UAB
Universitat Autònoma
de Barcelona

Facultat de Ciències

Grau en Matemàtiques

Treball de Fi de Grau

Ajust d'Equacions Diferencials Ordinàries a Dades de Sèries Temporals Curtes

---

Gloria Tarragona Ruiz

Tutor: Lluís Alsedà Soler

Any acadèmic: 2024/25

Convocatòria: Juny

Resum

La modelització a partir d'equacions diferencials ordinàries és una tècnica àmpliament utilitzada en camps com la biologia, la física o la química. Per ajustar els paràmetres d'aquests models a dades empíriques sovint es recorre a tècniques d'optimització. En aquest treball es presenta una metodologia per ajustar models diferencials a sèries temporals curtes mitjançant mètodes heurístics: *Simulated Annealing* i Algorismes Genètics, tant en versió discreta com contínua. Per tal de validar el rendiment d'aquests mètodes en un cas real, s'estudia un model existent de la dinàmica poblacional de les gavines al Delta de l'Ebre, utilitzant onze observacions corresponents al període 2006–2016. Els resultats mostren que el *Simulated Annealing* i l'algorisme genètic continu convergeixen a solucions òptimes semblants a les reportades a la literatura, mentre que l'algorisme genètic discret presenta un rendiment inferior. Aquest estudi posa de manifest la utilitat dels mètodes heurístics per a l'ajust de models diferencials en contextos en què les dades disponibles són limitades.

Índex

1 Introducció

La modelització d'un sistema dinàmic mitjançant equacions diferencials ordinàries és una eina fonamental en un ampli ventall de disciplines com la biologia, la física, la química, l'economia o l'enginyeria. Aquests models permeten descriure l'evolució temporal d'un sistema en funció d'un conjunt de paràmetres. Per a l'estimació d'aquests paràmetres, s'utilitzen sovint tècniques d'ajust de models diferencials a dades obtingudes experimentalment, que consisteixen a optimitzar els valors dels paràmetres per minimitzar una funció d'error entre el model i les dades.

Els mètodes clàssics d'optimització, com els mètodes basats en gradient, requereixen certes hipòtesis sobre la funció objectiu com per exemple que sigui suau, diferenciable i amb pocs òptims locals. No obstant això, en moltes aplicacions pràctiques, es disposa només d'un nombre reduït d'observacions a causa de limitacions econòmiques, tècniques o logístiques en la recollida de dades. Aquesta manca d'informació, pot provocar que múltiples conjunts de paràmetres proporcionin ajusts similars, dificultant el procés d'optimització amb estratègies tradicionals.

Davant aquesta problemàtica, aquest treball proposa l'ús de mètodes heurístics, en particular els algorismes genètics i l'algorisme *Simulated Annealing*. Aquests mètodes permeten explorar eficaçment espais de cerca complexos i evitar l'estancament en òptims locals gràcies a la utilització de tècniques estocàstiques. Els algorismes genètics actuen sobre una població inicial aleatòria aplicant operadors probabilístics de selecció, recombinació i mutació per explorar l'espai de cerca i fer evolucionar la població cap a les regions de baix cost. D'altra banda, el *Simulated Annealing* introdueix un procés d'acceptació probabilístic de solucions no òptimes en les fases inicials que permet evitar la convergència a òptims locals.

En aquest context, es plantegen dos objectius principals: El primer és definir un procediment per a l'ajust de models diferencials a sèries temporals curtes mitjançant optimització heurística. En aquest sentit, s'han estudiat els aspectes pràctics i fonaments teòrics dels dos mètodes heurístics proposats. D'altra banda, el segon objectiu és posar a prova aquesta metodologia en un cas d'estudi real per validar-la empíricament i identificar les condicions que afavoreixen una millor convergència. Per fer-ho s'utilitza un model de la dinàmica de la població de gavines del Delta de l'Ebre i una sèrie temporal del nombre d'individus en la població al llarg dels anys 2006-2016.

Per assolir aquests objectius s'han realitzat diverses contribucions. En primer lloc, s'han desenvolupat implementacions pròpies, en llenguatge `C`, de tres mètodes d'optimització heurística: Algorismes Genètics, tant en versió discreta com versió contínua i el *Simulated Annealing*. En segon lloc, s'ha aplicat la metodologia al cas esmentat i s'ha dut a terme una anàlisi comparativa dels resultats obtinguts per a cada algorisme. Finalment, s'han efectuat diverses proves amb diferents configuracions d'hiperparàmetres dels algorismes per a observar com afecta la seva elecció a la convergència i estabilitat dels resultats.

El treball s'estructura en set seccions. A la secció 2 es presenta el mètode d'ajust de models diferencials a dades experimentals. A la secció 3 s'introdueixen els mètodes heurístics i perquè són útils per aquest tipus de problemes. A les seccions 4 i 5 es detallen els elements, l'estructura i el funcionament dels Algorismes Genètics i el *Simulated Annealing*. La secció 6 es dedica a l'aplicació pràctica del mètode al cas de la població de gavines del Delta de l'Ebre, i a l'anàlisi dels resultats obtinguts. Finalment, a la secció 7 es presenten les conclusions generals i es proposen futures línies de treball.

Per acabar, cal destacar que aquest treball no només proposa una estratègia eficient per a l'ajust de models diferencials en casos amb poques dades, sinó que també aporta eines versàtils i adaptables a problemes d'optimització complexos en contextos més generals. En aquest sentit, la contribució del treball és rellevant tant en l'àmbit de la modelització matemàtica com en el de l'optimització.

2 Metodologia per a l'ajust de models diferencials a dades

En aquesta secció es presenta una metodologia per a l'ajust de models diferencials a partir de dades experimentals, proposada per D. Brewer *et al.* [Bre+08, p.519-522].

Considerem un model definit mitjançant una equació diferencial ordinària de la forma:

$$\frac{dx(t)}{dt} = f(x(t), t, \alpha)$$

on $x(t)$ és el vector de variables que evolucionen al llarg del temps, f representa un camp de vectors i α és un conjunt de paràmetres.

A priori, els valors dels paràmetres són desconeguts, excepte en alguns casos on poden ser mesurats empíricament o deduïts a partir de principis fonamentals. En general, per estimar els paràmetres, és necessari utilitzar un mètode d'ajust del model diferencial a partir d'un conjunt de dades experimentals $D := \{\pi(t_i) : i = 1, \ldots, n\}$.

Aquests mètodes es defineixen a partir de dos elements: una funció d'error $F_D(\alpha)$ que quantifica la diferència entre el model amb paràmetres α i les dades D i un mètode d'optimització que cerca el valor $\hat{\alpha}$ que minimitza la funció $F_D(\alpha)$. A partir d'aquests components, la metodologia per a l'estimació de paràmetres es construeix de la següent manera:

1. Per poder determinar com de bé s'ajusta el model a les dades, en primer lloc, és necessari obtenir les solucions de l'equació diferencial $x(t_i)$ per $i = 1, ..., n$. Considerem els següents problemes de valor inicial:

$$\begin{cases} \dfrac{dx(t)}{dt} = f(x(t), t, \alpha), \quad t \in [t_0, t_i] \quad \text{per } i \in \{1, ..., n\} \\ \\ x(t_0) = x_0 \end{cases}$$

   Implementant un mètode de resolució numèrica obtenim les solucions aproximades que denotem com $u(t_i)$. Notem que el valor x_0 pot ser desconegut o es pot partir d'aproximacions poc precises, aleshores és convenient interpretar la condició inicial com un paràmetre afegit i treballar amb una funció d'error $F_D(\alpha, x_0)$ que dependrà del conjunt de paràmetres α i la condició inicial x_0.

2. El següent pas és la selecció d'una funció d'error $F_D(\alpha, x_0)$. Notem que l'error serà petit quan els punts $\pi(t_i)$ i $x(t_i)$ estiguin a prop, i conseqüentment l'error serà gran quan estiguin lluny. Aleshores, la manera més natural de quantificar l'error és amb la distància entre la solució de l'equació diferencial i les dades en els temps t_i, per exemple, usant la norma L^2.

$$F_D(\alpha, x(t_0)) = \sqrt{\sum_{i=1}^{n} ||x(t_i) - \pi(t_i)||^2}$$

3. Finalment, sigui $\mathscr{P}$ l'espai de definició dels paràmetres x_0 i α, se selecciona un mètode d'optimització per a resoldre el següent problema de minimització:

$$\begin{aligned} &\min \ F_D(x_0, \alpha) \\ &\text{subjecte a } (x_0, \alpha) \in \mathscr{P} \end{aligned}$$

La tria del mètode dependrà del paisatge de la funció $F_D(\alpha, x_0)$ a l'espai $\mathscr{P}$. Donada una funció d'error que no tingui gaires mínims locals es poden usar algorismes de minimització senzills com el mètode del descens del gradient. D'altra banda, en el cas de paisatges més complexos es necessita l'ús de mètodes heurístics.

3 Optimització heurística: balanç entre exploració i explotació

Segons Martí et al. [MR11, p.17], els mètodes heurístics són algorismes que cerquen solucions aproximades a problemes d'optimització. Davant de problemes complexos, el temps necessari per aplicar un mètode exacte, suposant que existeix un, és més elevat que en el cas dels mètodes heurístics, tant que fins i tot pot ser que no sigui possible la seva implementació. Això fa que moltes vegades la resolució de problemes del món real s'hagi de dur a terme a partir d'aquests mètodes.

Els mètodes heurístics cerquen en grans espais de solucions a partir de la combinació de dues estratègies: l'explotació i l'exploració. El terme explotació fa referència a la cerca de noves solucions al voltant del millor punt trobat durant el procés d'optimització amb l'objectiu de localitzar ràpidament l'òptim. D'altra banda, l'exploració consisteix en la cerca de noves regions desconegudes per a reduir la incertesa del problema i fer créixer el coneixement sobre les opcions disponibles.

Dit d'una altra manera, segons Li *et al.* [LSZ24, p.161-162], l'explotació se centra a accelerar el procés d'optimització i l'exploració en evitar la convergència cap a òptims locals. Tal com expliquen Crepinšek *et al.* [ČLM13, p.1] l'equilibri entre aquestes dues estratègies és essencial per a garantir l'eficiència de l'algorisme. Notem la importància d'aquest balanç a partir d'un exemple. Considerem la funció de Rastrigin en una variable:

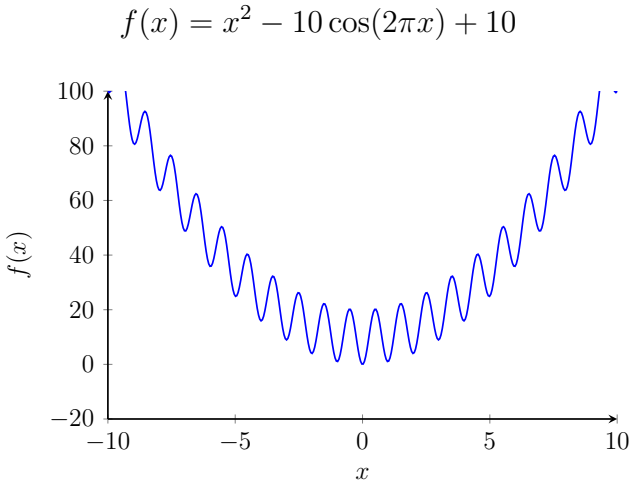


Figura 1: Funció Rastrigin unidimensional

Aquesta funció té un mínim global a $x = 0$, però una gran quantitat d'òptims locals al voltant d'aquest punt. Si un algorisme dedica massa temps a explorar aquests mínims locals sense aprofitar les bones solucions trobades, es generarà una gran despesa computacional divagant per l'espai sense obtenir el mínim global. D'altra banda, si l'algorisme només explota les millors solucions trobades, pot quedar-se atrapat en un d'aquests mínims locals i no convergir a la solució òptima.

Aquest exemple mostra com una gestió inadequada de les estratègies d'exploració i explotació pot afectar el rendiment de l'algorisme. Per aquest motiu, els mètodes heurístics incorporen mecanismes que regulen aquest equilibri al llarg del procés d'optimització. A continuació es presenten dos mètodes heurístics: els Algorismes Genètics i l'algorisme *Simulated Annealing*.

4 Algorismes Genètics

4.1 Introducció

A la natura, els individus d'una població competeixen entre ells pels recursos disponibles. Aquesta competència fa que els individus amb millors capacitats adaptatives tinguin més probabilitats de sobreviure i reproduir-se, transmetent així aquestes característiques favorables a les generacions futures. Com a resultat, després de diverses generacions, la població es va adaptant progressivament a l'entorn. Aquesta és la idea fonamental de la teoria evolutiva per selecció natural de Charles Darwin.

Tal com exposa Mitchell [Mit98, p.2-3], John Holland va estudiar formalment el fenomen d'adaptació en la natura i va desenvolupar maneres d'importar els mecanismes de l'adaptació natural als sistemes de computació. Aplicant aquests principis, Holland va desenvolupar els algorismes genètics (GA), que presenta a la seva obra *Adaptation in Natural and Artificial Systems* [Hol75] com una eina per resoldre problemes difícils per la seva complexitat o incertesa.

El desenvolupament dels algorismes genètics és anàleg al procediment evolutiu en la natura. Essencialment, aquests algorismes transformen una població d'individus en una de nova a partir d'un procés de selecció i d'operadors genètics com la recombinació i la mutació. Per entendre el seu funcionament, cal primer introduir els conceptes biològics i els seus equivalents en l'àmbit computacional.

Els organismes vius estan formats per cèl · lules on es guarda la seva informació genètica, concretament en estructures anomenades cromosomes. Aquests es poden dividir en segments més petits anomenats gens, que cadascun codifica una característica concreta de l'individu. Els valors possibles que pot prendre un gen s'anomenen al · lels. Per exemple, el gen del color dels ulls té com a possibles al · lels blau, verd o marró. Cada gen ocupa una posició específica dins del cromosoma anomenada locus.

El conjunt de tots els gens d'un individu s'anomena genotip. D'altra banda, el conjunt de totes les característiques observables d'un individu, per exemple el color, la forma, la mida, etc, s'anomena fenotip. És important destacar que en l'evolució la selecció natural actua sobre el fenotip, ja que selecciona els individus amb millors característiques. D'altra banda, la reproducció, que consisteix en la recombinació de material genètic i les mutacions produïdes per errors de còpia dels cromosomes dels progenitors, actua sobre el genotip.

La següent taula descriu com s'adapta aquesta terminologia biològica a la notació dels algorismes genètics. Diversos autors han presentat aquesta correspondència de manera detallada, com és el cas de Sivanandam *et al.* [Siv+08, p. 16–20] i Mitchell [Mit98, p. 5–6].

Concepte biològic	**Algorismes genètics**
Cromosoma	Cadena completa (individu)
Gen	Segment de cadena que codifica un paràmetre
Al · lel	Valor concret d'una posició dins la cadena
Locus	Posició dins la cadena
Genotip	Cadena de tots els gens (solució codificada)
Fenotip	Configuració de paràmetres (solució descodificada)

Taula 1: Equivalència entre conceptes biològics i computacionals d'un GA[1].

4.2 Elements, operadors i estructura dels algorismes genètics

En aquesta secció s'introdueixen els elements i operadors d'un algorisme genètic per a la resolució d'un problema d'optimització de la següent forma:

$$\begin{aligned} &\text{Sigui } f : X \to \mathbb{R} \\ &\text{trobar } x_0 \in X = X_1 \times ... \times X_m \subseteq \mathbb{R}^m, \text{ tal que } f(x_0) = \max_{x \in X} f(x) \end{aligned} \tag{4.1}$$

on anomenem *funció objectiu* a f i *espai de cerca* a X

4.2.1 Individus

Un *individu* és una solució candidata a resoldre un problema del tipus 4.1. Cada individu es pot descriure a partir de dues representacions: el fenotip i el genotip. El *fenotip* és el valor $x \in X$ de la solució en l'espai de cerca, mentre que el *genotip* $s \in S = S_1 \times \dots S_n$ descriu aquesta solució en forma de cadena codificada dins *l'espai de codificació* S. El genotip està format per gens. Un *gen* $s_i \in S_i$ és la codificació d'un paràmetre concret $x_i \in X_i$ de la solució. Per tal de relacionar aquests dos elements definim la següent funció que associa cada genotip amb el seu fenotip.

