file for Exploreed -->
<config>
   <run>
       <exploration_name>8x8-100-5-30-neutral</exploration_name>
       <priority value="1"/>
       <runs value="1"/>
       <output value="5"/>
       <log_level value="1"/>
   </run>
   <exploration>
      <name>8x8-100-5-30-neutral</name>
      <genome_size value="30"/>
      <rule_size value="6"/>
      <species>neutral</species>
      <phenotype_size value="8"/>
      <development_steps value="100"/>
      <population_size value="5"/>
      <generations value="3000"/>
      <terminator>generations</terminator>
      <initial_pattern>single cell</initial_pattern>
      <initial_type>black</initial_type>
      <evolution_method mutation_type="random" mutation_rate="3"/>
      <store_genome>no</store_genome>
      <seed_population>no</seed_population>
   </exploration>
</config>
```

Forsøket settes i gang med kommandoen[1]:

```
./exploreed -c sjakkoppsett.conf
```

[1]Plattformen har en minnelekasje, så ved større eksperimenter er det mer effektivt å starte med *runs* lik 1 og kun en *run* oppføring. For å utføre flere repetisjoner av et forsøk startes programmet slik: **a=10;while test $a -gt 0;do echo $((a–));./exploreed -c sjakkoppsett.conf;done** der **a** er antall repetisjoner av forsøket.

Tillegg B

Artikkel

Artikkelen med resultatene fra skaleringseksperimentene i kapittel 6 ble skrevet i sammarbeid med min veileder Gunnar Tufte. Resultatene fra eksperimentene på våre ulike plattformer var sammenfallende og viste at økt informasjon kan ha positiv innvirking på resultatet av utviklingsprosessen. Det kan også se ut til at innvirkningen kan være uavhengig av valget av evolusjonær algoritme, da de to systemene har ulike EAer.

Artikkelen ble akseptert til Codesoar workshoppen under GECCO konferansen 2006 der den vil bli presentert 9. juli 2006.

Errata til artikkelen:

I *Figur 6* skal **Step 4** inneholde 2 sorte celletyper til i celle(7,4) og celle(7,6) som vist i 6.4(e).

Regel 0 i **Table2** er feil og skal ha celletypen W (hvit) i **Result** gentypen tilsvarende regel 4 i figur 6.3(a).

Size Matters: Scaling of Organisms and Genomes for Development of Emergent Structures

Gunnar Tufte
Norwegian University of Science and Technology
Department of Computer and Information Science
Sem Selandsvei 7-9 7491 Trondheim Norway
gunnart@idi.ntnu.no

Joacim Thomassen
Norwegian University of Science and Technology
Department of Computer and Information Science
Sem Selandsvei 7-9 7491 Trondheim Norway
joacimt@stud.ntnu.no

ABSTRACT

An articial Development approach aimed at development of electronic circuits has functional circuits as the end product. Functionality of circuits are given by the topology of the phenotype. The article investigates if the iterative processing of the genome in a development process can take advantage of the information provided in the genome and the information available in the emerging phenotype. The development process described is a rule–based system on a non–uniform Cellular Automata topology. The experiments presented investigate scaling of available resources in the phenotype for evolution of structural properties and genome scaling for expressing functionality in the structure of the emerging phenotype.

Categories and Subject Descriptors

B.6.1 [Hardware]: Design Styles| Cellular arrays and automata

General Terms

Design

Keywords

Development, Cellular Computation, Scaling

1. INTRODUCTION

In Evolvable Hardware (EHW), Evolutionary Algorithms (EAs) are used to evolve electronic circuits. In general, a one–to–one mapping for the genotype–phenotype transition is assumed. Use of a one–to–one mapping assumes a genotype consisting of a complete blueprint of the phenotype. For an electronic circuit, the genotype thus completely describes the circuit topology i.e. the components and their interconnections. This description may be said to be a structural description. Interpretation of this structural information provides us with the circuit functionality. Electronic circuits may be said to be complex due to their complex structure and/or complex functionality.

The approach of evolutionary design [3] may be an alternative to today's view on how circuits and computation machines are designed and operates. If an alternative design approach is to be used it may be that today's view on a machine's components and architecture constrains the design approach [16].

The Cellular Computation paradigm [21] may be an alternative architecture. Cellular Computation oer a theoretical platform [20] realisable in today's hardware technology [27] exploiting the principles of a global behaviour emerging from local cell interactions in a parallel architecture.

The massive parallel computation power of cellular computation is hard to exploit using traditional design and programming methods. Moving away from a traditional approach towards adaptive methods [21], includes systems where the programmer (or designer) can not explicitly specify the complete system [20]. An adaptive approach, e.g. EAs, may be suited but constraints of the algorithm itself can be added into the design process. EAs are resource–greedy accompanied by direct mapping and often suer from a scaling problem [4].

One solution to the resource greedy nature of EAs is to follow nature's example and shrink the genotype in some way. Natures way of handling complexity clearly points in the direction of a non one–to–one mapping from genotype to phenotype. Natures process of development where a zygote develop to a multicellular organism [30] can be included in an EA to increase scalability [2].

In biological development, an initial unit | a cell, holds the complete building plan (DNA). It is important to note that this plan is generative | it describes how to build the system, not what the system will look like. Units have internal state, can communicate locally, can move, spawn other units or die. Groups of units may also exhibit group–wise behaviour i.e. a group state.

The global developmental stages from the zygote (fertilised egg) to the multicellular organism, although interdependent and not strictly sequential, may be categorised as pattern formation; morphogenesis; cell dierentiation and growth [30].

Evaluation of individuals in a developmental system is usually based on properties of the phenotype i.e. the nal developed phenotype or the emerging phenotype. In contrast to a one–to–one mapping where the phenotype may be

GECCO'06, July 812, 2006, Seattle, Washington, USA.

identical to the genotype [12]. The genotype in a development system may only provide information regarding the phenotype or it's functionality as the genome is processed by the development process.

For development of structures e.g. ags [17], a graphical representation of a ag is the output produced. Evaluation is based on structural properties of the phenotype given by cells representing structural parts of a global property e.g. a desired pattern of cells expressing colours. The development processin these ag examples have built a structure out of cell interactions i.e. the global structure emergesout of local interactions.

Development of functionality i.e. a phenotype structure capable of computation, can use an evaluation based on the behaviour of the emerging phenotype. The Cellular Computation paradigm [21] used herein oer massive parallel computation power in a cellular array. The emergent phenotype consistsof computation elements in a cellular array. As such, the evaluation can be based on the functional properties expressedin the cellular array i.e. cellular computation [23].

As stated above the processing of genomesin a development system makesit quite dieren t from a system using a one−to−onemapping. The processing of the genome in the development processin the cellular development approach used herein is basedon generegulation [24]. Gene regulation implies that dieren t parts of the genome are expressedin dieren t cells at dieren t time in the emerging phenotype. The phenotype emergesas a result of an interplay between the genome and the emerging phenotype i.e. a process of gene regulation.

Way back in the work of Von Neumann [29] it is shown that the number of available cell types and the number of available construction instructions inuence on what cellular structure that can be built. In the work of Sipper [20] the number of available cell typesrequired for general computation is investigated for the purposeof cellular machines. Inspired by such approacheswe want to investigate how scaling of available information inuence on evolution of developmental genomes.

