Optimització d'un solucionador de 2048 utilitzant algorismes genètics al núvol

Marc Egea i Sala

Grau d'enginyeria informàtica

Àrea d'intel·ligència artificial

Consultor: **Dr. David Isern Alarcón**

Professor responsable de l'assignatura: **Dr**. **Carles Ventura Royo**

Juny del 2017

Aquesta obra està subjecta a una llicència de
Reconeixement 3.0 Espanya de Creative Commons

Fitxa del treball final

Títol del treball:	*Optimització d'un solucionador de 2048 utilitzant algorismes genètics al núvol*
Nom de l'autor:	*Marc Egea i Sala*
Nom del consultor/a:	*Dr. David Isern Alarcón*
Nom del PRA:	*Dr. Carles Ventura Royo*
Data de lliurament:	*06/2017*
Titulació o programa:	*Grau d'enginyeria informàtica*
Àrea del Treball Final:	*Intel·ligència artificial*
Idioma del treball:	*Català*
Paraules clau:	*Algorismes genètics, 2048, Computació al núvol*

Resum del Treball:

Aquest treball descriu l'intent de dissenyar un solucionador que jugui al joc de tauler 2048 de manera ràpida utilitzant només l'estat del tauler actual, sense aplicar cap tècnica de cerca sobre els moviments futurs.

Per fer-ho, s'ha dissenyat una funció d'avaluació que assigna un valor numèric a cada tauler. Aquesta funció modula a quins elements dona més importància depenent d'un conjunt de pesos que rep com a paràmetre. El solucionador aplica aquesta funció als possibles moviments en cada jugada, i es decideix per aquella que tingui una avaluació més alta.

Per tal d'optimitzar aquesta funció s'ha implementat un algorisme genètic per trobar la configuració dels paràmetres de la funció que maximitzen el resultat del solucionador. Alhora, s'ha desenvolupat un mètode per executar aquest algorisme aprofitant els avantatges de la computació al núvol per millorar el cost en temps i diners de la solució proposada.

S'han dut a terme diferents experiments per determinar quina és la influència dels diferents elements de l'algorisme genètic sobre el rendiment de les funcions generades i intentar maximitzar les opcions d'obtenir una bona solució. Els resultats mostren com la mida de la població o el número de generacions influeixen positivament en les puntuacions obtingudes, però no s'ha pogut establir l'impacte de les probabilitats d'aparellament i de mutació.

En conclusió, s'ha pogut desenvolupar un mètode per executar algorismes genètics en una infraestructura al núvol, però no s'ha aconseguit obtenir un solucionador de 2048 eficaç utilitzant aquest mètode.

Abstract:

This paper describes the attempt to design a solver for 2048, the board game, which only analyzes the current board state, without applying any search technique on future movements.

For this purpose, an evaluation function has been designed. This function assigns a numeric value to a board and decides which board elements are more important for the evaluation based on a set of weights received as a parameter. The solver applies this function to all possible moves on every play, and decides in favor of the one with a higher evaluation.

A Genetic Algorithm has been implemented to search for the function's parameters maximizing the solver's results. At the same time, a method for executing this algorithm in the cloud has been designed, improving both time and money costs of the proposed solution.

A serie of experiments have been run to determine the influence of the genetic algorithm's elements on the performance of the generated functions and to search for combinations to maximize the options of getting a good solution. The results show how the population size and the number of generations positively affect the final scores, but it hasn't been possible to establish the impact of the crossover and mutation probabilities.

In conclusion, a method to execute genetic algorithms in the cloud has been developed, but it hasn't been possible to build an effective 2048 solver using this method.

Índex de capítols

Índex de figures i taules

1. Introducció

1.1. Context i justificació del treball

El 2048[1] és un videojoc per un sol jugador amb un tauler de 4x4 posicions que poden contenir una fitxa o estar buides. Cada fitxa conté un número, i les dues primeres són col·locades aleatòriament pel joc amb un 2 o un 4. A partir d'aquí, el jugador pot manipular el tauler movent les fitxes cap a l'esquerra, la dreta, amunt o avall. Quan s'arraconen dues fitxes del mateix valor es fusionen en una sola del doble de valor. Per a cada moviment de l'usuari, el joc afegeix una nova fitxa en una posició aleatòria. L'objectiu del joc és arribar a fer una fitxa de valor 2048 (tot i que després es pot seguir jugant fins que el tauler estigui ple).

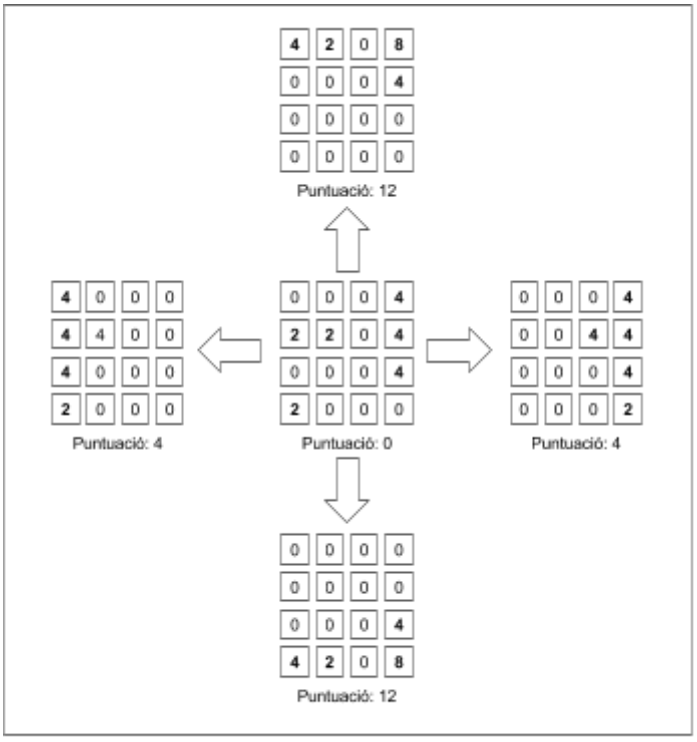


Figura 1: Exemple de possibles moviments d'un tauler

Actualment, tots els algorismes eficaços[2] per jugar al 2048 es basen en una exploració espacial de l'arbre de possibles taulers des del tauler actual. Aquests algorismes ponderen els millors i pitjors escenaris possibles per prendre una decisió. Com és comprensible, com més extensa sigui la cerca sobre l'arbre de possibles taulers, més efectius són aquests algorismes. El cost d'això és el temps que es triga a avaluar cada moviment a fer.

[1] https://github.com/gabrielecirulli/2048
[2] *What is the optimal algorithm for the game 2048?*, 2017

Un problema a resoldre, per això, és aconseguir un algorisme que sigui capaç de ser competitiu però responent en temps real, sense necessitat d'explorar els possibles taulers futurs. Durant aquest treball s'intentarà demostrar la tesi que és possible determinar quina de les (com a màxim) quatre opcions és millor només comparant els taulers resultants i ignorant completament el factor aleatori del joc.

Per tal d'explorar aquesta idea, es proposa construir una funció que assigni un valor numèric a un tauler basant-se només en atributs observables (número de peces, puntuació, disposició de les peces, etc.) d'aquest tauler.

A partir d'aquí, s'utilitzarà un algorisme genètic per optimitzar el pes que cadascun d'aquests atributs han de tenir sobre l'avaluació que s'assigna al tauler. L'objectiu és identificar, a base d'iteracions (generacions), quins d'aquests atributs s'han de valorar més, quins menys (o gens) i quins han de tenir, de fet, un impacte negatiu de cara a la puntuació final.

Aquesta aproximació requereix una gran capacitat de càlcul, ja que es tracta d'una optimització per força bruta. Per tant, es tracta d'un bon problema per aplicar tècniques de paral·lelització. Concretament, un dels objectius del treball és minimitzar el temps d'execució aprofitant els avantatges de la computació al núvol. Això ha de permetre moltes operacions intensives alhora sense haver de mantenir la infraestructura que suporti els pics de treball, utilitzant només els recursos necessaris en cada moment.

Aquest enfocament portaria, doncs, a gastar molt de temps en optimitzar la funció d'avaluació per permetre que aquesta es pugui executar de manera instantània en el futur. Tot i que aparentment un programa que jugui al 2048 no té una gran aplicació pràctica, la tècnica utilitzada per obtenir solucions a un problema amb unes regles poc clares és perfectament extrapolable a altres camps.

1.2. Objectius del treball

Aquest treball té tres grans objectius:

- Explorar l'ús dels algorismes genètics per optimitzar els paràmetres d'una funció. En concret, estudiar com afecten les diferents configuracions d'un algoritme genètic (número de generacions, població, possibilitats de mutació, etc.) a una funció d'avaluació de taulers de 2048.
- Dissenyar un sistema que permeti executar en paral·lel aquests algorismes. Això inclou aprofitar al màxim els nuclis del processador on s'executin i també l'habilitat de dur a terme múltiples experiments en paral·lel aprofitant l'ús de tecnologies al núvol.
- Finalment, i com a conseqüència dels dos punts anteriors, obtenir un algorisme per jugar al 2048 que aconsegueixi el millor resultat possible sense explorar en cap moment l'arbre de possibles taulers per a cada moviment.

1.3. Enfocament i mètode seguit

Des de la concepció, aquest ha sigut un treball amb una gran dosi d'incertesa i un fort component experimental. En aquest capítol es descriu com s'han enfocat la planificació i el desenvolupament per tal d'acotar el treball al temps disponible i poder obtenir resultats rellevants.

1.3.1. Llenguatge de programació

S'ha escollit Perl 5 com a llenguatge de programació per dur a terme les tasques de desenvolupament d'aquest treball. Hi ha tres grans motius per aquesta decisió:

- Es tracta d'un llenguatge interpretat i d'alt nivell. Això el fa molt adequat pel prototipatge de solucions, ja que permet fer proves molt ràpid i, per tant, iteracions més curtes.
- Compta amb una gran quantitat de llibreries disponibles sobre diferents àmbits. El seu repositori de distribucions, CPAN, té més de 185.000 llibreries disponibles[3].
- L'autor del treball compta amb experiència prèvia amb el llenguatge i l'utilitza en el seu dia a dia laboral. Això permet centrar-se en el problema a resoldre i no afegir la dificultat de dominar un llenguatge nou i el seu ecosistema.

1.3.2. Decisions basades en dades

Sempre que s'ha pres una decisió s'ha intentat obtenir dades que permetin decidir quina de les alternatives triar. Per fer això s'han utilitzat, per exemple, eines per dur a terme comparatives[4] i anàlisis de rendiment[5].

Com es mostrarà en aquesta memòria, s'han dut a terme diversos experiments i proves de concepte fins a arribar a la solució presentada. El disseny modular ha facilitat que es poguessin provar diferents algorismes genètics, funcions d'avaluació, configuracions de pesos, etc.

L'ús de testos automàtics ha permès comprovar que diferents implementacions es comporten de la mateixa manera. Així, s'han pogut provar diversos dissenys per a cadascun dels mòduls i escollir els que obtenen un rendiment més bo.

---

[3] "The Comprehensive Perl Archive Network (CPAN) currently has 185,579 Perl modules in 35,260 distributions, written by 13,090 authors, mirrored on 246 servers" (*The Comprehensive Perl Archive Network*, 2017).
[4] La llibreria Benchmark: https://metacpan.org/pod/Benchmark
[5] La lliberia NYTProf: https://metacpan.org/pod/Devel::NYTProf

1.3.3. Millora progressiva

El disseny i la implementació finals han sigut el resultat d'un seguit d'iteracions i refinaments. S'ha intentat començar sempre amb proves de concepte que s'han anat millorant fins a acabar amb la forma final presentada.

Alguns exemples d'això són:

- Abans d'implementar l'algorisme genètic es va utilitzar una llibreria externa[6]. Durant el desenvolupament del treball es van detectar diferents colls d'ampolla en aquest mòdul i es va decidir desenvolupar un algorisme genètic de zero. Aquesta decisió va permetre millorar el rendiment i la qualitat i comprensió del codi resultant.
- Al capítol 4.2.2 s'explica com s'ha automatitzat el sistema d'execució d'experiments per utilitzar tot el potencial de la computació al núvol. Per arribar a aquest estat, però, primer es van executar els experiments en local i, després, en infraestructura al núvol, però manualment. Fins i tot la solució final proposada ha passat per diferents nivells d'automatització.
- La primera versió del codi que gestiona les regles del 2048 era més clara i estructurada que la versió entregada. Durant les anàlisis de rendiment es va detectar que era on es gastava més temps i es van dur a terme una sèrie d'optimitzacions per millorar-ne el rendiment.

Aquesta aproximació permet prendre riscos calculats: cada decisió suposa un pas molt petit i, si no dona bon resultat, s'hi ha invertit poc temps i es pot avançar en una altra direcció. Això minimitza el risc d'exhaurir el temps amb una solució mig acabada.

[6] La llibreria AI::Genetic: https://metacpan.org/pod/AI::Genetic

1.4. Planificació del treball

La temporització del treball s'estructura en quatre grans blocs que coincideixen amb les entregues que preveu l'assignatura. Dins de cada bloc es detallen les tasques per entregar. Cada una de les tasques té un cost aproximat d'una setmana i, en general, no depèn de la finalització de cap altra tasca dins el mateix bloc.

En el cas de la memòria, aquesta es va desenvolupant durant tota la vida del projecte. En cada bloc s'especifica els apartats mínims que s'han de tenir acabats en aquell moment (correccions i canvis generals a banda).

Aquesta planificació intenta adaptar-se al procés de desenvolupament del treball. És molt difícil determinar la disponibilitat real d'hores de feina en cada moment i, per tant, s'ha optat per intentar generar blocs setmanals. S'estima que, en general, al cap d'una setmana s'han de disposar de les hores necessàries per finalitzar una tasca qualsevol.

Tot i que és imperatiu fer una planificació a priori, aquest model s'aproxima una mica més a les metodologies àgils, en què cada setmana es decidirà la distribució de subtasques segons la disponibilitat de temps d'aquella setmana en concret.

S'ha fet un esforç per generar tasques que tinguin el mínim nombre de dependències entre elles, però sí que es considera que s'ha d'acabar cada bloc abans de poder començar el següent. Per tot això, i com que no es pot assegurar en quin moment es duran a terme les tasques ni si es realitzaran en paral·lel o en sèrie, s'ha descartat l'ús d'un diagrama de *Gantt*, que no s'ajustaria a la realitat.

Els blocs de tasques són:

1.4.1. PAC 1 (2 setmanes)

- Pla de treball.

1.4.2. PAC 2 (4 setmanes)

- Desenvolupar motor de joc.
- Funció d'avaluació.
- Desenvolupar algorisme genètic.
- Memòria:
    - Iteració sobre l'estructura de la memòria.
    - Descripció i justificació de la implementació del motor de joc i l'algorisme genètic.

1.4.3. PAC 3 (4 setmanes)

- Estadístiques dels experiments.
- Disseny d'arquitectura.
- Desplegament de la solució. Depèn del disseny d'arquitectura.
- Memòria:
    - Iteració sobre l'estructura de la memòria.
    - Descripció i justificació de l'arquitectura.

1.4.4. Entrega final (4 setmanes)

- Execució de simulacions.
- Memòria.
- Formulari d'autoavaluació.
- Presentació.

1.5. Sumari de productes obtinguts

Aquest treball de final de grau consta d'una sèrie de productes que s'han entregat conjuntament:

- **Pla de treball** (*megeasa_platreball.pdf*): document que detalla l'abast i la planificació inicials del projecte.
- **Codi font** (*megeasa_codi/*): tot el codi font generat i necessari per executar les simulacions.
- **Memòria** (*megeasa_memòria.pdf*): aquest document, en què es recullen els aspectes més rellevants del desenvolupament del projecte i els resultats obtinguts. Consta dels capítols següents:
    1. Introducció: detalla els objectius, l'abast i la planificació del treball.
    2. Funció d'avaluació: descripció del mètode escollit per avaluar taulers de 2048.
    3. Estratègia d'optimització: descripció de com s'ha dissenyat i utilitzat un algorisme genètic per optimitzar el resultat d'un solucionador de 2048.
    4. Computació en paral·lel al núvol: descripció del disseny de l'arquitectura de la solució per executar els experiments en paral·lel sobre un proveïdor de computació al núvol.
    5. Resultat dels experiments: resum dels experiments realitzats i els seus resultats.
    6. Conclusions: valoració dels resultats, acompliment dels objectius i vies obertes de desenvolupament.
    7. Glossari: recull de termes rellevants en aquest document amb les seves definicions.
    8. Bibliografia: recull de referències consultades en l'elaboració d'aquest treball.
    9. Annexos: material complementari sobre el desenvolupament del treball.
- **Formulari d'autoavaluació** (*megeasa_autoavaluació.pdf*): avaluació de les competències utilitzades durant aquest projecte.
- **Presentació** (*megeasa_presentació.avi i megeasa_presentació.pdf*): defensa en vídeo del treball de final de grau i les seves transparències.
- **Resultats** (*megeasa_codi/*)**:** fitxers resultants de l'execució dels experiments.

