

PROJECTE FINAL DE CARRERA

LLICENCIATURA EN CIÈNCIES I TÈCNIQUES ESTADÍSTIQUES

DATA MINING EN SÈRIES TEMPORALS: Aportació Pràctica Volum II: Annexos

Autor: Raquel Delriu Campillo

Director: Pilar Muñoz Gràcia

UNIVERSITAT POLITÈCNICA DE CATALUNYA
Biblioteca



1400363947



Facultat de Matemàtiques
i Estadística

UNIVERSITAT POLITÈCNICA DE CATALUNYA

DADES DEL PROJECTE:

Nom de l'estudiant: RAQUEL DEL RÍO AMPILIO

DNI: 40995152K

Títol del Projecte: DATA MINING EN SÈRIES TEMPORALS: APROXIMACIÓ
PRÀCTICA

Director del Projecte: M. PIQUER MUÑOZ GRANADA

Tutor del Projecte:

QUALIFICACIÓ

MATRÍCULA D'HONOR

MEMBRES DEL TRIBUNAL (nom i signatura)

President: M. ANTONI MARÍN VASQUEZ

Vocal: Ricard Gavaldà Ricard Gavaldà

Secretari M. PIQUER MUÑOZ JG

Data: 20/10/04

Data Mining en Sèries Temporals: aportació pràctica

Volum II: Annexos

Autor: Raquel Delriu Campillo
Director: Pilar Muñoz Gràcia

Octubre 2001

ÍNDEX

Volum I. Memòria

Pròleg i agrairments.....	i
Índex.....	iii
Capítol 1. Introducció.....	1
1.1 Time Series Data Mining.....	1
1.2 Sumari	2
Capítol 2. Sèries Temporals i Data Mining.....	4
2.1 Anàlisi Clàssic de Sèries temporals. Limitacions.....	4
2.2 El Data Mining.....	7
2.3 Sèries Temporals i Data Mining: metodologies existents.....	12
Capítol 3. Time Series Data Mining: conceptes bàsics.....	20
3.1 Descripció intuïtiva del mètode.....	20
3.2 Definició dels conceptes bàsics.....	25
Capítol 4. Time Series Data Mining: explicació del mètode.....	34
4.1 Mètode del Time Series Data Mining.....	34
4.2 Avaluació dels resultats.....	47
Capítol 5. Algorismes Genètics.....	52
5.1 Què són els algorismes genètics?.....	52
5.2 Algorismes genètics com a tècnica d'optimització.....	54
5.3 Un algorisme genètic simple.....	57
5.4 Particularitats i modificacions pròpies del TSDM.....	65
Capítol 6. Exemples bàsics.....	67
6.1 Sinusoïdal.....	67
6.2 Moviments sísmics.....	87
6.3 Soroll blanc.....	101
Capítol 7. Aplicació a algunes sèries temporals típiques.....	106
7.1 Sèrie Tempcast: estacionalitat.....	107
7.2 Variància no constant.....	124
7.3 Sèrie Airpass: tendència, estacionalitat i variància no constant.....	135
7.4 Detecció de dades atípiques.....	143
7.5 Conclusions.....	153
Capítol 8. Aplicació a sèries reals.....	154
8.1 Sèrie Speech.....	154
8.2 Sèrie Lynx.....	172
Capítol 9. Ampliacions del mètode.....	189
9.1 Detecció i predicció d' <i>Events Minims</i>	189
9.2 Funció d'events a s passes vista.....	200
Capítol 10. Conclusions.....	209
Bibliografia.....	212

Volum II. Annexos

Índex.....	i
Annex 0. Aclariments sobre la funció de repulsió.....	1
Annex 1. Algorisme Genètic Simple.....	5
Annex 2. Implementació del TSDM.....	41
Annex 3. Millores en l'eficiència de l'algorisme.....	74

ANNEX 0. ACLARIMENTS SOBRE LA FUNCIÓ DE REPULSIÓ

En el capítol 3 s'ha introduït la funció de repulsió $b(\delta)$, com una funció auxiliar que permet, tot introduint-la com a segona funció objectiu en la formulació de l'optimització, obtenir per a iguals valors de la funció objectiu principal $f(P)$, clusters amb un valor de δ moderat.

Les tres alternatives per a la formulació de l'optimització que es proposaven en el capítol 4 són:

$$\max f(P) \text{ subjecte a } \min \delta \quad (\text{A2.1})$$

$$\max f(P) \text{ subjecte a } \min b(\delta) \quad (\text{A2.2})$$

$$\max f(P) \text{ subjecte a } \max \delta \quad (\text{A2.3})$$

Es decideix optar per la tercera perquè és la que deixa més equilibrats l'error de mal classificació positiva (classificar com a patró temporal caracteritzador d'events una observació que no ho és) i l'error de mal classificació negativa (classificar com a patró no caracteritzador d'events una observació que si que ho sigui). Aquest fet es produeix perquè, tal com s'establia al capítol 3 es compleix que:

$$\delta_{\min}^* \leq \delta_b^* \leq \delta_{\max}^* \quad (\text{A2.4})$$

on δ_{\min}^* és el radi òptim obtingut al resoldre la formulació (A2.1), δ_b^* el radi òptim al resoldre (A2.2) i δ_{\max}^* al resoldre (A2.3).

Ja que en el mètode implementat s'ha decidit usar la funció $b(\delta)$ per tal de moderar el radi, cal comprovar que efectivament es compleix (A2.4). Com que els radis òptims no poden trobar-se de manera analítica, el que s'ha fet per comprovar-ho és executar un cert nombre de vegades el mètode amb cadascuna de les formulacions, cosa que només ha suposat una petita modificació en el programa, en la funció Selecció_Torneig (veure Annex 2).

S'ha executat el programa amb per tots els casos amb els paràmetres de les taules A2.1 i A2.2.

Paràmetres del mètode TSDM

f	β	g	Q	distància
$f(P) = \frac{\mu_M - \mu_{M^c}}{\sqrt{\frac{\sigma_M^2}{c(M)} + \frac{\sigma_{M^c}^2}{c(M^c)}}}$	-	$g(t) = x_{t+1}$	2	Euclidea

Taula A2.1

Paràmetres de l'algorisme genètic	
Paràmetres	Valor triat
Grandària de la població	100
Probabilitat de creuament	0.5
Probabilitat de mutació	0.001
Comptador d'elit	1
Participants al torneig	2
Longitud del gen	8
Rati de convergència	1

Taula A2.2

S'ha triat aquesta funció objectiu perquè és la que menys influència té en el radi, tal com ja s'ha esmentat en ocasions anteriors.

Amb els radis obtinguts en cada cas s'ha realitzat un t-test de comparació de mitjanes per tal de comprovar (A2.4).

Els resultats de les execucions realitzades són els de la taula A2..3.

Comparació de radis segons la formulació de l'optimització			
nº d'execució	Formulació amb $\min \delta$	Formulació amb $\min b(\delta)$	Formulació amb $\max \delta$
1	1.0678	2.2858	2.5027
2	1.4682	2.3358	2.0522
3	1.0678	2.1189	2.1857
4	1.3347	4.0043	3.4537
5	1.6684	2.2858	2.9876
6	0.6673	2.1189	2.1023
7	1.3014	1.7686	1.4182
8	1.3347	2.8531	2.4526
9	1.9521	1.8687	3.4537
10	0.7675	3.6787	2.0689
11	1.7519	1.7686	1.852
12	0.5672	2.3692	2.1189
13	0.1835	1.8186	2.5361
14	1.9521	1.1846	1.5683
15	1.6017	1.9187	2.486
16	1.1012	1.9688	2.5527
17	1.1178	1.535	1.3181
18	2.3192	2.1189	2.0355
19	3.9042	0.3369	3.0533
20	1.2179	1.9354	2.8698
21	1.5183	1.7868	3.1868
22	1.3514	2.3525	2.1189
23	0.6673	2.0188	1.852
24	1.4849	2.0856	1.585
25	1.9187	1.6518	1.6351
26	1.4515	1.8186	2.3859
27	1.4015	1.9855	2.0522

28	1.6017	1.6851	2.2529
29	0.5339	2.5861	2.3859
30	2.0522	1.9521	0.7841
31	1.6017	1.4015	3.5872
32	1.1345	1.7352	3.1701
33	1.4182	1.5516	2.5027
34	1.3014	1.5688	2.1023
35	1.8019	2.2691	1.535
36	1.4349	3.2035	2.9699
37	1.4682	2.7863	1.4349
38	2.219	1.6684	2.5194
39	1.2847	2.3358	2.9866
40	1.0678	1.5016	1.7185
41	0.5672	1.535	2.1857
42	0.3503	2.5194	0.5005
43	1.6184	1.6851	2.2524
44	1.6017	1.7185	2.1189
45	1.6017	2.1356	2.5154
46	2.1189	3.0032	1.8015
47	1.0845	1.9855	2.0522
48	1.6017	1.9187	2.5194
49	1.6851	2.0188	2.9198
50	1.1846	2.4026	2.0022
Mitjana	$\bar{\delta}_{\min} = 1.4095$	$\bar{\delta}_b = 2.0628$	$\bar{\delta}_{\max} = 2.2540$
Variància	$s^2_{\min} = 0.3470$	$s^2_b = 0.3539$	$s^2_{\max} = 0.4106$

Taula A2.3

Els dos contrastos d'hipòtesi que es volen realitzar són:

$$\left. \begin{array}{l} H_0 : \bar{\delta}_{\min} \geq \bar{\delta}_b \\ H_1 : \bar{\delta}_{\min} < \bar{\delta}_b \end{array} \right\} \quad (1)$$

$$\left. \begin{array}{l} H_0 : \bar{\delta}_b \geq \bar{\delta}_{\max} \\ H_1 : \bar{\delta}_b < \bar{\delta}_{\max} \end{array} \right\} \quad (2)$$

Pel primer cas el valor de l'estadístic $T = -5.46$ amb un $p\text{-valor} = 0.000$ per tant es rebutja la hipòtesi nul.la i es pot donar per cert que en mitjana el radi òptim per la formulació amb min δ és menor que per la formulació amb min $b(\delta)$.

Pel segon cas el valor de l'estadístic $T = -1.53$ amb un $p\text{-valor} = 0.065$. Així, encara que per molt poc, en aquest cas s'accepta la hipòtesi nul.la si es treballa amb el 95% de confiança. El que passa és que la hipòtesi nul.la inclou el cas d'igualtat, cosa que també verificaria l'expressió (A2.4). Cal doncs realitzar el test següent:

$$\left. \begin{array}{l} H_0 : \bar{\delta}_b = \bar{\delta}_{\max} \\ H_1 : \bar{\delta}_b \neq \bar{\delta}_{\max} \end{array} \right\}$$

En aquest cas s'obté, al 95%, un valor de T=-1.53 amb un p-valor=0.13. S'accepta doncs la hipòtesi nul.la i queda així demostrada l'expressió (A2.4), que és el que es volia.

ANNEX 1. ALGORISME GENÈTIC SIMPLE

Aquest annex pretén presentar el codi en llenguatge C d'un algorisme genètic simple, seguint l'estructura explicada al capítol 5. Les parts més importants com els operadors genètics o bé la funció que passa de generació en generació són comentades, així com l'estructura de dades usada per representar la població que va passant per l'algorisme, els individus d'aquesta població, etc....per tal d'acabar de comprendre com funcionen aquests algorismes i de constatar la sorprenent senzillesa de les operacions que utilitzen.

A1.1 Descripció del programa AGS.c

Estructures de dades

El primer que cal plantejar-se és quin tipus d'estructures de dades són les necessàries per a implementar l'algorisme. Tal com estan plantejats els algorismes genètics hi ha tres estructures bàsiques que cal definir: els cromosomes, els individus i la població. En el quadre següent es presenta el codi que s'ha usat per tal de definir aquestes estructures.

```
#define maxpop 2000
#define lgen 8
#define dim 1
#define lcrom (dim*lgen)

//ESTRUCTURES DE DADES

typedef struct {
    int C[lcrom];
} cromosoma;

typedef struct {
    cromosoma crom;
    float x;
    float fitness;
    int parent1,parent2,locus,triat;
    int npreselec;
} individu;

typedef struct {
    individu P[maxpop];
} poblacio;
```

Cromossoma

Tal com ja s'ha esmentat els cromossomes contenen els valors codificats de les variables del problema. Com que s'ha decidit usar una codificació amb bits els cromossomes no són més que taules d'enters d'una certa longitud predeterminada (`lcrom`), que contindran en cada posició o un 0 o un 1. `lcrom` es determina segons el nombre de variables de la funció (`dim`) i la longitud que es vulgui usar per representar codificada cada variable (`lgen`). En aquest cas com que la funció que s'usarà només té una variable la longitud del cromossoma és igual que la del gen, però en el TSDM, en què com a mínim s'usen 3 variables, `lcrom` augmenta, ja que aleshores el cromossoma conté les codificacions de les tres variables juxtaposades.

Individu

Els individus que conferiran la població que anirà canviant de generació a generació han de contenir la pròpia informació com a cromossomes, que és la que passarà pels operadors genètics, però també el valor del cromossoma decodificat (`x`), el valor associat de la funció objectiu (`fitness`), indicadors per saber quins individus són els seus pares i el punt de tall que han patit aquest (`parent1, parent2, locus`) i altres indicadors necessaris pels processos de selecció i creuament (`npreselec, triat`).

Població

Per acabar, la població que inicialment es genera de manera aleatòria i que després es va modificant generació a generació no és més que una taula d'individus.

Operadors Genètics

Un algorisme genètic es basa en les operacions de Selecció, Creuament, Mutació i Reinscripció de les quals es presenta tot seguit el codi tot comentant-ne els punts més importants.

Selecció

El tipus de selecció que s'ha implementat és el de **Selecció per Torneig**.

```
poblacio Seleccio_Torneig(int tournsize,poblacio pop)
{
    int i,j,ale1,ale2,numtorneig,numround,round;
    poblacio popfinal;

    round=popsize/2;
    numtorneig=1;
    i=0;

    while(numtorneig<=tournsize)
    {
        numround=0;
        while(numround<round)
        {
            //Selecció aleatòria sense reemplaçament
            ale1=intranuni(popsize);
            while(pop.P[ale1].triat==1) { ale1=intranuni(popsize); }
            pop.P[ale1].triat=1;

            ale2=intranuni(popsize);
            while((ale2==ale1) || (pop.P[ale2].triat==1))
            { ale2=intranuni(popsize); }
            pop.P[ale2].triat=1;

            //Competició
            if(pop.P[ale1].fitness>=pop.P[ale2].fitness)
            {
                popfinal.P[i].crom=pop.P[ale1].crom;
                popfinal.P[i].x=pop.P[ale1].x;
                popfinal.P[i].fitness=pop.P[ale1].fitness;
                popfinal.P[i].locus=pop.P[ale1].locus;
                popfinal.P[i].triat=0;
                popfinal.P[i].npreselec=ale1;
            }
            else {
                popfinal.P[i].crom=pop.P[ale2].crom;
                popfinal.P[i].x=pop.P[ale2].x;
                popfinal.P[i].fitness=pop.P[ale2].fitness;
                popfinal.P[i].locus=pop.P[ale2].locus;
                popfinal.P[i].triat=0;
                popfinal.P[i].npreselec=ale2;
            }
            numround++;
            i++;
        }
        for(j=0;j<popsize;j++) { pop.P[j].triat=0; }
        numtorneig++;
    }
    return popfinal;
}
```

S'observa com, en cada ronda d'un torneig, dos individus són triats aleatoriament, si no havien estat triats abans (ja es va esmentar que el torneig es feia sense reemplaçament) i que competeixen segons el valor de la seva funció objectiu. Observar que per a la selecció aleatòria s'usen funcions de generació aleatòria de nombres enters (*intranuni*), que seran explicades més endavant.

Creuament i Mutació

Aquest dos operadors genètics han estat implementats dins una mateixa funció anomenada Generació, el codi de la qual és:

```
poblacio Generacio(poblacio antpop,float sumfitness,float LB, float UB)
{
    int i,j,locus;
    float aux;
    poblacio noupop;
    cromosoma *noucrom;

    for(i=0;i<popsize;i=i+2)
    {
        //Creuament
        noucrom=
        Creuament(antpop.P[i].crom,antpop.P[i+1].crom,pcros,&locus,lcrom);
        //Mutació
        noupop.P[i].crom=Mutacio(noucrom[0],pmutacio,lcrom);
        noupop.P[i+1].crom=Mutacio(noucrom[1],pmutacio,lcrom);
        //Actualització
        for(j=0;j<2;j++)
        {
            aux=bit2float(noupop.P[i+j].crom,lcrom);
            noupop.P[i+j].x=Reescalat(aux,minstring,maxstring,LB,UB);
            noupop.P[i+j].fitness=fobj(noupop.P[i+j].x);
            noupop.P[i+j].parent1=antpop.P[i].npreselec;
            noupop.P[i+j].parent2=antpop.P[i+1].npreselec;
            noupop.P[i+j].locus=locus;
        }
    }
}
```

Com que en el procés de selecció ja s'han triat els individus de manera aleatòria, el procés de Generació s'efectua amb els individus d'aquesta població intermèdia agafats consecutivament de dos en dos. Les operacions de Creuament i Mutació actuen directament sobre els cromosomes associats a cada parell d'individus. Un cop creuats i mutats, cal tornar a obtenir el valor de la variable associat a cada cromosoma, ja que com que el cromosoma ha canviat, també haurà variat el valor de x que representa i per tant també el valor de la funció objectiu associada. Això és el que fa la part del codi que s'ha anomenat

Actualització. La part més important de l'actualització és la que consisteix en calcular el nou valor de x associat al cromossoma, assenyalada en vermell en el codi. La funció bit2float decodifica el cromossoma mentre que la funció Reescalat modifica x per tal que segueixi estan en l'espai inicial de cerca.

El codi concret per a les funcions de Creuament i Mutació és el següent:

```
cromosoma *Creuament
(cromosoma crom1,cromosoma crom2,float pcros,int *locus,int longcrom)
{
    int j,xsite;
    cromosoma *noucromos;
    noucromos=(cromosoma *)malloc(2*sizeof(cromosoma));

    if(flip(pcros))
    {
        xsite=intranuni(longcrom);
        *locus=xsite;
        //Caps
        for(j=0;j<xsite;j++) {
            noucromos[0].C[j]=crom1.C[j];
            noucromos[1].C[j]=crom2.C[j];
        }
        //Cues
        for(j=xsite;j<lcrom;j++) {
            noucromos[0].C[j]=crom2.C[j];
            noucromos[1].C[j]=crom1.C[j];
        }
    }
    else { noucromos[0]=crom1; noucromos[1]=crom2; *locus=0; }
    return noucromos;
}
```

La funció Creuament combina els bits de dos cromosomes en la manera com s'ha explicat al capítol 5, i ho fa segons una certa probabilitat que se li passa com a paràmetre. El punt de tall està generat aleatoriament. Retorna el parell de cromosomes un cop creuats.

```
cromosoma Mutacio(cromosoma crom,float pmutacio,int longcrom)
{
    int j;
    for(j=0;j<lcrom;j++)
    {
        if(flip(pmutacio)) crom.C[j]=!crom.C[j];
    }
    return crom;
}
```

La funció Mutacio actua sobre un sol cromossoma. El recorre bit a bit, mutant cadascun d'ells segons la probabilitat de mutació, que també es passa com a paràmetre.

Tant en aquesta funció com en l'anterior són molt importants les funcions que tenen a veure amb generació de nombres aleatoris (`intranuni`, `flip`), al igual que en la selecció. Totes aquestes funcions es comentaran més endavant.

Reinscripció

La funció Reinscripció és la més senzilla de tots els operadors genètics, i tal com ja s'ha explicat consisteix en reintroduir a la població aquell individu que era el millor abans de realitzar les operacions de selecció, creuament i mutació.

```
void Reinscripcio
(poblacio antpop,poblacio noupop,int indmaxact,int indmin)
{
    noupop.P[indmin]=antpop.P[indmaxact];
}
```

Com es pot observar, a la funció se li passen la població anterior (`antpop`) i la modificada (`noupop`) i els indexos del millor individu de la població anterior (`indmaxact`) i del pitjor de la nova població (`indmin`). La funció l'únic que fa és substituir un per l'altre en la població nova. Per calcular aquests indexos i altres elements d'estadística descriptiva de la població s'ha implementat una funció: la funció Estadístic, que es veurà més endavant.

Iniciització

Si es recorda l'estructura general d'un algorisme genètic, el primer pas era el de la Iniciització que consistia en obtenir una població inicial de manera aleatòria. Això és el que s'aconsegueix amb la funció `IniPop`.

```

poblacio IniPop(float LB,float UB)
{
    poblacio antpop;
    int i,j;
    float aux;

    for(i=0;i<popsize;i++)
    {
        for(j=0;j<lcrom;j++) antpop.P[i].crom.C[j]=flip(0.5);
        aux=bit2float(antpop.P[i].crom,lcrom);
        antpop.P[i].x=Reescalat(aux,minstring,maxstring,LB,UB);
        antpop.P[i].fitness=fobj(antpop.P[i].x);
        antpop.P[i].parent1=0;
        antpop.P[i].parent2=0;
        antpop.P[i].locus=0;
    }
    return antpop;
}

```

Aquesta funció crea una població inicial de cromosomes aleatoriament, usant la funció flip abans esmentada, que no és més que la generació d'un 1 o un 0 de manera aleatòria. Així és com s'obté de manera aleatòria la població inicial. La resta de parts de IniPop el que fan és passar els valors del cromosomes a valors reals, i reescalar-los per situar-los a l'espai de cerca que interessa que s'haurà introduït prèviament com a paràmetre (LB, UB). A més es calcula el valor de la funció objectiu associat a aquest valor de x obtingut.

Decodificació, reescalat i funció objectiu

Aquestes tres funcions són importants perquè permeten passar de treballar amb cromosomes a treballar amb valors reals, que són de fet els que es busquen com a solució al problema d'optimització plantejat.