Available information is herein scaled by two parameters. First, the available size of the phenotype i.e. the number of cells that can be exploited by the growth and dieren tiation process. Second, the available number of genesin the gene regulation network. The scaling of genome i.e. number of genes, changes the available actions a cell can express and the number of regulation criteria the cell can interpret.

To explore the inuence of these two scaling parameters two very dieren t approaches are presented. However both uses the same cellular development process. Two dieren t EAs are used. An Evolutionary Strategy (ES) is used in experimental investigation of the scaling of available cells i.e. phenotype scaling. A Genetic Algorithm (GA) is used to investigate the inuence of available genes i.e. genome scaling.

The two experiments presented targets evolution of genomes for structural properties and functional properties respectiv e. The goal of presenting two so dieren t approaches is to show that they have interesting concurrent results that appear independent of the EA included in the system.

The developmental model used in both experiments is based on cellular development. The model include a cell with functional components in addition to the required genome and development mechanisms. As stated the main long term goal is functional circuits. As such, functional cell components are required to expressfunctionality. Expressing functionality adds an extra processto the developmental system. In addition to the processingof the developmental genometo generate the emerging organism the functional components of the emerging organism must be processed.

The resourcesrequired to processthe developmental genome as to generate the emerging phenotype usually depend on the size of the genomeand the amount of development steps. The processingof the genomemay be a sequential or parallel process. To express cells functionality the cell's functional components must also be processed. As such, the amount of computational power required in each cell depends on the processing of the genome and the processing necessaryto express the cell's functionality. The total amount of computational power is given by the size of the cellular array, i.e. the computational power required by the total number of cells.

Herein the functional components of the cell are only processedin the experiment of functional properties. The functionality is evaluated on the nal development step. However, the functionality expressedin the organism at the − nal development step arises as a result of processing of the functional components on all available development steps. To be able to carry out experiments including processingof the functional components and the desired genome size in realistic time a hardware experimental platform has been implemented. The hardware platform oer true parallel computation for expressingthe functionality of the cells, i.e. the processingtime is not inuenced by the amount of cells used. The processingof the genomeis a partly processedin parallel oering a speed−upmaking it possibility to scaleup the genomesize and still be able to perform experiments in realistic time for the desired amount of cells.

The speed−upoered by the hardware platform is necessary for the experiments regarding functional properties. In the experiments regarding structural properties the genome size is much smaller and the functional components of the cells are not processed. As such, a software solution is possible. The reason for using both software and a hardware platform in experiments is to be able to exploit the raw power of hardware when necessaryand to be able to take advantage of the exibility oered by software. Flexibility here relates to the ability to exible monitoring of internal processesof the development process. However, running a software solution is constrained by increased computation time for large genomes,simulation of functional components and the amount of possible cells.

The article is laid out asfollows: The cellular developmental model is presented in Section 2. Section 3 describes the ES used and the experiments for phenotype scaling. The GA used and experimental results for genome scaling are presented in Section 4. A discussion of the experiments are given in section 5. Finally, Section 6 concludes the work.

2. DEVELOPMENT MODEL

Including properties of biological development asdescribed above into artificial development do not need to achieve a realistic model of development but are rather used to increase the power of evolutionary computation or Evolvable hardware (EHW) [6, 14, 18, 26, 28]. However, artificial development may also be used in studies by biologists as demonstrated by Kumar's [10] computational models of develop−

ment based on, e.g. real biochemical pathways. The complex 3D shape and form in Kumar's experiments, e.g. [11], illustrate evolution together with development which achieve complex structures.

Today's electronic circuits are based on 2D silicon technology. As such, it is possible and desirable to limit the complexity of the shape and form of the target for development. In [6, 26] a 2D circuit close to the internal architecture of FPGA devices are used as the target for development of electronic circuits. Both approaches uses a intracellular communication restricted to a 2D Von Neumann neighbourhood. However, the architecture and computation paradigme used is quite dieren t. In Gordons work [8] successful evolution of adders and parity problems was demonstrated. The architecture of the circuit and computation performed may be close to a traditional circuit even if the circuit solutions found are not traditional adder designs. The architecture of the developed circuit consists of combinatorial function generators connected by wiring. As such, the circuit is a combinatorial circuit consisting of inputs, computational elements and outputs. The work herein is based on the development model in [26] using a dieren t computational paradigm, i.e. cellular computation. The computational elements, i.e. functional components of the cell, are a sequential circuit including memory and combinatorial logic. The architecture of the developmental circuits is based on the cellular computation paradigm. The circuits functionality is an emergent property. As such, the work of Gordon targets development of combinatorial circuits the work herein is aimed towards development of emergent sequential circuits.

In [7] Gordon applies the developmental model to the task of evolving pattern to demonstrate scalability. The results show that the development model for electronic circuits also can be applied to develop patterns. Herein a similar approach is taken. The development model is applied to the task of development of a specic goal pattern. However, the possible cell types are restricted to include only the required types i.e. the number of exploitable cell states (given by the cell types) are at a minimum. In contrast the possible cell types (or cell states given by the possible protein concentration) in [7] are not constrained as the goal is scaling of the phenotype itself. Scaling herein investigate scaling of available resources in the phenotype for evolution of structural properties.

Figure 1 shows the building block of the developmental system presented herein | the cell. The cell is divided into three parts: the genome (the building plan); the development process (mechanisms for cell growth and dieren tiation) and the functional component of the cell. The genome consists of rules of how to construct the multicellular organism i.e. a cellular array of functional components.

The genome is based on two types of rules i.e. change and growth rules. Cell growth is a mechanism to expand the organism and dieren tiation changes a cell's type i.e. functionality. These rules are restricted to expressions consisting of the type of the target cell and the types of the cells in its von Neumann neighbourhood. The rules consist of a result and a condition. The result part of a change rule states the type of cell the target is going to be changed into. The conditional part describes the type of neighbourhood cells to trigger the rule. A growth rule result gives the direction of growth; grow from north Gn , east Ge , south Gs or west Gw . Rules have the following valid conditions: valid cell

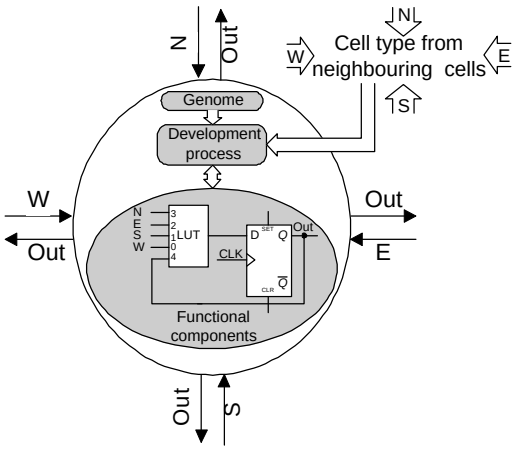


Figure 1: Components of the cellular Development Model.

types, don't care (DC), or empty. DC is not a valid target condition.

Change rules have one restriction: a target cell can not be changed from an empty to a valid non-empty cell type. The reason behind this restriction is that we want growth to handle the expansion of the organism. Growth rules have two restrictions. First, the target cell must be empty { this is to prevent growing over an existing cell and, therefore, specialising the cell with a new cell type. Secondly, the cell to be copied into the target can not be empty.