Definició 4.1. Siguin $S_1, ..., S_m$ conjunts i X l'espai de cerca, la següent funció

$$\begin{array}{rccc} m: & S = S_1 \times \dots \times S_m & \longrightarrow & X_1 \times \dots \times X_m \\ & (s_1, \dots, s_m) & \longmapsto & (x_1, ..., x_m) \end{array}$$

s'anomena *funció morfogènica*.

[1]Al llarg d'aquest treball, prendrem cromosoma i genotip com a termes equivalents, tot i que en implementacions més complexes poden no coincidir.

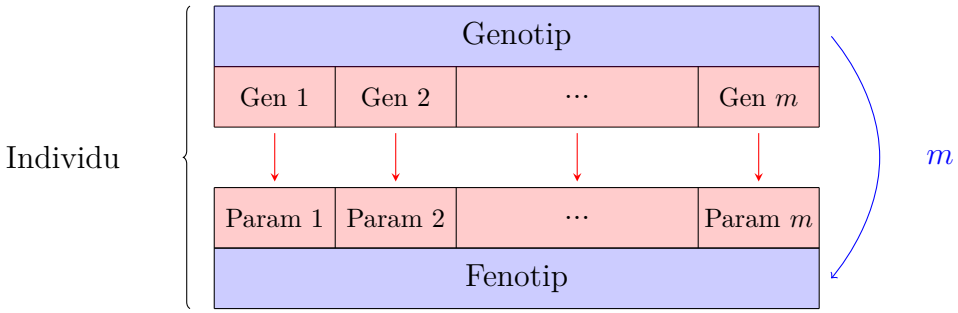


Figura 2: Representació d'un individu i relació entre genotip i fenotip. Inspirada en Sivanandam *et al.* [Siv+08, p. 40].

Observació. Segons Sivanandam *et al.* [Siv+08, p. 40], en general, no és necessari que la funció morfogènica sigui bijectiva. Com a mínim necessitem que sigui exhaustiva, és a dir que cada possible solució $x \in X$ tingui un genotip que la codifiqui. Això permet garantir l'exploració de tot l'espai. La injectivitat no és una condició indispensable, tot i que és útil, ja que el fet que moltes codificacions representin una mateixa solució pot provocar que l'algorisme centri la cerca més en unes regions que d'altres.

Per poder seleccionar una funció morfogènica, és necessari, en primer lloc, seleccionar la representació dels gens, és a dir seleccionar els conjunts $S_1, \ldots, S_m$ que definiran l'espai de codificació S. Aquest procés s'anomena *codificació*. Presentem dos mètodes per dur a terme aquesta tasca: la codificació binària i la codificació real.

Codificació binària

Aquest és el mètode de l'algorisme genètic clàssic de Holland. Consisteix a utilitzar cadenes de bits per a la representació dels gens. Sigui l_i la longitud del gen i per $i = 1, \ldots, m$ tenim que $S = \prod_{i=1}^{m} \{0,1\}^{l_i}$. La longitud dels gens se selecciona en funció de la precisió necessària per a cada paràmetre.

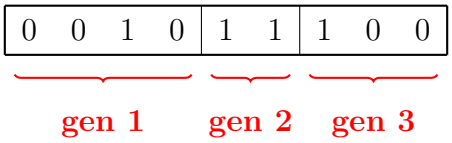


Figura 3: Exemple cromosoma en codificació binària

Veiem alguns exemples de com s'utilitza aquesta codificació.

Exemple 4.2. Siguin $a_i, b_i \in \mathbb{Z}$ amb $a_i < b_i$ suposem que es vol codificar un paràmetre x_i amb un espai de cerca $X_i := \{x_i \in \mathbb{Z} | a_i \leq x_i \leq b_i\}$. Com que és un conjunt finit de nombres enters es pot descriure de manera exacta en representació binària.

Sigui una cadena de bits de longitud l el nombre de valors N que es poden representar en binari satisfà la següent desigualtat:

$$2^l \geq N \tag{4.2}$$

Tenim que $|X_i| = (b_i - a_i) + 1$, aleshores de l'equació 4.2 es dedueix que per poder representar el conjunt X_i cal seleccionar com a longitud el valor
$l_i = \lceil \log_2(b_i - a_i + 1) \rceil$

Exemple 4.3. Siguin $a_j, b_j \in \mathbb{R}$ amb $a_j < b_j$ suposem ara que es vol codificar un paràmetre x_j amb un espai de cerca $X_j = [a_j, b_j]$. En aquest cas tenim un espai continu. Per poder implementar una codificació binària necessitem reduir-nos a un conjunt finit de nombres reals, és a dir dur a terme una discretització de l'espai.

Per obtenir una precisió p_j necessitem treballar amb un conjunt amb $\lceil \frac{b_j - a_j}{p_j} \rceil$ solucions candidates equidistants. De manera anàloga a l'exemple anterior es dedueix que per representar en codificació binària aquest nombre de valors és necessari que $l_j = \lceil \log_2(\lceil \frac{b_j - a_j}{p_j} \rceil) \rceil$.

Un cop determinada la longitud, considerem la següent funció:

$$\begin{aligned} s_j : \{0, \ldots, 2^{l_j} - 1\} &\longrightarrow & X_j \\ u_j &\longmapsto & a_j + u_j \frac{b_j - a_j}{2^{l_j} - 1} \end{aligned}$$

La imatge $X'_j := s_j(\{0, \ldots, 2^{l_j} - 1\})$ serà l'espai de cerca discretitzat per al paràmetre x_j. Suposem ara que $X = [a_1, b_1] \times \ldots \times [a_m, b_m] \subseteq \mathbb{R}^m$. Considerant la funció de discretització anterior per cada paràmetre podem definir la següent funció que discretitza l'espai X:

$$\begin{aligned} s : \prod_{j=1}^{m} \{0, ..., 2^{l_j} - 1\} &\longrightarrow & X \\ (u_1, ..., u_m) &\longmapsto & (s_1(u_1), ..., s_m(u_m)) \end{aligned}$$

Aleshores el nou espai de cerca serà $X' := s(X)$. A partir d'aquesta funció podem construir la funció morfogènica considerant la composició

$$m = s \circ b : \quad \prod_{j=1}^{m} \{0,1\}^{l_j} \to X'$$

on b és la bijecció que transforma el valor binari de cada component a un enter a partir de la suma de potències de 2.

Codificació Real

Aquesta codificació consisteix en representar els gens com a nombres reals. Com que $X \subseteq \mathbb{R}^m$ notem que la representació del fenotip i el genotip és la mateixa, per tant, $S = X$ i la funció morfogènica és la identitat.

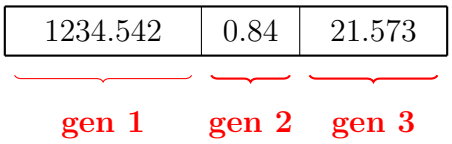


Figura 4: Exemple codificació real

En funció del tipus d'espai de codificació S, distingim dos tipus d'algorismes genètics:

- **Algorismes Genètics Discrets**: Operen sobre un espai de codificació discret, com és el cas de la codificació binària.

- **Algorismes Genètics Continus**: En la teoria, operen sobre un espai de codificació continu, com és el cas de la codificació real. No obstant això, en la pràctica sempre es treballa amb una discretització induïda per la precisió finita de la màquina.

4.2.2 Poblacions

Els algorismes genètics són algorismes basats en població, és a dir, en lloc d'operar sobre una única solució, ho fan sobre una població. Una *població* és una mostra d'individus de mida $n = 2k$ per algun $k \in \mathbb{N}$ fixat. El valor n s'anomena mida de la població. Segons Sivanandam *et al.* [Siv+08, p.41-42], els dos aspectes més importants d'aquests objectes són els següents:

- **Selecció de la mida de la població:** La selecció de l'hiperparàmetre n dependrà de la complexitat del problema. Notem que com més gran sigui la mida de la població més diversitat d'individus es tindrà, això afavoreix a l'exploració de l'espai de cerca. D'altra banda, un valor de n massa gran augmenta significativament el cost computacional, cosa que pot alentir la convergència.

- **Inicialització de la població:** La població inicial en els algorismes genètics consisteix a generar una mostra aleatòria de n individus uniformement distribuïda. La uniformitat en la mostra és essencial per garantir l'exploració de tot l'espai de cerca, si no l'algorisme podria explorar només una petita part i mai trobar l'òptim global.

Un cop determinada la població inicial $P(0)$ de mida n, els algorismes genètics a cada generació (i.e iteració) t modifiquen una població $P(t)$ obtenint una de nova $P(t+1)$. El següent diagrama descriu com l'algorisme opera sobre una població durant una generació.

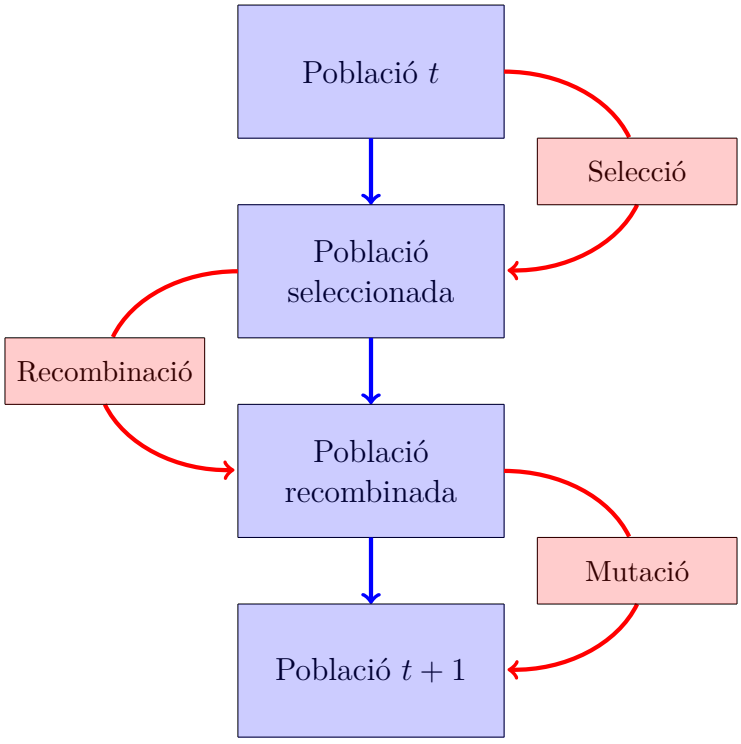


Figura 5: Esquema d'una generació d'un GA

Aleshores una generació d'un algorisme genètic es construeix a partir de 3 operadors: selecció dels millors individus, recombinació del material genètic i mutació dels nous cromosomes.

4.2.3 Selecció

La *selecció* és un operador genètic que selecciona els individus amb millors característiques per a reproduir-se. Per dur a terme el procés de selecció és necessari donar una noció de la qualitat dels individus de la població.

Definició 4.4. Donada una funció morfogènica m i un individu amb genotip $s \in S$ i fenotip $m(s)$ s'anomena *fitness* al valor $f(m(s))$.

Notem que en el context de maximitzar, com més gran sigui el fitness, millor serà l'individu. Aleshores, el procés de selecció consisteix a seleccionar n individus d'una població P, evidentment amb reemplaçament, en funció del seu fitness per a reproduir-se. Presentem dos mètodes per dur a terme aquesta tasca: la selecció proporcional al fitness i la selecció per torneig.

Selecció proporcional al fitness

És el mètode de l'algorisme genètic clàssic de Holland. Aquest es basa a convertir el fitness en una distribució de probabilitat i seleccionar els individus generant una mostra d'aquesta distribució. Denotem F_i al fitness de l'individu i d'una població P. Per determinar la distribució es considera l'interval $[0, r]$ on $r = \sum_{i=1}^{n} F_i$ i es divideix en subintervals de longitud F_i. Aleshores, per seleccionar els n individus es genera una mostra aleatòria $C = \{c_1, \ldots, c_n\}$ amb $C \sim \text{Unif}(0, r)$. Cada individu serà seleccionat tantes vegades com valors mostrals hagin caigut en el seu interval.

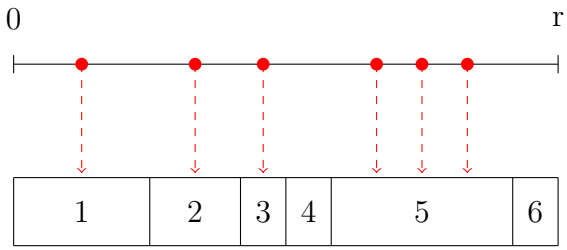


Figura 6: Exemple de selecció proporcional al fitness amb una població de 6 individus. Inspirada en [Luk09, p.41]

A partir del següent exemple de Luke [Luk09, p.42] veiem que aquest mètode presenta un problema. Suposem que tenim una funció objectiu $f(x) \in [0, 1]$ $\forall x \in X$. A la fase final tots els individus tindran fitness propers a 1, per exemple 0.97, 0.98 o 0.99. En aquesta situació ens interessa seleccionar els individus amb fitness 0.99, però per la construcció de la selecció proporcional al fitness tots els individus tindran gairebé la mateixa probabilitat. Aleshores pràcticament s'està realitzant una selecció aleatòria. Com a solució a aquest problema sorgeix la tècnica de selecció més utilitzada avui en dia en el camp dels algorismes genètics: la selecció per torneig.

Selecció per torneig

Aquesta selecció consisteix a triar l'individu amb un fitness més alt d'entre t individus seleccionats aleatòriament [2]. Segons Luke [Luk09, p. 43], els avantatges principals d'aquest mètode són:

- No es basa en els valors absoluts del fitness, si no en comparacions entre els individus. Això el converteix en un mètode robust davant les possibles particularitats de la funció fitness.

- És fàcil d'implementar, ja que no requereix càlculs addicionals com la construcció de distribucions de probabilitat.

- L'hiperparàmetre de la mida de torneig t, permet ajustar com d'exploratori o explotador es comporta l'algorisme. Notem que la probabilitat de seleccionar el millor individu de la població és:

$$\mathbb{P}(\text{seleccionar millor individu}) = 1 - \left(1 - \frac{1}{n}\right)^t \tag{4.3}$$

  A partir de l'expressió 4.3, observem que aquesta probabilitat creix a mesura que augmenta el valor de t. Per tant, valors petits afavoreixen a l'exploració, ja que permeten una major diversitat en la població. D'altra banda, quan $t \to \infty$ la probabilitat que aparegui l'individu amb millor fitness tendeix a 1, per tant, els valors grans de t afavoreixen a l'explotació.

4.2.4 Recombinació

La *recombinació* és un operador que a partir de dos cromosomes progenitors, recombina el seu material genètic, obtenint dos cromosomes fills. Aquest operador actua amb una probabilitat p_c anomenada *probabilitat de recombinació*. En el cas que no hi hagi recombinació, els cromosomes fills seran simplement una còpia dels progenitors.

Notem que la recombinació actua sobre el genotip de l'individu, per tant, els mètodes varien en funció de la codificació seleccionada. Donat un cromosoma amb codificació binària de longitud l destaquen tres mètodes: recombinació d'un punt, recombinació de dos punts i recombinació uniforme.