Decodificació

La funció que passa de cromossomes a valors reals és la funció `bit2float`, el codi de la qual és:

```
float bit2float(cromosoma crom,int longcrom)
{
    int j;
    float acum,powerof2;

    acum=0.0;
    powerof2=1.0;
    for(j=0;j<longcrom;j++)
    {
        if(crom.C[j]==1) { acum=acum+powerof2; }
        powerof2=powerof2*2;
    }
    return acum;
}
```

Recordar que per la manera com està feta la funció la codificació en bits s'entén que és d'esquerra a dreta. Amb això es vol dir que si la taula de bits és $[a_0a_1a_2a_3a_4a_5a_6a_7]$ amb $a_i \in \{0,1\}$, està representant el nombre

$$a_0 \cdot 2^0 + a_1 \cdot 2^1 + a_2 \cdot 2^2 + a_3 \cdot 2^3 + a_4 \cdot 2^4 + a_5 \cdot 2^5 + a_6 \cdot 2^6 + a_7 \cdot 2^7$$

Reescalat

Al generar els cromossomes aleatoriament o bé la passar-lo pels operadors genètics, el valor de la variable x que tenen associat pot ser qualsevol entre 0 (valor associat a la cadena de bits amb tot de zeros) i $2^{lcrom} - 1$ (valor associat a la cadena de bits amb tot de uns), on $lcrom$ és la longitud del cromosoma fixada a priori. Com que la optimització es vol realitzar en un cert espai (pel cas d'una funció amb una sola variable, en un interval, $[LB, UB]$), cal reescalar els valors obtinguts per traslladar-los a aquest espai. Aquest és l'objectiu de la funció Reescalat.

```

float Reescalat
(float x, float antinf, float antsup,float nouinf,float nousup)
{
    float a,b;

    a=(nousup-nouinf)/(antsup-antinf);
    b=nouinf-a*antinf;
    return a*x+b;
}

```

Com que el que es voldrà sempre que s'usa aquesta funció és passar de l'espai $[0, 2^{lchrom} - 1]$ a l'espai $[LB, UB]$, els paràmetres de la funció prendran els valors següents:

$$\text{antinf}=0 \quad \text{antsup}=2^{lchrom} - 1 \quad \text{nouinf}=LB \quad \text{nousup}=UB$$

Funció objectiu

A tall d'exemple s'han provat per aquest algorisme genètic simple les tres funcions objectiu següents:

```

float fobj(float x)
{
    int n;
    n=2;

    return (float) (pow(x, n));
}

float fobj(float x)
{
    int n;

    return (float) (sin(x));
}

float fobj(float x)
{
    if((x>=0) & (x<1)) return (float) x;
    else if((x>=1) & (x<2)) return (float) 2-x;
    else if((x>=2) & (x<3)) return (float) 2*x-4;
    else if((x>=3) & (x<4)) return (float) -2*x+8;
    else return (float) 0;
}

```

Pas d'una generació a l'altra

La part de l'algorisme que efectua el pas de generació en generació, és a dir, que combina tots els operadors genètics, és la següent:

Iteració

```

while(CriteriParada(stop,numgen,maxgen,maxact,minact)==0)
{
    //Selecció la població que passara pels operadors genetics
    antpop=Seleccio_Torneig(2,antpop);
    //Generació
    noupop=Generacio(antpop,sumfitness,LB,UB);
    //Estadístiques de la nova població
    Estadistic(noupop,&indmax,&indmin,&avg,&min,&max,&sumfitness);
    //Reinsercio si es que es vol i calcul de les noves estadístiques
    if(reinser==1)
    {
        Reinsercio(antpop,noupop,indmaxact,indmin);
        Estadistic(noupop,&indmax,&indmin,&avg,&min,&max,&sumfitness);
    }

    //Sortida de resultats per cada iteració
    Sortida(noupop,numgen,avg,min,max,sumfitness);
    fprintf(dadgraf,"%f %f\n",max,avg);
    //Actualització de la població
    antpop=noupop;
    //Pel control del millors i pitjors individus d'una població
    maxact=max; avgact=avg; minact=min;
    indmaxact=indmax; indminact=indmin;

    //Augment del nombre de generació
    numgen++;
}

```

Criteri de parada

El criteri de parada funciona tal com s'ha explicat al capítol 5: si en una generació concreta el pitjor valor de la funció objectiu i el millor són prou iguals, l'algorisme s'atura i si no, segueix fins un nombre màxim d'iteracions que s'ha fixat a priori.

```
int CriteriParada
(float stop,int numgen,int maxgen,float maxact,float minact)
{
    float rati;
    if(triestop==1)
    {
        if(numgen>maxgen) {
            fprintf(dadw,"\\nNombre d'iteracions realitzades: %d\\n",numgen-1);
            return 1;
        }
        else return 0;
    }
    else {
        rati=minact/maxact;
        if(rati>=stop) {
            fprintf(dadw,"\\nHem parat pel criteri de convergencia.
                \\nNºiteracions:%d\\n",numgen-1);
            return 1;
        }
        else return 0;
    }
}
```

Funcions auxiliars

Estadístiques

Aquesta és una funció important perquè permet trobar, en cada generació, l'individu amb el valor optim de la funció. Serà aquest el que, per l'última generació, es donarà com a òptim.

Aquesta funció permet a més seguir si en mitjana s'està millorant en valor de la funció objectiu, cosa que servirà per fer un seguiment gràfic de la convergència de l'algorisme, com es veurà més endavant.

```

void Estadistic
(poblacio pop,int *indmax,int *indmin,float *avg,float *min,float *max,
 float *sumfitness)
{
    int i;

    *indmax=0;
    *indmin=0;
    *sumfitness=pop.P[0].fitness;
    *min=pop.P[0].fitness;
    *max=pop.P[0].fitness;

    for(i=1;i<popsiz; i++)
    {
        *sumfitness=*sumfitness+pop.P[i].fitness;
        //Calcul del màxim
        if(pop.P[i].fitness>*max) { *max=pop.P[i].fitness; *indmax=i; }
        //Calcul del mínim
        if(pop.P[i].fitness<*min) { *min=pop.P[i].fitness; *indmin=i; }
    }
    //Mitjana
    *avg=*sumfitness/popsiz;
}

```

Funcions d'aleatorietat

Tal com ja s'ha esmentat en varíes ocasions i a causa del caràcter estocàstic dels algorismes genètics, les funcions de generació de nombres aleatoris són molt importants. En tota la implementació es fa ús de tres funcions d'aquest tipus, que es presenten tot seguit:

```

float ranuni(float LB, float UB)
{
    enter32 a,c;
    float sol;
    a=3715318133;
    c=1;
    llavor_enter=a*llavor_enter+c;
    sol=(float)(1.0*llavor_enter);
    return LB+(sol/float_232)*(UB-LB); // entre LB i UB
}

```

Aquesta funció genera nombres reals aleatoris entre LB i UB. És un generador de nombres aleatoris (de fet pseudoaleatoris) del tipus conegut com *Generador Congruencial Lineal* construit segons [ref]. La qualitat del generador depèn de l'elecció dels paràmetres *a* i *c*. A []

es troben detallats alguns estudis teòrics que determinen les condicions per la selecció dels valors d'aquests paràmetres.

```
int intranuni(int fitsup)
{
    int x,aux;
    aux=(int)floor(float_232*ranuni(0.0,1.0));
    srand((unsigned)aux);
    x=(int)((1.0*rand())/(1.0*RAND_MAX))*fitsup;
    return x;
}
```

Aquesta funció usa l'anterior per obtenir nombres enters aleatoris entre 0 i fitsup.

```
int flip(float p)
{
    if(p-1.0==0)
        return 1;
    return (ranuni(0.0,1.0)<=p);
}
```

Funció que retorna un 1 o un 0 de manera aleatòria segons el valor d'una certa probabilitat. Si $p=0.5$ la funció simula el llançament d'una moneda, en que és igual de probable que surti cara (0) com creu (1).

Existeixen altres funcions auxiliars, bàsicament funcions d'entrada de paràmetres i d'escriptura de resultats que no tenen interès de cara a l'anàlisi de l'algorisme.

A l'Annex2 es presenta el codi complet del programa que a partir d'aquest s'ha usat per implementar el TSD. Allà apareixen també aquestes funcions auxiliars.

A1.2 Exemples d'execució

S'executará el programa amb tres funcions objectius:

- $f(x) = x^2$: una funció senzilla amb el màxim en un extrem de l'interval de cerca.
- $f(x) = \sin x$: una funció també senzilla però que té el punt màxim en un punt intermig de l'espai de cerca.

$$\bullet \quad f(x) = \begin{cases} x & x \in [0,1) \\ 2 - x & x \in [1,2) \\ 2 \cdot x - 4 & x \in [2,3) \\ 8 - 2 \cdot x & x \in [3,4) \\ 0 & \text{altrament} \end{cases}$$

Una funció definida a trossos amb dos màxims locals, dels quals cal detectar-ne el global.

Funció x^2

Es vol trobar el màxim daquesta funció dins un cert interval, que per exemple es fixa a $[-1.5, 5]$.

Evidentement el màxim es troba precisament en l'extrem superior de l'interval $x=5$, amb un valor de 25.

Els paràmetres d'entrada que el programa AGS. c permet i els valors que se'ls hi han assignat en aquest cas són els de la taula A1.1.

Paràmetre	Valor
Espai de cerca: límit inferior	-1.5
Espai de cerca: límit superior	5
Grandària de la població	20
Probabilitat de creuament	1
Probabilitat de mutació	0.02
Criteri de parada	Rati
Rati de convergència	1
Reinserció?	Si

Taula A1.1

La manera com s'introdueixen els paràmetres queda reflectida a la figura A1.1, en la qual es veu la pantalla d'entrada i el resultat que surt per pantalla després d'executar el programa: és només el temps d'execució. La resta de resultats surten en arxius complementaris que es veuran tot seguit.

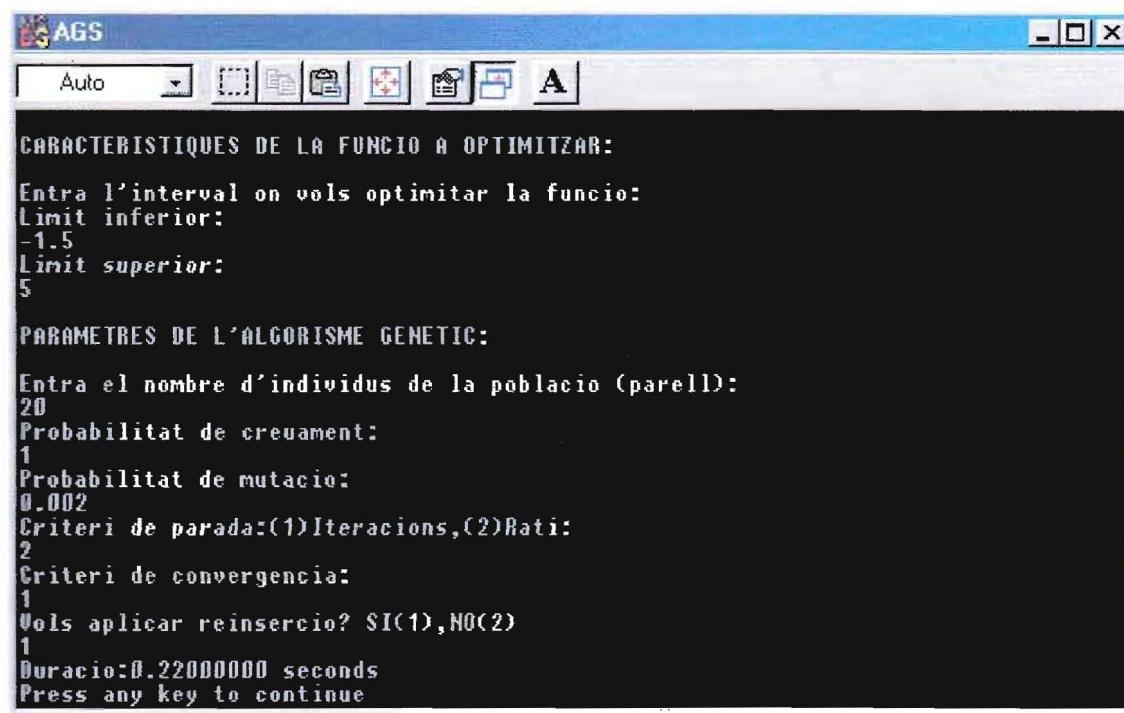


Figura A1.1

Per una banda els resultats de totes les iteracions de l'algorisme i el punt òptim s'escriuen en l'arxiu res.txt, i són les següents:

POBLACIO INICIAL			
	Cromossoma	x	f(x)
0	01110000	-1.14313722	1.30676270
1	11100011	3.57254910	12.76310730
2	11001111	4.69411755	22.03474045
3	01000001	1.81372559	3.28960061
4	10110011	3.72549033	13.87927818
5	00111101	3.29215693	10.83829689
6	11010100	-0.40392154	0.16315262
7	11111000	-0.70980394	0.50382161
8	11111011	4.18431377	17.50848198
9	00000010	0.13137257	0.01725875
10	10000100	-0.65882349	0.43404838
11	00101000	-0.99019605	0.98048824
12	01010010	0.38627455	0.14920802
13	10001001	2.19607854	4.82276106
14	10011010	0.76862746	0.59078819
15	11011011	4.08235312	16.66560745
16	00011001	2.37450981	5.63829708
17	01011010	0.79411769	0.63062292
18	00001011	3.80196095	14.45490742

19	11011101	3.26666665	10.67111111
----	----------	------------	-------------

ESTADISTIQUES INICIALS:**Mitja:** 6.867118**Max:** 22.034740**Min:** 0.017259**Sumfitness:** 137.342361**-----GENERACIO 1-----**

	Pares	Locus	Cromossoma	x	f(x)
0	(4,18)	1	10001011	3.82745099	14.64938068
1	(4,18)	1	00110011	3.70000005	13.69000053
2	(13,8)	1	11111011	4.18431377	17.50848198
3	(13,8)	1	10001001	2.19607854	4.82276106
4	(14,7)	6	10011000	-0.86274511	0.74432909
5	(14,7)	6	11111010	0.92156863	0.84928876
6	(2,15)	3	11011011	4.08235312	16.66560745
7	(2,15)	3	11001111	4.69411755	22.03474045
8	(16,19)	0	11011101	3.26666665	10.67111111
9	(16,19)	0	00011001	2.37450981	5.63829708
10	(7,8)	6	11111001	2.55294132	6.51750946
11	(7,8)	6	11111000	-0.70980394	0.50382161
12	(4,18)	1	10001111	4.64313745	21.55872536
13	(4,18)	1	00110011	3.70000005	13.69000053
14	(17,2)	4	01011111	4.87254906	23.74173355
15	(17,2)	4	11001010	0.61568630	0.37906963
16	(5,15)	6	00111111	4.92352962	24.24114418
17	(5,15)	6	11011001	2.45098042	6.00730515
18	(19,1)	7	11011101	3.26666665	10.67111111
19	(19,1)	7	11100011	3.57254910	12.76310730

ESTADISTIQUES NOVES:**Mitja:** 11.367376**Max:** 24.241144**Min:** 0.379070**Sumfitness:** 227.347534**-----GENERACIO 2-----**

	Pares	Locus	Cromossoma	x	f(x)
0	(19,7)	0	11001111	4.69411755	22.03474045
1	(19,7)	0	11100011	3.57254910	12.76310730
2	(14,16)	0	00011111	4.82156849	23.24752235
3	(14,16)	0	01011111	4.87254906	23.74173355
4	(6,13)	6	11011011	4.08235312	16.66560745
5	(6,13)	6	00110011	3.70000005	13.69000053
6	(2,18)	4	11111101	3.36862755	11.34765148
7	(2,18)	4	11011011	4.08235312	16.66560745
8	(0,1)	1	10110011	3.72549033	13.87927818
9	(0,1)	1	00001011	3.80196095	14.45490742
10	(16,0)	2	00001011	3.80196095	14.45490742
11	(16,0)	2	10111111	4.94901991	24.49279785
12	(2,19)	6	11111011	4.18431377	17.50848198
13	(2,19)	6	11100011	3.57254910	12.76310730
14	(12,6)	4	10001011	3.82745099	14.64938068
15	(12,6)	4	11011111	4.89803934	23.99078941
16	(1,14)	1	01011111	4.87254906	23.74173355
17	(1,14)	1	00110011	3.70000005	13.69000053
18	(13,18)	2	00011101	3.19019604	10.17735100
19	(13,18)	2	11100011	3.77647066	14.26173019

ESTADISTIQUES NOVES:

Mitja: 16.911022

Max: 24.492798

Min: 10.177351

Sumfitness: 338.220428

-----GENERACIO 3-----

	Pares	Locus	Cromossoma	x	f(x)
0	(9, 7)	6	00001011	3.80196095	14.45490742
1	(9, 7)	6	11011011	4.08235312	16.66560745
2	(11, 2)	7	10111111	4.94901991	24.49279785
3	(11, 2)	7	00011111	4.82156849	23.24752235
4	(15, 14)	5	11011011	4.08235312	16.66560745
5	(15, 14)	5	10001111	4.64313745	21.55872536
6	(3, 8)	4	01010011	3.64901972	13.31534481
7	(3, 8)	4	10111111	4.94901991	24.49279785
8	(0, 16)	1	11011111	4.89803934	23.99078941
9	(0, 16)	1	01001111	4.66862774	21.79608536
10	(12, 6)	6	11111001	2.55294132	6.51750946
11	(12, 6)	6	11111111	5.00000000	25.00000000
12	(9, 15)	3	00011111	4.82156849	23.24752235
13	(9, 15)	3	11001011	3.87843156	15.04223156
14	(11, 19)	3	10110011	3.72549033	13.87927818
15	(11, 19)	3	11111111	5.00000000	25.00000000
16	(3, 2)	5	01011111	4.87254906	23.74173355
17	(3, 2)	5	00011111	4.82156849	23.24752235
18	(0, 7)	3	11011011	4.08235312	16.66560745
19	(0, 7)	3	11001111	4.69411755	22.03474045

ESTADISTIQUES NOVES:

Mitja: 19.752817

Max: 25.000000

Min: 6.517509

Sumfitness: 395.056335

-----GENERACIO 4-----

	Pares	Locus	Cromossoma	x	f(x)
0	(2, 12)	5	10111111	4.94901991	24.49279785
1	(2, 12)	5	00011111	4.82156849	23.24752235
2	(7, 13)	7	10111111	4.94901991	24.49279785
3	(7, 13)	7	11001011	3.87843156	15.04223156
4	(19, 15)	0	11111111	5.00000000	25.00000000
5	(19, 15)	0	11001111	4.69411755	22.03474045
6	(11, 5)	1	10001111	4.64313745	21.55872536
7	(11, 5)	1	11111111	5.00000000	25.00000000
8	(8, 3)	1	10011111	4.84705877	23.49397850
9	(8, 3)	1	01011111	4.87254906	23.74173355
10	(15, 7)	6	11111111	5.00000000	25.00000000
11	(15, 7)	6	10111111	4.94901991	24.49279785
12	(12, 8)	4	00011111	4.82156849	23.24752235
13	(12, 8)	4	11011111	4.89803934	23.99078941
14	(19, 11)	6	11001111	4.69411755	22.03474045
15	(19, 11)	6	11111111	5.00000000	25.00000000
16	(17, 9)	5	00011111	4.82156849	23.24752235
17	(17, 9)	5	01001111	4.66862774	21.79608536
18	(4, 2)	2	11111111	5.00000000	25.00000000
19	(4, 2)	2	10011011	4.03137255	16.25196457

ESTADISTIQUES NOVES:

Mitja: 22.908297

Max: 25.000000

Min: 15.042232

Sumfitness: 458.165924

-----GENERACIO 5-----

	Pares	Locus	Cromossoma	x	f(x)
0	(1,15)	4	00011111	4.82156849	23.24752235
1	(1,15)	4	11111111	5.00000000	25.00000000
2	(2,4)	4	10111111	4.94901991	24.49279785
3	(2,4)	4	11111111	5.00000000	25.00000000
4	(9,5)	3	01001111	4.66862774	21.79608536
5	(9,5)	3	11011111	4.89803934	23.99078941
6	(0,10)	7	10111111	4.94901991	24.49279785
7	(0,10)	7	11111111	5.00000000	25.00000000
8	(12,18)	2	00111111	4.92352962	24.24114418
9	(12,18)	2	11011111	4.89803934	23.99078941
10	(1,7)	5	00011111	4.82156849	23.24752235
11	(1,7)	5	11111111	5.00000000	25.00000000
12	(4,2)	1	10111111	4.94901991	24.49279785
13	(4,2)	1	11111111	5.00000000	25.00000000
14	(15,16)	7	11111111	5.00000000	25.00000000
15	(15,16)	7	00011111	4.82156849	23.24752235
16	(18,14)	1	11001111	4.69411755	22.03474045
17	(18,14)	1	11111111	5.00000000	25.00000000
18	(9,0)	6	01011111	4.87254906	23.74173355
19	(9,0)	6	10111111	4.94901991	24.49279785

ESTADISTIQUES NOVES:

Mitja: 24.125452

Max: 25.000000

Min: 21.796085

Sumfitness: 482.509033

-----GENERACIO 6-----

	Pares	Locus	Cromossoma	x	f(x)
0	(5,3)	6	11011111	4.89803934	23.99078941
1	(5,3)	6	11111101	3.36862755	11.34765148
2	(1,7)	4	11111111	5.00000000	25.00000000
3	(1,7)	4	11111111	5.00000000	25.00000000
4	(2,14)	3	10111111	4.94901991	24.49279785
5	(2,14)	3	11111111	5.00000000	25.00000000
6	(17,6)	4	11111111	5.00000000	25.00000000
7	(17,6)	4	10111111	4.94901991	24.49279785
8	(11,13)	1	11111111	5.00000000	25.00000000
9	(11,13)	1	11111111	5.00000000	25.00000000
10	(12,3)	2	10111111	4.94901991	24.49279785
11	(12,3)	2	11111111	5.00000000	25.00000000
12	(14,17)	1	11111111	5.00000000	25.00000000
13	(14,17)	1	11111111	5.00000000	25.00000000
14	(7,9)	2	11011111	4.89803934	23.99078941
15	(7,9)	2	11111111	5.00000000	25.00000000
16	(8,1)	5	00111111	4.92352962	24.24114418
17	(8,1)	5	11111111	5.00000000	25.00000000
18	(13,11)	3	11111111	5.00000000	25.00000000
19	(13,11)	3	11111111	5.00000000	25.00000000

ESTADISTIQUES NOVES:

Mitja: 24.102438

Max: 25.000000

Min: 11.347651

Sumfitness: 482.048767

-----GENERACIO 7-----

	Pares	Locus	Cromossoma	x	f(x)
0	(6,17)	6	11111111	5.00000000	25.00000000
1	(6,17)	6	11111111	5.00000000	25.00000000
2	(4,19)	6	10111111	4.94901991	24.49279785
3	(4,19)	6	11111111	5.00000000	25.00000000
4	(9,13)	1	11111111	5.00000000	25.00000000
5	(9,13)	1	11111111	5.00000000	25.00000000
6	(5,11)	7	11111111	5.00000000	25.00000000
7	(5,11)	7	11111111	5.00000000	25.00000000
8	(18,3)	0	11111111	5.00000000	25.00000000
9	(18,3)	0	11111111	5.00000000	25.00000000
10	(17,2)	7	11111111	5.00000000	25.00000000
11	(17,2)	7	11111111	5.00000000	25.00000000
12	(4,8)	6	10111111	4.94901991	24.49279785
13	(4,8)	6	11111111	5.00000000	25.00000000
14	(3,9)	1	11111111	5.00000000	25.00000000
15	(3,9)	1	11111111	5.00000000	25.00000000
16	(5,15)	4	11111111	5.00000000	25.00000000
17	(5,15)	4	11111111	5.00000000	25.00000000
18	(7,19)	2	10111111	4.94901991	24.49279785
19	(7,19)	2	11111111	5.00000000	25.00000000

ESTADISTIQUES NOVES:

Mitja: 24.923920

Max: 25.000000

Min: 24.492798

Sumfitness: 498.478394

-----GENERACIO 8-----

	Pares	Locus	Cromossoma	x	f(x)
0	(9,3)	0	11111111	5.00000000	25.00000000
1	(9,3)	0	11011111	4.89803934	23.99078941
2	(14,19)	5	11111111	5.00000000	25.00000000
3	(14,19)	5	11111111	5.00000000	25.00000000
4	(16,10)	7	11111111	5.00000000	25.00000000
5	(16,10)	7	11111111	5.00000000	25.00000000
6	(4,6)	2	11111111	5.00000000	25.00000000
7	(4,6)	2	11111111	5.00000000	25.00000000
8	(1,13)	4	11111111	5.00000000	25.00000000
9	(1,13)	4	11111111	5.00000000	25.00000000
10	(10,4)	3	11111111	5.00000000	25.00000000
11	(10,4)	3	11111111	5.00000000	25.00000000
12	(0,14)	7	11111111	5.00000000	25.00000000
13	(0,14)	7	11111111	5.00000000	25.00000000
14	(16,17)	7	11111111	5.00000000	25.00000000
15	(16,17)	7	11111111	5.00000000	25.00000000
16	(9,6)	7	11111111	5.00000000	25.00000000
17	(9,6)	7	11111111	5.00000000	25.00000000
18	(7,5)	7	11111111	5.00000000	25.00000000
19	(7,5)	7	11111111	5.00000000	25.00000000

ESTADISTIQUES NOVES:

Mitja: 24.949539

Max: 25.000000

Min: 23.990789

Sumfitness: 498.990784

-----GENERACIO 9-----

	Pares	Locus	Cromossoma	x	f(x)
0	(2,11)	6	11111111	5.00000000	25.00000000
1	(2,11)	6	11111111	5.00000000	25.00000000
2	(6,3)	6	11111111	5.00000000	25.00000000
3	(6,3)	6	11111111	5.00000000	25.00000000
4	(15,19)	0	11111111	5.00000000	25.00000000
5	(15,19)	0	11111111	5.00000000	25.00000000
6	(5,14)	6	11111111	5.00000000	25.00000000
7	(5,14)	6	11111111	5.00000000	25.00000000
8	(7,12)	3	11111111	5.00000000	25.00000000
9	(7,12)	3	11111111	5.00000000	25.00000000
10	(15,13)	4	11111111	5.00000000	25.00000000
11	(15,13)	4	11111111	5.00000000	25.00000000
12	(10,19)	2	11111111	5.00000000	25.00000000
13	(10,19)	2	11111111	5.00000000	25.00000000
14	(16,4)	4	11111111	5.00000000	25.00000000
15	(16,4)	4	11111111	5.00000000	25.00000000
16	(8,12)	3	11111111	5.00000000	25.00000000
17	(8,12)	3	11111111	5.00000000	25.00000000
18	(18,11)	5	11111111	5.00000000	25.00000000
19	(18,11)	5	11111111	5.00000000	25.00000000

ESTADISTIQUES NOVES:

Mitja: 25.000000

Max: 25.000000

Min: 25.000000

Sumfitness: 500.000000

Hem parat pel criteri de convergencia.

