Entrega Práctica Genéticos TSP

Iñigo Biedma Ramos

1. el modelo

1.1. Cromosomas

A la hora de representar el problema del TSP existen dos formas básicas de representarlo: Representar la posicion de las ciudades, o representar la distancia entre ellas.

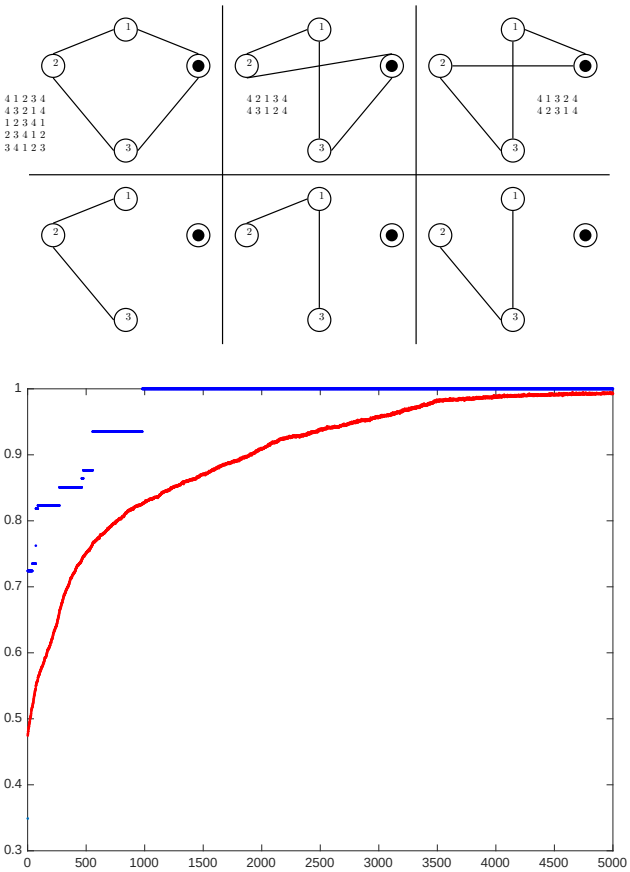

Figura 1: test

Prueba de codigo matlab:

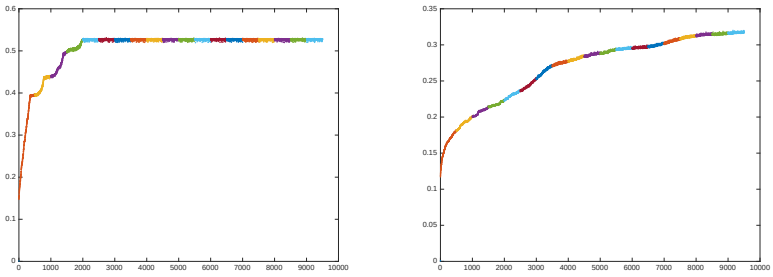

Figura 2: asd

```
case 0, % cruzamiento basado en orden
% elegidos i,j
% El elemento i esta en la posicion U2/V2(i)
%for k = 1:n
%    U2(i) = find(U==i);
%    V2(i) = find(V==i);
%end
Y = U(i:j);
Z = V(i:j);
% tomaremos valores desde la posicion j+1
V2 = circshift(V,[0 -j]);
U2 = circshift(U,[0 -j]);
% los valores que no esten ya colocados
V2 = MY_setdiff(V2,Y);
U2 = MY_setdiff(U2,Z);
% los ponemos
Y = [V2(n-j+1:end) Y V2(1:(n-j))];
Z = [U2(n-j+1:end) Z U2(1:(n-j))];
```

Tenemos n peticiones de uso de un laboratorio, cada uno con un tiempo de comienzo si y un tiempo de finalización ti para cada petición i. Asumimos que todos los tiempos de comienzo y final son diferentes. Dos peticiones entran en conflicto si se solapan (es decir si el tiempo de comienzo de una petición es anterior al tiempo de finalización de otra). Nuestro objetivo es seleccionar un mayor número de peticiones que no contengan solapamientos. (Por ejemplo si tenemos estas tres peticiones [0, 3], [2, 5], [4 7] entonces seleccionaremos la primera y la tercera). Diseña un algoritmo voraz que calcule la solución óptima.

Necesitamos pensar en la acción de selección del algoritmo voraz que elaboraremos. Como la elección de una petición depende de si se solapa o no con el resto, y no queremos revisar todas cada vez, las ordenaremos por tiempo de comienzo. La última tarea la vamos a tomar siempre, ya que no condiciona al resto. Después escogeremos el resto teniendo en cuenta que no se solapen, de la que mayor tiempo de comienzo a la de menor.