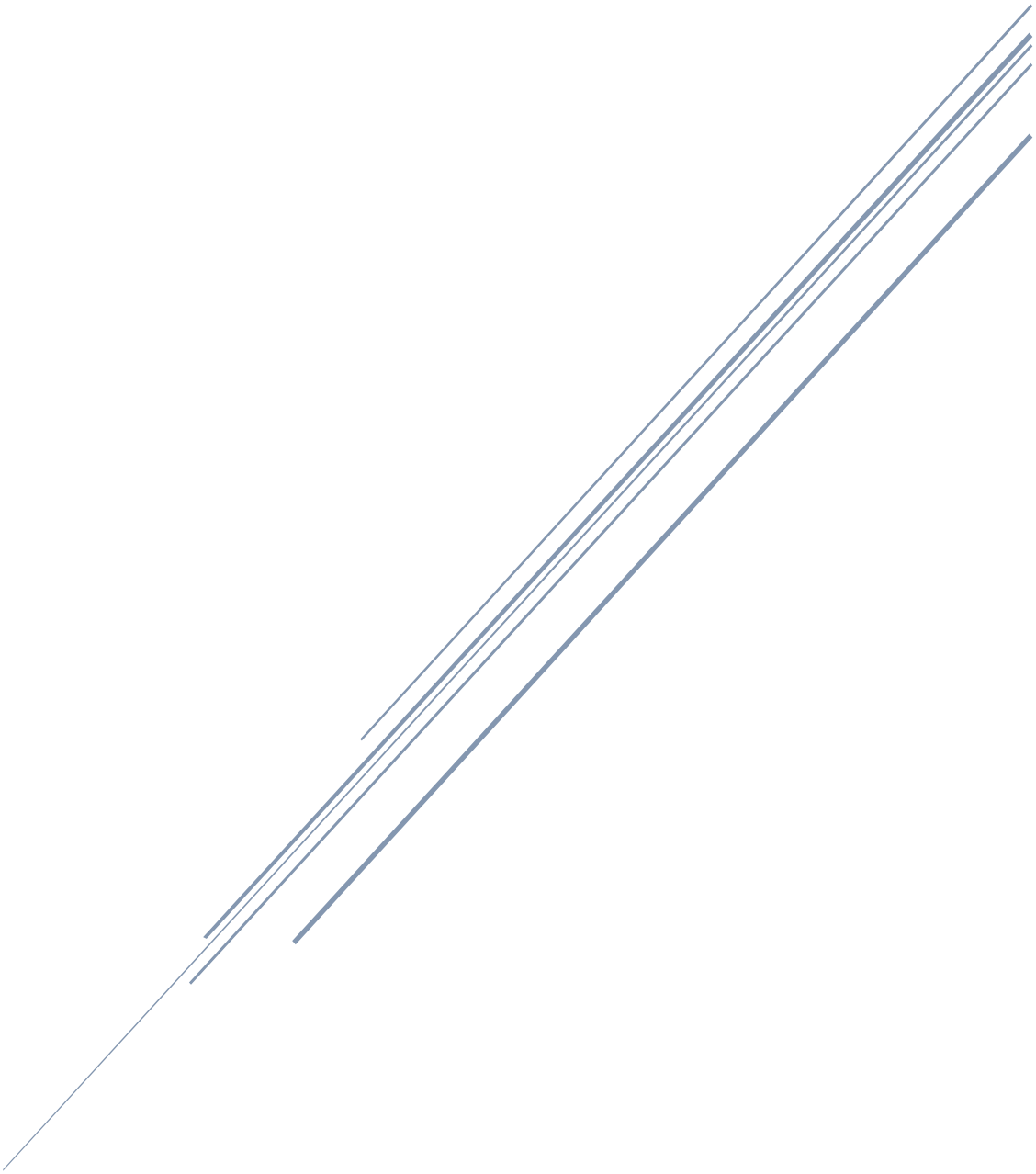
Práctica Final.