Nº iteracions: 9

OPTIM:

x: 5.00000000 f: 25.00000000

En els resultats s'observa com l'algorisme ha trobat l'òptim esperat en 9 iteracions. D'una generació a una altra es pot anar veient com els cromossomes es combinen acostant-se cada cop més al cromossoma 11111111 que és el cromossoma òptim. Els cromossomes que no s'assemblen a aquest van despareixen i els que tenen un esquema o plantilla semblant a l'òptim són els que van sobrevivint i recombinant-se fins arribar a l'òptim.

El programa genera un altre arxiu de resultats, resgraf.txt, en el qual es guarden, per cada iteració, el valor màxim de la funció objectiu obtingut en aquella generació i el valor mig de la funció objectiu calculat amb tots els individus de la generació.

Els resultats de l'arxiu resgraf.txt es presenten a continuació de manera numèrica i gràfica.

```
//Màxim //Mitjana
22.034740 6.867118
24.241144 11.367376
24.492798 16.911022
25.000000 19.752817
25.000000 22.908297
25.000000 24.125452
25.000000 24.102438
25.000000 24.923920
25.000000 24.949539
25.000000 25.000000
```

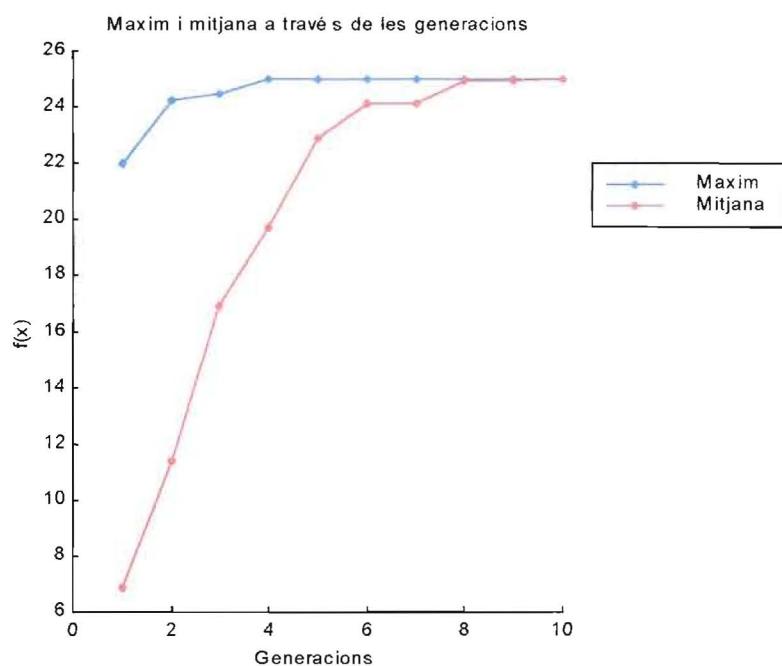


Figura A1.2

En el gràfic s'aprecia com efectivament es dóna la convergència de l'algorisme. Encara que des de la quarta generació l'òptim ja s'havia assolit com a valor màxim, el que es desitja és obtenir la convergència en mitjana per assegurar un òptim el més global possible.

Funció Sinus

Es vol trobar el màxim de la funció sinus en l'interval $[0, \pi]$. Analíticament es troba que aquest màxim és el punt $x = \frac{\pi}{2} = 1.5708$ amb valor màxim 1.

Els paràmetres d'entrada en aquest cas són els de la taula A1.2 i la pantalla d'entrada del programa junt amb el temps d'execució a la figura A1.3.

Paràmetre	Valor
Espai de cerca: límit inferior	0
Espai de cerca: límit superior	3.14159265
Grandària de la població	50
Probabilitat de creuament	1
Probabilitat de mutació	0.003
Criteri de parada	Rati
Rati de convergència	1
Reinserció?	Si

Taula A1.2

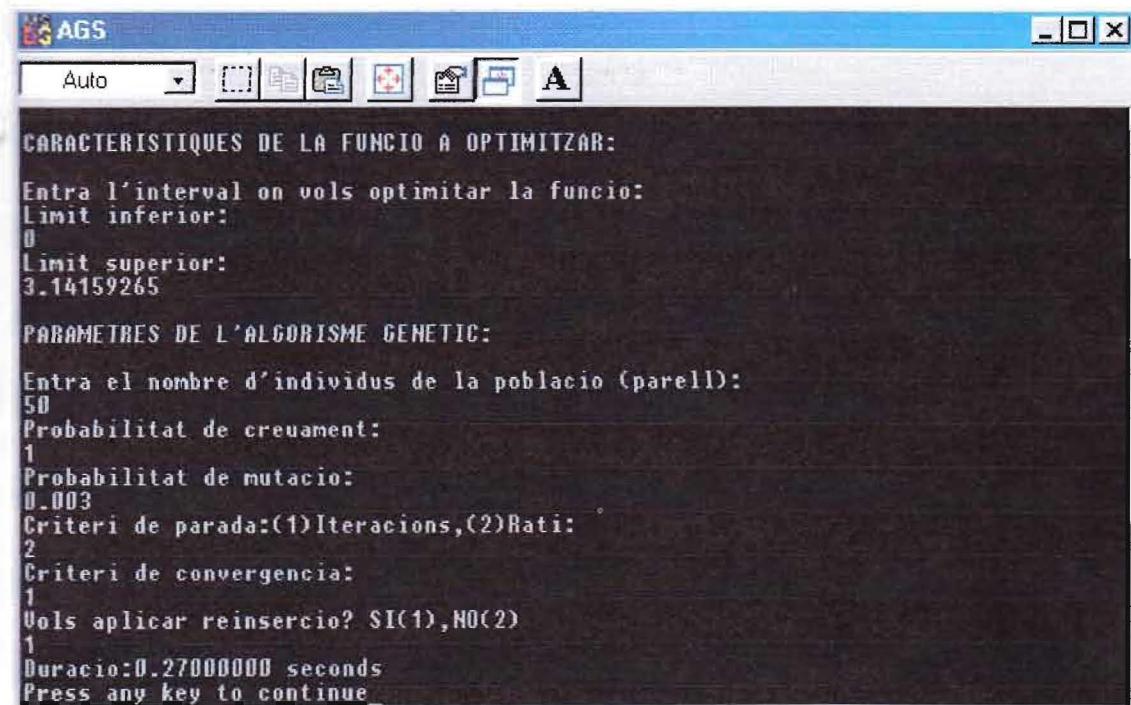


Figura A1.3

Dels resultats obtinguts en aquest cas, com que la població és de més grandària, només es presenten els de les dues primeres generacions i les dues últimes, encara que l'arxiu res.txt conté la informació per totes les generacions.

POBLACIO INICIAL

	Cromossoma	x	f(x)
0	10001010	0.99791771	0.84034407
1	10011001	1.88495564	0.95105648
2	11011001	1.90959561	0.94315439
3	11011010	1.12111747	0.90058672
4	11100101	2.05743527	0.88390964
5	11111111	3.14159274	0.00000009
6	11111111	3.14159274	0.00000009
7	11111010	1.17039728	0.92090553
8	10001000	0.20943952	0.20791170
9	01000101	1.99583542	0.91102260
10	00110111	2.90751338	0.23194747
11	00111011	2.71039391	0.41796014
12	11100101	2.05743527	0.88390964
13	00010100	0.49279886	0.47309357
14	11001111	2.99375319	0.14730151
15	01111111	3.12927294	0.01231940
16	10000001	1.58927631	0.99982923
17	00111111	3.10463285	0.03695139
18	00001110	1.37983680	0.98182255
19	10001010	0.99791771	0.84034407
20	01000010	0.81311816	0.72643363
21	10111100	0.75151831	0.68274891
22	00000100	0.39423910	0.38410577
23	00011000	0.29567933	0.29138976
24	11011001	1.90959561	0.94315439
25	00010010	0.88703799	0.77520400
26	01000001	1.60159636	0.99952573
27	01010111	2.88287330	0.25584275
28	01110110	1.35519695	0.97684836
29	10011111	3.06767297	0.07385239
30	10000011	2.37775445	0.69169843
31	10011101	2.27919483	0.75940484
32	10111111	3.11695290	0.02463727
33	10110001	1.73711598	0.98620075
34	11100000	0.08623980	0.08613294
35	00110101	2.11903524	0.85344368
36	10010011	2.47631431	0.61727816
37	11000000	0.03695992	0.03695150
38	01001000	0.22175950	0.21994637
39	00110101	2.11903524	0.85344368
40	10111001	1.93423557	0.93467975
41	00000101	1.97119546	0.92090547
42	11110111	2.94447327	0.19584532
43	00001011	2.56255412	0.54721946
44	01101101	2.24223495	0.78292751
45	10111010	1.14575744	0.91102272
46	00011010	1.08415747	0.88390970
47	11001001	1.81103587	0.97128099
48	00011000	0.29567933	0.29138976
49	01110001	1.74943602	0.98408633

ESTADISTIQUES INICIALS:

Mitja: 0.604918

Max: 0.999829

Min: 0.000000

Sumfitness: 30.245878

-----GENERACIO 1-----

	Pares	Locus	Cromossoma	x	f(x)
0	(11,1)	3	00111001	1.92191565	0.93898833
1	(11,1)	3	10011011	2.67343378	0.45124403
2	(35,7)	2	00111010	1.13343740	0.90587342
3	(35,7)	2	11110101	2.15599513	0.83360231
4	(39,17)	1	00111111	3.10463285	0.03695139
5	(39,17)	1	00110101	2.11903524	0.85344368
6	(49,33)	0	10110001	1.73711598	0.98620075
7	(49,33)	0	01110011	2.53791428	0.56767452
8	(24,44)	4	11011101	2.30383468	0.74314481
9	(24,44)	4	01101001	1.84799576	0.96182561
10	(4,16)	6	11100101	2.05743527	0.88390964
11	(4,16)	6	10000000	0.01231997	0.01231966
12	(0,41)	2	10000101	1.98351550	0.91603357
13	(0,41)	2	00001010	0.98559773	0.83360243
14	(19,14)	6	10001011	2.57487416	0.53686643
15	(19,14)	6	11001110	1.41679680	0.98816550
16	(48,43)	7	00011001	1.87263572	0.95479131
17	(48,43)	7	00001010	0.98559773	0.83360243
18	(26,30)	4	01000011	2.39007449	0.68274879
19	(26,30)	4	10000001	1.58927631	0.99982923
20	(20,36)	3	01010011	2.48863435	0.60753882
21	(20,36)	3	10000010	0.80079818	0.71791196
22	(18,47)	4	00001001	1.77407598	0.97940975
23	(18,47)	4	11001110	1.41679680	0.98816550
24	(45,44)	0	01101101	2.24223495	0.78292751
25	(45,44)	0	10111010	1.14575744	0.91102272
26	(4,26)	1	11000001	1.61391628	0.99907047
27	(4,26)	1	01100101	2.04511523	0.88960403
28	(25,40)	0	10111001	1.93423557	0.93467975
29	(25,40)	0	10010010	0.89935791	0.78292763
30	(21,49)	1	11110001	1.76175594	0.98182255
31	(21,49)	1	10111100	0.75151831	0.68274891
32	(42,46)	6	11110110	1.36751688	0.97940975
33	(42,46)	6	00011011	2.66111398	0.46220371
34	(33,31)	4	10111101	2.32847476	0.72643346
35	(33,31)	4	10010001	1.68783617	0.99315864
36	(16,2)	0	11011001	1.90959561	0.94315439
37	(16,2)	0	10000001	1.58927631	0.99982923
38	(7,18)	4	11111110	1.56463647	0.99998105
39	(7,18)	4	00001010	0.98559773	0.83360243
40	(23,12)	2	00100101	2.02047539	0.90058666
41	(23,12)	2	11011000	0.33263925	0.32653874
42	(24,41)	4	11010101	2.10671520	0.85979980
43	(24,41)	4	00001001	1.77407598	0.97940975
44	(35,47)	1	01001001	1.79871583	0.97413862
45	(35,47)	1	10110101	2.13135505	0.84695822
46	(1,14)	2	10001111	2.96911311	0.17162563
47	(1,14)	2	11011001	1.90959561	0.94315439
48	(43,28)	5	00001110	1.37983680	0.98182255
49	(43,28)	5	01110011	2.53791428	0.56767452

ESTADISTIQUES NOVES:**Mitja:** 0.793362**Max:** 0.999981**Min:** 0.012320**Sumfitness:** 39.668125**-----GENERACIO 2-----**

	Pares	Locus	Cromossoma	x	f(x)
0	(36, 43)	7	11011001	1.90959561	0.94315439
1	(36, 43)	7	00001101	2.16831493	0.82673419
2	(26, 9)	0	01101001	1.84799576	0.96182561
3	(26, 9)	0	11000001	1.61391628	0.99907047
4	(30, 39)	3	11101010	1.07183754	0.87808126
5	(30, 39)	3	00010001	1.67551613	0.99452192
6	(5, 33)	0	00011011	2.66111398	0.46220371
7	(5, 33)	0	00110101	2.11903524	0.85344368
8	(27, 6)	6	01100101	2.04511523	0.88960403
9	(27, 6)	6	10110001	1.73711598	0.98620075
10	(15, 0)	7	11001111	2.99375319	0.14730151
11	(15, 0)	7	00111000	0.34495920	0.33815828
12	(47, 42)	2	11010101	2.10671520	0.85979980
13	(47, 42)	2	11011001	1.90959561	0.94315439
14	(37, 40)	3	10000101	1.98351550	0.91603357
15	(37, 40)	3	01100001	1.65087616	0.99679530
16	(38, 22)	2	11001001	1.81103587	0.97128099
17	(38, 22)	2	00111110	1.52767646	0.99907047
18	(14, 48)	1	10001110	1.39215684	0.98408633
19	(14, 48)	1	00001011	2.56255412	0.54721946
20	(35, 19)	6	10010001	1.68783617	0.99315864
21	(35, 19)	6	10000001	1.58927631	0.99982923
22	(32, 16)	7	11110111	2.94447327	0.19584532
23	(32, 16)	7	00011000	0.29567933	0.29138976
24	(23, 38)	7	11001110	1.41679680	0.98816550
25	(23, 38)	7	11111110	1.56463647	0.99998105
26	(36, 35)	3	11010001	1.71247613	0.98998022
27	(36, 35)	3	10011001	1.88495564	0.95105648
28	(2, 15)	6	00111010	1.13343740	0.90587342
29	(2, 15)	6	11001110	1.41679680	0.98816550
30	(20, 16)	2	01011001	1.89727569	0.94717729
31	(20, 16)	2	00010011	2.46399426	0.62692380
32	(4, 42)	0	11010101	2.10671520	0.85979980
33	(4, 42)	0	00111111	3.10463285	0.03695139
34	(5, 13)	6	00110110	1.33055699	0.97128105
35	(5, 13)	6	00001001	1.77407598	0.97940975
36	(17, 6)	7	00001011	2.56255412	0.54721946
37	(17, 6)	7	10110000	0.16015963	0.15947580
38	(25, 19)	7	10111011	2.72271371	0.40673658
39	(25, 19)	7	10000000	0.01231997	0.01231966
40	(26, 12)	6	11000001	1.61391628	0.99907047
41	(26, 12)	6	10000101	1.98351550	0.91603357
42	(0, 9)	7	00111001	1.92191565	0.93898833
43	(0, 9)	7	01101001	1.84799576	0.96182561
44	(22, 10)	5	00001101	2.16831493	0.82673419
45	(22, 10)	5	11100001	1.66319621	0.99573416
46	(37, 32)	3	10010110	1.29359698	0.96182567
47	(37, 32)	3	11100001	1.66319621	0.99573416
48	(48, 29)	0	10010010	0.89935791	0.78292763
49	(48, 29)	0	00001110	1.37983680	0.98182255

ESTADISTIQUES NOVES:**Mitja:** 0.794184**Max:** 0.999981**Min:** 0.012320**Sumfitness:** 39.709179**-----GENERACIO 8-----**

	Pares	Locus	Cromossoma	x	f(x)
0	(25,13)	0	10000001	1.58927631	0.99982923
1	(25,13)	0	10000001	1.58927631	0.99982923
2	(0,26)	1	00000001	1.57695639	0.99998105
3	(0,26)	1	00000001	1.57695639	0.99998105
4	(14,18)	3	00000001	1.57695639	0.99998105
5	(14,18)	3	11100001	1.66319621	0.99573416
6	(45,1)	5	00000001	1.57695639	0.99998105
7	(45,1)	5	00000001	1.57695639	0.99998105
8	(7,35)	7	00000001	1.57695639	0.99998105
9	(7,35)	7	00000001	1.57695639	0.99998105
10	(11,47)	0	00000001	1.57695639	0.99998105
11	(11,47)	0	01100001	1.65087616	0.99679530
12	(17,32)	1	10000001	1.58927631	0.99982923
13	(17,32)	1	00000001	1.57695639	0.99998105
14	(44,40)	0	00000001	1.57695639	0.99998105
15	(44,40)	0	10000001	1.58927631	0.99982923
16	(2,37)	2	00000001	1.57695639	0.99998105
17	(2,37)	2	00000001	1.57695639	0.99998105
18	(19,29)	6	10000001	1.58927631	0.99982923
19	(19,29)	6	00000001	1.57695639	0.99998105
20	(16,39)	5	10000001	1.58927631	0.99982923
21	(16,39)	5	10000001	1.58927631	0.99982923
22	(33,3)	0	00000001	1.57695639	0.99998105
23	(33,3)	0	00000001	1.57695639	0.99998105
24	(10,15)	4	00000001	1.57695639	0.99998105
25	(10,15)	4	00000001	1.57695639	0.99998105
26	(9,32)	2	01000001	1.60159636	0.99952573
27	(9,32)	2	00000001	1.57695639	0.99998105
28	(1,19)	2	00000001	1.57695639	0.99998105
29	(1,19)	2	10000001	1.58927631	0.99982923
30	(30,40)	7	00100001	1.62623632	0.99846357
31	(30,40)	7	00000001	1.57695639	0.99998105
32	(47,38)	1	00000001	1.57695639	0.99998105
33	(47,38)	1	10000001	1.58927631	0.99982923
34	(35,2)	0	00000001	1.57695639	0.99998105
35	(35,2)	0	00000001	1.57695639	0.99998105
36	(7,3)	0	00000001	1.57695639	0.99998105
37	(7,3)	0	00000001	1.57695639	0.99998105
38	(10,45)	2	00000001	1.57695639	0.99998105
39	(10,45)	2	00000011	2.36543465	0.70054293
40	(8,17)	0	10000001	1.58927631	0.99982923
41	(8,17)	0	00100001	1.62623632	0.99846357
42	(33,0)	6	00000001	1.57695639	0.99998105
43	(33,0)	6	00000001	1.57695639	0.99998105
44	(34,13)	2	10000001	1.58927631	0.99982923
45	(34,13)	2	10000001	1.58927631	0.99982923
46	(39,16)	5	10000001	1.58927631	0.99982923
47	(39,16)	5	10000001	1.58927631	0.99982923
48	(29,37)	6	00000001	1.57695639	0.99998105
49	(29,37)	6	00000001	1.57695639	0.99998105

ESTADISTIQUES NOVES:**Mitja:** 0.993731**Max:** 0.999981**Min:** 0.700543**Sumfitness:** 49.686554**-----GENERACIO 9-----**

	Pares	Locus	Cromossoma	x	f(x)
0	(9,42)	7	00000001	1.57695639	0.99998105
1	(9,42)	7	00000001	1.57695639	0.99998105
2	(32,14)	0	00000001	1.57695639	0.99998105
3	(32,14)	0	00000001	1.57695639	0.99998105
4	(38,19)	4	00000001	1.57695639	0.99998105
5	(38,19)	4	00000001	1.57695639	0.99998105
6	(6,17)	6	00000001	1.57695639	0.99998105
7	(6,17)	6	00000001	1.57695639	0.99998105
8	(4,28)	3	00000001	1.57695639	0.99998105
9	(4,28)	3	00000001	1.57695639	0.99998105
10	(22,12)	1	00000001	1.57695639	0.99998105
11	(22,12)	1	10000001	1.58927631	0.99982923
12	(24,29)	7	00000001	1.57695639	0.99998105
13	(24,29)	7	10000001	1.58927631	0.99982923
14	(44,37)	7	10000001	1.58927631	0.99982923
15	(44,37)	7	00000001	1.57695639	0.99998105
16	(43,16)	7	00000001	1.57695639	0.99998105
17	(43,16)	7	00000001	1.57695639	0.99998105
18	(7,36)	0	00000001	1.57695639	0.99998105
19	(7,36)	0	00000001	1.57695639	0.99998105
20	(13,2)	4	00000001	1.57695639	0.99998105
21	(13,2)	4	00000001	1.57695639	0.99998105
22	(15,34)	1	10000001	1.58927631	0.99982923
23	(15,34)	1	00000001	1.57695639	0.99998105
24	(31,22)	6	00000001	1.57695639	0.99998105
25	(31,22)	6	00000001	1.57695639	0.99998105
26	(49,29)	5	00000001	1.57695639	0.99998105
27	(49,29)	5	10000001	1.58927631	0.99982923
28	(38,34)	0	00000001	1.57695639	0.99998105
29	(38,34)	0	00000001	1.57695639	0.99998105
30	(15,25)	6	10000001	1.58927631	0.99982923
31	(15,25)	6	00000001	1.57695639	0.99998105
32	(24,2)	7	00000001	1.57695639	0.99998105
33	(24,2)	7	00000001	1.57695639	0.99998105
34	(32,13)	5	00000001	1.57695639	0.99998105
35	(32,13)	5	00000001	1.57695639	0.99998105
36	(36,4)	0	00000001	1.57695639	0.99998105
37	(36,4)	0	00000001	1.57695639	0.99998105
38	(7,19)	5	00000001	1.57695639	0.99998105
39	(7,19)	5	00000001	1.57695639	0.99998105
40	(42,16)	3	00000001	1.57695639	0.99998105
41	(42,16)	3	00000001	1.57695639	0.99998105
42	(10,48)	2	00000001	1.57695639	0.99998105
43	(10,48)	2	00000001	1.57695639	0.99998105
44	(6,40)	2	00000001	1.57695639	0.99998105
45	(6,40)	2	10000001	1.58927631	0.99982923
46	(3,9)	6	00000001	1.57695639	0.99998105
47	(3,9)	6	00000001	1.57695639	0.99998105
48	(37,45)	7	00000001	1.57695639	0.99998105
49	(37,45)	7	10000001	1.58927631	0.99982923

