Escola Universitària
Politècnica de Mataró

Enginyeria Tècnica Industrial: Especialitat Electrònica Industrial

Sintonia de PIDs amb Heurístiques

MARC BAYÓN ROBLEDILLO
JOAN TRIADÓ I AYMERICH

TARDOR 2010

Resum

Aquest projecte tracta sobre la sintonia d'un controlador amb mètodes heurístics.

En primer lloc es fa un estudi sobre els mètodes heurístics a on es veuen tres tipus principals d'algoritmes o metaheurístiques de cerca: la cerca tabú, la recuita simulada i l'algoritme genètic.

A l'estudi es poden veure les característiques generals i una definició de la estructura general de cada mètode. Després de l'estructura s'anomena una sèrie d'aspectes generals que s'han de tenir en compte per a desenvolupar un programa a partir de les tres estratègies.

Per últim, es tracta el disseny i desenvolupament d'un algoritme genètic a través de Matlab per a trobar les constants Kp, Ki i Kd d'un control PID en un sistema en llaç tancat.

Resumen

Este proyecto trata sobre la sintonía de un controlador con métodos heurísticos.

En primer lugar se realiza un estudio sobre los métodos heurísticos dónde se pueden ver tres tipos principales de algoritmos o metaheurísticas de búsqueda: la búsqueda tabú, el recocido simulado y el algoritmo genético.

En el estudio se puede ver las características generales i una definición de la estructura general de cada método. Después de la estructura, se nombra una serie de aspectos generales a tener en cuenta para desarrollar un programa a partir de las tres estrategias.

Por último, se trata el diseño i desarrollo de un algoritmo genético a través de Matlab, para encontrar las constantes de un control PID en un sistema de lazo cerrado.

Abstract

This project deals about the use of heuristic methods for the tuning of a controller.

First of all, a study about heuristic methods is carried out. Three main algorithms are analyzed, i.e., tabu search, simulated annealing and genetic algorithm.

This study includes the quote of the characteristics and the backbone of the techniques as well as some issues to be considered in order to develop a custom program from those strategies.

At last, the process of creation of a genetic algorithm to use on PID controller tuning is explained. An own program is devised.

Índex

Llista de figures:

Llista de taules:

1. Introducció

1.1. Objecte del projecte

L'objecte del projecte és desenvolupar un estudi sobre la sintonia de PIDs mitjançant l'ús dels mètodes heurístics.

Aquest estudi està dividit en una primera part a on s'estableix una introducció teòrica a les heurístiques, i una segona part a on es duu a terme una aproximació pràctica.

Els algoritmes analitzats són:

- Cerca tabú
- Recuita simulada
- Algoritme genètic

1.2.Antecedents i motivacions

La proposta de projecte va ser presentada per en Joan Triadó, professor de l'assignatura de Regulació Automàtica.

Durant les classes de Regulació, es van estudiar els sistemes de control i la possibilitat d'afegir una realimentació, o *feedback*, per a millorar la resposta d'un sistema.

Més endavant es va observar que la implementació de diferents tipus de control ens permetien evolucionar en el tractament de la resposta per a arribar a obtenir més prestacions. Aquest projecte es basarà en un tipus concret: el control PID.

Mitjançant l'ús d'eines informàtiques, es va veure que es podien modificar còmodament els diferents paràmetres: Proporcional, Derivatiu i Integral.

D'altra banda, la metaheurística ens aporta una alternativa a l'hora de realitzar la sintonia dels esmentats paràmetres. Aquest sistema, intenta arribar a trobar la resposta òptima a partir de mètodes de tanteig. La proximitat entre la resposta òptima real i la resposta òptima obtinguda vindrà donada per la qualitat de l'algorisme i altres factors.

En definitiva, la metaheurística representa una opció a considerar a la indústria, a on s'ha de tenir en compte convenis a l'hora de relacionar els conceptes de qualitat, temps i diners per a escollir les opcions més viables.

Una de les eines amb les que hem treballat a la carrera ha estat MatLab. És per això que resulta interessant treballar amb aquest programari per realitzar tots els càlculs, i per analitzar les opcions que incorpora relacionades amb els algoritmes heurístics.

Com a conclusió, el que es proposa és veure les possibilitats que ofereixen els sistemes heurístics en comparació amb la sintonia PID estudiada a classe; tot plegat per tal de dotar de més capacitat de crítica al futur enginyer.

1.3.Abast

El projecte està centrat en la sintonia de PID; per tant, en primer lloc es farà un breu repàs de la teoria pertanyent a l'assignatura de Regulació Automàtica.
S'establiran unes bases de la regulació de processos (e.g. determinació de les parts indicades i identificació del model)
Es parlarà sobre el control de processos i es tractarà el control PID; i.e., estudi dels paràmetres i efecte sobre la resposta, anàlisi del mode.
Es farà una introducció sobre l'heurística fent una definició de caràcter general. Després es tractarà la metaheurística i s'analitzaran diverses tècniques o estratègies.
Tot seguit es desenvoluparà una aplicació basada en una de les metaheurístiques estudiades per a utilitzar-la en la sintonia d'un control PID.
Per a acabar, es realitzarà una comparació de la resposta obtinguda amb els algoritmes heurístics i de la desposta calculada empíricament i s'obtindran conclusions (e.g. avantatges i desavantatges).

Observacions:
La programació dels algoritmes per als experiments es realitzarà amb el llenguatge propi de Matlab.

2. Introducció als controladors PID

Abans de tractar amb la sintonia de controladors PID a partir dels mètodes heurístics, es farà un breu resum dels paràmetres relacionats amb aquest tipus de control.

2.1. El controlador PID

La connexió controlador és la que es mostra a la següent figura:

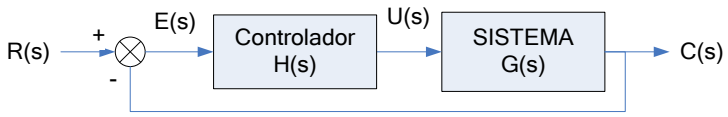


Figura 2.1: Configuració bàsica de la connexió d'un controlador PID en el domini de la freqüència

L'entrada del sistema controlat o referència està representada per la variable **R(s)** i la sortida o consigna, per la variable **C(s)**.

La entrada del controlador és l'error, **E(s)**, analitzat en el següent apartat. La sortida del controlador i entrada de la planta o sistema és **U(s)**.

Les equacions de transferència del controlador i del sistema són, respectivament **H(s)** i **G(s)**.

La equació del sistema en llaç tancat és la següent:

$$ftlt(s) = \frac{C(s)}{R(s)}$$

$$C(s) = G(s)H(s)E(s) = GH(s) \cdot E(s) = \quad E(s) = R(s) - C(s)$$

$$C(s) = GH(s) \;\; R(s) - C(s) \;\; = GH(s)R(s) - GH(s)C(s)$$

$$C(s) \;\; 1 - GH(s) \;\; = GH(s)R(s)$$

Per tant:

$$ftlo(s) = \frac{C(s)}{R(s)} = \frac{GH(s)}{1 - GH(s)} \tag{2.1}$$

El controlador està format pels elements *Proporcional*, *Integrador* i *Derivatiu*. Les possibilitats a l'hora de construir un control són PI, PD o PID.

Les equacions pertanyents són:

Proporcional

El paràmetre que es relaciona amb el proporcional és la constant **Kp**.

$$H_p(s) = K_P \tag{2.2}$$

Integrador

La variable relacionada amb l'integrador és el període **Ti**. La funció de transferència és:

$$H_I(s) = \frac{1}{T_i\ s} \tag{2.3}$$

Derivatiu

La variable relacionada és el període **Td**:

$$H_D(s) = \frac{1}{T_d\ s} \tag{2.4}$$

Un anàlisi més profund sobre la Figura 2.1 mostra que la configuració bàsica dels tres elements, PID és la següent:

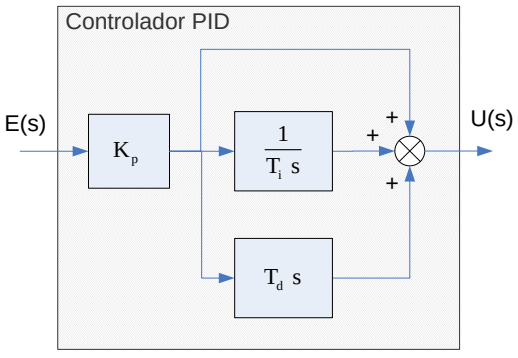


Figura 2.2: Configuració bàsica del controlador

Per tant la funció de transferència d'un controlador PID amb estructura bàsica és la següent:

$$U(s) = K_p\left[1+\frac{1}{T_i s}+T_d s\right]E(s)$$

$$H(s) \;=\; K_p\left[1+\frac{1}{T_i s}+T_d s\right] \tag{2.5}$$

Una altra forma de representar aquesta equació és treballar amb les constants Kp, Ki i Kd, a on:

$$Kp = Kp \tag{2.6}$$

$$Ki = \frac{Kp}{Ti} \tag{2.7}$$

$$Kd = Kp{\cdot}Td \tag{2.8}$$

Amb això, la equació 2.5 queda de la següent forma:

$$H(s) \;=\; K_p\left[1+\frac{1}{T_i s}+T_d s\right] = Kp+\frac{Ki}{s}+Kd{\cdot}s \tag{2.9}$$

[1]

2.2.Estudi de l'error

Un cop s'ha analitzat les funcions de transferència del PID es pot arribar a estudiar el seu efecte sobre l'error.

A partir de la Figura 2.2 es pot desenvolupar el següent:

$$E(s) = R(s)-C(s) = \left.\begin{cases} C(s) = U(s){\cdot}G(s) = H(s){\cdot}E(s){\cdot}G(s) = GH{\cdot}E(s) \\ U(s) = H(s){\cdot}E(s) \end{cases}\right\} = R(s)-GH{\cdot}E(s)$$

$$E(s)\,1+GH \;= R(s)$$

És a dir que:

$$E(s) = \frac{1}{1+GH} R(s) \tag{2.10}$$

A partir de les equacions 2.2, 2.3, 2.4 i 2.10 es pot arribar a les següents conclusions:

L'efecte d'un proporcional fa disminuir l'error, però no l'elimina, tal i com mostra l'equació 1.6:

$$E(s) = \frac{1}{1+K_p G} R(s) \tag{2.11}$$

Per tal d'eliminar l'error necessitem l'acció d'un integrador:

$$E(s) = \frac{1}{1+\frac{1}{T_i s} G} R(s) \tag{2.12}$$

Segons l'equació 1.7, l'error s'arriba a eliminar, però es torna molt lent per l'efecte de la integral.

L'aplicació de la component derivativa porta a:

$$E(s) = \frac{1}{1+T_d s G} R(s) \tag{2.13}$$

Amb això, el sistema adquireix velocitat però en contrapartida, es crea un sobre impuls, que en els pitjors dels casos pot desenvolupar en una resposta oscil·lant.

La correcta sintonització dels 3 elements, permetrà que els seus efectes treballin en harmonia i que la resposta sigui la millor (estable, ràpida i sense error en estat estàtic).

2.3. Control sobre un sistema de tercer ordre

Funció de transferència:

$$G(s) = \frac{1}{(s+1)(s+2)(s+3)} \tag{2.14}$$

Per a trobar els valors del PID s'ha fet servir el mètode de sintonia en llac tancat de Ziegler-Nichols per a un sistema continu, present als apunts de Regulació [1].