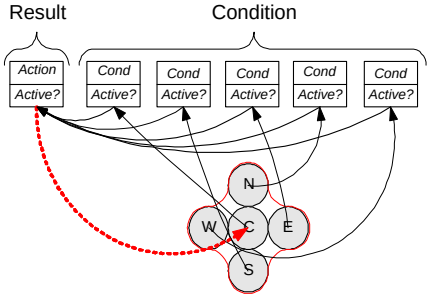


Figure 2: Regulation of genes as interplay between the genome and the emerging phenotype.

Firing of a rule can cause the target cell to change type, die (implemented as a change of type) or cause another cell to grow into it. The current cell together with the neighbouring cells control whether a rule is to be red or not. Figure 2 illustrates the process of evaluating a rule. For each cell condition, the cell type is compared and if the condition is true then that part of the rule is active. If all conditions are active then the result will become active and the rule will re. Activation of the result gene is expressed in the emerging phenotype according to the action specied in the result.

In a development genome multiple rules are present. Multiple rules imply that more than one rule of a given cell may

be activated at the same time if their conditions hold. To ensure unambiguous rule ring, rule regulation is part of the development process. If the rst rule is activated, the second and third rule can not be activated. Activ ation of the second rule prevent activation of the third rule, etc i.e a gene regulation network.

The development process presented is an autonomous process. All cells in the cellular array can run the development process in parallel. The functional components of the cells are also a parallel architecture. As such parallel cellular development starting from a single cell can develop to a multicellular functional organism i.e. a cellular array constructed of dieren t cell types.

The functional components of the cell is an Sblock [9]. The context of the Sblock's look-up table (LUT) denes it's functionalit y and is herein also used to dene Sblocks as cell types based on their functionalit y. The LUT is the combinatorial component and the ip-op is the memory element capable of storing the cells state. The output value of an Sblock is synchronously updated and sent to all its four neighbours | its Von Neumann neighbourhood, and as a feedback to itself.

One update of the cell's type given by the genome and the execution of the development process is termed a development step (DS). A development step is a synchronous update of all cells in the cellular array. The update of the cells functional components i.e. one clock pulse on the ip-op, is termed a state step (SS).

The POE-model [19] has been established as a taxonomy of biological inspired hardware. The POE-model classies biological inspiration in the design of computing machines along three axes: phylogeny, ontogeny, and epigenesis. The phylogeny axis encompasses evolution. Ontogeny embraces systems taking advantage of inspiration from biological development. Systems capable of acquiring and exploiting information i.e. learning, are placed along the epigenesis axis. A common property for designs exploiting ontogeny and epigenesis, is the abilit y to shrink the level of information needed to form a large complex organism [28].

A possible approach to hardware development of cell based circuits is to include the genome, functional units and development process in each cell [14, 15]. In the POEtic project the goal is to include all of the three axes of the POE-model in digital hardware. Others have included the evolutionary process in the hardware platform [13] targeting fault-tolerance.

Another approach is to simplify each cell, in keeping with the properties of cellular architectures, by removing the genome and development process from the cell itself [25] thus increasing the parallelism available to the functional parts of the cell. The principle of a common genome is still retained as all the cells' development actions are controlled by the same shared genome even if it is not stored in every cell.

Figure 3 illustrates the hardware implementation of the cells with a centralised genome and development process used in section 4. The xed partition consists of a communication module (COM) for external communication and genome download. The CTRL module manages the other modules. The dieren t Cell types, i.e. LUT denitions are stored in the Sblock denition module. The genome is stored in the Rule Memory. The Development Process updates the emerging phenotype based on the properties of the emerging phenotype and the genome. CONFIG is the interface

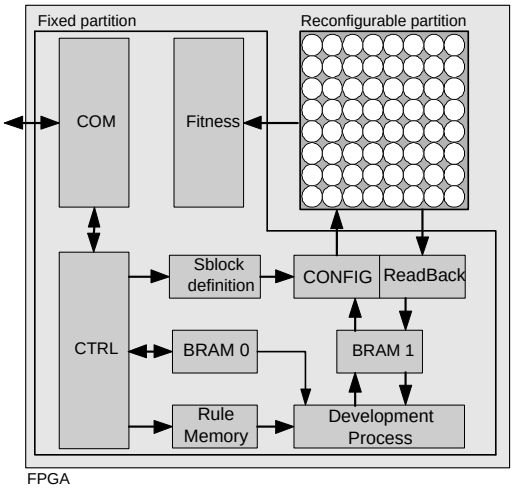


Figure 3: Dev elopmen t pro cess in tegrated on-chip together with an Sblo ck arra y.

to update the phenotype in the recongurable partition of the FPGA. The ReadBack module can be used to collect information from the phenotype. The emerging phenotype develops in the recongurable partition. Each circle include the functional components of the cell shown in Figure 1. The Fitness module can be used to implement hardware tness functions [1]. A detailed description of the modules can be found in [25].

3. PHENOTYPE SCALING

The objective is to evolve genomes that can produce an emerging phenotype using structural properties of the phenotype as the main evaluation criteria. The structural property chosen is development of a predened pattern expressed by the cell types in the nalised phenotype i.e. at the last available development step.

The target pattern is a chessboard. To represent the chessboard only two types of cells are required i.e. cell type is expressed as black or white. The development process starts from a single cell and develops to an organism of size given by the predened maximum phenotype size. In the initial condition for development all cells but the rst single cell is dened as empty.

The chessboard pattern is highly scalable and can provide a good foundation for exploration of the evolution and development processes.

In this experiment the main scaling parameter investigated is the phenotype size i.e. organism. As such, the number of available cells in the phenotype is scaled while keeping the genotype size constant. The experiments are repeated for an increasing genotype size as an attempt to relate the results to the genome scaling for evolution of functional properties presented in section 4.

The development process used is the development model presented in section 2. Even though a hardware implementation is available the experiments in section 3 are carried out on a software platform. The experiments only take advantage of the cell types. The functional components of the

cell in Figure 1 are not implemented. However, the cells expresstheir type as an intercellular property. In an evaluation only targeting structural properties the cell type itself is enough to expressa cellular building block in the phenotype structure.

3.1 Evolutionary Algorithm

The evolutionary strategy chosenincorporates survivor selection with elitism. Only cloning and mutation are used as genetic operators. The population size is constant and only the best individual is cloned to next generation. If there are two or more best individuals equally t the newest one is preferred. The rest of the individuals are deleted and the new population is lled with mutants of the best. All individuals are re-evaluated in each generation.

If an individual is mutated the mutation operator can change one, two or three genes in the genome. The mutation points are chosen randomly. A gene at a mutation point is assigned a new random value. This might result in no change.

Table 1: Dened Cell types.

Cell Type	Cell name	Graphical representation
B	Black	●
W	White	○
Z	Empty	◍

Table 1 list the available cells and their graphical representation used herein. The only cell types available are the empty, black and white. In addition to the given cell types the growth directions and DC condition are valid genetypes in a rule.

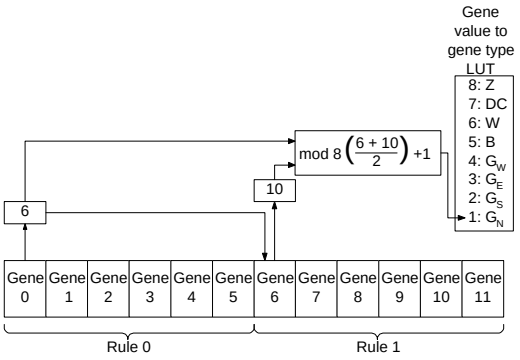


Figure 4: Mapping of gene values to gene types.