[2] Aquest mètode es pot adaptar fàcilment a problemes de minimització triant l'individu amb fitness més baix en lloc del més alt

Per a presentar aquests mètodes denotarem als dos cromosomes dels progenitors com $u^a = (u_1^a, \ldots, u_l^a)$, $u^b = (u_1^b, \ldots, u_l^b) \in \{0,1\}^l$ i denotem als cromosomes fills que es generen a partir de la recombinació com $v^a = (v_1^a, \ldots, v_l^a)$, $v^b = (v_1^b, \ldots, v_l^b) \in \{0,1\}^l$

Recombinació d'un punt

Aquest és el mètode de l'algorisme genètic clàssic de Holland. La recombinació d'un punt consisteix a generar un nombre aleatori $c \in \{1, \ldots, l-1\}$ que determinarà el punt de tall. A partir d'aquest valor, la recombinació es produeix intercanviant els valors u_i^a i u_i^b tals que $c < i$ per $i = 1, \ldots, l$, és a dir, els cromosomes fills es generen a partir de les següents expressions:

$$v_i^a = \begin{cases} u_i^a & \text{si } i \leq c \\ u_i^b & \text{si } i > c \end{cases} \qquad v_i^b = \begin{cases} u_i^b & \text{si } i \leq c \\ u_i^a & \text{si } i > c \end{cases}$$

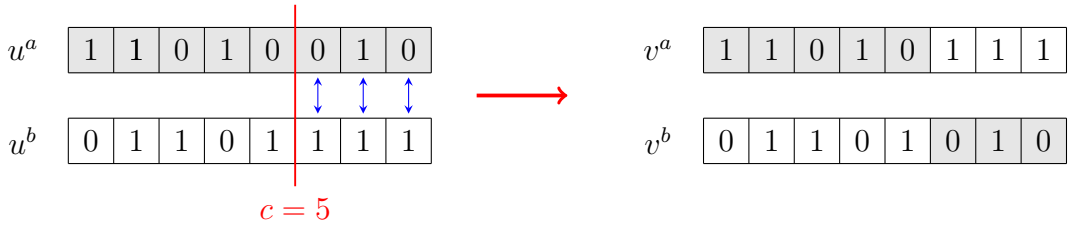


Figura 7: Exemple recombinació d'un punt amb cromosomes de longitud $l = 8$ i punt de tall $c = 5$

Recombinació de dos punts

Aquest mètode selecciona aleatòriament dos punts de tall $c, d \in \{1, \ldots, l-1\}$. Suposem que $c < d$. A partir d'aquest valor, la recombinació es produeix intercanviant els valors u_i^a i u_i^b tals que $c < i \leq d$ per $i = 1, \ldots, l$, és a dir els cromosomes fills es generen a partir de les següents expressions:

$$v_i^a = \begin{cases} u_i^b, & \text{si } c < i \leq d \\ u_i^a, & \text{altrament} \end{cases} \qquad v_i^b = \begin{cases} u_i^a, & \text{si } c < i \leq d \\ u_i^b, & \text{altrament} \end{cases}$$

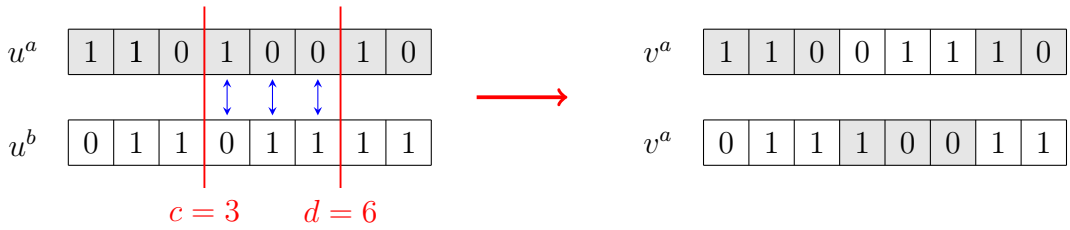


Figura 8: Exemple recombinació de dos punts amb cromosomes de longitud $l = 8$ i amb punts de tall $c = 3$ i $d = 6$

Recombinació uniforme

Aquest mètode consisteix a intercanviar els valors u_i^a i u_i^b amb una probabilitat p_u per $i = 1, \ldots, l$, és a dir els cromosomes fills es generen a partir de les següents expressions:

$$v_i^a = (1 - Z_i)u_i^a + Z_i u_i^b$$
$$v_i^b = (1 - Z_i)u_i^b + Z_i u_i^a$$

on $Z_i \sim \text{Bernoulli}(p_u)$

Figura 9: Exemple recombinació uniforme amb $p_u = 1/2$

Notem que en el cas de cromosomes representats en codificació real, les estratègies presentades anteriorment només reordenen els al · lels, sense aprofitar la naturalesa contínua de l'espai de codificació. Per aquest motiu, en aquest cas és més adequat usar mètodes de recombinació basats en la interpolació lineal dels valors dels progenitors.

Per a la presentació dels següents mètodes suposarem que $S = \prod_{i=1}^{m}[a_i, b_i]$ i denotarem $x^a = (x_1^a, \ldots, x_m^a)$, $x^b = (x_1^b, \ldots, x_m^b) \in S$ als dos cromosomes dels progenitors i denotarem com $y^a = (y_1^a, \ldots, y_m^a)$, $y^b = (y_1^b, \ldots, y_m^b) \in S$ als cromosomes fills que és generen a partir de la recombinació.

Recombinació lineal

Aquest mètode genera els dos cromosomes fills a partir de les combinacions convexes entre els al · lels x_i^a i x_i^b per $i = 1, \ldots, m$

$$y_i^a = \alpha x_i^a + (1 - \alpha) x_i^b$$
$$y_i^b = \beta x_i^a + (1 - \beta) x_i^b$$

on $\alpha, \beta \in [0, 1]$

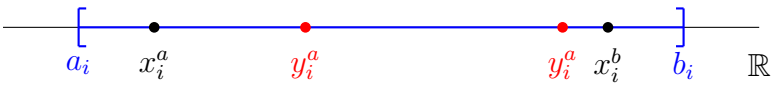


Figura 10: Recombinació lineal

Recombinació lineal estesa

Aquest mètode es construeix igual que la recombinació lineal, però amb la diferència que els valors dels fills poden sortir fora de l'interval definit pels pares. Això afavoreix a l'exploració de l'espai de cerca. Notem que aleshores es poden generar fills fora dels intervals de definició dels paràmetres $[a_i, b_i]$. En aquest cas, serà necessari forçar-los a estar dins el seu interval. Els dos fills es defineixen a partir de la següent expressió:

$$y_i^a = b(\alpha x_i^a + (1 - \alpha) x_i^b)$$
$$y_i^b = b(\beta x_i^a + (1 - \beta) x_i^b)$$

on $\alpha, \beta \in [-p_i, p_i]$ i $b(z) = \min(\max(z, a_i), b_i)$. Anomenem *factor d'extrapolació del paràmetre* x_i a l'hiperparàmetre p_i.

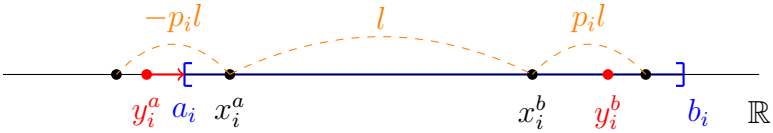


Figura 11: Recombinació lineal estesa

4.2.5 Mutació

La *mutació* és un operador que modifica aleatòriament el material genètic dels cromosomes fills amb l'objectiu de mantenir la diversitat genètica en la població. De la mateixa manera que la recombinació, la mutació també actua sobre el genotip dels individus i, per tant, la seva implementació dependrà de la codificació seleccionada.

Per a la codificació binària presentem el mètode de mutació *bit-flip*. Per a introduir aquest procediment denotem per $u = (u_1, ..., u_l) \in \{0,1\}^l$ a un cromosoma en codificació binària de longitud l abans de l'efecte de mutació i denotem per $u' = (u'_1, \ldots, u'_l) \in \{0,1\}^l$ al cromosoma després d'aplicar l'operador mutació.

Mutació bit-flip

La mutació bit-flip consisteix a invertir [3] els bits u_i per $i = 1, \ldots, l$ amb una probabilitat p_m anomenada *probabilitat de mutació*. És a dir, el cromosoma u' s'obté a partir de la següent expressió:

$$u'_i = (1 - Z_i)u_i + Z_i(1 - u_i)$$

on $Z_i \sim \text{Bernoulli}(p_m)$

Per a la codificació real presentem el mètode de mutació uniforme. Per a introduir aquest procediment suposem que $S = \prod_{i=1}^{m}[a_i, b_i]$. Denotem per $x = (x_1, ..., x_m) \in S$ a un cromosoma en codificació real abans de l'efecte de mutació i denotem per $x' = (x'_1, \ldots, x'_m) \in S$ al cromosoma després d'aplicar l'operador mutació.

Mutació uniforme

La mutació uniforme consisteix a mutar els al $\cdot$ lels x_i per $i = 1, \ldots, m$ pertorbant uniformement el seu valor amb probabilitat de mutació p_m dins del seu interval de definició S_i. El cromosoma resultant x' es defineix a partir de la següent expressió:

$$x'_i = (1 - Z_i)\, x_i + Z_i \cdot b(x_i + \delta_i)$$

on $\delta_i \sim \text{Unif}(-\varepsilon_i, \varepsilon_i)$, $Z_i \sim \text{Bernoulli}(p_m)$ i $b(z) = \min(\max(z, a_i), b_i)$. Anomenem *rang de mutació del paràmetre* x_i a l'hiperparàmetre ε_i.

[3] Canviar els bits amb valor 1 per 0 i els bits amb valor 0 per 1

4.2.6 Criteris de convergència

La iteració descrita a la figura 5 s'aplica successivament sobre les poblacions fins que se satisfà una condició de parada i, per tant, es considera que l'algorisme ha convergit. Generalment, s'aplica algun dels següents criteris:

- **Nombre màxim d'iteracions:** L'algorisme para quan s'arriba a un nombre màxim d'iteracions especificat.

- **Manca de canvi en el fitness:** L'algorisme genètic para quan no hi ha canvi en el millor fitness durant un nombre especificat d'iteracions.

4.2.7 Estructura de l'algorisme

Un cop definits els seus elements i operadors es pot descriure l'estructura d'un algorisme genètic estàndard a partir del següent pseudocodi [4]

Algorithm 4.1 Algorisme Genètic (GA) (adaptat de [Luk09, p.35])

```
1:  procedure GA(N)                                   ▷ N és el tamany de la població
2:      P ← {}
3:      for N vegades do
4:          P ← P ∪ {nou individu aleatori}           ▷ Població inicial aleatòria
5:      end for
6:      Millor ← □
7:      repeat                                        ▷ Iteració sobre P(t)
8:          for P_i ∈ P do
9:              AvaluarFitness(P_i)
10:             if Millor = □ or Fitness(P_i) > Fitness(Millor) then
11:                 Millor ← P_i
12:             end if
13:         end for
14:         Q ← {}                                    ▷ Població P(t+1)
15:         for N/2 vegades do
16:             Pare P_a ←Seleccionar(P)
17:             Pare P_b ← Seleccionar(P)
18:             Fills F_a, F_b ← Recombinacio(Copia(P_a),Copia(P_b))
19:             Q ← P ∪ {Mutar(F_a), Mutar(F_b)}
20:         end for
21:         P ← Q                                     ▷ Actualitzar població
22:     until Se satisfaci una condició de parada
23:     return Millor
24: end procedure
```

[4] Aquest pseudocodi resol un problema de maximització. Per a problemes de minimització, caldria substituir el signe > per < a la línia 10.

4.3 Anàlisi teòrica

L'objectiu d'aquesta secció és descriure de manera teòrica el comportament dels algoritmes genètics. Aquest estudi presenta certes dificultats. Segons *U.BodenHofer* [Bod03, p.33-34], aquestes complicacions es deuen principalment al caràcter estocàstic, la complexitat de la seva estructura interna i la gran varietat d'operadors possibles d'aquests algorismes. A diferència dels mètodes deterministes, no sempre és possible garantir la convergència a l'òptim, sinó només descriure el comportament esperat.

En aquest context, a continuació, presentem diversos resultats teòrics que expliquen per què els algorismes genètics són efectius en molts problemes d'optimització. Per simplicitat ens restringirem a la versió clàssica dels algorismes genètics amb els seus elements corresponents: Poblacions de n cromosomes binaris de longitud l, selecció proporcional al fitness, recombinació d'un punt amb probabilitat p_c i mutació bit-flip amb probabilitat p_m.

4.3.1 Teorema dels esquemes

Definició 4.5. Un *esquema* H és una cadena de la forma $H = (h_1, ..., h_l) \in \{0, 1, *\}^l$. Les components $h_i \neq *$ s'anomenen *posicions fixades* i les components $h_i = *$ s'anomenen *comodins*.

Definició 4.6. *L'ordre d'un esquema* H, denotat per $o(H)$, es defineix com el seu nombre de posicions fixades.

$$o(H) = |\{i \in \{1, ..., l\} \mid h_i \neq *\}|$$

Definició 4.7. *La longitud d'un esquema* H, denotada per $\delta(H)$, es defineix com la distància entre la primera i l'última posició fixada de l'esquema.

$$\delta(H) = \max\{i \in \{1, ..., l\} \mid h_i \neq *\} - \min\{i \in \{1, ..., l\} \mid h_i \neq *\}$$

Observació. Sigui $E = \{0, 1\}^l$ l'espai de totes les cadenes de bits de longitud l, observem que un esquema H defineix un subconjunt $g(H) \subseteq E$ considerant la següent aplicació.

$$\begin{aligned} g: \quad \{0, 1, *\}^l &\longrightarrow \mathcal{P}(E) \\ H \quad &\longmapsto \quad \{x \in E \mid x_i = h_i \ \forall i \in \{1, ..., l\} \text{ tals que } h_i \neq *\} \end{aligned}$$

Definició 4.8. Sigui H un esquema s'anomena *instàncies de l'esquema* H a les cadenes $x \in g(H)$

Exemple 4.9. Les instàncies de l'esquema $H_1 = 1 * * * 1$ són totes les cadenes de 5 bits que comencen i acaben per 1. Tenim a més $o(H_1) = 2$ i $\delta(H_1) = 4$

Geomètricament, l'espai $E = \{0,1\}^l$ es pot interpretar com a hipercubs de dimensió l, identificant cada vèrtex amb una cadena.

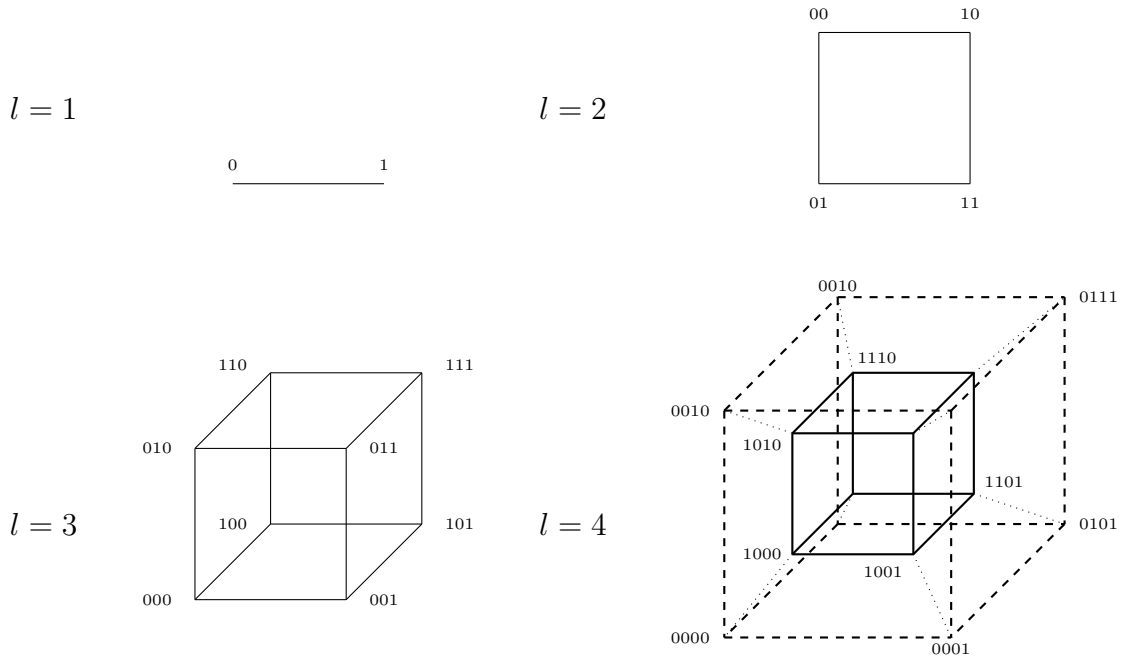


Figura 12: Identificació dels vèrtexs dels hipercubs de dimensió l amb el conjunt $\{0,1\}^l$ per $l = 1, 2, 3, 4$. Adaptació de la figura original de Bodenhofer [Bod03, p. 35].

També tenim que els esquemes H d'ordre $o(H) = k$ es poden interpretar com a hiperplans de dimensió $l - k$ d'aquest hipercub.