ESTADISTIQUES NOVES:**Mitja:** 0.999957**Max:** 0.999981**Min:** 0.999829**Sumfitness:** 49.997829**-----GENERACIO 10-----**

Pares	Locus	Cromossoma	x	f(x)
0	(8,7)	5 00000001	1.57695639	0.99998105
1	(8,7)	5 00000001	1.57695639	0.99998105
2	(39,31)	6 00000001	1.57695639	0.99998105
3	(39,31)	6 00000001	1.57695639	0.99998105
4	(44,20)	7 00000001	1.57695639	0.99998105
5	(44,20)	7 00000001	1.57695639	0.99998105
6	(24,46)	6 00000001	1.57695639	0.99998105
7	(24,46)	6 00000001	1.57695639	0.99998105
8	(37,29)	6 00000001	1.57695639	0.99998105
9	(37,29)	6 00000001	1.57695639	0.99998105
10	(43,15)	6 00000001	1.57695639	0.99998105
11	(43,15)	6 00000001	1.57695639	0.99998105
12	(9,12)	1 00000001	1.57695639	0.99998105
13	(9,12)	1 00000001	1.57695639	0.99998105
14	(25,23)	0 00000001	1.57695639	0.99998105
15	(25,23)	0 00000001	1.57695639	0.99998105
16	(47,3)	3 00000001	1.57695639	0.99998105
17	(47,3)	3 00000001	1.57695639	0.99998105
18	(18,5)	1 00000001	1.57695639	0.99998105
19	(18,5)	1 00000001	1.57695639	0.99998105
20	(36,17)	0 00000001	1.57695639	0.99998105
21	(36,17)	0 00000001	1.57695639	0.99998105
22	(41,35)	0 00000001	1.57695639	0.99998105
23	(41,35)	0 00000001	1.57695639	0.99998105
24	(6,47)	7 00000001	1.57695639	0.99998105
25	(6,47)	7 00000001	1.57695639	0.99998105
26	(37,36)	4 00000001	1.57695639	0.99998105
27	(37,36)	4 00000001	1.57695639	0.99998105
28	(19,26)	6 00000001	1.57695639	0.99998105
29	(19,26)	6 00000001	1.57695639	0.99998105
30	(7,15)	5 00000001	1.57695639	0.99998105
31	(7,15)	5 00000001	1.57695639	0.99998105
32	(40,0)	5 00000001	1.57695639	0.99998105
33	(40,0)	5 00000001	1.57695639	0.99998105
34	(28,25)	7 00000001	1.57695639	0.99998105
35	(28,25)	7 00000001	1.57695639	0.99998105
36	(20,12)	0 00000001	1.57695639	0.99998105
37	(20,12)	0 00000001	1.57695639	0.99998105
38	(35,34)	0 00000001	1.57695639	0.99998105
39	(35,34)	0 00000001	1.57695639	0.99998105
40	(9,41)	7 00000001	1.57695639	0.99998105
41	(9,41)	7 00000001	1.57695639	0.99998105
42	(10,2)	1 00000001	1.57695639	0.99998105
43	(10,2)	1 00000001	1.57695639	0.99998105
44	(31,43)	5 00000001	1.57695639	0.99998105
45	(31,43)	5 00000001	1.57695639	0.99998105
46	(4,17)	6 00000001	1.57695639	0.99998105
47	(4,17)	6 00000001	1.57695639	0.99998105
48	(42,21)	6 00000001	1.57695639	0.99998105
49	(42,21)	6 00000001	1.57695639	0.99998105

```

ESTADISTIQUES NOVES:
Mitja: 0.999981
Max: 0.999981
Min: 0.999981
Sumfitness: 49.999050
Hem parat pel criteri de convergencia.
Nº iteracions:10

OPTIM:
x:1.57695639      f:0.99998105

```

Efectivament s'ha arribat a l'òptim que es volia (i si no és exacte és per un problema de precisió). La figura A1.4 mostra l'evolució del màxim i la mitjana a través de les generacions, que altre cop va realitzant salts tot acostant-se cap a l'òptim.

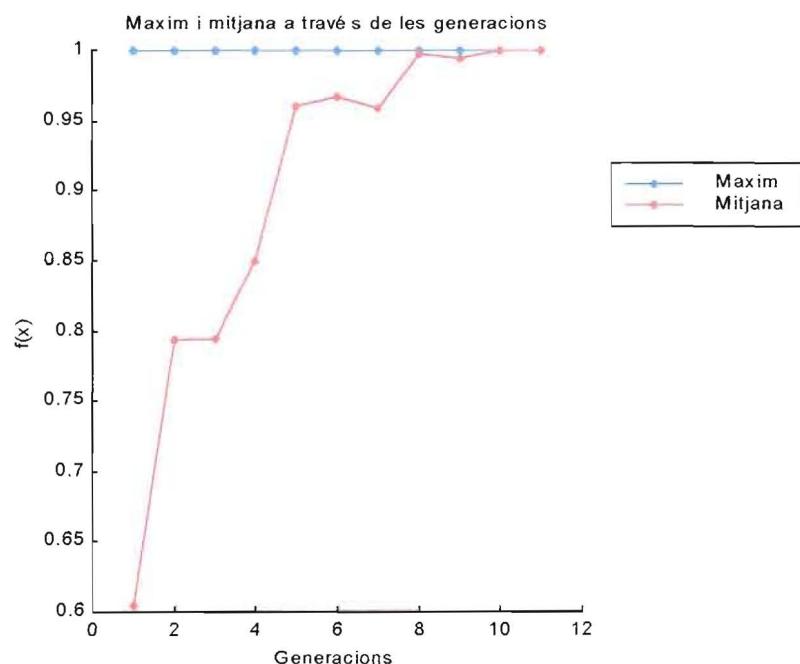


Figura A1.4

Funció definida a trossos

Per acabar es presenten els resultats per la funció definida a trossos

$$f(x) = \begin{cases} x & x \in [0,1) \\ 2 - x & x \in [1,2) \\ 2 \cdot x - 4 & x \in [2,3) \\ 8 - 2 \cdot x & x \in [3,4) \\ 0 & \text{altrament} \end{cases}$$

La representació gràfica d'aquesta funció és la de la figura A1.5.

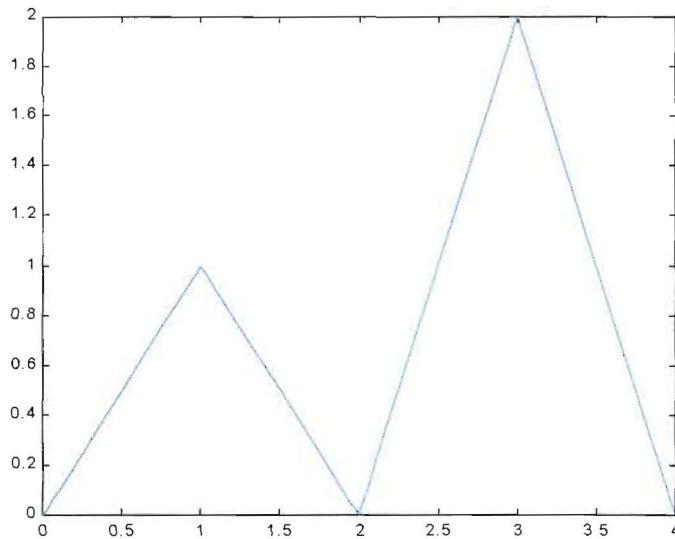


Figura A1.5

Aquesta funció té el màxim a $x=3$ amb un valor de 2.

Els paràmetres de l'algorisme en aquest cas i la pantalla d'entrada es troben representats a la taula A1.3 i a la figura A1.6.

Paràmetre	Valor
Espai de cerca: límit inferior	0
Espai de cerca: límit superior	4
Grandària de la població	60
Probabilitat de creuament	1
Probabilitat de mutació	0.0015
Criteri de parada	Rati
Rati de convergència	1
Reinserció?	Si

Taula A1.3

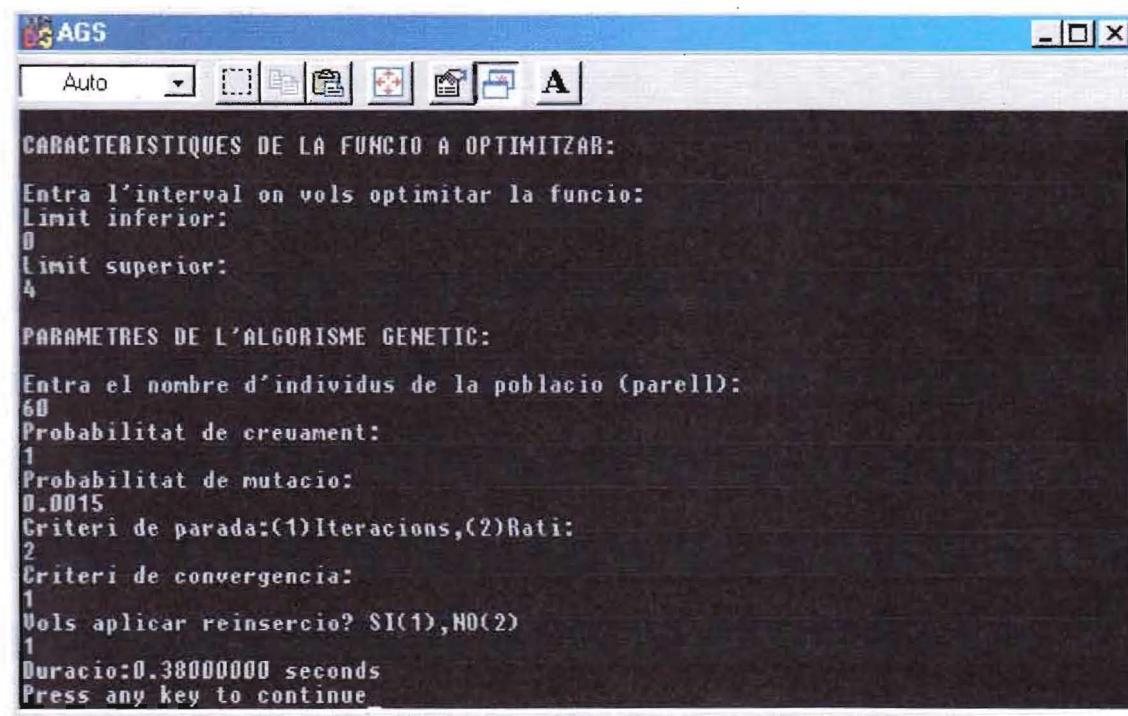


Figura A1.6

Com en el cas anterior només es presenten alguns dels resultats que ja donen idea de com ha actuat l'algorisme.

POBLACIO INICIAL

	Cromossoma	x	f(x)
0	01011101	2.91764712	1.83529425
1	11000111	3.56078458	0.87843084
2	00001000	0.25098041	0.25098041
3	01111000	0.47058827	0.47058827
4	01011011	3.41960812	1.16078377
5	10000011	3.02745128	1.94509745
6	10011001	2.40000010	0.80000019
7	00100111	3.57647085	0.84705830
8	00101111	3.82745123	0.34509754
9	01100000	0.09411766	0.09411766
10	01000010	1.03529418	0.96470582
11	00100010	1.06666672	0.93333328
12	01111100	0.97254908	0.97254908
13	10010111	3.65490222	0.69019556
14	11010010	1.17647064	0.82352936
15	11110001	2.24313736	0.48627472
16	00010111	3.63921595	0.72156811
17	11111110	1.99215698	0.00784302
18	00010100	0.62745100	0.62745100
19	11100111	3.62352967	0.75294065
20	11011001	2.43137264	0.86274529
21	10111100	0.95686281	0.95686281
22	10001101	2.77647066	1.55294132
23	01100100	0.59607846	0.59607846
24	00010001	2.13333344	0.26666689
25	01111100	0.97254908	0.97254908
26	10001100	0.76862752	0.76862752

27	01111101	2.98039222	1.96078444
28	11010001	2.18039227	0.36078453
29	00110001	2.19607854	0.39215708
30	00000100	0.50196081	0.50196081
31	11000000	0.04705883	0.04705883
32	00110000	0.18823531	0.18823531
33	01110101	2.72941184	1.45882368
34	11100100	0.61176473	0.61176473
35	10101101	2.83921576	1.67843151
36	11100010	1.11372554	0.88627446
37	10111000	0.45490199	0.45490199
38	10101000	0.32941177	0.32941177
39	10101100	0.83137262	0.83137262
40	00001000	0.25098041	0.25098041
41	10011010	1.39607847	0.60392153
42	01110001	2.22745109	0.45490217
43	11011010	1.42745101	0.57254899
44	00110010	1.19215691	0.80784309
45	01011111	3.92156887	0.15686226
46	10011100	0.89411771	0.89411771
47	11000000	0.04705883	0.04705883
48	00101001	2.32156873	0.64313745
49	01011111	3.92156887	0.15686226
50	01100001	2.10196090	0.20392179
51	11110110	1.74117661	0.25882339
52	11101011	3.37254930	1.25490141
53	10101000	0.32941177	0.32941177
54	01010011	3.16862774	1.66274452
55	00110110	1.69411778	0.30588222
56	00110110	1.69411778	0.30588222
57	01101010	1.34901965	0.65098035
58	10100001	2.08627462	0.17254925
59	00110100	0.69019610	0.69019610

ESTADISTIQUES INICIALS:

Mitja: 0.695163

Max: 1.960784

Min: 0.007843

Sumfitness: 41.709805

-----GENERACIO 1-----

	Pares	Locus	Cromossoma	x	f(x)
0	(23,10)	6	01100110	1.60000014	0.39999986
1	(23,10)	6	010000000	0.03137255	0.03137255
2	(5,58)	1	10100001	2.08627462	0.17254925
3	(5,58)	1	10000011	3.02745128	1.94509745
4	(46,36)	1	11100010	1.11372554	0.88627446
5	(46,36)	1	10011100	0.89411771	0.89411771
6	(0,6)	4	01011001	2.41568637	0.83137274
7	(0,6)	4	10011101	2.90196085	1.80392170
8	(4,22)	2	01001101	2.79215693	1.58431387
9	(4,22)	2	10011011	3.40392184	1.19215631
10	(12,26)	0	10001100	0.76862752	0.76862752
11	(12,26)	0	01111100	0.97254908	0.97254908
12	(59,1)	1	01000111	3.54509830	0.90980339
13	(59,1)	1	10110100	0.70588237	0.70588237
14	(54,7)	6	01010011	3.16862774	1.66274452
15	(54,7)	6	00100111	3.57647085	0.84705830
16	(14,49)	1	11011111	3.93725514	0.12548971
17	(14,49)	1	01010010	1.16078436	0.83921564

18	(21, 34)	2	10100100	0.58039218	0.58039218
19	(21, 34)	2	11111100	0.98823535	0.98823535
20	(37, 27)	3	10111101	2.96470594	1.92941189
21	(37, 27)	3	01111000	0.47058827	0.47058827
22	(25, 39)	3	01101100	0.84705889	0.84705889
23	(25, 39)	3	10111100	0.95686281	0.95686281
24	(35, 33)	2	10110101	2.71372557	1.42745113
25	(35, 33)	2	01101101	2.85490203	1.70980406
26	(30, 43)	4	00001010	1.25490201	0.74509799
27	(30, 43)	4	11010100	0.67450982	0.67450982
28	(57, 52)	2	01101011	3.35686302	1.28627396
29	(57, 52)	2	11101010	1.36470592	0.63529408
30	(7, 14)	6	00100110	1.56862760	0.43137240
31	(7, 14)	6	10010011	3.15294147	1.69411707
32	(57, 21)	1	00111100	0.941117653	0.941117653
33	(57, 21)	1	11101010	1.36470592	0.63529408
34	(48, 6)	4	00101001	2.32156873	0.64313745
35	(48, 6)	4	10011001	2.40000010	0.80000019
36	(4, 42)	7	01011011	3.41960812	1.16078377
37	(4, 42)	7	01110001	2.22745109	0.45490217
38	(41, 44)	7	10011010	1.39607847	0.60392153
39	(41, 44)	7	00110010	1.19215691	0.80784309
40	(43, 19)	3	11000111	3.56078458	0.87843084
41	(43, 19)	3	11111010	1.49019611	0.50980389
42	(51, 59)	3	11110100	0.73725492	0.73725492
43	(51, 59)	3	00110110	1.69411778	0.30588222
44	(5, 30)	7	10000010	1.01960790	0.98039210
45	(5, 30)	7	00000101	2.50980401	1.01960802
46	(16, 46)	5	00010100	0.62745100	0.62745100
47	(16, 46)	5	10011111	3.90588260	0.18823481
48	(22, 27)	6	10001101	2.77647066	1.55294132
49	(22, 27)	6	01111101	2.98039222	1.96078444
50	(33, 0)	1	01011101	2.91764712	1.83529425
51	(33, 0)	1	01110101	2.72941184	1.45882368
52	(25, 1)	3	01100111	3.60784340	0.78431320
53	(25, 1)	3	11011100	0.92549026	0.92549026
54	(35, 29)	2	10110001	2.21176481	0.42352962
55	(35, 29)	2	00101101	2.82352948	1.64705896
56	(11, 9)	7	00100110	1.56862760	0.43137240
57	(11, 9)	7	01100000	0.09411766	0.09411766
58	(36, 54)	5	11100011	3.12156892	1.75686216
59	(36, 54)	5	01010010	1.16078436	0.83921564

ESTADISTIQUES NOVES:

Mitja: 0.932549

Max: 1.960784

Min: 0.031373

Sumfitness: 55.952938

-----GENERACIO 13-----					
	Pares	Locus	Cromossoma	x	f(x)
0	(43,55)	6	11111101	2.99607849	1.99215698
1	(43,55)	6	11011101	2.93333340	1.86666679
2	(1,26)	3	11111101	2.99607849	1.99215698
3	(1,26)	3	11111101	2.99607849	1.99215698
4	(53,23)	2	11111101	2.99607849	1.99215698
5	(53,23)	2	11111101	2.99607849	1.99215698
6	(58,39)	2	11111101	2.99607849	1.99215698
7	(58,39)	2	11111101	2.99607849	1.99215698
8	(28,24)	3	11111101	2.99607849	1.99215698
9	(28,24)	3	11111101	2.99607849	1.99215698
10	(35,3)	1	11111101	2.99607849	1.99215698
11	(35,3)	1	11111101	2.99607849	1.99215698
12	(31,46)	4	11111101	2.99607849	1.99215698
13	(31,46)	4	11111101	2.99607849	1.99215698
14	(49,14)	1	11111101	2.99607849	1.99215698
15	(49,14)	1	11111101	2.99607849	1.99215698
16	(27,32)	4	11101101	2.87058830	1.74117661
17	(27,32)	4	11111101	2.99607849	1.99215698
18	(48,41)	6	11111101	2.99607849	1.99215698
19	(48,41)	6	11111101	2.99607849	1.99215698
20	(13,29)	2	11111101	2.99607849	1.99215698
21	(13,29)	2	11111101	2.99607849	1.99215698
22	(51,15)	7	11111101	2.99607849	1.99215698
23	(51,15)	7	11111101	2.99607849	1.99215698
24	(20,19)	1	11111101	2.99607849	1.99215698
25	(20,19)	1	11111101	2.99607849	1.99215698
26	(30,16)	0	11111101	2.99607849	1.99215698
27	(30,16)	0	11111101	2.99607849	1.99215698
28	(21,38)	1	11111101	2.99607849	1.99215698
29	(21,38)	1	11111101	2.99607849	1.99215698
30	(37,18)	7	11111101	2.99607849	1.99215698
31	(37,18)	7	11111101	2.99607849	1.99215698
32	(0,9)	0	11111101	2.99607849	1.99215698
33	(0,9)	0	11111101	2.99607849	1.99215698
34	(54,49)	1	11111101	2.99607849	1.99215698
35	(54,49)	1	11111101	2.99607849	1.99215698
36	(44,47)	5	11111101	2.99607849	1.99215698
37	(44,47)	5	11111101	2.99607849	1.99215698
38	(24,27)	5	11111101	2.99607849	1.99215698
39	(24,27)	5	11111101	2.99607849	1.99215698
40	(58,50)	1	11111101	2.99607849	1.99215698
41	(58,50)	1	11111101	2.99607849	1.99215698
42	(19,29)	2	11111101	2.99607849	1.99215698
43	(19,29)	2	11111101	2.99607849	1.99215698
44	(22,42)	6	11111101	2.99607849	1.99215698
45	(22,42)	6	11111101	2.99607849	1.99215698
46	(33,16)	3	11111101	2.99607849	1.99215698
47	(33,16)	3	11111101	2.99607849	1.99215698
48	(20,55)	5	11111101	2.99607849	1.99215698
49	(20,55)	5	11111101	2.99607849	1.99215698
50	(39,23)	2	11111101	2.99607849	1.99215698
51	(39,23)	2	11111101	2.99607849	1.99215698
52	(8,13)	1	11111101	2.99607849	1.99215698
53	(8,13)	1	11111101	2.99607849	1.99215698
54	(52,17)	2	11111101	2.99607849	1.99215698
55	(52,17)	2	11111101	2.99607849	1.99215698

56	(46, 35)	0	11111101	2.99607849	1.99215698
57	(46, 35)	0	11111101	2.99607849	1.99215698
58	(53, 14)	3	11111101	2.99607849	1.99215698
59	(53, 14)	3	11111101	2.99607849	1.99215698

ESTADISTIQUES NOVES:

