Bioscena
Simulació d'un sistema biològic evolutiu amb interacció entre els individuus

David Garcia
vokimon@jet.es

7 de setembre de 1999

Índex

0.1 Abstract

Aquest projecte planteja la simulació d'un sistema biològic natural evolutiu amb interacció entre els organismes dins d'un medi de variabilitat controlada. Es tracta de reproduir comportaments naturals o lògics en individuus mitjançant el procés evolutiu. Es vol estudiar si el fet d'acostar un algorisme genètic al procés real natural tenint més en compte aspectes biològics i naturals dona una major adaptabilitat a un medi variable.

La part pràctica consisteix en la implementació d'un prototip que permeti a l'usuari veure les relacions que s'estableixen entre els individuus al llarg del procés evolutiu.

0.2 Resum

L'objectiu del present treball de fi de carrera és implementar una eina d'estudi ampliable, d'aplicació als camps de la biologia i la vida artificial que permeti a un usuari configurar, analitzar e intervindre en la dinàmica d'un sistema biològic evolutiu simulat amb interacció entre organismes i entre cada organisme i el medi variable.

Per això, primer s'estudiaran els processos naturals (evolutius, etològics i ecològics), i es triaran aquells que siguin més susceptibles de èsser implementats i que afegeixin més realisme i generalitat al model. Hi haurà processos que s'implementaran directament al model, d'altres que s'espera que emergeixin. Tot i així, cal que considerem també els processos emergents de cara a fer el model per fer possible que els organismes els adoptin i detectar quan i com ho fan.

L'aplicació proveirà eines de configuració i intervenció perque l'usuari pugui controlar la forma en que varia el medi i, així, poder contrastar-ho amb els resultats obtinguts.

També s'implementaran eines d'anàlisis per tal de que es puguin detectar els fenòmens que en l'estudi previ dels processos naturals es considerin interesants.

El sistema ha de ser prou flexible per permetre l'experimentació amb configuracions prou variades. A més, cal donar a l'usuari programador l'espai necessari per modificar algun aspecte concret del model o ampliar les opcions donades, tot modificant el codi font.

Part I

Introducció

Capítol 1

Introducció al projecte

1.1 Àmbit del projecte

Aquest projecte és dins de l'àmbit de la vida artificial (Artificial Life), disciplina que recull coneixements d'informàtica i biologia per recrear fenòmens biològics en un entorn artificial.

Aquesta disciplina va sorgir de cara a la biologia com a camp de proves alternatiu a la vida real, però, les aplicacions van extenent-se, per exemple, aportant noves perspectives a l'analisis, simulació i predicció de sistemes complexos no biològics i a algorismes per la resolució de problemes als sistemes informàtics.

Les aplicacions de la vida artificial tenen un seguit de característiques més o menys comunes. Les principals són:

Mètode sintètic: En comptes d'analitzar la vida, sintetitzem artificialment sistemes amb un comportament similar partint de premises que obtingudes de l'analisis dels sistemes reals.

Construccio Botton-Up: Es parteix de unitats petites, definint les interaccions locals, en comptes de partir del compontament global desitjat i anar perfilant com han de ser els components. El comportament global del sistema és un comportament que no estava explícitament dissenyat.

Emergència: El fet de que apareixin aquests comportaments globals no dissenyats explícitament a partir d'un entramat complex d'interacions simples s'anomena emergència.

No lligat als sistemes reals: Donat el seu caracter sintètic la vida no es limita a la vida coneguda sinó que prova de extreure propietats generals per a qualsevol forma de vida possible.

Paral·lelisme implícit: La complexitat dels sistemes vius es deguda, en part, a que els diferents processos es donen en paral·lel. Per fer això en els sistemes de vida artificial, tot i que no sempre sigui possible dedicar un processador a cada procés, sí que caldria fer servir tècniques de temps compartit, que, macroscòpicament, els diferents processos donin la impressió d'executar-se paral·lelament.

1.2 Antecedents

Concretament, l'objectiu del present treball de fi de carrera és implementar una eina d'estudi i experimentació amb aplicació als camps de la biologia i la vida artificial que permeti a un usuari configurar, analitzar e intervindre en la dinàmica d'un sistema biològic evolutiu simulat amb interaccions organisme-organisme i organisme-medi.

Dels treballs que s'han fet sobre vida artificial, cal destacar, com a precedents, per la seva relació amb aquest projecte, els següents:

- El projecte 'Tierra' que encapçala Thomas S. Ray [Ray93] un biòleg que va aplicar els seus coneixements de genètica i de ecologia i els va traduir a un medi informàtic on petites porcions de codi competien per la memòria i el temps del processador per replicar els seus gens.

- Les experiències que Richard K. Beleg i Filippo Menezer van fer sobre el model Latent

Energy Environments (LEE). LEE és un model que es basa en l'evolució de sistemes cognitius interaccionen amb un medi químic de complexitat controlada. El més remarcable d'aquest projecte, de banda de les diverses conclusions que n'han anat treient en diferents estudis, és, per un costat, la integració d'algorismes evolutius i xarxes neuronals, i, d'altre, el fet de basar-se en un model metabòlic.

Bioscena moltes idees dels LEE, com es veurà en la resta de la memòria, però, a diferència d'aquests, no pretén simular organismes amb comportaments cognitius, sinò organismes amb comportaments reactius. La diferència principal radica en que els comportaments reactius estan controlats principalment pel codi genétic i l'interacció *química* amb el medi que s'en deriva i, els comportaments cognitius estan controlats principalment per un sistema nerviós central i un procés d'aprenentatge.

Per modelar el comportament dels organismes de Bioscena, es fan servir rafagues d'instruccions que formen el genotip de forma semblant a com funcionava 'Tierra'. La diferència important és que, a 'Tierra', el conjunt d'instruccions feien de genotip, medi i fenotip, tot alhora, i en Bioscena la idea és separar el conjunt d'instruccions del medi i tant medi com genotip actuaran sobre el fenotip.

La simulació suposa, per a totes les decisions de disseny on sigui necessari que els organismes són unicel·lulars. Aquesta suposició no vol dir que els resultats siguin aplicables només a aquest tipus d'organisme, però, facilita el disseny, donat que el model d'un organisme pluricel·lular és molt complexe. A més hi ha la possibilitat, si el medi fós suficientment ampli, de que l'evolució portés als organismes unicel·lulars a formar estructures cooperatives més grans similars als organismes pluricel·lulars.

Resumint la comparació entre els tres sistemes:

Tierra	Sopa primigenia, organització cel·lular procariota
Bioscena	Organismes unicel·lulars eucariotes no cognitius
LEE's	Organismes cognitius, model d'alt nivell

1.3 Objectius del projecte

Els objectius principals del projecte són:

1. Fer un estudi dels processos naturals (evolutius, ecològics i etològics) que es donen a la natura. Per un costat, cal triar els que afegeixirien més realisme i generalitat al model en cas d'implementar-se. Per un altre costat, cal triar aquells fenòmens que es donen en la natura que no cal implementar directament, sinó, que s'espera que puguin emergir. L'esforç de l'analisis ha d'anar adreçat a modelar aquelles característiques no cognitives de caire general, i sobretot, dels organismes unicel·lulars, tot i que no cal descartar que emergeixin estructures pluricel·lulars o cognitives.

2. Fer un estudi bibliogràfic dels processos anteriors ja implementats. En quines condicions concretes s'han fet i quins resultats han donat.

3. Dissenyar i implementar el model amb tota la informació recollida, possibilitant, sense dirigir-la, l'aparició dels fenòmens emergents esperats, i l'adaptació dels organismes a variacions que es poden donar durant la vida d'un organisme o al llarg de generacions. El model tindrà diversos elements principals:

    - El biosistema, que coordina la resta d'elements.
    - El biòtop, que és el medi on viuran els organismes. Cal que permeti configuracions molt diverses per donar possibilitats d'experimentació. Per modificar el biòtop farem servir, sobretot, els agents, que són modificadors programables del biòtop.

- La comunitat, que controla el conjunt d'organismes presents al medi i la informació que no és intrínseca a ells, com ara la seva posició en el medi i la població a la que pertany. Els organismes proveeixen al medi les accions que volen realitzar i un conjunt d'operacions que el medi pot fer sobre ells.

Els elements del sistema han d'estar dèbilment acoblats per tal de poder, en un futur, intercanviar-los per uns altres amb un mínim d'efectes laterals.

4. Afegir al model les eines d'anàlisis han de poder donar informació útil del que està passant al biosistema: Per això, cal dividir la comunitat en poblacions i analitzar les interaccions entre elles. Cal posar una cura especial en detectar l'aparició de fenòmens emergents. Les eines de configuració i intervenció han de servir per controlar la forma en que canvia aquest medi, de forma que, els resultats obtinguts amb les eines d'analisis siguin confrontables i els usuaris puguin extreure conclusions.

A aquests objectius es poden sumar altres objectius adicionals que es cobriran de forma secundària.

1. El sistema ha de poder abocar-se totalment o parcial a disc per tornar-se a restaurar. Això faria possible execucions molt més llargues i obtenir poblacions més evolucionades.
2. Ha de ser possible canviar alguns aspectes de la configuració del biòtop o intervindre en la població (extraccions, introduccions, clonacions...) sobre la marxa.
3. Establir uns procediments de modificació per tal de que l'usuari-programador no necessiti comprendre tot el sistema per fer una modificació puntual.

Els organismes de la simulació han de fer front a un medi variable, amb canvis caòtics o periòdics que es poden produir freqüentment al llarg de la vida d'un organisme o de forma progressiva en el decurs de diverses generacions. Amb la finalitat de que la comunitat tingui capacitat

de reaccionar davant de tots aquests canvis, s'incorporen, als mecanismes evolutius que intervenen en la simulació, algunes característiques que es donen en els entorns evolutius naturals i que, clàssicament, no s'incorporen en els entorns evolutius computacionals.

Les característiques naturals que s'estudiarà introduïr a l'entorn evolutiu són:

Mecanismes d'expressió gènica (transcripció, maduració i traducció): Útils per implementar les altres característiques

Regulació sobre l'expressió gènica: Control dels gens que es transcriuen segons la presència o no d'alguns factors. Ajuda a adaptar-se, sense l'utilització d'un sistema cognitiu, a canvis en el medi tan esporàdics que el procés evolutiu no els soporti.

Promotor/teminador, zones no codificadores i longitud de cromosoma variable: Permet solucions obertes no parametritzades (Nombre i longitud variable pels gens)

Paràmetres del procés evolutiu codificats parcialment al genotip: Ens permetrà tenir uns paràmetres optimitzats per a cada situació concreta.

Genotips no haplonts i al·lels (Encara he de considerar si implementar-ho): Mantenen la variabilitat genètica de la descendència augmentant així la capacitat de canvi i adaptació.

Cariotip multicromosòmic: Divideix la dotació gènica en subunitats d'alt nivell que permeten traspasar divers material gènic de cop i possibilita les mutacions cromosòmiques que poden emergir en creuament.