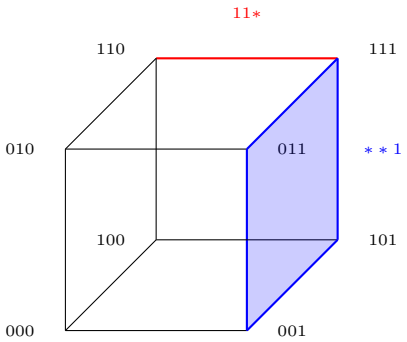


Figura 13: Exemple d'identificació d'esquemes d'ordre 1 i 2 com a hiperplans d'un hipercub de dimensió 3

Observació. En una codificació de cromosomes com a cadenes de l bits, el nombre total d'esquemes és 3^l, ja que cada posició pot tenir un 0, un 1 o un comodí *. Cada cadena de bits és instància de 2^l esquemes, corresponents a les combinacions possibles de mantenir les seves posicions fixades o substituir-les per un comodí. Aleshores en una població de cromosomes de mida n, es representen entre $2^l - n \cdot 2^l$ esquemes, en funció del grau de similitud entre els individus de la població.

A partir d'aquesta observació, Holland [Hol75, Capítol 4] va suggerir que, tot i que els algorismes genètics treballen explícitament amb una població de cromosomes, implícitament processen un nombre molt més gran d'esquemes simultàniament. Aquest fenomen és conegut com a *paral · lelisme implícit.* Aquesta idea, el va portar a analitzar la dinàmica de creixement i decreixement de les instàncies dels esquemes.

En conseqüència, va descriure com evolucionava el nombre d'instàncies d'un esquema en generacions consecutives, a partir de la formulació del teorema fonamental de la teoria dels algorismes genètics: El *teorema dels esquemes.* A continuació presentem aquest resultat i la seva demostració seguint la desenvolupada per Mitchell [Mit98, p.28-30].

Teorema 4.10 (Teorema dels esquemes, J. Holland, [Hol75, Capítol 6])**.** *Sigui H un esquema amb almenys una instància a temps t. Sigui $m(H,t)$ el nombre d'instàncies d'H a temps t i $u(H,t)$ la mitjana de fitness observada (i.e la mitjana del fitness f de les instàncies $x \in H$ a temps t). Si denotem $\mathbb{E}[m(H,t+1)]$ al nombre esperat d'instàncies de l'esquema H a temps $t+1$ després d'efectuar els efectes de selecció, recombinació i mutació de l'algorisme genètic clàssic s'obté:*

$$\mathbb{E}[m(H,t+1)] \geq m(H,t)\frac{u(H,t)}{\bar{f}(t)}\left(1 - p_c\ \frac{\delta(H)}{l-1}\right)(1-p_m)^{o(H)}$$

Nota: Dividirem la demostració en 3 parts. En primer lloc, estudiarem els efectes de la selecció en el nombre esperat d'individus de l'esquema, aquest nombre el denotarem com $\mathbb{E}_{sel}[m(H,t+1)]$. Seguidament, veurem l'efecte de la recombinació en el nombre anterior, que denotarem com $\mathbb{E}_{sel,cross}[m(H,t+1)]$. Finalment, treballarem amb els efectes de la mutació obtenint la cota de $\mathbb{E}_{sel,cross,mut}[m(H,t+1)] = \mathbb{E}[m(H,t+1)]$.

Demostració. Denotem $P(t)$ la població de cromosomes a temps t. Donada una cadena $x \in P(t)$, la seva probabilitat de ser escollida per a reproduir-se en una iteració del procés de selecció és

$$p = \frac{f(x)}{\sum_{y \in P(t)} f(y)}$$

Sigui n la mida de la població seleccionem n individus de manera independent, aleshores el nombre de vegades que x és escollit per a reproduir-se és una variable aleatòria amb distribució Binomial de paràmetres p i n. En conseqüència, el nombre esperat de fills d'una cadena x és

$$np = n \ \frac{f(x)}{\sum_{y \in P(t)} f(y)} = \frac{f(x)}{\bar{f}(t)}$$

El nombre esperat d'instàncies després de la selecció, serà per definició

$$\mathbb{E}_{sel}[m(H, t+1)] = \sum_{x \in H} \frac{f(x)}{\bar{f}(t)} = m(H,t)\frac{u(H,t)}{\bar{f}(t)}$$

Ja que $u(H,t) = \frac{\sum_{x \in H} f(x)}{m(H,t)}$

Notem que la recombinació pot crear i destruir instàncies de l'esquema H, com el nostre objectiu és trobar una cota inferior només considerarem el seu caràcter destructiu. Denotem $S_c(H)$ a la probabilitat que H sobrevisqui la recombinació i $D_c(H)$ la probabilitat que sigui destruït.

Perquè la recombinació destrueixi un esquema H ha de seleccionar un punt de tall c que es trobi entre la primera i la última posició fixada, aleshores hi ha $\delta(H)$ punts de tall que poden trencar l'esquema dels $l-1$ punts totals.

$$D_c(H) \leq p_c \left(\frac{\delta(H)}{l-1} \right)$$

Notem que obtenim una cota superior, ja que pot ser que algun dels $\delta(H)$ punts no trenqui l'esquema. Aleshores tenim

$$S_c(H) = 1 - D_c(H) \geq 1 - p_c \left(\frac{\delta(H)}{l-1} \right)$$

Per tant tenim la següent cota inferior per al nombre esperat d'instàncies de l'esquema H a temps $t+1$ després dels efectes de selecció i recombinació suposant que

aquests dos operadors s'apliquen de manera independent

$$\mathbb{E}_{sel,cross}[m(H,t+1)] \geq m(H,t)\frac{u(H,t)}{\bar{f}(t)}\left(1-p_c\ \frac{\delta(H)}{l-1}\right)$$

Finalment estudiem l'efecte de la mutació. De la mateixa manera que la recombinació només considerarem el seu caràcter destructiu. Denotem $S_m(H)$ a la probabilitat que H sobrevisqui a la mutació

Sigui p_m la probabilitat que un gen sigui mutat, aleshores tenim que la probabilitat que un esquema H sobrevisqui és la probabilitat que els gens definits no siguin mutats. Per tant tenim que

$$S_m(H) = (1-p_m)^{o(H)}$$

Doncs, el nombre esperat d'instàncies de l'esquema H a temps $t+1$ després dels efectes de selecció, recombinació i mutació suposant que es realitzen de manera independent és tal com volíem veure:

$$\mathbb{E}[m(H,t+1)] \geq m(H,t)\frac{u(H,t)}{\bar{f}(t)}\left(1-p_c\ \frac{\delta(H)}{l-1}\right)(1-p_m)^{o(H)}$$

□

D'aquest teorema es dedueix que els esquemes curts, de baix ordre amb fitness observat per sobre la mitjana augmenten exponencialment el seu nombre de proves en generacions consecutives. Aquests esquemes reben el nom de *building blocks*. Aquesta interpretació ens porta a analitzar si augmentar exponencialment el nombre de proves dels *building blocks* és una bona estratègia.

4.3.2 Localització òptima de les proves

Recordem que en els problemes amb els quals treballen els algorismes genètics és crucial el balanç entre l'exploració i l'explotació. Això ens porta a preguntar-nos si l'estratègia que segueixen els algorismes genètics és l'òptima per a aconseguir aquest equilibri. El dilema de l'exploració-explotació es pot modelar amb el problema de la màquina escurabutxaques amb dos braços.

L'escenari del problema és el següent: Considerem que un apostador té N monedes per jugar a una màquina escurabutxaques amb dos braços. Suposem que el braç de l'esquerra i de la dreta tenen uns beneficis mitjans μ_1, μ_2 i variàncies σ_1^2, σ_2^2 respectivament. Sense perdre generalitat, podem suposar que $\mu_1 > \mu_2$.

Els valors dels beneficis mitjans i variàncies són desconeguts per l'apostador, però poden ser estimats jugant les monedes en els diferents braços i observant les recompenses obtingudes. L'objectiu del problema és maximitzar la recompensa total durant les N tirades. Observem que és un objectiu dual: per una banda, ens interessa endevinar el braç amb millor taxa de recompensa, però també és important maximitzar la recompensa en el transcurs de guanyar informació assignant els assajos als dos braços. Aleshores el balanç entre l'exploració i l'explotació és un aspecte fonamental en aquest problema.

Holland [Hol75, Capítol 6] utilitza aquest problema per entendre l'eficiència dels algorismes genètics. En aquest context, analitza quina és la millor estratègia per a repartir les proves entre els dos braços. A partir del següent resultat, proporciona una fórmula asimptòtica per al nombre òptim d'assajos a assignar al braç aparentment subòptim.

Teorema 4.11 (J. Holland, [Hol75, p.76-82])**.** *Considerem N assajos per ser col·locats entre dues variables aleatòries X_1 i X_2 amb mitjanes $\mu_1 > \mu_2$ i variàncies σ_1^2 i σ_2^2 respectivament. Denotem $\xi_1(N)$ i $\xi_2(N)$ les variables aleatòries amb major/menor benefici mitjà observat després de N assajos. Aleshores si les dues variables són igual de probables inicialment de ser les millors, és a dir $\mathbb{P}(\mu_1 > \mu_2) = \mathbb{P}(\mu_2 > \mu_1)$ tenim que el valor mínim de la pèrdua esperada es dona quan el nombre d'assajos col·locats a $\xi_2(N)$ és* [5]

$$n^\star \sim b^2 \ln\left(\frac{N^2}{8\pi b^4 \ln(N^2)}\right) \tag{4.4}$$

on $b = \frac{\sigma_1}{\mu_1 - \mu_2}$

A partir de l'equació 4.4 obtenim:

$$N - n^\star \sim N \sim \sqrt{8\pi b^4 \ln(N^2)} \exp\left(\frac{n^\star}{2b^2}\right) \tag{4.5}$$

A partir de l'expressió 4.5 s'observa que l'estratègia òptima és augmentar exponencialment el nombre de proves en el millor braç observat. Aquest resultat es pot generalitzar amb una màquina de k braços. Segons Bodenhofer [Bod03, p.41], s'obté de manera anàloga que les millors alternatives han d'obtenir un nombre exponencialment creixent de proves.

[5] $Y(t) \sim Z(t)$ vol dir que $\lim_{t\to\infty} \frac{Y(t)}{Z(t)} = 1$.

Bäck [Bäc96, p.127] presenta la següent connexió entre aquest problema i els algorismes genètics: Per una cadena de longitud l, cada conjunt de j posicions fixades defineix una màquina. Cadascúna d'aquestes és un esquema d'ordre j amb unes posicions fixades localitzades de manera específica. En total hi ha $\binom{l}{j}$. Cadascúna d'aquestes màquines té 2^j braços que representen les combinacions de valors binàris que poden tenir les posicions fixades. Cada instància d'un esquema dins de la població representa una prova en un braç concret i el fitness mitjà de l'esquema correspon al benefici mitjà d'aquest braç. D'aquesta manera, dins de cada conjunt de posicions, els diferents esquemes competeixen com fan els braços d'una màquina.

Aquesta analogia mostra que l'estratègia dels algorismes genètics d'augmentar exponencialment el nombre d'instàncies dels *building blocks* en la població és òptima per regular el balanç de l'exploració i explotació.

5 *Simulated Annealing*

5.1 Introducció

El recuit o *annealing* és un procés físic que consisteix a escalfar un sòlid fins a una temperatura específica on totes les partícules s'ordenen de manera aleatòria, per després refredar-lo lentament fins que les partícules s'ordenen en un estat baix d'energia.

Durant la fase de refredament, per cada valor de *temperatura* fix T, el sòlid pot arribar al que s'anomena *equilibri tèrmic*. Diem que un sòlid arriba a l'equilibri tèrmic si la probabilitat que es trobi en un estat i amb energia E_i ve donada per la distribució de Boltzmann:

$$\mathbb{P}(X = i) = \frac{1}{Z(T)} \cdot \exp(-\frac{E_i}{k_b T})$$

on $Z(T) = \sum_{j \in S} \exp(-\frac{E_j}{k_b T})$ i k_b la *constant de Boltzmann*

Notem que a mesura que la temperatura decreix, la distribució es centra en els estats amb baixa energia i quan la temperatura $T \to 0$ només el valor mínim d'energia té probabilitat d'aparèixer diferent de zero.

L'algorisme *Simulated Annealing* estableix una analogia entre aquest procés físic i la resolució d'un problema d'optimització combinatoria per cercar l'òptim. Per poder presentar aquest mètode és necessari, en primer lloc, definir els algorismes Monte Carlo i en particular l'algorisme de Metropolis.

5.1.1 Algorismes Monte Carlo

Els algorismes Monte Carlo són algorismes d'optimització basats en l'exploració del veïnat dels punts de l'espai de solucions. Comencem introduint les següents definicions

Definició 5.1. Sigui $(\mathcal{R}, f)$ un problema d'optimització combinatoria on $\mathcal{R}$ és l'espai de possibles solucions i f és la funció objetiu, anomenem *estructura de veïnat* a una aplicació $\mathscr{N} : \mathcal{R} \to \mathcal{P}(\mathcal{R})$ tal que per a cada solució $i \in \mathcal{R}$ defineix un subconjunt $\mathcal{R}_i \subseteq \mathcal{R}$ de solucions. Anomenem *veïnat de la solució* i al conjunt $\mathcal{R}_i$.

Definició 5.2. Sigui $(\mathcal{R}, f)$ un problema d'optimització combinatòria amb una estructura de veïnat $\mathscr{N}$ anomenem *mecanisme de generació* a una manera de seleccionar una solució $j \in \mathcal{R}_i$ a partir de qualsevol veïnat $\mathcal{R}_i$.

Definició 5.3. Sigui $(\mathcal{R}, f)$ un problema d'optimització combinatòria amb estructura de veïnat $\mathscr{N}$, i sigui $\{x_k\}_{k=0}^K$ una successió d'iterats amb $x_k \in \mathcal{R}$. Anomenem *criteri d'acceptació* una aplicació que assigna, per a cada parell de solucions $i, j \in \mathcal{R}$, una probabilitat

$$\mathbb{P}(x_{k+1} = j \mid x_k = i),$$

tal que $\mathbb{P}(x_{k+1} = j \mid x_k = i) > 0$ només si $j \in \mathscr{N}(i)$, i

$$\sum_{j \in \mathscr{N}(i)} \mathbb{P}(x_{k+1} = j \mid x_k = i) = 1.$$

Definició 5.4. Un *algorisme Monte Carlo* per a un problema d'optimització combinatòria $(\mathcal{R}, f)$ amb estructura de veïnat $\mathscr{N}$ consisteix en un procés estocàstic que genera una successió d'iterats $\{x_k\}_{k=0}^K \subseteq \mathcal{R}$ mitjançant la repetició dels següents passos:

1. A partir de la solució actual $x_k = i$, es genera una solució candidata $j \in \mathscr{N}(i)$ mitjançant un mecanisme de generació.
2. S'accepta la solució candidata amb una probabilitat donada per un criteri d'acceptació $\mathbb{P}(x_{k+1} = j \mid x_k = i)$.
3. Si la solució candidata és acceptada, s'assigna $x_{k+1} := j$; altrament, $x_{k+1} := x_k$.

Anomenem *transició* al conjunt d'aquests tres passos, és a dir a l'aplicació del mecanisme de generació i el mecanisme d'acceptació i la selecció de la següent iteració.

5.1.2 Algorisme de Metropolis

Per simular l'evolució de l'equilibri tèrmic d'un sòlid a un valor de temperatura fixa T, Metropolis *et al* [Met+53], van proposar *l'algorisme de Metropolis*, un algorisme de Monte Carlo que utilitza el següent criteri d'acceptació:

$$\mathbb{P}(x_{k+1} = j \mid x_k = i) := \begin{cases} 1 & \text{si } \Delta E < 0 \\ \exp\left(-\frac{\Delta E}{c}\right) & \text{si } \Delta E \geq 0 \end{cases}$$

on $\Delta E := E_j - E_i$ s'anomena *diferència d'energia* i $c := k_b T$ *paràmetre de control.*

Tal i com expliquen van Laarhoven *et al.* [LA87, p.8], seguint aquest criteri anomenat *criteri de Metropolis*, el sistema evoluciona a l'equilibri tèrmic, és a dir, després d'un gran nombre d'iteracions la distribució de probabilitat tendeix a la distribució de Boltzmann.

5.2 Algorisme *Simulated Annealing*

L'algorisme *Simulated Annealing* (SA) cerca l'òptim d'un problema d'optimització combinatòria $(\mathcal{R}, f)$ establint la següent analogia entre un sistema de partícules i el problema d'optimització: Els punts i de l'espai de solucions $\mathcal{R}$ representen els estats de partícules i la funció objectiu f representa l'energia E del sòlid. El procediment consisteix en una successió d'algorismes de Metropolis aplicats sobre una successió decreixent de valors del paràmetre de control $\{c_k\}_{k\in\mathbb{N}}$, mantenint cada valor c_k durant L_k iterats. Aquesta successió es defineix a partir d'un procés de refredament.