Els valors de Kpo i To es poden trobar fent un estudi de la freqüència de la planta. S'ha seguit el mateix mètode que [2]. Fent servir Matlab, el lloc geomètric de les arrels dóna el següent:

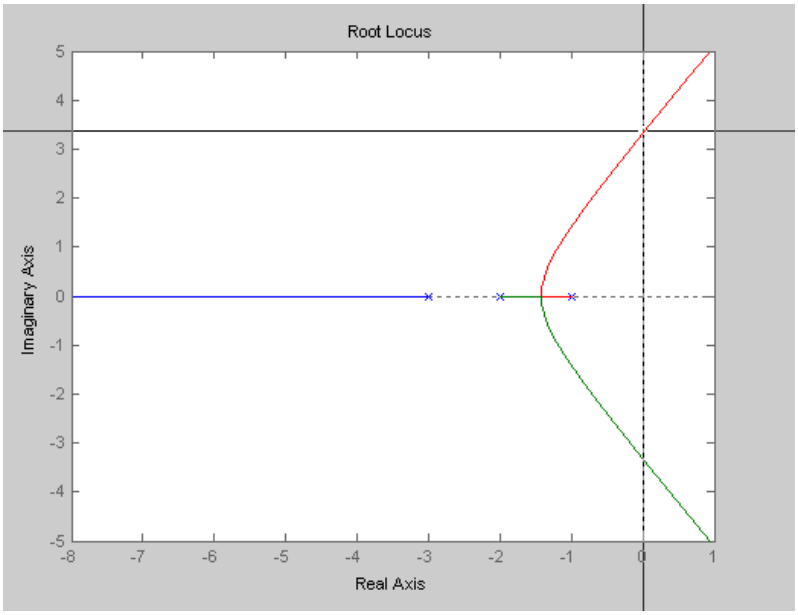


Figura 2.3: Lloc geomètric de les arrels

Els valors de freqüència i magnitud obtinguts són els següents:

$$\omega_o = 3,3369 \rightarrow T_o = \frac{2\pi}{\omega_o} = \frac{2\pi}{3,3369} = 1,8829 \text{ s} \tag{2.15}$$

Amb una freqüència associada de:

$$k_{po} = 61.0597 \tag{2.16}$$

Per tant, els valors de les constants esdevenen:

$$\boxed{Kp} = 0,6 \cdot Kpo = 0,6 \cdot 61,0597 = \underline{36,6358} \quad (2.17)$$

$$Ti = \frac{To}{2} = \frac{1,8829}{2} = 0,9415 \text{ s} \rightarrow \boxed{\text{Ki}} = \frac{\text{Kp}}{\text{Ti}} = \frac{36,6358}{0,9415} = \underline{38,9134} \quad (2.18)$$

$$Td = \frac{To}{8} = \frac{1,8829}{8} = 0,2354 \text{ s} \rightarrow \boxed{\text{Kd}} = \text{Kp} \cdot \text{Td} = 36,6358 \cdot 0,2354 = \underline{8,6229} \quad (2.19)$$

A la següent figura es pot comparar la diferència en les respostes del sistema quan només hi ha la planta i quan hi connectem el controlador:

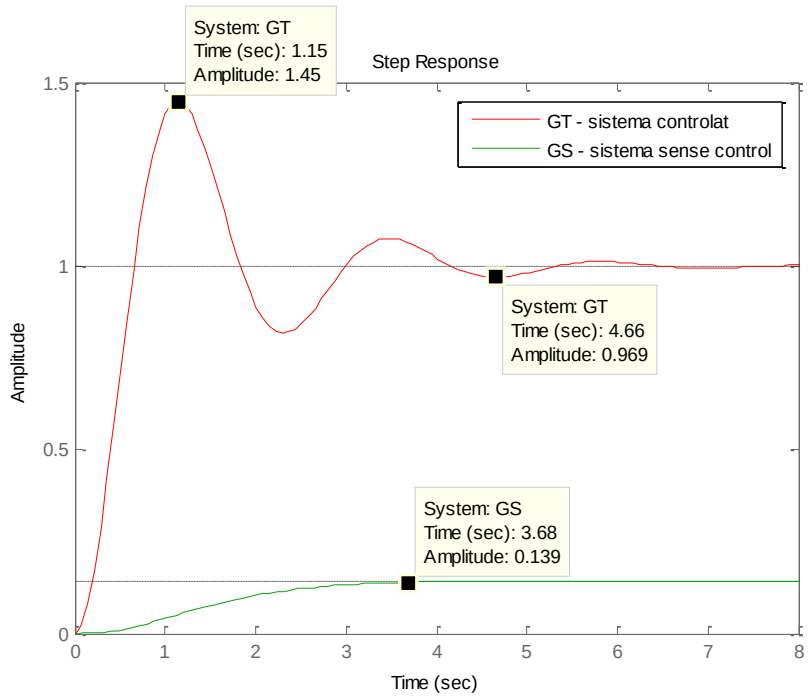


Figura 2.4: Sortida per al sistema amb i sense control

El sistema per si mateix té un error en estat estable d'unes 0.85 unitats i s'estabilitza al voltant dels 3.7 segons.

En canvi, el sistema controlat té un error en estat estable nul, però un sobre impuls de 0.15 unitats. La senyal s'estabilitza al voltant dels 4.7 segons (es considera que s'estabilitza quan la oscil·lació es troba dins d'un marge del 5% del valor en estat estable).

3. L'heurística

3.1. Els mètodes heurístics de cerca

Un mètode heurístic de cerca és un procediment que busca descobrir una solució factible molt bona, però no necessàriament òptima, per a resoldre un problema determinat.

La forma per a trobar aquesta solució es basa en el tanteig sobre la resposta a partir de diverses tècniques més o menys complexes. Generalment, el mètode heurístic consistirà en un algoritme iteratiu. S'escollirà un valor inicial, i a partir d'aquest s'anirà buscant solucions i s'escolliran les que millorin l'anterior cicle de la iteració, repetint el procés durant un interval de temps determinat.

A partir del paràgraf anterior es pot arribar a la conclusió de què el problema afegit a l'ús d'aquests mètodes recau en el desconeixement sobre la qualitat de la resposta obtinguda:

Quina és la diferència entre aquesta solució (*solució heurística*) i la resposta obtinguda a través del càlcul (*solució empírica*)?

La qualitat de l'heurística vindrà donada per la resposta a la pregunta anterior. Un bon mètode heurístic serà aquell que ens aproximi al màxim a la solució empírica (solució òptima). Més endavant es parlarà més a fons aquest aspecte a partir dels conceptes d'òptim local i òptim global.

Tornant a la definició, per a la resolució de cada problema tenim un mètode heurístic associat. Això significa que el disseny del mètode heurístic es duu a terme envers al tipus de problema (i.e. ad hoc segons "el llibre".)

El següent exemple intenta identificar el punt òptim (el màxim més alt) d'una funció polinòmica.

Es pot observar la diferència entre els dos tipus d'òptim.

Es coneix com a **òptim local**, qualsevol màxim de la població.

Quan un òptim local és el major dels òptims, dins d'un interval, es coneix com a **òptim global**.

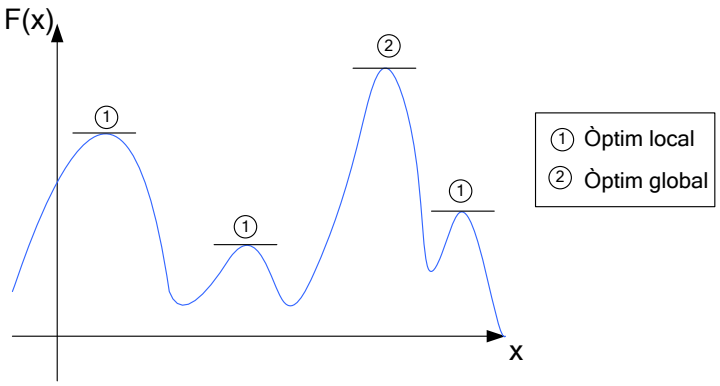


Figura 3.1: Exemple d'un procediment de cerca gradient. A partir de la llegenda es poden identificar els òptims locals i el global

Per tal de trobar els màxims de la funció a l'interval definit per la gràfica (Figura 3.1), s'hauria de recórrer al mètode matemàtic: derivar la funció, igualar a zero i trobar les solucions de la nova equació.

Es tindrà en compte la suposició de què els màxims del polinomi són els mostrats a la imatge ja esmentada.

El mètode heurístic pertanyent seguiria un **procediment de millora local** de cerca gradient. Aquest algoritme es coneix com a d'*escalada de muntanya*: s'inicia escollint un valor de prova inicial i continua escollint valors en un seguit d'iteracions, buscant a les mostres veïnes (representat com a valors de l'eix de les x), i escollint una solució millor a la de la iteració anterior (corresponent a l'eix de les ordenades). La Figura 3.2 representa el patró típic.

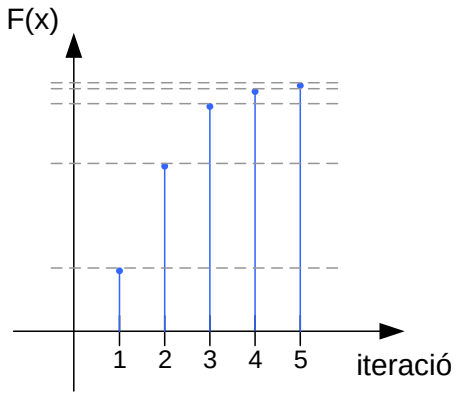


Figura 3.2: Relació de les iteracions amb la solució de prova d'un procediment de millora local

A cada iteració, la diferència entre la solució de prova actual i l'anterior es va reduint.

Això és degut a que a mesura que ens anem acostant al màxim, el pendent es va reduint.

El procediment va analitzant solucions de prova fins a arribar a la 5 iteració, a on l'accepta com a la solució final.

Aquest patró es podria ajustar a qualsevol dels màxims de la Figura 3.2.

S'ha de tenir en compte que l'algoritme es mou a cegues dins de la regió de mostres. Sobre la Figura 3.1 es pot veure clarament aquest pot arribar a qualsevol dels 4 màxims; però no serà capaç d'identificar-ne més de un, donat que cada cop que s'ha de desplaçar cap a un altre màxim, ha de superar una zona de descens.

El desavantatge d'aquest tipus de procediment és que només permet identificar un màxim. És a dir, l'òptim trobat dependrà del punt inicial: per a identificar l'òptim global haurà de situar-se, forçadament, a les seves proximitats.

Una possible solució, pot ser augmentar la probabilitat per a trobar l'òptim global: fer córrer l'algoritme diversos cops a partir de solucions de prova aleatòries. No obstant això, només tindria utilitzat quan l'interval de mostres és relativament reduït; un fet bastant poc atractiu quan s'augmenta el nombre de variables i/o es compliquen les regions factibles.

La metaheurística és capaç d'utilitzar la informació que va obtenint per a conduir la cerca cap a l'òptim global.

Permet crear un procés **capaç de sortir d'un òptim local** i de realitzar una cerca eficaç de l'òptim global.

3.2.La metaheurística

La metaheurística correspon a un tipus general de mètode de solució. Aquesta és capaç d'organitzar la interacció entre procediments de millora local i estratègies de més alt nivell.

A partir de diferents mètodes d'escapament, es podrà continuar analitzant solucions de prova menors a la de l'òptim local.

La metaheurística, és una heurística tan potent que és capaç d'adaptar-se fàcilment a la resolució d'un grup de problemes.

A continuació, es torna a analitzar el patró d'aquest tipus de procediments.

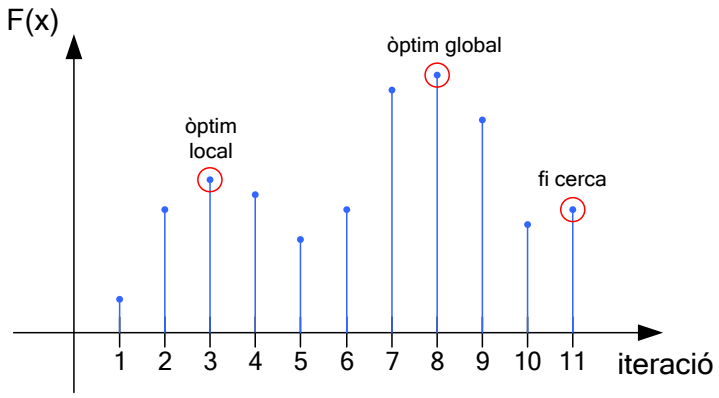


Figura 3.3: A la figura es pot observar el procés patró que segueix el procediment de cerca

El procediment és capaç d'arribar a trobar un òptim local a la iteració 3. Fins ara, de la mateixa manera que el mètode heurístic analitzat anteriorment.

A la iteració 4, s'ha aconseguit passar d'aquest òptim i continuar analitzant l'interval de mostres. Amb això s'ha conduit la cerca cap a l'òptim global, el qual ha estat trobat a la iteració 8.

En acabat, s'ha dut a terme una verificació: s'ha continuat l'exploració de les mostres fins a aturar-se a la iteració 11.

A partir del patró definit a la Figura 3.3 podem destacar el següent:

El procediment és capaç de trobar de forma relativament eficaç solucions bones. Això comporta una forma eficient d'abordar problemes grans i complicats.

D'altra banda, tot i seguir un procés de verificació, no existeix una garantia de que la solució es sigui realment la millor o n'estigui a prop de ser-ho.

En aquest cas, es podria intentar recórrer a un algoritme que proporcioni més garanties d'haver trobat l'òptim global.

Per consegüent, es pot constatar que les metaheurístiques busquen abordar problemes molt grans i complexes per a ser resolts amb algoritmes exactes.
[3], [4]

3.2.1. La metaheurística i els mètodes heurístics

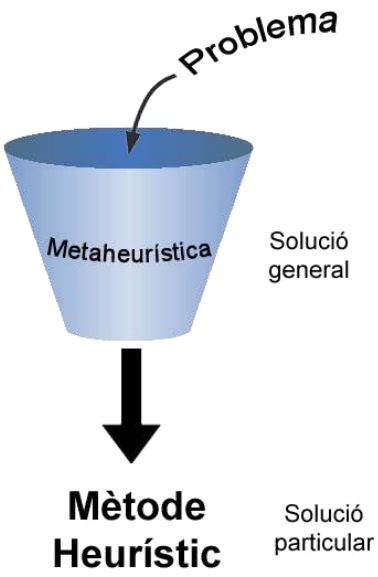


Figura 3.4: A partir d'un problema es pot fer servir la metaheurística per al desenvolupament d'un mètode heurístic adequat

La metaheurística s'utilitza com a base per al disseny de mètodes heurístics.

Proporciona una estructura general i uns criteris estratègics per al desenvolupament de mètodes específics. Amb això ja no cal començar a dissenyar des de 0.

A la Figura 3.4 es pot observar com a partir d'una solució general podem obtenir una solució particular per a la resolució d'un problema.

3.3. Tipus de metaheurístiques

Per tal d'aprofundir més en aquest tema, tot seguit s'analitzaran algunes de les metaheurístiques que existeixen. En concret es parlarà de la *cerca tabú*, del *temperat simulat*, i dels *algoritmes genètics*.