1.4 Contingut de la memòria

```
// Aleaturitzador.cpp: implementation of the CAleaturitzador class.
//
```

```
//////////////////////////////////////////////////////////////////////

#include "Aleaturitzador.h"
#include "FuncioAgent.h"
#include "Color.h"

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CAleaturitzador::CAleaturitzador()
{
m_tipus+="/Aleaturitzador";
m_probabilitat.m_encerts=1;
m_probabilitat.m_mostra=1;
m_reAccio = NULL;
m_accionat = false;
}

CAleaturitzador::~CAleaturitzador()
{
if (m_reAccio) delete m_reAccio;
}

//////////////////////////////////////////////////////////////////////
// Redefinibles
```

```
//////////////////////////////////////////////////////////////////////

void CAleaturitzador::operator()(void)
{
m_accionat=false;
// Nomes si es dona la probabilitat i existeix l'accio, l'executem
if (m_probabilitat.esDona())
{
CMultiAgent::operator()();
m_accionat=true;
}
else if (m_reAccio)
(*m_reAccio)();
}

void CAleaturitzador::dump(CMissatger & msg)
{
CMultiAgent::dump(msg);
msg << "- Probabilitat "
<< m_probabilitat.m_encerts << " "
<< m_probabilitat.m_mostra << endl;
if (m_reAccio) msg << "- ReAccio " << m_reAccio->nom() << endl;
}

bool CAleaturitzador::configura(string parametre, istream & valor, t_diccionariAgents & diccio
{
```

```
if (parametre=="Probabilitat") {
uint32 mostra, encerts;
if (!(valor>>encerts))
errors << "Format invalid per la probabilitat de '" << nom() << "'" << endl;
else if (!(valor>>mostra))
errors << "Falta la quantitat de mostra per a la probabilitat de '" << nom() << "'" << endl;
else
probabilitat(mostra, encerts);
return true;
}
if (parametre=="ReAccio") {
string nomSubordinat;
t_diccionariAgents::iterator it;
if (!(valor>>nomSubordinat))
errors << "Format invalid per a l'especificacio de subordinat de '" << nom() << "'" << endl;
else if (m_reAccio)
errors << "L'agent '" << nom() << "' ja tenia una reAccio" << endl;
else if ((it=diccionari.find(nomSubordinat))==diccionari.end())
errors << "L'agent subordinat '" << nomSubordinat << "' de l'agent '" << nom() << "' no esta de
else if ((*it).second->subordinant())
errors << "L'agent subordinat '" << nomSubordinat << "' de l'agent '" << nom() << "' ja esta su
else
reAccio(it->second);
return true; // Parametre interceptat
}
// Li deixem a la superclasse que l'intercepti si vol
```

```
return CMultiAgent::configura(parametre, valor, diccionari, errors);
}

list<CAgent*> CAleaturitzador::subordinats() {
list<CAgent*> l=CMultiAgent::subordinats();
if (m_reAccio) l.push_back(m_reAccio);
return l;
}

////////////////////////////////////////////////////////////////////
// Operacions
////////////////////////////////////////////////////////////////////

void CAleaturitzador::probabilitat(uint32 mostra, uint32 encerts)
{
m_probabilitat.m_mostra=mostra;
m_probabilitat.m_encerts=encerts;
}

void CAleaturitzador::reAccio(t_accio * a)
{
m_reAccio=a;
if (a&&!a->subordinant(this))
warning << "Subordinant l'agent '" << a->nom() << "' a un segon subordinant" << endl;
}
```

```
//////////////////////////////////////////////////////////////////////
// Proves
//////////////////////////////////////////////////////////////////////

static void Ko () {out << "-";}
static void Ok () {out << "O";}

void CAleaturitzador::ProvaClasse()
{
out << "\033[J";// Un clrscr xapuser pero standard (ANSI)
out << blanc.brillant() << "Provant Agent Aleaturitzador" << blanc.fosc() << endl;
int exits=0;
int intents=0;
CAleaturitzador aleaturitzador;
CAgent * ag1= new CFuncioAgent(Ok);
CAgent * ag2= new CFuncioAgent(Ko);
aleaturitzador.probabilitat(40,25);
aleaturitzador.accio(ag1);
aleaturitzador.reAccio(ag2);
for (int i=400; i--;) {
aleaturitzador();
intents++;
if (aleaturitzador.m_accionat)
exits++;
}
out << endl << "S'han executat " << exits << " de " << intents << endl;
```

```
aleaturitzador.dumpAll(out);

cin.get();

}
```

Part II

Part teòrica

Capítol 2

Coneixements teòrics sobre biologia

2.1 Introducció

Aquest capítol introdueix alguns conceptes sobre procediments que es donen a la natura que necessitem coneixer per, posteriorment, aplicar-los en dos àmbits. Per un costat, necessitem coneixer els procediments genètics que es donen a la natura per adaptar-los als algorismes genètics. Per un altre costat, necessitem coneixements sobre etologia i ecologia per interpretar i analitzar els resultats de la simulació.

2.2 Genètica mendeliana

2.2.1 Gen, Al·lel, Genotip i Fenotip

Els primers estudis genètics de la història [MENDEL 1866] van pendre una perspectiva externa als individuus per estudiar l'herència (als organismes que es reprodueixen sexualment). És a dir, a partir dels caràcters observables d'individuus de diferents generacions, van intentar deduir els mecanismes o factors que intervenen en l'herència.

Aquest estudis, i les seves ampliacions posteriors, van madurar un seguit de conceptes que han perdurat fins avui en dia. S'expliquen a continuació:

El caràcter observable en el que ens fixem l'anomenem **fenotip**, per exemple, el color dels ulls. Un **gen** és cadascun dels factors hereditaris que controlen aquest fenotip. El normal és que siguin diversos gens els que controlin un fenotip el conjunt dels quals formen el seu **genotip**. Per exemple, imaginem que el color dels ulls el controla un genotip format per dos gens. Generalment, la correspondència entre genotip i fenotip no és directa, perque en el fenotip pot intervindre l'entorn. El conjunt de tots els gens que té un organisme (no pas considerant un sol fenotip) és el **genoma**.

Un **al·lel**, és cadascuna de les alternatives (o "valors") que pot adoptar un gen. Per exemple, imaginem que tenim les alternatives A, B, C y D. Dos al·lels es consideren diferents només si, en algun cas, el fet de tenir un o l'altre afecta al fenotip obtingut. Dos gens poden tenir els mateixos al·lels possibles. Si això passa i, a més, l'efecte d'un és equivalent al de l'altre, diem que són **gens homòlegs** El genotip és **homozigot** si els seus gens només presenten un al·lel per cada conjunt de gens homòlegs (Raça pura). En canvi, el genotip és **heterozigot** si els seus gens presenten al·lels diversos (Híbrid).

2.2.2 Expressió dels al·lels al fenotip

Un individuu homozigòtic presenta el caràcter fenotípic representatiu de l'al·lel. Aquest caràcter s'en diu el fenotip homozigòtic per aquest al·lel.

En la natura, sovint, els genotips homozigòtics estan associats, directament o indirecta, a fenotips negatius. Això és degut a que els individuus homozigòtics donen molt poca variabilitat genètica. Si una població acaba convertint-se en homozigotica, per exemple, per excesiva endogàmia, aquest gen es queda estancat i no dòna variabilitat. Si, per adaptar-se a un canvi en l'entorn cal canviar aquest gen, una població homozigòtica tindrà menys capacitat de resposta perqué dependrà de les mutacions. En els gens que no interesa aquest estancament la pròpia dotació genètica s'autopenalitza. Si una població no castiga la homozigosis, té més probabilitat

de tornar-s'en.

A vegades la millor forma de penalitzacio són els gens letals. Un genotip letal és aquella combinació d'al·lels que no es presenten mai donat que els individuus no sobrepassen l'estat embrionari.

Quan el genotip és heterozigot ens podem trobar les següents relacions entre els al·lels dels gens homòlegs, segons la forma d'expressar-se en el fenotip:

- **Dominància - Recessivitat:** Un al·lel (dominant) s'imposa sobre l'altre (recessiu), i, el fenotip resultant és el mateix que el d'un homozigot amb l'al·lel dominant.
- **Herència intermitja:** El fenotip no és l'equivalent a cap dels dos fenotips homozigots sinó que és un fenotip intermig.
- **Codominància:** Es presenten els fenotips dels dos al·lels a la vegada.
- **Superdominància o heterosis:** La presència de l'al·lel recessiu reforça el fenotip de al·lel dominant més que si fós un homozigot.

2.2.3 Fenotips de distribució contínua

Alguns caràcters de l'individu, com per exemple l'altura o la inteligència, no presenten unes alternatives fenotípiques tan discontínues sinó que són més contínues.

El caràcter pot dependre de diversos gens **poligen** amb la qual cosa es produeix una distribució normal en el caràcter. Com més gens s'hi impliquin, més graus discontinus hi haurà a la distribució. Per exemple:

Número de gens implicats	Distribució del caracter
1 (haploide)	1:1
2	1:2:1
4	1:2:3:4:3:2:1
6	1:6:15:20:15:6:1

Arriba un moment que hi intervenen tants gens que el gradient no és identificable. A més, considerant el fet de que l'entorn afecta al fenotip, tenim que un individuu amb genotip AABBBB pot ser fenotipicament més semblant a un homozigot A que un individuu amb genotip AAABBB.

2.2.4 Interaccions entre gens no homòlegs

Així com diversos gens homòlegs poden contribuir al fenotip d'un caràcter, gens no homòlegs també poden contribuir-hi, però en aquest cas la influència dels gens no es equivalent (per la definició anterior de gens homòlegs).

Una de les interaccions entre gens no homòlegs és la **epistàsia**: un gen (epistàtic) controla l'activació o desactivació d'un altre (hipostàtic).

2.2.5 Mutació gènica

El concepte d'herència tal i com el definia Mendel, s'enfrontava amb les noves idees de Darwin sobre l'evolució [DARWIN 1859]. El concepte que tenia Darwin sobre l'evolució és que als individuus es produien petits canvis (mutacions) dels quals la natura escollia els més apropiats per a l'entorn. En el concepte de Mendel sobre l'herència, no es creaven al·lels nous, les generacions posteriors tenen simplement una recombinació dels al·lels de les generacions anteriors. De Vries, un dels redescobridors dels postulats de Mendel, va introduir el concepte de mutació gènica [DE-VRIES 1900] que ve a reconciliar els dos corrents i donar una explicació a l'aparició de al·lels diferents dintre d'un gen.

Una mutació genica o puntual és l'aparició sobtada d'una nova alternativa (al·lel) per a un gen.

La mutació (tan si és gènica com si és una altra de les que veurem més endavant) és un fenòmen aleatori que es dona amb una determinada freqüència que sol ser molt baixa. La **freqüència de mutació** és una probabilitat que es mesura per a un gen donat, i durant una generació.

La probabilitat de mutació génica és mantè constant, tot i que diferent per a cada gen. Això s'explica per la diferent longitud de gens, o per la quantitat de mutacions que no impliquen un canvi d'al·lel (com es veu a la secció **??**).

Als organismes pluricel·lulars, les mutacions poden ser **somàtiques** o **germinals**. Una mutació somàtica, només afecta a una cel·lula i a totes les que en deriven. Per exemple, els tumors, les pigues... són mutacions somàtiques. Una mutació germinal es produeix a les cèl·lules del teixit que donaran lloc a les gametes. En conseqüència, aquesta mutació afectara, no només a la cèl·lula i a les que en derivin sinó que també a la descendència.

2.2.6 Sumari de conceptes aplicables al projecte

L'aproximació mendeliana als mecanismes de l'herència, tot i ser una primera aproximació, ja ens dona alguns conceptes que no són presents a la implementació clàssica dels algorismes genètics.

El primer concepte és el de gens homòlegs. A l'algorisme genètic clàssic, els gens no tènen homòlegs a menys que es considerin així a la funció d'evaluació i, generalment, no es fa. En el cas de tenir homòlegs, caldria resoldre el problema de l'heterozigosi la qual cosa implica un cost computacional superior que és sovint inútil en els problemes que tenen una funció d'avaluació constant. Aquest cost computacional, com a mínim, és el que implica duplicar la informació continguda al cromosoma.

En canvi, als problemes on la funció d'avaluació varia al llarg del temps, com és el cas d'un entorn biològic, és positiu guardar-se aquesta variabilitat en forma d'heterozigosis. Un genotip que no tingui gens homòlegs té els mateixos desavantatges que s'han comentat abans per un genotip homozigot, però, amb el desavantatge afegit de que mai pot arribar a ser heterozigot.