The genome is a symbolic representation of positive integers. Each rule is indirectly represented by six genes in sequence. The representation of each gene type i.e. B, W, Z, DC and growth direction, is not direct. Instead each gene value represents an address to another gene in the genome.

Figure 4 illustrates the process of mapping the positive integer gene representation in the EA to the gene type representation used in the development genomes. In the example a two rule genome is shown. The integer representation represent each gene by a value from 0 to the number of available genes. Here 0 to 11. Each gene value is mapped to a gene type in the following manner: The gene value, here 6 for the rst gene, points to gene number 6. Gene number 6 has the gene value 10. The gene type for the rst gene is found by adding 1 to mod 8 of the average of the sum of the two genesgene values. The result is used as an address to point in to a gene value to gene type LUT. As shown the gene type for the rst gene in the rst rule is a growth condition (Gn).

The genotype representation used in the EA have a neutral mapping [5] to genomes that can be processed by the development processdescribed in section 2. The selection of the newest individual if equally high tness is obtained can exploit neutral jumps in the phenotype space [5].

3.2 Evaluation

As stated in section 1 evaluation in a development processdepends on the interplay between the processing of the genome in the cells and the emerging phenotype. As such, the development processrequire genomescapable of growth and dieren tiation to provide feedback to the EA. As an attempt to provide genomescontaining valid rules to the development processvalid gene entry in rules are rewarded | a rule tness. To avoid waste of computation time only rule tness is calculated for genomes containing no valid rules i.e. genomesare not processed by the development process only rule tness points are given.

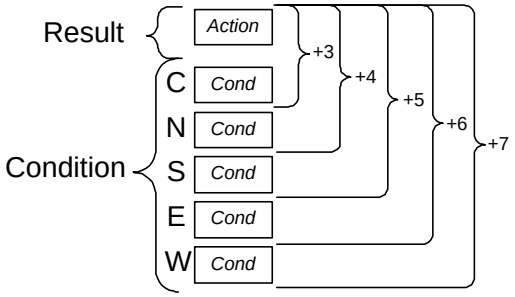


Figure 5: Fitness points based on correct composition of genes in a rule.

The rule tness is a function based on three properties of each rule in the genome. First, each gene is given a reward independent of rule composition. That is, if a gene entry is valid i.e. no DC in the action and no growth entry in the conditional part. Second,the rule is given an increasing number of points based on the correctness of the rule composition i.e. the restrictions for growth and change rules. Third, a valid rule is given a reward.

The three dieren t rule tness properties are rewarded and weighted. Each correct gene is given 2 points. The calculation for correctness in rule composition is shown in Figure 5. If the action part and centre (C) condition is correct for a change or growth rule 3 points are given. If the action, centre and north (N) condition is fullled 4 more points are given, etc. A total of 25 points if all gene entries are correct. Each rule in compliance with all restrictions is given 20 points. Note that rule tness value depends on genome size.

Genomes including valid rules are developed and given a fitness value based on the number of correct cell types in the finalised developed phenotype i.e. on the last available development step. Each structural correct cell type i.e. black or white is given 50 points.

The development example in Figure 6 shows the steps for the best individual after 3000 generations. This example develops from single white cell to a multicellular organism of 8 by 8 cells expressing the desired chessboard pattern. The example shows a perfect solution. The phenotype is structural stable at development step 12 i.e. no change in the phenotype from development step 12.

The organism in Figure 6 obtained a fitness of 3200 for it's structural composition. A rule fitness of 286 in total where 100 was rewarded for the 5 correct rules. The individual is rewarded a total fitness value of 3486.

Table 2: Genome for phenotype in Figure 6. Rule 4 have highest priority.

Rule	Result [Action]	Center [Cond]	North [Cond]	South [Cond]	East [Cond]	West [Cond]
4	Gw	Z	DC	Z	Z	DC
3	Ge	Z	DC	Z	DC	DC
2	Gs	Z	DC	DC	DC	DC
1	W	W	W	Z	Z	DC
0	B	B	B	B	B	B

The genome for the development process shown in Figure 6 is presented in Table 2. The five exploited rules consists of three growth rules and two change rules.

3.3 Experiment and Results

To investigate how phenotype size influence on the result produced by the EA the number of cells available to development was increased. Four different runs were conducted for phenotypes of size 16, 64, 256 and 1024 cells. The genotype size was set to 4 rules.

The initial condition was set to a single cell of type black (the zygote). The number of available development steps was set to 100. The population size is set to 5. The maximum number of available generations is 3000.

Table 3: Results of rule size 4: phenotype size scaled from 16 to 1024 cells.

Rules	Size	SR
4	16	33%
4	64	41%
4	256	34%
4	1024	44%

Table 3 shows the results of the four runs increasing the maximum size of the phenotype. In the table Rules gives the number of possible rules in the genome. Size is the maximum number of cells available for development i.e. organism size. The SuccessRatio (SR) gives the percentage of perfect solutions found over the 100 runs.

In Table 4 the number of rules is increased to 5 rules. The size of the phenotype is increased from 16 to 1024 cells. The results of the runs are presented in the table.

Finally in Table 5 the number of rules is increased to 6. The size of the phenotype is again increased from 16 to 1024 cells as for the two previous experiments. The results are presented in the table.

Table 4: Results of rule size 5: phenotype size scaled from 16 to 1024 cells.

Rules	Size	SR
5	16	55%
5	64	64%
5	256	59%
5	1024	61%

Table 5: Results of rule size 6: phenotype size scaled from 16 to 1024 cells.

Rules	Size	SR
6	16	68%
6	64	70%
6	256	73%
6	1024	66%

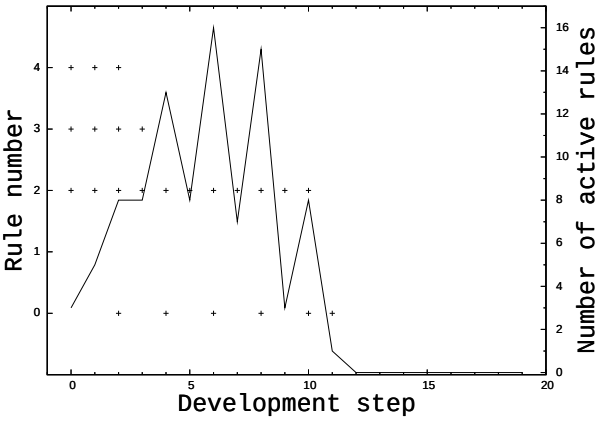


Figure 7: Gene activation pattern showing the rule activity in the development of the phenotype in Figure 6.

The gene activation pattern for development of one of the 5 rule genotypes that produced a perfect solution at the final development step is presented in Figure 7. Figure 6 shows the development of the phenotype. The plot in Figure 7 illustrates the gene activation together with the number of active rules in the organism at each development step. Rule numbers from 0 to 4 are placed on the left Y-axis. The mark (+) in the plot indicates that the rule was activated at the given development step. The right Y-axis show the number of cells in the organism with a active rule on a given development step. The number of active rule cells at each development step is illustrated by the plotted line.

The plot has an increasing number of active rules from the first development step to a maximum of 16 at development step 6. From development step 6 the number of active rules decreases down to 0 at the development step 12. The phenotype has reached a state of structural stability.

