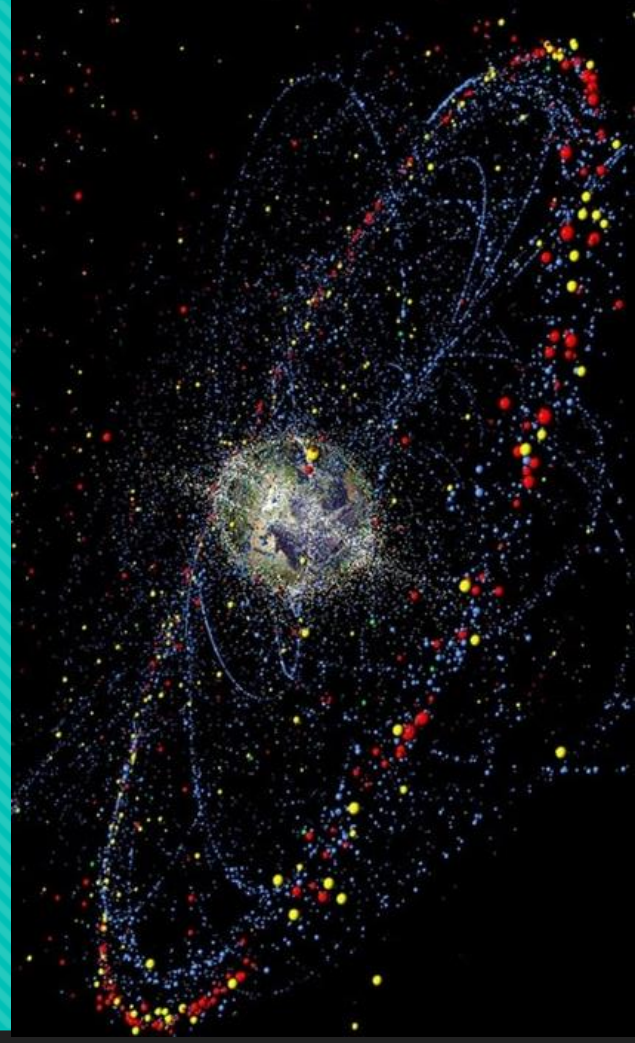
Estaciones Satelitales



Evaluación Continua

Asignatura: SI

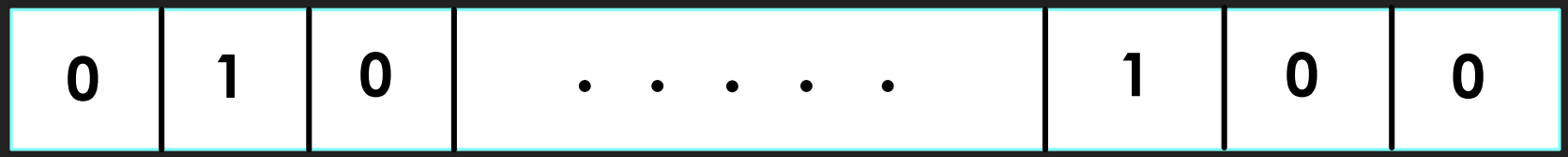
Participantes:
Iñaki Urrutia
Rafael Gutiérrez
Rafael Román

Indice

- Estructura de Estados
- Algoritmo Genético
  - Parámetros
  - Composición
- Algoritmo Taboo
  - Parámetros
  - Composición
- Gráficas

Estructura de Datos

- Para representar el espacio de estados, utilizaremos un vector binario de longitud NSatels (NSatels siendo el número de Satélites) donde el 1 en la posición **i** describe al i-ésimo satélite como representante. Y el 0 en la posición **j** significa que el satélite j-ésimo es un representado.



Estructura de Datos

- Para aumentar la eficiencia y reducir el tiempo de ejecución del programa, al inicio de este hemos creado una matriz **distancias** donde almacenaremos la distancia entre cada uno de los satélites. (Esto se realiza en Matlab mediante el comando *dist)*
- Además, la posición aleatoria de cada satélite se almacenará en **matPos**