Diseño óptimo de un parque eólico con algoritmos metaheurísticos de optimización



Carmen Martínez Estévez – 09070605A
Alberto Martínez Ortega – 04638602P

El algoritmo evolutivo que nos ha tocado implementar es Harmony Search (HS). Hemos decidido aplicarlo sobre un conjunto binario de 400 unos y ceros, que posteriormente se convierte a una matriz de 20x20 para cumplir con los requisitos de la función objetivo.

En primer lugar, declaramos el tamaño del grid y el número de turbinas a 20. Así mismo, reservamos memoria para una matriz del tamaño de 30 fias por 400 columnas para los valores de la Harmony Memory, y un vector de 30 valores para almacenar el fitness entre iteraciones.

```
Kgr = 20;    % Tamaño del Grid
Nturb = 20; % Numero de Turbinas

HM = zeros(30, Kgr*Nturb);
fitness=zeros(1,30);
```

Después, procedemos a llenar de valores a la matriz HM. Para ello, creamos un vector auxiliar llamado padre de 20x20. Utilizamos el método randperm para establecer 20 unos en las 400 posiciones posibles. Posteriormente, calculamos el fitness de la matriz padre y lo guardamos en el vector fitness. Finalmente, hacemos uso del método reshape para reorganizar la matriz padre en un vector de 1x400 y así poder añadirlo a la Matriz HM.

```
for i=1:30
    padre=zeros(20);
    padre(randperm(Kgr^2,Nturb)) = 1;     % Gen Nturb in a grid of Kgr x Kgr
    [pwr_T2,gan_T2,cost_T2,obj_T2] = f_powerPlantsT_fast(vVec,padre);
    fitness(i)=pwr_T2;
    vector=reshape(padre,[1,400]);
    HM(i,:)=vector;
end
```

Decidimos realizar 300 iteraciones. En cada iteración realizamos lo siguiente:

El primer paso es realizar el **HMCR**. En este paso inicializamos la matriz de hijos del tamaño de 30 filas por 400 columnas. Con dos bucles for anidados vamos recorriendo todas las posibles posiciones de la matriz de hijos y rellenándolas con valores. Dichos valores surgen de lo siguiente: generamos aleatoriamente un número que nos sirva como probabilidad., Si este número es menor o igual que 0,97, el valor para ese hijo será el mismo que el que tenía su padre; lo sacaremos de la Harmony Memory. Por el contrario, si dicho número es mayor de 0.97, generaremos el valor con una probabilidad del 50% hacia 1 y del otro 50% hacia el 0.

```
for j=1:300
    %HMCR
    hijos=zeros(30,400);
    for k=1:30
        for l=1:400
            probabilidad=rand;
            if probabilidad<=0.97
                hijos(k,l)=HM(k,l);
            else
                valor_aleatorio=rand;
                if(valor_aleatorio)<=0.5
                    valor=0;
                else
                    valor=1;
                end
                hijos(k,l)=valor;
            end
        end
    end
```

El siguiente paso es ajustar el número de turbinas, para que no sobrepase las 20 por fila. Para ello recorremos la matriz de hijos, y con un contador obtenemos el número de 1s por fila. Después inicializamos un vector del tamaño del contador para guardar los índices de las posiciones en las que hay unos. Nuevamente, recorremos la matriz para ir cogiendo dichas posiciones y guardándolas en el vector indices.