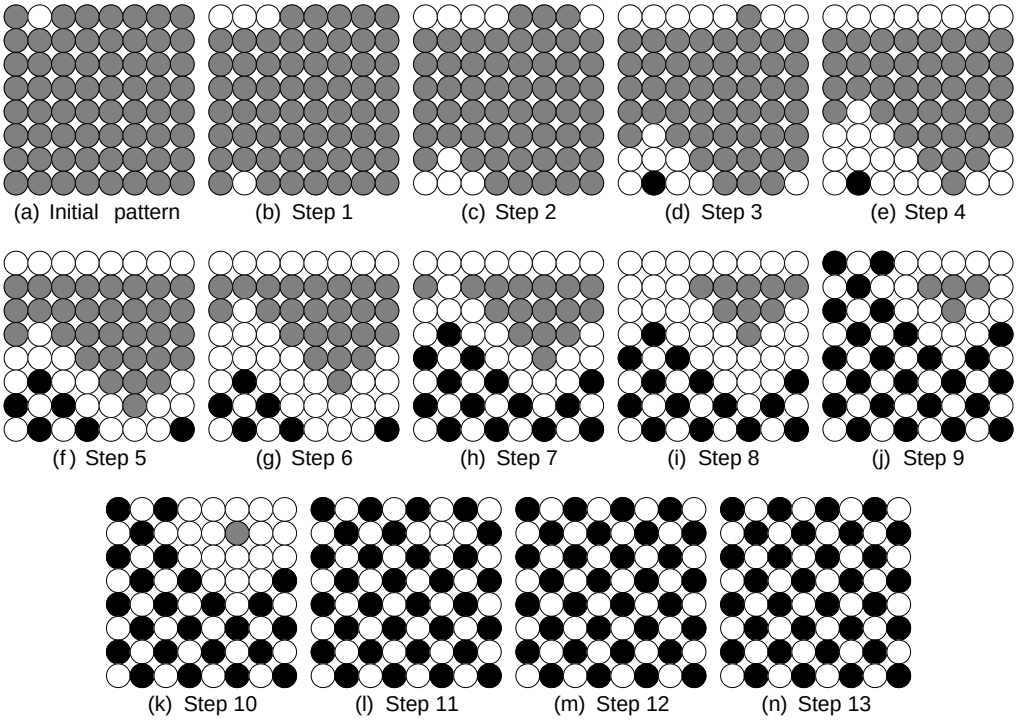


Figure 6: Development steps for final individual. The pattern is stable from step 12 until last step, step 20. Grey cells are empty(Z).

4. GENOME SCALING

From the task of evolving genomes for development of organism with structural properties in the previous section the task is now changed to evolve genomes for development of organisms with functional properties.

Functionality is expressed by the output of the functional components of the cell shown in Figure 1. As such, state steps are required to measure functionality in the phenotype.

Evolution of static and sequential behaviour is denied in [23]. Herein static behaviour is the target behaviour. Static behaviour is expressed as a global property of all cells included in the phenotype at a given state step at a given development step. The global properties chosen is the number of cells outputting a logical "1". As such, the GA search for genomes that can produce a phenotype consisting of suitable cell types at a given development step producing a desired state output at a given state step.

To be able to produce an organism able to output the desired function of 1024 logical "1"s, the genome must be capable of development of a structure that exploits all available cells. This structural requirement is not given in the fitness function but is a result of the fact that empty cells must be occupied to change their output value. As such, the functional goal can not be achieved if the phenotype structures do not exploit all available cells.

The cellular array used consists of 1024 cells {an array of 32 by 32 cells. The genome size is scaled by increasing the number of available rules that can be exploited by the development process.

The experiments are executed on a cPCI machine including a cPCI PC running the GA. The genomes are transferred on the cPCI bus to the FPGA card. The development process and functional behaviour of the cellular array are executed in the FPGA. Information from the developing phenotype is available by readback to the GA.

4.1 Evolutionary Algorithm

The approach here is to evolve genomes that can produce an emerging phenotype with functional properties. A fixed number of cell types to be exploited by evolution is chosen. The number of development steps and state steps is set to be fixed. As such, evolution must be able to find a solution within the given parameters. The set of cell types used are given in Table 6 together with their functional LUT definition and graphical symbols for graphical presentation.

The EA chosen is a straight forward Genetic Algorithm (GA) [22]. The GA's crossover operator for multiple crossover points is modified. The modification has two purposes. First, to avoid the context of genes to be disturbed, i.e. genes representing a cell type or growth direction is copied unmodified by the crossover operator. Second, the number of crossover points is variable within a given range. The variable number of points is implemented to be able to move rules up and down in the ranking, i.e. placement within the genome.

The representation of cell types and growth directions are not represented as symbols. Each gene is represented by a given number of bits. Seventeen bits are used to represent

Table 6: Dened Cell typ es and their functionalit y

Cell Type	LUT hex	Function name	Graphical representation
0	0xF F F F 0000	Empty	
1	0xF F F F F F F E	T 1	
2	0xF F F E F E E 8	T 2	
3	0xF E E 8E 880	T 3	
4	0xE 8808000	T 4	
5	0x80000000	T 5	
6	0x96696996	XOR5	
7	0x00000001	$\overline{T\ 1}$	
8	0x00010117	$\overline{T\ 2}$	
9	0x011717F	$\overline{T\ 3}$	
10	0x177F 7F F F	$\overline{T\ 4}$	
11	0x7F F F F F F F	$\overline{T\ 5}$	
12	0x69969669	$\overline{XOR5}$	

the twelve used cell typesin Table 6 together with the four possiblegrowth directions and the don't care condition. The genevalue is the number (sum) of bits set to logical "1" in a gene.

Further, the genevalue does not directly reect the gene typei.e. cell type, growth direction or condition. Instead of a direct translation of genevaluesto genetypesa look–uptable is used to translate the genevalue to a genetype. The trans–lation table used is given:

9; 15; 2; 3; 4; 5; 6; 17; 14; 7; 16; 17; 8; 0; 13; 11; 12; 10

A genevalue of zero point to the rst entry in the look–up table dened as cell type 9 in Table 6. The entries 13 { 16 representsthe four growth directions and 17 the don't care condition. Using a look–up table for gene translation has two important properties. First, it is easy to remove cell typesfrom the available cells as cell typesnot representedin the table can not be expressedin the phenotype. Second, the table can be exploited by the genetic operators. A single bit mutation in a gene will change the gene type one step up or down the table.

The purp oseof the uneven distribution of genevalues to genetypescausedby the genevalue representation is to keep the experiments herein compatible with ongoing work.

4.2 Evaluation

The initial condition is applied before development starts. This meansthat all cells are set or reset depending on the given initial condition. To avoid empty cells updating their output values from their von Neumann neighbourhood, all cells of type Empty are set to update their outputs based on only their own output value at the previous clock pulse. As such, a given empty cell will retain its initial state until the emerging organism grows into it.

In contrast to the experiments in section 3, where a part of the tness function is dened to reward rules that are com–posed correctly, here the tness function rewards genomes for including rules that is exploited during development.

In section 3 the argument for rewarding correct composi–tion within a rule was to favour such genomesto be able to exploit the genetic material towards genomesthat was ca–pable of growth and dieren tiation i.e. a zygote that starts to develop. Here a dieren t approach is taken. All genomes in the initial population are made of genecombinations that are valid rules. However, there are no mechanism preventing the genetic operators to produce invalid rules.