Algoritmo Genético

Parámetros

- **Variables de población:**
  - **Pob**: generación actual
  - **NPob**: numero de individuos de cada generación
  - **FitPob**: evaluacion de cada individuo de la poblacion
  - **Padres**: representa el indice de los individuos que se van a cruzar
  - **parejas**: las parejas ya formadas
  - **Pmut**: probabilidad de cada individuo de mutar
- **Variables de parada:**
  - **MAX_itera**: numero de maximas iteraciones
  - **itera**: iteración actual
  - **mejora:** controla que la población vaya mejorando
- **Variables de Satelites**
  - **NSatels**: numero de satelites (en este caso:500)
  - **NManagers**: numero de representantes

Algoritmo Genético

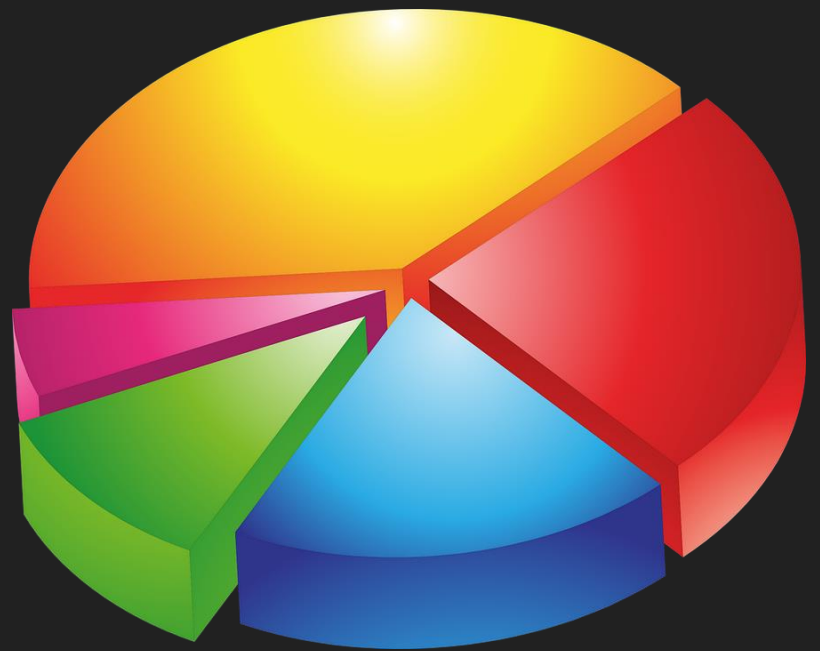
Composición

- Selección: Por Ruleta
- Cruce: Uniforme por Máscara (Modificado)
- Mutación: Inversión
- Reemplazo: Elitista (Modificado)

Selección: Por Ruleta

```
%%% Seleccion por Ruleta %%%

function [Padres] = Selection_Roulette(FitPob,k)
    Padres = zeros(1,k);
    PUnit = 1 / sum(FitPob);
    PrbAcum = cumsum(PUnit * FitPob);
    i = 1;
    while i <= k
        r = rand;
        j = find(r>=PrbAcum,1,'last’);
        c = ismember(j,Padres);
        if c == 0
            Padres(i) = j;
            i = i + 1;
        end
    end
end
```