Mitja: 1.985882

Max: 1.992157

Min: 1.741177

Sumfitness: 119.152946

-----GENERACIO 14-----

	Pares	Locus	Cromossoma	x	f(x)
0	(43, 48)	7	11111101	2.99607849	1.99215698
1	(43, 48)	7	11111101	2.99607849	1.99215698
2	(27, 12)	6	11111101	2.99607849	1.99215698
3	(27, 12)	6	11111101	2.99607849	1.99215698
4	(42, 3)	0	11111101	2.99607849	1.99215698
5	(42, 3)	0	11111101	2.99607849	1.99215698
6	(52, 20)	6	11111101	2.99607849	1.99215698
7	(52, 20)	6	11111101	2.99607849	1.99215698
8	(24, 17)	4	11111101	2.99607849	1.99215698
9	(24, 17)	4	11111101	2.99607849	1.99215698
10	(57, 59)	4	11111101	2.99607849	1.99215698
11	(57, 59)	4	11111101	2.99607849	1.99215698
12	(31, 41)	5	11111101	2.99607849	1.99215698
13	(31, 41)	5	11111101	2.99607849	1.99215698
14	(37, 40)	1	11111101	2.99607849	1.99215698
15	(37, 40)	1	11111101	2.99607849	1.99215698
16	(23, 34)	2	11111101	2.99607849	1.99215698
17	(23, 34)	2	11111101	2.99607849	1.99215698
18	(45, 44)	2	11111101	2.99607849	1.99215698
19	(45, 44)	2	11111101	2.99607849	1.99215698
20	(29, 46)	2	11111101	2.99607849	1.99215698
21	(29, 46)	2	11111101	2.99607849	1.99215698
22	(5, 49)	7	11111101	2.99607849	1.99215698
23	(5, 49)	7	11111101	2.99607849	1.99215698
24	(7, 39)	5	11111101	2.99607849	1.99215698
25	(7, 39)	5	11111101	2.99607849	1.99215698
26	(28, 4)	0	11111101	2.99607849	1.99215698
27	(28, 4)	0	11111101	2.99607849	1.99215698
28	(8, 9)	1	11111101	2.99607849	1.99215698
29	(8, 9)	1	11111101	2.99607849	1.99215698
30	(14, 17)	0	11111101	2.99607849	1.99215698
31	(14, 17)	0	11111101	2.99607849	1.99215698
32	(31, 8)	3	11111101	2.99607849	1.99215698
33	(31, 8)	3	11111101	2.99607849	1.99215698
34	(56, 52)	5	11111101	2.99607849	1.99215698
35	(56, 52)	5	11111101	2.99607849	1.99215698
36	(21, 30)	1	11111101	2.99607849	1.99215698
37	(21, 30)	1	11111101	2.99607849	1.99215698
38	(23, 44)	5	11111101	2.99607849	1.99215698
39	(23, 44)	5	11111101	2.99607849	1.99215698
40	(49, 34)	4	11111101	2.99607849	1.99215698
41	(49, 34)	4	11111101	2.99607849	1.99215698
42	(42, 57)	3	11111101	2.99607849	1.99215698
43	(42, 57)	3	11111101	2.99607849	1.99215698
44	(48, 39)	7	11111101	2.99607849	1.99215698
45	(48, 39)	7	11111101	2.99607849	1.99215698
46	(6, 54)	6	11111101	2.99607849	1.99215698

47	(6, 54)	6	11111101	2.99607849	1.99215698
48	(32, 27)	7	11111101	2.99607849	1.99215698
49	(32, 27)	7	11111101	2.99607849	1.99215698
50	(15, 29)	5	11111101	2.99607849	1.99215698
51	(15, 29)	5	11111101	2.99607849	1.99215698
52	(19, 59)	2	11111101	2.99607849	1.99215698
53	(19, 59)	2	11111101	2.99607849	1.99215698
54	(53, 47)	3	11111101	2.99607849	1.99215698
55	(53, 47)	3	11111101	2.99607849	1.99215698
56	(36, 13)	4	11111101	2.99607849	1.99215698
57	(36, 13)	4	11111101	2.99607849	1.99215698
58	(24, 28)	2	11111101	2.99607849	1.99215698
59	(24, 28)	2	11111101	2.99607849	1.99215698

ESTADISTIQUES NOVES:

Mitja: 1.992157

Max: 1.992157

Min: 1.992157

Sumfitness: 119.529419

Hem parat pel criteri de convergencia.

Nº iteracions: 14

OPTIM:

x: 2.99607849

f: 1.99215698

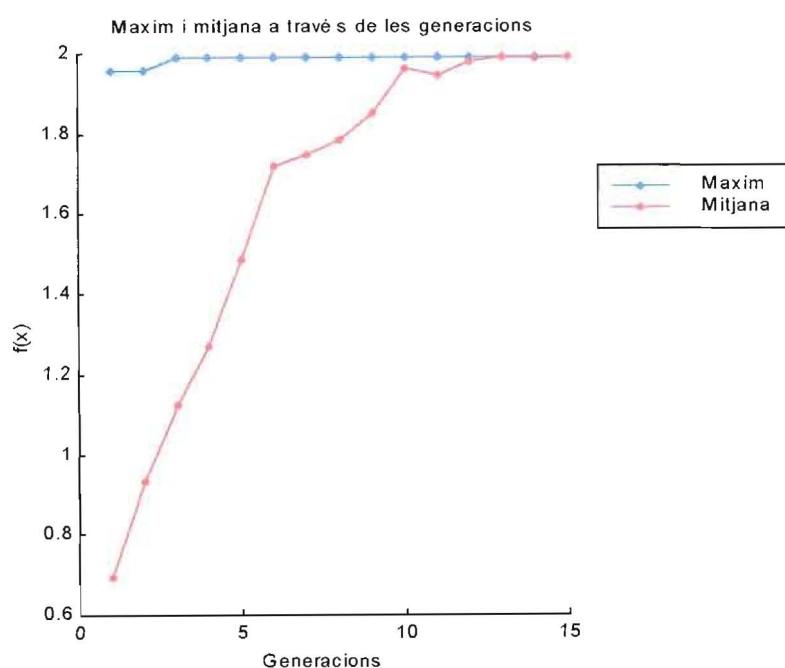


Figura A1.7

L'algorisme ha detectat el màxim a $x=2.99$ com era d'esperar.

ANNEX 2. IMPLEMENTACIÓ DEL TSDM

Tot i trobar-se en un Annex, aquesta part del projecte és de fet la més important perquè és presenta la implementació en llenguatge C, en el programa `TSDM.c`, del mètode del TSDM que ha donat lloc a tots els resultats obtinguts al llarg del treball.

El capítol es divideix en tres parts: una primera part sobre comentaris a la implementació, bàsicament sobre funcions pròpies del TSDM. Sobre l'algorisme genètic que porta incorporat el mètode així com de les estructures de dades que s'usen no s'explica res perquè ja se li ha dedicat l'Annex 1. Precisament s'ha volgut separar l'algorisme genètic de la implementació pròpia del TSDM per fixar millor les idees de cada part. Tot seguit es presenta el codi íntegre del programa `TSDM.c` al qual han fert referència els comentaris així com de les funcions més necessàries de Matlab. Per acabar, un exemple d'execució del mètode, utilitzant tant el programa `TSDM.c` com les funcions de Matlab que permeten realitzar la fase test del mètode i tots els resultats gràfics.

A2.1 Funcions pròpies del TSDM

I. Fase prèvia

La fase prèvia del mètode que consisteix a separar la sèrie en sèrie d'aprenentatge i sèrie test cal realitzar-la abans d'aplicar el programa, i executar aquest només amb la sèrie d'aprenentatge. La sèrie test es guardarà per avaluar els resultats amb Matlab.

II. Fase d'Aprenentatge

El programa `TSDM.c` consisteix de fet en la fase d'aprenentatge del mètode ja que té com a objectiu la cerca del patró temporal òptim.

El primer pas a l'hora d'aplicar el mètode era l'establiment de l'objectiu de TSDM, així com la tria de paràmetres del mètode. Aquests pas és el que es realitza amb la funció `EntradaDades()` que es pot veure al final d'aquest capítol. S'observarà que la versió del programa que s'ha presentat inclou la tria de paràmetres el més exhaustiva possible, en el sentit que es pot triar entre totes les funcions d'events que s'han utilitzat al llarg del treball, inclosa

$g(t) = x_{t+s}$ i que permet també triar entre realitzar maximització o minimització. Cal esmentar dos paràmetres que no han aparegut fins ara i que són propis del TSDM.

- *Tolerància usada en la selecció:* en la selecció de l'algorisme genètic s'usa la funció $b(\delta)$ i ja s'ha explicat que serveix per desempatar en cas que dos punts tinguin igual valor de la funció objectiu principal. Cal una tolerància que digui quan es pot considerar que aquests dos valors són iguals.
- *Coeficient de reducció del radi:* en l'algorisme genètic anterior hi havia com a paràmetres d'entrada l'interval de cerca. En aquest cas els intervals de cerca són calculats pel propi mètode com es veurà més endavant. El radi es busca de manera que sigui >0 (sino no tindria sentit) i $<$ que la màxima distància entre dos punts de la sèrie dividida per un cert coeficient, que és el que es demana al entrar els paràmetres.

El proper pas de la fase d'aprenentatge és realitzar la representació en l'espai dels retards i en l'espai dels retards augmentat. Això és el que fa la funció `TractamentInicialSerie()`. Aquesta funció calcula els retards usant una funció anomenada `Retards`, que a més calcula el valor associat de la funció d'events, tenint d'aquesta manera els punts en l'espai augmentat. En el tractament inicial de la sèrie s'efectua a més l'establiment de l'espai de cerca: les variables que representen retards es busquen entre el valor mínim de la sèrie i el màxim, mentre que el radi es busca entre 0 i una certa fracció de la màxima distància entre dos punts de la sèrie.

Un cop realitzat aquest tractament inicial ja s'està en condicions de passar a l'algorisme genètic. El primer pas és crear una població inicial de la mateixa manera que es va veure a l'Annex 1. Es recorda que en l'Annex 1 es va veure que la població es generava de manera aleatòria com a cromossomes i que calia trobar el corresponent valor real, cosa que s'efectuava amb la funció `bit2float`. En el cas de l'Annex 1 com que la funció objectiu només tenia una variable directament es transformava el cromosoma nombre real. En el cas del TSDM el nombre de variables és 3 per $Q=2$ (x_{t-1}, x_t, δ) o 4 per $Q=3$ ($x_{t-2}, x_{t-1}, x_t, \delta$). Tal com s'ha explicat al capítol 5, el cromosoma contindrà les codificacions de totes les variables juxtaposades. Cal una funció que extreguir el gen corresponent a cada variable i després es decodificarà el gen. Aquesta funció és la funció `ExtreureVar`.

Un cop inicialitzada la població ja es pot passar aplicar l'algorisme genètic que a grans trets és actua igual que en l'Annex 1. Només cal esmentar dues variacions:

- En la Selecció per Torneig s'ha inclos el “tiebreak” mitjançant la funció $b(\delta)$.

- Cal fer una transformació a la funció objectiu per tal que sigui positiua doncs és necessari per l'algorisme. Això no afecta als valors trobats perquè només s'usa la funció transformada dins l'algorisme genètic.

També cal esmentar que la funció objectiu resulta evidentement més complicada en aquets cas que en cap dels casos de l'Annex 2 i que a més permet triar entre procés de maximització o minimització, i dins de la maximització entre la funció de comparació de mitjanes o la que té en compte també el nombre d'events.

Els arxius de sortida del programa són:

- res.txt: conté l'evolució dels cromosomes i els corresponents valors reals a través de les diferents generacions tal com s'ha vist en l'Annex 1.
- resgraf.txt: com abans conté la mitjana de la funció objectiu i el màxim a través de les generacions.
- resoptim.txt: conté el punt òptim per poder-lo llegir des de Matlab i realitzar la fase test.

Per acabar, al final hi ha la funció TractamentOptim que calcula les estadístiques de l'òptim (mitjana dins el cluster, variància, etc...).

A2.2 Codi

Tal com ja s'ha dit, tot seguit es presenta el codi del programa TSDM.c que calcula el patró temporal òptim i també les funcions necessàries de Matlab que realitzen la fase test i l'anàlisi gràfic dels resultats. En quan al programa TSDM es presneta el codi per la versió que calcula patrons temporals amb $Q=2$. Per fer-ho amb $Q=3$ només calen unes petites modificacions i per això no s'ha creut necessari presentar el codi que hagués estat en la seva major part, repetitiu.

Programa TSDM.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

#define enter32 unsigned long int
#define maxpop 200
#define lgen 8
#define Q 2
#define dim (Q+1)
#define lcrom (dim*lgen)
```

//ESTRUCTURES DE DADES

```

typedef struct {
    int C[lcrom];
} cromosoma;

typedef struct {
    cromosoma crom;
    float x[dim];
    float fitness;
    float fobjectiu; //on hi haurà la fobj reescalada pq sigui >0
    int parent1,parent2,locus,triat;
} individu;

typedef struct {
    individu P[maxpop];
} poblacio;

```

//DEFINICIO DE LES FUNCIONS

```

//Funcions que tenen a veure amb el tractament inicial de la sèrie,
//propri del TSDM
void EntradaDades();
void TractamentInicialSerie(float *X0,float *X1,float *g,float *LB,float *UB);
void Stat_Eventness(float *g,int *set,int *card,float *aver,float *vari);
void Retards(float *S,float *past,float *actual,float *future);
float distancia(float *x,float *y,int n);

```

//Funcions auxiliars del AG

```

poblacio IniPop(float *X0,float *X1,float *g,float *LB,float *UB);
void Estadistic(poblacio pop,int *indmax,int *indmin,float *avg,float
*min,float *max,float *sumfitness);
void EstadisticObj(poblacio pop,int *indmax,int *indmin,float *avg,float
*min,float *max,float *sumfitness);

```

//Operadors genètics

```

poblacio Seleccio_Torneig(int tournsize,poblacio pop,float *X0,float *X1);
poblacio Generacio(poblacio antpop,float sumfitness,float *X0,float *X1,float
*g,float *LB, float *UB);
cromosoma Mutacio(cromosoma crom,float pmutacio,int longcrom);
cromosoma *Creuament(cromosoma crom1,cromosoma crom2,float pcros,int
*locus,int longcrom);
void Reinsercio(poblacio antpop,poblacio noupop,int indmaxact,int indmin);
int CriteriParada(float stop,int numgen,int maxgen,float maxact,float minact);

```

//Funcions random auxiliars

```

float ranuni(float LB,float UB);
int intranuni(int fitsup);
int flip(float p);

```

//Funcions de decodificació i reescalat

```

float bit2float(int *subcadena,int longcrom);
float Reescalat(float x, float antinf, float antsup, float nouinf, float
nousup);
int *ExtreureVar(cromosoma crom,int posicio);

```

//Tot el que té a veure amb la funció objectiu

```

float fobj(float x1,float x2,float delta,float *Z0,float *Z1,float *g);
float b(float x1,float x2,float delta,float *X0,float *X1);

```

//Funcions d'escriptura i tractament de l'òptim

```

void SortidaInicial(poblacio antpop,float avg,float min,float max,float
sumfitness,float *LB,float *UB);

```

```

void Sortida(poblacio antpop,int numgen,float avg,float min,float max,float
sumfitness);
void WriteCrom(cromosoma crom,int longcrom);
void TractamentOptim(float x1,float x2,float delta,float *X0,float *x1,
float*g);

//VARIABLES GLOBALS
enter32 llavor_enter;
float float_232;
int realsize,popsize,maxgen;
int reinser,triadist,triag,triaf,triastop,s,p,partradi,triaopti;
float pmutacio,pcros,stop,minstring,maxstring,tol,beta;
FILE *dadent,*dadw,*dadgraf,*dadoptim;
char noms[20];

//PROGRAMA PRINCIPAL
void main()
{
    float *X0,*x1,*g,*LB,*UB;
    float avg,min,max,sumfitness,maxact,avgact,minact;
    float avgObj,minObj,maxObj,sumfitnessObj,maxactObj,avgactObj,minactObj;
    poblacio antpop,noupop;
    int i,numgen,indmax,indmin,indmaxact,indminact;
    int indmaxObj,indminObj,indmaxactObj,indminactObj;

    //Per calcular el temps de CPU
    clock_t start, finish;
    double duration;

    //Variables que necessitem al llarg de tot el procés
    llavor_enter=time(NULL);
    float_232=(float)(4294967295UL) + 1; // 2^32 com a float
    minstring=0.0;
    maxstring=(float)(pow(2,lgen)-1);

    //Fites de l'espai de cerca
    LB=(float*)malloc(dim*sizeof(float));
    UB=(float*)malloc(dim*sizeof(float));

    //INICIALIZACIO

    //Entrada de dades
    EntradaDades();

    //Alocarem espai
    X0=(float*)malloc(realsize*sizeof(float));
    X1=(float*)malloc(realsize*sizeof(float));
    g=(float*)malloc(realsize*sizeof(float));

    //Lectura i tractament inicial de la serie
    TractamentInicialSerie(X0,X1,g,LB,UB);

    //Per calcular el temps de CPU
    start=clock();

    //Creacio de la població inicial
    //d'individus, per tant en cromosomes i en floats
    antpop=IniPop(X0,X1,g,LB,UB);

    //Calcul de les estadístiques iniciales
    Estadistic(antpop,&indmax,&indmin,&avg,&min,&max,&sumfitness);
}

```

```

//Escriptura en fitxer de la poblacio inicial
//i seguiment del mínim i la mitjana de la funció objectiu
SortidaInicial(antpop,avg,min,max,sumfitness,LB,UB);
fprintf(dadgraf,"%f %f\n",max,avg);

maxact=max; avgact=avg; minact=min;
indmaxact=indmax; indminact=indmin;//Per controlar els canvis en la fobj.
numgen=1;

//Correccio perquè la funció objectiu sigui positiva
for(i=0;i<popsiz; i++)
{
    if(antpop.P[i].fitness+fabs(min)>0)
        antpop.P[i].fobjectiu=(float)(antpop.P[i].fitness+fabs(min));
    else antpop.P[i].fobjectiu=0;
}

//Calcul de les estadístiques amb la funció positiva
EstadisticObj
(antpop,&indmaxObj,&indminObj,&avgObj,&minObj,&maxObj,&sumfitnessObj);

maxactObj=maxObj; avgactObj=avgObj; minactObj=minObj;
indmaxactObj=indmaxObj; indminactObj=indminObj;

//TRACTAMENT PAS A PAS DE LA POBLACIO

while(CriteriParada(stop,numgen,maxgen,maxactObj,minactObj)==0)
{

    //Seleccionem la poblacio que passara pels op. genetics
    antpop=Seleccio_Torneig(2,antpop,X0,X1);
    //Creuament,mutació
    noupop=Generacio(antpop,sumfitnessObj,X0,X1,g,LB,UB);
    //Estadístiques
    EstadisticObj
    (noupop,&indmaxObj,&indminObj,&avgObj,&minObj,&maxObj,&sumfitnessObj);

    //Fem reinsercio si es que volem i recalculem estadístiques
    if(reinser==1){
        Reinsercio(antpop,noupop,indmaxactObj,indminObj);
        EstadisticObj
    (noupop,&indmaxObj,&indminObj,&avgObj,&minObj,&maxObj,&sumfitnessObj);
    }

    //Calculem el necessari per la sortida i la printem
    //així com el seguiment del màxim i la mitjana de la funció objectiu
    Estadistic(noupop,&indmax,&indmin,&avg,&min,&max,&sumfitness);
    Sortida(noupop,numgen,avg,min,max,sumfitness);
    fprintf(dadgraf,"%f %f\n",max,avg);

    maxact=max; avgact=avg; minact=min;
    indmaxact=indmax; indminact=indmin;

    //Correccio perquè la funció objectiu sigui positiva
    for(i=0;i<popsiz; i++)
    {
        if(noupop.P[i].fitness+fabs(min)>0)
            noupop.P[i].fobjectiu=(float)(noupop.P[i].fitness+fabs(min));
        else noupop.P[i].fobjectiu=0;
    }
}

```

```

    }
EstadisticObj
(noupop,&indmaxObj,&indminObj,&avgObj,&minObj,&maxObj,&sumfitnessObj);

maxactObj=maxObj; avgactObj=avgObj; minactObj=minObj;
indmaxactObj=indmaxObj; indminactObj=indminObj;

//Actualitzen la població
antpop=noupop;
numgen++;

}//Fi del tractament de la població

//Printem l'òptim a l'arxiu de resultats
fprintf(dadw, "\nOPTIM:\n");
fprintf(dadw, "x1:%.8f\tx2:%.8f\ptradi:%.8f\tf:%.8f\n",
    noupop.P[indmax].x[0], noupop.P[indmax].x[1], noupop.P[indmax].x[2],
    noupop.P[indmax].fitness);
//Printem l'òptim i el nombre d'iteracions a l'arxiu que anirà al Matlab
fprintf(dadoptim, "%.8f\t%.8f\t%.8f\n",
    noupop.P[indmax].x[0], noupop.P[indmax].x[1], noupop.P[indmax].x[2]);
fprintf(dadoptim, "%d\t%d\t%d\n", numgen-1, 0, 0);
//Calcul dels resultats relacionats amb l'òptim
TractamentOptim
(noupop.P[indmax].x[0], noupop.P[indmax].x[1], noupop.P[indmax].x[2], X0, X1, g);

//Per calcular el temps de CPU
finish=clock();
duration=(double)(finish - start) / CLOCKS_PER_SEC;
printf("Duracio:%.2.8f seconds\n", duration);

//Alliberar i tancar
free(LB);
free(UB);
free(X0);
free(X1);
free(g);
fclose(dadw);
fclose(dadgraf);
fclose(dadoptim);
fclose(dadent);
}

//FUNCIONS DE TRACTAMENT INICIAL DE LA SERIE, PRÒPIES DEL TSDM
//Funció de tria de paràmetres del mètode i del AG
void EntradaDades()
{
    dadw=fopen("res.txt","w");
    if(dadw==NULL) { printf("\nNo puc obrir l'arxiu d'escriptura\n"); exit(1);}

    dadgraf=fopen("resgraf.txt","w");
    if(dadgraf==NULL)
    { printf("\nNo puc obrir l'arxiu d'escriptura\n"); exit(1);}

    dadoptim=fopen("resoptim.txt","w");
    if(dadoptim==NULL)
    { printf("\nNo puc obrir l'arxiu d'escriptura\n"); exit(1);}

    printf("\nCARACTERISTIQUES DE LA SERIE A TRACTAR\n\n");

    printf("Entra el nombre d'observacions de la serie:\n");
    scanf("%d",&realsize);

    printf("Arxiu amb la serie inicial:\n");
    scanf("%s",noms);
}

```

```
dadent=fopen(noms,"r");
if(dadent==NULL) { printf("\nNo puc obrir l'arxiu d'entrada\n"); exit(1);}

printf("Distancia a usar:(1)Euclidea, (2)l1, (3)lp\n");
scanf("%d",&triadist);
if((triadist!=1) & (triadist!=2) & (triadist!=3))
{ printf("\nValor no permes.Torna a executar el programa\n"); exit(1);}
if(triadist==3)
{
    printf("Entra el valor de p:\n");
    scanf("%d",&p);
}

printf("Vols maximitzar (1) o minimitzar (2) la funcio?\n");
scanf("%d",&triaopti);
if((triaopti!=1) & (triaopti!=2))
{ printf("\nValor no permes.Torna a executar el programa\n"); exit(1);}
if(triaopti==1)
{
    printf("Funcio objectiu:(1) Dif. de mitjanes,
          (2) A mes segons nombre d'events\n");
    scanf("%d",&triaf);
    if((triaf!=1) & (triaf!=2))
    { printf("\nValor no permes.Torna a executar el programa\n"); exit(1);}
    if(triaf==2)
    {
        printf("Valor de Beta:\n");
        scanf("%f",&beta);
        if((beta>1) || (beta<0))
            { printf("\nBeta ha de pertanyer a [0,1].
                    Torna a executar el programa\n"); exit(1);}
    }
}

printf("Funcio eventness a usar:(1) g(t)=xt+1, (2) g(t)=xt+2,
       (3) g(t)=xt+s:\n");
scanf("%d",&triag);
if((triag!=1) & (triag!=2) & (triag!=3))
{ printf("\nValor no permes.Torna a executar el programa\n"); exit(1);}
if(triag==3){

    printf("Entra el valor de s:\n");
    scanf("%d",&s);
}

printf("Tolerancia usada en la seleccio:\n");
scanf("%f",&tol);

printf("Coeficient de reduccio del radi:\n");
scanf("%d",&partradi);

printf("\nPARAMETRES DE L'ALGORISME GENETIC\n\n");

printf("Tamany de la poblacio que passara per l'algoritme (parell):\n");
scanf("%d",&popsize);

printf("Probabilitat de creuament:\n");
scanf("%f",&pcros);
if((pcros>1) || (pcros<0))
{ printf("\npcros ha de pertanyer a [0,1].
        Torna a executar el programa\n"); exit(1);}

printf("Probabilitat de mutacio:\n");
scanf("%f",&pmutacio);
if((pmutacio>1) || (pmutacio<0))
{ printf("\npmutacio ha de pertanyer a [0,1].
        Torna a executar el programa\n"); exit(1);}
```