A tness function that included activated rules in the emerging phenotype together with the number of cells out–putting a logical "1" at the nal state step at the nal de–velopment step was dened. The tness function is the sum of activated rules multiplied by 2 added to the number of cells outputting a logical "1". As such, the number of cells outputting a logical "1" denes the successof the structure developed. Exploited rules keep useful exploitable genetic material in the population.

The development example in gure 8 shows the develop–ment of an organism of 32 by 32 cells over 50 development steps. In the example a single cell of type XOR5 outputting a logical "1" (the zygote) develops to a multicellular organ–ism of 1024 cells. The target for evolution is to produce organisms outputting as many logical "1"s as possible i.e. 1024. The nalised organism at the last available devel–opment step is a structure of cell type 8 ($\overline{T\ 2}$) and 11 ($\overline{T\ 5}$). A genome size of 32 rules was used. Out of the available rules 6 rules was exploited. The presented emerg–ing phenotype is a result of 8619 generation of evolution.

The functional property of the phenotype was given a score of 1024 points based on the number of logical "1" outputted on the last state step of the last available de–velopment step. The rule tness was given 12 points based on the 6 exploited rules out of the available 32 rules in the genome. A tness of 1038 points.

Table 7: Genome for phenot yp e in Figure 8. Rule 31 have highest priorit y.

Rule	Result [Action]	Cen ter [Cond]	North [Cond]	South [Cond]	East [Cond]	W est [Cond]
0	Gs	D C	D C	Gs	3	D C
1	D C	D C	Gs	Gs	D C	D C
2	D C	D C	Gs	Gs	6	D C
3	Gs	0	6	Gw	Gs	D C
4	8	6	D C	D C	D C	8
5	D C	D C	5	Gs	8	Gw
6	6	5	Gs	Gs	Gw	Gs
7	D C	D C	D C	Gw	8	D C
8	8	6	D C	D C	6	D C
9	D C	Gw	5	6	6	Gs
10	D C	D C	6	Gw	Gw	4
11	D C	6	5	D C	D C	D C
12	D C	6	Gw	Gs	D C	0
13	D C	6	D C	Gs	D C	8
14	D C	Gw	Gs	D C	5	Gw
15	D C	Gw	6	6	Gw	Gs
16	Gn	0	D C	D C	D C	D C
17	Gs	0	6	D C	6	5
18	Gw	0	D C	D C	D C	D C
19	D C	Gs	D C	Ge	6	5
20	D C	D C	Gw	5	4	D C
21	6	D C	8	D C	Gs	Gs
22	Gs	0	D C	D C	D C	D C
23	6	8	11	D C	5	D C
24	D C	Gw	Gs	0	Gs	D C
25	D C	Gs	Gw	D C	D C	Gs
26	8	D C	5	Gs	Gw	Gs
27	Gw	D C	0	Gs	D C	D C
28	Gw	0	Gw	Gs	Gs	0
29	Gs	0	Gs	D C	8	D C
30	Gs	0	5	Gw	D C	Gs
31	11	8	D C	6	D C	D C

The genome for the development processshown in Fig–ure 8 is shown in Table 7. The 6 exploited rules consists of three growth rules and three change rules.

4.3 Experiment and Results

The initial state is set to be a single cell of type 6 (XOR5)

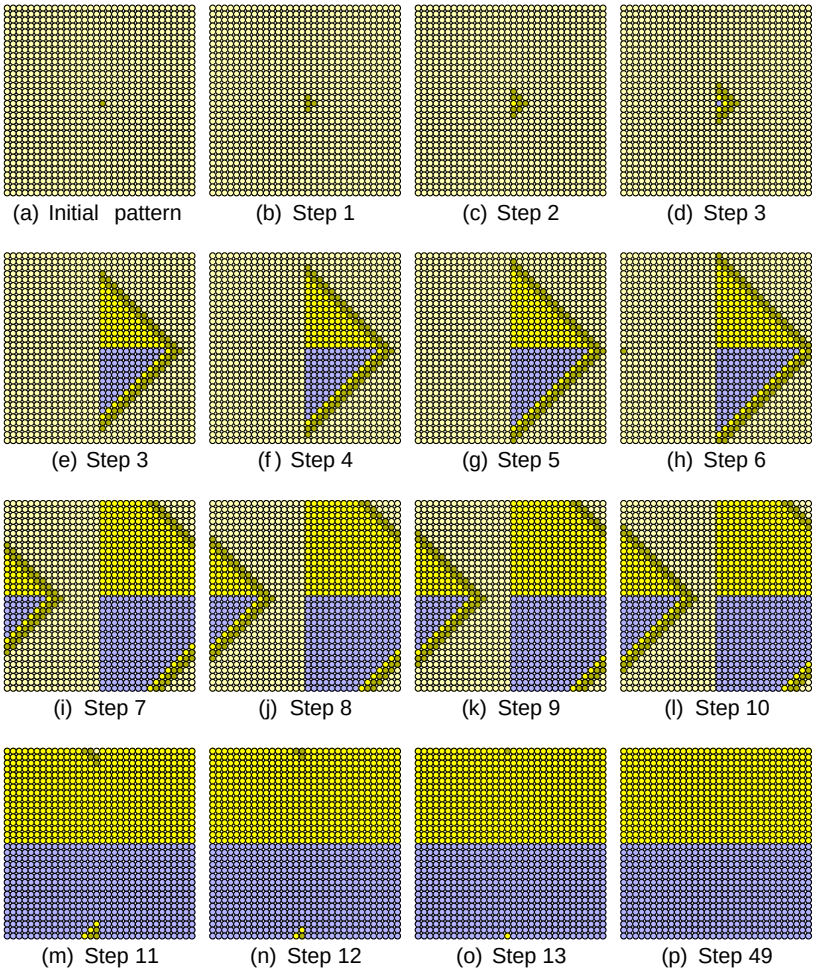


Figure 8: Selected development steps for development of static behaviour. The output states of the cells makes a bit array of 1024 cells outputting a logical "1".

the cell is set to output a logical "1". The population size is set to 16. The genome size was scaled from 4 to 32 rules. Development steps were set to 50. State steps for each development step was set to 30. The number of activated rules and the cell types in the cellular array was recorded together with the tness value.

Note that due to it's functional properties of once set always producing a logical "1" the cell type T 1 is not included in the runs.

Table 8: Result of searching for static behaviour.

Rules	Used rules best/mean	"1"s best	Fitness best/mean	Generation best/mean
4	4/3.4	909	917/438.2	10721/9222,6
8	6/4.8	920	932/744.7	8816/10731.4
16	7/5.5	1024	1038/931.9	17685/10000.3
32	8/6.8	1024	1040/1010.4	15565/10057.6

The result can be found in Table 8. The table contains the 4 dierent runs. Each run was repeated 10 times. For each run the number of active rules, number of cells outputting a logical "1" and the tness score for the best and mean over the 10 runs is presented. In the last column the respective generation for the best individual is presented together with the mean generation for the best of 10 runs.

The gene activation pattern for development of one of the 32 rule genotypes that produced a perfect solution of outputting 1024 logical "1"s at the nal development step is presented in Figure 9. Figure 8 shows the development of the phenotype. The plot in Figure 9 illustrates the gene activation together with the number of active rules in the organism at each development step. Rule numbers from 0 to 31 are placed on the left Y-axis. The mark (+) in the plot indicates that the rule was activated at the given development step. The right Y-axis show the number of cells in the organism with an active rule on a given development step.