Definició 5.5. Un *procés de refredament* és una estratègia que determina l'evolució del paràmetre de control c. Es defineix a partir dels següents elements:

- Una successió finita de valors del paràmetre de control $\{c_k\}_{k=0}^{K}$ amb $c_k \in \mathbb{R}_{>0}$ per $k = 0, \ldots, K$ determinada a partir de:
  - Un *valor inicial* c_0
  - Una *funció de decreixement* φ tal que $c_{k+1} = \varphi(c_k)$ per $k = 0, ..., K-1$
  - Un *criteri de parada* que determina el nombre d'iteracions K
- Un nombre finit d'iteracions L_k per a cada valor c_k amb $k = 0, \ldots, K$.

Segons D. Delahaye [DCM18, p.18] el procés de refredament més comú és el *refredament geomètric*. A continuació presentem aquest mètode, però abans introduïm la següent definició.

Definició 5.6. Sigui L_a el nombre de transicions acceptades després de L transicions proposades per un algorisme de Metropolis amb paràmetre de control c, anomenem *proporció d'acceptació* al valor.

$$\chi(c) = \frac{L_a}{L}$$

Refredament geomètric

- **Valor inicial del paràmetre de control:** L'objectiu és trobar un valor inicial c_0 prou gran perquè gairebé totes les transicions siguin acceptades, és a dir $\chi(c_0)$ sigui proper a 1. El mètode per determinar aquest valor és el següent: Se selecciona un valor per c_0 i s'aplica un algorisme de Metropolis d'un nombre de transicions L. El valor $\chi(c_0)$ es compara amb un valor fixat χ_0. Si la proporció d'acceptació $\chi(c_0)$ és menor que χ_0 es duplica el valor de c_0. El procés continua fins que per un valor c_0 s'obté una proporció $\chi(c_0) > \chi_0$, aleshores es pren aquest valor com a paràmetre de control inicial de l'algorisme *Simulated Annealing*.

- **Funció de decreixement:** Aquest procés utilitza com a funció de decreixement

  $$\varphi(c_k) = \alpha c_k$$

  on α és un hiperparàmetre tal que $\alpha \in (0, 1)$. Notem que com més gran sigui el valor d'α, més lentament decreix el paràmetre de control.

- **Criteri de parada:** L'algorisme finalitza quan el sistema es congela, és a dir per algún $k \in \mathbb{N}$, se satisfà una de les següents condicions de parada:
  - La proporció d'acceptació $\chi(c_k)$ és menor que un valor fixat χ_f.
  - El paràmetre de control c_k és menor que un valor fixat c_{min}

- **Nombre d'iteracions:** Per que els algorismes de Metropolis arribin a l'equilibri tèrmic, s'ha de permetre que l'algorisme accepti un nombre suficient de transicions L_a^{max}. La selecció d'aquest valor depèn del problema. D'altra banda, com que el nombre d'acceptacions disminueix amb el temps, el nombre de transicions s'acota inferiorment per un valor L^{max}.

Un cop introduïts aquests elements, el funcionament d'un algorisme *Simulated Annealing* amb procés de refredament geomètric es pot descriure a partir del següent pseudocodi:

Algorithm 5.1 Algorisme *Simulated Annealing* (SA)

procedure SA($f, S, \alpha, c_0, c_{min}, L^{\max}, L_a^{\max}, \chi_f, \chi_0, k_b$)
  $c \leftarrow c_0$
  Triar aleatòriament $x \in S$
  $\chi \leftarrow 0$
  while $\chi < \chi_0$ **do** ▷ Càcul temepratura inicial
    $x' \leftarrow$ mecanisme_generacio(x)
    $\Delta f \leftarrow f(x') - f(x)$
    if $\Delta f < 0$ **then**
      $x \leftarrow x'$
      $L_a \leftarrow L_a + 1$
    else if rand$(0,1) < \exp(-\Delta f/c)$ **then**
      $x \leftarrow x'$
      $L_a \leftarrow L_a + 1$
    end if
    $L \leftarrow L + 1$
    $c_0 \leftarrow 2c_0$
  end while
  while $c > c_{min}$ and $L_a/L \geq \chi_f$ **do** ▷ Algorismes de Metropolis
    $L \leftarrow 0$ ▷ Comptador de transicions
    $L_a \leftarrow 0$ ▷ Comptador d'acceptacions
    while $L < L^{\max}$ **and** $L_a < L_a^{\max}$ **do**
      $x' \leftarrow$ mecanisme_generacio(x) ▷ Veí de x
      $\Delta f \leftarrow f(x') - f(x)$
      if $\Delta f < 0$ **then**
        $x \leftarrow x'$ ▷ Millora acceptada
        $L_a \leftarrow L_a + 1$
      else if rand$(0,1) < \exp(-\Delta f/c)$ **then**
        $x \leftarrow x'$ ▷ Acceptació Metropolis
        $L_a \leftarrow L_a + 1$
      end if
      $L \leftarrow L + 1$
    end while
    $c \leftarrow \alpha \cdot c$ ▷ Reducció de temperatura
  end while
  return x
end procedure

5.3 Model matemàtic de l'algorisme

En aquesta secció presentem com es pot descriure matemàticament l'algorisme *Simualted Annealing*. Comencem introduint les següents definicions:

Definició 5.7. Una cadena de Markov és un procés estocàstic a temps discret $\{X_k : k = 0, 1, \ldots\}$ amb un espai d'estats discret $\mathcal{A}$ tal que per a qualsevol $k \geq 0$ i per qualsevol $x_0, x_1, \ldots, x_{k+1} \in \mathcal{A}$ se satisfà:

$$\mathbb{P}(X_{k+1} = x_{k+1} | X_0 = x_0, X_1 = x_1, \ldots, X_k = x_k) = \mathbb{P}(X_{k+1} = x_{k+1} | X_k = x_k)$$

Definició 5.8. Siguin $(i, j) \in \mathcal{A} \times \mathcal{A}$ una parella d'estats d'una Cadena de Markov. S'anomena *probabilitat de transició* la probabilitat d'anar de l'estat i en temps k a l'estat j en temps $k + 1$, que denotem per

$$P_{ij}(k, k+1) = \mathbb{P}(X_{k+1} = j \mid X_k = i).$$

Anomenem *matriu de transició* la matriu $P(k, k+1) \in \mathcal{M}_{|\mathcal{A}|\times|\mathcal{A}|}([0, 1])$ que té per components les probabilitats de transició, és a dir:

$$P(k, k+1) = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1|\mathcal{A}|} \\ p_{21} & p_{22} & \cdots & p_{2|\mathcal{A}|} \\ \vdots & \vdots & \ddots & \vdots \\ p_{|\mathcal{A}|1} & p_{|\mathcal{A}|2} & \cdots & p_{|\mathcal{A}||\mathcal{A}|} \end{bmatrix} \quad \text{on } p_{ij} = P_{ij}(k, k+1).$$

Definició 5.9. Diem que una Cadena de Markov és *homogenia* si les probabilitats de transició no depenen del temps, altrament, si depenen del temps, diem que la Cadena de Markov és *no homogènia*.

Observació. Notem que el *Simulated Annealing* és un procés estocàstic discret que satisfà la propietat de Markov, ja que és una seqüència de transicions, on el resultat d'una transició només depèn del resultat de l'anterior, és a dir, de la configuració actual. Com que, a més, el paràmetre de control va decreixent el *Simulated Annealing* es pot descriure com una cadena de Markov no homogènia que depèn del paràmetre de control c_k i que té com a espai d'estats l'espai de solucions i.e $\mathcal{A} = \mathcal{R}$.

Definició 5.10. Les probabilitats de transició de l'algorisme *Simulated Annealing* venen donades per l'expressió:

$$\forall i, j \in \mathcal{R} \quad P_{ij}(k, k+1) = P_{ij}(c_k) = \begin{cases} G_{ij}(c_k) A_{ij}(c_k) & \text{si } i \neq j \\ 1 - \sum_{l=1, l\neq i}^{|\mathcal{R}|} G_{il}(c_k) A_{il}(c_k) & \text{si } i = j \end{cases} \tag{5.1}$$

on $G_{ij}(c_k)$ és la probabilitat de generar l'estat j des de l'estat i i $A_{ij}(c_k)$ és la probabilitat d'acceptar l'estat j des de l'estat i. Les probabilitats $G_{ij}(c_k)$ i $A_{ij}(c_k)$ s'anomenen respectivament *probabilitats de generació* i *probabilitats d'acceptació*. Les probabilitats d'acceptació estan determinades per la següent expressió:

$$A_{ij}(c_k) = \min\{1, \exp\left(\frac{f(j) - f(i)}{c_k}\right)\}$$

Observem que l'algorisme *Simulated Annealing* obté el mínim global, si després d'un nombre K de transicions tenim que:

$$\mathbb{P}(X_K \in \mathcal{R}^\star) = 1 \tag{5.2}$$

on $\mathcal{R}^\star$ és el conjunt de configuracions mínimes globals.

En la següent secció presentem un teorema de convergència proposat per Hajek [Haj88] que estableix condicions necessàries i suficients, imposant requisits sobre l'estructura de la cadena de Markov i la velocitat de refredament, per a garantir que l'equació 5.2 es compleix asimptòticament, és a dir, que se satisfà

$$\lim_{k\to\infty} \mathbb{P}(X_k \in \mathcal{R}^\star) = 1,$$

5.4 Convergència asimptòtica a l'òptim

Per a les següents definicions considerem $\{X_k\}_{k\geq 0}$ una cadena de Markov no homogènia amb un espai d'estats $\mathcal{R}$ i probabilitats de transició $P_{ij}(k, k+1)$.

Definició 5.11. Diem que un estat i és *assolible a l'altura* E des d'un estat j si existeix una successió d'estats $\{i_l\}_{0\leq l\leq p}$ amb $i_0 = j$ i $i_p = i$ tal que satisfà:

(i) $P_{i_l,i_{l+1}}(l, l+1) > 0 \qquad \forall\, 0 \leq l < p$

(ii) $f(i_l) \leq E \qquad \forall\, 0 \leq l \leq p$

Definició 5.12. Diem que una cadena de Markov és *irreductible* si donats dos estats qualssevol i i j és assolible a alguna altura E des de j.

Definició 5.13. Diem que una cadena de Markov satisfà la *propietat de reversibilitat feble* quan per a qualsevol nombre $E \in \mathbb{R}$ i dos estats i i j qualsevol, i és assolible a altura E des de j si i només si j és assolible a altura E des de i

En aquest context podem definir un mínim local d'un problema d'optimització combinatori $(\mathcal{R}, f)$ de la següent manera:

Definició 5.14. Diem que un estat i és un mínim local si no existeix cap estat j amb $f(j) < f(i)$ tal que és assolible des de i amb altura $f(i)$. Sigui i un mínim local, definim la profunditat de i denotada per $d(i)$ com el mínim $E > 0$ tal que existeix un estat j amb $f(j) < f(i)$ assolible a altura $f(i) + E$. En el cas que i sigui també un mínim global, es defineix la seva profunditat com $d(i) = +\infty$

Teorema 5.15 (B. Hajek, [Haj88])**.** *Donada una cadena de Markov no homogènia definida per la matriu de transició 5.1 suposem és irreductible i satisfà la propietat de reversibilitat feble. A més, suposem que la successió* $\{c_k\}_{k\geq 0}$ *satisfà:*

(i) $c_k \geq c_{k+1} \qquad \forall k = 0, 1, \ldots$

(ii) $lim_{k\to\infty} c_k = 0$

Sigui $d^\star$ *el màxim de les profunditats* $d(i)$ *de totes les configuracions* i*, que són mínims locals, però no globals, aleshores*

$$\lim_{k\to\infty} \mathbb{P}(x_k \in S^\star) = 1 \tag{5.3}$$

si i només si

$$\sum_{k=0}^{\infty} \exp\left(-\frac{d^\star}{c_k}\right) = \infty \tag{5.4}$$

Observació. Suposem que parametritzem el paràmetre de control de la següent manera:

$$c_k = \frac{a}{\log(k+1)}$$

Aleshores podem reescriure l'equació 5.4 de la següent forma:

$$\sum_{k=0}^{\infty} \left(\frac{1}{k+1}\right)^{\frac{d^\star}{a}} = \infty \tag{5.5}$$

A partir de 5.5 veiem que les equacions 5.3 i 5.4 se satisfan si només si $a \geq d^\star$.

Tot i que a la pràctica, aquesta funció de decreixement no s'utilitza habitualment perquè és massa costosa computacionalment, la reducció logarítmica permet veure un cas en què podem donar condicions suficients per a la convergència.

A partir del teorema 5.15 tenim les següents condicions necessàries per a la convergència a l'òptim del *Simulated Annealing*:

- L'estructura de veïnat i mecanisme de generació ha de permetre arribar a tots els punts de l'espai $\mathcal{R}$ des de qualsevol altre punt per garantir la irreductibilitat.

- L'algorisme ha de satisfer la propietat de reversibilitat feble. Una manera intuïtiva i usada habitualment és a partir de la simetria del mecanisme de generació

- La successió del paràmetre de control $\{c_k\}$ ha de convergir a zero i decréixer prou lentament per assegurar la divergència de la sèrie $\sum_k \exp(-d^\star/c_k)$. En cas contrari, l'algorisme pot quedar atrapat en un òptim local.

6 Ajust del model de dinàmica poblacional de les gavines del Delta del Ebre

Un cop introduïdes les eines necessàries, l'objectiu és aplicar la metodologia proposada utilitzant diferents variants dels mètodes heurístics presentats. El problema en el qual ens centrarem és l'ajust d'un model de dinàmica de població de gavines del delta de l'Ebre proposat i resolt per Oro *et al.* [Oro+23]. Concretament, ens centrarem en les dades de la denominada Fase de Col · lapse (2006-2016). En primer lloc, presentem el model de la dinàmica poblacional.

6.1 Modelització de la dinàmica de la població

Denotem per $x(t)$ la mida de la població a temps t. Aleshores el model s'escriu com:

$$\frac{d}{dt}x(t) = (\vartheta + \omega)x(t)\left(1 - \frac{x(t)}{K}\right) - \varepsilon x(t) - [\rho x(t) + \lambda D(x(t))] \tag{6.1}$$

En primer lloc, l'equació considera un creixement exponencial proporcional al paràmetre $\gamma = \vartheta + \omega$ que inclou les taxes de reproducció (ϑ) i d'immigració (ω). Aquest creixement de la població està regulat per una funció logística de la capacitat de càrrega K. Aquesta funció restringeix el creixement, el qual no pot ser infinit en sistemes biològics a causa de factors com el límit de recursos, la competència intraespecífica o l'espai disponible.

El següent terme considera el decreixement de la població proporcional a la taxa de mortalitat fixada $\varepsilon = 0.11$, estimada experimentalment. Els últims dos factors de l'equació modelen la dispersió. El primer representa una dispersió exponencial proporcional a una constant ρ. El segon terme introdueix la dispersió produïda pel fenomen de còpia social a partir de la funció $D(x(t))$.

Definim el paràmetre $\varphi = \gamma - \epsilon - \rho$. Aquest agrupa els processos de reproducció, immigració, mort i dispersió exponencial. Definim també el paràmetre $\beta = \gamma/K$ que acompanya al terme no lineal del model. A partir d'aquests nous factors podem

reescriure l'equació 6.1 de manera compacta, reagrupant els termes en tres tipus diferents: creixement de la població, competició intraespecífica i dispersió per còpia social.

$$\frac{d}{dt}x(t) = \varphi x(t) - \beta x(t)^2 - \lambda D(x(t)) \tag{6.2}$$

A la següent taula trobem determinats tots els paràmetres que intervenen en la formulació del model.