3.3.1. Cerca tabú

La cerca tabú és capaç de **conduir la cerca gràcies a l'ús d'una memòria**.

Aquesta metaheurística té un comportament similar a la pauta ja definida a l'apartat anterior:

Parteix d'un procediment per a trobar òptims globals que incorpora dins de la seva estructura un procediment de cerca local, normalment invocat com a subrutina.

Identifica un òptim local, llavors és capaç de desplaçar-se a les solucions de prova del veïnat (no necessita que cada solució sigui millor que l'anterior). Un cop trobat el punt al qual la solució de prova sigui millor que l'òptim local, es torna a aplicar el procediment de millora local per a trobar un nou òptim local.

A cada iteració, se selecciona el moviment que tingui el pendent positiu major, o en el seu defecte, el pendent negatiu menor. Aquesta dinàmica també és coneix com a *enfocament de l'ascens més empinat/descens més suau*.
[3]
La seqüència de valors obtinguts amb aquest procediment s'assimila a la mostrada a la Figura 3.3.

De tota manera, s'ha de tenir en compte que a l'hora de deixar un òptim local, el mateix algoritme pot retornar al mateix òptim i conduir així al procés en un bucle.

Per tal d'evitar aquest inconvenient, la cerca tabú prohibeix temporalment els moviments que puguin desembocar en el bucle (anomenats *moviments tabú*), emmagatzemant-los en una memòria, formant el que anomenarem *llista tabú*, la qual seguirà el comportament d'un *buffer* de dades. Si algun dels moviments tabú proporcionen una solució factible millor que la trobada, es podran utilitzar.

També existeixen altres tècniques per a perfeccionar la cerca tabú i incloure solucions no explorades, com ara la intensificació i la diversificació. En el primer cas, es pot intensificar l'exploració de certes parts de la regió factible quan s'ha identificat que la probabilitat de trobar bones solucions és elevada. En el segon cas, es forçarà l'entrada de la cerca a diferents zones de la regió factible [4],[5].

Esquema d'algoritme de cerca tabú:

Per començar, s'escollirà una solució de prova inicial factible.

El procés d'iteració seguirà un procediment de millora local per a definir els moviments factibles a les solucions veïnes, evitant l'ús dels presents a la llista tabú (si no generen una millor solució a l'actual).

S'estudiaran els moviments restants i s'escollirà aquell que proporcioni la millor solució.

La solució serà escollida per al següent cicle, encara que sigui pitjor a la solució de prova actual.

Afegirem aquest moviment a la llista tabú, eliminant-ne el més antic quan no hi hagi prou espai.

Finalment, haurem d'escollir un criteri de detenció de l'algoritme. Les possibles solucions són, entre altres:

- fixar el número d'iteracions
- limitar el temps

- fixar el número d'iteracions consecutives que no produeixin cap milloria al millor valor de la funció objectiu
- parar quan no es trobin moviments factibles a les proximitats de la solució de prova actual.

[4].

Desenvolupament del mètode heurístic

Com a metaheurística, la cerca tabú només constitueix la base de l'algoritme.

Existeix una sèrie de detalls a considerar en última instància:

- Quin és el procediment de cerca local a utilitzar?
- Com ha de definir el procediment l'estructura del veïnat que especifica quines solucions són veïnes immediates de qualsevol solució de prova (que es poden buscar a la mateixa iteració)?
- Quina és la forma en la qual els moviments tabú s'han de representar a la llista tabú?
- Quin moviment tabú s'ha d'afegir a la llista tabú a cada iteració?
- Quant s'ha de mantenir un moviment tabú a la llista?
- Quina és la regla de detenció?

[4]

L'obtenció d'un mètode heurístic adient a la resolució d'un problema dependrà de la correcta selecció dels paràmetres.

3.3.2. Recuita simulada

La recuita simulada és una metaheurística que també està basada en un procés de cerca i que permet escapar d'un òptim local.

Seguint el mecanisme de l'ascens de muntanya, es concentra en trobar el pendent més alt. Partint d'un determinat punt, l'algoritme tracta d'avançar cap a altres solucions de prova de forma aleatòria descartant algunes passes que portin a pendents descendents [4].

El fet de decidir aleatòriament les solucions de prova augmenta les possibilitats d'explorar la major zona possible de la regió factible i trobar així l'òptim global.

Donat que la majoria de passes són ascendents, l'algoritme tendeix a aproximar-se a les parts amb els òptims locals més alts. És per aquest motiu que de a mesura que avança la cerca, es van descartant cada cop més passos descendents.

D'aquesta manera, transcorregut un cert temps, el procés trobarà i ascendirà fins al pendent més alt el que portarà a trobar el major òptim.

Aquest mecanisme està basat en la analogia amb un procés de recuita física:

Quan es vol recoure un material o vidre, el primer que es fa és fondre'l a una temperatura molt alta.

Tot seguit es procedeix a refredar la substància lentament fins a arribar a un estat estable d'energia que posseeix les propietats físiques desitjables.

Treballant a una determinada temperatura (T) durant el procés, el nivell d'energia dels àtoms de la substància fluctua però tendeix a disminuir.
[4]

La recuita simulada va desplaçant-se de la solució de prova actual a una nova solució de prova (anomenada veí immediat) que es troba en el veïnat.

Per a la selecció del veí es farà servir la **regla del moviment:**

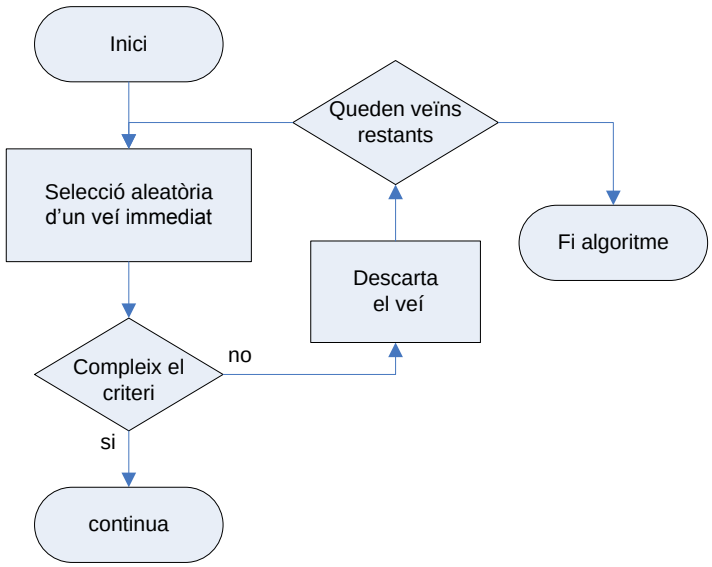


Figura 3.5: Estructura de la regla de selecció de moviment

Es selecciona un dels veïns immediats (candidat) de forma aleatòria.

Un cop escollit es comprova el criteri a partir de la funció objectiu. Si es compleix el criteri el candidat passa a ser la solució de prova actual. En cas contrari, es rebutja el candidat i es passa a escollir un nou candidat.

En el cas de que no quedin veïns restants, finalitza l'algoritme.

Per a determinar el criteri, identifiquem les següents variables:

- Zc – valor de la funció objectiu de la solució de prova actual

- Zn – valor de la funció objectiu del candidat a la pròxima solució de prova

- T – tendència d'acceptació del candidat actual quan no representa una millora sobre la solució de prova actual.

El valor de la tendència és correspon al de la temperatura en l'analogia de la recuita física.

Llavors, en funció dels esmentats paràmetres i de l'objectiu, que pot ésser minimitzar o maximitzar la funció objectiu

- Minimitzar:
    - Si $Z_c \geq Z_n$ s'accepta el candidat
    - Si $Z_c < Z_n$ s'accepta sota la probabilitat d'acceptació

- Maximitzar
    - Si $Z_n \geq Z_c$ s'accepta el candidat
    - Si $Z_n < Z_c$ s'accepta sota la probabilitat d'acceptació

La fórmula de càlcul de la probabilitat és el següent:

Minimització:

$$p\ \ acceptació\ = e^x \qquad \text{on } x = \frac{Z_c - Z_n}{T} \tag{3.1}$$

Maximització:

$$p\ \ acceptació\ = e^x \qquad \text{on } x = \frac{Z_n - Z_c}{T} \tag{3.2}$$

[4]

Aquest model matemàtic intenta seguir el comportament de la fluctuació de l'energia. Suposa que els canvis ocorren de forma aleatòria i a més només accepta alguns dels increments.

A l'hora de veure la evolució de l'algoritme, és important observar que la probabilitat de selecció dels passos descendents depèn de forma proporcional de la diferència de valors de la funció objectiu i de forma inversament proporcional del valor de la tendència T.

A la següent gràfica es pot observar com canvia el valor de la probabilitat per a diferents valors de T:

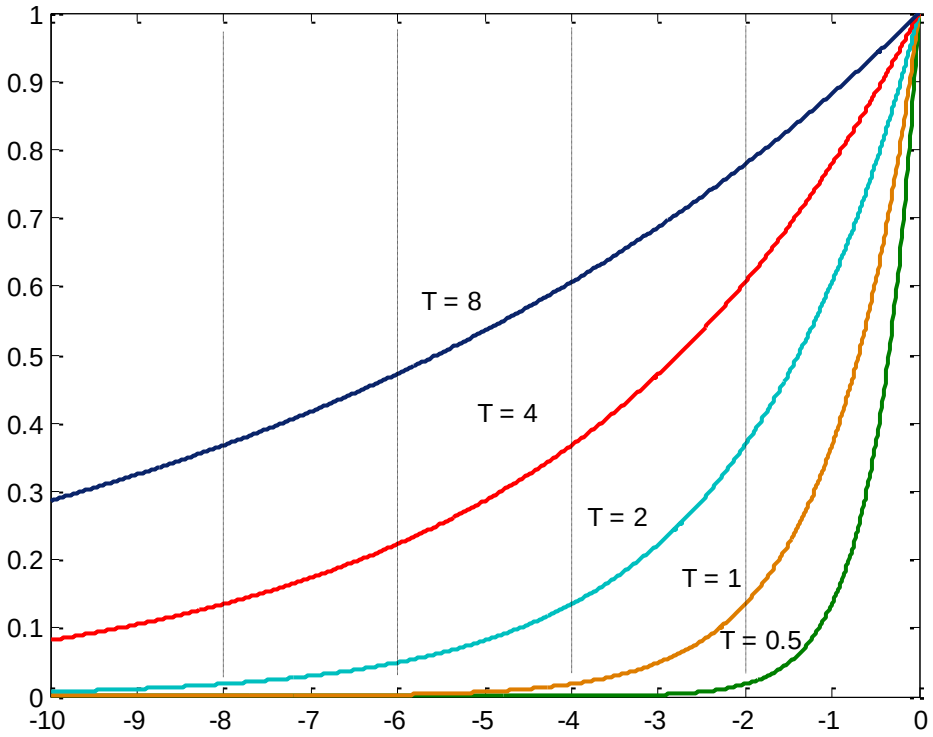


Figura 3.6: Representació dels valors de la probabilitat en funció de la diferència de x per a diferents valors de T (constants i variables)

A primer cop d'ull es pot veure com la selecció de passos poc descendents té una probabilitat major que la selecció de passos molt descendents.

Un anàlisi més exhaustiu mostra com la probabilitat és nul·la a partir de cert valor de x, i que aquest valor va augmentant en amplitud (és més negatiu) conforme augmenta T.

Segons Hillier [4], "El valor de T controla el grau d'aleatorietat del procés per a permetre passos descendents".

La recuita simulada treballa amb valors de T elevats al principi, el que permet que s'avanci en direccions gairebé aleatòries. Però el valor de T disminueix gradualment a mesura que

van passant iteracions. Això implica que en cada iteració es va escurçant l'elecció de passos descendents.

Esquema de la recuita simulada

L'algoritme comença escollint una solució de prova inicial. És important començar amb una solució de prova factible.

A partir d'aquest punt comença la iteració. A partir de la regla de selecció de moviment, s'escollirà el candidat a ser la pròxima solució de prova. En el cas de que no hi hagi solucions de prova restants, finalitza l'algoritme (veure Figura 3.5).

Amb la nova solució de prova, cal verificar el valor de T i, si es compleixen les condicions adequades, disminuir-ne el valor.

El següent pas és repetir el procés (reiniciar la iteració) sempre que no es compleixi la condició de detenció.

La regla de detenció es dóna per a l'últim valor de T, quan s'acaben les iteracions o no queden veïns restants.

Desenvolupament del mètode heurístic

Les característiques finals que donaran nom i cognoms al mètode són:

- Com es selecciona la solució de prova actual?

- Quina és l'estructura del veïnat? I per tant, quins són els veïns immediats d'una solució de prova?
- Quin és el criteri de selecció aleatòria (de solucions veïnes) a utilitzar en la regla de moviment?
- Quin és el programa de T a utilitzar?
- Quin és el valor inicial?
- Quins valors va adoptant?
- Cada quantes iteracions cal variar T?