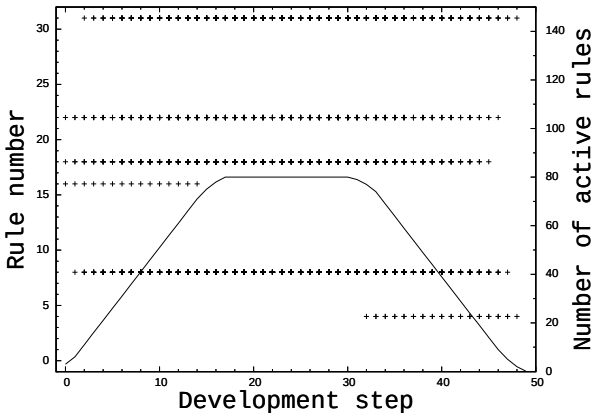


Figure 9: Gene activ ation pattern showing the rule activit y in the dev elopmen t of the phenot yp e in Fig– ure 8.

The number of cells with active rules at each development step is illustrated by the plotted line.

The plot has an increasing number of active rules from the rst development step to a maximum of 80 at development step 17. From development step 17 to 30 the number of active rules is constant at 80 active rules. At development step 32 the number of active rules decreasesdown to 0 at the last development step. The phenotype has reached a state of structural stabilit y. Note that not all of the produced phenotypesin the experiment reached structural stabilit y.

5. DISCUSSION

In the experiments considering phenotype scaling the struc– ture emergesas a result of the interplay betweenthe genome and the phenotype. The goal structure may be said not to be a true emerging global structure. Each cell can take ac– tion to expressit's correct cell type only by examining the local neighbourhood. However, the genomepresented to the development processmust be arranged in such a way that it is able to develop the target pattern.

The true parallel cellular development processrequire gene regulation. Regulating the genesto grow and dieren tiate without constructing structural conditions that is not cov– ered in the limited size of the genome or leads to oscillating substructures. In the parallel cellular development process all cells execute the development processin parallel. Cells can only rely on the information provided by it's neighbours at the current development step. As such, a cells action based on the structural information provided is unambigu– ous regarding the cells development action i.e. change or growth.

The result of evolution of developmental genomesfor solv– ing the cellular structure of a chessboard indicate that the ES can take advantage of the increased number of available cells to increase the successratio.

The increasedsuccessratio if the number of available rules was increased also show that the ES was able to eectiv e exploit the available gene regulation network.

In the experiments for genomescaling a true global prop– erty that can only be achieved as an emerging property out of local interactions is the target. The genome and the de– velopment processmust construct a structure of functional components that can meet the target criteria. The removal of the cell type T 1 ensurethat a solution exploit the func– tional components of the cells using the state steps. There are no cells avilable that can be used by growth and dif– ferentiation alone to build a structure of cells outputting a logical "1".

The result of the experiment shows that evolution was able to exploit the increased genome size toward better so– lutions. However, it is clear that only parts of the genome are exploited (Figure ??. Looking at the genome for the organism in Figure 8 illustrates how only six of the rules are exploited. However, the genomecarry information from the evolutionary processthat is not exploited in the current generation.

Looking at the graphical presertation of development in Figure 8 shows that the nal phenotype is constructed of two cell types only. However, the phenotype is developed exploiting a third cell type in the growth and dieren tiation process. This third cell is part of the actual result obtained even if it is not presert at the nal development step. The cell is part of the functionalit y expressed. The nal output bit pattern is a result of all development steps and state steps from the rst cell outputting a logical "1".

If the preserted example of gene regulation plots in Fig– ure 7 and Figure 9 and graphical presertation of the de– veloping organisms in Figure 6 and Figure 8 are examined both EAs found stable structural phenotypesat the last de– velopment step available. There are no specic mechanisms for self regulation available. Evolution is capable of nd– ing genomesthat develop to organisms were self–regulation emergesout of the local cell interactions.

In the experiments considering phenotype scaling and ex– periments for genome scaling a common obstacle must be solved. The number of available development steps was set to be xed. As such, the structural requirement in both casesof exploiting all available cells the developmental genome must provide a growth ratio that can expand the organism in the appointed number of development steps.

6. CONCLUSIONS

The experiments regarding phenotype scaling preserted in section 3 and genome scaling in section 4 indicate that increased number of available cells in the phenotype and the size of the genome inuence the results in a positive direction.

The fact that both EAs were able to exploit the devel– opment model was encouraging. This may indicate that a development mapping may not be that sensitive to the EA implementation and parameter tuning.

The increased performance in the experiments by the in– creased size of the genome and to some extent the pheno– type size indicates that the development processis capable of exploiting the increased information available. Informa– tion here is an increased number of structural chessboard cells that can be used in the developmental computation to construct the nalized board. For increased genome size the number of available rules is leading to a larger number of instructions i.e. rules, that can be exploited during the development process.

7. REFERENCES

[1] K. Aamot. Kunstig utvikling: Utvidelse av fpga−basert sblock−plattform. Master's thesis, The Univ ersity of Scienceand technology, Norway, 2005.

[2] P. Bentley and S. Kunar. Three ways to grow designs: A comparison of embryongeniesfor an evolutionary design problem. In GECCO, pages35{43, 2000.

[3] P. J. Bentley, editor. Evolutionary Design by Computers. Morgan Kaufmann Publishers,Inc, San Francisco, USA, 1999.

[4] S. Droste, T. Jansen, G. Rudolph, H. Schwefel, K. Tinnefeld, and I. Wegner. Theory of evolutionary algorithems and genetic programming. In H. P. Schwefel, I. Wegener, and K. Weinert, editors, Advancec in Computational Intel ligence Theory and Practice, pages107{144. Springer, 2003.

[5] M. Ebner, M. Shackleton, and R. Shipman. How neutral networks inuence evolvabilit y. Complexity, 7(2):19{33, 2001.

[6] T. G. W. Gordon. Exploring models of development for evolutionary circuit design. In 2003 Congress on Evolutionary Computation (CEC 2003), pages 2050{2057. IEEE, 2003.

[7] T. G. W. Gordon and P. J. Bentley. Bias and scalability in evolutionary development. In the 2005 Genetic and Evolutionary Computation Conference, pages83 { 90. ACM Press, 2005.

[8] T. G. W. Gordon and P. J. Bentley. Development brings scalability to hardware evolution. In the 2005 NASA/DOD Conference on Evolvable Hardware (EH05), pages272 {279. IEEE, 2005.

[9] P. Haddow and G. Tufte. An evolvable hardware FPGA for adaptiv e hardware. In Congress on Evolutionary Computation(CEC00), pages553{560, 2000.

[10] S. Kumar. Investigating Computational Models of Development for the Construction of Shape and Form. PhD thesis, Univ ersity College London (UCL), 2004.

[11] S. Kumar and P. J. Bentley. Biologically inspired evolutionary development. In 5th International Conference on Evolvable Systems(ICES03), Lecture Notes in Computer Science, pages57{68. Springer, 2003.

[12] S. Kumar and P. J. Bentley, editors. On Growth, Form and Computers. Elsevier Limited Oxford UK, 2003.

[13] J. Liu, H.and Miller and A. Tyrrell. Intrinsic evolvable hardware implementation of a robust biological development model for digital systems. In the 2005 NASA/DOD Conference on Evolvable Hardware (EH05), pages87 { 92. IEEE, 2005.