```

printf("Criteri de parada:(1)Iteracions,(2)Rati:\n");
scanf("%d",&triastop);
if((triastop!=1) && (triastop!=2))
{ printf("\nValor no permes.Torna a executar el programa\n"); exit(1);}
if(triastop==1)
{
    printf("Maxim nombre de generacions:\n");
    scanf("%d",&maxgen);
}
else {
    printf("Criteri de convergencia:\n");
    scanf("%f",&stop);
    if(stop>1)
    { printf("\nEl rati de convergencia ha de pertanyer a [0,1].
        Torna a executar el programa\n"); exit(1);}
}

printf("Vols aplicar reinsercio? SI(1),NO(2)\n");
scanf("%d",&reinser);
if((reinser!=1) && (reinser!=2))
{ printf("\nValor no permes.Torna a executar el programa\n"); exit(1);}

}

//Es calculen els punts en l'espai dels retards
//i els límits de l'espai de cerca.
void TractamentInicialSerie(float *X0,float *X1,float *g,float *LB,float *UB)
{

int i,j;
float *S,*xi,*xj,aux,maxdistcol,maxdistfila,distcol,distfila;
float max0,min0,max1,min1;

//Lectura serie original
S=(float*)malloc(realsize*sizeof(float));
for(i=0;i<realsize;i++) { fscanf(dadent,"%f\n",&aux); S[i]=aux; }

xi=(float*)malloc(Q*sizeof(float));
xj=(float*)malloc(Q*sizeof(float));

//Calcul punts en l'espai de cerca
Retards(S,X0,X1,g);

//Calcul de les fites LB,UB per cada dimensió
max0=X0[0]; max1=X1[0];
min0=X0[0]; min1=X1[0];
maxdistcol=0;
for(i=0;i<realsize;i++) {
    if(X0[i]>max0) max0=X0[i];
    if(X1[i]>max1) max1=X1[i];
    if(X0[i]<min0) min0=X0[i];
    if(X1[i]<min1) min1=X1[i];

    maxdistfila=0;
    for (j=i+1;j<realsize;j++)
    {
        xi[0]=X0[i]; xi[1]=X1[i];
        xj[0]=X0[j]; xj[1]=X1[j];

        distfila=distanzia(xi,xj,Q);
        if(distfila>maxdistfila) { maxdistfila=distfila; }
    }
    distcol=maxdistfila;
    if(distcol>maxdistcol) maxdistcol=distcol;
}
}

```

```

LB[0]=min0; UB[0]=max0;
LB[1]=min1; UB[1]=max1;
LB[2]=0.0; UB[2]=(maxdistcol/2)/partradi;

free(xi);
free(xj);
}

//Calcul de la mitjana, variància i cardinal dins el cluster o fora,
//segons es cridi la funció amb set=P o set=NP
void Stat_Eventness(float *g,int *set,int *card,float *aver,float *vari)
{
    int i;
    *aver=0;
    *vari=0;
    *card=0;

    for(i=0;i<realsize;i++)
    {
        if(set[i]==1) { *aver=*aver+g[i]; *card=*card+1; }

        if(*card==0) { *aver=0.0; *vari=1.0; }
        else {
            *aver=*aver/(*card);
            for(i=0;i<realsize;i++) {
                if(set[i]==1) *vari=*vari+(g[i]-(*aver))*(g[i]-(*aver));
            }
            *vari=*vari/(*card);
        }
    }
}

//Calcul dels punts a l'espai dels retards i a l'esperi augmentat
//així com de la funció d'events
void Retards(float *S,float *past,float *actual,float *future)
{
    int i;
    if(triag==1)
    {
        //g(t)=xt+1 s=1
        for(i=0;i<realsize-2;i++) past[i]=S[i];
        for(i=1;i<realsize-1;i++) actual[i-1]=S[i];
        for(i=2;i<realsize;i++) future[i-2]=S[i];
        realsize=realsize-2;
    }
    else if(triag==2)
    {
        //g(t)=xt+2 s=2
        for(i=0;i<realsize-3;i++) past[i]=S[i];
        for(i=1;i<realsize-2;i++) actual[i-1]=S[i];
        for(i=3;i<realsize;i++) future[i-3]=S[i];
        realsize=realsize-3;
    }
    else //en general
    {
        //g(t)=xt+s
        for(i=0;i<realsize-s;i++) past[i]=S[i];
        for(i=1;i<realsize-(s-1);i++) actual[i-1]=S[i];
        for(i=s+1;i<realsize;i++) future[i-(s+1)]=S[i];
        realsize=realsize-(s+1);
    }
}

//Distància entre 2 punts (es pot triar el tipus de distància)
float distancia(float *x,float *y,int n)
{
    int i,pdist;

```

```

float dist;
double expo;

if(triadist==1) pdist=2;
else if (triadist==2) pdist=1;
else pdist=p;

dist=0.0;
for(i=0;i<n;i++) { dist=(float)(dist+pow(fabs(x[i]-y[i]),pdist));
}

expo=(1.0/pdist);
dist=(float)pow(dist,expo);
return dist;
}

//FUNCIONS AUXILIARS DEL AG
//Inicialització de la població que passa per l'AG
poblacio IniPop(float *X0,float *X1,float *g,float *LB,float *UB)
{
poblacio antpop;
int i,j,k;
float aux;
int *subcadena;
for(i=0;i<popszie;i++)
{
//Generacio aleatòria de cromossomes
for(j=0;j<lcrom;j++) antpop.P[i].crom.C[j]=flip(0.5);

//Transformació a valors reals
for(k=0;k<dim;k++)
{
subcadena=ExtreureVar(antpop.P[i].crom,k);
aux=bit2float(subcadena,lgen);
antpop.P[i].x[k]=Reescalat(aux,minstring,maxstring,LB[k],UB[k]);
}
antpop.P[i].fitness=
fobj(antpop.P[i].x[0],antpop.P[i].x[1],antpop.P[i].x[2],X0,X1,g);
antpop.P[i].fobjectiu=antpop.P[i].fitness;
antpop.P[i].parent1=0;
antpop.P[i].parent2=0;
antpop.P[i].locus=0;
}
return antpop;
}

//Estadístiques de la funció objectiu sobre una població
void Estadistic(poblacio pop,int *indmax,int *indmin,float *avg,float
*min,float *max,float *sumfitness)
{
int i;
*indmax=0;
*indmin=0;
*sumfitness=pop.P[0].fitness;
*min=pop.P[0].fitness;
*max=pop.P[0].fitness;
for(i=1;i<popszie;i++)
{
*sumfitness=*sumfitness+pop.P[i].fitness;
if(pop.P[i].fitness>*max) { *max=pop.P[i].fitness; *indmax=i;}
if(pop.P[i].fitness<*min) { *min=pop.P[i].fitness; *indmin=i;}
}
*avg=*sumfitness/popszie;
}

```

```

//Estadístiques de la funció objectiu positiva sobre una població
void EstadisticObj(poblacio pop,int *indmax,int *indmin,float *avg,float
*min,float *max,float *sumfitness)
{
    int i;
    *indmax=0;
    *indmin=0;
    *sumfitness=pop.P[0].fobjectiu;
    *min=pop.P[0].fobjectiu;
    *max=pop.P[0].fobjectiu;
    for(i=1;i<popsize;i++)
    {
        *sumfitness=*sumfitness+pop.P[i].fobjectiu;
        if(pop.P[i].fobjectiu>*max) { *max=pop.P[i].fobjectiu; *indmax=i;}
        if(pop.P[i].fobjectiu<*min) { *min=pop.P[i].fobjectiu; *indmin=i;}
    }
    *avg=*sumfitness/popsize;
}

//OPERADORS GENETICS

//Selecció per Torneig introduint la variació de b(delta)
poblacio Seleccio_Torneig(int tournsize,poblacio pop,float *X0,float *X1)
{
    int i,j,ale1,ale2,numtorneig,numround,round;
    poblacio popfinal;
    float bale1,bale2;

    round=popsize/2;
    numtorneig=1;
    i=0;
    while(numtorneig<=tournsize)
    {
        numround=0;
        while(numround<round) {

            //S'escullen dos nombres aleatoris sense reemplacament
            ale1=intranuni(popsize);
            while(pop.P[ale1].triat==1) { ale1=intranuni(popsize); }
            pop.P[ale1].triat=1;

            ale2=intranuni(popsize);
            while((ale2==ale1) || (pop.P[ale2].triat==1)) { ale2=intranuni(popsize); }
            pop.P[ale2].triat=1;

            if(pop.P[ale1].fobjectiu>pop.P[ale2].fobjectiu+tol) {

                popfinal.P[i].crom=pop.P[ale1].crom;
                for(j=0;j<dim;j++) popfinal.P[i].x[j]=pop.P[ale1].x[j];
                popfinal.P[i].fitness=pop.P[ale1].fitness;
                popfinal.P[i].fobjectiu=pop.P[ale1].fobjectiu;
                popfinal.P[i].parent1=pop.P[ale1].parent1;
                popfinal.P[i].parent2=pop.P[ale1].parent2;
                popfinal.P[i].locus=pop.P[ale1].locus;
                popfinal.P[i].triat=0;
            }
            else if(pop.P[ale1].fobjectiu<pop.P[ale2].fobjectiu-tol) {

                popfinal.P[i].crom=pop.P[ale2].crom;
                for(j=0;j<dim;j++) popfinal.P[i].x[j]=pop.P[ale2].x[j];
                popfinal.P[i].fitness=pop.P[ale2].fitness;
                popfinal.P[i].fobjectiu=pop.P[ale2].fobjectiu;
                popfinal.P[i].parent1=pop.P[ale2].parent1;
                popfinal.P[i].parent2=pop.P[ale2].parent2;
                popfinal.P[i].locus=pop.P[ale2].locus;
                popfinal.P[i].triat=0;
            }
            else { //i aquest és el cas en que s'ha de trencar l'empat
        }
    }
}

```

```

//Radi en comptes de b per si no es vol
//usar la funció de repulsió
//bale1=pop.P[ale1].x[2];
//bale2=pop.P[ale2].x[2];

//Valor de b(delta)per cada punt
bale1=
b(pop.P[ale1].x[0],pop.P[ale1].x[1],pop.P[ale1].x[2],X0,X1);
bale2=
b(pop.P[ale2].x[0],pop.P[ale2].x[1],pop.P[ale2].x[2],X0,X1);

//restricció de minimització
if(bale1<=bale2) {
//restricció de maximització
//if(bale1>bale2) {

    popfinal.P[i].crom=pop.P[ale1].crom;
    for(j=0;j<dim;j++) popfinal.P[i].x[j]=pop.P[ale1].x[j];
    popfinal.P[i].fitness=pop.P[ale1].fitness;
    popfinal.P[i].fobjectiu=pop.P[ale1].fobjectiu;
    popfinal.P[i].parent1=pop.P[ale1].parent1;
    popfinal.P[i].parent2=pop.P[ale1].parent2;
    popfinal.P[i].locus=pop.P[ale1].locus;
    popfinal.P[i].triat=0;
}
else {
    popfinal.P[i].crom=pop.P[ale2].crom;
    for(j=0;j<dim;j++) popfinal.P[i].x[j]=pop.P[ale2].x[j];
    popfinal.P[i].fitness=pop.P[ale2].fitness;
    popfinal.P[i].fobjectiu=pop.P[ale2].fobjectiu;
    popfinal.P[i].parent1=pop.P[ale2].parent1;
    popfinal.P[i].parent2=pop.P[ale2].parent2;
    popfinal.P[i].locus=pop.P[ale2].locus;
    popfinal.P[i].triat=0;
}
}
numround++;
i++;
}
for(j=0;j<popszie;j++) { pop.P[j].triat=0;}
numtorneig++;
}

return popfinal;
}

//Inclou Creuament, Mutacio i Actualització
poblacio Generacio(poblacio antpop,float sumfitness,float *X0,float *X1,float
*g,float *LB, float *UB)
{
    int i,j,k,locus,*subcadena;
    float aux;
    poblacio noupop;
    cromosoma *noucrom;

    for(i=0;i<popszie;i=i+2)
    {
        //Creuament
        noucrom=
        Creuament(antpop.P[i].crom,antpop.P[i+1].crom,pcros,&locus,lcrom);
        //Mutacio
        noupop.P[i].crom=Mutacio(noucrom[0],pmutacio,lcrom);
        noupop.P[i+1].crom=Mutacio(noucrom[1],pmutacio,lcrom);
        //Actualitzacio
        for(j=0;j<2;j++)
        {

```

```

    for(k=0;k<dim;k++)
    {
        subcadena=ExtreureVar(noupop.P[i+j].crom,k);
        aux=bit2float(subcadena,lgen);
        noupop.P[i+j].x[k]=Reescalat(aux,minstring,maxstring,LB[k],UB[k]);
    }

    noupop.P[i+j].fitness=
    fobj(noupop.P[i+j].x[0],noupop.P[i+j].x[1],noupop.P[i+j].x[2],X0,X1,g);
    noupop.P[i+j].parent1=i;
    noupop.P[i+j].parent2=i+1;
    noupop.P[i+j].locus=locus;
}
}

return noupop;
}

//Mutacio d'un cromossoma
cromosoma Mutacio(cromosoma crom,float pmutacio,int longcrom)
{
    int j;
    for(j=0;j<lcrom;j++){
        if(flip(pmutacio)) crom.C[j]=!crom.C[j];
    }
    return crom;
}

//Creuament de dos cromossomes
cromosoma *Creuament(cromosoma crom1,cromosoma crom2,float pcros,int
*locus,int longcrom)
{
    int j,xsite;
    cromosoma *noucromos;
    noucromos=(cromosoma *)malloc(2*sizeof(cromosoma));
    if(flip(pcros)){
        xsite=intranuni(longcrom);
        *locus=xsite;

        //Caps
        for(j=0;j<xsite;j++){
            noucromos[0].C[j]=crom1.C[j];
            noucromos[1].C[j]=crom2.C[j];
        }

        //Cues
        for(j=xsite;j<lcrom;j++){
            noucromos[0].C[j]=crom2.C[j];
            noucromos[1].C[j]=crom1.C[j];
        }
    }
    else { noucromos[0]=crom1; noucromos[1]=crom2; *locus=0; }
    return noucromos;
}

//Operador d'elit
void Reinsercio(poblacio antpop,poblacio noupop,int indmaxact,int indmin)
{
    noupop.P[indmin]=antpop.P[indmaxact];
}

//Criteri de parada per aturar l'algorisme
int CriteriParada(float stop,int numgen,int maxgen,float maxact,float minact)
{
    float rati;
    if(triestop==1){

```

```

if(numgen>maxgen) {
    fprintf(dadw,"nNombre d'iteracions realitzades: %d\n",numgen-1);
    return 1;
}
else
{
    rati=minact/maxact;
    if(rati>=stop) {
        fprintf(dadw,"nHem parat pel criteri de convergencia.\n"
                "Nº iteracions:%d\n",numgen-1);
        return 1;
    }
    else return 0;
}

//FUNCIONS RANDOM AUXILIARS (les mateixes que en Annex 1)
float ranuni(float LB,float UB)
{
    enter32 a,c;
    float sol;
    a=3715318133;
    c=1;
    llavor_enter=a*llavor_enter+c;      //aixo ja pren modul de m.
    sol=(float)(1.0*llavor_enter);
    return LB+(sol/float_232)*(UB-LB);   // entre LB i UB
}

int intranuni(int fitsup)
{
    int x,aux;
    aux=(int)floor(float_232*ranuni(0.0,1.0));
    srand((unsigned)aux);
    x=(int)((1.0*rand())/(1.0*RAND_MAX))*fitsup;
    return x;
}

int flip(float p)
{
    if(p-1.0==0) return 1;
    return (ranuni(0.0,1.0)<=p);}

//DECODE i FUNCIO OBJECTIU

//De taula de bits a nombre real
float bit2float(int *subcadena,int longcrom)
{
    int j;
    float acum,powerof2;

    acum=0.0;
    powerof2=1.0;
    for(j=0;j<longcrom;j++)
    {
        if(subcadena[j]==1) { acum=acum+powerof2; }
        powerof2=powerof2*2;
    }
    return acum;
}

//Per situar el punt en l'espai de cerca
float Reescalat(float x, float antinf, float antsups, float nouinf, float nousup)
{
    float a,b;
    a=(nousup-nouinf)/(antsup-antinf);
    b=nouinf-a*antinf;
}

```

```

    return a*x+b;
}

//Extracció d'un gen de dins d'un cromosoma
int *ExtreureVar(cromosoma crom,int posicio)
{
    int j,*subcadena,inici,final;
    subcadena=(int*)malloc(lgen*sizeof(int));
    inici=lgen*posicio;
    final=inici+lgen;
    for(j=inici;j<final;j++) subcadena[j-inici]=crom.C[j];
    return subcadena;
}

//TOT EL QUE TE A VEURE AMB LA FUNCIO OBJECTIU
//Funció objectiu (permets tota la casuística)
float fobj(float x1,float x2,float delta,float *z0,float *z1,float *g)
{
    int *Ind_P,*Ind_noP,card_P,card_noP,card_tot,i;
    float *x,*Xiact,dist,invbeta,mean_P,var_P,mean_noP,var_noP;
    float f,g0,aux1,aux2;

    Ind_P=(int*)malloc(realsize*sizeof(int));
    Ind_noP=(int*)malloc(realsize*sizeof(int));

    x=(float*)malloc(Q*sizeof(float));
    x[0]=x1;
    x[1]=x2;
    Xiact=(float*)malloc(Q*sizeof(float));

    g0=g[0];
    for(i=0;i<realsize;i++){
        Ind_P[i]=0;Ind_noP[i]=0;
        if(triaf==2) { if(g[i]<g0) g0=g[i];}
    }

    for (i=0;i<realsize;i++)
    {
        Xiact[0]=z0[i];
        Xiact[1]=z1[i];
        dist=distancia(x,Xiact,Q);
        if (dist<delta) Ind_P[i]=1;
        else Ind_noP[i]=1;
    }
    Stat_Eventness(g,Ind_P,&card_P,&mean_P,&var_P);
    if(triaopti==1)
    {
        if (triaf==1){
            Stat_Eventness(g,Ind_noP,&card_noP,&mean_noP,&var_noP);
            aux1=mean_P-mean_noP;
            if(card_P==0) aux2=(var_noP/card_noP);
            else if(card_noP==0) aux2=(var_P/card_P);
            else aux2=(var_P/card_P)+(var_noP/card_noP);
            f=(float)(aux1/sqrt(aux2));
        }
        else {
            invbeta=(float)(1.0)*(1/beta);
            card_tot=realsize;
            aux1=(float)card_P/(float)card_tot;
            if (aux1>=beta) { f=mean_P; }
        }
    }
}
```

```

    else { f=(mean_P-g0)*aux1*invbeta)+g0; }
}
}
else f=-mean_P;

free(x);
free(Xiact);

return f;
}

//Funció de repulsió per moderar el radi del cluster
float b(float x1,float x2,float delta,float *X0,float *X1)
{
    int i,*P,*NP,cP,cNP;
    float *vi,*fi,*sumf,*zeros,summP,summNP,norma,bvalor,hi,mi;
    P=(int*)malloc(realsize*sizeof(int));
    NP=(int*)malloc(realsize*sizeof(int));
    for(i=0;i<realsize;i++) { P[i]=0;NP[i]=0; }
    cP=0;
    cNP=0;

    zeros=(float*)malloc(Q*sizeof(float));
    vi=(float*)malloc(Q*sizeof(float));
    fi=(float*)malloc(Q*sizeof(float));
    sumf=(float*)malloc(Q*sizeof(float));

    //Part que fem per tots els punts de l'espai
    zeros[0]=0;zeros[1]=0;
    sumf[0]=0;
    sumf[1]=0;
    for(i=0;i<realsize;i++){
        vi[0]=X0[i]-x1;
        vi[1]=X1[i]-x2;
        norma=distancia(vi,zeros,Q);
        if(norma<delta) { P[i]=1; cP++; }
        else { NP[i]=1; cNP++; }

        hi=(float)(fabs(delta-norma));
        mi=1/(hi*hi); //en general 1/(hi^p)
        if(hi<=delta) { fi[0]=mi*(vi[0]/norma); fi[1]=mi*(vi[1]/norma); }
        else { fi[0]=-mi*(vi[0]/norma); fi[1]=-mi*(vi[1]/norma); }
        sumf[0]=sumf[0]+fi[0];
        sumf[1]=sumf[1]+fi[1];
    }

    //Part que fem pels punts de dins del cluster
    summP=0;
    for(i=0;i<cP;i++){
        if(P[i]==1){
            vi[0]=X0[i]-x1;
            vi[1]=X1[i]-x2;
            norma=distancia(vi,zeros,Q);

            hi=(float)(fabs(delta-norma));
            mi=1/(hi*hi); //en general 1/(hi^p)
            summP=summP+mi;
        }
    }
    //Part que fem pels punts de fora del cluster
    summNP=0;
    for(i=0;i<cNP;i++){
        if(NP[i]==1){
            vi[0]=X0[i]-x1;
            vi[1]=X1[i]-x2;
            norma=distancia(vi,zeros,Q);

            hi=(float)(fabs(delta-norma));
        }
    }
}

```

```

    mi=1/(hi*hi); //en general 1/(hi^p)
    summNP=summNP+mi;
}
}
norma=distancia(sumf,zeros,Q);
bvalor=(float)(norma+fabs(summP-summNP));

free(P);
free(NP);
free(vi);
free(fi);
free(sumf);

return bvalor;
}

//FUNCIONS D'ESCRIPCIURA

//Escriure un cromosoma
void WriteCrom(cromosoma crom,int longcrom)
{
    int i,aux;

    for(i=0;i<longcrom;i++) { aux=crom.C[i]; fprintf(dadw,"%d",aux);}

}

void SortidaInicial(poblacio antpop,float avg,float min,float max,float
sumfitness,float *LB,float *UB)
{
    int i;

    fprintf(dadw,"\nPOBLACIO INICIAL\n");

    fprintf(dadw,"nFites:\n");
    for(i=0;i<dim-1;i++) fprintf(dadw,"nx{i} pertany a
[%f,%f]\n",i,LB[i],UB[i]);
    fprintf(dadw,"nradi pertany a [%f,%f]\n",LB[2],UB[2]);

    fprintf(dadw,"nCROMOSOMES\n");
    for(i=0;i<popsize;i++)
    {
        fprintf(dadw,"%i\t",i);
        WriteCrom(antpop.P[i].crom,lcrom);
        fprintf(dadw,"\n");
    }

    fprintf(dadw,"nFLOATS\n");
    fprintf(dadw,"nx1:\ttx2:\ttradi:\t\ttf:\n");
    for(i=0;i<popsize;i++)
    {

fprintf(dadw,"n%d\t%.8f\t%.8f\t%.8f\t%.12f\n",i,antpop.P[i].x[0],antpop.P[i].
x[1],antpop.P[i].x[2],antpop.P[i].fitness);
    }

    fprintf(dadw,"n\nESTADISTIQUES INICIALES:");
    fprintf(dadw,"nMitja: %f",avg);
    fprintf(dadw,"nMax: %f",max);
    fprintf(dadw,"nMin: %f",min);
    fprintf(dadw,"nSumfitness: %f",sumfitness);
}

```