El concepte de mutació gènica, és el concepte de mutació puntual dels algorismes genètics clàssics. El que s'aporta de nou és el concepte de probabilitat de mutació variable segons el gen, i la diferència entre mutació somàtica i germinal de cara a transmetre les mutacions a la

descendència. Als apartats següents anirem modificant i enriquint el concepte de mutació a mida que es vagi desgavellant, en aquest capítol, la natura dels gens.

De cara a l'anàlisis dels resultats, pot ser interesant detectar si hi ha algun mecanisme que afavoreixi els genotips heterozigots i la presència de fenotips continus. També pot interesar detectar si es donen casos d'epistàsia, tot i que, com que els mecanismes d'epistàsia poden ser molt complexos, caldria limitar a alguns casos de baix nivell.

2.3 Teoria cromosòmica

2.3.1 Els cromosomes i l'herència

Els estudis de Morgan et al. varen demostrar que els gens no eren quelcom independent sinó que estaven en estructures superiors formant els **cromosomes**.

El gens es troben localitzats en un punt concret del cromosoma (**locus**). Quan reconvinem el material genètic de dos progenitors, els gens a locus més propers dins d'un cromosoma tenen molta més probabilitat d'heretar-se conjuntament. Això implica, els gens que controlen cada caràcter no s'hereten de forma tan independent com formula la tercera llei de Mendel. La tercera llei de Mendel diu que els caracters s'hereten de forma independent. Això és veritat, no en els caracters sinó en els gens i sempre que els gens tinguin un locus suficientment allunyats o en cromosomes diferents.

Anomenem **cariotip** al conjunt de cromosomes que té una espècie (en quant a nombre i morfologia).

Els cromosomes són homòlegs si tènen la mateixa morfologia i contènen gens homòlegs als mateixos *locus*.

Un cariotip és **diplont** si està format per n parelles de cromosomes homòlegs. És a dir, cada cromosoma té un altre d'homòleg, de tal forma que hi ha dos dotacions cromosòmiques

homòlogues. Generalment són organismes que es reprodueixen sexualment, i, de cada parella d'homòlegs, cada progenitor n'ha aportat un.

Quan només hi ha una dotació cromosòmica el cariotip es diu que és **haplont**. Si n'hi ha tres dotacions homòlogues, **triplont**, si n'hi ha quatre, **tetraplont** i, si n'hi ha més, **poliplont**.

2.3.2 Mitosi i meiosi

Les cel·lules següeixen

2.3.3 Reproducció i sexualitat. Cicles biològics

Sexualitat i reproducció són dos procesos amb origen evolutiu i funció diferent. Si bé l'objectiu de la reprodució era obtindre individuus que siguin còpies genètiques dels seus progenitors, l'objectiu de la sexualitat és el de la recombinació genètica per provar generar mes variabilitat genètica.

Alguns organismes unicel·lulars, per exemple, tenen comportaments sexuals no lligats a la reproducció com ara la conjugació que consisteix en l'intercanvi simple de material genètic sense que es produeixi cap nou individuu.

Dels procesos sexuals en resulta una gama d'individuus diferents molt més amplia del que en resulta amb processos exclusivament asexuals. Això dona més agilitat al procés evolutiu, sobretot de cara a variacions en el medi. Permet que, si un organisme adquireix per mutació una característica positiva, aquesta característica es propagui per la població sense necessitat de que hi hagi una substitució dràstica de la descendència sense mutació per la descendència amb mutació.

Als organismes que es reprodueixen sexualment, a la fecundació, sempre intervenen dos cèl·lules haplonts (gametes), una de cada progenitor, que es junten per formar un zigot diplont. Pot passar que la meiosi es produeixi abans de la fecundació, i que els individuus madurs siguin

diplonts (**cicle diplont**) o que la meiosis es produeixi després de la fecundació amb la qual cosa els individuus son haplonts (**cicle haplont**).

Alguns organismes alternen generacions haplonts amb les diplonts (**cicle haplodiplont**. Una generació haplont produeix les gametes que intervenen a la fecundació, formant un zigot que dona un individuu diplont. Aquest crea cèl·lules esporogènees (diplonts) que per meiosi dona lloc a meiospores haplonts de les que es torna a formar un individuu haplont.

2.3.4 Mutacions cromosòmiques

El fet de que els gens estiguin dins de l'estrutura cromosòmica dona lloc a altres tipus de mutacions que no són pas les gèniques. Ténen a veure amb la distribució dels gens a dins dels cromosomes, i, així com les mutacions gèniques explicaven l'origen dels diferents genotips, les mutacion cromosòmiques expliquen l'origen dels diferents cariotips per cada espècie.

Les **mutacions cromosòmiques estructurals** són aquelles en les que no intervenen cromosomes integres sino fragments d'aquests.

- Es produeix una **deficiència o delecció** quan un cromosoma en perd un segment. Les causes poden ser una falla en el procés de replica o, en el moment del crossover.
- La **duplicació** consisteix en el fet de que un segment cromosòmic es dupliqui, generalment, en sèrie.
- La **translocació** és una mutació cromosòmica que consisteix canviar el locus d'una seqüència gènica. Si un organisme es reprodueix sexualment, sovint, la translocació causa, en els descendents, duplicació o deficiencia del gen translocat.
- La **inversió** consisteix en el canvi de sentit d'un segment de cromosoma.

De banda de les mutacion estructurals, es donen **mutacions per canvi en el nombre de**

cromosomes. La causa del canvi en el nombre de cromosomes pot ser:

- **Fusió cèntrica:** Dos cromosomes no homòlegs es fusionen pel seu centròmer.
- **Escissió cèntrica:** Un cromosoma se escindeix en dos pel seu centròmer.
- **No disyunció en la meiosi:** En la meiosi no es reparteixen per igual les cromàtides de tal forma que una gameta es queda amb més cromàtides del normal i l'altre, amb menys. Es tracta d'una **euploidia** si és un canvi en el nombre de dotacions gèniques. Per exemple, que d'un diploide en sorgeixi un triploide. Pel contrari, si el que hi ha hagut és la falta o la duplicació d'un sol cromosoma, es tracta d'una **aneuploidia**

2.3.5 Sumari de conceptes aplicables al projecte

La influència del locus de cada gen en el fet de que dos gens tendeixin a heretar-se junts, és un efecte que, als algorismes genètics clàssics i amb segons quins problemes, resulta negatiu. A la natura és un efecte positiu per que la posició dels gens no es fixa sino que evoluciona juntament amb els individuus i, si dos gens són bons si s'hereten junts, l'evolució tendirà a posar-los junts en el cariotip, i, si fós bo que es recombinin de forma equitativa, l'evolució tendirà a posar-los separats.

A la simulació, de cara a deixar via oberta a diversos comportaments sexuals o a organismes asexuals, farem servir la separació entre els conceptes de sexualitat i reproducció.

La introducció d'organismes diplonts o poliplonts suposaria implementar un mecanisme de resolució del fenotip heterozigòtic als gens homòlegs. També sorgirien alguns altres problemes, relacionats amb el creuament, que es comenten en el següent apartat.

2.4 Genètica biomol·lecular

2.4.1 Concepte clàssic de gen

Fins ara, hem considerat un gen com quelcom indivisible. Abans de coneixer a fons l'estructura mol·lecular de l'ADN, els biòlegs consideraven el gen com a:

- **Unitat funcional:** De cara a controlar un caràcter.
- **Unitat de reconbinació:** Unitat estructural bàsica e indivisible del cromosoma.
- **Unitat de mutació:** El gen és el que canvia com un tot.

Més tard, van descobrir que els cromosomes estaven formats per seqüències d'ADN. L'estudi mol·lecular de l'ADN va donar una idea més en detall de com s'expressen, com es reconbinen i com muten els gens.

2.4.2 Estructura mol·lecular de l'ADN

monocatenario bicatenario circular linial

Model de Watson i Crick (Heleicoidal)

direccional, si és bicatenari cada cadena és complementaria però, la direcció de transcripció és inversa.

	Púriques		**Pirimídiniques**
Bases al ADN	Adenina	lliga amb	Timina
	Guanina	lliga amb	Citosina

	Púriques		**Pirimídiniques**
Bases al ARN	Adenina	lliga amb	Uracil
	Guanina	lliga amb	Citosina

2.4.3 Expressió gènica

L'expressió gènica és la forma que tènen els gens, continguts en els cromosomes, per arribar a afectar al caràcter fenotípic que controlen. En general, cada gen contingut als cromosomes està associat a la producció d'una proteina que pot ser enzimatica o estructural. A més, aquesta associació és linial, com ja veurem, això vol dir que

A la natura, l'expressió gènica no es fa directament de l'ADN a les proteines sinó que es fa servir unes cadenes d'ARN com a intermediaries. En conseqüència la expressió gènica es fa en tres passos:

Transcripció

La transcripció és el primer pas de l'expressió gènica i l'únic en el que pren part l'ADN. Consisteix en sintetitzar una cadena d'ARN complementària a una subseqüència d'ADN. Es divideix en tres fases:

- **Fase d'iniciació:** Primer, l'enzim transcriptor reconeix una seqüència de nucleòtids, anomenada **regió promotora**, que és la que indica el punt de la cadena on cal començar una transcripció.
- **Fase de allargament de cadena:** Després, a mida que avança en la direcció de transcripció, va enganxant nucleòtids d'ARN complementaris als que es va trobant a la cadena d'ADN.
- **Fase de finalització:** El procés arriba a la seva fi, quan l'enzim arriba a una seqüència d'ADN determinada, **regió terminadora**, que indica la seva fi.

Processat o maduració

Les cadenes d'ARN$_m$ inmadur, a les cèl·lules eucariotes (amb nucli diferenciat), pateixen tot un seguit de transformacions abans de traduïr-se als ribosomes. Aquest processat no es fa als organismes procariotes (nucli dispers) donat que ARN$_m$ es tradueix directament, abans, i tot, de acabar-se de transcriure.

Per un costat, la seqüència d'ADN que hi ha transcrita al ARN$_m$, no tota conté informació útil. Els **exons** són els segments que codifiquen informació útil, i els **introns** són els segments que no. Una de les transformacions que es fan en aquesta fase és eliminar els introns.

Una altra transformació és afegir al lider i al trailer un cap i una cua respectivament que ajudaran a iniciar i finalitzar la traducció.

Algunes mol·lecules d'ARN (ARN$_t$, ARN$_r$...) que també es transcriuen de l'ADN, pateixen un processat més especialitzat. Aquestes mol·lècules d'ARN no es fan servir per codificar proteines però, com es veurà en la fase següent, tenen un paper importantísim en la síntesis de proteines.

Traducció

Durant la traducció, l'ARN$_m$ madur s'interpreta per anar enganxant la seqüència d'aminoàcids d'una proteina.

Cada tres nucleòtids d'ARN$_t$ formen un codó. Cada codó té un aminoàcid associat, segons la taula de sota. La concatenació dels aminoàcids segons la seqüència de codons és el que forma la proteina.

Els quatre valors possibles pels tres nucleòtids d'un codó donen $4^3 = 64$ combinacions. Però, a la natura, es donen només 21 aminoàcids. Es dedueix, llavors, que **la codificació és redundant** i produeix **sinònims**. Un parell o tres de codons són **muts** i no tenen traducció. Tot i no tenir un aminoàcid associat, veurem que els codons muts són molt útils.