2. Funció d'avaluació

El solucionador proposat en aquest treball utilitza una funció que assigna un valor numèric a un tauler de 2048. Llavors, des de l'inici del joc fins que s'acabi, s'avaluaran els següents quatre taulers possibles i s'escollirà moure en la direcció del tauler amb una avaluació més alta.

D'aquesta manera, si s'aconsegueix una funció d'avaluació que sempre retorni més puntuació per taulers que són més favorables pel resultat final, el solucionador serà capaç d'obtenir puntuacions finals més bones.

El problema és, doncs, trobar un mecanisme per assignar una puntuació a un tauler de manera que els taulers que portin a un joc més llarg tinguin una puntuació més alta. En aquest cas, s'ha seguit l'estratègia de determinar un seguit d'atributs observables del tauler i assignar-los un valor numèric. Llavors, cada atribut es pondera utilitzant un conjunt de pesos [P1..P8], que determinaran quina importància es donarà a cada atribut.

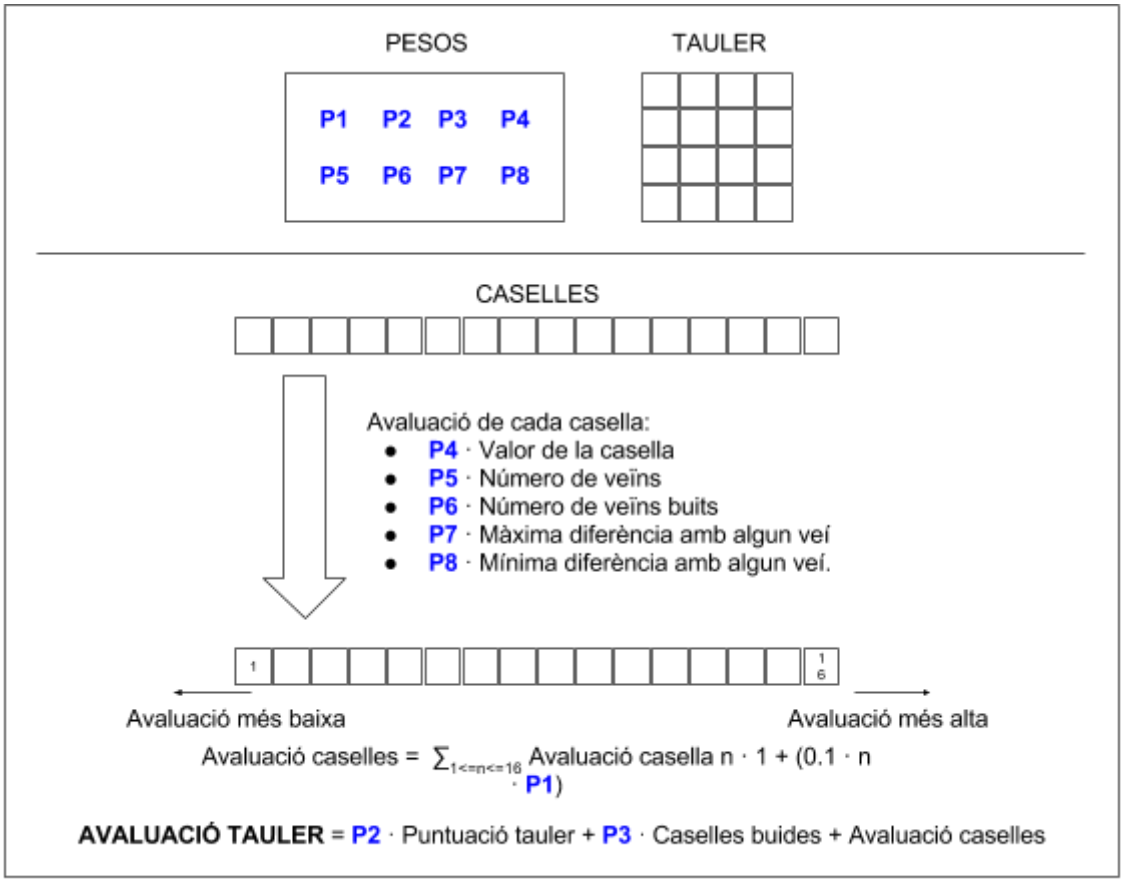


Figura 2: Efecte dels pesos dins la funció d'avaluació

La figura 1 resumeix com influeixen els diferents pesos en l'avaluació d'un tauler. Per obtenir l'avaluació del tauler, se sumaran la dels seus atributs i la de les seves caselles.

Per obtenir l'avaluació de les caselles, primer s'avalua cada casella individualment i després s'ordenen segons aquesta puntuació. Cada casella es multiplica, després, per un factor de ponderació basat en aquesta posició.

Tot el projecte gira al voltant d'optimitzar aquesta funció per tal que l'avaluació que retorni permeti escollir més bé el següent moviment en una partida de 2048. Aquesta optimització es basarà a trobar el valor dels vuit pesos que maximitzi la puntuació de partides jugades amb aquesta funció d'avaluació.

No hi ha cap relació entre aquests pesos. Això vol dir que el valor que prengui cadascun és completament independent del de la resta. Precisament, la feina de l'algorisme genètic serà trobar combinacions de valors que maximitzin el resultat de les partides jugades.

2.1. Avaluació dels atributs del tauler

L'avaluació dels atributs del tauler és la suma de dos elements:

- El número de caselles buides multiplicat per P2.
- La puntuació actual del tauler multiplicat per P3.

2.2. Avaluació d'una casella

L'avaluació de cada casella del tauler és la suma dels següents elements:

- El valor de la casella multiplicat per P4.
- El número de caselles adjacents multiplicat per P5.
- El número de caselles adjacents buides multiplicat per P6.
- La diferència de valor més petita amb una casella adjacent multiplicada per P7.
- La diferència de valor més gran amb una casella adjacent multiplicada per P8.

2.3. Factor de ponderació de la posició

Un cop assignat un valor numèric a cada casella, aquestes s'ordenen segons aquest valor. Llavors, cadascun es multiplica per un factor proporcional a P1. Aquest càlcul permet matisar la importància de les caselles en la puntuació total.

Un valor de P1 alt implicarà que les caselles menys significatives (amb una avaluació menor) s'han de considerar poc, mentre que un valor baix voldrà dir que totes les caselles tenen aproximadament el mateix valor. Un valor negatiu, en canvi, voldrà dir que s'ha de penalitzar la puntuació de les caselles més significatives.

2.4. Justificació dels atributs triats

D'entrada, considerar els espais en blanc i la puntuació del tauler semblen decisions prou evidents: d'una banda, quan ens quedem sense espai la partida s'acaba, i de l'altra, estem intentant maximitzar la puntuació.

En canvi, avaluar les caselles individualment respon a una tesi personal: en un tauler de 2048 hi ha caselles més importants que d'altres per determinar com de bo és aquest tauler.

L'algoritme genètic hauria d'ajudar, precisament, a desxifrar quins són els atributs que fan que una casella sigui més important que la resta. Una de les estratègies més recomanades[7] és intentar mantenir sempre la fitxa més alta en una cantonada.

Si aquesta assumpció és certa, l'algorisme genètic hauria d'obtenir valors positius per P4 (les caselles amb un valor més alt serien més importants) i valors negatius per P5 (les caselles amb menys veïns, les cantonades, serien les més importants).

La ponderació de la posició ha de permetre establir si realment hi ha caselles més importants que d'altres, i en quina mesura.

[7] "The first step with all these Candy Crush-meets-Sudoku number games is to understand that the corner is your best friend. For me, it's the upper left. It's just how I play, and any of the corners will do. That strategy lets you build toward a singular tile without moving it around and disrupting your ability to merge it with other large tiles when the time comes." (Statt, 2014)

3. Estratègia d'optimització

Donat el problema d'obtenir un solucionador eficaç de 2048, en aquest treball s'ha seguit l'estratègia d'emular la selecció natural per obtenir, de manera automatitzada, versions cada cop millors d'aquest solucionador.

Per fer-ho, s'ha dissenyat un algorisme genètic en el qual diferents versions de la funció d'avaluació competeixen entre elles i en què les més aptes s'aparellen i muten per obtenir versions cada vegada més òptimes.

Per utilitzar l'algorisme genètic com a mecanisme per optimitzar els pesos, aquests s'hauran de codificar en forma de cromosoma. Al capítol 3.3 s'explica amb detall com es poden obtenir [P1..P8] a partir de cromosomes.

En aquest capítol es descriu el disseny d'aquest algorisme i les diferents decisions tècniques que s'han pres durant aquest procés.

3.1. Algorisme genètic

Un algorisme genètic permet optimitzar els paràmetres d'una funció[8] per a un resultat òptim mitjançant iteracions sobre possibles configuracions de paràmetres. Per fer-ho, utilitza estratègies inspirades en la selecció natural, com ara la codificació en cromosomes i gens, l'aparellament o les mutacions.

La idea bàsica és trobar una manera de modelar els paràmetres d'una funció com a una llista de gens, anomenat cromosoma. Llavors, s'escullen els cromosomes més aptes i mitjançant el seu aparellament i mutació, s'obté generació rere generació fins a trobar cromosomes amb una aptitud acceptable.

L'aptitud dels cromosomes serà, en aquest cas, el resultat d'aplicar els gens del cromosoma sobre la funció que es vol optimitzar. Els cromosomes més aptes seran aquells que maximitzin el valor d'aquesta funció.

Els paràmetres d'entrada de l'algorisme genètic que s'ha dissenyat són la *mida de la població*, el *número de generacions*, la *probabilitat d'aparellament* i la *probabilitat de mutació.*

L'algorisme generarà una població inicial en què els individus són cromosomes amb els seus gens escollits aleatòriament. Llavors, obtindrà noves generacions a partir de l'anterior fins a arribar al número de generacions desitjat. Finalment, l'última generació obtinguda contindrà els cromosomes més aptes.

[8] "One common application of GAs is function optimization, where the goal is to fins a set of parameter values that maximize a complex multiparameter function" (Mitchell, 1999, pàgina 7)

3.1.1. Obtenció d'una nova generació

A partir d'una generació amb un número *N* d'individus, l'algorisme genètic obté una nova generació afegint *N* nous individus a la generació anterior i després escollint els *N* més aptes.

Els individus que s'afegeixen a la generació anterior s'obtenen de dos en dos, de la següent manera:

- Se seleccionen dos candidats de l'actual generació per a l'aparellament.
- Els candidats intenten un aparellament, que té èxit i produeix dos nous individus depenent només de la *probabilitat d'aparellament* de l'algorisme.
- Depenent de la *probabilitat de mutació*, cadascun d'aquests nous individus mutarà un dels seus gens abans d'afegir-se a la nova generació.
- En cas que l'aparellament no tingui èxit, s'afegeixen dos cromosomes amb gens aleatoris a la generació.

Aquest sistema permet assegurar que cada generació serà, com a mínim, tan apta com l'anterior, ja que un individu de la població només es reemplaça si s'ha obtingut un cromosoma més apte que ell.

3.1.2. Selecció de candidats

La funció de selecció de candidats retorna dos individus dins d'una generació. Per fer-ho, utilitza un mètode de ruleta[9] que consisteix a:

- Generar una llista en què cada individu dins de la generació hi apareix un número de vegades proporcional a la seva aptitud.
- Seleccionar dos individus d'aquesta llista a l'atzar.

D'aquesta manera, tots els individus tenen opcions d'aparellar-se, però els més aptes tenen més possibilitats de fer-ho. Cal tenir en compte que, amb aquest mètode de selecció, un mateix cromosoma pot ser retornat per duplicat. Si això passa sovint, en poques generacions la població constarà de cromosomes molt iguals i només es comptarà amb les mutacions per variar la càrrega genètica i obtenir nous cromosomes.

[9] "The most common fitness-proportionate selection technique is called *Roulette Wheel Selection*. Conceptually, each member of the population is allocated a section of an imaginary roulette wheel. Unlike a real roulette wheel the sections are different sizes, proportional to the individual's fitness, such that the fittest candidate has the biggest slice of the wheel and the weakest candidate has the smallest. The wheel is then spun and the individual associated with the winning section is selected. The wheel is spun as many times as is necessary to select the full set of parents for the next generation." (Dyer, 2010, capítol 3)

3.1.3. Aparellament

S'ha decidit seguir una estratègia d'aparellament uniforme[10]: L'aparellament entre dos cromosomes *A* i *B* produeix dos nous cromosomes, *X* i *Y*, els gens dels quals es determinaran de la següent manera:

- Si aquell gen és igual a *A* i *B*, llavors *X* i *Y* obtenen el valor dels pares.
- Si el gen és diferent a *A* i *B*, llavors es decideix aleatòriament si *X* obté el gen d'*A* i *Y* obté el gen de *B* o viceversa.

La intenció és privilegiar els gens que estan donant bons resultats. En cas de discrepància, els gens es distribuiran aleatòriament entre els dos descendents per tenir l'oportunitat de trobar quins són més aptes.

Aquest tipus d'aparellament, a més, funciona amb els tipus de cromosomes que s'han utilitzat (vegeu el capítol 3.2). S'ha descartat utilitzar estratègies de tall en aquest treball, ja que s'ha considerat un risc desenvolupar funcions d'aparellament per a cada cromosoma (alguns gens tenen significat conjunt, no individual). Això permet introduir variacions controlades dins els cromosomes, que d'altra manera podrien tendir a homogeneïtzar-se.

3.1.4. Mutació

La mutació d'un cromosoma es porta a terme escollint un gen a l'atzar del cromosoma i donant-li un valor aleatori d'entre els possibles.

Una manera alternativa d'enfocar-ho podria haver set donar a cada gen la probabilitat de mutar, de manera que per a un cromosoma hi pugui haver diverses mutacions.

No obstant això, es va considerar que podria alterar massa els gens descendents de cromosomes amb bona aptitud i es va descartar. Aquesta decisió impactarà en l'estabilitat dels gens entre generacions.

3.1.5. Fitxer de resultats

Com a resultat de l'execució de l'algorisme genètic s'obté un fitxer CSV, amb una fila per cada individu de cada generació. La primera columna contindrà la generació, la segona contindrà l'aptitud i la resta de columnes defineixen el cromosoma.

Per tant, en un experiment amb una població de 100 individus durant 100 generacions s'obtindrà un fitxer de 10.001 línies (100 x 100, més la capçalera).

[10] "Uniform crossover selects the two parents for crossover. It creates two child offspring of n genes selected from both of the parents uniformly. The random real number decides whether the first child select the i^{th} genes from first or second parent" (Umbarkar and Seth, 2015)

Aquests fitxers s'utilitzen per generar els gràfics (Annex IV) que permeten l'anàlisi dels resultats del capítol 5. Aquests gràfics recullen l'evolució de l'aptitud màxima i mitjana entre generacions.

Tant els fitxers CSV com els gràfics generats s'adjunten com a resultats del treball.

3.2. Cromosomes

En un algorisme genètic com el que s'ha implementat, un cromosoma és simplement un conjunt de gens, cadascun dels quals té una posició i un valor determinat. A més, dins un cromosoma, cada gen pot prendre un valor d'entre un conjunt finit.

Si aquests gens s'utilitzen com a paràmetres de la funció que es vol optimitzar, l'algorisme genètic provarà combinacions que maximitzin el resultat de la funció.

No obstant això, la nostra funció d'avaluació té una sèrie de paràmetres determinats (els pesos [P1...P8]). Es planteja, doncs, el problema de dissenyar un model de cromosoma en què els seus gens es puguin transformar en pesos de manera unívoca.

Com que els valors possibles dels gens han de ser finits (i si pot ser, petits) perquè l'operació de mutació tingui una aleatorietat controlada, s'han considerat dos tipus de cromosomes:

- Cromosomes en què el valor dels gens pot ser 0 o 1.
- Cromosomes en què el valor dels gens pot ser P1, P2, P3, P4, P5, P6, P7 o P8.

Tenint això en compte, s'han dissenyat dos tipus de cromosomes, els cromosomes de suma zero i els de valor independent. De cadascun, se n'han implementat diverses variants.

3.2.1. Cromosomes de suma zero

Els cromosomes de suma zero consideren que la suma total dels pesos ha de ser zero. Els gens d'aquest tipus de cromosomes poden tenir qualsevol valor entre P1 i P8.