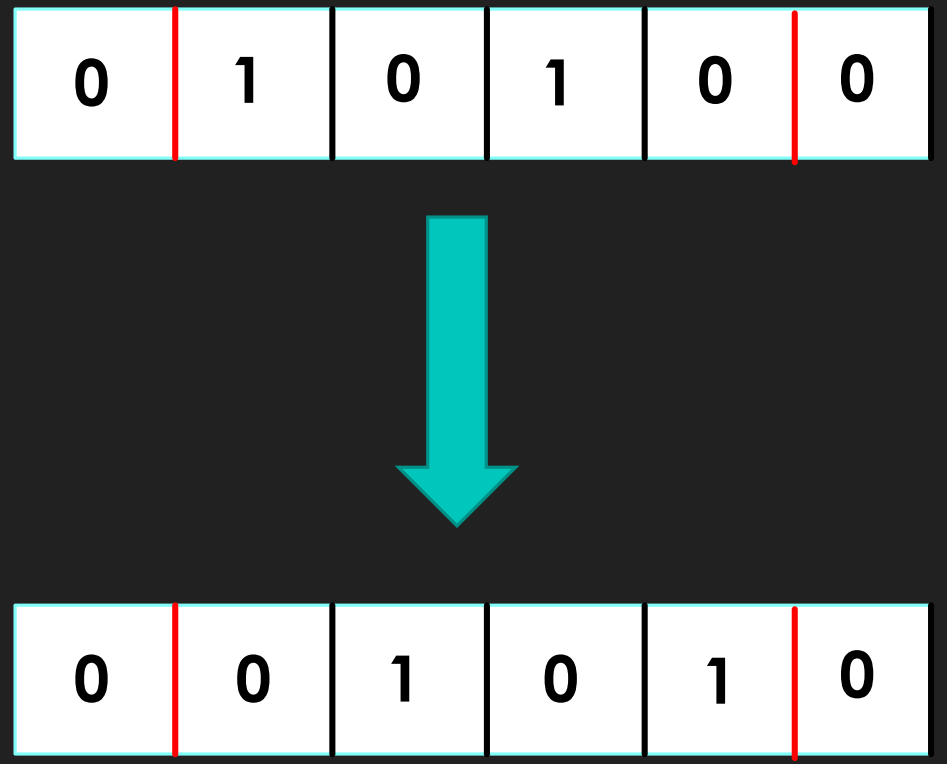
Cruce Uniforme por Máscara

```
function C = MaskCrossover(A,B,NManagers)

%We create the Mask
Mask = randi([0,1],1,size(A,2));
C = zeros(2,size(A,2));
C = logical(C);
%First Son: 0 to the first parent, 1 to parent
2
m1 = find(Mask ==false);%Positions where Mask 0
m2 = find(Mask == true);%Positions where Mask 1
C(1,m1) = A(m1);
C(1,m2) = B(m2);
%Second Son: 0 to the parent 2, 1
to the parent 1
m1 = find(Mask == true);
%Positions where Mask is 1.
m2 = find(Mask == false);
%Positions where Mask is 0.
C(2,m1) = A(m1);
C(2,m2) = B(m2);
%% Comprobamos que haya 40
representantes%
```

Mutación por Inversión

```
function mutado = inversion(original)
%original: vector que queremos mutar
    puntos = randperm(length(original));
    if puntos(1) > puntos(2)
        p1 = puntos(2);
        p2 = puntos(1);
    else
        p1 = puntos(1);
        p2 = puntos(2);
    end
    clear puntos;
    mutado = original;
    mutado(:, p1:p2 ) = flip(mutado(:, p1:p2));
end
```



Reemplazo Elitista

```
function [Pob_,Ev_] = ElitistReplace(Pob,NewPob,Ev,NewEv)

    tempPob = [Pob;NewPob];
    tempEv = [Ev,NewEv];
    [tempPob,ind] = unique(tempPob,'rows’);
    tempEv = tempEv(1,ind);
    tempPob = [tempPob,tempEv’];
    tempPob = sortrows(tempPob,length(tempPob),'descend’);
    Pob_ = tempPob(1:size(Pob,1),1:size(tempPob,2)-1);
    Ev_ = tempEv(1:size(Pob,1));

end
```

Algoritmo de Búsqueda Taboo

Parámetros

- **Variables de parada**
  - **stucked**: contador de generaciones que no mejoran
  - **MAX_itera**: numero de maximas iteraciones
- **Variables de Satélites**
  - **matPos**: matriz de posiciones
  - **distancias**: matriz de distancia euclidea entre los satelites
  - **NSatels**: numero de satelites (en este caso:500)
  - **NManagers**: numero de representantes(en este caso:40)
- **Variables internas del algoritmo**
  - **It**: iteracion actual
  - **current**: individuo actual
  - **list_suc**: lista de sucesores
  - **fitOldBest**: evaluación del mejor individuo
  - **fitBest**: evaluación del major actual
  - **TabuList**: Lista Taboo

Función sucesores

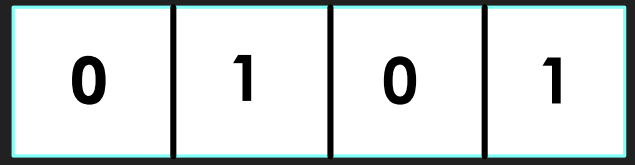
```
function [lista_sucesores, posChanged] = sucesores(original, distancias)

lista_sucesores = repmat(original, length(original), 1);
diagonal = lista_sucesores(logical(eye(length(original))));
lista_sucesores(logical(eye(length(original)))) = original(1);
lista_sucesores(:,1) = diagonal;
lista_sucesores = double(unique(lista_sucesores,'rows'));

lista_sucesores(:,end+1) = EvaluaPoblacion_Satels(lista_sucesores,
distancias);

[lista_sucesores , posChanged] = sortrows(lista_sucesores,
size(lista_sucesores,2));
lista_sucesores = logical(lista_sucesores(:,1:end-1));
end
```