[4]

Exemple

Per tal d'il·lustrar el funcionament del temperat simulat, se seguirà l'exemple de programació no lineal de Hillier [6]

L'objectiu és maximitzar la funció de la equació 3.3 per a l'interval 0 a 31.

$$f(x) = 12x^5 - 975x^4 + 28.000x^3 - 345.000x^2 + 1.800.000x \qquad (3.3)$$
$$\text{per } 0 \leq x \leq 31$$

A partir de la gràfica, es determina que els màxims de la funció (òptims locals) són a: x = 5, x=20 I x=31.

En aquest cas el màxim global és x=20.

Definició de l'algoritme (veure Figura 3.7)**:**

Maximitzar f(x) subjecte a $L \leq x \leq U$ a on L correspon a 0 i U correspon a 31.

Per a la solució inicial es farà servir el valor mig entre els límits:

$$x = \frac{U - L}{2} = \frac{31 - 0}{2} = 15,5 \tag{3.4}$$

En quant a la selecció del veïnat, es treballarà amb la generació aleatòria de números dins de la regió especificada.

Els veïns immediats es troben dins de l'interval determinat per sigma. Aquest mètode dona importància a les solucions que es troben relativament a prop (a 3 desviacions estàndards de l'actual) però també permet desplaçar-se a les més allunyades:

$$\sigma = \frac{U - L}{6} = \frac{31 - 0}{6} = 5,167 \tag{3.5}$$

Llavors, els candidats es troben a partir de la generació aleatòria tenint en compte una distribució normal centrada a 0 i amb desviació estàndard sigma.

$$x = x + N(0, \sigma) \tag{3.6}$$

Per últim cal seleccionar el programa de la temperatura. S'ha decidit treballar amb 5 valors de T que duraran 5 iteracions. Els valors de temperatura corresponen a:

$$T_1 = 0.2 Z_c \tag{3.7}$$

$$T_i = 0.5 T_{i-1} \qquad (\text{per a } i = 2,...,5) \tag{3.8}$$

S'obté el valor de la imatge de la solució de prova inicial, a partir de les equacions 3.3 i 3.4 :

$$Z_c = f(15,5) = 3.741.121 \tag{3.9}$$

Amb això, es poden tornar a editar les equacions 3.7 i 3.8 per a obtenir els resultats de la Taula 3.1:

T1	748224,20
T2	374110,00
T3	187056,03
T4	93528,02
T5	46764,01

Taula 3.1: Valors del programa de temperatura

A partir d'aquí ja es pot realitzar el programa. La implementació a codi s'ha dut a terme sobre Matlab. L'algoritme es pot trobar annexat al final del projecte.

El valor de la solució inicial és conegut (equació 3.4) i el valor de T1 també (Taula 3.1).

Llavors a partir de l'equació 3.6, el primer candidat generat és:

$$x = 20,924 \tag{3.10}$$

Amb valor d'imatge:

$$Z_n = f(x) = 4.360.198,20 \tag{3.11}$$

Donat que Zn és major que Zc, s'accepta el candidat que passa a ser la nova solució de prova.

En la següent iteració, el candidat és:

$$x = 18,05 \tag{3.12}$$

$$Z_n = f(x) = 4.246.632,18 \tag{3.13}$$

Ara Zn és menor que Zc i per tant cal estudiar la probabilitat, a partir de la equació 3.2:

$$p = e^{x} = \left\{ x = \frac{Z_n - Z_c}{T} = \frac{4.246.632,18 - 4.360.198,20}{748224,2} = -0.1518 \right\}$$

$$p = e^{-0.1518} = 0.8592 \tag{3.14}$$

En aquest cas, el candidat té una probabilitat del 85,9% de ser escollit. A partir de la comparació de p amb un número aleatori es troba que el número és inferior a p i per tant el candidat s'escull com a nova solució de prova.
L'algoritme continua i al final s'arriba a obtenir el següent:

Iteració	T	x	f(x)
0		15,50	3741120,69
1	748224,14	20,94	4360198,20
2		18,05	4246621,18
3		20,33	4395041,44
4		22,22	4162422,00
5		22,35	4133544,45
6	374112,07	21,07	4346660,57
7		21,57	4282591,88
8		20,72	4375860,73
9		19,53	4390225,88
10		**20,16**	**4398829,58**
11	187056,03	17,82	4211301,41
12		21,05	4348296,19
13		20,24	4397443,39
14		21,65	4270509,12
15		20,55	4386003,16
16	93528,02	17,26	4114721,14
17		19,16	4369623,31
18		19,59	4392512,01
19		19,35	4381756,07
20		20,39	4393177,61
21	46764,01	18,91	4349794,08
22		19,62	4393564,40
23		20,23	4397641,22
24		18,24	4273927,30
25		20,19	4398319,46

Taula 3.2: Progressió de la recuita simulada

Quan finalitza, es busca la solució amb la imatge més gran. En aquest cas, la solució x=20,16, trobada a la iteració 10.

La solució trobada per l'algoritme és molt pròxima al valor de l'òptim global calculat abans.

L'algoritme comentat anteriorment segueix la següent estructura:

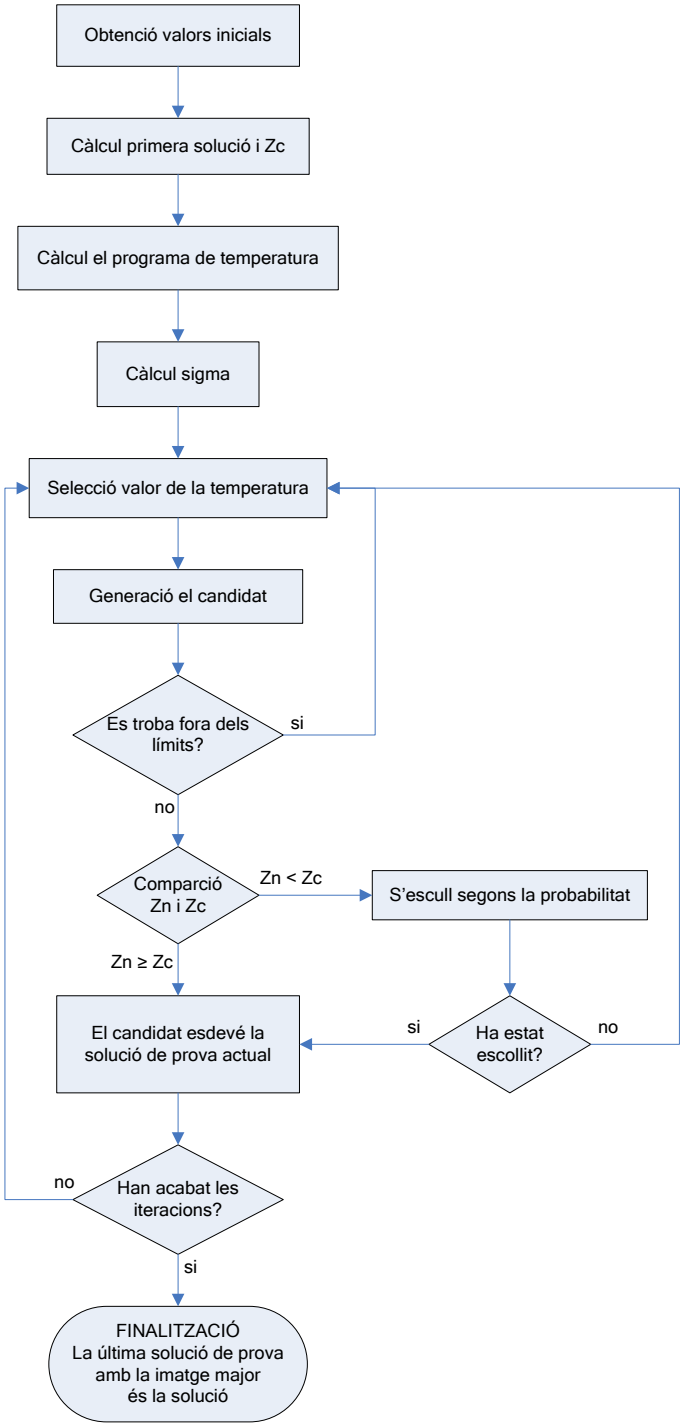


Figura 3.7: Esquema de l'exemple de l'algoritme de recuita simulada

3.3.3. Algoritme genètic

Aquest mètode és una alternativa eficaç per a explorar diverses zones de la regió factible ja que permet evolucionar cap a aquelles més favorables.

Es basa en la teoria de biològica de l'evolució enunciada per Charles Darwin, al S.XIX, la *selecció natural*.

La població d'una espècie presenta variacions individuals per a cada membre. Les variacions avantatjoses comporten una millor adaptació al medi, i per tant, més probabilitat de perpetuar-se a les següents generacions. Aquest concepte es coneix com a la *supervivència del més fort*.

En el camp de la genètica, això es tradueix com a que cada generació hereta les característiques de l'anterior. Els gens fan que els fills adquireixi part dels cromosomes dels pares. Com millors siguin els gens, més probabilitats de sobreviure tindran els fills.

A mesura que van passant generacions, la població tendeix a estar formada pels individus més ben adaptats.

A més, la població és capaç d'evolucionar gràcies a l'aparició de noves característiques gràcies a la mutació genètica.

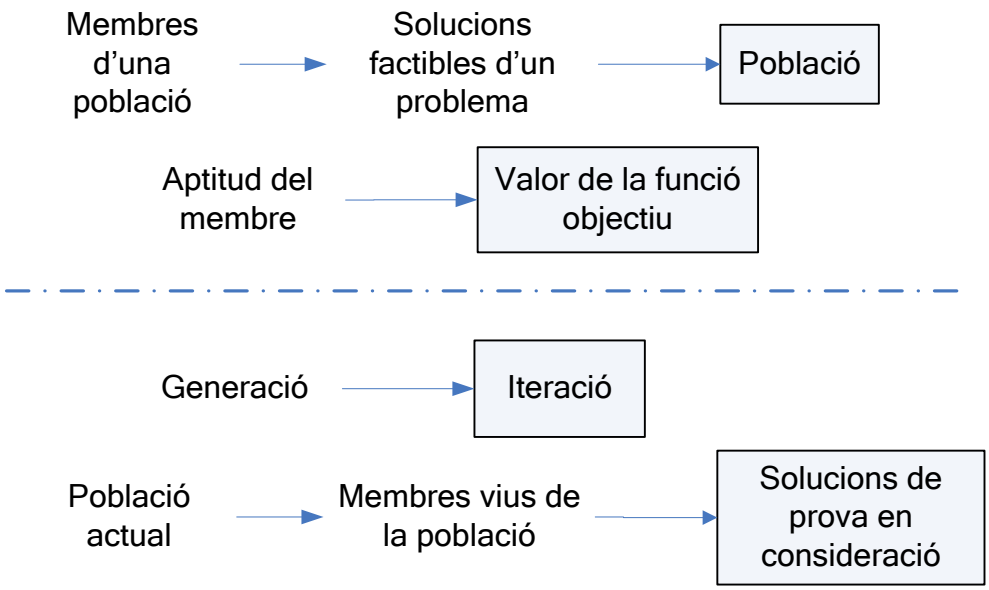


Figura 3.8: Equivalències entre termes de la genètica i l'optimització

En aquest cas, durant cada iteració no tenim una única solució de prova sinó que considerem tota la població. A generació els fills es converteixen en pares, els quals són aparellats aleatòriament per a la següent.

A l'hora de determinar la qualitat solució, es farà una assignació de probabilitat, a on rebran una bonificació els membres més joves; en especial, aquells més aptes. Passades unes iteracions, els membres de l'espècie han evolucionat, el que ha permès la generació d'una població millorada de solucions de prova.

Quan finalitza l'algoritme, acaba amb la solució de prova millor considerada.

Esquema d'algoritme genètic

Tot seguit es presenten les bases per a l'elaboració d'un algoritme genètic:

En primer lloc es començarà escollint la primera generació de la població de solucions de prova..que serà un número parell de solucions de prova factibles. El criteri pot ésser aleatori, però el número serà parell.

A partir del valor de la funció objectiu, s'avaluarà l'aptitud de cada membre.

En quant a cada iteració, se seguirà un procés aleatori que propiciï la selecció dels als membres més aptes de la població actual, que seran assignats com a pares, i aparellats (de forma aleatòria) en un nombre parell.

Els fills estaran formats per una barreja de les característiques dels pares. Es tindrà en compte un factor de mutació que determinarà la freqüència amb que s'aplicaran variacions sobre aquestes característiques.

Un cop trobats els fills, aquest passen a ser els membres de la població actual per a tornar a repetir el procés d'iteració.

Un cop es compleixi la condició d'aturada, finalitza l'algoritme.

La millor solució de prova de la població esdevé la solució.

Desenvolupament del mètode heurístic

Per a desenvolupar el mètode heurístic, cal respondre a més a més a les següents preguntes:

- Quina ha de ser la mida de la població?
- Quin és el criteri de selecció dels membres de la població actual per a identificar als pares?
- Quines han de ser les característiques heretades dels fills?
- Com s'han d'aplicar les mutacions sobre els gens?
- Quina és la regla de detenció?

La combinació de les diferents respostes serà la que acabarà de donar forma a l'algoritme. Com sempre, existeixen diferents respostes a cada qüestió. La elecció de la adequada s'haurà de fer tenint en compte el problema amb el qual es vol treballar.

3.3.4. Algoritme híbrid

Una de les opcions que apareix a partir del coneixement de diverses metaheurístiques és la combinació de les estratègies que ofereix cadascuna.