Per aconseguir que els pesos sumin 0, la meitat de les posicions sumaran una unitat al valor del pes que tinguin codificat i l'altra meitat n'hi restaran una. A la figura 2 es mostra un exemple de com interpretar els gens d'un cromosoma de suma zero.

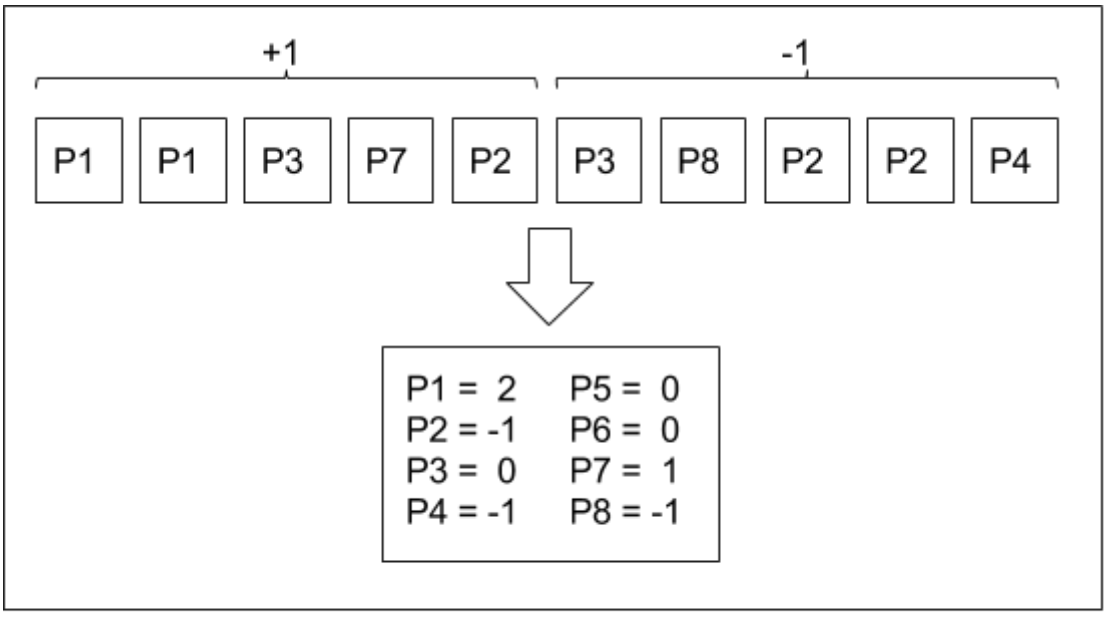


Figura 3: Exemple de cromosoma de suma zero de 10 gens.

Segons aquest model, els valors dels pesos estan connectats entre ells, i augmentar el valor d'un dels pesos voldrà dir disminuir el valor dels altres. Els valors màxims i mínims que podran tenir els pesos dependran del número de gens utilitzats en els cromosomes.

S'ha de tenir en compte, però, que els valors tendiran a estar molt més agrupats al voltant del 0 que no pas als extrems, ja que un valor molt alt o molt baix significa que la majoria d'una de les meitats dels gens tenen el mateix valor.

3.2.2. Cromosomes de valor independent

En els cromosomes de valor independent els gens només poden tenir els valors 0 o 1.

L'estratègia d'aquests cromosomes és assignar un número concret de gens a cada pes, i que el valor del pes sigui una interpretació en binari del seu conjunt de gens: un número racional amb un valor acotat al número de bits utilitzats per representar-lo.

A la figura 3 es mostra com interpretar el valor de dos pesos en un cromosoma de valor independent de 5 bits. S'hi pot observar com el valor de cada pes està definit per la representació binària dels seus gens, en què el primer denota el signe positiu o negatiu del valor.

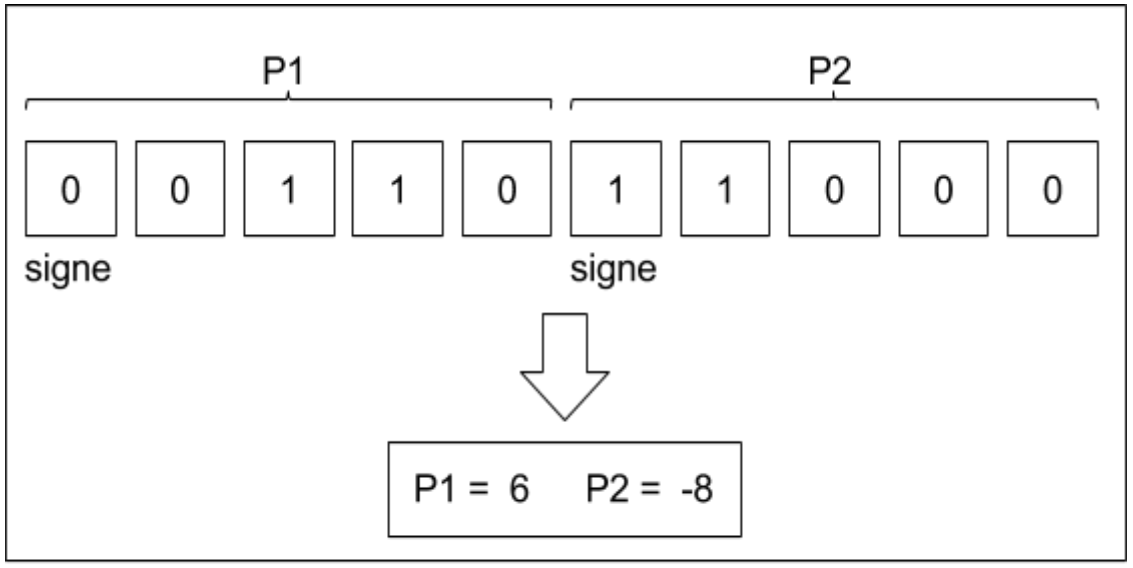


Figura 4: Exemple de cromosoma de valor independent de 5 bits (mostra només dos pesos).

Sota aquest model hi ha alguns gens que tenen un impacte molt més gran que d'altres, ja que actuaran com a bits de més pes de la representació binària o bé com a bits de signe.

S'espera que els individus que tinguin puntuacions diferents dels seus progenitors siguin aquells que mutin aquests bits representatius.

El número de bits que es vulguin utilitzar per a cada pes determinarà la mida total del cromosoma i el rang de valors que poden assolir aquests pesos. En aquest cas, els valors de cada pes no són interdependents i poden evolucionar per separat.

3.2.3. Cromosomes implementats

La taula 1 resumeix les característiques de tots els cromosomes sobre els quals s'han fet experiments.

Taula 1: Resum dels cromosomes implementats

Nom	Tipus	Gens	Rang de valors dels pesos
10Genes	Suma zero	10	Entre -5 i 5
100Genes	Suma zero	100	Entre -50 i 50
1000Genes	Suma zero	1.000	Entre -500 i 500
5000Genes	Suma zero	5.000	Entre -2500 i 2500
8Bits	V. independent	64	Entre -127 i 127
8Bits1decimal	V. independent	64	Entre -12,7 i 12,7
9Bits	V. independent	72	Entre -255 i 255
9Bits1decimal	V. independent	72	Entre -25,5 i 25,5
10Bits	V. independent	80	Entre -511 i 511
10Bits1decimal	V. independent	80	Entre -55,1 i 55,1

Per descobrir quins tipus de cromosomes modelen més bé el conjunt de pesos per a aquest problema s'han realitzat una sèrie d'experiments comparatius. Al capítol 5.2 es descriu com es comporta l'algorisme genètic segons el tipus de cromosoma estudiat.

2.3.4. Aptitud d'un cromosoma

Determinar l'aptitud d'un cromosoma és el mateix que determinar com de bona és la funció d'avaluació amb els pesos que s'obtenen d'aquest cromosoma.

La manera de determinar com de bona és una funció d'avaluació és utilitzar-la per jugar partides. La puntuació mitjana d'aquestes partides serà el valor que s'utilitzarà com a qualitat de la funció.

Al capítol 5.1 es pot veure com s'ha calculat el número concret de partides que s'han de jugar per tal de determinar l'aptitud d'un cromosoma.

3.3. Implementació

En aquest capítol s'expliquen els blocs que s'han desenvolupat per resoldre el problema de construir un algorisme que jugui al 2048. Cada bloc ha estat dissenyat per actuar per separat, de manera que siguin components intercanviables durant el procés d'expermientació.

Aquests blocs son l'*Algorisme genètic*, el *Cromosoma*, el *Jugador*, l'*Avaluador* i el *Tauler*. La combinació d'aquests blocs conforma el disseny de la solució, com es pot observar a la figura 4.

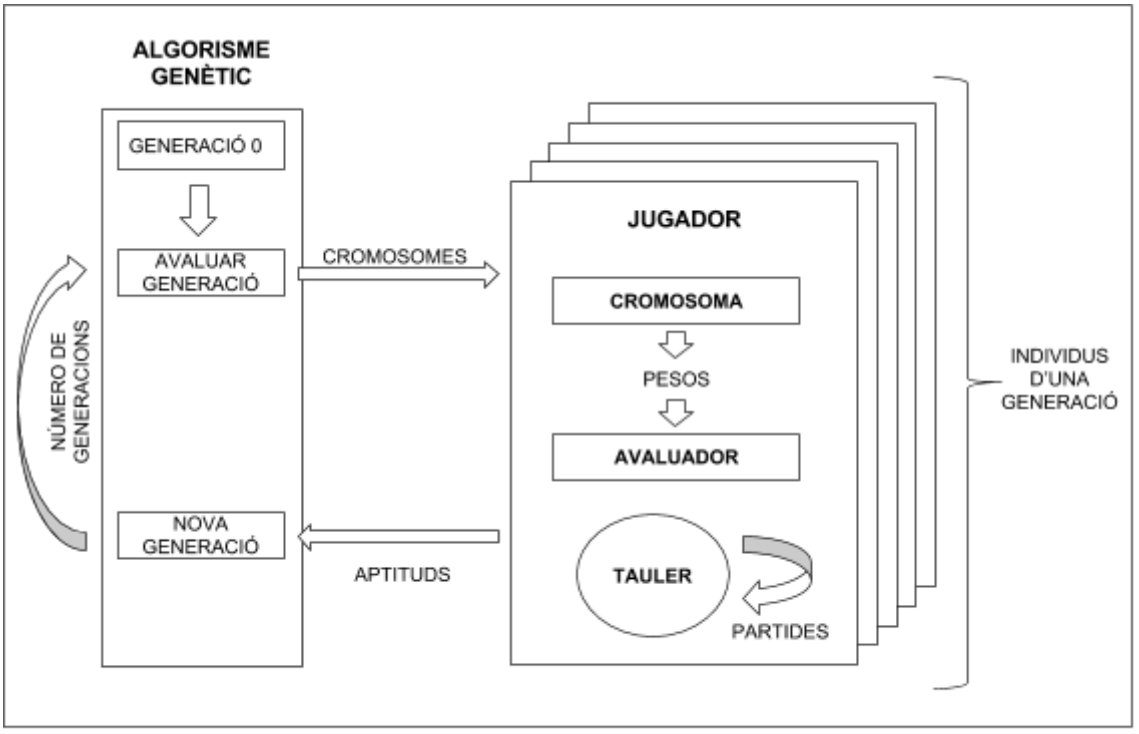


Figura 5: Diagrama d'interacció dels diferents mòduls.

La funció de l'*Algorisme genètic* és optimitzar els *Cromosomes* per obtenir aquells que generin individus amb una aptitud més alta. Per fer-ho, obté una generació d'individus, n'avalua l'aptitud i aplica les regles d'aparellament i mutació per generar una nova generació d'individus cada cop més aptes. Aquest procés es repeteix durant un número fixat de generacions.

Els individus, en aquest cas, són *Jugadors*. Un *Jugador* es construeix a partir d'un *Cromosoma* i és capaç d'avaluar la seva aptitud jugant al 2048. Aquesta aptitud es calcula jugant un número fixat de partides, de les quals s'obté la puntuació mitjana. Els individus més aptes són aquells que obtenen puntuacions més bones de manera consistent.

Els *Jugadors* fan partides utilitzant el *Tauler*. Cada nova partida comença amb un *Tauler* nou, que retorna els possibles moviments que es poden dur a terme. Llavors, el *Jugador* escull el millor d'aquests moviments utilitzant un *Avaluador*, que atorga una puntuació a cada *Tauler*. El que tingui una puntuació més alta serà el que el *Jugador* consideri el seu millor moviment.

L'*Avaluador* puntua diferents atributs del tauler i en pondera la importància utilitzant una llista de pesos. El contingut d'un *Cromosoma* permet construir aquest conjunt de pesos per l'*Avaluador*. Per tant, els valors concrets d'aquests pesos dependran del *Cromosoma* de cada individu.

Mentre es van optimitzant els *Cromosomes* en realitat el que s'optimitza són els pesos que configuren un *Avaluador*. Finalment, el *Cromosoma* més apte permet generar la funció més efectiva per jugar al 2048.

L'execució de tot el cicle es denomina experiment. Durant la realització del treball s'han portat a terme múltiples experiments amb diferents configuracions de cadascun dels components. Els resultats d'aquests experiments es poden consultar al capítol 5.

4. Computació en paral·lel al núvol

Un dels punts en contra dels algorismes genètics és que, en tractar-se d'optimitzacions per força bruta, requereixen un ús intensiu del processador durant molt de temps. La solució descrita a l'apartat anterior, implementada sense cap més consideració, porta a una solució que corre amb un únic procés.

Això fa que l'única manera d'accelerar els càlculs sigui augmentar la freqüència del processador. Aquesta és una estratègia amb poc recorregut, ja que la velocitat de la majoria de processadors es mou en el mateix ordre de magnitud i hi ha límits físics que no es poden superar.

4.1. Computació en paral·lel

L'estratègia per augmentar la velocitat d'execució ha sigut intentar executar en paral·lel l'avaluació de l'aptitud dels individus d'una generació. Són càlculs que es poden fer per separat ja que són completament independents els uns dels altres. A més, el gruix del temps d'execució es gasta dins aquestes avaluacions.

Per tal de separar aquests càlculs s'utilitzarà una estratègia de *fork* per engegar nous processos, cadascun dels quals s'encarregarà de calcular l'aptitud d'una part proporcional dels individus de la generació. Seguint aquesta estratègia, l'algorisme pot aprofitar al màxim el temps de procés en màquines amb múltiples nuclis, utilitzant tants processos com nuclis té la màquina.

Per a aconseguir aquest objectiu, s'ha modificat l'algorisme genètic perquè es comporti de la següent manera:

- Es divideixen els individus de la generació entre tots els processos.
- Cada procés obté l'aptitud dels seus individus.
- S'espera que tots els processos hagin acabat.
- S'obté una nova generació (aparellament i mutació) en un sol procés.

4.2. Computació al núvol

Degut a la naturalesa exploratòria del projecte, es va decidir des d'un principi que era molt important que els experiments fossin tant ràpids d'executar com fos possible per poder maximitzar el número de proves. Una altra característica molt desitjable era poder fer els experiments en paral·lel.

Amb els canvis a la implementació per poder avaluar els individus en paral·lel s'obre la porta a poder aprofitar al màxim els ordinadors amb més d'un nucli. De totes maneres, si només es disposa d'un ordinador no es poden realitzar experiments en paral·lel sense que hagin de competir pels recursos i, en definitiva, trigar el mateix que si s'haguessin executat l'un darrere l'altre.

Per tal d'agilitzar l'execució dels experiments es va decidir utilitzar la plataforma de computació al núvol EC2 d'AWS. Aquest servei permet executar màquines virtuals (instàncies) a l'infraestructura d'AWS, pagant només pel temps efectiu d'execució (si una instància està aturada, no es paga pel seu ús).

4.2.1. Subhasta d'instàncies

Dins el servei EC2 es poden llogar diferents tipus d'instàncies. Es diferencien entre elles pels tipus de processador, la quantitat i velocitat de la memòria RAM, el número d'operacions a disc per segon, l'ús o no de targetes gràfiques, etc.

Cada tipus d'instància té un preu per hora i el model habitual per treballar amb EC2 és reservar instàncies a aquest preu. Una instància reservada sempre estarà disponible i no es destruirà mentre l'usuari no ho decideixi. Tanmateix, pot estar apagada i llavors no generarà costos, però mentre estigui engegada es pagarà sempre aquest preu per hora[11].