Función sucesores



Función sucesores

0	1	0	1
0	1	0	1
0	1	0	1
0	1	0	1

Función sucesores

0

1

0

1

0	1	0	1
0	1	0	1
0	1	0	1
0	1	0	1

Función sucesores

0

1

0

1

0	1	0	1
0	0	0	1
0	1	0	1
0	1	0	0

Función sucesores

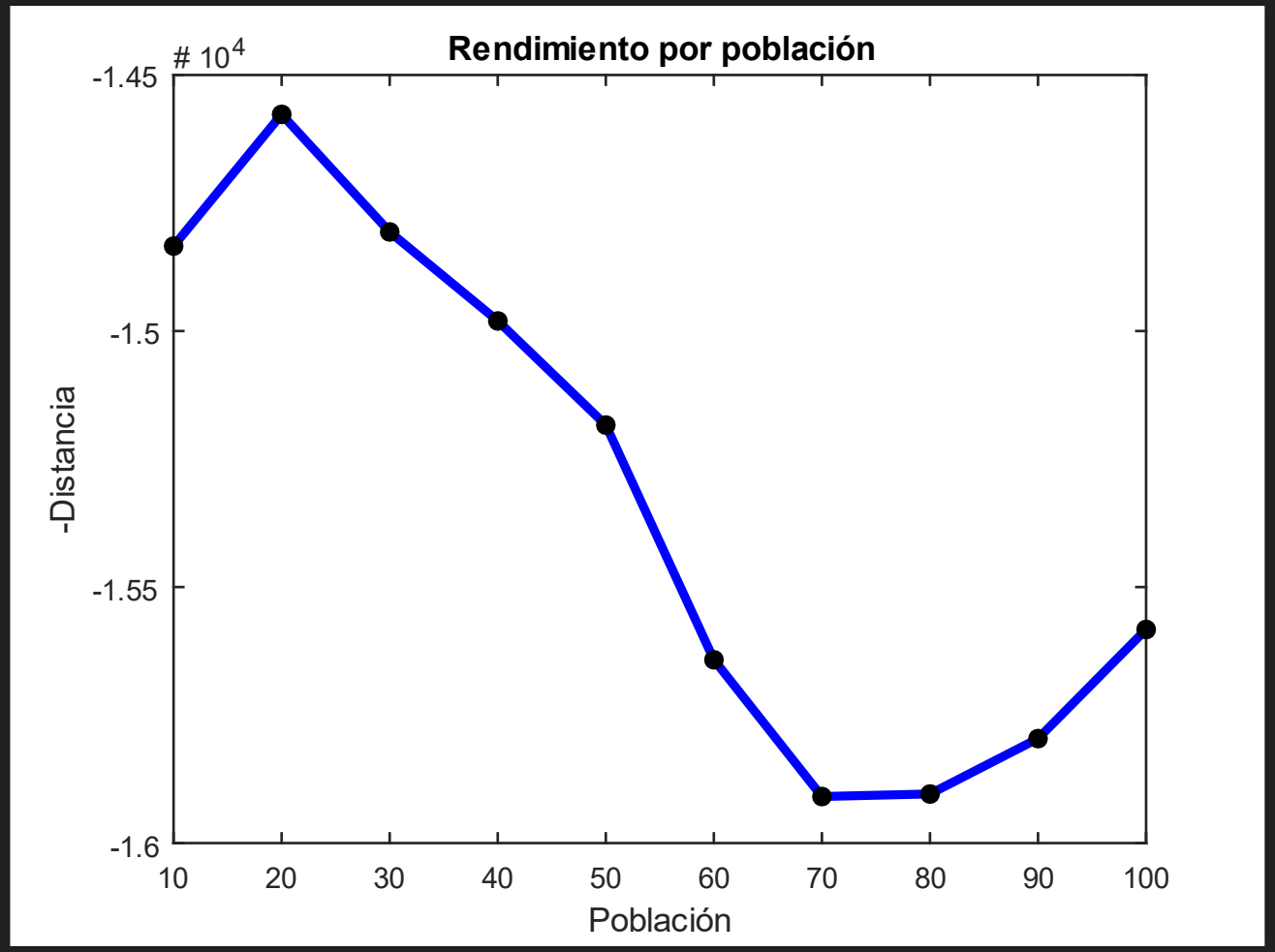
0	1	0	1
1	0	0	1
0	1	0	1
1	1	0	0

Función sucesores

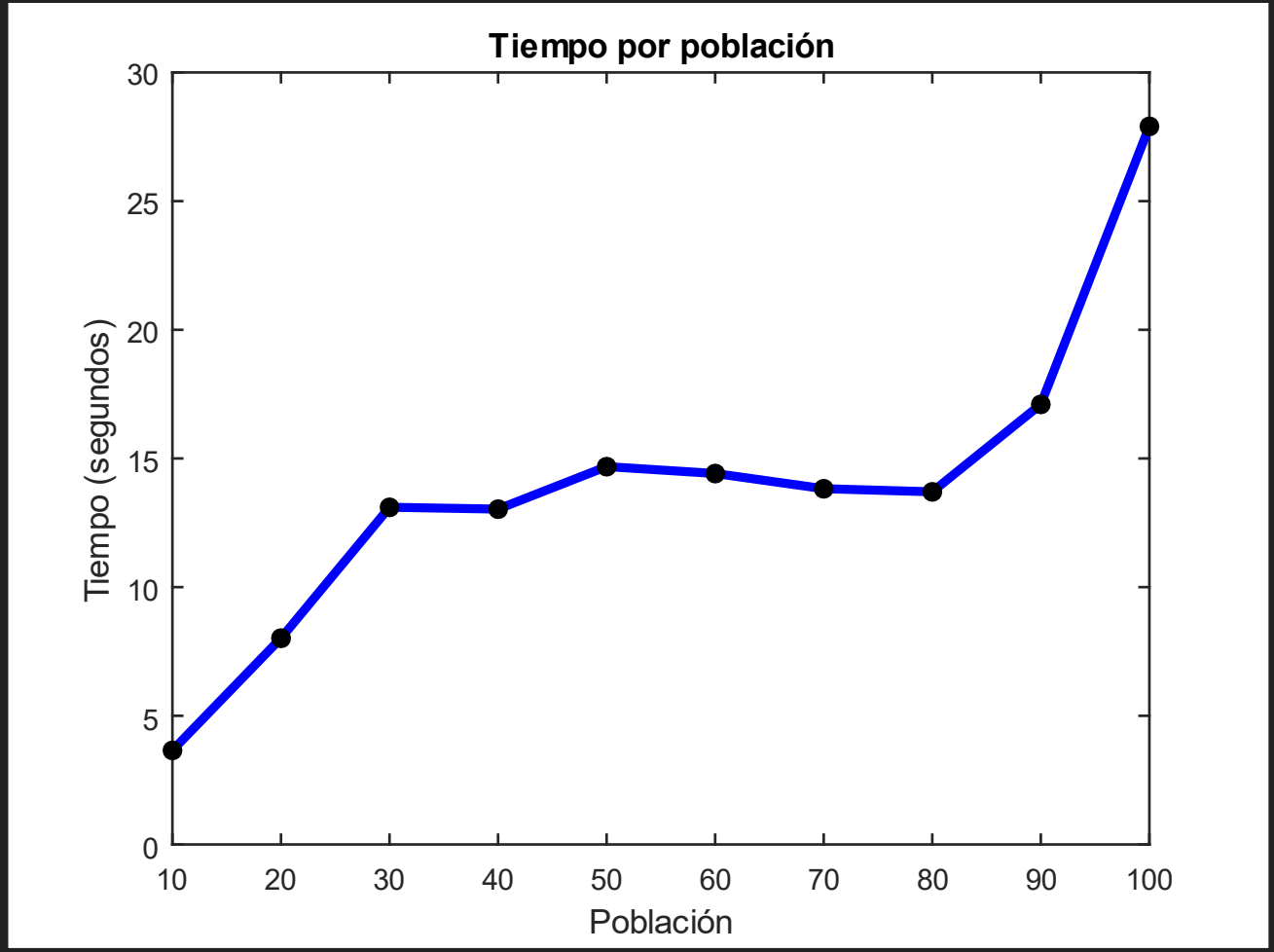
0	1	0	1
1	0	0	1

1	1	0	0

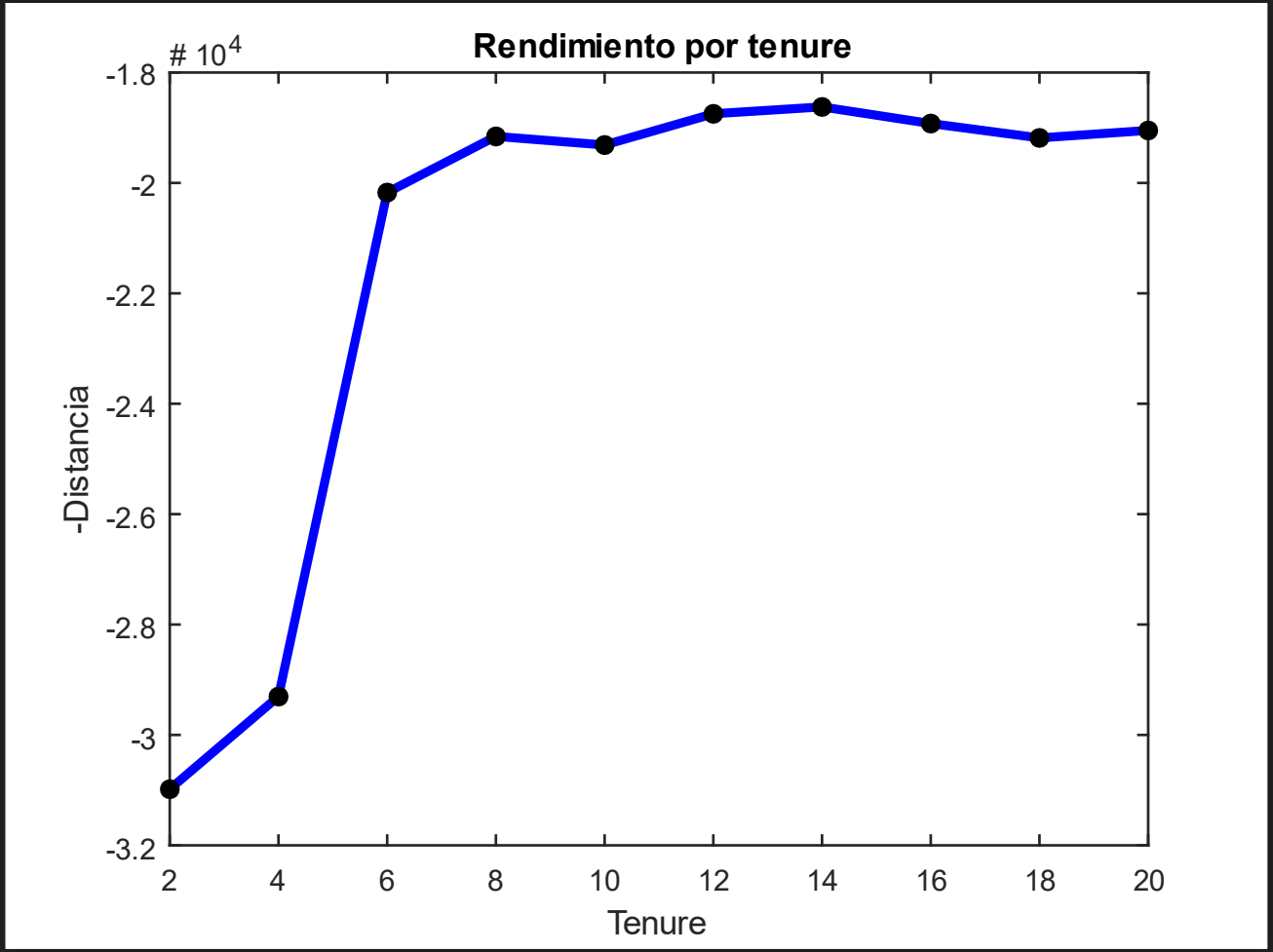