Aquesta taula és el codi genètic que tradueix els codons a aminoàcids:

L'associació la fan mol·lècules d'ARN$_t$ que tènen a un extrem un anticodó per lligar-se a un codó, i, per l'altre extrem un radical amb el qual es lliguen a l'aminoàcid corresponent. Les mol·lècules d'ARN$_t$ també estan codificades a l'ADN i, per tant, l'associació codó-aminoàcid és també informació genètica. Sorprenentment, el codi genètic és pràcticament universal, donant-se petites variacions, només a l'ADN mitocondrial. La raò és que en els organismes mínimament evolucionats, un canvi en el codi genètic suposaria tants canvis que segurament serien letals, en canvi en els organismes primigenis, seria possible trobar més diversitat de codis. [1]

2.4.4 Regulació de l'expressió dels gens

Mutacions i creuaments són els mecanismes d'adaptació al medi que permeten a una població adaptar-se de generació en generació als canvis graduals en el medi. Però, hi ha canvis que són tan freqüents que els ha d'afrontar l'individuu que els pateix i no es pot esperar a que varii el genotip de la seva descendència. Són les respostes que produeix l'individuu a les variacions del medi.

Perque un organisme es pugui adaptar a diverses situacions de l'entorn cal que tingui mecanismes per regular l'expressió dels seus gens segons aquestes situacions. Per exemple, si el medi es torna massa àcid cal generar enzims que ho compensin, però, quan es massa bàsic, la producció d'aquests enzims, no només és un gast inútil d'energies sinó que podria ser contraproduent.

Cal llavors un mecanisme que permeti a l'organisme controlar la seva producció enzimàtica. Aquest control no es podria fer efectiu, si l'ARN$_m$ no tingués una vida molt limitada. La vida de l'ARN$_m$ i la vida de la majoria de proteines enzimàtiques, ve fixada per un compromís entre economia en la seva producció i la capacitat de reacció de l'organisme.

[1] Això dòna una idea de en quin punt de l'evolució, les mitocondries passaren simbiòticament a formar part dels altres organismes

Aquest límit en la vida del ARN_m i de les proteines que en genera, ens permet una regulació basada en el control de la producció d'ARN_m. Si es deixa de produir, hi haurà un moment en que les proteines que genera no hi seran presents. A continuació s'explica els factors que intervenen en la síntesis d'ARN_m per a un gen donat.

El gens tenen una **probabilitat transcripció** que depen de l'afinitat de l'enzim transcriptor amb el seu promotor i de la seva repetició al llarg del genotip. Aquesta probabilitat és inherent al genotip, però, pot ser modificada amb **regulació activa**. Segon l'efecte de la regulació diem que el control que fa és:

- **Control negatiu:** Si es fa mitjançant **agents represors** que impideixen l'unió de l'enzim transcriptor amb el promotor bloquejant la síntesi de ARN_m.
- **Control positiu:** Si es fa mitjançant **agents activadors** que es junten amb el promotor per fer-lo més afí amb l'enzim transcriptor.

Els agents represors/activadors són proteines que es sintetitzen també a partir de l'ADN i que actuen o no, segons les condicions d'ambient. Quan aquestes condicions ambientals són principis actius es diu que són:

- **Sistemes inducibles:** Si actuen per la ausencia d'un principi actiu.
- **Sistemes represibles:** Si actuen per la presència d'un principi actiu (correpressor o coactivador).

Per exemple, si el gen que volem regular és un catabolitzador d'una sustància A, l'organisme pot fer servir un control negatiu induible, de tal forma que es desactivi quan no hi hagi A, i/o un control positiu represible, perque s'acceleri la producció quan hi hagi A.

2.4.5 Mutació i creuament a nivell mol·lecular

Com que amb la genètica mol·lecular hem vist que la unitat de convinació no era el gen sinó el nucleòtid, cal reformular els conceptes de mutació gènica i cromosòmica.

La mutació puntual és causada per un canvi en les bases dels nucleòtids de l'ADN, deguda a l'inestabilitat química del propi ADN o a agents externs.

- **Mutació per transició de bases:** Intercanvi per l'altre base del mateix grup (púriques o pirimírique)
- **Mutació per transversió de bases:** Intercanvi per la base no complementaria de l'altre grup.
- No es donen, de forma natural, les mutacions directes entre bases complementaris.

Tant la transició com la transversió, es donen per la modificació química d'una de les bases (tautomerització) que, en duplicar-se l'ADN, no es lliga amb la seva base complementària sinó amb una altra. Quedant les bases desajustades. Encara cal que passi per uns quants filtres perque la mutació es faci efectiva a la descendència:

- Existeixenn mecanismes de reparació basats en el fet de que, tot i que una base hagi canviat, l'altre pot seguir igual, si no són complementàries, vol dir que hi ha hagut una mutació. L'enzim corrector modifica una de les dues bases per tal de que siguin complementàries, però, pot modificar la mutada o la bona.
- En duplicar-se la cadena, per una divisió cel·lular, si encara no s'ha 'corregit´ la mutació, la meitat de la descendència durà la mutació complerta i l'altra meitat el genoma primitiu.
- Tant si es repara com si es queda la mutació sense reparar, la probabilitat de traspasar-la a un descendent és del 50%.

- La mutació es pot produir a un segment d'ADN no codificant (introns i zones intermitges)
- Els codons sinònims fan que alguns canvis en les bases no impliquin canvi de pèptid.
- Alguns canvis de pèptids als extrons tampoc no són significatius pel fenotip.
- En organismes pluricel·lulars, cal que sigui una mutació al teixit germinal que dona lloc als nous individuus. Si es dona al teixit somàtic, només afecta a les cèl·lules filles al mateix organisme.

2.4.6 Sumari de conceptes aplicables al projecte

A nivell mol·lecular, trobem alternatives molt riques al AG clàssic. Algunes de les quals han estat estudiades en la bibliografia.

El gens són de longitud variable. El que ho permet són les seqüències promotora i terminadora que els delimiten. Mayer va experimentar aquest tipus de codificació, mitjaçant promotors i terminadors, en cromosomes de longitud fixa, variant el nombre, posició i longitud dels gens/paràmetres. Raich va trobar ideal una codificació molt semblant (feia servir una longitud en comptes d'una seqüència terminadora) per problemes orientats a disseny que ténen un nombre variable de paràmetres i que requereixen solucions obertes.

No tot l'ADN es transcriu, com a mínim les seqüències promotores i terminadores, no ho fan. A més, tenim els introns que s'eliminen durant la fase de maduració.

2.5 Genètica de poblacions

2.6 Ecologia

Part III

Part pràctica

Capítol 3

Disseny de l'aplicatiu

3.1 Estructura del medi

3.1.1 Visió general

El present apartat descriu el funcionament del medi on viuran els organismes i alguns detalls de disseny i d'implementació. De cara a permetre ampliar fàcilment el model, hem volgut fer un disseny del medi que permeti adaptacions a futures necessitats de modelatge. Per un costat es descriu el model genèric i, per un altre, es descriuen les particularitzacions que s'han implementat per a aquest projecte.

La generalitat del model s'ha volgut limitar al conjunt de biòtops discrets. Això implica dues coses:

- Les posicions dins del biòtop estan quantitzades. No hi ha més que un nombre limitat de posicions a diferència de l'espai continu real.
- Tot i que podem considerar una posició discreta com a representant d'una zona limitada del substrat, les propietats dins d'aquesta zona del substrat són uniformes.

Aquesta limitació juntament amb la discretització del temps és comuna a la majoria de sistemes

artificials. L'única forma de limitar aquest efecte és fer que el pas de quantització sigui tan petit que el seu efecte quedi minimitzat.

El nostre model general és un model que consta de posicions discretes amb les seves corresponents propietats i que tènen relacions de veinatge amb les altres posicions segons una topologia.

Així doncs, tenim dos elements que podem modelar-los independentment.

- **Posicions del substrat:** Elements discrets que indiquen les propietats d'una zona del biòtop.
- **Topologia:** Controla les relacions de veinatge i la identificació de les posicions per part de la resta del sistema.

Combinant aquests dos elements, es pot obtindre un conjunt molt ric de biòtops.

3.1.2 Topologies

La topologia determina les relacions de veinatge entre les cel·les. Si les posicions del biòtop fòssin nodes d'un graf, la topologia representaria els vertexs que els uneixen.

Per exemple, podem adaptar la topologia per convertir-la en una topologia 2D, molt vàlida per simular biosistemes terrestres sense estratificar, o podem adaptar-ho a una topologia 3D que és més realista per simular medis fluids, com l'aigua o estratificats com els boscos.

Com que el nombre de cel·les és limitat, el conjunt de cel·les formaran una regió limitada. La topologia ha de determinar quines són les veïnes de les cel·les de les vores. Com a exemple considerem una topologia 2D rectangular. Si féssim que les cel·les limítrofes no tinguin veïnes més enllà dels límits ens trobarem davant d'una regió limitada. Si fem que les cel·les d'un dels costats es conectin amb les cel·les del costat oposat, obtindrem una topologia de superfície cilíndrica. Si fem el mateix amb tots quatre costats obtindrem una topologia de superfície toroidal.

Per a aquest projecte, s'ha implementat una topologia toroidal perque, en principi es vol controlar la complexitat del sistema i, un biòtop amb topologia limitada n'afegiria donat que introdueix situacions a les que hauria de fer front l'organisme, per exemple, quan està en un límit i quan no hi està. En una topologia toroidal no hi ha vores i per tant els organismes no s'hi han d'enfrontar a aquest element de complexitat.

Per facilitar la implementació, hem fet que la veina directa d'una posició a l'estrem dret sigui una posició a l'estrem esquerre però, no pas la que està a la mateixa línia sinó la que està una línia abaix. L'única diferència que introdueix això és que anant sempre a la dreta o a la esquerra, sense variar la direcció podem recòrrer tot el biòtop.

Cada posició té 8 cel·les veines directes. Un desplaçament a qualsevol d'aquestes 8 cel·les es pot codificar amb 3 bits.

100	000	001
101	Origen	010
110	111	011

La concatenació de N desplaçaments bàsics aleatoris tendeix a formar una distribució normal entorn al centre. Com els vectors de desplaçament tenen 32 bits podriem codificar fins a 10 desplaçaments bàsics consecutius en un vector de desplaçament. Però, com ens serà molt útil poder activar i desactivar cada desplaçament bàsic, farem servir un quart bit per a cada bàsic per dir si està habilitat o inhibit el desplaçament, i en un vector hi caben, doncs, 8 desplaçaments bàsics.

h0	d0	h1	d1	h2	d2	h3	d3	h4	d4	h5	d5	h6	d6	h7	d7
1	101	1	101	1	010	1	110	1	110	1	110	1	110	1	110

Si cal considerar cap ordre en el càlcul dels desplaçaments bàsics, es fa de més significatiu a menys.

La codificació dels desplaçaments bàsics s'ha fet de tal manera que, si invertim bit a bit un vector de desplaçament a excepció dels bits d'habilitació, obtenim un desplaçament invers. És

a dir, donada la direcció *desp*, (desp XOR 0x77777777) en dòna la direcció inversa, la qual cosa serà molt útil de cara a afavorir l'aparició de conductes d'evasió.

L'altre funció important de la topologia és assignar a cada cel·la un identificador únic dins de la topologia. La resta del sistema farà servir aquest identificador per referenciar una posició i, en cas de necessitar-ho, demanarà a la topologia quin és el substrat per aquesta posició.

En la present implementació s'han fet servir enters del 0 a N com a identificadors de les posicions, on N es el nombre de cel.les. Aquests identificadors ens permeten fer els desplaçaments de forma molt optima si assignem el números per ordre a les cel·les de cada fila. Només cal afegir (o treure), a l'identificador de la posició origen, un número, que depen del desplaçament, i ajustar el resultat en cas de sortir-se de límits, per calcular l'identificador de la posició destí.

En resum, una topologia ofereix els següents serveis als seus clients:

- Donar l'identificador de la posició destí a partir de l'identificador d'una posició origen i d'un desplaçament.
- Donar l'identificador d'una posició escollida aleatoriament.
- Determinar si un identificador donat és vàlid dintre de la topologia.
- Accedir al subtrat corresponent a un identificador de posició.
- Donar el desplaçament que apropa una posició d'origen donada a una posició destí per l'itinerari més curt.