A banda de reservar instàncies per un temps indefinit, AWS manté una subhasta d'instàncies en què el preu per hora és, típicament, de quatre a cinc vegades més barat que el preu d'instàncies reservades. Aquest preu és públic i depèn dels excedents de màquines que hi hagi en cada moment i dels preus que els clients estan disposats a pagar.

Quan es demana una instància de subhasta s'estableix el preu màxim per hora que es pagarà. Si el preu de subhasta d'aquell tipus d'instància puja d'aquest límit, la instància es destruirà per fer lloc als nous postors.

[11] "At the start of each instance hour, you are charged based on the Spot price. If your Spot instance is interrupted in the middle of an instance hour because the Spot price exceeded your bid, you are not charged for the hour of use that was interrupted. However, if you terminate your Spot instance in the middle of an instance hour, you are charged for the hour." (*Spot Instances*, 2017)

S'ha de tenir en compte que, independentment del límit establert, es pagarà només el preu de subhasta en cada moment[12].

El perill, en aquest cas, és dur a terme experiments amb una duració de diverses hores i que la instància on s'estiguin executant pugi de preu i s'apagui; en aquest cas, s'ha de repetir l'experiment sencer. Per evitar o mitigar aquest problema, es podrien aplicar diverses solucions:

- Guardar periòdicament resultats parcials i habilitar un mode de continuació que llegeixi aquests resultats parcials i acabi l'experiment des d'aquell punt.
- Oferir un preu per instància molt superior al que hi hagi al moment de realitzar l'experiment.

No obstant això, durant la realització dels experiments s'han utilitzat instàncies de subhasta amb un preu un 10% per sobre del de mercat i no s'ha donat ni un cas de màquina destruïda per sobrepreu, fins i tot en experiments de més de 48 hores.

4.2.2. Automatització

Una de les característiques més interessants de proveïdors com AWS és que és possible interactuar amb la infraestructura de manera programàtica. Això permet automatitzar tots els passos per portar a terme un experiment, des que es reserva la instància fins que s'apaga.

Una de les parts més importants d'aquest treball és, precisament, aquesta automatització, que es tradueix en la comanda que permet executar experiments al núvol (vegeu el fitxer *script/run_experiment_in_ec2.pl*). Per utilitzar-la, caldrà especificar:

- La configuració de l'experiment (número d'individus, de generacions, tipus de cromosoma i probabilitats d'aparellament i mutació).
- El tipus d'instància sobre la qual es vol executar l'experiment.
- El preu per hora disposat a pagar (o el sobrepreu sobre el mercat).
- El contenidor AWS S3 on es deixaran els resultats.

[12] "Everyone pays that same Spot price for that period, regardless of whether their bid price was higher. You never pay more than your bid price per hour, and often pay less per hour. For example, if you bid $0.25 per hour, and the Spot price is $0.20 per hour, you only pay $0.20 per hour. If the Spot price drops, you pay the new, lower price. If the Spot price rises, you pay the new price if it is equal to or less than your bid price. If the Spot price rises above your bid price, then your Spot instance is interrupted." (*Spot Instances*, 2017)

Amb aquestes dades, la comanda utilitzarà l'API pública d'AWS per tal de:

- Reservar una instància de subhasta del tipus demanat.
- Escollir la imatge del sistema operatiu que es carregarà (en aquest cas, s'utilitzarà Ubuntu 16.04) en aquesta instància.
- Configurar, aprofitant el mecanisme anomenat *User Data*, què s'executarà dins la instància el primer cop que s'engegui. En aquest cas, es proveeixen instruccions per:
    - Descarregar el codi d'aquest treball des del repositori Git.
    - Instal·lar les dependències de sistema.
    - Instal·lar les dependències de Perl.
    - Executar l'experiment amb la seva configuració.
    - Enviar els resultats de l'experiment al contenidor S3.
    - Apagar la instància.

Tan bon punt el preu ofert superi el de la subhasta (cosa que passarà immediatament si s'ha establert un preu per sobre del mercat) aquesta instància s'engegarà. Gràcies a la seva configuració, quan s'engegi començarà a executar l'experiment i, quan acabi, s'apagarà.

Amb aquesta aproximació, a la pràctica es poden fer experiments sota demanda amb una sola comanda. Es poden fer tants experiments a l'hora o utilitzar els tipus d'instància que es consideri. A més, un procés completament automatitzat evita l'error humà i permet ajustar el cost al mínim imprescindible.

4.2.3. Autenticació i seguretat

Com és evident, AWS implementa mecanismes per evitar que qualsevol pugui accedir als diferents recursos de la seva infraestructura. Per tal que el procés d'execució d'experiments funcioni, abans s'han de realitzar una sèrie de configuracions relacionades amb la seguretat.

D'entrada, s'ha de tenir un compte habilitat a AWS, i s'han de generar unes credencials, que s'han de guardar a l'ordinador (típicament en un fitxer de configuració o en variables d'entorn).

També s'ha de crear el contenidor S3 que guardarà els resultats i crear un nou rol que atorgui accés d'escriptura a aquest contenidor. Aquesta configuració es porta a terme utilitzant el servei AWS IAM. Quan es configuri una nova instància, s'haurà d'especificar que aquesta implementa el rol.

Amb aquestes dues accions es podrà assegurar que l'usuari pot utilitzar l'API d'AWS de manera autenticada (i, per tant, assignar les peticions al seu compte) i permetrà que les instàncies que executin experiments puguin guardar els resultats per ser consultats més endavant.

4.3. Costos i rendiment

El servei EC2 ofereix molts tipus d'instàncies, però per dur a terme aquest treball s'han avaluat només les instàncies de la família c4, que ofereixen un alt rendiment de processador, però menys memòria i pitjors temps d'accés a disc que d'altres famílies.

El codi d'aquest treball és intensiu en temps de processament, però no utilitza una gran quantitat de memòria i no fa gaire accés a disc (només per escriure els resultats). Per aquest motiu s'ha considerat que la família c4 era l'adequada: permet pagar el mínim preu per número de nuclis i velocitat de processador.

La taula 2 reflecteix el temps i cost aproximat d'evolucionar una generació de 100 individus utilitzant les diferents instàncies de la família c4. Aquestes dades s'han obtingut experimentalment.

Taula 2: Comparativa d'instàncies EC2

Tipus	**Nuclis**	**Cost per hora**	**Preu nucli/hora**	**Temps per generació**	**Cost per generació**
c4.large	2	0,0290$	0,0145$	82s	0,000693$
c4.xlarge	4	0,0588$	0,0145$	44.5s	0,000717$
c4.2xlarge	8	0,1140$	0,01425$	22.5s	0,000740$
c4.4xlarge	16	0,2201$	0,013765$	13s	0,000795$
c4.8xlarge	36	0,4371$	0,012142$	6.6s	0,000801$

Cal recalcar que els càlculs no poden ser exactes ja que els preus de subhasta varien i el temps per generació no serà mai uniforme. No obstant això, hi ha dues tendències a destacar:

- El preu per nucli/hora disminueix quan s'utilitzen màquines més potents.
- Malgrat això, el cost per generació és més petit en màquines menys potents.

Aquesta discrepància es deu al funcionament del sistema de paral·lelització: el temps total de crear una generació nova serà el temps del nucli que trigui més a calcular l'aptitud dels seus individus. Això és així perquè no es pot començar el procés d'aparellament i mutació fins que tots els nuclis acabin i es tingui l'aptitud de tots els individus.

Com més nuclis hi hagi, és més possible que la càrrega es distribueixi de manera desigual, ja que a priori no és possible saber quant es trigarà a avaluar l'aptitud dels individus: els més aptes trigaran més a avaluar la seva aptitud, ja que les seves partides seran més llargues.

Tenint en compte aquestes dades, en aquest treball s'han utilitzat principalment dos tipus d'instàncies:

- c4-xlarge quan s'han executat diversos experiments en paral·lel per comparar-ne els resultats.
- c4-8xlarge quan s'han realitzat els experiments més llargs, per minimitzar el risc que ens retirin les instàncies de subhasta per una variació de preu.

Aquestes decisions han tingut més a veure amb la planificació temporal que amb el cost. En aquest projecte concret, s'ha considerat que tenir els resultats en el moment adequat compensava la petita diferència de preu.

Això no obstant, si només es té en compte el criteri econòmic, els resultats obtinguts posen de manifest que surt a compte utilitzar les instàncies menys potents durant més temps.

4.4. Altres solucions estudiades

A banda de la solució implementada, també s'han estudiat i descartat dues arquitectures alternatives: les funcions Lambda i un clúster MapReduce.

4.4.1. Funcions Lambda

Els principals proveïdors de computació al núvol, amb AWS al capdavall, comencen a oferir productes dins del que han anomenat "*serverless*". El producte icònic d'aquesta tecnologia és AWS Lambda, que permet oblidar-se de gestionar l'escalat de la infraestructura i pagar només pel temps d'execució de les funcions, no pel temps real en què els servidors que la suporten estan engegats.

S'ha estudiat implementar el càlcul d'aptitud dels cromosomes en una funció Lambda. D'aquesta manera, l'algorisme genètic podria cridar a aquesta funció per a obtenir l'aptitud dels individus sense haver-se de preocupar de com es paral·lelitzen les execucions, només hauria de ser capaç de fer les crides en paral·lel i esperar els resultats.

En teoria, aquesta seria una opció més bona que la triada, ja que hauria de permetre mantenir els costos més continguts i despreocupar-se d'encendre i apagar servidors. No obstant això, els costos i el rendiment se'n veuen ressentits per la manera com AWS implementa el servei[13]:

- AWS Lambda està pensat (i surt a compte) per executar tasques puntuals, no per computació intensiva.
- Quan rep una petició per executar una funció Lambda, AWS aprovisiona un servidor de poca capacitat i executa la funció.
- Si hi ha diverses peticions concurrents, reaprofita aquest servidor per executar les peticions que li van arribant.
- Quan aquest primer servidor arriba a una càrrega de treball alta, AWS aprovisiona un nou servidor, que un cop encès comença a atendre noves peticions.
- En un escenari amb moltes peticions en poc temps, es paga una gran quantitat de temps i diners en el procés d'aprovisionament. Per una banda, el temps que triga en engegar el servidor es compta com a temps d'execució de la funció. D'altra banda, els servidors tenen poca capacitat de càlcul en paral·lel i arriben sovint al límit on s'han d'aprovisionar nous servidors.

[13] *AWS Lambda: How It Works,* 2017

Aquesta opció disminueix el cost de gestionar la infraestructura, però a canvi s'ha de gestionar la complexitat de les crides a un servidor remot (retards, connexions trencades, etc.) per calcular les aptituds.

4.4.2. Clúster MapReduce

Com que es tracta d'un problema computacionalment exigent que es vol paral·lelitzar, també s'ha estudiat l'ús d'una tecnologia àmpliament utilitzada en l'actualitat, Hadoop, per treballar sota un marc de *MapReduce*.

En el nostre cas, es pot utilitzar el càlcul d'aptitud com a funció de *Map* (que assignaria un número a cada individu), mentre que l'algorisme genètic pot fer la funció de *Reduce* (que obtindria una nova generació). En aquest cas, s'hauria d'executar el cicle de *MapReduce* tantes vegades com generacions es vulguin obtenir.

AWS ofereix una solució al núvol per executar un clúster Hadoop. Hadoop, però, i en concret AWS EMR, està molt enfocat cap al tractament de Big Data. No s'ha considerat que sigui una solució adequada per aquest treball pels següents motius:

- Obliga a implementar de zero l'algorisme genètic, ja que no hi ha cap llibreria que pugui treballar amb la interfície que proporciona Hadoop Streaming.
- El temps de configuració i el manteniment de la infraestructura són més alts en una solució basada en Hadoop. S'hauria de pagar, com a mínim, un temps no despreciable en aprendre la plataforma.
- De la mateixa manera, els costos d'executar un clúster Hadoop al núvol són més alts que que la solució proposada, ja que involucren més elements pels quals es paga.
- Els beneficis que proporciona l'ús de Hadoop per un volum de dades tan petit com es gestiona en aquest treball no són clars. No hi ha cap càlcul que no es pugui dur a terme dins d'una sola màquina (ni molt menys) i s'hagi de distribuir.
- La dificultat de depurar algorismes en un clúster Hadoop és més alta que no pas quan tot s'executa en la mateixa màquina.
- Per tant, s'estaria pagant el preu d'una major complexitat i cost econòmic a canvi d'un increment poc apreciable (o negatiu) del rendiment.

5. Resultats dels experiments

Com a part dels productes obtinguts en aquest treball hi ha els fitxers amb els resultats de tots els experiments que s'han realitzat. Per a cada experiment s'inclou la sortida de l'experiment en format CSV i un gràfic de resum.

En aquest capítol s'analitzen els resultats dels experiments i s'extreuen conclusions sobre com afecten els diferents paràmetres de l'algorisme genètic a l'aptitud dels cromosomes obtinguts. Aquest anàlisi s'utilitzarà després a l'apartat 5.7 per configurar un experiment amb la configuració que es considera que maximitzarà els resultats.

En diferents apartats d'aquest capítol es farà referència als experiments, però no es reproduirà el gràfic d'evolució de l'aptitud. A l'Annex VI s'inclou un resum de tots, ordenats des de l'experiment 001 fins al 028. Els resultats en CSV s'entreguen a banda i no es reprodueixen en aquest document.

Per comparar experiments s'analitzaran diferents aspectes dels seus gràfics d'evolució:

- Millor aptitud[14] aconseguida: hauria de ser el principal factor a tenir en compte i el que determinés l'èxit d'un experiment i permetés comparar quin conjunt de paràmetres és més bo que d'altres. No obstant això, com que els cromosomes inicials són aleatoris, els experiments no es fan en igualtat de condicions i no és un indicador decisiu.
- Progressió de l'aptitud màxima: mentre que l'aptitud mitjana gairebé sempre descriu el mateix tipus de corba, en diferents experiments observem progressions de l'aptitud màxima molt diferents. És interessant fixar-se si l'aptitud màxima creix durant tot l'experiment de manera més o menys constant o fa salts molt bruscos. La pujada constant indica que hi ha prou diversitat genètica perquè els aparellaments produeixin individus més aptes. El cas contrari significa que un nou individu aleatori o una mutació han produït el salt i que, per tant, no hi ha cap garantia de seguir creixent en un futur.
- Diferència entre l'aptitud màxima i mitjana: si és petita significa que hi ha poca varietat genètica i que tots els cromosomes s'assemblen molt. En aquest cas, serà molt difícil que l'aptitud màxima segueixi creixent. Comparar aquesta diferència entre experiments pot donar una indicació de les possibilitats d'evolució futura.

[14] En aquest capítol, s'utilitza *millor aptitud* per anomenar el cromosoma amb l'aptitud més alta de l'experiment i *aptitud màxima* per referir-se a l'evolució d'aquest màxim entre generacions.

5.1. Número de partides per assignar l'aptitud

Una de les decisions que s'ha pres durant aquest projecte és que la mitjana de puntuació de les partides és una bona aproximació per descriure la qualitat d'un algorisme jugant al 2048. Un cop establert això, s'ha hagut d'establir el número òptim de partides que s'han de dur a terme per fer servir aquest valor com a aptitud dels cromosomes.

Amb un número molt alt de partides es pot considerar que la mitjana de puntuacions s'aproparà molt a l'esperança real de puntuació. Tot i així, valors alts de partides jugades suposen un cost alt de computació. Malgrat les millores en aquest sentit que suposen la paral·lelització dels càlculs, la duració de l'experiment seguirà sent proporcional al número de partides jugades pel *Jugador*.

Disminuir el número de partides jugades implicarà una millora en la velocitat de l'algorisme, que es podrà traslladar, per exemple, en la possibilitat d'utilitzar més i majors generacions. Es fa evident, per això, que si es juguen poques partides el resultat tindrà més a veure amb la sort que no pas l'aptitud real de cada algorisme concret.

Per determinar el número de partides que s'utilitzarien per la resta d'experiments s'han calculat les diferències de puntuació mitjana respecte una base (2.000 partides) per diferents valors de partides. El codi i els resultats d'aquest càlcul es poden trobar a l'Annex II.

Tal com era d'esperar, a mesura que disminueix el número de partides, augmenta la diferència mitjana, la màxima i la desviació tipus. No obstant això, i de manera sorprenent, fins i tot per valors relativament petits (10) de partides jugades els resultats es desvien menys d'un 15%.

Amb desviacions tan petites surt a compte jugar menys partides, ja que es pot suplir aquesta pèrdua de precisió avaluant més generacions, aprofitant que cada cromosoma triga fins a dues ordres de magnitud menys a avaluar-se. S'ha de tenir en compte, per això, que aquests valors tan petits de desviació es poden deure a partides que acaben molt ràpidament (perquè són algorismes molt poc aptes) i que, per tant, tenen molt poc marge de variació.