Comportamiento Algoritmo Genético



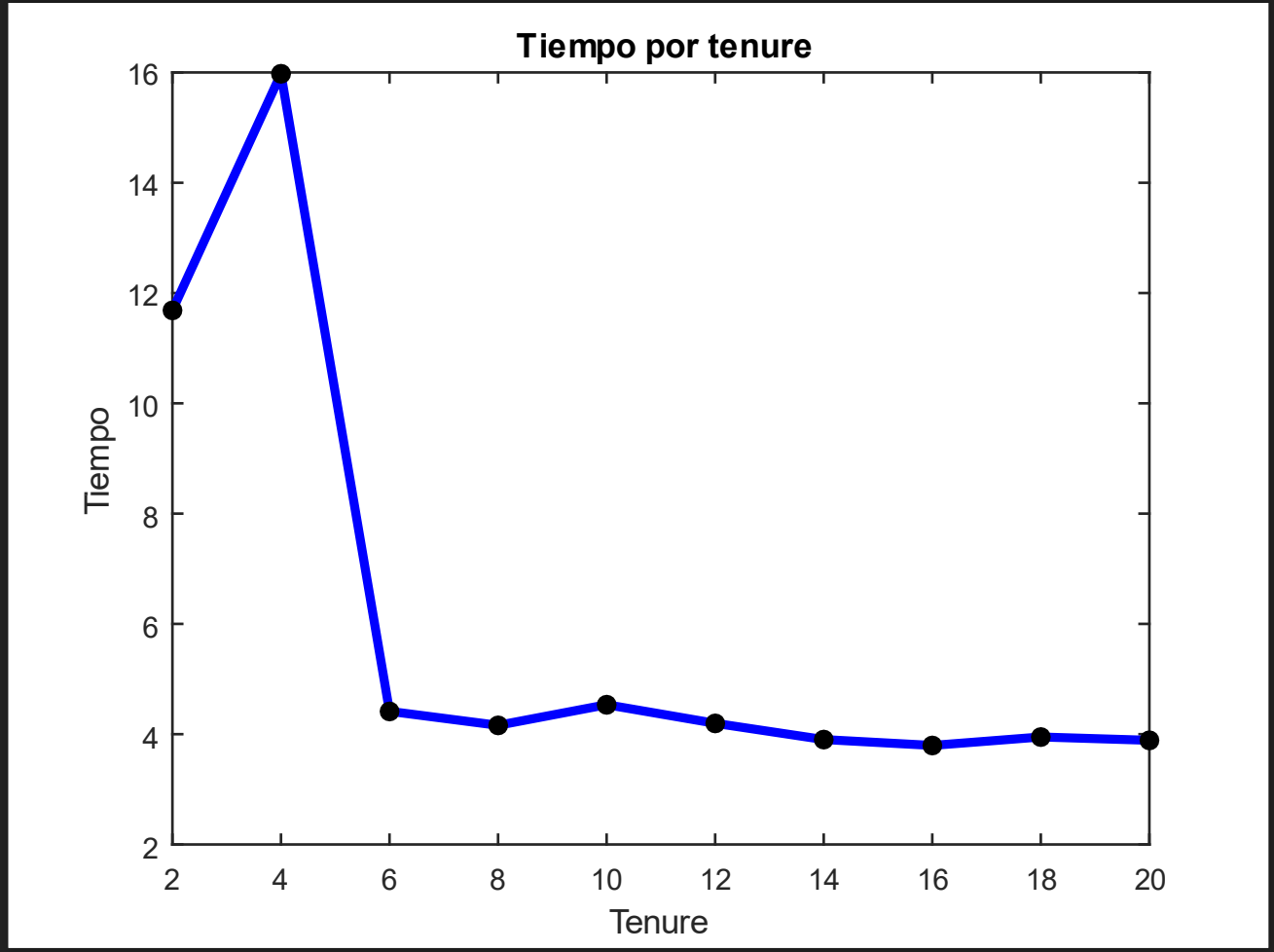
Comportamiento Algoritmo Genético



Comportamiento Taboo



Comportamiento Taboo



GitHub

github.com/rafaroman18/SatellitesIS

README.md

SatellitesIS

Final Practice of the subject Intelligent Systems

This is the task we have to implement on Matlab:

On a bidimensional *AxB* there N posible stations which we have to select **M** representant stations of the rest (N>M):S= {R1,R2,...,Rm}.

A station Ri of the subset S is representant of another no selected station when the euclid distance is less than than any other member Rj from S. The position of the N stations is static and is set at the beginning of the problem.

We pretend to find the representant station combination which minimize the global euclid distance of this problem.

The data structure we will use is a boolean array of 1x500 with 40 1's and 500 0's where each 1 describe the representant by the index of the array.

Example: `array = [1 1 0 1....] This means the station 1, 2 and 4 are representants in this case`

And to make the evaluation functions faster, we will create at the beginning of the program a 500x500 matrix where we will set

Gracias por vuestra atención
¿Alguna pregunta?