Paràmetres	**Unitats**	**Rang de valors**	**Significat**
ϑ	any^{-1}	$[0, +\infty)$	Taxa de reproducció
ω	any^{-1}	$[0, +\infty)$	Taxa d'immigració
K	ocells	$[1, +\infty)$	Nombre màxim d'individus que pot sostenir un ecosistema (capacitat de càrrega)
ε	any^{-1}	0.11	Taxa de mortalitat
ρ	any^{-1}	$\mathbb{R}^+$	Taxa de dispersió
$\gamma = \vartheta + \omega$	any^{-1}	$[0, +\infty)$	Taxa de creixement a causa de la reproducció i immigració
$\alpha = \gamma - \varepsilon$	any^{-1}	$[-\varepsilon, +\infty)$	Creixement poblacional net sense dispersió lineal
$x(0)$	ocells	$[0, K]$	Condició inicial de l'equació 6.1
$\varphi = \alpha - \rho$	any^{-1}	$(-\infty, \alpha]$	Taxa de creixement poblacional incloent dispersió lineal
$\beta = \frac{\gamma}{K}$	$(\text{ocells} \times \text{any})^{-1}$	$[0, +\infty)$	Taxa de creixement intrínseca sobre la capacitat de càrrega
λ	any^{-1}	$\mathbb{R}^+$	Taxa de dispersió per còpia social

Taula 2: Paràmetres del model de la dinàmica de població de les gavines del Delta de l'Ebre [Oro+23, p.8]

Un cop determinat el model, cal seleccionar una funció $D(x(t))$ que representi la dispersió induïda per la còpia social. En aquest cas s'escull la sigmoide d'Elliot, ja que la seva versatilitat permet obtenir una gran varietat de formes en funció dels seus paràmetres i, per tant, permet trobar diferents comportaments. Aleshores, la funció de dispersió social proposada ve donada per la següent expressió:

$$D(x,\mu,\sigma,\delta) := \begin{cases} \frac{1-\mathcal{E}_{dir}(x,\mu,\sigma,\delta)}{1-\mathcal{E}_{dir}(0,\mu,\sigma,\delta)} & \text{si } 0 \leq x \leq \delta \\ \frac{1-\mathcal{E}(x,\sigma,\delta)}{1-\mathcal{E}_{dir}(0,\mu,\sigma,\delta)} & \text{si } x > \delta \end{cases}$$

on

$$\mathcal{E}_{dir}(x,\mu,\sigma,\delta) := \left(\mu\frac{\Theta+\sigma\delta}{2\Theta+\sigma\delta}\left(1-\frac{x}{\delta}\right)+\frac{x}{\delta}\right)\mathcal{E}(x,\sigma,\delta)$$

$$\mathcal{E}(x,\sigma,\delta) := \frac{\sigma(x-\delta)}{\Theta+\sigma|x-\delta|}$$

Aquesta funció s'anomena sigmode d'Elliot Θ-escalada, σ-intensificada i $\delta-$desplaçada. Considerarem que tots els paràmetres són positius i el paràmetre $\Theta := 1000$ fix. Aquest controla l'escala de la variable independent x. A la següent taula trobem determinats tots els paràmetres que intervenen en la funció de dispersió per còpia social proposada:

Paràmetres	**Rang de valors**	**Significat**
μ	$\mathbb{R}^+$	Tendència de la funció de dispersió per a poblacions petites dispersió
σ	$\mathbb{R}^+$	Agudesa o suavitat de la funció de dispersió
δ	$\mathbb{R}^+$	Transició entre poblacions grans i petites

Taula 3: Paràmetres del model de la dinàmica de població de les gavines del Delta de l'Ebre [Oro+23, p.15]

6.2 Plantejament del problema de minimització

En aquest apartat treballarem amb el model diferencial 6.1 i amb el següent conjunt de dades.

$$\pi(t, t = 0:11) = [15329, 14177, 13031, 9762, 11271, 8688, 7571, 6983, 4778, 2067, 1586, 793]$$

És important remarcar que l'ajust a les dades parteix amb els paràmetres β,α,γ i K fixats als valors trobats per Oro *et al.* [Oro+23, p.13] a partir de l'ajust a les dades de la fase inicial. Aleshores el model depèn de 6 paràmetres: $x_0, \varphi, \lambda, \mu, \sigma$ i δ. Denotem la solució de l'equació diferencial com $x(t) := x_{x_0,\varphi,\lambda,\mu,\delta,\sigma}(t)$ per $t \in [0, 11]$.

Considerem l'espai de paràmetres $\mathscr{P} := [0, K] \times [-\infty, \alpha] \times \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}^+ \times \mathbb{R}^+$ i definim la següent funció d'error:

$$\begin{aligned} F: \quad & \mathscr{P} \longrightarrow \mathbb{R}_{\geq 0} \\ & (x_0, \varphi, \lambda, \mu, \sigma, \delta) \longmapsto \sqrt{\sum_{i=1}^{11} (x(i) - \pi(i))^2} \end{aligned} \tag{6.3}$$

Aleshores l'ajust del model consisteix a resoldre el següent problema d'optimització:

$$\begin{gathered} \min F(x(0), \varphi, \lambda, \mu, \sigma, \delta) \\ \text{subjecte a } (x(0), \varphi, \lambda, \mu, \sigma, \delta) \in \mathscr{P}, \\ \text{i } x(t) \in [0, K] \quad \text{per a } t \in [0, 11] \end{gathered}$$

Un cop plantejat el problema és necessari trobar un compacte $\mathscr{K} \subset \mathscr{P}$ que contingui el mínim per dos motius: Com que F és contínua pel teorema de Weierstrass la funció té un mínim a $\mathscr{K}$. D'altra banda, necessitem una reducció de l'espai de paràmetres per poder aplicar algorismes de minimització. A partir d'arguments semi-analítics D.Oro *et al.* [Oro+23, p.27-31] proposen el següent problema reduït

$$\begin{gathered} \min F(x(0), \varphi, \lambda, \mu, \sigma, \delta) \\ \text{subjecte a } (x(0), \varphi, \lambda, \mu, \sigma, \delta) \in \mathscr{K}, \\ \text{i } x(t) \in [0, K] \quad \text{per a } t \in [0, 11], \end{gathered} \tag{6.4}$$

on $\mathscr{K} := [12726, 17932] \times [0.12, \alpha] \times [300, 3000] \times [0, 10] \times [0, 50] \times [0, 20000]$

En les seccions 6.3, 6.4 i 6.5 s'implementen en llenguatge `C` diferents variants dels mètodes heurístics presentats a les seccions 4 i 5 i es compara la seva eficiència per a la resolució del problema de minimització 6.4. Tots els fitxers de codi font, així com els resultats numèrics obtinguts en les diferents execucions es troben disponibles en un repositori de GitHub. L'enllaç es proporciona a l'apèndix B.

6.3 Cerca de l'òptim a partir d'un Algorisme Genètic Discret

Per a la resolució del problema D.Oro *et al.* [Oro+23, p.35-37] proposen una implementació que consisteix a aplicar successivament un algorisme genètic estàndard sobre diferents poblacions on afegim els millors individus obtinguts anteriorment. Com que en la primera aplicació no es té cap coneixement previ s'afegeix el millor individu trobat en una exploració a força bruta. D'aquesta manera les poblacions

inicials cada cop es van tornant més elitistes. Finalment, es pren com a solució el millor individu trobat durant tot el procés. A partir d'aquest mètode s'obté que l'òptim del problema 6.4, que denotarem com $x^\star$, és:

Paràmetres	Valors
x_0	15670.5560275192783593
φ	0.2497248909716255
λ	1570.2313809039706030
μ	0.0000000000000000
σ	0.4904756364357690
δ	8944.2282749675759987
$fitness$	2566.999667640135158

Taula 4: Òptim $x^\star$ i el seu valor fitness

En aquesta secció implementarem una versió estàndard d'un algorisme genètic discret i compararem la seva eficiència amb el mètode anterior.

6.3.1 Codificació

Per a la codificació dels individus de l'algorisme genètic discret seleccionem el mètode de codificació binària. Realitzem la següent discretització a l'espai de cerca dels paràmetres de manera anàloga a l'exemple 4.3:

Paràmetres	Rang de cerca	Precisió	Espai de codificació	Pas de discretització	Aplicació de translació genotip → fenotip
x_0	$[12726, 17932]$	10^{-2}	$S_{x_0} := \{0, \dots, 2^{19} - 1\}$	$\frac{5206}{2^{19}-1}$	$\boldsymbol{s_{x_0}} : u \mapsto 12726 + u \cdot \frac{5206}{2^{19}-1}$
φ	$[0.12, \alpha]$	10^{-10}	$S_\varphi := \{0, \dots, 2^{32} - 1\}$	$\frac{\alpha - 0.12}{2^{32}-1}$	$\boldsymbol{s_\varphi} : u \mapsto 0.12 + u \cdot \frac{\alpha - 0.12}{2^{32}-1}$
λ	$[300, 3000]$	10^{-2}	$S_\lambda := \{0, \dots, 2^{19} - 1\}$	$\frac{2700}{2^{19}-1}$	$\boldsymbol{s_\lambda} : u \mapsto 300 + u \cdot \frac{2700}{2^{19}-1}$
μ	$[0, 10]$	10^{-6}	$S_\mu := \{0, \dots, 2^{19} - 1\}$	$\frac{10}{2^{19}-1}$	$\boldsymbol{s_\mu} : u \mapsto u \cdot \frac{10}{2^{19}-1}$
σ	$[0, 50]$	10^{-4}	$S_\sigma := \{0, \dots, 2^{24} - 1\}$	$\frac{50}{2^{24}-1}$	$\boldsymbol{s_\sigma} : u \mapsto u \cdot \frac{50}{2^{24}-1}$
δ	$[0, 20000]$	0.5	$S_\delta := \{0, \dots, 2^{16} - 1\}$	$\frac{20000}{2^{16}-1}$	$\boldsymbol{s_\delta} : u \mapsto u \cdot \frac{20000}{2^{16}-1}$

Taula 5: Discretització de l'espai de cerca i translació de genotip a fenotip. Adaptada a partir de Oro *et al.* [Oro+23, p.33].

Tot i que la definició formal de la funció morfogènica es realitza sobre l'espai de cadenes de bits, a la pràctica treballem sobre l'espai de codificació $S := S_{x_0} \times S_\varphi \times S_\lambda \times S_\mu \times S_\sigma \times S_\delta$ gràcies a la bijecció entre els dos espais. Aleshores els gens es representaran mitjançant nombres enters (tipus `unsigned int` [6]).

[6] Aquest tipus de dada permet representar enters sense signe de 32 bits, és a dir els valors $\{0, \dots, 2^{32} - 1\}$, per tant, és adequada per a representar tots els gens

Definim la següent funció morfogènica:

$$m: \quad S \longrightarrow \mathscr{K}$$

$$(u_{x_0}, u_\varphi, u_\lambda, u_\mu, u_\sigma, u_\delta) \longmapsto (s_{x_0}(u_{x_0}), s_\varphi(u_\varphi), s_\lambda(u_\lambda), s_\mu(u_\mu), s_\sigma(u_\sigma), s_\delta(u_\delta))$$

6.3.2 Avaluació del fitness

Per dur a terme l'avaluació dels individus, es considera la següent funció fitness:

$$\text{fitness}: \quad S \longrightarrow \mathbb{R}_{\geq 0}$$

$$(u_{x_0}, u_\varphi, u_\lambda, u_\mu, u_\sigma, u_\delta) \longmapsto F(s_{x_0}(u_{x_0}), s_\varphi(u_\varphi), s_\lambda(u_\lambda), s_\mu(u_\mu), s_\sigma(u_\sigma), s_\delta(u_\delta))$$

on F és la funció objectiu definida a l'equació 6.3.

Notem doncs que fer aquesta avaluació necessitem els valors $x(i)$ per $i = 1, \ldots, 11$. Les solucions aproximades de l'equació diferencial $u(i)$ per $i = 1, \ldots, 11$ s'obtenen a partir d'una rutina de Runge-Kutta-Fehlberg 7-8 amb pas adaptatiu mitjançant els codis en `C` proporcionats per Alsedà [Als25].

6.3.3 Operadors

Per implementar l'operador selecció utilitzem el mètode de selecció per torneig. Aquest mètode es troba definit per a un problema de maximització a la secció 4.2.3. Com ara l'objectiu és resoldre un problema de minimització aplicarem la modificació de triar l'individu amb menor fitness d'entre els t escollits aleatòriament.

Per a la implementació dels operadors recombinació i mutació usem respectivament els mètodes de recombinació d'un punt i mutació *bit-flip* aplicats individualment a cada gen. Recordem que aquests operadors sí que modifiquen l'estructura del genotip. Considerant que s'han utilitzat nombres enters per a la codificació dels gens s'utilitzen operacions *bitwise*[7] per poder modificar la seva representació binària.

[7]Les operacions *bitwise* són operacions a nivell de bits que manipulen la representació binària dels nombres enters. Per una descripció més detallada i exemples de la seva aplicació en operadors genètics, veure l'Apèndix A.

6.3.4 Selecció d'hiperparàmetres

Un procés crucial en la implementació d'algorismes genètics és la selecció d'hiperparàmetres, ja que permeten equilibrar les estratègies d'exploració i explotació. En l'algorisme genètic discret plantejat tenim quatre hiperparàmetres:

- `POPSIZE`: Mida de la població
- `T`: Mida de torneig
- `P_C`: Probabilitat de recombinació
- `P_M`: Probabilitat de mutació

Per dur a terme la millor selecció dels hiperparàmetres no disposem de resultats generals. Segons Mitchell [Mit98, p.175], els hiperparàmetres normalment interaccionen els uns amb els altres de manera no lineal, així que no és possible optimitzar-los alhora. La tria d'aquests valors depèn del problema a resoldre.

Notem que el nostre problema de minimització treballa amb un espai de solucions molt gran, de cardinalitat 2^{129}. Això ens porta a pensar que per poder garantir una bona representació de les possibles solucions i l'exploració de l'espai de cerca, és necessari tenir una població formada per bastants individus. Per aquest motiu en les següents execucions prendrem el valor $\texttt{POPSIZE} = 15000$.

Un cop escollida la mida de la població és necessari seleccionar un valor per a la mida de torneig de la selecció. Com que no tenim gaire coneixement previ del problema, implementarem l'algorisme amb diferents valors, concretament amb $\texttt{T} \in \{15, 75, 125\}$.

Segons Bäck [Bäc96, p.114] diversos autors han proposat valors diferents per a la probabilitat de recombinació: DeJong va suggerir el valor $\texttt{P\_C} = 0.6$, posteriorment Grefensttete va proposar el valor $\texttt{P\_C} = 0.95$, finalment Schaffer *et al.* van plantejar com a millor selecció de l'hiperparàmetre els valors $\texttt{P\_C} \in [0.75, 0.95]$. D'altra banda, segons Goldberg [Gol89, p.111] estudis posteriors han suggerit que triar $\texttt{P\_C} = 1.0$ és millor quan els errors estocàstics de mostreig es redueixen gràcies a l'ús de mètodes de selecció més precisos.

En la nostra implementació usarem $\texttt{P\_C} = 1$ perquè gràcies a la grandària de la població i a l'aplicació del mètode de selecció per torneig amb mides considerables, la variància de la mostra es redueix. Per aquest motiu aquesta probabilitat de recombinació pot afavorir a la implementació de l'algorisme.

Per últim, triem el valor de la probabilitat de mutació. Recordem que aquest operador té la funció d'evitar la pèrdua de diversitat en la població, però és important destacar que valors grans poden provocar problemes. Segons Goldberg [Gol89, p.111] si la probabilitat de mutació és massa elevada l'algorisme genètic es converteix en un *random walk*, és a dir bàsicament es realitza una cerca aleatòria.

Segons Bäck [Bäc96, p.113], els autors anteriors proposen també diferents valors per aquest hiperparàmetre: DeJong proposa el valor `P_M` $= 0.001$, Grefensttete el valor `P_M` $= 0.01$ i Schaffer *et al.* suggereixen el rang `P_M` $\in [0.005, 0.01]$. Seguint aquestes propostes, implementarem l'algorisme amb els valors `P_M` $\in \{0.001, 0.01, 0.05, 0.1\}$

6.3.5 Resultats

En aquesta secció presentem els resultats obtinguts mitjançant la implementació de l'algorisme genètic discret descrit. Per a avaluar el funcionament del mètode s'han realitzat 25 execucions per a cada combinació d'hiperparàmetres de la selecció anterior. Com a criteri de parada per l'algorisme s'ha seleccionat la manca de millora del valor de fitness durant 100 generacions consecutives. La següent taula mostra l'individu amb fitness més baix obtingut en cada tanda de realitzacions en funció dels valors `T` i `P_M`.