Tenint tot això en compte, s'ha decidit utilitzar 20 partides jugades per assignar l'aptitud d'un cromosoma, ja que la velocitat per poder realitzar nous experiments es considera més important en aquesta fase que no la precisió dels resultats.

5.2. Impacte del tipus de cromosoma

Per comparar com es comporta cada tipus de cromosoma s'han dut a terme un seguit d'experiments, en què s'han mantingut constants la resta de paràmetres.

Constants	
Població:	100
Generacions:	1.000
Probabilitat d'aparellament:	95%
Probabilitat de mutació:	5%
Variacions	
Experiment 001	Cromosoma 10Genes
Experiment 002	Cromosoma 100Genes
Experiment 003	Cromosoma 1000Genes
Experiment 004	Cromosoma 5000Genes
Experiment 005	Cromosoma 8bit
Experiment 006	Cromosoma 8bit1decimal
Experiment 007	Cromosoma 9bit
Experiment 008	Cromosoma 9bit1decimal
Experiment 009	Cromosoma 10bit
Experiment 010	Cromosoma 10bit1decimal

Entre els cromosomes de suma zero, el tipus 1000Genes és el que obté una millor aptitud més alta. El tipus 100Genes té una millor aptitud molt semblant, però la seva evolució és més accidentada, i durant més de la meitat de les generacions no es produeixen millores, fet que comparteix amb el tipus 5000Genes. En canvi, el tipus 10Genes té una progressió més esglaonada, però la seva millor aptitud és molt baixa.

D’altra banda, en cromosomes de valor independent es pot observar com l’evolució de l’aptitud màxima és molt plana entre els tipus de 8 i 10 bits. El tipus 9bits mostra una millora de l’aptitud constant, però amb una millor aptitud força per sota del tipus 9bits1decimal.

Es pot comprovar en els resultats dels experiments que els cromosomes que donen més bons resultats (1000Genes i 9bit1decimal) són aquells que mantenen els valors dels pesos aproximadament entre el rang [-25, 25]. La resta d’experiments tenen rangs de valors més estrets (10Genes, 100Genes i 8bit1decimal) o bé més amples (5000Genes, 8bit, 9bit, 10bit i 10bit1decimal).

D’això se’n pot extreure que quan les diferències entre els valors possibles dels pesos són molt altes, l’espai de cerca és massa gran i trobar un equilibri entre els pesos es fa més difícil. Això es tradueix en una manca de regularitat en la millora de l’aptitud màxima.

A l’altre extrem, si el rang de valors dels pesos és massa estret, no es marquen prou diferències entre ells i no compleixen la seva funció de ponderació. Els tipus de cromosoma amb aquesta característica tendeixen a tenir una aptitud màxima més estable i baixa.

5.3. Impacte de la mida de la població

Per determinar l'impacte de la mida de la població s'han dut a terme un seguit d'experiments, en què s'han mantingut constants la resta de paràmetres.

Constants	
Cromosoma:	1000Genes
Generacions:	1.000
Probabilitat d'aparellament:	95%
Probabilitat de mutació:	5%
Variacions	
Experiment 011	Població 10 individus
Experiment 012	Població 50 individus
Experiment 003	Població 100 individus
Experiment 013	Població 250 individus
Experiment 014	Població 500 individus

Es poden observar tres tendències a mesura que es fa augmentar la població dels experiments:

- Les diferències entre les aptituds mitjana i màxima tendeixen a eixamplar-se, cosa que indica una major diversitat genètica. En aquest cas sembla evident que augmentar la població ha de fer augmentar la diversitat genètica.
- La millor aptitud tendeix a ser més alta. Això és consistent amb la idea intuïtiva que com més individus tinguem, hi ha més possibilitats de tenir individus molt aptes.
- La millora de l'aptitud màxima és més menys constant. En una població més gran, els individus més aptes tenen menys possibilitats d'aparellar-se entre ells i les millores en l'aptitud màxima passen menys sovint.

En experiments amb generacions infinites, aquells amb major població sempre tindran una millor aptitud més alta. Tanmateix per a números de generacions concrets, una població més gran pot suposar una millor aptitud més baixa, ja que la millora de l'aptitud màxima es produeix de manera menys freqüent.

5.4. Impacte del número de generacions

Per determinar l'impacte del número de generacions s'han dut a terme un seguit d'experiments, en què s'han mantingut constants la resta de paràmetres.

Constants	
Cromosoma:	9bit1decimal
Població:	100
Probabilitat d'aparellament:	95%
Probabilitat de mutació:	5%
Variacions	
Experiment 015	100 generacions
Experiment 016	500 generacions
Experiment 008	1.000 generacions
Experiment 017	2.000 generacions
Experiment 018	5.000 generacions

La conclusió en aquest cas és força clara: a mesura que augmenta el número de generacions es produeixen més millores en l'aptitud màxima. Això no resulta gens sorprenent, ja que amb més generacions es proven més combinacions diferents i sempre es guarden els millors resultats.

En els experiments observats no hi ha una relació directa entre el número de generacions i la millor aptitud. Això podria ser perquè els experiments no parteixen de les mateixes condicions inicials. Com a regla general, si es produeixen més millores en l'aptitud màxima, s'arribarà a una millor aptitud més alta.

5.5. Impacte de la probabilitat d'aparellament

Per determinar l'impacte de la probabilitat d'aparellament s'han dut a terme un seguit d'experiments, en què s'han mantingut constants la resta de paràmetres.

Constants	
Cromosoma:	9bit1decimal
Població:	100
Generacions:	1.000
Probabilitat de mutació:	5%
Variacions	
Experiment 019	Probabilitat d'aparellament del 99%
Experiment 008	Probabilitat d'aparellament del 95%
Experiment 020	Probabilitat d'aparellament del 90%
Experiment 021	Probabilitat d'aparellament del 80%
Experiment 022	Probabilitat d'aparellament del 65%
Experiment 023	Probabilitat d'aparellament del 50%

Tot i que no s'observen grans diferències entre els experiments, es poden realitzar un parell d'observacions:

- La cota superior (99%) de probabilitat d'aparellament presenta una evolució de l'aptitud màxima molt poc constant, probablement per una manca de varietat genètica, ja que s'observa poca diferència entre les aptituds màximes i mitjanes.
- La cota inferior (50%) presenta el mateix problema, però per causes diferents: en cada generació entren molts genomes aleatoris, que s'aparellen entre ells i dilueixen la importància dels gens que havien donat lloc a millores en l'aptitud.
- En la resta de valors no es pot apreciar una tendència clara.

No es pot establir amb seguretat quina és la probabilitat d'aparellament òptima, tot i que s'estima que ha d'estar entre el 65% i el 95%.

5.6. Impacte de la probabilitat de mutació

Per determinar l'impacte de la probabilitat de mutació s'han dut a terme un seguit d'experiments, en què s'han mantingut constants la resta de paràmetres.

Constants	
Cromosoma:	9bit1decimal
Població:	100
Generacions:	1.000
Probabilitat d'aparellament:	5%
Variacions	
Experiment 024	Probabilitat de mutació de l'1%
Experiment 008	Probabilitat de mutació del 5%
Experiment 025	Probabilitat de mutació del 10%
Experiment 026	Probabilitat de mutació del 25%
Experiment 027	Probabilitat de mutació del 50%

Amb els experiments duts a terme no s'ha pogut determinar quin és l'impacte de la probabilitat de mutació sobre els resultats. No es poden trobar patrons en els gràfics de resultats que indiquin quins valors milloren els resultats. És cert que l'experiment 027 mostra una variació gairebé nul·la de l'aptitud màxima, però s'ha de tenir en compte que es troba un individu amb una aptitud alta molt aviat.

S'ha de considerar que el mecanisme de mutació implementat només canvia un gen del cromosoma cada vegada, la qual cosa fa que tingui un impacte molt petit sobre el resultat final de l'experiment.

5.7. Experiment final

Per a l'experiment final s'han utilitzat els paràmetres que en els apartats anteriors han donat resultats més bons.

Experiment final: 028	
Cromosoma:	9bit1decimal
Població:	200
Generacions:	8.000
Probabilitat d'aparellament:	80%
Probabilitat de mutació:	10%

En aquest cas, era desitjable realitzar un experiment amb una població més gran i més generacions. Això no obstant, les restriccions de temps (vegeu el capítol 4.3) han obligat a reduir aquests valors. Aquest experiment ha trigat aproximadament 30 hores en una instància de tipus c4.8xlarge. Si es volgués, per exemple, augmentar la població fins a 1.000 i les generacions a 15.000, l'experiment hagués trigat més d'una setmana.

Els resultats d'aquest experiment no es poden considerar exitosos. La millor aptitud obtinguda (1.650) està per sota de molts altres experiments en què s'han utilitzat molts menys recursos. La intenció d'aquest experiment era combinar els paràmetres amb un impacte positiu més gran sobre el resultat per aconseguir la millor puntuació de tot el treball, però malauradament no ha sigut així.

Hi ha diversos motius possibles per explicar-ho:

- Les probabilitats d'aparellament i de mutació actuen conjuntament per marcar com evoluciona la població i no se n'hauria d'analitzar el comportament per separat. En comptes d'això, s'hauria d'haver observat com es comporten les variacions de les dues variables juntes.
- És possible que s'hagi arribat a un límit en l'aproximació seguida. Les millors aptituds de tots els experiments es mouen en valors similars (entre 1.500 i 1.900). Això pot indicar que la funció d'avaluació, tal com està plantejada, no és un bon mecanisme per determinar el potencial d'un tauler de 2048.

- Finalment, no s'ha de descartar el factor atzar. En els algoritmes genètics tant la generació inicial com els aparellaments o les mutacions es produeixen de manera aleatòria. És possible que la sort jugui un paper més important que la configuració de l'algoritme a l'hora d'obtenir la millor aptitud.

L'única bona noticia és que el creixement de l'aptitud màxima es manté d'una manera més o menys constant. Això significa que, amb prou generacions, és possible que aquesta configuració acabés donant resultats positius.

6. Conclusions

Per a realitzar aquest treball s'han invertit més de tres mesos de feina i s'han jugat, de manera automàtica, més de cent milions de partides partides de 2048, comptant només els resultats presentats en aquest document. En aquest capítol es justifica el seguiment de la planificació, el compliment dels objectius inicials i les línies de treball futur que no s'han pogut explorar en l'àmbit d'aquest treball.

6.1. Seguiment de la planificació

En general, el treball ha evolucionat seguint la planificació establerta. Els blocs de treball flexibles han permès que els imprevistos menors s'anessin corregint dins de cada entrega i els objectius estaven clars en cada moment.

Tanmateix, l'abast del treball era molt ambiciós, i s'ha hagut de fer un esforç per mantenir el focus durant el desenvolupament i aconseguir complir els objectius. Treballar amb una estratègia de millora incremental ha ajudat a acotar l'abast dins del temps disponible (intentar portar a terme alguna de les millores de l'apartat 6.3 hauria suposat no poder acabar el treball).

Els principals punts de desviació han set els següents:

- Un canvi de feina a l'inici del projecte va tenir un impacte en l'entrega de la PAC2, que no va aconseguir tots els objectius en el termini previst: tant el codi com la documentació es trobaven lleugerament endarrerits respecte la planificació. Aquesta desviació es va compensar durant la PAC3 sense cap esforç afegit.
- Davant la manca de progrés en els resultats, a l'inici de l'entrega final s'han tornat a revisar punts ja acabats en iteracions anteriors. Això ha suposat un esforç més gran del previst en l'última etapa del projecte.
- La memòria del projecte s'ha revisat i adaptat en cada entrega. Malgrat que aquest era un esforç previst, el *feedback* obtingut del consultor i d'altres revisors ha suggerit millores que han incrementat el cost previst inicialment.

La crítica més important a la planificació és no haver començat abans a fer experiments de llarga durada. Això ha resultat un coll d'ampolla perquè fins a l'inici de l'entrega final no s'ha pogut detectar que els resultats no eren tan bons com els esperats, i en aquell moment el marge de maniobra era molt petit per poder refer els passos anteriors.

En l'etapa de planificació s'hauria d'haver detectat que això podria ser un problema i haver posat èmfasi en un prototipatge més ràpid de la solució completa per obtenir resultats més de pressa.

Cal tenir en compte, per això, que probablement només s'hauria pogut mitigar lleugerament aquest problema: abans de poder començar a fer experiments de llarga durada, tant l'algorisme genètic com l'estratègia d'execució al núvol havien d'estar en un estat avançat de desenvolupament.

6.2. Compliment dels objectius

Al capítol 1.2 es plantegen tres objectius per a aquest treball: l'estudi dels algoritmes genètics per optimitzar una funció, dissenyar un sistema que permeti executar aquests algorismes en paral·lel i, finalment, obtenir un solucionador de 2048.

6.2.1. Estudi dels algorismes genètics

Malgrat que no s'han obtingut bons resultats (vegeu l'apartat 6.1.3), es considera que s'ha complert aquest objectiu: durant el decurs d'aquest treball s'ha implementat un algorisme genètic des de zero i s'han dissenyat experiments per comprovar-ne la seva eficàcia.

S'han treballat aspectes com la selecció, l'aparellament i la mutació de cromosomes, s'han dissenyat diferents tipus de cromosomes i s'ha implementat una manera fàcil de fer experiments escollint els seus paràmetres.

Es podrien haver implementat més estratègies de selecció, aparellament i mutació, però la durada dels experiments hagués fet impossible provar el seu impacte. Per aquest motiu, es va considerar millor escollir només una estratègia i utilitzar-la per a tots els experiments.

No obstant això, hi ha una funcionalitat que, vist en retrospectiva, hauria aportat molt de valor al desenvolupament del treball: l'habilitat de guardar el progrés d'un experiment i poder-lo reprendre a continuació. Això, a banda d'evitar perdre el progrés, permetria dur a terme diferents experiments amb la mateixa població inicial per valorar amb més efectivitat l'impacte de la resta de paràmetres.

6.2.2. Execució en paral·lel i al núvol

Dels tres objectius del treball, en aquest s'han obtingut els resultats més positius. Es considera que la solució entregada és molt bona tècnicament, ja que permet:

- Fer experiments molt exigents en temps de processador utilitzant la infraestructura d'AWS de manera totalment desassistida i gairebé transparent a l'usuari.
- Maximitzar el rendiment que es treu a cada instància aprofitant tots els nuclis del seu processador.
- Aprofitar el mecanisme de subhasta d'instàncies per reduir dràsticament el cost dels experiments.
- Estimar el cost (utilitzant la taula 2) en temps i diners d'un experiment abans de portar-lo a terme.
- Modular aquests costos utilitzant diferents tipus d'instàncies sense complexitat addicional.

Aquest treball s'ha centrat a utilitzar els serveis d'AWS com a proveïdor, però aquesta solució permet, amb canvis molt petits, utilitzar la mateixa aproximació amb les infraestructures al núvol de Google, Microsoft o qualsevol altre proveïdor. Per a cadascun, les estratègies de preu seran diferents, però la majoria dels mòduls són reaprofitables.

6.2.3. Solucionador de 2048

El tercer objectiu, tanmateix, no es pot considerar assolit. Els solucionadors obtinguts compleixen les restriccions establertes (no exploren l'arbre de possibles taulers i, per tant, són molt ràpids), però es queden molt lluny de guanyar el joc de manera consistent.

Com es pot veure a l'Annex III, la puntuació mitjana de les partides jugades aleatòriament és de 1.095 punts, aproximadament. D'altra banda, per guanyar el joc de 2048 s'han d'obtenir, com a mínim, 20.000 punts. De tots els experiments realitzats, la puntuació (mitjana) més alta obtinguda és de 1.909 punts. Això situa l'estratègia seguida en aquest treball més a prop de jugar aleatòriament que d'aconseguir guanyar el joc.

Es contemplen dos motius per explicar la manca de resultats en aquest punt:

- La funció d'avaluació de taulers dissenyada pot ser ineficaç per determinar la qualitat d'un tauler de 2048. Els atributs per avaluar el tauler s'han escollit d'una manera subjectiva, i és possible que les tesis de l'autor sobre com avaluar un tauler siguin equivocades. En aquest sentit, hauria set molt positiu fer proves amb funcions d'avaluació diferents.
- Els experiments proposats tenen un cost de computació molt alt. Malgrat que això se sabia des de la planificació del treball, s'ha subestimat el temps necessari per a cada experiment. Això comporta dos problemes:
    - El disseny de nous experiments sovint té a veure amb resultats d'experiments anteriors. Amb una durada dels experiments tan alta, el número d'iteracions que es poden realitzar baixa. Com a conseqüència, es poden provar menys variants que potencialment generarien millors resultats.
    - Per a cada configuració de paràmetres només s'ha pogut realitzar un sol experiment. Idealment, s'haurien de dur a terme diversos experiments amb cada configuració per poder extreure'n conclusions estadísticament significatives. Durant el treball, aquest fet pot haver dut a treure conclusions que no eren acurades.
    - Alguns experiments no han pogut tenir el número de generacions desitjades. L'algorisme dissenyat espera una millora contínua de l'aptitud: donades prou generacions es podrien haver obtingut millors resultats.