Aprofitar els avantatges particulars de cada metaheurística per a desenvolupar un algoritme híbrid és una bona forma de millorar el seu comportament.

Hillier [4] proposa dos exemples d'algoritme híbrid.

En el primer algoritme híbrid, utilitza la oscil·lació estratègica de la cerca tabú per a ajustar els valors de la temperatura (en la recuita simulada) i variar T en un sentit o altre a través dels nivells a on es troben les millors solucions.

En el segon cas, es combina la estratègia de la llista de candidats de la cerca tabú a la regla de selecció de moviment de la recuita.
L'objectiu és identificar múltiples veïns per a intentar trobar un moviment amb millora abans d'aplicar la regla autoritzada, per a acceptar o rebutjar el candidat actual a ser la pròxima solució de prova. Aquesta solució pot proporcionar millores significatives.

4. Aplicació de l'heurística a la sintonia PID

L'objectiu d'aquest projecte ha estat utilitzar la heurística per a optimitzar els valors d'un control PID. En aquest capítol es desenvolupa tot el procés aplicat al control PID

4.1. Funció objectiu. Criteris d'error

Per a determinar la funció objectiu s'han utilitzat 3 criteris d'error diferents [8]:

Criteri ISE:

$$J_{ISE} = \int_0^t \left[r(t) - y(t) \right]^2 dt \tag{4.1}$$

Criteri IAE:

$$J_{IAE} = \int_0^t \left| r(t) - y(t) \right| dt \tag{4.2}$$

Criteri ITAE:

$$J_{ITAE} = \int_0^t t \cdot \left| r(t) - y(t) \right| dt \tag{4.3}$$

L'objectiu de l'algoritme és minimitzar el valor de qualsevol de les anteriors equacions.

La implementació dels criteris d'error s'ha realitzat amb Matlab. Les sentències són:

Criteri	Codi
IAE	`J=sum(abs(e))`
ITAE	`J=sum(t.*abs(e))`
ISE	`J=sum(e.*e)`

Taula 4.1: Implementació dels criteris d'error a Matlab

4.2.Disseny d'un algoritme genètic

4.2.1. Introducció

La pauta comentada al capítol 2 s'ha utilitzat com a base en la generació del mètode heurístic que segueixi l'algoritme genètic.

L'esquema general que del programa correspon a la Figura 4.1. A partir de la imatge es poden observar tots els passos a seguir.

El bloc número 1 correspon a la creació de la primera generació. Aquí es crea la primera generació de cromosomes o solucions de prova.

Al bloc número 2 es determina l'aptitud de les solucions de prova. Per a això s'ha de trobar les constants del controlador, generar la funció de transferència i per últim estudiar tot el sistema. A partir de diferents criteris d'error s'emmagatzema el valor del criteri en una variable amb la qual es determina l'aptitud de cada solució de prova en particular.

La selecció de membres (bloc número 3) es divideix en la identificació dels membres elit i la dels pares.

La fase 4 tracta la reproducció; la creació de la següent generació de solucions. S'aparellen els individus que han estat seleccionats com a pares i es generen els fills per diferents tècniques. En acabar la nova generació passa a ser la generació actual.

L'últim punt és la iteració a la qual es revisa si es compleix la condició de finalització. Aquí és a on es porta el control de totes les generacions.

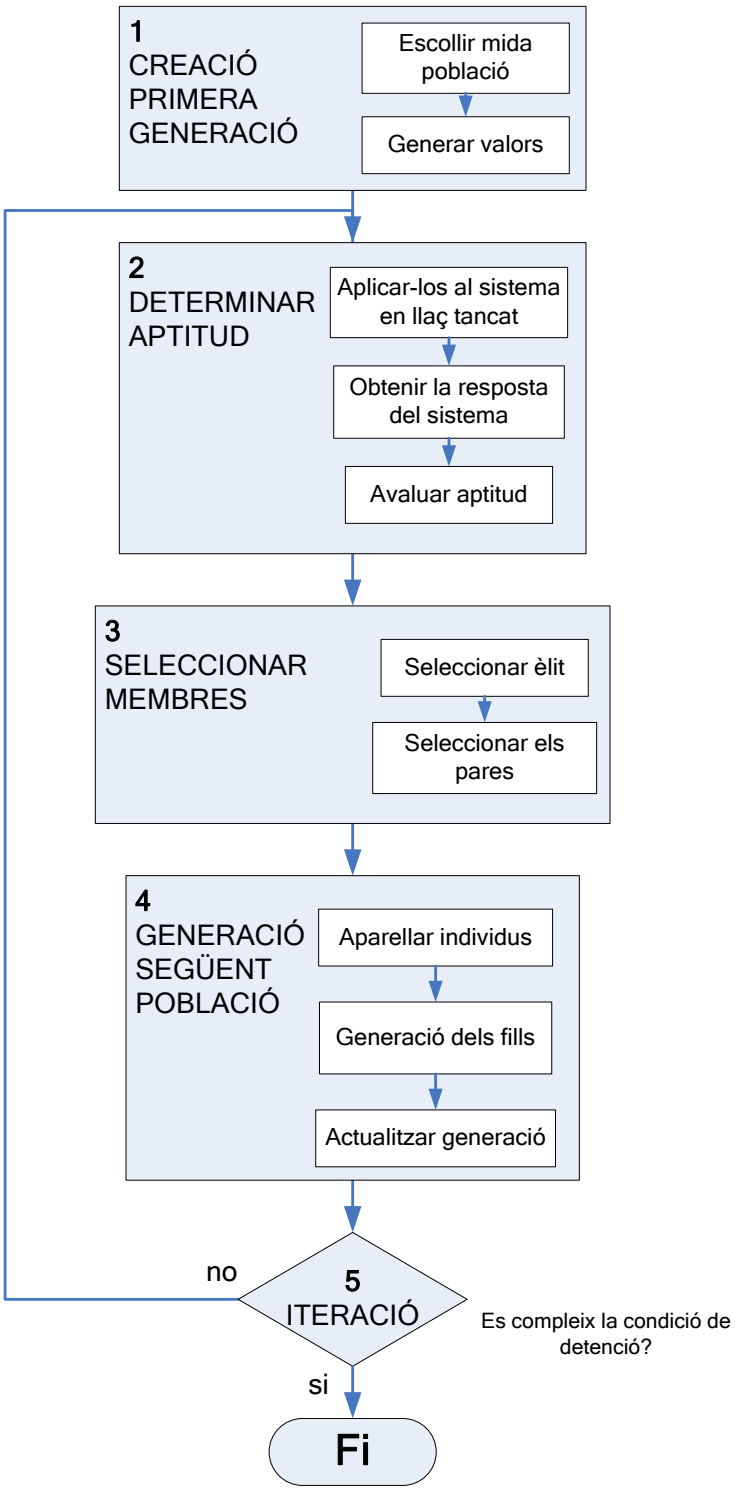


Figura 4.1: Diagrama de flux de l'algoritme genètic

A continuació es tracta cada pas amb més detall. A l'apartat 4.3 hi ha una llista amb les relacions dels fitxers amb el codi i les fases. El codi pertanyent a cada fitxer es troba annexat al final del projecte.

4.2.2. Condicions d'entrada de l'algoritme

Paràmetres de la planta:

- Funció de transferència

La funció de transferència es pot introduir ingressant el nom de la variable.
La variable ha de ser de tipus *tf* (funció de transferència), i l'ordre del numerador mai pot ésser superior al del denominador.

Paràmetres d'inicialització:

- Mida de la població

La mida de la població correspon al número de solucions amb el qual es treballarà. El valor ha de ser un enter positiu i parell.

A l'hora de decidir un valor adequat per a la mida de la població hi ha diverses opinions envers si ha d'ésser un valor gran o petit per obtenir millors resultats [2], [8]. En aquest cas s'ha decidit assignar un valor de 50 membres per defecte donat que es treballa amb només 3 valors per solució de prova.

- Límits per als valors del control

Els límits per als valors del control especifiquen els marges de generació de les constants Kp, Ki i Kd. Els valors per defecte s'han determinat entre -100 i 100 seguint els passos de Griffin [2].

Paràmetres de l'algoritme:

- Mida de la elit

La mida de la elit correspon al número de membres que passen directament a la següent generació (veure apartat 0). La elit ha de ser un valor enter positiu i menor que la mida de la població.

Quan se li assigna un valor de 0, la generació de fills només es farà mitjançant l'encreuament i la mutació.

- Fracció d'encreuament

La fracció d'encreuament és un paràmetre que controla la fracció de la reproducció corresponent a l'encreuament.

- Probabilitat de mutació

Indica la probabilitat d'una solució de prova de rebre mutació. El rang de la probabilitat és de 0% a 100%.

- Tipus de criteri d'error

Indica la funció objectiu a utilitzar. Els mètodes que es poden escollir corresponen a les equacions 4.1, 4.2 i 4.3.

- Número d'iteracions de l'algoritme

El criteri d'aturada de l'algoritme es basa en limitar el número de iteracions o generacions, i ha de ser un enter positiu.

4.2.3. Creació de la primera generació

Tipus de codificació

En primer lloc s'ha de decidir el tipus de valor amb el qual es vol treballar. Existeixen diferents opcions, com ara treballar amb el valor codificat a binari o en decimal [6], representació en coma fixa o en coma flotant. Reeves realitza una breu descripció dels avantatges i els inconvenients a l'hora d'escollir un conveni o altre [8].

Per a aquest projecte s'ha decidit treballar amb codificació decimal i amb variables de tipus *double*.

Treballar amb lògica binaria comporta una pèrdua de resolució en la conversió binari a real per a cada operació. La conversió implica més lentitud.
A més, treballar en coma flotant proporciona més resolució.

Mètode de generació

La creació de la primera generació és una part molt important de l'algoritme. És aquí a on neixen totes les possibilitats per a trobar un bon òptim.

De tots els diferents mètodes de generació aleatòria, en aquest cas s'ha decidit seguir els passos de la guia de referència de l'eina Genetic Algorithm de matlab [9] i de Ibrahim [10] i utilitzar la distribució uniforme.

La funció que s'ha utilitzat en MATLAB és *unifrnd*.
La directiva d'aquesta funció és *unifrnd(A, B, M, N)*. A on els límits per als valors generats són A i B. El resultat es dona en forma de matriu de mida M i N:

A - límit inferior

B - límit superior

M – número de files

N – número de columnes

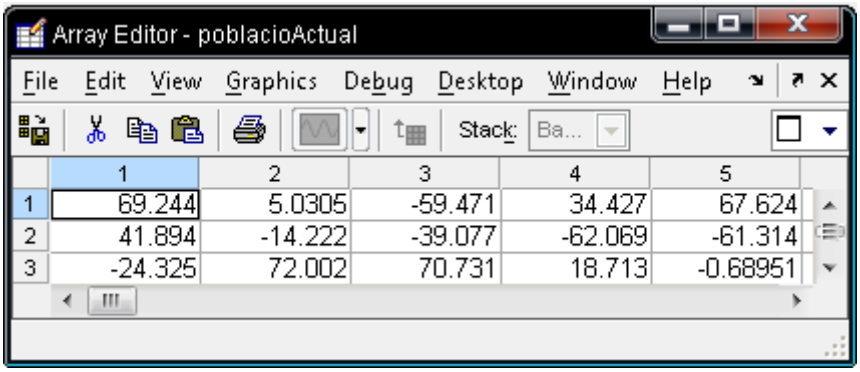
Els límits per als valors depenen de les condicions d'entrada així com la mida de la població.

Per a emmagatzemar la població s'ha definit la variable *poblacioActual*. Pel que respecta a la mida, tindrà les columnes especificades per *midaPoblacio*. Kp, Ki i Kd donen 3 files. Es pot veure més clar sobre la Figura 4.2.

La creació de la població actual s'ha fet generant 3 vectors i emmagatzemant-los dins d'una matriu:

```
poblacioActual =  [
unifrnd(limInfKp,limSupKp,1,midaPoblacio) ;
unifrnd(limInfKi,limSupKi,1,midaPoblacio) ;
unifrnd(limInfKd,limSupKd,1,midaPoblacio)];
```

Un exemple del valor de poblacioActual pot ser el de la Figura 4.2:



	1	2	3	4	5
1	69.244	5.0305	-59.471	34.427	67.624
2	41.894	-14.222	-39.077	-62.069	-61.314
3	-24.325	72.002	70.731	18.713	-0.68951

Figura 4.2: Exemple de valors de poblacioActual

Treballar amb aquesta estructura permet poder fer referència als membres de la taula fent servir índexs. Per exemple, l'accés a la cel·la seleccionada a la imatge es fa a través de la següent sentència:

```
poblacioActual(1,1)
```

Un script o fitxer .m és un arxiu de text que conté un conjunt de sentències. Per a executar un script només cal escriure el nom al *workspace*. Els scripts també poden ser invocats des de altres scripts o funcions.

4.2.4. Avaluació de l'aptitud

L'avaluació de l'aptitud és un procés que s'ha de determinar per a tots els elements o solucions de la població. En altres paraules, s'avalua l'aptitud de totes les columnes de la matriu *poblacioActual* de forma individual.

Coneixent el número de vegades que s'ha de realitzar aquest procés, es farà servir la directiva *for*, la qual permet fer un número determinat d'iteracions.