3.1.3 Substrats

Un substrat particularitza les propietats del medi a una posició. El substrat pot tenir diverses propietats segons la complexitat que desitjem per al medi.

Les dos propietats més importants del substrat són qui ocupa el substrat, i quins nutrients hi han.

Ocupants

S'ha restringit l'ocupació de les posicions per part dels organismes a un sól organisme per posició i a una sola posició per organisme. La raó ha estat fer possible referenciar un organisme per la seva posició. Això simplificarà, a les relacions entre organismes, com referir l'altre organisme. També, com a avantatge adicional, simplifica la interfície amb l'usuari per seleccionar un organisme seleccionant la seva posició.

Nutrients

En quant els nutrients que hi pot haver en una mateixa posició del substrat, cada posició tè un nombre de nutrients màxim definit. Quan s'afegeixen nutrients per damunt d'aquest nombre, els nutrients més antics desapareixen.

Els nutrients estan diferenciats qualitativament amb un sencer que indica el seu tipus.

La recollida de nutrients, es fa amb un patró de cerca pel tipus de nutrient i una tolerància a nivell de bit segons la funció de compatibilitat estandard. Com s'explica a l'apartat 4.3, aquesta funció retorna cert si

```
((Patro^Clau)&Random&~Tolerancia)==0
```

de tal forma que un 1 a una posició de la tolerància significa que, encara que no es correspongui aquest bit no es tindrà en compte. La cerca es fa des dels més nous fins els més vells.

Movilitat

3.2 Agents

3.2.1 Trets generals

Els agents són objectes que disparen una acció determinada quan són cridats. La majoria d'accions es produeixen sobre el biòtop, sobre la configuració d'altres agents o sobre paràmetres globals de configuració.

El paper dels agents a dins del sistema és permetre a l'usuari controlar com evolucionarà l'entorn on es mouran els organismes de cara a obtindre resultats que s'hi puguin contrastar. Per sí mateix, el conjunt d'agents implementats permet configuracions molt complexes, però, a més, ofereix un seguit d'eines molt útils per que l'usuari-programador pugui ampliar aquests agents.

Quan un agent es requerit per realitzar la seva acció, es diu que l'agent ha sigut **accionat**. Quan algú requereix un valor contingut en l'estat de l'agent es diu que l'agent ha sigut **consultat**.

Direm que un agent A té com a **subordinat** a un altre agent B si és A qui acciona a B. Ho representarem així: $A \rightarrow B$. L'estructura de subordinació ha de ser un arbre on els subordinats són els fills d'aquell a qui es subordinen.

Direm que un agent A és **depenent** d'un altre agent B si A, quan és accionat, necessita consultar l'estat de l'agent B. Ho representarem així: A(B). La consulta no ha d'implicar cap modificació ni recàlcul d'estat en l'agent consultat. Això permet que no hi hagi restriccions en l'estructura de dependència i que puguin existir dependències creuades. [1]

Tots els agents duen un nom assocciat que, per defecte, coincideix amb un prefixe i un número de sèrie únic entre tots els agents. Aquest nom es pot canviar per un que sigui més mnemotècnic per a l'usuari.

[1] Tot i així, és important preveure que l'ordre d'accionat entre agents interdependents podria implicar variacions en els resultats.

3.2.2 Agents Subordinadors

Els agents subordinadors són agents que, quan són accionats, accionen tot un seguit d'agents subordinats.

Agents Múltiples

L'agent múltiple acciona una i només una vegada cadascun dels agents subordinats cada cop que és accionat.

Agents Temporitzadors

Els agents temporitzadors són agents multiples que no sempre que reben un accionat el propaguen cap els subordinats. Estableixen dos periodes, un actiu i un altre inactiu. Els accionats només es propagen als subordinats durant el periode actiu.

Els periodes es defineixen mitjançant tres paràmetres: El periode mínim és el número d'accionats que dura el periode com a mínim. Aquest mínim es pot augmentar de forma no determinística el resultat de sumar-li n vegades un número aleatori en l'interval [0,m] (els corxets indiquen que els extrems estan inclosos). n és el número de daus, i m és el valor màxim o magnitud del dau.

D'aquesta especificació es pot deduir algunes dades, pot ser, més intuitives per a l'usuari:

- El valor màxim que pot adoptar el periode és el mínim més n*m
- Un sól dau equival a una distribució uniforme entre els límits
- A mesura que incrementem el nombre de daus, la distribució dels periodes s'aproxima a una distribució normal entorn al centre entre el valor màxim i mínim, amb un desviació típica cada vegada menor.

Per defecte, els paràmetres que introdueixen indeterminisme en els temporitzadors, com són els daus, estan ajustats de manera que el seu efecte sigui nul. Si no es toca res més que els periodes mínims, actuarà de forma determinista. De la mateixa manera, els periodes mínims dels cicles estan ajustats, per defecte, per que sempre s'estigui en un cicle actiu. D'aquesta forma, si no es configura res, l'efecte d'un temporitzador és el d'un agent múltiple ordinari.

Paràmetres per defecte i una execució:

- Cicle Actiu (minim=1 daus=0 magnitud=0)
- Cicle Inactiu (minim=0 daus=0 magnitud=0)
- Periode Actual (actiu)
- Periode Restant (1)

Paràmetres ilustratius:

- Cicle Actiu (minim=0 daus=2 magnitud=3)
- Cicle Inactiu (minim=4 daus=0 magnitud=0)
- Periode Actual (actiu)
- Periode Restant (1)

A continuació hi ha una execució dels paràmetres anteriors. Els guions representen accionats durant el periode inactiu i les O's representen accionats durant el periode actiu.

```
OO----OOOOO----OO----O----OOOO----O----OOO----OOOO----OOO----OOOO----O----OO----
OOO----OOOO--------OOO----O--------OOO----OOOO----O----OOOO----OOOO--------OOOO-
---OOO----OOO----OOO----OOOOO----O----OOOOOO----OOOO----O----OOOO----OOO----OO--
--OO----OOOO----OOO----OOO----OOOOO----OOO----OOOO----OOO----OO----OO----OO----O
OOOOO----OOO----OOOOO----OOOO----OO----OOOOO----OOOO----OOO----OOO----OOOOO----O
```

Agents Probabilitzadors

Els agents probabilitzadors són també agents múltiples que controlen si l'accionat es propaga cap els subordinats o no. Però, a diferència dels agents temporitzadors, ho fan mitjançant una llei probabilística. Si es dòna la probabilitat, s'accionen els subordinats, si no es dòna, no s'accionen.

La probabilitat es defineix amb el nombre de vegades que es donaria la probabilitat en un tamany de mostra. Per exemple, podem definir una probabilitat dient que es dòna 3 de cada 14 vegades. 14 és el tamany de mostra i 3 les vegades que es donaria en la mostra.

Els paràmetres estan ajustats per defecte a valors que fan del probabilitzador un agent múltiple ordinari.

Paràmetres ilustratius:

- Probabilitat (mostra=40 encerts=25)

A continuació hi ha una execució dels paràmetres anteriors. Els guions representen accionats en els quals no s'ha donat la probabilitat i les O's, accionats en els quals sí s'ha donat.

```
OOOO-O-O-O-OO----OOOO-OOOOOOOO-OOO-OOOOOOOOOOO-OOOOOOOOO---O-OOOOOOO-O--OOOO-OO-
OOO-OOO-O-OOOO-OOO-O-OOO-OOOOOOOOOOO-OOOOOOO-OOOOOO--OOO-OOO-OO-O-OOOOO--O---O-
OO--O---OOO-OOOOOOOOO-O-OOO-O-OOO-O--OOOOOOOO-OO-O-OO-OOO-OO-OOOO-OOOOOO-OOO-OOO
--OOOO-OOOO-O-OOO-OO-OO-O--OO-OOOO--O--O-O-OO--OO-O-O-OOO---O-O-OO------OO-OO-O-
OOO-OO-O-OO-OO-OOOOO----OO-OO-----O-OOOOO--O---OOOOOOOOOO-O-OO-O-OOOOO---OOO-OO-
```

Agents Iteradors

Els agents iteradors són agents múltiples que no limiten els accionats que arriben als seus subordinats, sinó que el que fan és multiplicar els accionats que li arriben.

Més concretament, quan un agent iterador és accionat, els seus subordinats, són accionats un número de vegades que es calcula a partir d'un mínim i uns daus com els que feiem servir per als periodes dels temporitzadors.

Per defecte, la part indeterminística (els daus) no té cap efecte, i la part deterministica (el mínim) està posada a un valor (1) que el fa equivalent a un agent múltiple ordinari.

Paràmetres ilustratius:

- Iteracions (minim=2 daus=2 magnitud=4)

A continuació hi ha una execució dels paràmetres anteriors. Els parèntesis agrupen els accionaments dels subordinats que es fan sota un mateix accionament de l'iterador.

```
(OOOOOO)(OOOOOOOO)(OOOOOOO)(OOOOO)(OOO)(OOOOOOO)(OO)(OOOOOOO)(OOOOOOO)(OOOOOO)(O
OOOOOOO)(OOOOO)(OOOOOO)(OOO)(OOOOOO)(OOOO)(OOOOOO)(OOOOOO)(OOO)(OOOOOO)
```

Amb dos accions subordinades una execució quedaria com això:

```
(OEOEOEOEOEOEOEOEOEOE)(OEOEOEOEOEOEOEOEOE)(OEOEOEOEOEOEOE)(OEOEOEOE)(OEOEOEOEOEO
EOEOE)(OEOEOEOEOEOEOEOE)(OEOEOEOEOEOEOEOEOE)(OEOE)(OEOEOEOEOEOEOE)(OEOEOEOEOEO
EOE)(OEOEOEOEOEOE)(OEOEOEOEOEOE)(OEOEOEOEOE)(OEOEOEOEOEOEOE)(OEOEOEOEOEOE)(OEOEO
EOEOEOE)(OEOEOEOEOEOE)(OEOEOEOEOE)(OEOEOEOEOE)(OEOEOEOEOE)
```

on les E's representen l'execució de la segona acció subordinada.

3.2.3 Agents Posicionadors

Els agents posicionadors controlen una posició en la topologia del biòtop. No tènen subordinats, però, generalment hi ha agents que en depenen del seu valor i segons el tipus de posicionador per recalcular la seva posició fan servir altres agents dels quals depenen.

Posicionador Bàsic: No modifica la seva posició si és accionat. A menys que, per configuració, es fixi a una posició concreta, s'inicialitza amb una posició aleatoria vàlida dins de la topologia,

Posicionador Aleatori: Cada cop que és accionat pren una posició aleatoria vàlida dintre de la topologia.

Posicionador Zonal: Cada cop que és accionat pren una posició aleatoria dintre d'una zona. La zona es defineix per una posició central, determinada per un altre posicionador de qual depen, i un radi, que no és més que el nombre de desplaçaments aleatoris que es fan a partir d'aquesta posició central per trobar la posició final. Les posicions tendeixen a adoptar una distribució normal en l'entorn de la posició central.

Posicionador Seqüencial: Cada cop que és accionat pren la posició del següent posicionador que hi ha en una seqüència de posicionadors. Els agents posicionadors de la seqüència són dependència del posicionador seqüencial.

Posicionador Direccional (Itinerari): Cada cop que és accionat pren la posició que en resulta d'aplicar-li un desplaçament a la posició anterior. El desplaçament el determina un agent direccionador que és dependència. Els agents direccionadors s'expliquen al següent apartat.

A continuació es presenten exemples d'execució d'un posicionador aleatori, un de seqüencial i un de zonal aplicats sobre una topologia toroidal.