6.3. Línies de treball obertes

Com s'explica al capítol 1.3, s'ha utilitzat una aproximació progressiva per tal d'assegurar que el treball es porta a terme en el termini establert. Aquests són alguns dels aspectes que es podrien millorar si es disposés de més temps

- Utilitzar un llenguatge de programació de més baix nivell. Utilitzar Perl 5 ha permès fer iteracions de millora contínua molt curtes. No obstant això, convertir el codi a un llenguatge de més baix nivell permetria millorar els temps d'execució i portar a terme més experiments en menys temps. Aquesta millora permetria treballar en alguns dels punts següents.
- Dur a terme més experiments amb els mateixos paràmetres per obtenir resultats més fiables. Les limitacions de temps i pressupost han fet que només s'hagi dut a terme un sol experiment amb cada configuració de paràmetres. Això pot portar a algunes conclusions equivocades per culpa del factor atzar.
- Implementar i provar altres mètodes de selecció, aparellament i mutació, i també altres tipus de cromosomes.
- Observar quins pesos tenen menys rellevància i eliminar-los de la funció d'avaluació. Això hauria de permetre millorar els resultats més ràpidament, ja que s'haurien d'optimitzar menys paràmetres diferents.
- Aprofitar l'ecosistema d'AWS per afegir millores a l'execució remota d'experiments. Alguns exemples podrien ser enviar un correu cada cop que s'acaba un experiment o generar gràfiques i resums dels experiments automàticament.

Alguns d'aquests punts són prou importants per justificar l'extensió d'un nou treball. Tanmateix, la línia de treball oberta més important és utilitzar la tècnica per executar algorismes genètics al núvol desenvolupada en aquest treball i aplicar-la a altres problemes.

Obtenir un solucionador de 2048 és un problema amb un impacte molt menor. Fins i tot és possible que no sigui un problema adequat per aplicar-hi algorismes genètics (ja que hi ha solucions molt bones aplicant altres mètodes). Alxò no obstant, hi ha altres problemes interessants on l'habilitat de realitzar experiments sota demanda sense haver de disposar d'una infraestructura pròpia pot marcar una diferència molt important.

7. Glossari

Definició dels termes i acrònims més rellevants utilitzats en aquest treball que no es descriuen en la resta de capítols:

API (Application Programming Interface): Conjunt de mètodes que un sistema defineix i exposa per poder-se comunicar amb ell. En l'àmbit d'aquest treball, aquesta comunicació es porta a terme a través del protocol HTTP.

AWS (Amazon Web Services): Conjunt de serveis al núvol oferts per l'empresa Amazon. Inclou computació, emmagatzematge, bases de dades i d'altres serveis distribuits.

AWS EC2 (Elastic Cloud Computing): Servei de computació al núvol sota demanda. Permet llogar màquines virtuals (anomenades instàncies) i pagar només pel temps efectiu d'ús.

AWS EMR (Elastic Map Reduce): Servei d'allotjament i gestió de clústers Hadoop sota demanda.

AWS IAM (Identity and Access Management): Servei gratuït d'autorització a la resta de serveis d'AWS.

AWS Lambda: Plataforma serverless de computació al núvol. Permet executar codi en resposta a events (crides HTTP, fitxers pujats a S3, missatges a serveis de cues, etc.) i en gestiona automàticament la infraestructura necessària.

AWS S3 (Simple Storage Service): Servei d'emmagatzematge d'objectes distribuït. Permet guardar i recuperar fitxers sense haver de reservar un espai de disc concret. El servei permet la replicació automàtica del contingut i cobra per l'espai utilitzat i l'accés als fitxers.

CSV (Comma separated value): Tipus de document senzill per representar dades en forma de taula, en què les columnes se separen per comes i les files per salts de línia.

Big Data: Conjunts de dades tan grans o complexes que no es poden tractar en sistemes informàtics tradicionals. Generalment, aquestes dades són més grans que la memòria disponible en un sol sistema.

Fork: Crida dels sistemes UNIX que permet a un procés crear una copia de si mateix, que actua com a procés fill del procés original. Els processos resultants d'un *fork* tenen el mateix codi i una copia de la memòria del pare en el moment de fer el *fork*.

Hadoop: *framework* de codi obert, mantingut per la fundació Apache, per programar aplicacions distribuïdes que processin grans quantitats de dades fent servir un model *MapReduce*.

MapReduce: Model de programació per a processar conjunts de dades grans amb un algorisme paral·lel i distribuït entre diferents nodes d'un clúster. Es compon d'un procés de *Map*, que filtra i ordena les dades i un procés de *Reduce*, que fa una operació d'agregació.

Serverless: Les arquitectures serverless substitueixen les màquines virtuals de llarga duració per la capacitat de computació efímera que es crea per atendre a una petició i desapareix després del seu ús. El seu nom (literalment, "sense servidor") significa que és el proveïdor del servei el que s'encarrega de gestionar els servidors que gestionen l'execució de codi i que això es fa de manera transparent a l'usuari.

8. Bibliografia

Dyer, Daniel W. (2010). "*Evolutionary Computation in Java*" [llibre en línia, data de consulta: maig de 2017].
<http://watchmaker.uncommons.org/manual/index.html>

Mitchell, Melanie (1999) *An Introduction to Genetic Algorithms (Complex Adaptive Systems)* (5a. ed.). Cambridge: The MIT Press

Statt, Nick (2014, març). "*2048 starts easy; gets hard. Here's how to make it easy again*". *CNET* [artícle en línia, data de consulta: febrer de 2017].
<https://www.cnet.com/news/2048-starts-easy-gets-hard-heres-how-to-make-it-easy-again/>

Umbarkar and Seth (2015). "*Crossover Operators in Genetic Algorithms: a Review*" [artícle en línia, data de consulta: març de 2017].
<http://ictactjournals.in/paper/IJSC_V6_I1_paper_4_pp_1083_1092.pdf>

AWS Lambda: How It Works [documentació oficial d'AWS, data de consulta: març de 2017]
<http://docs.aws.amazon.com/lambda/latest/dg/lambda-introduction.html>

The Comprehensive Perl Archive Network [CPAN, data de consulta: maig de 2017]
<http://www.cpan.org>

Using Spot Instances [documentació oficial d'AWS, data de consulta: març de 2017]
<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances.html>

What is the optimal algorithm for the game 2048? [Stackoverlfow, data de consulta: febrer de 2017]
<https://stackoverflow.com/questions/22342854/what-is-the-optimal-algorithm-for-the-game-2048>

9. Annexos

En aquest capítol s'inclouen resultats, descripcions d'operacions, porcions de codi especialment rellevants i, en general, material complementari sobre el desenvolupament del treball.

9.1. Annex I: Instal·lació i execució d'experiments

A continuació, es detallen tots els passos necessaris per instal·lar i executar el codi desenvolupat sobre un sistema basat en Ubuntu.

9.1.1. Instal·lació

- Descarregar el codi:

```
git clone https://github.com/meis/2048GA/
cd 2048GA/
```

- Instal·lació de les dependències de sistema:

```
./script/install_dependencies.sh
```

- Instal·lació de les llibreries de Perl necessàries:

```
carton install
```

9.1.2. Execució d'experiments

- Per executar un experiment, es pot utilitzar la comanda:

```
carton exec -- script/run_experiment.pl
```

- Es poden veure totes les opcions d'execució amb el paràmetre '-h':

```
carton exec -- script/run_experiment.pl -h
run_experiment.pl [-h] [long options...] <some-arg>
       --generations INT  Number of generations
       --population INT   Players in each generation
       --games INT        Number of games for fitness

       --crossover NUM    Crossover rate
       --mutation NUM     Mutation rate

       --chromosome STR   Chromosome class to use

       --forks INT        Number of forks

       --stdout           Write results to stdout insted of a file

       -h --help          print usage message and exit
```

9.2. Annex II: impacte del número de partides en l'avaluació de l'aptitud

A continuació es detalla el codi que s'ha utilitzat per determinar el número de partides que han de determinar l'aptitud d'un cromosoma. Sobre una població de 100 individus amb genomes aleatoris s'han jugat números cada vegada menors de partides per comprovar com s'allunya la mitjana de puntuació a mesura que disminueixen el nombre de partides jugades.

Per a cada quantitat de partides jugades es calcula la diferència respecte el valor de referència (2.000 partides, en aquest cas) en cada individu. Després, es calculen la mitjana, el màxim i la desviació tipus dels agregats.

El codi (es pot trobar al fitxer *script/test_number_of_plays_difference.pl*) és el següent:

```
#!/usr/bin/env perl
use v5.10;
use FindBin;
use lib "$FindBin::Bin/../lib";
use Chromosome;
use Player;
use List::Util qw/max/;
use Statistics::Basic qw(:all);

my $tests = 100;
my @times = (2000, 1000, 500, 100, 50, 20, 10, 5, 1);
my @deviations = map { [] } @times;

for my $test (1..$tests) {
	my $bits = [ map { int(rand(2)) % 2} 0..39 ];
	my $reference;

	my $n = 0;

	for my $how_much (@times) {
		my $chromosome = Chromosome->new({ genes => $bits });
		my $player = Player->new({ chromosome => $chromosome });
		my $fitness = $player->play($how_much);

		if ($n == 0) {
			$reference = $fitness;
		}
		else {
			my $diff = (($fitness - $reference) / (($fitness +
$reference) / 2)) * 100;
			push @{$deviations[$n]}, abs $diff;
		}

		$n++;
	}
}
```

```
say "Deviations:";
for my $d (1..@times -1) {
       say '* Playing '. sprintf("%5s", $times[$d]) . ' times:'
       . '  Mean: '    . sprintf("%03.2f%", mean @{$deviations[$d]})
       . '  Max: '      . sprintf("%03.2f%", max @{$deviations[$d]})
       . '  Std: '      . sprintf("%03.2f%", stddev @{$deviations[$d]});
}
```

La seva execució ha retornat el resultat següent:

```
Deviations:
* Playing  1000 times:  Mean:  1.67% Max:  5.65%  Std:  1.25%
* Playing   500 times:  Mean:  1.97% Max: 10.41%  Std:  1.68%
* Playing   100 times:  Mean:  4.53% Max: 15.47%  Std:  3.34%
* Playing    50 times:  Mean:  5.61% Max: 18.53%  Std:  4.00%
* Playing    20 times:  Mean:  8.99% Max: 35.39%  Std:  6.31%
* Playing    10 times:  Mean: 13.90% Max: 46.85%  Std:  9.01%
* Playing     5 times:  Mean: 16.77% Max: 55.25%  Std: 13.64%
* Playing     1 times:  Mean: 35.51% Max: 111.72% Std: 24.67%
```

9.3. Annex III: puntuació mitjana de partides jugades aleatòriament

A continuació es reprodueix el codi que s'ha utilitzat per determinar la puntuació mitjana que s'obté jugant partides aleatòries de 2048.

```
#!/usr/bin/env perl
use v5.10;
use FindBin;
use lib "$FindBin::Bin/../lib";

use Board;
use List::Util qw/shuffle/;

my $number_of_plays = shift || 1;
play($number_of_plays);

sub play {
	my $self = shift;
	my $times = shift || 1;
	my $total_score = 0;

	say "Playing $number_of_plays random games";

	for (0..$number_of_plays -1) {
	my $board = Board->new;

	while (!$board->finished) {
		my @moves = shuffle $board->available_moves;
		$board = $board->move(shift @moves);
	}

	$total_score += $board->score;
	}

	say "Average score: " . $total_score / $number_of_plays;
}
```

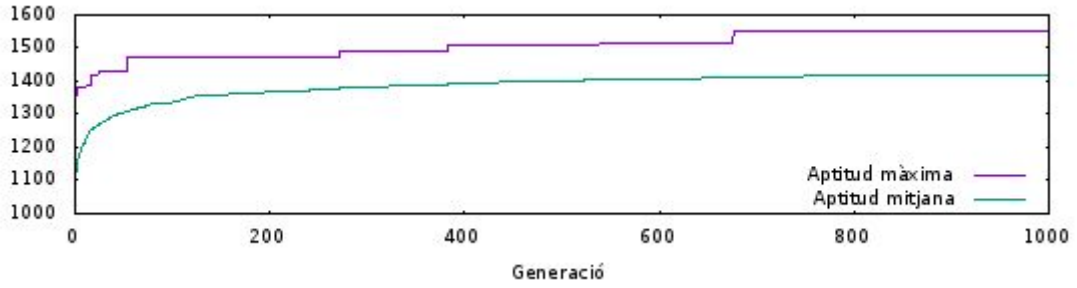
Aquest mètode (que es pot trobar al fitxer *script/random_play.pl*) s'ha utilitzat per jugar 100.000 partides aleatòries i determinant que la puntuació mitjana esperada és d'aproximadament 1.095 punts.

```
Playing 100000 random games
Average score: 1095.25024
```

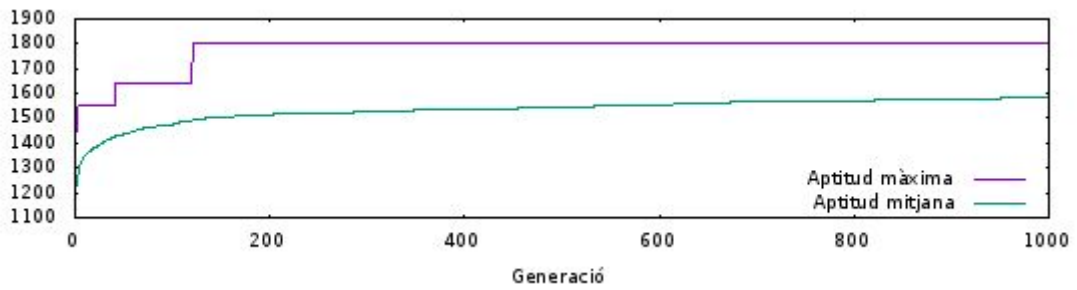
9.4. Annex IV: Gràfiques dels experiments

A continuació es llisten les gràfiques que resumeixen els experiments portats a terme en aquest treball. Cada gràfica reflecteix l'evolució de l'aptitud mitjana i del millor individu a través de les generacions.