Una vez tenemos las posiciones, calculamos la diferencia que hay entre los 20 unos que debe haber y los unos que realmente hay en la fila. Si esta diferencia es mayor que 0 y hasta que sea igual a 0, escogemos un valor aleatorio del vector indices, buscamos la posición en la matriz de hijos y sustituimos ese 1 por un 0.

```
%ajustamos 1s
for o=1:30
   contador=0;
   for u=1:400
       if hijos(o,u)==1
           contador=contador+1;
       end
   end

   indices=zeros(1,contador);
   indice=1;
   for u=1:400
       if hijos(o,u)==1
           indices(indice)=u;
           indice=indice+1;
       end
   end

   diferencia=contador-20;
   if diferencia>0

        while(diferencia~=0)
            numero = randsample(indices,1);
            if hijos(o,numero)==1
                hijos(o,numero)=0;
                diferencia=diferencia-1;
            end
        end
```

En caso contrario, si la diferencia fuese negativa, debido a que la probabilidad de encontrar un 0 es mucho mayor, calculamos un número al azar entre el 1 y el 400. Comprobamos, por si acaso, que efectivamente sea un 0 y añadimos un 1 en su lugar.

```
    elseif diferencia<0

        while(diferencia~=0)
            numero=(400-1).*rand(1)+1;
            numero=int16(numero);
            if hijos(o,numero)==0
                hijos(o,numero)=1;
                diferencia=diferencia+1;
            end
        end

    end

end
```

El siguiente paso es el PAR. Aquí calculamos una probabilidad PAR que va a ser generada aleatoriamente. Comprobamos si esta probabilidad es menor o igual que 0.03, en cuyo caso generamos otro valor aleatorio y comporbamos si este nuevo valor es menor o igual que 0.5 y si es así asignamos un 0 en la posición hijos(s,d). Si este segundo valor generado es mayor que 0.5 entonces asignamos un 1 en hijos(s,d). Si la porbabilidad_par es mayor que 0.03 no hacemos ningún cambio.

```
%PAR
for s=1:30

    for d=1:400

        probabilidad_par=rand;
        if(probabilidad_par<=0.03)
            valor_aleatorio_par=rand;
            if(valor_aleatorio_par)<=0.5
                valor=0;
            else
                valor=1;
            end
        hijos(s,d)=valor;
        end
    end
end
```

Después volvemos ajustar el número de turbinas con el mismo método que hemos explicado antes de PAR → 'ajustamos 1s'.

Reajustamos cada fila de la matriz hijos a un tamaño de 20x20. Después calculamos su fitness. Si este fitness es mayor que el que tenía su padre, sustituimos la fila de HM por la fila de los hijos, así como su fitness.

```
for y=1:30

    hijos_matriz=reshape(hijos(y,:),[20,20]);
    [pwr_T2_hijo,gan_T2,cost_T2,obj_T2] = f_powerPlantsT_fast(vVec,hijos_matriz);
    if fitness(y)<pwr_T2_hijo
        HM(y,:)=hijos(y,:);
        fitness(y)=pwr_T2_hijo;
    end
end
```

Los resultados obtenidos son los siguientes:

```
fitness =

   1.0e+07 *

  Columns 1 through 11

    5.2714    5.2682    5.2627    5.2651    5.2694    5.2718    5.2741    5.2668    5.2607    5.2864    5.2726

  Columns 12 through 22

    5.2743    5.2802    5.2733    5.2786    5.2663    5.2717    5.2757    5.2725    5.2709    5.2748    5.2695

  Columns 23 through 30

    5.2752    5.2803    5.2818    5.2660    5.2654    5.2808    5.2664    5.2777

El mejor fitness es: 5.286447e+07
```

El mejor fitness obtenido ha sido de 5.2865*10^7 unidades de potencia. Varía respecto a las anteriores ejecuciones por la aleatoriedad inicial de los padres y por el método ajustar unos. No obstante, siempre se ha encontrado un valor fitness similar al obtenido.

La ejecución completa dura una hora, con unos 13 segundos por iteración. Para su ejecución con los mismos valores propuestos por el enunciado no hace falta realizar ningún cambio en el código. Si se desea probar con más padres o con un grid de tamaño distinto a 20x20, deberán realizarse cambios en todos los bucles y asignaciones. Si se cambiase el número de ejecuciones, se deberá ir al final del programa para poder mostrar los resultados de forma correcta.