3.2.4 Agents Direccionadors

Els agents direccionadors controlen una direcció per calcular desplaçaments dintre de la topologia del biòtop. La seva utilitat principal radica en controlar la posició d'un posicionador de tipus direccional, però, no es descarten altres aplicacions futures.

Direccionador Bàsic: No modifica la seva direcció si és accionat.

Direccionador Aleatori: Cada cop que és accionat pren una direcció aleatoria.

Direccionador Seqüencial: Cada cop que és accionat pren la direcció del següent direccionador que hi ha en una seqüència de direccionadors. Els agents direccionadors de la seqüència són dependència del direccionador seqüencial.

A continuació es presenten exemples d'execució d'un posicionador direccional, que depen de diferents tipus de direccionadors.

Per obtindre aquests resultats ha calgut fer servir subordinadors per que el posicionador s'accionés més que el direccionador. La topologia de l'exemple és toroidal.

3.2.5 Agents Actuadors

Els agents actuadors són els que finalment modifiquen el substrat. Els actuadors depenen d'un agent posicionador que els indica la cel·la que han de modificar.

La majoria dels agents que hem vist fins ara eren molt independents davant de les modificacions en la topologia i en la composició del substrat que es puguin fer més endavant. Les especialitzacions dels actuadors, en canvi, han de dependre per força del substrat i la seva composció, perqué actuen sobre ell. Segueixen sent independents, però, de la topologia.

Aquí a sota, expliquem alguns actuadors vàlids pel substrat implementat en aquest treball.

Agents Nutridors

Aquests actuadors depositen nutrients al substrat. Un tipus de nutrient es codifica amb un enter de 32 bits sense signe. El tipus de nutrient que es depositarà s'especifica amb dos nombres enters de 32 bits sense signe. El primer enter indica el número del tipus bàsic, i el segon indica els bits del tipus bàsic que poden variar aleatoriament.

Per exemple, el parell $\begin{matrix} \text{element bàsic:} & \text{0x0000000000000000} \\ \text{variabilitat:} & \text{0xFFFFFFFF00000000} \end{matrix}$ genera elements que tenen la part baixa igual que l'element bàsic (a zero) i la part alta al atzar.

Agents Desnutridors

Els desnutridors són molt semblants als nutridors, però, en comptes de afegir nutrients, en treuen. Es pot treure selectivament cercant un element químic que s'apropi al que s'indica o es pot especificar una tolerància per a certs bits.

3.2.6 Arxius de configuració d'agents

Motivació i criteris de disseny

De cara a poder passivitzar un biosistema a disc per poder-ho restaurar posteriorment, caldria també poder passivitzar i restaurar l'estat dels seus agents. L'arxiu de configuració d'agents és un arxiu de text que conté l'estat i l'estructura dels agents d'un biosistema, que es pot extreure en un moment donat i restaurar-ho posteriorment.

Els agents a un biosistema, com s'ha dit abans, formen una estructura d'arbre segons les seves relacions de subordinació. Cada arxiu de configuració conté un arbre d'agents subordinats partint d'un agent arrel. Seria possible penjar tota l'estructura d'un arxiu de configuració i subordinar-ho a un agent d'una estructura ja existent a un biosistema. Es podria formar una mena de biblioteca d'arxius amb configuracions comunes que es podrien convinar per muntar ràpidament l'estructura d'agents d'un biosistema.

Estructura

Els arxius de configuració d'agents ténen una estructura molt simple: Primer van unes linies de text que determinen, per cada agent, el seu nom i el seu tipus. Un cop definits els noms i els tipus, es configuren els paràmetres per a cada agent.

La definició dels noms i els tipus es fa amb una línia per cada agent sent la primera línia la que defineix l'agent que està en l'arrel de la estructura de subordinació. A cada línia es posa, per

ordre i *separats per espais* un signe asterisc, el nom i el tipus.

Quan es carrega un arxiu de configuració d'agents, si és possible a cada agent se li dòna el nom amb el que apareix a l'arxiu, però això no és posible si ja existeix un amb el mateix nom. En aquest cas, se li dòna un nom per defecte i s'hi tradueixen totes les posteriors referències al nom antic.

Els noms poden contenir qualsevol caracter que no es consideri un espai a C (espais, tabuladors, retorns...).

El tipus s'especifica amb un identificador propi de cada tipus d'agent. Dins d'un arxiu de configuració d'agents, es reconeixen els següents tipus:

Nom de tipus a la memòria	**Nom del tipus a un fitxer de configuració**
Agent Subordinador Multiple	Agent/Multiple
Agent Subordinador Temporitzador	Agent/Multiple/Temporitzador
Agent Subordinador Iterador	Agent/Multiple/Iterador
Agent Subordinador Aleaturitzador	Agent/Multiple/Aleaturitzador
Posicionador Fixe	Agent/Posicionador
Posicionador Aleatori	Agent/Posicionador/Aleatori
Posicionador Zonal	Agent/Posicionador/Zonal
Posicionador Direccional (Itinerari)	Agent/Posicionador/Direccional
Direccionador Fixe	Agent/Direccionador
Direccionador Aleatori	Agent/Direccionador/Aleatori
Actuador Nutridor	Agent/Actuador/Nutridor
Actuador Desnutridor	Agent/Actuador/Nutridor/Invers

Es fan servir identificadors jerarquics molt semblants als que es fan servir a UNIX per identificar els directoris. La jerarquia de noms el que especifica aquí és una jerarquia de tipus i subtipus de tal forma que si, per exemple, a un lloc es requereix *Agent/Posicionador*, aquest lloc el pot ocupar tant un *Agent/Posicionador/Direccional* com un *Agent/Posicionador/Zonal*.

Una definició de noms i tipus podria quedar com segueix:

```
* Agent_0000 Agent/Multiple
* Agent_0001 Agent/Multiple/Temporitzador
```

```
* Agent_0002 Agent/Direccionador/Aleatori
* Agent_0003 Agent/Posicionador/Direccional
* Agent_0004 Agent/Multiple/Iterador
* Agent_0005 Agent/Actuador/Nutridor
```

Aquí, *Agent_0000* seria l'agent arrel.

Un cop definits els noms i els tipus dels agents, cal configurar els seus paràmetres. Per configurar un agent primer cal posar una línia amb el signe + i el nom de l'agent separats per un espai, i, després, tot un seguit de línies de configuració de paràmetres. Les linies de configuració de paràmetres comencen amb un signe menys i el nom del paràmetre i es segueix amb els valors que necessita el paràmetre per configurar-se, tot separat per espais. Com veurem al següent exemple, és normal que un paràmetre s'especifiqui amb diversos valors separats per espais. La posició dels valors acostuma a ser significativa o sigui que és important mantenir l'ordre.

Per configurar un Nutridor es faria de la següent forma

```
+ Agent_0004
- Posicionador Agent_0003
- Composicio 31 0
```

Quan un paràmetre necessita com a valor un altre agent, fa servir els seu nom com a referència.

Al següent apartat, es detalla els paràmetres que controlen cada tipus d'agent.

3.2.7 Paràmetres configurables per a cada tipus d'agent

El que segueix és una especificació de com es configuren els paràmetres dels tipus d'agent implementats mitjançant el fitxer de configuració. Per fer-ho fem servir la següent estructura:

Per a cada paràmetre de cada tipus d'agent es fa una petita explicació i es detallen, ordenats

tal qual han d'aparèixer, els valors que el defineixen.

Els valors dels paràmetres es detallen posant un tipus de dada, dos punts, una petita explicació del valor i, entre parèntesis, les restriccions que s'hi apliquen.

Als agents implementats, els tipus de dada possibles pels valors són:

- **agent:** Nom d'un agent especificat a la definició de noms i tipus
- **uint32:** Senser sense signe codificable en 32 bits i expressat en base decimal
- **id(Alternativa1/Alternativa2...):** Un dels identificadors posats com a alternativa

Després dels paràmetres de cada tipus hi ha un exemple de com quedarien les línies de configuració.

MultiAgent (Agent/Multiple)

Accio: Determina un agent subordinat. Es repeteix tantes vegades com subordinats tingui.

- agent: agent que es subordina (No ha de ser subordinat de cap altre)

Exemple de configuració texte

```
+ AgentMultiple1:
- Accio Posicionador1
- Accio Posicionador2
- Accio Direccionador1
```

Temporitzador (Agent/Multiple/Temporitzador)

Accio: Determina un agent subordinat. Es repeteix tantes vegades com subordinats tingui.

- agent: agent que es subordina (No ha de ser subordinat de cap altre)

CicleActiu: Determina quan triguen els periodes de temps actius

- uint32: periode minim
- uint32: número de daus
- uint32: magnitud dels daus (Van de zero a la magnitud)

CicleInactiu: Determina quan triguen els periodes de temps inactius

- uint32: periode minim
- uint32: número de daus
- uint32: magnitud dels daus (Van de zero a la magnitud)

AntiAccio: Agent subordinat especial que s'acciona en el cicle inactiu (Nomes un per temporitzador i es opcional)

- agent: agent que es subordina (No ha de ser subordinat de cap altre)

CicleActual: Valors del temporitzador quan es reemprengui la marxa

- id(Actiu/Inactiu): cicle actiu o inactiu
- uint32: periode restant del cicle actual

Exemple de configuració texte

```
+ Temporitzador1
- Accio Posicionador3
- CicleActiu 34 2 5
- CicleInactiu 2 4 4
- CicleActual 3 Inactiu
```

Iterador (Agent/Multiple/Iterador)

Accio: Determina un agent subordinat. Es repeteix tantes vegades com subordinats tingui.

- agent: agent que es subordina (No ha de ser subordinat de cap altre)

Iteracions: Determina quantes vegades es repiteixen els subordinats

- uint32: iteracions minimes
- uint32: número de daus
- uint32: magnitud dels daus (Van de zero a la magnitud)

PreAccio: Agent subordinat especial que s'executa un sol cop abans de tot (Nomes un per iterador i es opcional)

- agent: agent que es subordina (No ha de ser subordinat de cap altre)

PostAccio: Agent subordinat especial que s'executa un sol cop despres de tot (Nomes un per iterador i es opcional)

- agent: agent que es subordina (No ha de ser subordinat de cap altre)

Exemple de configuració

```
+ Iterador3
- Accio Posicionador4
- Accio Actuador2
- Iteracions 20 3 6
```

Aleaturitzador (Agent/Multiple/Aleaturitzador)

Accio: Determina un agent subordinat. Es repeteix tantes vegades com subordinats tingui.

- agent: agent que es subordina (No ha de ser subordinat de cap altre)

Probabilitat: La que hi ha d'accionar els subordinats

- uint32: número d'encerts que segons la probabilitat tenderien a donar-se en la mostra
- uint32: número de intents o mostra

ReAccio: Agent subordinat especial que s'acciona si no es dona la probabilitat (Nomes un per temporitzador i es opcional)

- agent: agent que es subordina (No ha de ser subordinat de cap altre)

Exemple de configuració

```
+ Aleaturitzador1
- Accio Posicionador3
- Probabilitat 20 100
```

Posicionador Fixe (Agent/Posicionador)

Posicio: Posició inicial

- uint32: valor de la posició (Ha d'existir a la topologia)

Exemple de configuració

```
+ Posicionador1
- Posicio 12
```

Posicionador Aleatori (Agent/Posicionador/Aleatori)

Posicio: Posició inicial

- uint32: valor de la posició (Ha d'existir a la topologia)

Exemple de configuració

```
+ Posicionador2
- Posicio 23
```

PosicionadorSequencial (Agent/Posicionador/Sequencial)

Posicio: Posició inicial

- uint32: valor de la posició (Ha d'existir a la topologia)

Sequencia : Determina una posició de la seqüència. Es repeteix tantes vegades com calgui.