El procés es divideix en les següents parts:

- Obtenció de les solucions de prova
- Aplicació al sistema en llaç tancat
- Obtenció de les variables manipulades
- Avaluació de l'aptitud

Obtenció de les solucions de prova

Això engloba la obtenció dels valors de les constants Kp, Ki i Kd:

```
Kp = poblacioActual(1,i);
Ki = poblacioActual(2,i);
Kd = poblacioActual(3,i);
```

Aquestes constants es faran servir per a crear la funció de transferència del sistema en llaç tancat (ftlt).

Aplicació al sistema en llaç tancat i obtenció de les variables modificades

Un cop obtingudes les constants, s'executa la funció *obteCriteri*, que torna el valor de la funció objectiu i s'emmagatzema al vector de criteris (J).

A la funció, es crea la funció del control, *ftcontrol* i la funció en llaç tancat *ftlt*, (veure equacions 2.1 i 2.9) i es fa una primera comprovació de la inestabilitat del sistema, abans de procedir a trobar l'error. Per a això es comprova que els pols del sistema en llaç tancat no siguin positius. Les solucions inestables reben un valor J molt elevat per a reduir la probabilitat d'ésser seleccionades [2], [10].

```
%Genero la funció de transferència en llaç tancat
s=tf('s');
ftcontrol= Kp + Ki/s + Kd*s;    %Construeixo la funció del controlador
ftlo = planta*ftcontrol;        %Trobo el sistema en llaç obert
ftlt= feedback(ftlo,1);         %Trobo el sistema en llaç tancat

%Estudio els pols del s.l.t
pols=pole(ftlt);                %Trobo els pols
```

```
%Recorro el vector pols i comparo tots els valors amb zero
for ii=1:size(pols)
    if pols(ii)>0
        J=5e6;
        return;
    end
end
```

Suposant que el sistema sigui estable, es procedeix a calcular el criteri d'error tal i com mostra la Taula 4.1.

Avaluació de l'aptitud

Per a avaluar l'aptitud s'ha fet servir el vector d'errors *J*. Com que l'algoritme es mou a cegues per tota la regió de valors, no es pot determinar quin n'és l'òptim. S'ha optat per treballar amb relacions entre les solucions actuals. L'aptitud de cada solució de prova s'avalua a partir de la següent fórmula:

$$aptitud(i) = 100\left(1 - \frac{J(i) - \min\{J\}}{J(i)}\right) \tag{4.4}$$

Aquesta estratègia assigna un valor de 100, a la solució de prova que tingui l'error més petit, i a partir d'aquest calcula la resta d'aptituds.

A l'hora d'aplicar l'equació 4.4 al programa, s'ha creat una funció anomenada *avalua*.

Al denominador se li ha afegit el valor de la constant *eps*, per a evitar obtenir un error quan es faci una divisió per 0.

Les entrades d'aquesta funció són el vector *J* i el criteri 1. El resultat de la funció s'emmagatzema al vector *aptitud*:

```
aptitud = avalua(J,1);
```

4.2.5. La reproducció. Selecció de membres i generació de la següent població

Evolució de la població

La selecció assegura que les solucions més aptes sobrevisquin. No obstant, l'evolució de la població està determinada per la reproducció.

El procés de reproducció emprat es compon de dues tècniques: l'encreuament i la mutació. Aquestes tècniques són les que han utilitzat alguns autors de la bibliografia [8], [9].

S'anomena encreuament (de l'anglès *crossover*) a l'acció de combinar la informació de dos pares per a crear un fill.

La mutació consisteix en l'adició de nova informació sobre els membres de la població. Aquesta adició té un caràcter aleatori, i consisteix en la modificació de les característiques de les solucions en particular. En aquest cas es modificarà de forma aleatòria a partir de la variació dels valors dels membres [8].

Reproducció d'una generació

A l'hora de determinar com s'aplicaran l'encreuament i la mutació, es pot arribar a dues conclusions. O bé es duen a terme sobre el mateix grup de membres, o bé per separat i a diferents grups [8].

Quan es treballa en sèrie (el primer cas), es fan servir normes aleatòries per a decidir quan s'ha d'aplicar encreuament i quan s'ha d'aplicar mutació (per exemple, alguns algoritmes fan servir una constant per a determinar la probabilitat de mutació).

Aquesta primera opció pot semblar útil en una primera instància. És obvi que l'algoritme evolucionarà en tant que s'està aplicant una variació durant cada iteració. Però arribats a cert punt, els pares es tornen molt semblants (fins i tot idèntics) i es pot trobar el cas de què l'encreuament esdevingui una clonació: si els pares són molt similars el fill serà una còpia dels pares. El perill que això comporta és el d'estancar l'evolució de l'algoritme i per tant invalidar-lo.

Una de les possibles solucions al problema anterior pot ésser escollir la segona opció (treballar en paral·lel) i augmentar l'acció de la mutació [8].

L'estratègia escollida ha estat la segona.

Dit això, en aquesta etapa es tindrà en compte:
-Selecció dels membres elit
-Selecció de la resta de membres. Els pares per a la fase de reproducció (encreuament i mutació).

Membres elit

Existeix un altre mètode per a garantir l'evolució de la població: garantir la supervivència dels millors membres (anomenats membres elit).

Una de les altres particularitats de designar una elit és que l'aplicació del crossover deixa de ser estrictament necessària.

Selecció de membres

La selecció de membres tindrà en compte:

- Selecció dels membres elit
- Selecció de la resta de membres; els pares per a la fase de reproducció (encreuament i mutació).

Per a la elit, es seleccionen els millors membres a partir dels valors del vector aptitud. El número de membres és una de les especificacions d'entrada de l'algoritme.

Per a escollir els millors, s'ha utilitzat la funció *sort*. Aquesta funció és capaç d'ordenar en ordre descendent els valors del vector aptitud i tornar-los en el vector ordenat i els índexs en el vector identitat.

Després s'emmagatzemen els índexs a la variable *elit*.

Per a la selecció de la resta de membres es crea la variable *pares*, que també conté índexs. La longitud d'aquest vector és diferència entre la mida de la població i els membre elit.

Per als mètodes de selecció, es pot optar per afavorir, o no, als membres més aptes. Reeves analitza les diverses opinions al respecte [8].

El mètode escollit ha estat la roda de ruleta o *roulette wheel*. La implementació en codi .m es troba a la funció *rouletteWheel*. Es pot trobar una explicació més a fons de la funció annexada al final del projecte.

Creació de la elit

Per a construir la següent població, es treballa sobre la variable *poblacioNova*.

Per a afegir els membres elit es fa servir la següent sentència:

```
poblacioNova = poblacioActual(:,elit)
```

A partir dels índexs de la variable elit, es poden seleccionar els membres corresponents de *poblacioActual*, i ésser assignats a la nova població.
Amb els dos punts, s'indica que es vol passar tota la columna.

La Figura 4.3 mostra un petit esquema a on es pot observar la generació actual, la generació següent i el procés de creació.

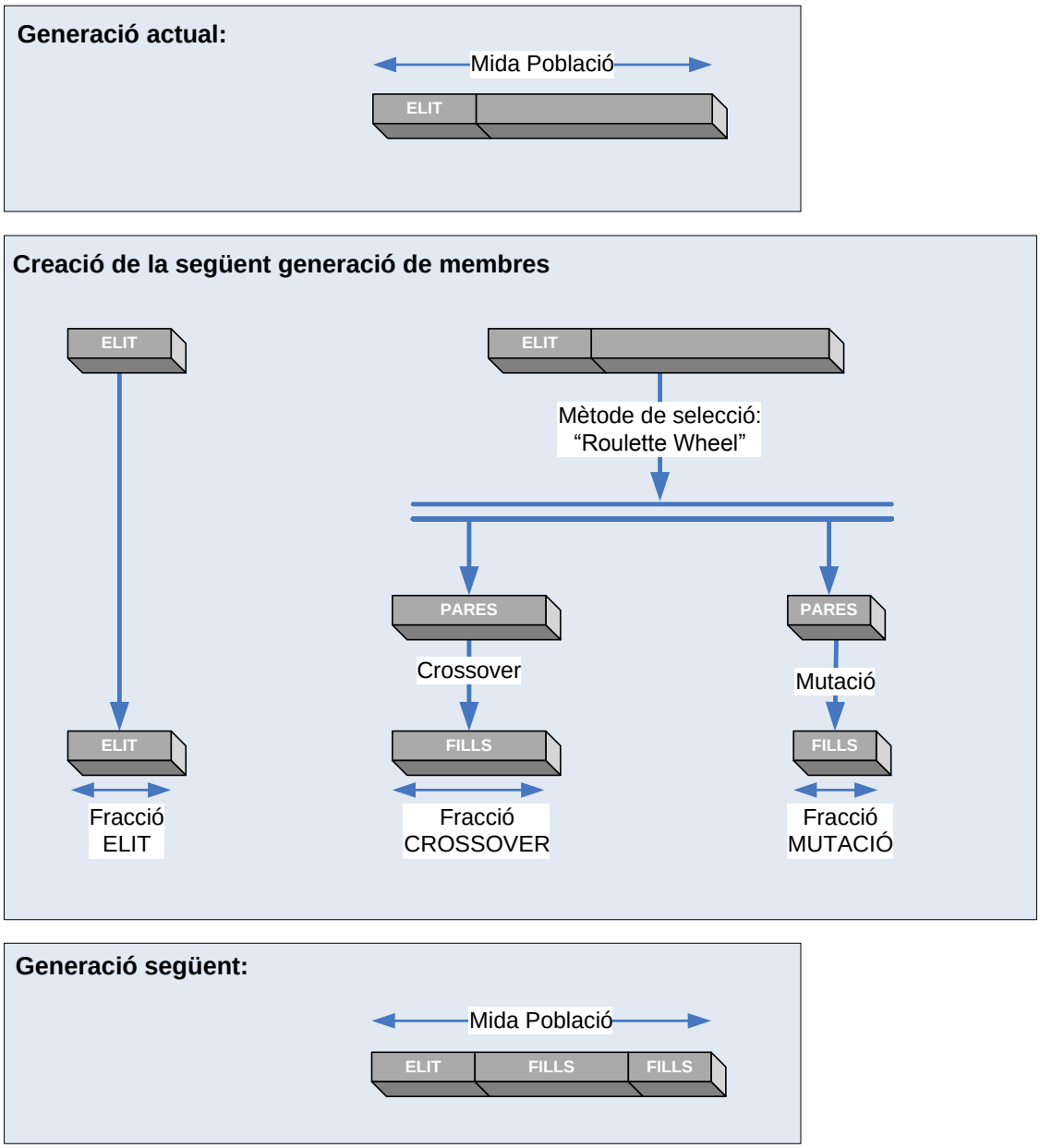


Figura 4.3: Composició de la població de fills

Més endavant, es divideix el vector de pares en dues parts, una per a encreuament i l'altra per a mutació.

Generació de fills per encreuament

La reproducció per encreuament es duu a terme seleccionant una parella a l'atzar i generant dos fills. Com que el vector de pares ja prové d'un mètode de selecció aleatori, els pares s'escullen en parelles directament (i.e. 1 i 2, 3 i 4...).

Tal i com s'ha comentat anteriorment, l'encreuament correspon a la barreja de les característiques dels pares. El punt d'encreuament és aquell a on es canvia l'origen de les dades.

Per exemple, en el cas d'una solució de prova amb tres valors, si els pares són:
pare 1 = [a1 a2 a3]
pare 2 = [b1 b2 b3]

Amb el punt d'encreuament al segon valor, el fill esdevé

fill = [a1 b2 b3]

La primera part del fill està formada pels membres del pare 1, però la segona pels membres del pare 2.

El crossover es pot classificar en funció del número de punts d'encreuament (encreuament d'un punt, de dos punts...).

Per a la implementació a Matlab, s'ha decidit determinar els punts a partir d'un mètode aleatori. A més, es tindrà en compte que per a cada parella de pares es generaran dos fills.

En primer lloc, es genera un vector binari que actua com a màscara per a determinar la composició del fill.

Amb un valor de 0 s'agafa el membre del primer pare; i el contrari per a 1.

A la guia de referència de GAtool [9] s'anomena a aquest mètode *Scathered*.

Per a la generació de la màscara es va optar per a fer servir la següent sentència:

```
vectorEncreuament = randint(1,3)
```

Tot i així, Reeves [8] proposa fer servir una millora d'aquest procés i fer un mètode aleatori sobre la distribució de Bernoulli. Amb el paràmetre de Bernoulli (*p*), es pot afavorir la creació de 1s o 0s.

Per això es va crear la funció *mascara*. Amb aquesta funció es crea la màscara binària i s'emmagatzema al *vectorEncreuament*.

És important tenir en compte que quan el paràmetre de Bernoulli és igual a 0.5, es comporta igual que el mètode anterior.

Els fills obtinguts s'afegeixen a la nova població.

Generació de fills per mutació

La mutació es duu a terme sobre els elements que han estat seleccionat com a pares i que corresponen a la fracció de mutació. Dins del vector pares és la part complementària a la fracció d'encreuament `i=midaEnc:midaPares`.

La probabilitat de mutació és un dels paràmetres d'entrada de l'algoritme.

El procés de mutació consisteix en modificar els dígits del pare, novament a partir d'una màscara generada per Bernoulli.