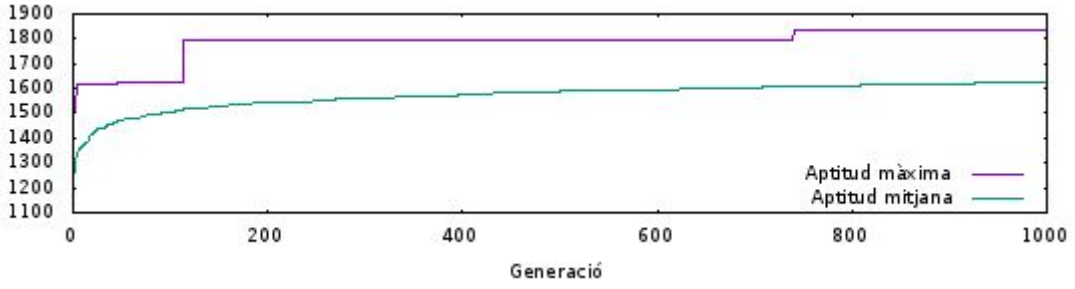
Experiment 001			
Població:	100	**Cromosoma:**	10Genes
Generacions:	1.000	**Prob. aparellament:**	95%
Millor aptitud:	1.544	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__0Sum__10Genes_generations=1000_population=100_games=20_crossover=0.95_mutation=0.05_1494829081.csv		



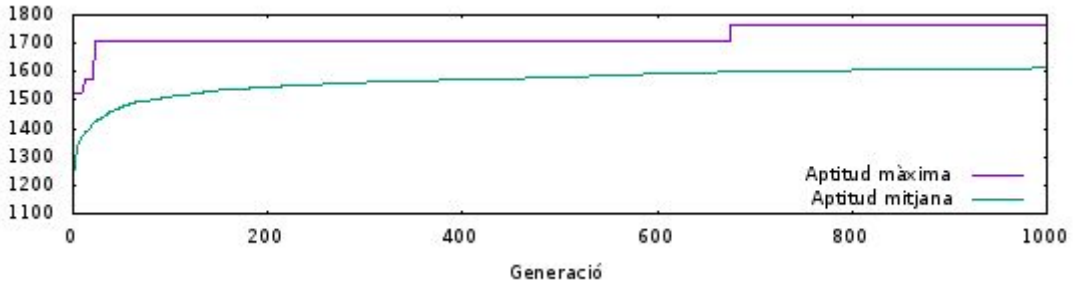
Experiment 002			
Població:	100	**Cromosoma:**	100Genes
Generacions:	1.000	**Prob. aparellament:**	95%
Millor aptitud:	1.806,2	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__0Sum__100Genes_generations=1000_population=100_games=20_crossover=0.95_mutation=0.05_1494829200.csv		



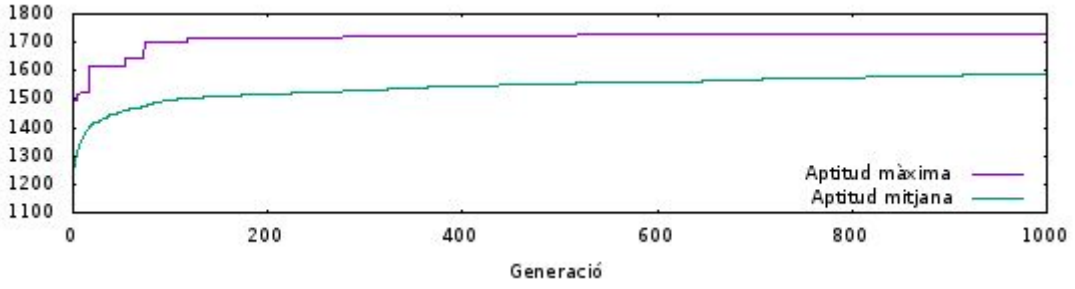
Experiment 003			
Població:	100	**Cromosoma:**	1000Genes
Generacions:	1.000	**Prob. aparellament:**	95%
Millor aptitud:	1.837,2	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__0Sum__1000Genes_generations=1000_population=100_games=20_crossover=0.95_mutation=0.05_1494829209.csv		



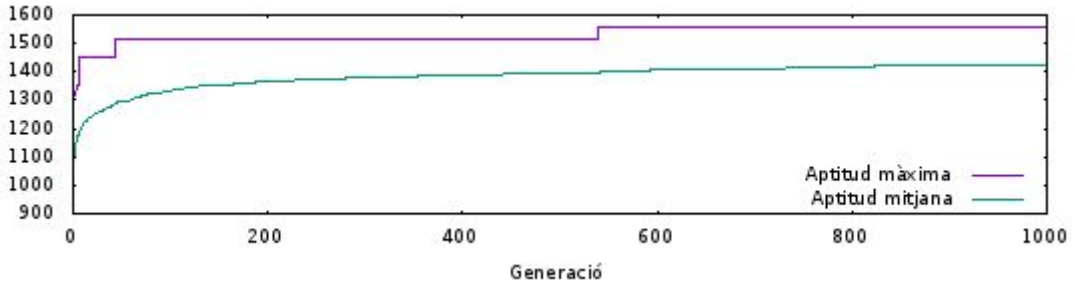
Experiment 004			
Població:	100	**Cromosoma:**	5000Genes
Generacions:	1.000	**Prob. aparellament:**	95%
Millor aptitud:	1.767,6	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__0Sum__5000Genes_generations=1000_population=100_games=20_crossover=0.95_mutation=0.05_1495300696.csv		



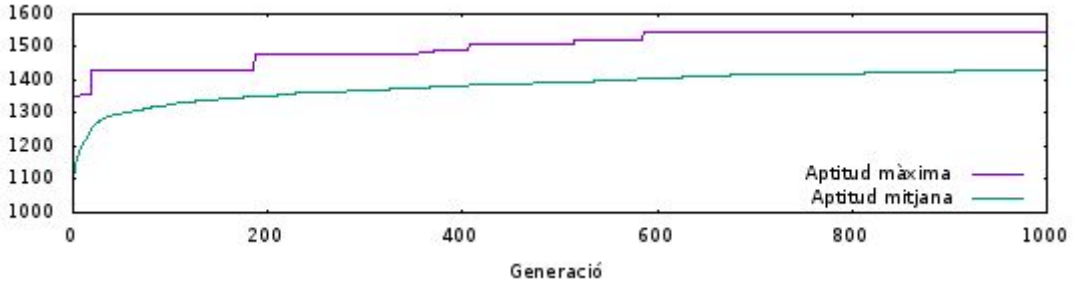
Experiment 005			
Població:	100	**Cromosoma:**	8bit
Generacions:	1.000	**Prob. aparellament:**	95%
Millor aptitud:	1.732,6	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__Independent__8bit_generations=1000_population=100_games=20_crossover=0.95_mutation=0.05_1494749052.csv		



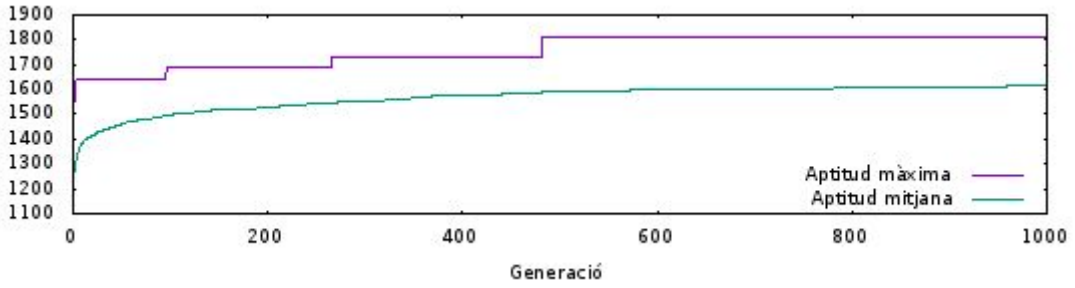
Experiment 006			
Població:	100	**Cromosoma:**	8bit1decimal
Generacions:	1.000	**Prob. aparellament:**	95%
Millor aptitud:	1.555	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__Independent__8bit1decimal_generations=1000_population=100_games=20_crossover=0.95_mutation=0.05_1494763414.csv		



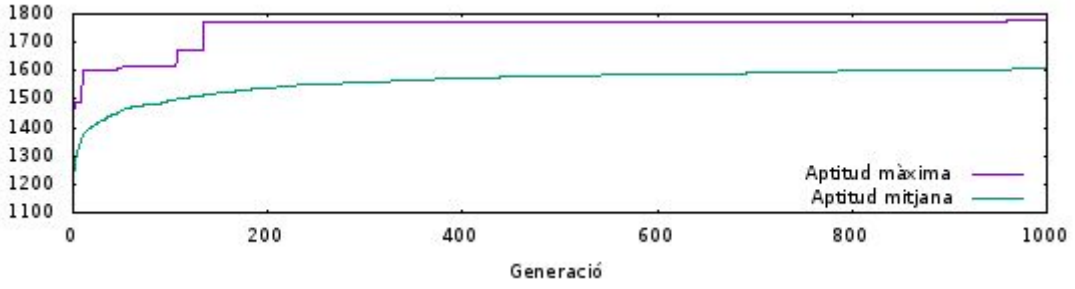
Experiment 007			
Població:	100	**Cromosoma:**	9bit
Generacions:	1.000	**Prob. aparellament:**	95%
Millor aptitud:	1.548	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__Independent__9bit_generations=1000_population=100_games=20_crossover=0.95_mutation=0.05_1494756847.csv		



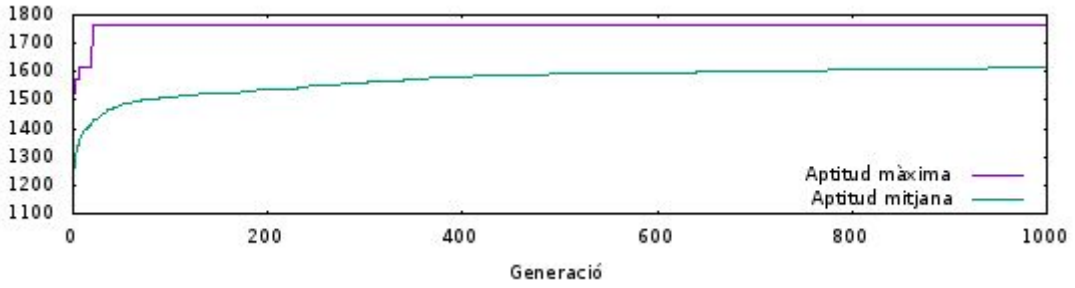
Experiment 008			
Població:	100	**Cromosoma:**	9bit1decimal
Generacions:	1.000	**Prob. aparellament:**	95%
Millor aptitud:	1.814,2	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__Independent__9bit1decimal_generations=1000_population=100_games=20_crossover=0.95_mutation=0.05_1494763402.csv		



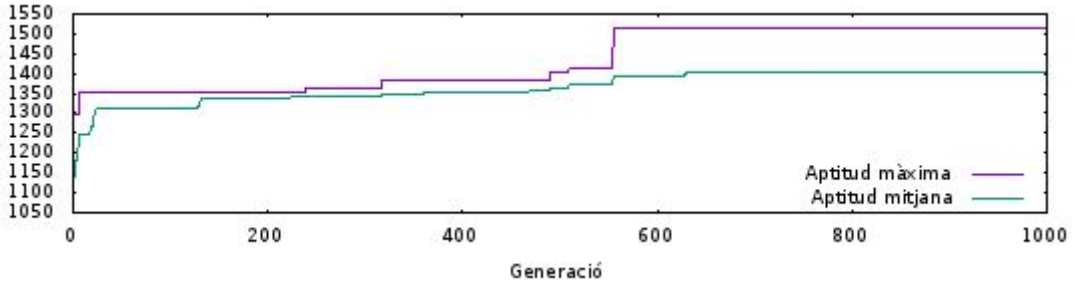
Experiment 009			
Població:	100	**Cromosoma:**	10bit
Generacions:	1.000	**Prob. aparellament:**	95%
Millor aptitud:	1.776,4	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__Independent__10bit_generations=1000_population=100_games=20_crossover=0.95_mutation=0.05_1494757119.csv		



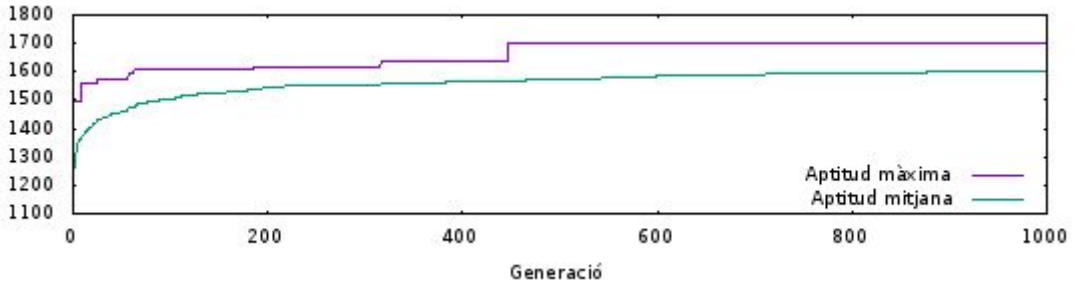
Experiment 010			
Població:	100	**Cromosoma:**	10bit1decimal
Generacions:	1.000	**Prob. aparellament:**	95%
Millor aptitud:	1.764,8	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__Independent__10bit1decimal_generations=1000_population=100_games=20_crossover=0.95_mutation=0.05_1494763392.csv		



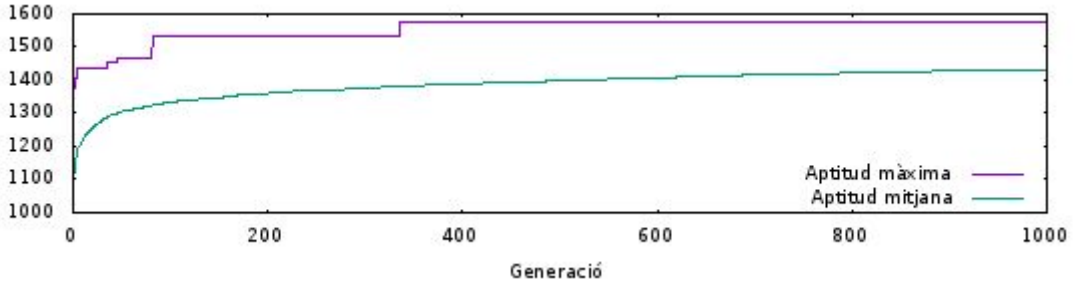
Experiment 011			
Població:	10	**Cromosoma:**	1000Genes
Generacions:	1.000	**Prob. aparellament:**	95%
Millor aptitud:	1.516,4	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__0Sum__1000Genes_generations=1000_population=10_games=20_crossover=0.95_mutation=0.05_1495364873.csv		



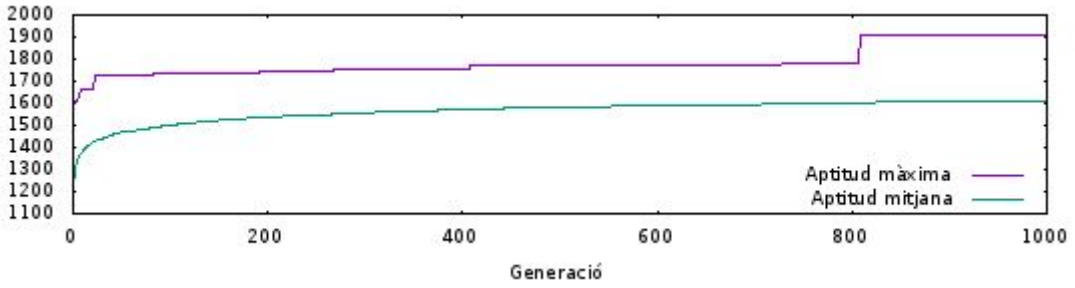
Experiment 012			
Població:	50	**Cromosoma:**	1000Genes
Generacions:	1.000	**Prob. aparellament:**	95%
Millor aptitud:	1.703,6	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__0Sum__1000Genes_generations=1000_population=50_games=20_crossover=0.95_mutation=0.05_1495364886.csv		



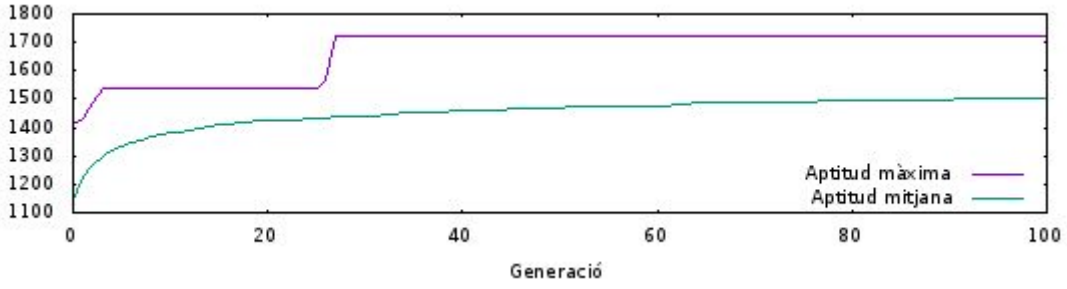
Experiment 013			
Població:	250	**Cromosoma:**	1000Genes
Generacions:	1.000	**Prob. aparellament:**	95%
Millor aptitud:	1.575,4	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__0Sum__1000Genes_generations=1000_population=250_games=20_crossover=0.95_mutation=0.05_1495364843.csv		



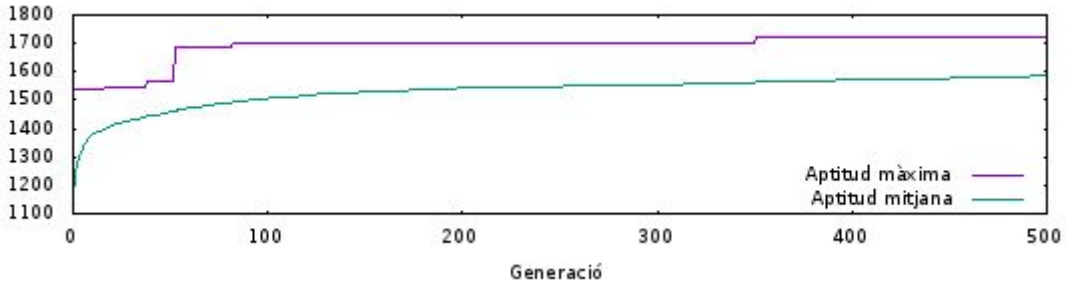
Experiment 014			
Població:	500	**Cromosoma:**	1000Genes
Generacions:	1.000	**Prob. aparellament:**	95%
Millor aptitud:	1.909	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__0Sum__1000Genes_generations=1000_population=500_games=20_crossover=0.95_mutation=0.05_1495364850.csv		