- uint32: valor de la posició (Ha d'existir a la topologia)

SequenciaActual :

- uint32: el número de seqüència de la següent posició (Si es passa es pren l'ultim)

Exemple de configuració

```
+ Posicionador3
- Posicio 23
- Sequencia 27
- Sequencia 50
- Sequencia 402
- SequenciaActual 2
```

PosicionadorZonal (Agent/Posicionador/Zonal)

Posicio: Posicio inicial

- uint32: valor de la posició (Ha d'existir a la topologia)

Posicionador: Dona la posició central de la zona

- agent: agent posicionador (dependència)

Radi: Nombre de desplacaments que pot fer la posició entorn al centre

- uint32: valor del radi

Exemple de configuració

```
+ Posicionador4
- Posicio 23
- Posicionador Posicionador1
- Radi 3
```

Itinerari (Agent/Posicionador/Direccional)

Posicio: Posició inicial

- uint32: valor de la posició (Ha d'existir a la topologia)

Direccionador: Dona la direcció del desplacament

- agent: agent direccionador (dependència)

Radi: Nombre de desplacaments que pot fer la posició respecte a la posició anterior

- uint32: valor del radi

Exemple de configuració

```
+ Posicionador5
- Posicio 23
- Direccionador Direccionador3
- Radi 1
```

Direccionador (Agent/Direccionador)

Direccio: Direcció inicial

- uint32: valor de la direcció

Exemple de configuració

```
+ Direccionador1
- Direccio 876342
```

DireccionadorAleatori (Agent/Direccionador/Aleatori)

Direccio: Direcció inicial

- uint32: valor de la direcció

Exemple de configuració

```
+ Direccionador2
- Direccio 23442684
```

DireccionadorSequencial (Agent/Direccionador/Sequencial)

Direccio: Direcció inicial

- uint32: valor de la direcció

Sequencia: Determina una direcció de la seqüència. Es repeteix tantes vegades com calgui.

- uint32: valor de la direcció

SequenciaActual: Determina el punt actual de la seqüència

- uint32: el número de seqüència de la següent direcció (Si es passa es pren l'ultim)

Exemple de configuració

```
+ Direccionador3
- Direccio 23
- Sequencia 27
- Sequencia 50
```

```
- Sequencia 402
- SequenciaActual 2
```

Nutridor (Agent/Actuador/Nutridor)

Posicionador: Dona la posició on s'actua

- agent: Agent posicionador (dependència)

Composicio: Determina els elements que es depositen

- uint32: element basic
- uint32: variabilitat, a 1 els bits que poden variar

Exemple de configuració

```
+ Actuador1
- Posicionador Posicionador4
- Composicio 13152450903 0
```

Desnutridor (Agent/Actuador/Nutridor/Invers)

Posicionador: Dona la posició on s'actua

- agent: Agent posicionador (dependència)

Composicio: Determina els elements que s'eliminen

- uint32: element basic
- uint32: tolerancia, a 1 els bits que no importa que coincideixin

Exemple de configuració

```
+ Actuador2
```

```
    - Posicionador Posicionador3
    - Composicio 8943742645 768764258
```

3.2.8 Exemple complert d'arxiu de configuració d'agents

A continuació es presenta un exemple complert:

```
* Agent_0000 Agent/Multiple
* Agent_0002 Agent/Posicionador/Direccional
* Agent_0005 Agent/Multiple/Iterador
* Agent_0004 Agent/Actuador/Nutridor
* Agent_0003 Agent/Posicionador/Zonal
* Agent_0006 Agent/Multiple/Temporitzador
* Agent_0001 Agent/Direccionador/Aleatori

+ Agent_0002
- Posicio 1271
- Radi 1
- Direccionador Agent_0001

+ Agent_0004
- Posicionador Agent_0003
- Composicio 31 0

+ Agent_0003
- Posicio 8
- Radi 1
```

```
- Posicionador Agent_0002


+ Agent_0005
- Accio Agent_0004
- Accio Agent_0003
- Iteracions 20 0 0


+ Agent_0001
- Direccio 2192479406


+ Agent_0006
- Accio Agent_0001
- CicleActiu 1 0 1
- CicleInactiu 5 0 1
- CicleActual 4 Inactiu


+ Agent_0000
- Accio Agent_0002
- Accio Agent_0005
- Accio Agent_0006
```

3.2.9 Disseny del abocat a disc i de la recuperació

3.3 Serveis que donen al biosistema els organismes

De cara a poder manegar

- L'organisme ha d'anar proveint al biosistema de instruccions que l'indiquin les seves accions.

Com es generen les instruccions és qüestió dels propis individuus.

- L'organisme proveeix al biosistema un conjunt de registres que formen el seu fenotip. El biosistema pot modificar-los i consultar-los. L'accés al fenotip no és exclusiu del biosistema sinó que el propi organisme i les seves estructures internes també poden accedir-hi paral·lelament.

- L'organisme ha d'implementar unes funcions vitals que modifiquin l'estat intern de l'organisme als que no tingui accés el biosistema per altres vies. Això no inclou, per exemple, canvis de posició. El biosistema ha de proporcionar tots els paràmetres de les funcions vitals que no siguin interns que ell coneixi: el fenotip del propi organisme o un d'aliè.

- També han d'implementar operadors per tal de crear nous organismes a partir d'altres i organismes aleatoris.

3.4 Estructura interna dels organismes

3.4.1 El cariotip

Definim el cariotip com el conjunt de cromosomes que conté un organisme, doncs en el present cas es consideren organismes unicel·lulars.

Cada cromosoma està format per una seqüència de bases representada cadascuna amb un sencer.

Cariotip-¿Cromosoma-¿Base-¿Codo-¿

3.4.2 El genotip

El genotip és la traducció del cariotip a elements significatius. És el conjunt de gens que s'interpreten

3.4.3 El fenotip

El que anomenem propiament fenotip és un conjunt de 32 registres de 32 bits que té cada organisme. Representen el cos físic de l'organisme. El fenotip es modifica per acció directa del genotip, però també es veu afectat pel medi mitjançant els sensors i, al mateix temps afecta al medi mitjançant els motors. A més, és un dels dos mitjans que tènen els organismes per reconeixer-se juntament amb la detecció de mol·lècules excretades.

3.4.4 Sensors i motors

3.4.5 Presa de nutrients

3.4.6 Metabolisme

Un cop els nutrients estan dins de l'organisme, pot metabolitzar-los, ja sigui per obtindre energia, per obtindre un producte d'excreció o totes dues coses.

L'organisme pot fer ús de l'energia que s'obte de les reaccions durant un temps limitat, si no es consumeix dintre d'aquest temps, aquesta es disipa.

Es preten que les diferents circumstàncies permetin que hi hagi un equilibri

Pot ser, una espècie tendeix a acomular masses nutrients. Si això passes, els seus depredadors tenderien a augmentar en nombre d'organismes i en nombre d'espècies.

3.5 Eines d'analisis

3.5.1 Mecanismes d'especiació

Per que l'usuari pugui extreure una informacio útil, cal que poguem agrupar els individus en espècies. És clar que a la nostra comunitat, al igual que a la natura, l'especiació és un fenòmen que cal que emergeixi, L'espècie, no és quelcom intrínsec a tot individu; és a dir, que el concepte

d'espècie no estarà implementat al codi genetic o als mecanismes de funcionament comuns dels individus sinó que caldrà observar-ho en el seu comportament.

El concepte clàssic d'espècie considera que dos individus són de la mateixa espècie si són capaços de donar descencència fèrtil.

A la biologia moderna es considera que la diferenciació de les espècies no està tant en la capacitat sinó en el fet mateix de reproduir-se. En això pot influir:

- Compatibilitat genètica
- Compatibilitat d'acoplament
- Compatibilitat geogràfica
- Altres

També, cal adonar-se de que totes aquestes consideracions es refereixen als individus que es reprodueixen sexualment i es creuen. Còm es poden identificar les espècies dels individus que es reprodueixen asexualment? En quin punt es considera que dos descendents d'un mateix individus sòn d'una espècie diferent?

A més, degut a la seva qualitat emergent, l'especiació no estarà sempre ben definida. També pot ser que la selecció a l'hora de reproduir-se tingui en compte altres mecanismes que no pas l'especiació.

Tota aquesta conjuntura ha obligat a deixar de banda el concepte d'espècie cap al concepte, una mica més relaxat, de grup reproductiu que, tot i la relaxació, segueix proporcionant informació útil sobre les diferents poblacions del biosistema. Considerem que un grup reproductiu és un conjunt d'organismes que es creuen entre sí o provenen d'ancestres que s'han creuat entre sí dintre d'un cert perióde de temps o d'un cert nombre de generacions.

La política que determina els grups reproductius es basa en un marcatge històric.

Cada individu s'associa amb un taxó que no és més que una seqüència de marques amb diferent antiguitat. Les marques es traspasen idèntiques a la descendència via mitosi.

Cada cert temps, es fa una discriminació que consisteix en fondre les dos marques més antigues de cada taxó en una sola i afegir una marca nova que diferenciarà els individus que fins llavors compartien les mateixes marques i que pendrà importància a mida que adquireixi antiguitat.

Cóm es produirà aquesta discriminació? Imaginem que existeixen individus amb els taxons següents:

A A A A A A A A A A A A A B A A A A A B B A A A A B B A A A B A A A A A	Taxons originals dels individuus població abans de la discretització	⇒	**A** A A A A **A** A A A A **B** A A A A **B** B A A A **B** B A A A **C** A A A A	Fusió dels dos taxons més antics
		⇒	A A A A A **A** A A A A A **B** B A A A A **A** B B A A A **A** B B A A A **B** C A A A A **A**	Discriminació dels individus amb les mateixes marques amb una nova:

D'altra banda, hem de considerar el que passa quan es creuen dos individuus que pertanyen a diferents taxons.

Quan dos individus es creuen, s'asimilen les marques des de la marca més antiga fins a la primera que els diferencia als dos (marca discriminant). Es a dir, a tots els taxons cal revisar les marques. Pot ser es veu més clar amb un exemple. Considerem el següent conjunt de taxons.

A A A A A A
A A A A A B
A A A A B B
A A A B A B
A A A B B A
A A A B B B
A A B A A A

Si es creuen AAAABB i AAABBA, cal considerar equivalemts les subseqüencies de marques AAAA i AAAB equivalents. Tots els taxons que comencin per AAAB els canviem per AAAA sense oblidar-nos de modificar la següent marca més jove que la discriminant amb l'objectiu de que els taxons asimilats mantinguin el sentit.

A A A A A A $\dashrightarrow$ A A A A A A
A A A A A B $\dashrightarrow$ A A A A A B
A A A A B B $\dashrightarrow$ A A A A B B
A A A B A B $\Rightarrow$ A A A *A* *C* B
A A A B B A $\Rightarrow$ A A A *A* *D* A
A A A B B B $\Rightarrow$ A A A *A* *D* B
A A B A A A $\dashrightarrow$ A A B A A A

De cara a la implementació, he considerat separar la major part del procés associat a la determinació de grups reproductius en un objecte independent anomenat taxonomista. Aquest objecte s'encarrega de mantenir els taxons al dia mitjançant una interfície estreta que manté amb el processador.

A dins de la comunitat es manté una informació mínima: l'identificador del taxó al que pertany cada individu. El tràfec d'informació a l'exterior del objecte taxonomista es basa exclusivament en el pas d'aquests identificadors. La interfície oferida permet al processador:

- Incrementar o decrementar la poblacio assignada a un taxó
- Creuar un parell de taxons
- Generar un nou taxó (Per individuus generats espontàneament)
- Envellir les marques i discriminar la població que comparteix el mateix taxó

- Determinar el grau de parentesc entre dos individuus

Quan hem de discriminar o quan fussionem dos taxons, necessitem que la Comunitat i el Taxonomista cooperin.

Quan cal discriminar la població, la Comunitat demana al Taxonomista, per a cada individu, un nou taxó, basant-se en el taxó antic i el número de individus que en queden sense discriminar d'aquest.