[14] D. Mange, S. Moshe, A. Stauer, and G. Tempesti. Towards robust integrated circuits: The embryonics approach. Proceedings of the IEEE, 88(4):516{543, April 2000.

[15] D. Mange, E. Sanchez, A. Stauer, G. Tempesti, P. Marchal, and C. Piruet. Embryonics: A new methodology for designing eld−programmable gate array with self−repairand self−replicating properties. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 6(3):387{399, September 1998.

[16] J. Miller and K. Downing. Evolution in materio: Looking beyond the silicon box. In 2002 NASA/DOD Conference on Evolvable Hardware, pages167{176. IEEE Computer Society Press, 2002.

[17] J. F. Miller. "evolving developmental programs for adaptation, morphogenesis,and self−repair. In Seventh European Conference on Articial Life, Lecture Notes in Articial Intelligence, pages256{265. Springer, 2003.

[18] J. F. Miller and P. Thomson. A developmental method for growing graphs and circuits. In 5th International Conference on Evolvable Systems (ICES03), Lecture Notes in Computer Science,pages 93{104. Springer, 2003.

[19] E. Sanchez, D. Mange, M. Sipper, M. Tomassini, A. Prez−Uribe, and A. Stauer. Phylogeny, ontogeny, and epigenesis: Three sources of biological inspiration for softening hardware. In T. Higuchi, M. Iwata, and W. Liu, editors, Evolvable Systems: from Biology to Hardware, ICES 96, volume 1259 of Lecture Notes in Computer Science, pages35{54. Springer, 1996.

[20] M. Sipper. Evolution of Parallel Cellular Machines The Cellular Programming Approach. Springer−Verlag, 1997.

[21] M. Sipper. The emergenceof cellular computing. Computer, 32(7):18{26, 1999.

[22] W. M. Spears. Gac ga archives source code collection webpage. http://www.aic.nrl.navy.mil/g alist /src /, 1991.

[23] G. Tufte. Cellular development: A search for functionalit y. In submited to Congress on Evolutionary Computation(CEC2006). IEEE, 2006.

[24] G. Tufte. Gene regulation mechanisms intro duced in the evaluation criteria for a hardware cellular development system. In submited to 1st NASA/ESA Conference on Adaptive Hardware and Systems (AHS−2006). IEEE, 2006.

[25] G. Tufte and P. Haddow. Biologically−inspired: A rule−basedself−reconguration of a virtex chip. In 4th International Conference on Computational Science 2004 (ICCS 2004), Lecture Notes in Computer Science,pages1249{1256. Springer, 2004.

[26] G. Tufte and P. C. Haddow. Building knowledg into developmental rules for circuite design. In 5th International Conference on Evolvable Systems (ICES03), Lecture Notes in Computer Science,pages 69{80. Springer, 2003.

[27] G. Tufte and P. C. Haddow. Towards development on a silicon−basedcellular computation machine. Natural Computation, 4(4):387{416, 2005.

[28] A. Tyrell, E. Sanchez, D. Floreano, G. Tempestti, D. Mange, J. Moreno, J. Roserberg, and A. E. P. Villa. Poetic tissue: An integrated archtecture for bio−inspired hardware. In 5th International Conference on Evolvable Systems(ICES03), Lecture Notes in Computer Science,pages127{140. Springer, 2003.

[29] J. Von Neumann. Theory of Self−Reproducing Automata. Univ ersity of Illinois Press, Urbana, IL, USA, 1966., 1966.

[30] L. Wolpert. Principles of Development, Second edition. Oxford Univ ersity Press, 2002.

Bibliografi

[1] P. Bentley and S. Kumar. Three ways to grow designs: a comparison of embryogenies for an evolutionary design problem. *GECCO-99. Proceedings of the Genetic and Evolutionary Computation Conference.*, vol.1:35–43, 1999.

[2] P.J. Bentley. *Digital Biology*. Simon and Schuster, 2001.

[3] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conseptual comparison. *ACM Computing Surveys*, 35(3):268–303, September 2003.

[4] K. Culik II, L.P. Hurd, and S. Yu. Computation theoretic aspects of cellular automata. *Physica D*, 45(1-3):357–78, 1990.

[5] R. Das, M. Mitchell, and J.P. Crutchfield. A genetic algorithm discovers particle-based computation in cellular automata. In *PPSN III: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature*, pages 344–353. Springer-Verlag, 1994.

[6] S. Droste, T. Jansen, G. Rudolph, H.P. Schwefel, K. Tinnefeld, and I. Wegener. Theory of evolutionary algorithms and genetic programming. *Advances in Computational Intelligence*, pages 107–144, 2003.

[7] M. Ebner, M. Shackleton, and R. Shipman. How neutral networks influence evolvability. *Complexity*, 7(2):19–33, 2001.

[8] A.E. Eiben. Evolutionary computing: the most powerful problem solver in the universe? *Dutch Mathematical Archive*, 5/3(2):126–131, 2002.

[9] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, 2003.

[10] P. C. Haddow and G. Tufte. Bridging the genotype-phenotype mapping for digital FPGAs. In *the 3rd NASA/DoD Workshop on Evolvable Hardware*, pages 109–115, 2001.

[11] P. C. Haddow, G. Tufte, and van Remortel P. Shrinking the genotype: L-systems for ehw ? In *4th International Conference on Evolvable Systems (ICES01)*, pages 128–139, 2001.

[12] S. Kumar. *Investigating Computational Models of Development for the Construction of Shape and Form*. PhD thesis, University College London, 2005.

[13] S. Kumar and P.J. Bentley. An introduction to computational development. In S. Kumar and P. J. Bentley, editors, *On Growth, Form and Computers*, pages 1–43. Elsevier Limited Oxford UK, 2003.

[14] J. F. Miller. evolving developmental programs for adaptation, morphogenesis, and self-repair". In *Seventh European Conference on Artificial Life*, Lecture Notes in Artificial Intelligence, pages 256–265. Springer, 2003.

[15] P.H. Raven, G.B. Johnson, J.B. Losos, and S.R Singer. *Biology*. McGraw-Hill, 6th ed. edition, 2002.

[16] D. Roggen and D. Federici. Multi-cellular development: is there scalability and robustness to gain? In X. Yao, E. Burke, J.A. Lozano, and al., editors, *Parallel Problem Solving from Nature - PPSN VIII. 8th International Conference. Proceedings (Lecture Notes in Comput. Sci. Vol.3242)*, pages 391–400, 2004.

[17] M. Sipper. *Evolution of Parallel Cellular Machines: The Cellular Programming Approach*. Springer-Verlag, 1997.

[18] M. Sipper, E. Sanchez, D. Mange, M. Tomassini, A. Perez-Uribe, and A. Stauffer. A phylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems. *IEEE Transactions on evolutionary computation*, 1(1), 1997.

[19] J. Thomassen. Biological inspired hardware: Exploration of evolution for artificial development. Website, 2006. `http://net.homelinux.org/master`.

[20] G. Tufte and P.C. Haddow. Towards development on a silicon-based cellular computing machine. *Natural Computing*, 4(4):387–416, 2005.

[21] G. Tufte and J. Thomassen. Size matters: Scaling of organisms and genomes for development of emergent structures. 2006.

[22] J. Von Neumann. Theory of self-reproducing automata. 1966.

[23] L. Wolpert. *Principles of Development, Second edition*. Oxford University Press, 2002.