Els bits a 1 indicaran els membres de la solució de prova a modificar. En aquest cas es tindrà en compte l'addició de dígits decimals a la modificació, de forma que sempre tinguem 4 dígits de representació.

Per a això s'ha imposat l'addició de xifres decimals de la següent forma:

Valor	Número de decimals
Major de 10 (Desenes, Centenes...)	2 xifres
Entre 10 i 1 (Unitats)	3 xifres
Més petit de 1 (dècimes, centèsimes...)	4 xifres

Taula 4.2: Relació entre la magnitud del número i els decimals

Un cop definits els dígits decimals, es procedeix a utilitzar la funció de mutació *mutaNumero*.

Aquesta funció modifica els dígits separant la part entera i la part decimal, convertint el número a cadena de text i per últim modificant els caràcters per a després passar-ho novament a número.

Per a la generació d'un caràcter aleatori de 0 a 9 s'ha fet servir la següent sentència:

```
num2str(fix(rand*10))
```

La generació del número també té en compte que aquest s'ha de trobar entre els límits establerts a l'inici del programa.

Actualitzar generació

Un cop s'ha acabat de generar la nova població, aquesta passa a ser la població actual per a la següent iteració.

4.2.6. Finalització del programa. Condició d'aturada

Tal i com es pot apreciar a la Figura 4.1, el procés es repeteix indefinidament fins que es compleixi la condició de sortida, que en aquest cas consisteix en el número d'iteracions (variable *generacions*).

Quan finalitza, el programa retorna la solució; els valors obtinguts de Kp, Ki i Kd.

4.3. Implementació sobre MATLAB

La implementació sobre Matlab s'ha dut a terme fent servir la eina Graphical User Interface (GUI). Aquesta utilitat ofereix una millor introducció de les dades.

Un dels altres avantatges que ofereix treballar així és la possibilitat d'incorporar més algoritmes, donat que moltes de les dades d'entrada seran comunes.

4.3.1. Interfície

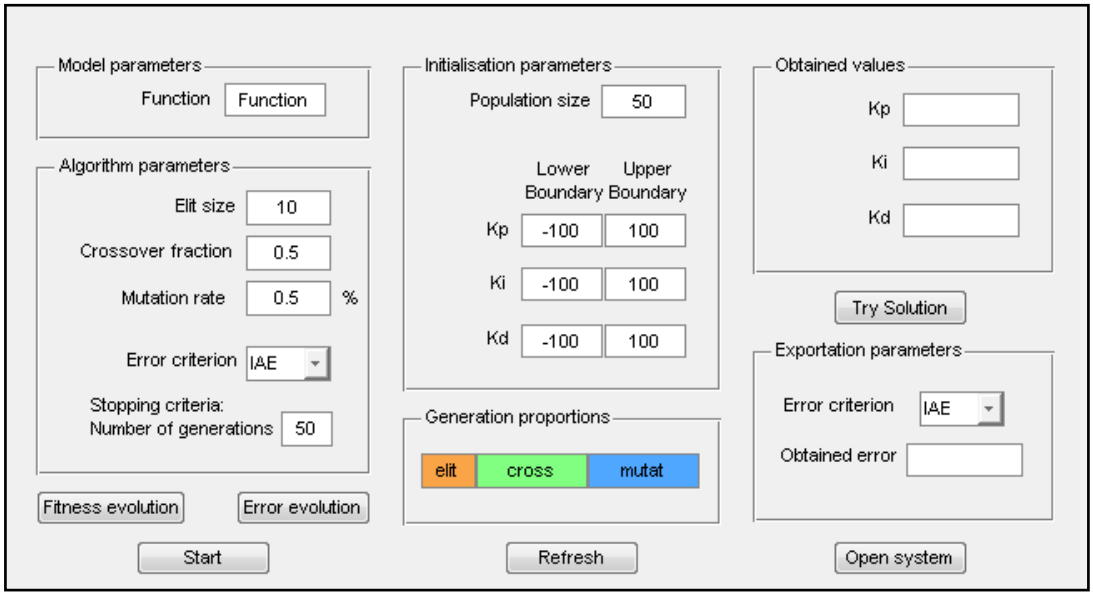


Figura 4.4: Captura de la interfície GA

A la imatge es pot observar com a part dels elements relatius a l'algoritme s'ha afegit una petita representació de les proporcions de generació de poblacions, i una altra per a emmagatzemar les variables de sortida.

4.3.2. Execució del programa

A part de l'algoritme ja comentat, el programa utilitza dos scripts per a la lectura (*llegeix*) i validació (*comprova*) dels paràmetres d'entrada.

En el cas de trobar algun error, el programa llença una finestra de diàleg amb el motiu de l'error i passa el focus d'atenció al quadre corresponent.

Per a arrencar l'algoritme cal prémer la tecla *Start*. Un cop iniciada la operació, apareix una barra de progrés per a mostrar l'estat de l'execució.

Quan finalitza, el programa utilitza l'script *mostraValor* per a introduir els valors de la solució obtinguda al quadre *Obtained values*.

4.3.3. Proporcions generació

El quadre de *Generation proportions* funciona al prémer el botó *Refresh*. Utilitza els valors de la mida de població (*Population size*), la elit (*Elit size*) i la fracció d'encreuament (*Crossover fraction*) per a manipular les amplades de cada fracció de la barra (*elit*, *cross* i *mutat*).

4.3.4. Prova de la solució obtinguda

A partir dels valors obtinguts podem observar el valor de l'error per als 3 criteris amb el botó *Try solution*.

La opció *Open system* mostra un gràfic amb la sortida del sistema en llaç tancat.

Les opcions *Fitness evolution* i *Error evolution* permeten veure la evolució de la població.

5. Resultats

5.1. Resultats de l'algoritme genètic

A continuació es realitzen diferents proves per a trobar les constants del control per a la funció de transferència de la equació

Per al sistema de la equació 2.14 i els valors presents a les equacions 2.17, 2.18 i 2.19, els errors són:

Proposta	Valor	Error	
Kp	36,6358	IAE	13,5969
Ki	38,9134	ITAE	16,3705
Kd	8,6229	ISE	6,5502

Taula 5.1: Error del controlador amb els valors calculats

Si en canvi busquem els valors a través dels algoritmes genètics es pot observar els següents resultats.

Paràmetre	Valor
Mida població	100
Límits Kp	0 – 80
Límits Ki	0 – 80
Límits Kd	0 – 80
Mida Elit	20
Fracció enc.	0.4
Prob. Mutació	100 %
Generacions	50

Taula 5.2: Paràmetres d'entrada de l'algoritme

Per a aquests valors, els resultats són els següents (classificats en funció del criteri d'error).

Criteri	Kp	Ki	Kd	IAE	ITAE	ISE
IAE	37,7677	78,239	77,7154	**4.2879**	8.2442	1.6608
ITAE	75,946	46,548	69,8032	7,6027	**3,2985**	3,726
ISE	79,8814	0,2626	14,896	12,1158	6051.83	**1,36**

Taula 5.3: Resultats obtinguts amb l'algoritme genètic en funció del criteri

Com es pot observar, l'algoritme troba diferents valors per a les constants. Tot i així els valors dels altres criteris tendeixen a créixer. En el cas del criteri ISE, es pot observar com encara que l'error ISE és de 1,36, per a l'ITAE ascendeix a uns 6051,83.

La progressió de l'aptitud es pot veure a la Figura 5.1.

Els colors propers al vermell indiquen una aptitud major mentre que els blaus indiquen una aptitud pròxima a 0.

A mesura que avancen les generacions els millors membres es van dipositant al principi de la població en un grup de 20 membres.

En quant a la resta de la població, es pot apreciar com hi ha un petit canvi al voltant del membre número 50.

La primera franja és la de l'encreuament i la proporció de membres propers al vermell és relativament alta. Amb això es pot comprovar com el mètode de selecció de roda de ruleta afavoreix els millors membres.

La tercera part, corresponent a la de mutació és la que produeix els valors més dispersos.

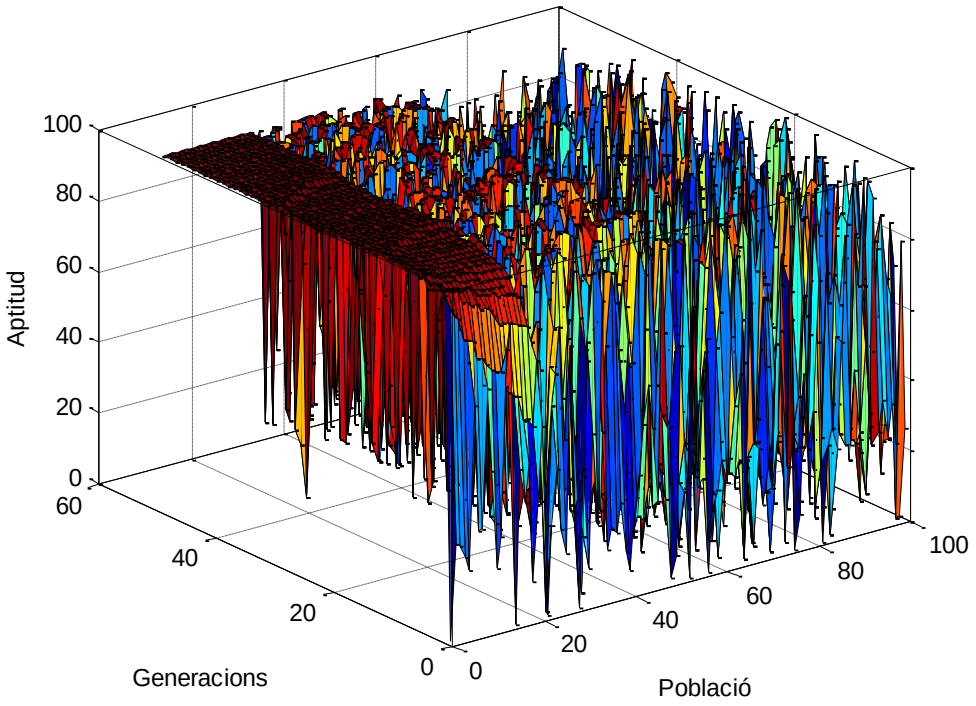


Figura 5.1: Evolució de l'aptitud al llarg de les iteracions

Si mantenim els mateixos paràmetres d'entrada però variem la fracció de l'encreuament, a un valor de 0.8, els membres amb bona aptitud creixen i pel contrari l'aparició de nous membres es redueix dràsticament:

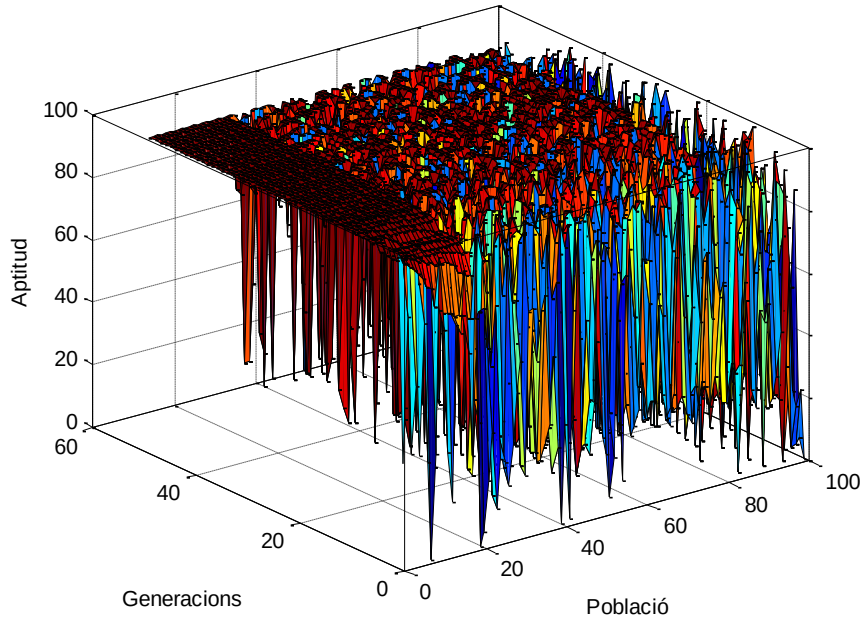


Figura 5.2: Evolució de la població quan es redueix la mutació

Ara negligim els membres elit i mantenim la fracció de 0.8 per a encreuament. Encara que desapareix la franja de la elit, encara es pot veure una evolució (però el número de membres al voltant del 100 % d'aptitud es redueix considerablement).