`P_M`	`T`	x_0	φ	λ	μ	σ	δ	$fitness$	norma
0.001	15	15817.05	0.22016	1425.21	0.0	0.9765	8654.30	2598.6084	0.143724
0.001	75	15776.39	0.22016	1506.49	0.1562	1.1718	8983.59	2617.2021	0.134460
0.001	125	15805.62	0.21658	1517.01	0.1953	1.2695	8954.29	2627.6106	0.150480
0.01	15	15748.77	0.23447	1497.06	0.0	0.6654	8754.40	2574.2502	0.074176
0.01	75	15737.74	0.23447	1481.24	0.0	0.6834	8827.95	2575.4464	0.075770
0.01	125	15735.50	0.23089	1455.93	0.0	0.8300	8906.38	2582.5130	0.093623
0.05	15	15754.47	0.22185	1630.78	0.4880	1.8570	9770.96	2640.9275	0.142910
0.05	75	15701.20	0.24162	1528.46	0.0000023	0.5859	8865.49	2569.0496	0.039323
0.05	125	15704.37	0.23359	1475.30	0.0000089	0.7812	8909.13	2579.0609	0.079258
0.1	15	15668.92	0.22919	1798.302	0.7204	1.8761	10134.43	2647.7717	0.157096
0.1	75	15598.52	0.22380	1378.24	0.0181	1.3092	9164.87	2628.1154	0.135878
0.1	125	15719.95	0.22184	1422.68	0.0439	1.1268	8989.70	2615.9871	0.134525

Taula 6: Resultats per diferents valors de `P_M` i `T`. La columna *norma* indica la distància en norma L^2 dels paràmetres obtinguts respecte dels paràmetres $x^\star$, després d'aplicar la normalització Min-Max[8] per evitar que les diferències d'escala afectin el resultat. La fila en blau correspon als valors de $fitness$ i *norma* més petits que s'han obtingut.

[8]La normalització Min-Max transforma un valor $x \in [x_{min}, x_{max}]$ en un valor $x' \in [0, 1]$ a partir de la següent expressió $x' = \frac{x - x_{min}}{x_{max} - x_{min}}$

Notem que tot i que l'algorisme ha sigut capaç de trobar regions de baix fitness en l'espai de cerca, no s'ha aconseguit trobar l'òptim $x^\star$. Això ens porta a pensar que un algorisme genètic discret, sense estratègies elitistes que incorporin coneixement previ sobre els bons individus, pot presentar certes limitacions a l'hora de resoldre un problema d'optimització a un espai de cerca continu.

6.4 Cerca de l'òptim a partir d'un Algorisme Genètic Continu

En la següent secció implementem un algorisme genètic continu per analitzar si aquesta variant és més adequada per a la resolució d'aquest tipus de problemes.

6.4.1 Codificació

Per a la codificació d'individus de l'algorisme genètic continu seleccionem el mètode de codificació real. Tot i que es tracta d'un problema continu, treballem amb un espai discretitzat induït per la precisió finita del tipus de dada `double`, denotem aquest espai com $\mathcal{K}$. Aleshores tenim que l'espai de codificació és $S := \mathcal{K}$ i la funció morfogènica la identitat.

$$m : \qquad \mathcal{K} \longrightarrow \mathscr{K}$$

$$(u_{x_0}, u_\varphi, u_\lambda, u_\mu, u_\sigma, u_\delta) \longmapsto (u_{x_0}, u_\varphi, u_\lambda, u_\mu, u_\sigma, u_\delta)$$

6.4.2 Avaluació del fitness

L'avaluació del fitness dels individus es realitza de manera anàloga a la descrita a la secció 6.3.2. En aquest cas, tenim la següent funció fitness:

$$\text{fitness} : \qquad \mathcal{K} \longrightarrow \mathbb{R}_{\geq 0}$$

$$(u_{x_0}, u_\varphi, u_\lambda, u_\mu, u_\sigma, u_\delta) \longmapsto F(u_{x_0}, u_\varphi, u_\lambda, u_\mu, u_\sigma, u_\delta)$$

on F és la funció objectiu definida a l'equació 6.3.

6.4.3 Operadors

Per a la implementació de l'algorisme genètic continu escollim el mateix mètode de selecció que en el cas discret, és a dir la selecció per torneig per a minimització.

A causa de la grandària de l'espai de cerca seleccionem com a operador de recombinació el mètode de recombinació lineal estesa. Notem que aquest procediment depèn dels factors d'extrapolació de cada paràmetre: $p_{x_0}, p_\varphi, p_\lambda, p_\mu, p_\sigma, p_\delta$. En la nostra implementació considerarem el mateix valor per aquests hiperparàmetres, que denotem com p.

Com a mètode de mutació seleccionem la mutació uniforme. Aquest procediment depèn del rang de mutació de cada paràmetre. Sigui $\theta \in \{x_0, \varphi, \lambda, \mu, \sigma, \delta\}$, definim aquests hiperparàmetres de la següent manera:

$$\varepsilon_\theta = r \cdot (\theta^{(max)} - \theta^{(min)})$$

on $\theta^{(min)}$ i $\theta^{(max)}$ són els valors mínims i màxims permesos per al paràmetre θ, definits pels extrems del compacte $\mathscr{K}$ i r denota un hiperparàmetre anomenat *factor de dispersió*, comú a tots els paràmetres, que regula la proporció de la mutació respecte als rangs de paràmetres $\Delta_\theta = \theta^{(max)} - \theta^{(min)}$.

6.4.4 Selecció d'hiperparàmetres

En l'algorisme continu seleccionat tenim sis hiperparàmetres:

- `POPSIZE`: Mida de la població
- `T`: Mida de torneig
- `P_C`: Probabilitat de recombinació
- `P_M`: Probabilitat de mutació
- `P`: Factor d'extrapolació
- `R`: Factor de dispersió

Per als hiperparàmetres `POPSIZE`, `T`, `P_C` i `P_M` seleccionarem els mateixos valors que en la implementació de l'algorisme genètic discret. La seva tria es justifica a la secció 6.3.4.

L'hiperparàmetre `P` controla el rang de recombinació. Notem que en les primeres generacions, a causa de la uniformitat de la mostra inicial un valor massa gran pot provocar que es generin molts valors fora de l'interval de definició i, per tant, hi hagi una pèrdua de diversitat perquè molts individus tinguin valors extrems. Per aquest motiu optem per una definició adaptativa de `P`:

$$\texttt{P} = \begin{cases} \frac{1}{\text{generació}} & \text{si generació} \leq 20 \\ 1 & \text{si generació} > 20 \end{cases}$$

D'aquesta manera es comença amb valors petits per controlar el rang de recombinació i aquest va augmentant fins a arribar a la generació 20, on la cerca es concentra en una regió específica, aleshores es fixa el valor $\texttt{P} = 1$ per poder afavorir l'exploració de l'espai.

D'altra banda, per al factor de dispersió seleccionem el valor $\texttt{R} = 0.05$. Aquest valor s'ha determinat de manera empírica a partir de diverses proves experimentals.

6.4.5 Resultats

Després de realitzar múltiples execucions per a cada combinació d'hiperparàmetres de la selecció anterior notem que totes aconsegueixen trobar l'òptim $x^\star$. Això posa de manifest la flexibilitat del mètode respecte a la selecció d'hiperparàmetres. Per analitzar l'eficiència de l'algorisme en funció dels valors `T` i `P_M` representem l'evolució del fitness del millor individu de la població al llarg de les generacions.

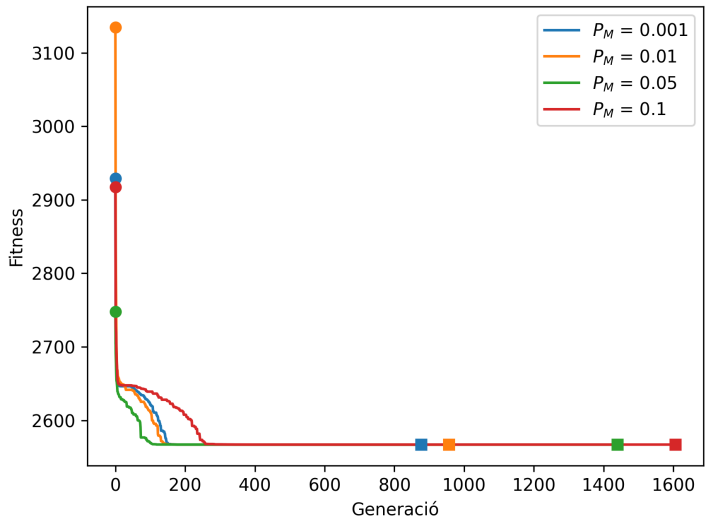
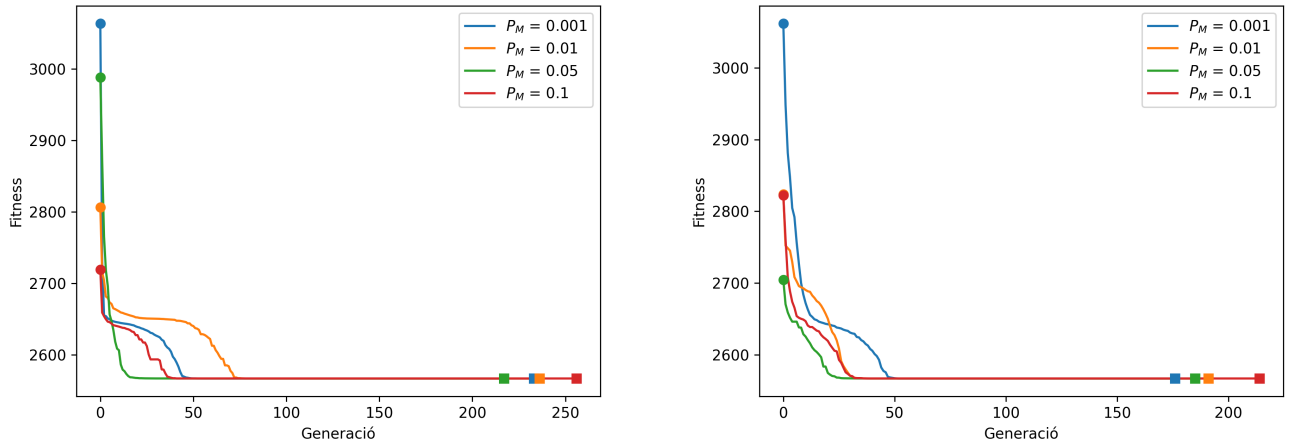


Figura 14: Evolució del fitness del millor individu de la població al llarg de les generacions per a diferents valors de $\texttt{P\_M} \in \{0.001, 0.01, 0.05, 0.1\}$ i $\texttt{T} = 15$. El marcador rodó indica el fitness més baix trobat en la població aleatòria inicial. El marcador quadrat indica l'última generació de l'algorisme, on es considera que l'algorisme ha convergit a $x^\star$ i, per tant, finalitza l'execució.

Observem a partir de la Figura 14 que per `T` $= 15$ el nombre de generacions que necessita l'algorisme per convergir és d'entre 878 i 1607. Això evidència, que un valor massa petit per a la selecció de torneig provoca que l'algorisme dediqui massa temps a l'exploració de l'espai de cerca, alentint la convergència a l'òptim. Fixada aquesta mida de torneig, el nombre de generacions necessàries per a la convergència a l'òptim varia en funció de la probabilitat de mutació `P_M`.

Els valors petits d'aquesta probabilitat afavoreixen a l'explotació. En conseqüència, notem que gràcies al seu caràcter explotador, les variants que utilitzen els valors 0.001 i 0.01, regulen millor el balanç entre aquestes dues estratègies i aconsegueixen convergir més ràpidament, en 878 i 957 generacions respectivament. D'altra banda, valors més grans, com 0.05 i 0.1, emfatitzen més encara el comportament exploratori i per tant la convergència es dona després d'un nombre molt més gran de generacions, 1440 i 1607 respectivament.



(a) `T` $= 75$ (b) `T` $= 125$

Figura 15: Evolució del fitness del millor individu de la població al llarg de les generacions per a diferents valors de `P_M` $\in \{0.001, 0.01, 0.05, 0.1\}$ i dos valors diferents de `T`. (La interpretació dels marcadors és la mateixa que en la Figura 14)

D'altra banda, a partir de la Figura 15, observem que variants amb pressió selectiva més elevada, com ara mides de torneig `T` $= 75$ o `T` $= 125$, regulen millor el balanç entre l'exploració i explotació. Aquests valors aconsegueixen accelerar la convergència al voltant de les 200 generacions, evitant alhora l'estancament en òptims locals.

Pel que fa a la probabilitat de mutació, l'evolució qualitativa del fitness del millor individu suggereix que el valor `P_M = 0.05` és el que millor equilibra aquestes dues estratègies. Cal destacar les configuracions d'hiperparàmetres que afavoreixen a una convergència més eficient, coincideixen amb les que obtenien solucions amb menor error en la implementació de l'algorisme genètic discret.

La següent taula descriu la informació quantitativa de les Figures 14 i 15, mostrant per a cada configuració d'hiperparàmetres el moment en què s'assoleix un fitness proper a l'òptim i la generació final de convergència.

`T`	`P_M`	Generació d'aproximació	Generació de convergència
15	0.001	172	878
15	0.01	172	957
15	0.05	146	1440
15	0.1	340	1607
75	0.001	52	234
75	0.01	77	237
75	0.05	29	218
75	0.1	41	257
125	0.001	52	177
125	0.01	38	192
125	0.05	32	186
125	0.1	37	215

Taula 7: Determina per les diferents seleccions dels hiperparàmetres `T` i `P_M` dos valors: La generació d'aproximació que es defineix com la generació a partir de la qual la diferencia del fitness de $x^\star$ i el valor obtingut és menor que 0.1 i la generació de convergència que representa la iteració on es considera que l'algorisme ha convergit i finalitza la seva execució.

Els resultats d'aquesta secció suggereixen que els algorismes genètics contínus amb codificació real són més adequats per a l'òptimització de paràmetres reals, ja que permeten trobar l'òptim reportat sense necessitat d'introduir estratègies elitistes que incorporin coneixament previ sobre l'espai de cerca. A més, aquesta variant permet una gran flexibilitat a l'hora d'escollir els hiperparàmetres.

6.5 Cerca de l'òptim a partir de l'algorisme *Simulated Annealing*

Tal com s'ha exposat a les seccions 6.3 i 6.4, dues variants d'algorismes genètics han trobat el mateix òptim: el GA discret amb estratègies elitistes i el GA continu. Això ens porta a pensar que aquest punt és una bona solució al problema 6.4. No obstant això, el nostre coneixement sobre l'espai de cerca és gairebé nul a causa de la seva complexitat. Aquest fet motiva a experimentar amb un mètode diferent, per comprovar si es troba la mateixa solució o pel contrari s'obté algun punt millor. En aquesta secció desenvolupem un algorisme *Simulated Annealing* per a la resolució del problema 6.4 i comparem els resultats obtinguts amb els de l'algorisme genètic continu.

Abans, però, cal destacar que la constant de Boltzmann k_b, tot i tenir un significat en el context físic, en l'àmbit computacional només reescala la temperatura. En conseqüència, sense pèrdua de generalitat, fixem $k_b = 1$ per a simplificar la implementació. A partir d'aquesta formulació, obtenim una equivalència entre el paràmetre de control c i la temperatura T. A partir d'ara, en aquesta aplicació, ens referirem a aquests termes anàlegs com a temperatura T.

6.5.1 Estructura de veïnat i mecanisme de generació

Notem que l'algorisme *Simulated Annealing* soluciona problemes d'optimització combinatòria, és a dir, amb un espai de solucions finit i discret. En la implementació d'aquest mètode prendrem com a discretització de l'espai de cerca $\mathscr{K}$, l'espai discretitzat induït per la precisió finita del tipus de dada `double` $\mathcal{K}$.

Sigui $x = (x_0, \varphi, \lambda, \mu, \sigma, \delta) \in \mathcal{K}$ definim la següent estructura de veïnat:

$$\mathscr{N}(x,T) = \begin{cases} \mathcal{K} & \text{si} \quad 1000 < T \\ \prod_{i=1}^{6}[x_i - 0.001 \cdot T \cdot \Delta_i,\, x_i + 0.001 \cdot T \cdot \Delta_i] & \text{si } 10^{-4} < T < 1000 \\ \prod_{i=1}^{6}[x_i - 0.1 \cdot T \cdot \Delta_i,\, x_i + 0.1 \cdot T \cdot \Delta_i] & \text{altrament} \end{cases}$$

on $\Delta_i := x_i^{(max)} - x_i^{(min)}$ denota la diferència entre el valor màxim i mínim de cada paràmetre, determinat pels extrems del compacte $\mathscr{K}$.