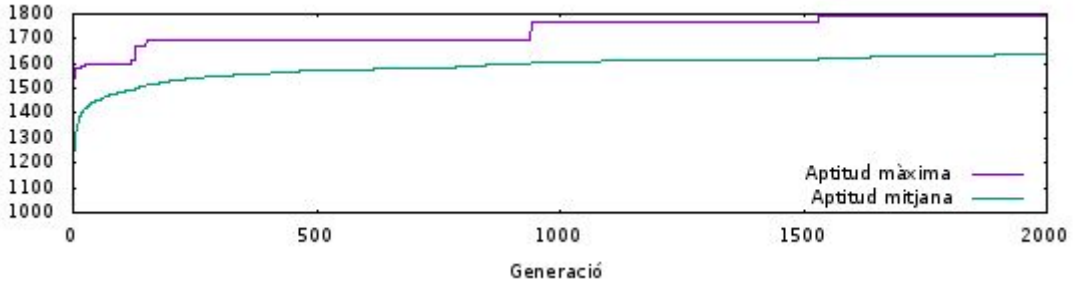
Experiment 015			
Població:	100	**Cromosoma:**	9bit1decimal
Generacions:	100	**Prob. aparellament:**	95%
Millor aptitud:	1.712,21	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__Independent__9bit1decimal_generations=100_population=100_games=20_crossover=0.95_mutation=0.05_1495469107.csv		



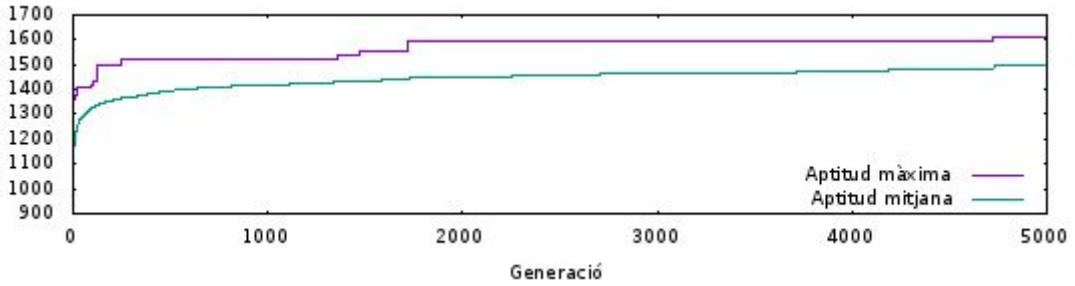
Experiment 016			
Població:	100	**Cromosoma:**	9bit1decimal
Generacions:	500	**Prob. aparellament:**	95%
Millor aptitud:	1.729,4	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__Independent__9bit1decimal_generations=500_population=100_games=20_crossover=0.95_mutation=0.05_1495435449.csv		



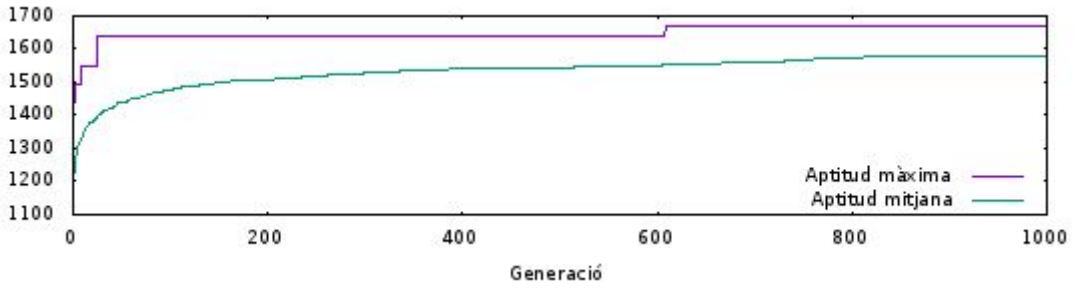
Experiment 017			
Població:	100	**Cromosoma:**	9bit1decimal
Generacions:	2.000	**Prob. aparellament:**	95%
Millor aptitud:	1.793,6	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__Independent__9bit1decimal_generations=2000_population=100_games=20_crossover=0.95_mutation=0.05_1495378364.csv		



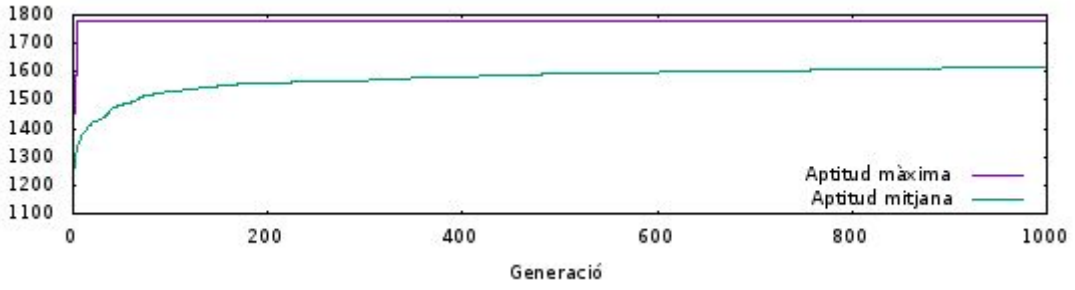
Experiment 018			
Població:	100	**Cromosoma:**	9bit1decimal
Generacions:	5.000	**Prob. aparellament:**	95%
Millor aptitud:	1.608,6	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__Independent__9bit1decimal_generations=5000_population=100_games=20_crossover=0.95_mutation=0.05_1495365000.csv		



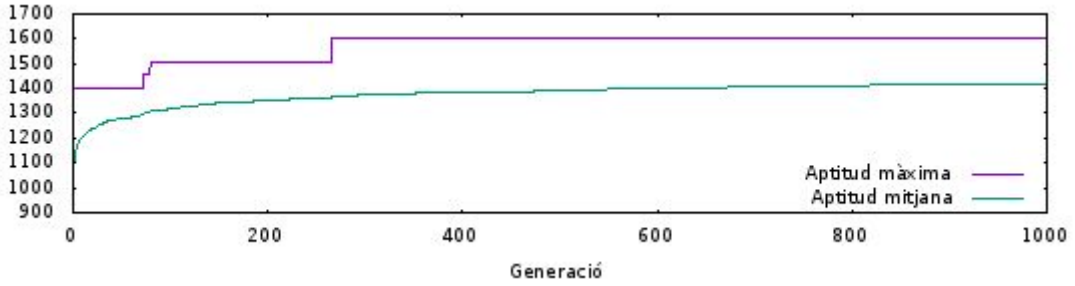
Experiment 019			
Població:	100	**Cromosoma:**	9bit1decimal
Generacions:	1.000	**Prob. aparellament:**	99%
Millor aptitud:	1.670,8	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__Independent__9bit1decimal_generations=1000_population=100_games=20_crossover=0.99_mutation=0.05_1495610033.csv		



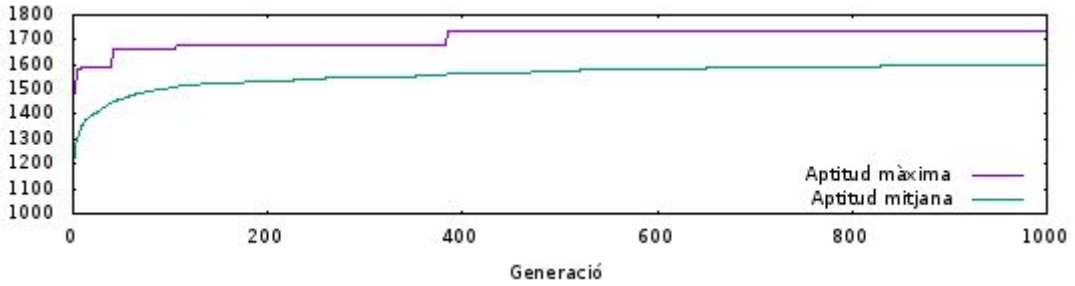
Experiment 020			
Població:	100	**Cromosoma:**	9bit1decimal
Generacions:	1.000	**Prob. aparellament:**	90%
Millor aptitud:	1.781,4	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__Independent__9bit1decimal_generations=1000_population=100_games=20_crossover=0.90_mutation=0.05_1495610045.csv		



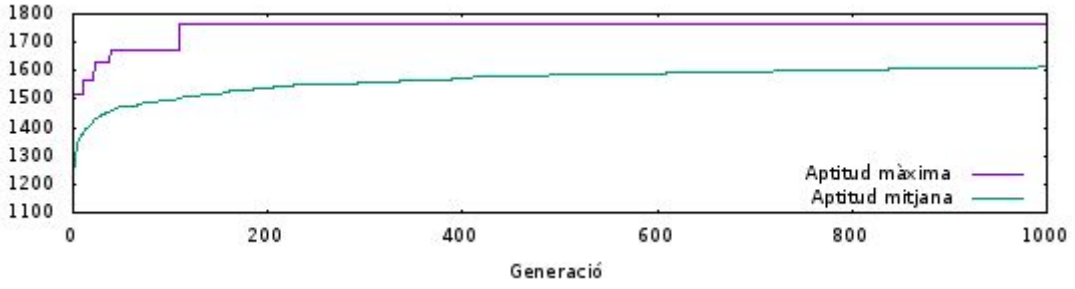
Experiment 021			
Població:	100	**Cromosoma:**	9bit1decimal
Generacions:	1.000	**Prob. aparellament:**	80%
Millor aptitud:	1.608,2	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__Independent__9bit1decimal_generations=1000_population=100_games=20_crossover=0.80_mutation=0.05_1495610049.csv		



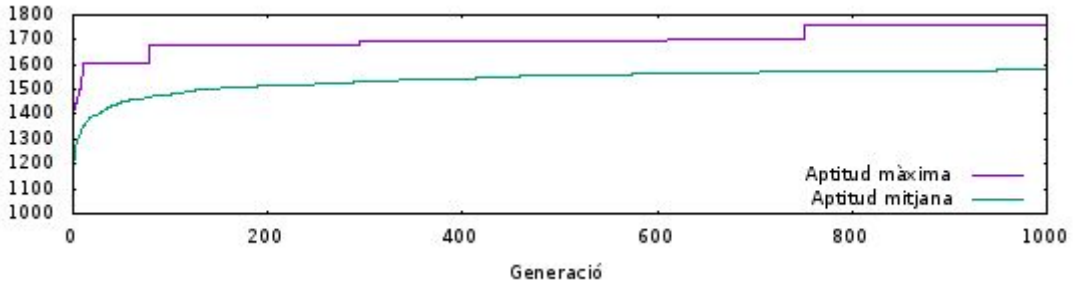
Experiment 022			
Població:	100	**Cromosoma:**	9bit1decimal
Generacions:	1.000	**Prob. aparellament:**	65%
Millor aptitud:	1.737	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__Independent__9bit1decimal_generations=1000_population=100_games=20_crossover=0.65_mutation=0.05_1495610056.csv		



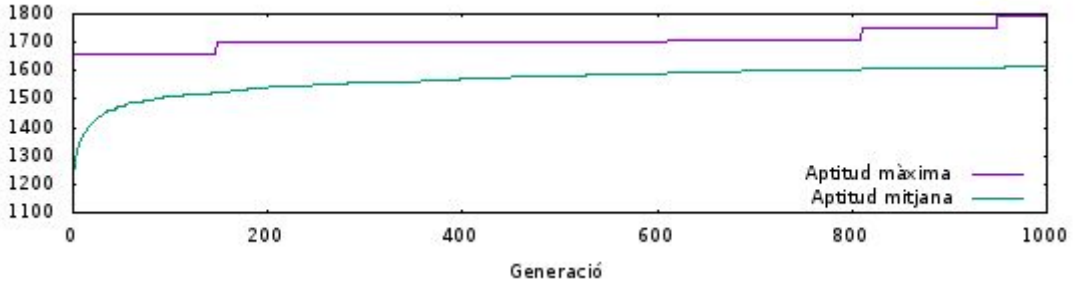
Experiment 023			
Població:	100	**Cromosoma:**	9bit1decimal
Generacions:	1.000	**Prob. aparellament:**	50%
Millor aptitud:	1.762	**Prob. mutació:**	5%
Fitxer:	experiment_chromosome=Chromosome__Independent__9bit1decimal_generations=1000_population=100_games=20_crossover=0.50_mutation=0.05_1495610066.csv		



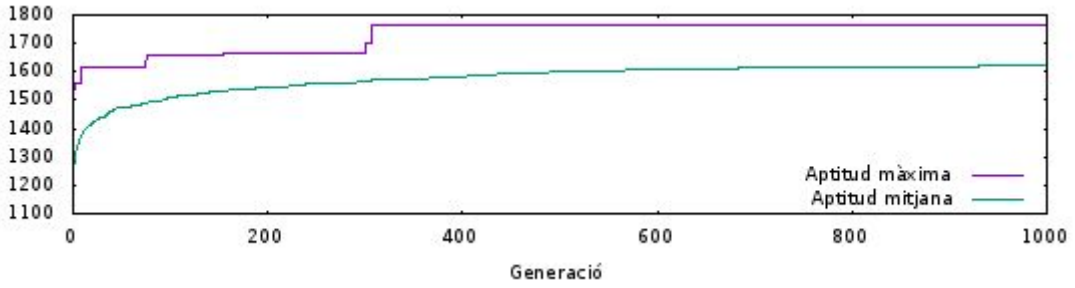
Experiment 024			
Població:	100	**Cromosoma:**	9bit1decimal
Generacions:	1.000	**Prob. aparellament:**	95%
Millor aptitud:	1.762,8	**Prob. mutació:**	1%
Fitxer:	experiment_chromosome=Chromosome__Independent__9bit1decimal_generations=1000_population=100_games=20_crossover=0.95_mutation=0.01_1495692876.csv		



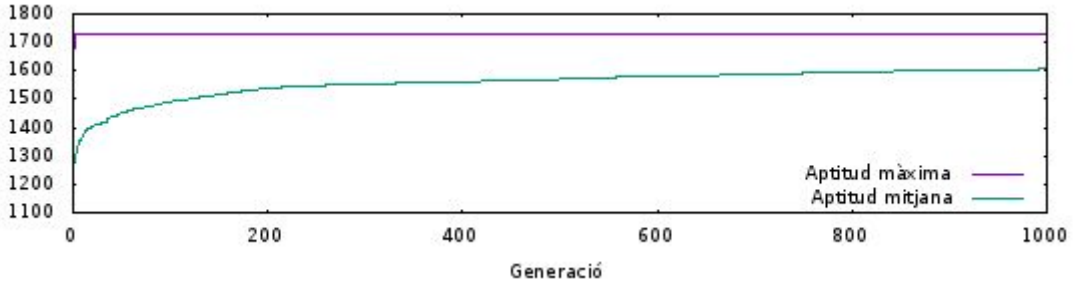
Experiment 025			
Població:	100	**Cromosoma:**	9bit1decimal
Generacions:	1.000	**Prob. aparellament:**	95%
Millor aptitud:	1.796	**Prob. mutació:**	10%
Fitxer:	experiment_chromosome=Chromosome__Independent__9bit1decimal_generations=1000_population=100_games=20_crossover=0.95_mutation=0.10_1495692861.csv		



Experiment 026			
Població:	100	**Cromosoma:**	9bit1decimal
Generacions:	1.000	**Prob. aparellament:**	95%
Millor aptitud:	1.762,2	**Prob. mutació:**	25%
Fitxer:	experiment_chromosome=Chromosome__Independent__9bit1decimal_generations=1000_population=100_games=20_crossover=0.95_mutation=0.25_1495692859.csv		



Experiment 027			
Població:	100	**Cromosoma:**	9bit1decimal
Generacions:	1.000	**Prob. aparellament:**	95%
Millor aptitud:	1.730,8	**Prob. mutació:**	50%
Fitxer:	experiment_chromosome=Chromosome__Independent__9bit1decimal_generations=1000_population=100_games=20_crossover=0.95_mutation=0.50_1495692835.csv		



Experiment 028			
Població:	200	**Cromosoma:**	9bit1decimal
Generacions:	8.000	**Prob. aparellament:**	80%
Millor aptitud:	1.650	**Prob. mutació:**	10%
Fitxer:	experiment_chromosome=Chromosome__Independent__9bit1decimal_generations=8000_population=200_games=20_crossover=0.8_mutation=0.1_1495879886.csv