```

void Sortida(poblacio antpop,int numgen,float avg,float min,float max,float
sumfitness)
{
    int i;

    fprintf(dadw,"\\n\\n-----GENERACIO %i-----",numgen);
    fprintf(dadw,"\\nCROMOSOMES:\\n");
    for(i=0;i<popsize;i++)
    {
        fprintf(dadw,"Crom %i:",i);
        WriteCrom(antpop.P[i].crom,lcrom);
        fprintf(dadw,"\\t\\t");

        //fprintf(dadw,"parel: %i\\tpare2: %i\\tlocus:
*i\\n",antpop.P[i].parent1,antpop.P[i].parent2,antpop.P[i].locus);
    }

    fprintf(dadw,"\\nFLOATS:");
    fprintf(dadw,"\\nnova x1:\\t\\tnova x2:\\t\\tnou radi:\\t\\tnova f:\\n");

    for(i=0;i<popsize;i++)
    {

fprintf(dadw,"\\n%d\\t%.8f\\t%.8f\\t%.8f\\t%.12f\\n",i,antpop.P[i].x[0],antpop.P[i].
x[1],antpop.P[i].x[2],antpop.P[i].fitness);
    }

    fprintf(dadw,"\\n\\nESTADISTIQUES NOVES:");
    fprintf(dadw,"\\nMitja: %f",avg);
    fprintf(dadw,"\\nMax: %f",max);
    fprintf(dadw,"\\nMin: %f",min);
    fprintf(dadw,"\\nSumfitness: %f",sumfitness);
}

void TractamentOptim(float x1,float x2,float delta,float *X0,float *X1,float
*g)
{
    int *Ind_P,*Ind_noP,card_P,card_noP,i;
    float *x,*Xiact,dist,mean_P,var_P,mean_noP,var_noP,zm;

    Ind_P=(int*)malloc(realsize*sizeof(int));
    Ind_noP=(int*)malloc(realsize*sizeof(int));

    for(i=0;i<realsize;i++)
    {
        Ind_P[i]=0;
        Ind_noP[i]=0;
    }

    x=(float*)malloc(Q*sizeof(float));
    x[0]=x1;
    x[1]=x2;

    Xiact=(float*)malloc(Q*sizeof(float));

    for (i=0;i<realsize;i++)
    {
        Xiact[0]=X0[i];
        Xiact[1]=X1[i];

        dist=distancia(x,Xiact,Q);
    }
}

```

```

if dist<delta) Ind_P[i]=1;
else Ind_noP[i]=1;

}

Stat_Eventness(g,Ind_P,&card_P,&mean_P,&var_P);
Stat_Eventness(g,Ind_noP,&card_noP,&mean_noP,&var_noP);
zm=(float) ((mean_P-mean_noP)/(sqrt((var_P/card_P)+(var_noP/card_noP))));



fprintf(dadw, "\nCaracterístiques del punt óptim:\n");
fprintf(dadw, "Cardinal de P:%d\n", card_P);
fprintf(dadw, "Mitjana en P:%f\n", mean_P);
fprintf(dadw, "Variància en P:%f\n", var_P);
fprintf(dadw, "Cardinal fora de P:%d\n", card_noP);
fprintf(dadw, "Mitjana fora de P:%f\n", mean_noP);
fprintf(dadw, "Variància fora de P:%f\n", var_noP);
fprintf(dadw, "Estadístic del test de mitjanes:%f\n", zm);

free(Ind_P);
free(Ind_noP);
free(x);
free(Xiact);
}

```

Programes Matlab

El programa principal és el programa TSDM.m que conté algunes funcions associades.

Què fa el TSDM.m?

- Representació gràfica de la sèrie a l'espai original, i als espais de retards.
- Avaluació del patró óptim: càlcul de les estadístiques, dels estadístics d'avaluació dels resultats i de les proporcions de mala classificació.
- Representació gràfica del patró óptim a l'espai dels retards i a l'espai original.
- Fase Test
- Representació gràfica dels resultats de la fase test.

El codi d'aquest programa és el següent:

```

%-----%
%-----%
[filename pathname]=uigetfile('.mat','Eixos i corbes de Aprenentatge',15,15);

%-----%
%-----%
figure
S=load([pathname,filename]);
[N,M]=size(S);

plot(S,'bd-','MarkerSize',4)
xlabel('t')
ylabel('wt')

```

CÒDIG DE PROGRAMACIÓ DEL ST-OPTIMI

```

% CÒDIG DE PROGRAMACIÓ DEL ST-OPTIMI

tri=questdlg('Quina funció avançada vols usar en l'espai de les projeccions?', 'Tm');
avançada='';
if strcmp(tri,'q(z)=z+s') triag=1;
elseif strcmp(tri,'q(z)=z-s') triag=2;
else triag=3;
end

if triag==1 s=1;
elseif triag==2 s=2;
else
    resp=inputdlg('Escríu el valor de s', 's per z-s',1);
    if strcmp(resp{1}, '') disp('Error! ');
    else s=str2num(resp{1});
    end
end

tria=questdlg('Quina distància vols usar en l'espai de les projeccions?', 'Tm');
distancia='';
if strcmp(tria,'Distància euclídea', 'Distància ll') triadist=1;
else triadist=2;
end

[X0,X1,g]=TractamentInicial(S,triag,s);

figure
plot(X0,X1,'o', 'MarkerEdgeColor', 'k', 'MarkerFaceColor', 'k')
 xlabel('x-1')
 ylabel('x')
 grid

figure
stem3(X0,X1,g,'r', 'fill')
 xlabel('x-1')
 ylabel('x')
 zlabel('q(z)=z+s')
 if triag==1 zlabel('q(z)=z+s+1')
 elseif triag==2 zlabel('q(z)=z-s+1')
 end

tria=questdlg('Vols evaluar els resultats?', 'avaluació', ...
{'Si', 'No'});
if strcmp(tria,'Si') aval=1;
else aval=2;
end

aval==1

tria=questdlg('Vols detectar punts o mínims?', 'tria optimització', ...
{'míms', 'imíms', 'Minims'});
if strcmp(tria,'Maxims') triaopti=1;
else triaopti=2;
end

resp=inputdlg('Liindar pvr i maxim o minima pvr? ', 'llindar', 1);
if strcmp(resp{1}, '') disp('Error! ');
else llindar=str2num(resp{1});
end

```

```

% Load optim values
optim=load('res_optim.mat');

temporal_pattern=optim(1,1:2);
radi=optim(1,3);
numiter=optim(2,1);

% Create X0 matrix
[N,ncol]=size(X0);
P=[];
NP=[];
for i=1:N
    Xiact=[X0(i),X1(i)];
    if (triadist==i) distancia=sqrt((temporal_pattern(i)-Xiact(1))^2+(temporal_pattern(2)-Xiact(2))^2);
    else distancia=abs(temporal_pattern(1)-Xiact(1))+abs(temporal_pattern(2)-Xiact(2));
    end;
    if (distancia<radi) P=[P;i];
    else NP=[NP;i];
    end;
end;

% Create P and NP matrices
sp=size(P);
cardP=sp(1);
[meanP,varP]=stat_eventness(g,P);

snp=size(NP);
cardNoP=snp(1);
[meanNoP,varNoP]=stat_eventness(g,np);

zm=(meanP-meanNoP)/sqrt((varP/cardP)+(varNoP/cardNoP));
if (triaopti==2) zm=abs(zm); end;
alpham=1-normcdf(zm);

% Plotting
figure
hold on
plot(X0,X1,'.','MarkerSize',10,'MarkerFaceColor','r')
plot(temporal_pattern(1),temporal_pattern(2),'s','MarkerSize',10,'MarkerFaceColor','b','MarkerEdgeColor','k','MarkerStyle','filled');
if (triadist==1) plot_circumf(temporal_pattern,radi)
if triap==1; plotcluster(temporal_pattern,radi,triap)
end;
xlabel('x1')
ylabel('x2')
grid
hold off
legend('Sèrie','Patró Temporal','Cluster')

% Plotting S
longi=size(S);
figure
hold on
grid
plot(S,'bd-', 'MarkerSize',4)
xlabel('x1')
ylabel('x2')
for i=1:cardP
    j=P(i);
    if(j<longi(1))
        temps=[j;j+1];
        patro=[temporal_pattern(1);temporal_pattern(2)];
        plot(temps,patro,'-','LineWidth',1.5,'MarkerEdgeColor','k','MarkerFaceColor','b','MarkerSize',6);
    end;
    if (triag==1)
        plot(j+2,g(j),'r*','MarkerSize',6)
        legend('x1','x2','Event predict',4);
    else % (triag==2)
        plot(j+3,g(j),'r*','MarkerSize',6)
        legend('x1','x2','Event predict',4);
    end;
    plot(j+s+1,g(j),'r*','MarkerSize',6)
end;

```

```

legend('txt','ps','Event predict',4)
;
;
hold off

%----- Variables de l'estat -----%
[N,m]=size(X0);
nevents=0;nfalseneg=0;nfalsepos=0;
for triopti=1
    for i=1:N
        if(S(i)>llindar) nevents=nevents+1; end;
    end;
    for i=1:cardP
        j=P(i);
        if(g(j)<llindar) nfalseneg=nfalseneg+1; end;
    end;
    for i=1:cardNoP
        j=NP(i);
        if(g(j)>llindar) nfalsepos=nfalsepos+1; end;
    end;
end;
for i=1:N
    if(S(i)<llindar) nevents=nevents+1; end;
end;
for i=1:cardP
    j=P(i);
    if(g(j)>llindar) nfalsepos=nfalsepos+1; end;
end;
for i=1:cardNoP
    j=NP(i);
    if(g(j)<llindar) nfalseneg=nfalseneg+1; end;
end;
end;
pfalseneg=nfalseneg/(N-nevents);
pfalsepos=nfalsepos/nevents;

%----- Resultats -----%
fprintf('RESULTATS DE LA PRUEBA DE APROXIMACIÓ ...');
fprintf('Patró Temporal optim p, x0: t1...t1',temporal_pattern(1));
fprintf('Patró Temporal optim p, x0: t1...t1',temporal_pattern(2));
fprintf('Radi Optimitzat',radi);
fprintf('Cardinal de P: t1...t1',cardP);
fprintf('Mitjana de q en P: t1...t1',meanP);
fprintf('Variància de q en P: t1...t1',varP);
fprintf('Cardinal fora de P: t1...t1',cardNoP);
fprintf('Mitjana de q fora de P: t1...t1',meanNoP);
fprintf('Variància de q fora de P: t1...t1',varNoP);
fprintf('Estadístic del test de mitjanes: t1...t1',zm);
fprintf('Significació: t1...t1',alpham);
fprintf('Interaccions: t1...t1',numiter);

fprintf('Nombre de èvents a priori: t1',nevents);
fprintf('Nombre de NO èvents a priori: t1',N-nevents);

fprintf('Nombre de èvents mal classificats: t1',nfalseneg);
fprintf('Probabilitat de mal classificació positiva: t1',pfalseneg);
fprintf('Nombre de NO èvents mal classificats: t1',nfalsepos);
fprintf('Probabilitat de mal classificació negativa: t1',pfalsepos);

%----- Voltes que es realitza la prova -----%
voltest=questdig('Volts que es realitzi la prova result', 'Diu Tot:', 'Si', 'No', 'No');
if strcmp(voltest,'Si')
    [filename, pathname]=uigetfile('..','Entree sobre t1',15,15);
    St=load([pathname,filename]);
    s=St(:,1);
    s=s'; % perquè el vector està en columnes
    s=sort(s);
    s=[s zeros(1,10)];
    Xit=Xt;
    [Xit,Xlt,gt]=TractamentInicial(St,triag,s);
    %----- Prova -----%
    [Pt,NPt]=test(X0t,Xlt,gt,radi,temporal_pattern,triadist);
    %----- Estadística -----%
    spt=size(Pt);
    card3Pt=spt(1);
    [meanPt,varPt]=stat_eventness(gt,Pt);

```

```

snpt=size(NPt);
cardNoPt=snpt(1);
[meanNoPt,varNoPt]=stat_eventness(gt,NPt);

zmt=(meanPt-meanNoPt)/sqrt((varPt/cardPt)+(varNoPt/cardNoPt));
%if triaopti==2) zmt=abs(zmt); %;
alphamt=l-normedf(zmt);

%-----%
[N,m]=size(X0t);
nevents=0;
nfalsepos=0;
nfalseneg=0;
%*(triaopti==1)
*(i=1:N) *(St(i)>llindar) nevents=nevents+1; % ;
*(i=1:cardPt)
j=Pt(i);
*(gt(j)<llindar) nfalsepos=nfalsepos+1; % ;
;
*(i=1:cardNoPt)
j=NPt(i);
*(gt(j)>llindar) nfalseneg=nfalseneg+1; % ;
%*%
%*(i=1:N) *(St(i)<llindar) nevents=nevents+1; % ;
*(i=1:cardPt)
j=Pt(i);
*(gt(j)>llindar) nfalsepos=nfalsepos+1; % ;
;
*(i=1:cardNoPt)
j=NPt(i);
*(gt(j)<llindar) nfalseneg=nfalseneg-1; % ;
;
;
pfalsepos=nfalsepos/nevents;
pfalseneg=nfalseneg/(N-nevents);

%-----%
fprintf('ANALISIULTATS DE LA PRUEBA ESTADISTICA');
fprintf('Cardinal de g en P: ',cardPt);
fprintf('Mitjana de g en P: ',meanPt);
fprintf('Variancia de g en P: ',varPt);
fprintf('Cardinal d\'una de P: ',cardNoPt);
fprintf('Mitjana de g d\'una de P: ',meanNoPt);
fprintf('Variancia de g d\'una de P: ',varNoPt);
fprintf('Estadistic del test de mitjanes: ',zmt);
fprintf('Significació: ',l-normedf(zmt));

fprintf('Nombre de events a priori 1: ',nevents);
fprintf('Nombre de NO events a priori 1: ',N-nevents);

fprintf('Nombre de events mal classificats: ',nfalsepos);
fprintf('Probabilitat de mal classificació positiva: ',pfalsepos);
fprintf('Nombre de NO events mal classificats: ',nfalseneg);
fprintf('Probabilitat de mal classificació negativa: ',nfalseneg);

%-----%
longi=size(St);
figure
hold on
grid
plot(St,'*o-','MarkerSize',4)
xlabel('t')
ylabel('y')
for i=1:cardPt
    j=Pt(i);
    if j<longi(1)
        temps=[j;j+1];
        patro=[temporal_pattern(1);temporal_pattern(2)];
        plot(temps,patro,'-'
        'Linewidth',1.5,'MarkeredgeColor','k','MarkerFaceColor','k','MarkerSize',6);
    *(triag==1)
        plot(j+2,gt(j),'*r','MarkerSize',6)
        legend('1st','2nd','Event pt=i-1',4)
    else*(triag==2)
        plot(j+3,gt(j),'*r','MarkerSize',5)
    end
end

```

```

        legend('gt','pt','Event predict',4)
        plot(j+s+1,gt(j),'r+', 'MarkerSize',6)
        legend('gt','pt','Event predict',4)
    end;
end;
hold off
end;

```

Ja no es presenta el codi de les funcions que s'usen dins aquest programa, però l'objectiu de cadascuna d'elles és:

```
...,[past,actual,funcio]=TractamentInicial(serie,triag,s)
```

Al igual que en el programa amb C, calcula els retards i la funció d'events.

```
...,[Ind_Pt,Ind_noPt]=test(20,z1,g,delta,tp,dis)
```

Calcula, donat un punt, quins punts pertanyen al cluster de centre aquest punt i radi delta.

```
,r [aver,varij]=stat_eventness(g,set)
```

Càcul de la mitjana i varància de g en el conjunt corresponent.