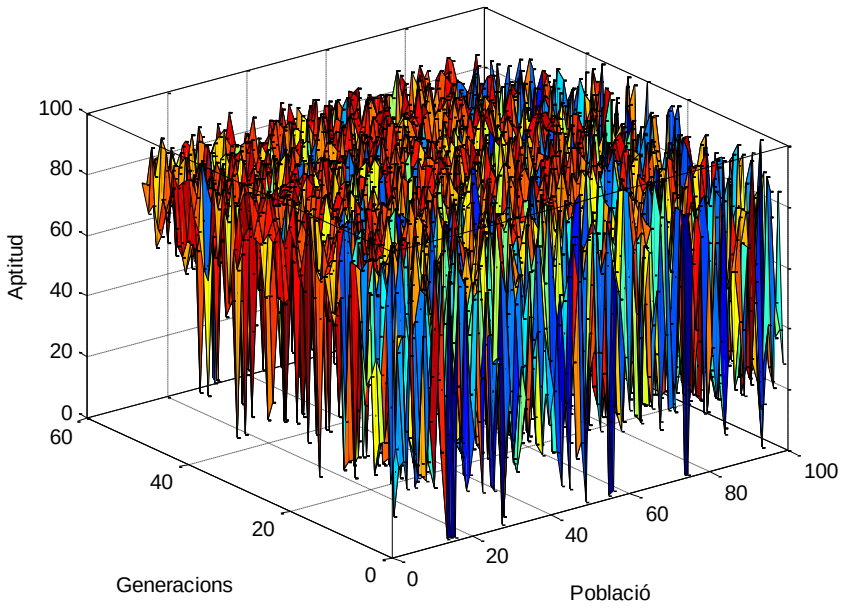


Figura 5.3: Evolució de la població sense membres elit

Per últim, si pel contrari reduïm l'encreuament l'algoritme tendeix a dispersar-se. S'ha tornat a treballar amb un grup de 20 membres elit per a tenir una referència; així es pot veure com la resta de solucions són, en major part, pitjors (hi ha més membres blaus):

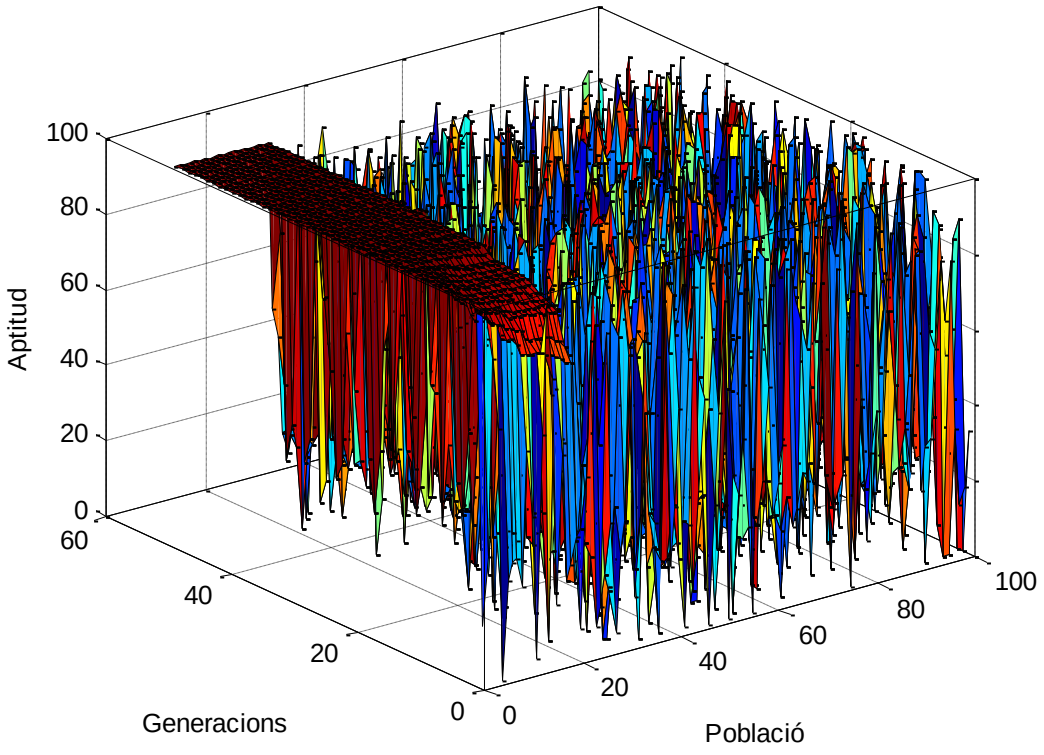


Figura 5.4: Evolució de la població sense encreuament

6. Conclusions i millores futures

6.1. Conclusions

Per a aplicar els algoritmes genètics a la sintonia PID i aconseguir més bons resultats és aconsellable començar amb uns valors inicials factibles. Quan es treballa des de zero per a la generació de la primera generació, com és el cas de la distribució uniforme, l'atzar juga una part important el que implica que el número de iteracions i la mida de la població han de ser molt més grans.
Una forma de reduir aquests valors i , conseqüentment, el temps de càlcul, és restringir els marges per als valors de Kp, Ki i Kd. No obstant això, el temps que necessiten els algoritmes genètics continua essent elevat.

L'eina GA de Matlab ha resultat molt útil a la hora de dissenyar algunes parts del codi.

6.2. Millores futures

En quant a les millores futures, és interessant treballar amb la funció objectiu sobre Simulink, d'aquesta manera es poden introduir retards purs a la funció de transferència.

Una de les altres millores que es poden estudiar és una funció de mutació que estudiï la evolució de la població per a incorporar números que representin variacions majors o menors. Per exemple, quan es disposa de valors dolents cal produir números nous, però quan es treballa sobre un valor bo és més important afegir petites modificacions.

Annex I. Codi del programa

Per a realitzar l'exemple de recuita simulada s'han programat dues funcions.

La primera funció, *fObjectiu*, és capaç de tornar el valor imatge d'una expressió per a un valor de x gràcies a la funció *eval*. Les entrades són la expressió a analitzar i el valor de la coordenada independent.

La segona funció, *recuita*, aplica l'algoritme de recuita simulada sobre un polinomi. Això és així donat que l'exemple tracta sobre la programació lineal.

Les dues primeres entrades de la funció especifiquen els límits de la regió a la qual es vol treballar: *liminf* per al límit inferior i *limsup* per al límit superior. Després s'ha d'introduir el valor de la solució de prova inicial, *solinicial*. Per últim, s'ingressa la expressió del polinomi com a cadena de text i aquesta queda emmagatzemada a *fobj*.

Els valors del programa de temperatura s'obtenen de la mateixa forma que a les equacions 3.7 i 3.8:

```
T=0.2*Zc;


for i=2:5
    T(i)=0.5*T(i-1);
end
```

En quant a la regla de selecció de moviment, la generació de candidats es fa utilitzant una distribució normal centrada a zero i amb desviació sigma:

```
sigma=(limsup-liminf)/6                  %Implementació equació 3.5
candidat=x+norminv(rand(),0,sigma)       %Implementació equació 3.6
```

En acabat, per tal de complir la regla de selecció quan es fa un pas descendent, es troba la probabilitat i es fa una comparació amb un número aleatori (funció *rand*):

```
probabilitat=exp((Zn-Zc)/temperatura);
if rand()< probabilitat
```

Quan el número aleatori és inferior a la probabilitat, el candidat passa a ser la solució actual.

La sentència per a realitzar el problema del capítol 2 és:

```
recuita(0,31,31/2,'12*x^5-975*x^4+28000*x^3-345000*x^2+1800000*x')
```

Annex II. Programa de l'algoritme genètic

S'ha de tenir en compte que la programació del codi es pot millorar en eficiència. S'han fet servir variables innecessàries per tal d'aclarir alguns processos, i s'han afegit multitud de comentaris.

El motiu ha estat intentar deixar una feina clara, entenedora i intuïtiva per a estudiants que vulguin utilitzar el material com a via d'aprenentatge o, fins i tot, com a base per a realitzar un nou projecte.

Si es fa us de la directiva *help* seguit del nom de la funció o *script* es pot obtenir una petita ajuda/resum.

Els scripts utilitzats són:

- algoritmeGA.m – per a l'algoritme genètic
- primGen.m – crea la primera generació
- detFit.m – determina la forma o aptitud d'una població
- seleccio.m – per a la fase de selecció
- generacio.m – per a la fase de reproducció
- llegeix.m – per a llegir totes les variables de la interfície gràfica
- comprova.m – per a comprovar totes les condicions de les variables
- actualitzaBarra.m – per a actualitzar la barra de la interfície gràfica
- mostraValor.m – per a veure els valors de la solució final al User Interface

Les funcions:

- avalua.m – per a avaluar l'aptitud d'una solució
- mascara.m – per a generar una màscara binària
- mutaNumero.m – per a aplicar la mutació a un número
- obteCriteri.m – per a obtenir el criteri d'una solució
- rouletteWheel.m – per a seleccionar un valor d'un grup a partir de la roda de ruleta

Annex III. Implementació del mètode de selecció: Roda de Ruleta

Aquest mètode de selecció es fa servir per a ponderar als elements de la ruleta.

Es disposa d'una distribució de valors sobre un cercle. El cercle està dividit en tantes seccions com valors tingui la distribució de números. La superfície de cada secció serà major o menor en tant a la probabilitat de ser escollit (en el cas de l'algoritme genètic, l'aptitud).

La següent figura mostra un exemple:

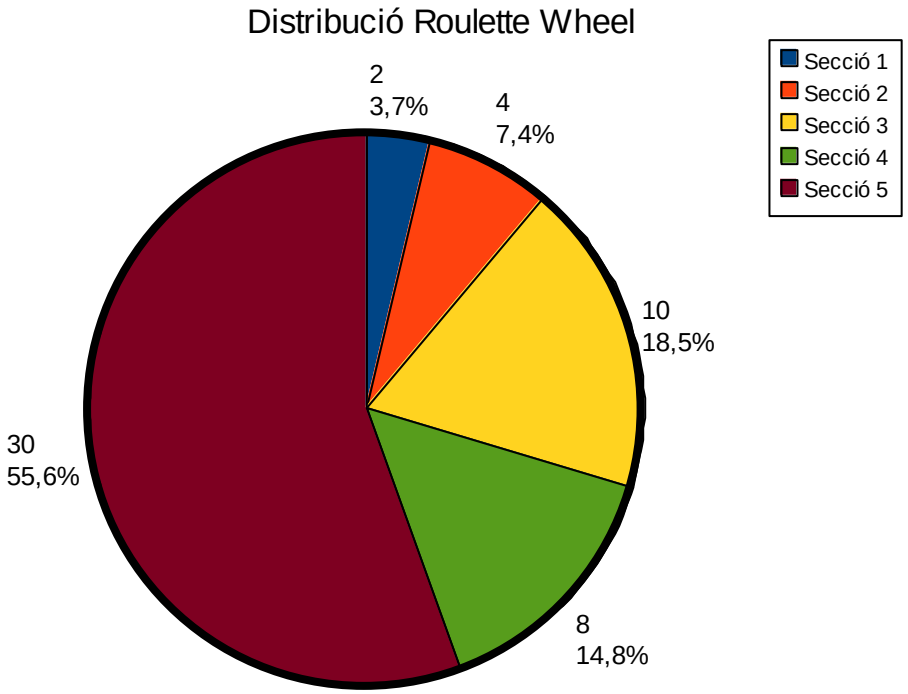


Figura I: Esquema de la distribució dels valors sobre la roda de ruleta

Disposem d'un conjunt d'elements o aptituds que van ordenats tal i com es mostra a la taula:

Secció	aptitud	probabilitat	Límit inferior	Límit superior
Secció 1	2	3,70%	0	2
Secció 2	4	7,40%	2	6
Secció 3	10	18,50%	6	16
Secció 4	8	14,80%	16	24
Secció 5	30	55,60%	24	54
total	54	100,00%		

Taula I: Càlcul de la probabilitat dels membres de la distribució

A partir la llegenda de la figura podem observar la secció de cada element.
Els límits corresponen a la intersecció entre les seccions, seguint l'ordre de les agulles del rellotge. Aquest es poden trobar a partir de la suma de l'aptitud del membre corresponent a la secció, tal i com es mostra a la taula.

En primer lloc, es genera un número aleatori que es trobi entre 0 i el **total**.
Després es realitza una comparació d'aquest número amb els límits inferior i superior de cada secció.

El mètode implementat a Matlab fa servir la funció **random** per a trobar un número entre 0 i la suma d'aptituds.
Es val de la sentència for per a recórrer a tots els elements de la taula, és a dir, tots els elements del vector aptitud. La comparació dels límits per a cada secció, correspondrà a una iteració.

Glossari

Bibliografia

[1] J. Triado. *Apunts Regulació Automàtica*. Quadrimestre primavera 2008. EUPMT

[2] I. Griffin. *On-line PID Controller Tuning using Genetic Algorithms*. 22 Agost 2003. pàgines 11 – 24.

[3] L. Riesberg. Genetic *Algorithms – New Tools for the Programmers' Toolbox*. 16 Abril 2003.

[4] *Investigación de Operaciones*. Hillier & Lieverman. 8 Ed. MC Graw Hill. 2010

[5] http://sci2s.ugr.es/docencia/algoritmica/metaheuristicas-vision-global.pdf. B. Melián, J.A. Moreno, J.M. Moreno. *Metaheuristics: A global view*. Inteligencia Artificial (2003).

[6] *Investigación de Operaciones*. Hillier & Lieverman. MC Graw Hill. 2010. *Ejemplo de programación no lineal*. p.587.

[7] T. O'Mahony, C.J. Downing, K. Fatla. *Genetic Algorithms for PID Parameter Optimisation: Minimising Error Criteria*.

[8] C. Reeves. *Genetic Algorithms*. Chapter 3. School of Mathematical and Information Sciences. Conventry University.

[9] *Genetic Algorithm Tool for MATLAB. Quickreference.*MATLAB

[10] S. B. M. Ibrahim. *The PID controller design using genetic algorithm*. University of Southern Queensland. Faculty of Engineering and Surveying. (October, 2005).