Com a mecanisme de generació se selecciona una distribució de probabilitat uniforme en $\mathscr{N}(x,T)$. Cal notar, però, que tot i que la distribució és uniforme en l'estructura de veïnat, a la pràctica, no totes les solucions veïnes tenen la mateixa

probabilitat de ser escollides. Per la definició de l'estructura de veïnat, hi ha solucions veïnes fora del compacte $\mathscr{K}$. Donat que l'algorisme ha de seleccionar solucions dins de l'espai de cerca, s'assigna a les proves generades fora del compacte els valors dels extrems. Aleshores els veïns fora $\mathscr{K}$ tenen probabilitat nul · la i els valors extrems una probabilitat efectiva superior a la resta d'estats.

6.5.2 Selecció d'hiperparàmetres

El *Simulated Annealing* amb procés de refredament geomètric té els següents hiperparàmetres:

- `L_max`: Nombre màxim de transicions per temperatura
- `La_max`: Nombre màxim d'acceptacions per temperatura
- `alp`: Factor de decreixement geomètric α
- `HT`: Proporció d'acceptació que indica temperatura alta χ_0
- `LT`: Proporció d'acceptació que indica temperatura baixa χ_f
- `T_0`: Temperatura inicial
- `T_min`: Temperatura mínima que pot assolir el sistema

La selecció dels hiperparàmetres, `L_max` i `La_max`, els quals controlen el nombre de transicions que es generen per a cada valor de temperatura, és un factor essencial perquè els algorismes de Metropolis assoleixin l'equilibri tèrmic. Com que estem treballant amb un problema complex, amb un espai de cerca gran, és important que l'algorisme avaluï un nombre elevat de solucions a cada pas. En conseqüència, se seleccionen els valors `L_max` $= 2000$ i `La_max` $= 200$.

Gràcies al teorema 5.15 s'estableix que una condició necessària per a la convergència a l'òptim és que la temperatura decreixi lentament. Segons van Laarhoven [LA87, p.61-62], per al refredament geomètric diversos autors proposen valors dins del rang $\alpha \in [0.5, 0.99]$. De manera empírica, a partir de diverses proves experimentals s'ha identificat que, per a la resolució d'aquest problema, és necessari una reducció extremadament lenta de la temperatura. En conseqüència, se selecciona el valor `alp` $= 0.99$.

Un cop determinats els hiperparàmetres més importants per a garantir la convergència a l'òptim seleccionem la resta a partir de l'experimentació amb diferents valors. Els hiperparàmetres seleccionats són: `HT` $= 0.8$, `LT` $= 10^{-6}$, `T_0` $= 100$, `T_min` $= 10^{-12}$.

6.5.3 Resultats

Després de realitzar diverses execucions s'observa que aquest algorisme també assoleix l'òptim $x^\star$. Per comparar l'eficiència dels dos mètodes que han obtingut $x^\star$ es calculen diferents mètriques de rendiment, a partir de 50 execucions dels algorismes SA i GA continu amb hiperparàmetres `T` $= 75$ i `P_M` $= 0.05$.

Mètode heurístic	**GA continu**	**SA**
Proporció d'èxit	0.22	0.64
Distància mitjana dels èxits a $x^\star$	$6.91 \cdot 10^{-6}$	$4.0875 \cdot 10^{-5}$
Temps d'execució [s]	70.71	55.37

Taula 8: Comparativa entre el GA continu amb `T` $= 75$ i `P_M` $= 0.05$ i el SA després de 50 execucions. S'entén per *èxit* qualsevol solució tal que la distància en norma L^2 entre la solució obtinguda i $x^\star$, un cop escalada amb Min-Max, és menor que 10^{-4}.

A partir de la taula 8, observem que, tot i que, l'algorisme genètic obté solucions més properes a l'òptim $x^\star$, aquest requereix un procés computacional més costos per dos motius: En primer lloc, l'algorisme genètic necessita un temps d'execució més elevat que el *Simulated Annealing*. En segon lloc, a partir de la proporció d'èxit observada s'estima la probabilitat que l'algorisme no s'estanqui en un òptim local, que és més gran en el cas del *Simulated Annealing*. Per tant, l'algorisme genètic necessita en mitjana de més proves per convergir al mínim $x^\star$.

Un cop determinades les estimacions dels paràmetres, per veure com s'ajusta el model 6.1 a les dades resolem l'equació diferèncial a partir de la rutina Runge-Kutta-Fehlberg 7-8 amb aquests valors.

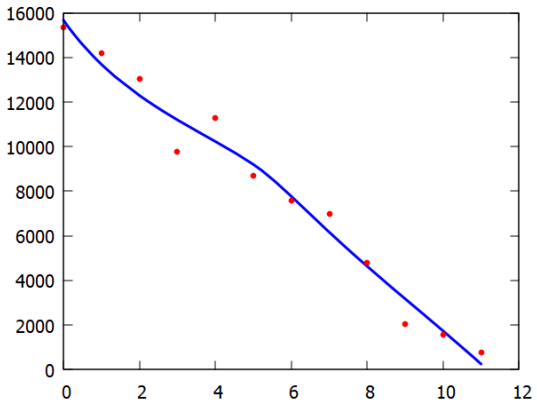


Figura 16: Representació de l'ajust del model a les dades. La funció blava representa la solució aproximada amb els paràmetres amb menor fitness obtinguts a partir de l'algorisme SA. Els punts vermells representen les dades $\pi(t, t = 0 : 11)$. La representació mitjançant els resultats obtinguts amb el GA continu és pràcticament idèntica per la proximitat de les estimacions dels dos mètodes.

7 Conclusions

Aquest treball ha proposat una metodologia per a l'ajust d'equacions diferencials ordinàries a sèries temporals curtes mitjançant tècniques d'optimització heurística, en particular a partir d'algorismes genètics, en versió discreta i contínua, i l'algorisme *Simulated Annealing*. Aquesta estratègia s'ha validat mitjançant l'estudi d'un cas real: la modelització de la dinàmica poblacional de gavines al Delta de l'Ebre entre els anys 2006 i 2016.

Els resultats obtinguts evidencien l'efectivitat dels mètodes heurístics en situacions amb dades limitades i paisatges amb múltiples òptims locals. La metodologia proposada ha permès reproduir la dinàmica general del sistema analitzat. No obstant a això, cal destacar que la manca d'informació pot dificultar la identificació dels paràmetres correctes. A causa d'aquesta incertesa, és important interpretar els resultats obtinguts amb actitud crítica.

L'algorisme *Simulated Annealing* ha destacat per la seva estabilitat en la resolució del problema, amb una proporció d'èxit elevada. A més, ha presentat el cost computacional més baix d'entre els algorismes aplicats. D'altra banda, l'algorisme genètic continu, tot i proporcionar una gran precisió en els ajustos, ha mostrat una menor estabilitat i un cost computacional més elevat en comparació amb el *Simulated Annealing*. Encara que s'ha observat que la selecció d'hiperparàmetres influeix a l'eficiència de l'algorisme en termes de regulació del balanç de l'exploració i explotació, totes les combinacions estudiades han aconseguit assolir l'òptim. Aquest fet posa de manifest.com la flexibilitat i robustesa de l'algorisme davant aquesta tria. Finalment, l'algorisme genètic discret ha mostrat un rendiment inferior en termes de qualitat de les solucions obtingudes. Això suggereix que la codificació binària i l'ús d'operadors de manipulació de bits no són adequats per a la resolució de problemes d'optimització amb paràmetres reals.

A partir d'aquest treball sorgeixen diverses futures línies d'estudi. En primer lloc, l'exploració de variants més avançades dels mètodes heurístics proposats, com l'ús estratègies adaptatives d'hiperparàmetres o operadors genètics i processos de refredament més complexos. En segon lloc, l'estudi d'estratègies híbrides que combinin mètodes de cerca estocàstics amb mètodes basats en gradient, per a la refinació de solucions a les fases finals. Com a extensió, també es podria desenvolupar altres mètodes heurístics per a comparar el seu rendiment amb els algorismes implementats. Finalment, per a obtenir una perspectiva més general dels resultats, seria interessant l'aplicació d'aquesta metodologia a problemes d'altres àmbits.

En conclusió, aquest treball presenta una metodologia que consolida les bases per a l'aplicació d'optimització heurística en l'ajust de models diferencials i posa de manifest el potencial dels mètodes heurístics com a eina d'optimització en contextos amb dades limitades.

Referències

[Als25] Lluís Alsedà. Comunicació privada. 2025.

[Bäc96] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. New York: Oxford University Press, 1996. ISBN: 978-0-19-509971-3.

[Bod03] Ulrich Bodenhofer. "Genetic algorithms: theory and applications". A: *Lecture notes, Fuzzy Logic Laboratorium Linz-Hagenberg, Winter* 2004 (2003).

[Bre+08] Daniel Brewer, Martino Barenco, Robin Callard, Michael Hubank i Jaroslav Stark. "Fitting ordinary differential equations to short time course data". A: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 366.1865 (2008), pàg. 519 - 544.

[ČLM13] Matej Črepinšek, Shih-Hsi Liu i Marjan Mernik. "Exploration and exploitation in evolutionary algorithms: A survey". A: *ACM computing surveys (CSUR)* 45.3 (2013), pàg. 1 - 33.

[DCM18] Daniel Delahaye, Supatcha Chaimatanan i Marcel Mongeau. *Simulated Annealing: From Basics to Applications*. Research Report hal-01887543. Available at HAL: `https://hal.science/hal-01887543`. ENAC - École Nationale de l'Aviation Civile, oct. de 2018. URL: `https://enac.hal.science/hal-01887543/document`.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989. ISBN: 978-0201157673.

[Haj88] Bruce Hajek. "Cooling schedules for optimal annealing". A: *Mathematics of Operations Research* 13.2 (1988), pàg. 311 - 329.

[Hol75] John H. Holland. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975.

[LA87] Peter J. M. van Laarhoven i Emile H. L. Aarts. *Simulated Annealing: Theory and Applications*. Springer Netherlands, 1987.

[LSZ24] Changhe Li, Changhe/Han Li (Shoufe) i Sanyou Zeng. *Intelligent Optimization*. Springer, 2024.

[Luk09] Sean Luke. *Essentials of Metaheuristics*. 2009.

[Met+53] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller i Edward Teller. "Equation of state calculations by fast computing machines". A: *The journal of chemical physics* 21.6 (1953), pàg. 1087 - 1092.

[Mit98] Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[MR11] Rafael Martí i Gerhard Reinelt. *The linear ordering problem: exact and heuristic methods in combinatorial optimization*. Vol. 175. Springer Science & Business Media, 2011.

[Oro+23] Daniel Oro, Lluís Alsedà, Alan Hastings, Meritxell Genovart i Josep Sardanyés. "Social copying drives a tipping point for nonlinear population collapse". A: *Proceedings of the National Academy of Sciences* 120.11 (2023), e2214055120.

[Siv+08] SN Sivanandam, SN Deepa, SN Sivanandam i SN Deepa. *Genetic algorithms*. Springer, 2008.

A Operadors bitwise en C aplicats a algorismes genètics

Els operadors *bitwise* són eines útils per a treballar directament amb els bits de dades a nivell binari. Gràcies a aquestes operacions podem modificar l'estructura binària d'un tipus de dada entera com per exemple `int`, `unsigned int`, etc. Això permet implementacions menys costoses computacionalment, gràcies a què no s'ha de convertir i emmagatzemar aquesta estructura de manera explícita. A continuació veiem quins són els operadors *bitwise* i alguns exemples de com funcionen. Alguns d'ells es poden relacionar amb connectors lògics de la lògica proposicional a partir de l'equivalència: 1 representa veritat i 0 representa fals.

- **Operador AND (&):** Compara dues representacions binàries A i B bit a bit. Si els dos bits són 1, aleshores assigna un 1, altrament assigna un 0. Actua com el connector lògic $\wedge$.

```
int a = 57        // 00111001
int b = 28        // 00011100
int c = a & b     // 00011000 (c = 24)
```

- **Operador OR (|):** Compara dues representacions binàries A i B bit a bit. Si almenys un dels dos bits és 1 assigna un 1, altrament assigna un 0. Actua com el connector lògic $\vee$.

```
int A = 57        // 00111001
int B = 28        // 00011100
int C = A | B     // 00111101 (C = 61)
```

- **Operador XOR (^):** Compara dues representacions binàries A i B bit a bit. Si els dos bits són iguals assigna un 0, altrament assigna un 1. Actua com el connector lògic $\veebar$.

```
int A = 57        // 00111001
int B = 28        // 00011100
int C = A ^ B     // 11100101 (C = 229)
```

- **Operador NOT ($\sim$):** Inverteix una representació binària A, és a dir canvia els bits 0 per 1 i els bits 1 per 0. Actua com l'operador lògic $\neg$.

```
int A = 57        // 00111001
int B = ~A        // 11000110 (C = 198)
```

- **Operador SHIFT RIGHT (>>):** Desplaça un nombre determinat de posicions els bits cap a la dreta i assigna el valor 0 a les posicions que han quedat lliures.

```
int A = 57        // 00111001
int B = A >> 2    // 00001110 (C = 14)
```

- **Operador SHIFT LEFT (<<):** Desplaça un nombre determinat de posicions els bits cap a l'esquerra i assigna el valor 0 a les posicions que han quedat lliures.

```
int A = 57        // 00111001
int B = A << 2    // 11100100 (C = 228)
```

A partir de combinacions d'aquests operadors es pot construir la recombinació d'un punt i la mutació bit-flip.

- **Recombinació d'un punt:** La recombinació entre dos cromosomes s'implementa construint dos cromosomes auxiliars formats per tot zeros excepte la part que s'ha de mantenir. Un cop es tenen aquests dos elements la recombinació es dona efectuant una operació OR entre aquests dos cromosomes. El següent exemple il · lustra aquest procediment:

```
int c  = 3       // Punt de tall
int P1 = 57      // P1: 001|11001   F1: 00111100
int P2 = 28      // P2: 000|11100   F2: 00011001

// (l - c) = 5

// Fill 1:

int AUX_1 = (P1 >> (l - c))      // 00000001
    AUX_1 = (AUX_1 << (l - c))   // 00100000

int AUX_2 = ~ 0                  // 11111111
    AUX_2 = 0 << (l - c)         // 11100000
    AUX_2 = ~ AUX_2              // 00011111
    AUX_2 = P2 & AUX_2           // 00011100

int F1 = AUX_1 | AUX_2           // F1: 00111100

// Fill 2:

AUX_1 = ~ 0                      // 11111111
AUX_1 = 0 << (l - c)             // 11100000
AUX_1 = ~ AUX_1                  // 00011111
AUX_1 = P1 & AUX_1               // 00011001

AUX_2 = (P2 >> (l - c))          // 00000000
AUX_2 = (AUX_2 << (l - c))       // 00000000

int F2 = AUX_1 | AUX_2           // F2: 00011001
```

- **Mutació bit-flip:** Per mutar un bit concret s'aplica una operació XOR amb el nombre en binari que té tot zeros excepte en la posició que es vol mutar, on té un 1. El següent exemple il · lustra aquest procediment:

```
int A = 57                       // 00111001
    aux   = (1<<3)               // 00001000
    A_mut = A ^ aux              // 00110001
```

B Repositori GitHub

Per facilitar la consulta del codi font, les dades i els scripts utilitzats en aquest treball, s'ha creat un repositori públic a GitHub. Aquest repositori permet compilar i executar els algorismes, així com reproduir les figures i els resultats que es mostren al llarg del document. Aquest conté:

- Implementació dels tres algorismes heurístics en llenguatge C.
- Scripts Bash per compilar i executar els algorismes.
- Scripts Python per al postprocessament i visualització dels resultats.
- Fitxers de dades resultants de múltiples execucions.
- Figures generades per a l'anàlisi.
- Un fitxer `README.md` que descriu detalladament l'estructura del projecte, els requisits d'execució i les instruccions per a compilar i reproduir els resultats.

El repositori es pot consultar i descarregar des de:

`https://github.com/tfg062025/TFG`