```
...clust []=plot_circumf(centre,radi)
...clust []=plotcluster(centre,radi,p)
...for [y]=ydist(x,centre,radi,p)
```

Ajuden a dibuixar el cluster a l'esapi dels retards.

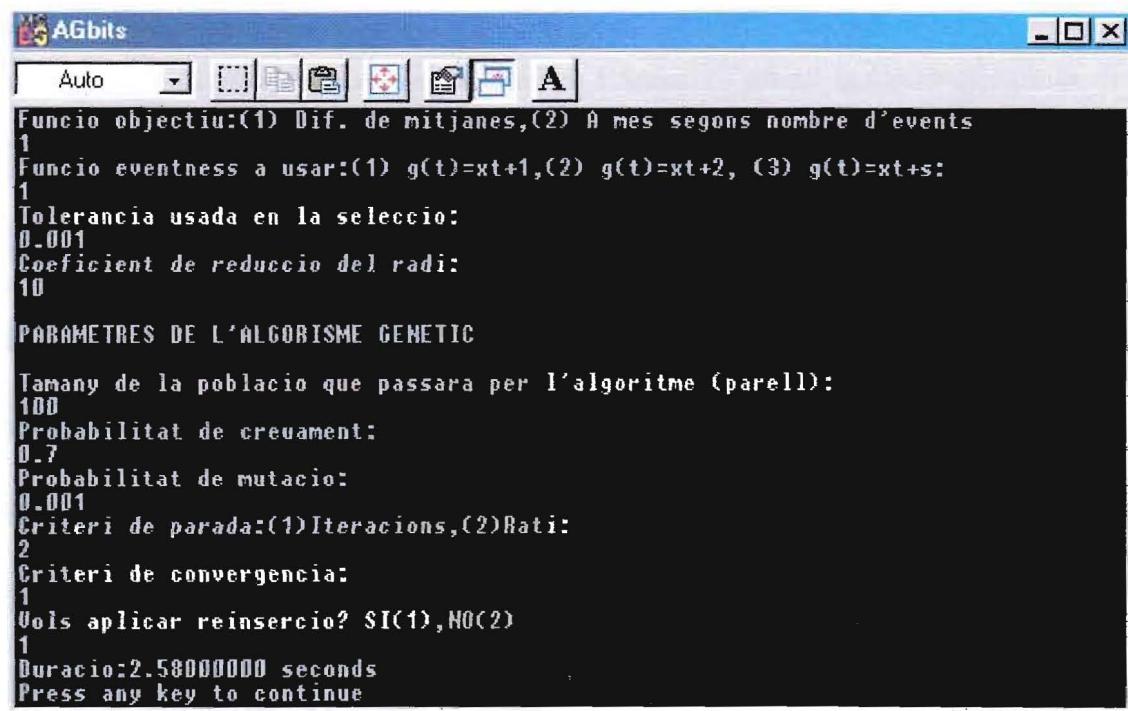
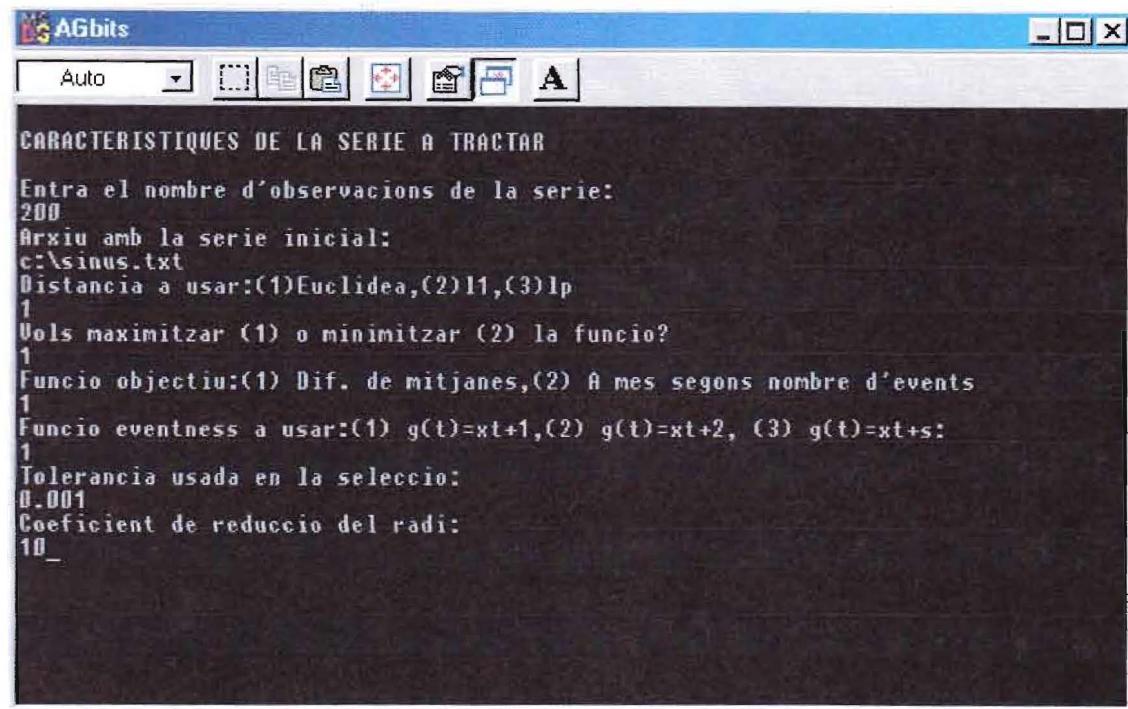
A2.3 Exemple d'execució

Es presenta tot seguit un exemple de tot el procés del mètode del TSDM. Es realitza l'exemple amb la sèrie Sinusoïdal introduida al capítol 6, amb les paràmetres de les taula 6.1 i 6.2.

Prèviament s'haurà separat la sèrie en sèrie d'aprenentatge i sèrie test.

Execució de TSDM.c

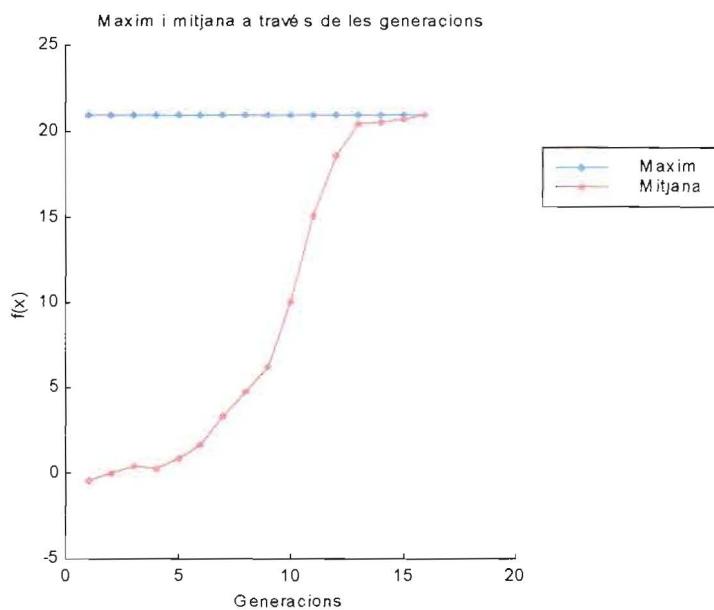
S'executa el programa TSDM.c i s'introduceixen tots els paràmetres que demana. La pantalla d'introducció de paràmetres és la següent:



Ja s'ha executat el programa. Tal com s'ha explicat abans s'obtenen 3 arxius.

El primer, res.txt, és un arxiu semblant al de l'Annex 1, en què es veu l'evolució de la població fins arribar a l'òptim. Com que en aquest cas la població és de 100 individus i l'arxiu té unes dimensions molt grans no s'exposen aquests resultats. El que si que es té, al igual que en l'Annex 1 és un arxiu amb l'evolució a través de les generacions de la mitjana i el màxim de

la funció objectiu, l'arxiu `resgraf.txt`. Si es representen gràficament donen idea de com ha anat la convergència del mètode.



S'observa com efectivament hi ha convergència en el mètode. Verifica si el valor al qual convergeix és correcte és més complicat que en l'Annex 1 perquè la funció objectiu és més complexa que en l'Annex 1 i no es pot calcular sense realitzar calculs iteratius amb l'ajuda de l'ordinador.

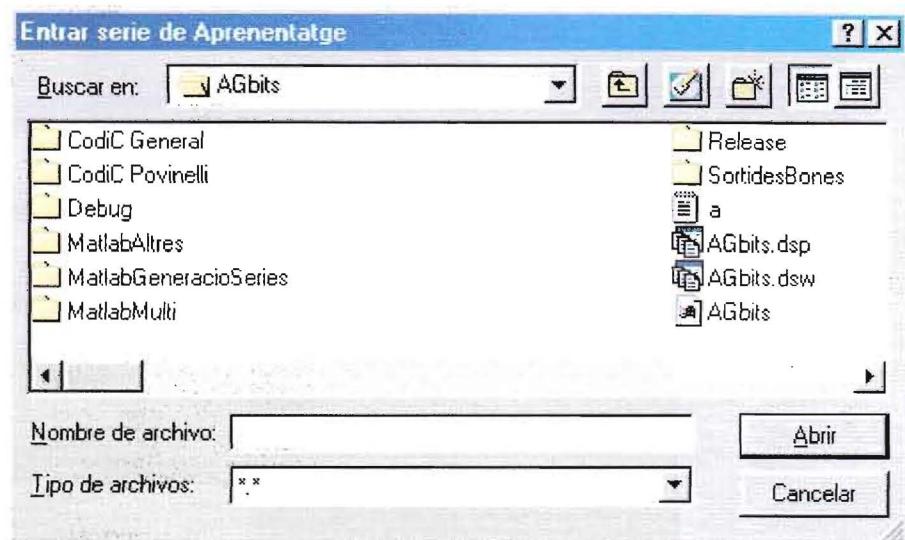
L'últim dels arxius és `resoptim.txt` en el qual hi l'òptim obtingut i el nombre d'iteracions. És l'arxiu que llegirà el programa Matlab per tal d'avaluar la bondat del patró obtingut i realitzar la fase test.

Execució de `TSDM.m`

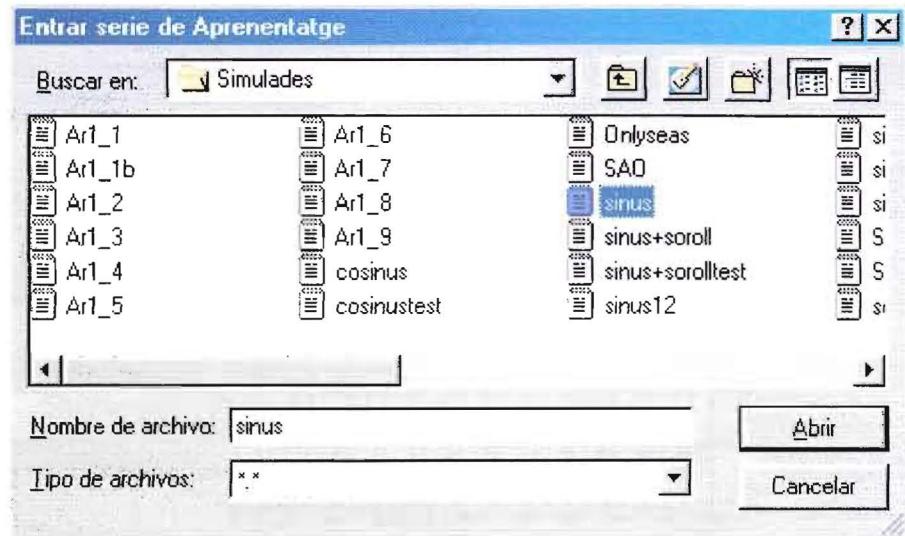
La crida que s`efectua des de Matlab és tan sols:

`>> TSDM`

Aleshores apareix la pantalla següent:

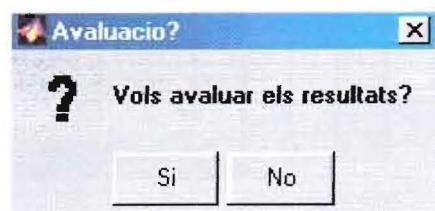
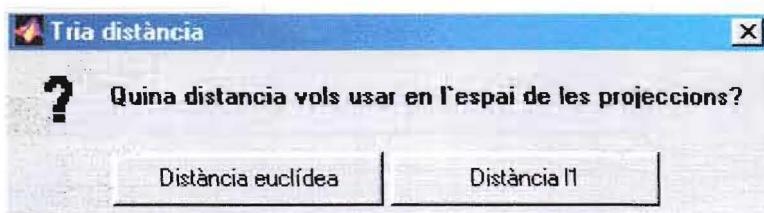


Es tria quina ha estat la sèrie usada en la fase d'aprenentatge:

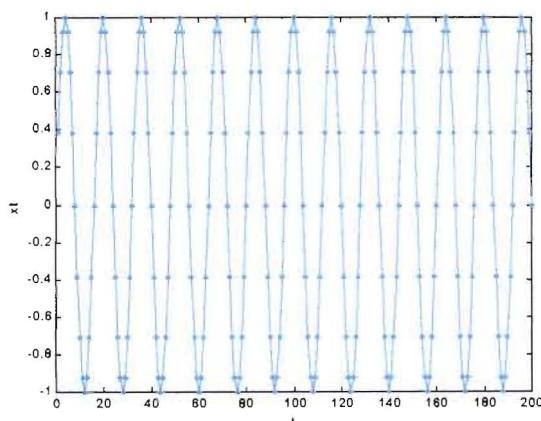


Tot seguit es comencen a demanar els paràmetres que s'han d'usar:

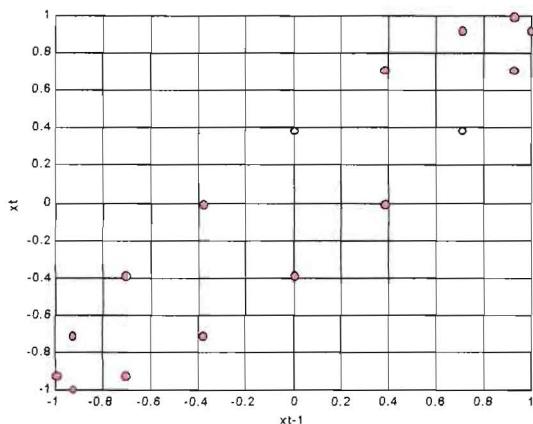
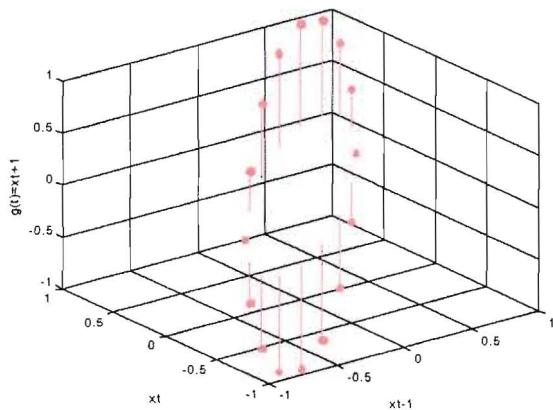




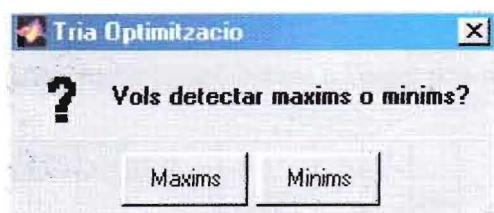
Es demana si es volen avaluar els resultats perquè aquesta funció pot servir també només per dibuixar la sèrie en l'espai del retards i en l'espai augmentat. Si ara es tria la opció **No**, els gràfics que ja es tenen són:



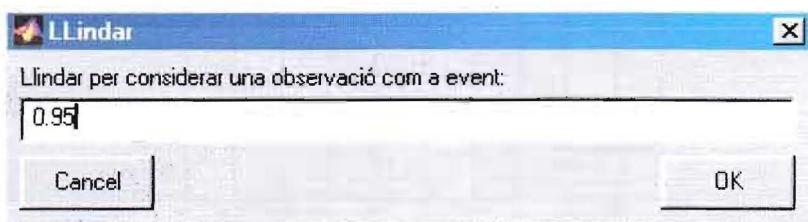
Sèrie a l'espai original

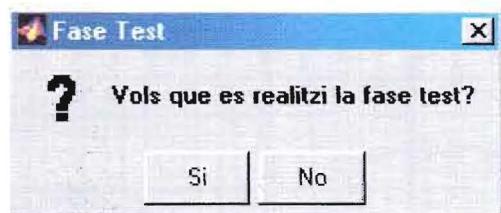
*Espai dels retards**Espai dels retards augmentat*

Si es decideix que si que es volen avaluar els resultats el procés continua:



Es tria el llindar d'events:



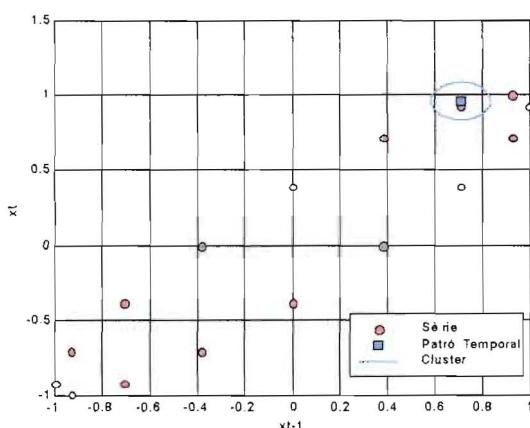


Si no es vol realitzar la fase test, els resultats per la fase d'aprenentatge ja han estat evaluats tant numèricament com gràficament:

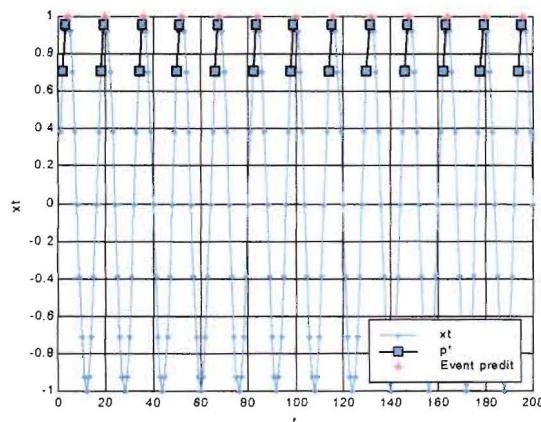
```
MATLAB Command Window
File Edit Window Help
RESULTATS DE LA FASE DE APRENENTATGE
Patró Temporal optim p, xt-1: 0.7098
Patró Temporal optim p, xt: 0.9608
Radi Optim: 0.1255
Cardinal de P: 13.0000
Mitjana de g en P: 1.0000
Variància de g en P: 0.0000
Cardinal fora de P: 185.0000
Mitjana de g fora de P: -0.0490
Variància de g fora de P: 0.4644
Estadístic del test de mitjanes: 20.9373
Significació: 0.00000000
Iteracions: 15

Nombre de events a priori: 13
Nombre de NO events a priori: 185
Nombre de events mal classificats: 0
Probabilitat de mal classificació positiva 0.0000
```

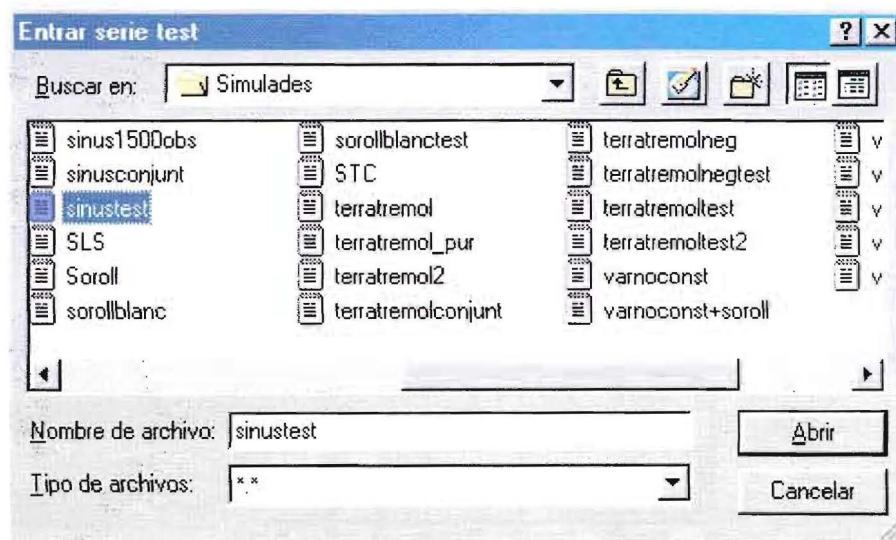
I els gràfics de representació del cluster i patró òptims a l'espai dels retards i l'espai original:



Espai dels retards

*Espai original*

Si es tria que si que es vol realitzar la fase test es continua el procés, havent de triar quina és la sèrie test:



Arribats a aquest punt si que el procés ha finalitzat. S'obtenen per pantalla els resultats de la sèrie test i el gràfic de la sèrie test junt amb el patró òptim i els events predicts.

MATLAB Command Window

File Edit Window Help

RESULTATS DE LA FASE TEST

Cardinal de P: 6.0000

Mitjana de g en P: 1.0000

Variancia de g en P: 0.0000

Cardinal fora de P: 92.0000

Mitjana de g fora de P: -0.0861

Variancia de g fora de P: 0.4693

Estadístic del test de mitjanes: 15.2080

Significació: 0.00000000

Nombre de events a priori: 6

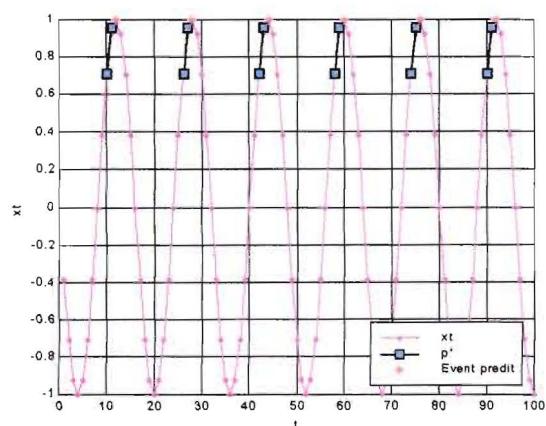
Nombre de NO events a priori: 92

Nombre de events mal classificats: 0

Probabilitat de mal classificació positiva 0.0000

Nombre de NO events mal classificats: 0

Probabilitat de mal classificació negativa 0.0000



Espai original:sèrie test

ANNEX 3. MILLORES EN L'EFICIÈNCIA DE L'ALGORISME

Tal com s'ha anat esmentat en alguns dels capítols, una de les possibles millores que es podria fer en el programa TSDM.c té a veure amb emmagatzemar de manera eficient els valors de la funció objectiu dins el procés de l'algorisme genètic. Aquesta millora permetria la cerca, de manera més ràpida i amb menys cost en espai, de patrons temporals d'ordre Q superior a 3.

Abans d'explicar en què consisteix aquesta millora es justifica el fet que per Q grans el mètode perdi en eficiència.

Càcul del cost l'algorisme

S'ha calculat quin és el cost de l'algorisme per veure si realment depèn de la longitud del patró temporal buscat. Tot seguit es presenten totes les funcions del programa TSDM.c junt amb el seu ordre d'eficiència i després el cost total del mètode.

Per continuar amb la nomenclatura usada en el programa la notació que s'usarà és la següent:

- *realsize*: nombre d'observacions de la sèrie a tractar.
- *popsize*: nombre d'individus en la població de l'algorisme genètic.
- *Q*: ordre del patró temporal buscat.
- *lcrom*: longitud del cromosoma (que de fet és $Q \cdot lgen$ i *lgen* és el que es fixa al començament i que en TSDM.c s'ha fixat a 8)
- *numgen*: nombre de generacions (iteracions) de l'algorisme genètic.

Funcions que tenen a veure amb el tractament inicial de la sèrie, pròpies del TSDM

- EntradaDades : $\theta(1)$ és a dir constant.
- TractamentInicialSerie: $\theta(\text{realsize}^2)$ és a dir quadràtic en el nombre d'observacions de la sèrie.
- Stat_Eventness: $\theta(\text{cardset})$ on *cardset* és el cardinal del conjunt sobre el qual treballa la funció. Com que com a molt aquest conjunt tindrà *realsize*-1 elements (cas en què el cluster òptim té un sol element) el cost de la funció és $\theta(\text{realsize})$.
- Retards: $\theta((Q+1) \cdot \text{realsize})$.
- distancia: $\theta(Q)$

Funcions auxiliars del algoritmes genètics

- IniPop: $\theta(Q \cdot \text{popsize} \cdot \text{realsize})$.
- Estadistic, EstadisticObj: $\theta(\text{popsize})$.

Operadors genètics

- Seleccio_Torneig: en el millor cas $\theta(\text{realsize})$ i en el pitjor (quan s'ha de calcular la funció $b(\delta)$) $\theta(\text{realsize}^2)$.
- Generacio: $\theta(Q \cdot \text{popsize} \cdot \text{realsize})$.
- Mutacio: $\theta(lcrom)$
- Creuament: $\theta(lcrom)$
- Reinsercio: $\theta(1)$
- CriteriParada: $\theta(1)$

Funcions random auxiliars

- ranuni, intranuni, flip: $\theta(1)$.

Funcions de decodificació i reescalat

- bit2float: $\theta(lcrom)$.
- Reescalat: $\theta(1)$.
- ExtreureVar: $\theta(lcrom)$.

Funcions que tenen a veure amb la funció objectiu

- fobj: $\theta(Q \cdot \text{realsize})$.
- b: $\theta(Q \cdot \text{realsize})$.

Funcions d'escriptura i tractament de l'òptim

- Sortida, SortidaInicial: $\theta(\text{popsize})$.
- WriteCrom: $\theta(lcrom)$.
- TractamentOptim: $\theta(Q \cdot \text{realsize})$.

Cal destacar que hi ha moltes funcions que no haurien de ser tant costoses per si mateixes, com per exemple IniPop o Generacio, però que inclouen una crida a la funció objectiu i com aquesta és molt costosa, fa augmentar-ne el cost.

Es comença a veure, tal com s'estableix a [20] que la funció objectiu és la que s'endú molta part del cost de l'algorisme.

El cost total de l'algorisme es calcula de la manera següent:

- **Incialització:** utilitza les funcions TractamentInicialSerie, IniPop,

Estadistic i EstadisticObj amb la qual cosa el seu cost es calcula com:

$$\theta(\text{realsize}^2) + \theta(Q \cdot \text{popsize} \cdot \text{realsize}) + \theta(\text{popsize}) = \theta(\text{Max}\{\text{realsize}^2, Q \cdot \text{popsize} \cdot \text{realsize}, \text{popsize}\})$$

- **Tractament pas a pas de la població:** és un bucle de *numgen* iteracions amb *numgen* desconeguda si no és que es fixa inicialment. Cada iteració del bucle té un cost de:

$$\theta(\text{Max}\{\text{realsize}^2, Q \cdot \text{popsize} \cdot \text{realsize}, \text{popsize}\})$$

i per tant el bucle sencer tindrà una cost de:

$$\text{numgen} \cdot \theta(\text{Max}\{\text{realsize}^2, Q \cdot \text{popsize} \cdot \text{realsize}, \text{popsize}\})$$

Així, l'algorisme sencer costa $\text{numgen} \cdot \theta(\text{Max}\{\text{realsize}^2, Q \cdot \text{popsize} \cdot \text{realsize}, \text{popsize}\})$.

Normalment $\text{popsize} \ll \text{realsize}$ ja que a més interessa aplicar el mètode a sèries de gran volum, per tant es pot dir que el cost és:

$$\text{numgen} \cdot \theta(Q \cdot \text{realsize}^2 \cdot \text{popsize})$$

S'aprecia per tant, com el cost augmenta proporcionalment al valor de *Q* tal com es volia veure. A més, com que la funció objectiu del TSDM depèn directament de *realsize* i aquest és un nombre que no es pot canviar com per exemple *popsize*, calcular-la de manera eficient o només calcular-la quan sigui necessari pot portar molts beneficis.

Solució proposada

La idea que es presenta en [20] és la següent: per la pròpia naturalesa dels algorismes genètics, la diversitat dels cromossomes d'una població a l'altra va disminuint a mesura que l'algorisme avança cap a l'òptim. Per tant, la funció objectiu es cridarà moltes vegades pel mateix cromossoma. Si hi hagués alguna manera de poder guardar tots els possibles valors que pot prendre la funció objectiu i consultar-los quan fos necessari, s'evitaria cada cop calcular la funció objectiu, cosa que com s'ha vist és molt costosa. El problema radica en què per exemple per *Q=4* el nombre de valors de la funció objectiu que caldria guardar és massa gran per poder-lo enmagatzemar en una taula: per *Q=4* la longitud del cromossoma és de $5 \cdot lgen = 5 \cdot 8 = 40$.

L'espai de cerca per tant té 2^{40} punts per als quals caldria enmagatzemar el valor corresponent de la funció objectiu. Però 2^{40} és de l'ordre de 10^{12} nombre que no pot ser enmagatzemat en una taula de manera eficient. Cal doncs alguna altra estructura de dades. El que es proposa a [20] és usar una taula de hashing.

Una descripció exhaustiva de l'estructura de dades Taula de Hashing es pot trobar a [27]. El que cal destacar és que les taules de hashing permeten buscar un element ,afegir-ne un de nou, i esborrar-ne en temps constant i que la memòria que utilitzen és proporcional al nombre d'elements que s'hi guarden. Per tant, en un cert moment de l'algorisme genètic en què es volgués saber el valor de la funció objectiu per un cert cromossoma, en comptes de calcular-lo només caldria anar a la taula de hashing, buscar aquest element i utilitzar el valor de la funció objectiu que s'havia enmagatzemat, cosa que es podria fer en temps constant. En la implementació realitzada, cada cop que es cridava a la funció objectiu es realitzava una operació de cost $\theta(Q \cdot \text{realsize})$, per tant la millora és substancial.

Com ja s'ha esmentat en el treball , una de les línies en les què es podria treballar per millorar el mètode és per tant, la introducció de les taules de hashing.