El nou taxo duu les marques X.X.X. ... .X.N on N és el número de queden sense discriminar.

Quan, fruit d'un creuament, es fusionen dos taxons, un dels dos taxons és assimilat per l'altre i, en conseqüència, els individus associats al taxò assimilat, cal associar-los al taxó assimilador.

Per dins, el taxonomista està composat per una llista indexada de taxons (taxonari). Els números d'índex es referencien des de cada individu pertanyent a la Comunitat.

Una alternativa al taxonari hagués sigut una implementació en arbre en comptes de la llista indexada. En cada node hi hauria una marca i a les fulles els identificadors de cada taxó. La implementació en arbre simplifica molt la lògica dels algorismes de discriminació i creuament però complica altres operacions internes que amb la llista indexada són trivials.

La llista indexada permet un accés directe als taxons i, a més, els manté ordenats de tal forma que les cerques de grups de parentesc tenen un cost temporal mínim.

3.5.2 Estadístiques de població

3.5.3 Estadístiques inter-poblacionals

Capítol 4

Manual del programador

4.1 Programació de noves topologies

Si l'usuari necessités crear un nou tipus de topologia, cal que la faci heretar de CTopologia, que és la classe que defineix el mínim per reservar memoria pel substrat de cada posició. CTopologia també estableix el protocol que han de seguir les subclasses, perqué la resta del sistema l'acepti sense haver de canviar-ho.

El secret està en el fet de que tot el sistema manega identificadors de posicions que són enters sense signe de 32 bits. Tot el significat que poden tenir aquests identificadors el manega la topologia internament.

Quan es deriva de CTopologia, el principal que caldria redefinir, si cal, és:

- Un **constructor** significatiu per a la topologia. Per exemple, en una topologia rectangular és significatiu indicar l'altura i l'amplada. El constructor de CTopologia simplement reserva espai per N casselles. Caldria calcular aquesta N per passar-se-la.
- `t_posicio CTopologia::desplacament (t_posicio origen, t_desplacament desplacament)`: Una funció per averiguar la posició destí en aplicar-li un vector de desplaçament a una posició origen. CTopologia, la defineix de tal forma que el resultat és una posició destí aleatoria.

- `bool CTopologia::esValid(t_posicio id)`: Una funció per saber si un identificador és vàlid. Només cal redefinir-ho si es modifica la correspondència directa entre identificador de posició i index de cassella en l'array de substrats reservada per CTopologia::CTopologia

- `t_posicio CTopologia::posicioAleatoria ()`: Una funció per obtindre aleatòriament una posició vàlida de la topologia. La funció general que no caldria redefinir seria

  ```
  {
  uint32 pos;
  do {pos=rnd.get();} while (!esValid(pos));
  return pos
  }
  ```

  però, CTopologia no fa servir aquest algorisme donat que optimitza agafant un número aleatori entre 0 i N. Aquesta optimització funciona mentre es mantingui la correspondència entre identificador i index abans comentada. Si la subclasse la trenca, es quan cal redefinir la funció.

- `bool CTopologia::unio (t_posicio origen, t_posicio desti, t_desplacament & desp)`: Una funció per calcular el primer d'un conjunt de desplacaments que cal fer per anar de l'origen al destí. Retorna cert si el desplacament és suficient per arribar a la posició destí. CTopologia, la defineix de tal forma que el resultat és un desplaçament aleatori i retorna sempre fals (mai hi arriba).

Pot ser molt ilustratiu, de cara a implementar noves topologies, fixar-se en les ja existents com CTopologiaToroidal.

4.2 Programació de nous agents

De cara a afegir nous agents al sistema, s'aconsella seguir els següents passos:

1. Llegir per sobre el codi dels agents ja implementats per assimilar les solucions que s'han donat a problemes que segurament es tornaran a repetir als nous agents. També convé mantenir uniforme l'estil de programació i l'ordre intern dels fitxers per fer-ho més mantenible a tercers. El més pràctic es partir d'una còpia d'un agent que tingui, estructuralment, tot o gran part del que interesa implementar.

2. Escollir la classe d'agent de la que volem heretar l'agent. Generalment voldrem que el nou agent pertanyi a un dels quatre grans grups funcionals d'agents:

   - Subordinadors (CMultiAgent i subclasses) si controla l'accionat d'altres agents
   - Posicionadors (CPosicionador i subclasses) si controla una posició en el biòtop
   - Direccionadors (CDireccionador i subclasses) si controla una direcció
   - Actuadors (CActuadors i subclasses) si modifica el substrat a una posició

   Si no pertany a cap dels quatre grups, caldria plantejar-se heretar de CAgent directament. En aquest cas, convé fer un esforç i fer una classe intermitja que pugui englobar altres agents en el futur. Anomenarem CAgentNou al nou agent afegit i CAgentVell a l'agent del qual heretem.

3. Adaptar el constructor de CAgentNou per que proveeixi els paràmetres del constructor de la superclasse. Posicionadors i direccionadors, per exemple, necessiten una referència a un biòtop en el constructor. A les classes derivades de CPosicionador està implementat com fer-ho.

4. Afegir dins del constructor, la línia.

   `m_tipus+="/ElMeuSubtipus";`

   que afegeix la cadena de subtipus a l'identificador de tipus que hereta de la superclasse.

5. Afegir els nous atributs (variables membre) dels que en depen l'estat de l'agent i les funcions d'accés als mateixos.

6. Inicialitzar dins del constructor els nous atributs als valors per defecte. Els atributs que siguin dependències amb altres agents, o agents subordinats, es recomana que siguin punters, i no referències, per poder-ho deixar sense especificar al constructor. S'inicialitzen sempre com a punter a NULL. Cal procurar que, si el punter no apunta a un agent vàlid el seu valor sigui NULL i tenir-ho en compte quan hi accedim per evitar accesos il·legals a memòria. Es veu clarament aquesta idea llegint el codi d'alguns agents que ho fan.

7. Afegir dins del destructor, l'alliberament de memòria ocupada pels agents subordinat. Les dependències no s'han de alliberar pas.

8. Redefinir la funció membre `virtual void CAgentNou::operator() (void)` per que faci el que hagi de fer quan l'agent és accionat. Si es tracta d'un actuador, no cal redefinir aquesta sino `virtual void CAgentNou::operator() (CSubstrat & s)` on `s` és el substrat que hem de modificar. [1]

9. Redefinir la funció `virtual void CAgentNou::dump(CMissatger & msg)` per que cridi a la funció corresponent de la superclasse (CAgentVell a l'exemple) i, després, inserti en el CMissatger les noves línies de configuració dels paràmetres que afegeix l'agent:

   ```
   void CAgentNou::dump(CMissatger & msg)
   {
   ```

[1] Veure l'apartat 4.3 que parla del que cal fer si es redefineix el substrat

```
CAgentVell::dump(msg);
msg << "- UnParametreNou " << m_valor1 << " " << valor2 << endl;
msg << "- UnAltreParametreNou " << m_valor3 << endl;
}
```

10. Redefinir la funció `virtual bool CAgentNou::configura(string parametre, istream & valors, t_diccionariAgents & diccionari, CMissatger & errors)` per mirar si el `parametre` és un dels que ha afegit CAgentNou. Si ho és cal parsejar l'istream `valors` en busca dels valors corresponents, reportar els errors que es produeixin pel CMissatger `errors` i retornar cert per dir que el paràmetre era de la classe. Si no ho és, cal cridar a la funció corresponent de la superclasse per que ho pugui interceptar ella. El diccionari serveix per, donat un nom d'agent de l'arxiu, obtindre un punter a l'agent que s'ha creat que, pot ser, té un nom diferent. El diccionari és un `map<string, CAgent*>`, el seu funcionament s'explica a qualsevol manual sobre les Standard Template Libraries de C++. La estructura general de la funció configura quedarà com això:

```
bool CAgentNou::configura(string parametre, istream & valors,
t_diccionariAgents & diccionari, CMissatger & errors)
{
if (parametre=="UnParametreNou") {
// Parsing dels valors...
return true;
}
if (parametre=="UnAltreParametreNou") {
// Parsing dels valors...
return true;
}
```

```
// Li deixem a la superclasse que l'intercepti si vol
return CAgentVell::configura(parametre, valors, diccionari, errors);
}
```

11. Si cap dels atributs (m_dependencia a l'exemple) és una dependència amb altre agent, cal redefinir la següent funció com segueix:

```
list<CAgent*> CAgentNou::dependencies() {
list<CAgent*> l=CAgentVell::dependencies();
if (m_dependencia) l.push_back(m_dependencia);
return l;
}
```

12. Si cap dels atributs (m_subordinat a l'exemple) és un agent subordinat, cal redefinir la següent funció com segueix:

```
list<CAgent*> CAgentNou::subordinats() {
list<CAgent*> l=CAgentVell::subordinats();
if (m_subordinat) l.push_back(m_subordinat);
return l;
}
```

13. Afegir a l'arxiu `Agent.cpp` un include a `AgentNou.h` i, a la funció estàtica CAgent::CreaAgent(...) una línia com les que ja n'hi ha per cada tipus d'agent, però, per a CAgentNou. Això permet que la funció CAgent::ParsejaArxiu pugui reconeixer el nou tipus als arxius de configuració.

De tots els punts anteriors el que potser és una mica més particularitzat són els atributs i els mètodes d'accés als mateixos, i el mètode d'accionament (o d'actuació en el cas dels actuadors).

Per a la resta de coses el més pràctic es fer un cut&paste dels agents ja implementats i retocar el mínim.

4.3 Programació de nous substrats

Bibliografia

[??98] ?? *Vida artificial, analisis y simulación*. PhD thesis, TODO: Investigar de que escuela es, 1998.

[AMR97] Jamshid Ghaboussi Anne M. Raich. Autogenesis and redundancy in GA representation. In *Proceedings of the 1997 International Conference on Genetics Algorithms ICGA97*. Michigan State University, 1997.

[Dar59] Charles Darwin. *The Origin of the Especies*. Guttemberg Project, 1859.

[FH91] Thomas Back Frank Hoffmeister. Genetic algortihms and evolution strategies: Similarities and differences technical report sys-91/2. Technical report, Systems Analysis Research Group, LSXI, Dept. of Computer Science, 1991.

[Fra97] Wolfgang Banzhaf; Peter Nordin; Frank D. Francone. Why introns in genetic programming grow exponentially. In *Proceedings of the 1997 International Conference on Genetics Algorithms ICGA97*. Michigan State University, 1997.

[Kar97] Hillol Kargupta. Relation learning in gene expression: Introns, variable length representation, and all that. In *Proceedings of the 1997 International Conference on Genetics Algorithms ICGA97*. Michigan State University, 1997.

[Lam86] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, 1986.

[Man] G. Mangiarotti. *Del gen al organismo. Biologia General*. Picin, Padova, Italia.

[May97] Helmut A. Mayer. ptga's, genetic algorithms using promoter/terminator sequences evolution of number, size and location of parameters and parts of representation. In *Proceedings of the 1997 International Conference on Genetics Algorithms ICGA97*. Michigan State University, 1997.

[Ray93] Thomas S. Ray. Jugué a ser dios y creé vida en mi computadora. *Epistemología e Informática*, pages 257–267, 1993.

[Ser96] Anselmo Pérez Serrada. *Una Introducción a la Computación Evolutiva*. PhD thesis, TODO: Investigar de que escuela es, Marzo 1996.

[Str] M. Strickberger. *Genética*. Ed. Omega, Barcelona, 3 edition.

[TS97] James A. Foster Terence Soule. Comments on the intron/exon distinction as it relates to the genetic programming and biology. In *Proceedings of the 1997 International Conference on Genetics Algorithms ICGA97*. Michigan State University, 1997.