

INDUSTRIALES
ETSII | UPM


POLITÉCNICA

Practica 2: EAs

- Objetivo: Algoritmos Evolutivos

Herramientas:

- Librería para algoritmos evolutivos **libga100** (lenguaje C)

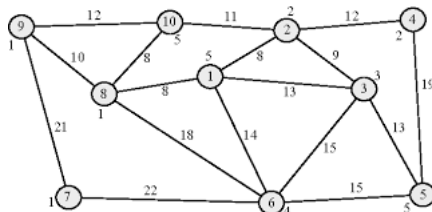




Parte 1: Eas para TSP

- TSP: Travelling Salesman Problem
- Encontrar el **ciclo** más breve que pase por todos los puntos de un conjunto

Sobre grafos: puntos = nodos
Arcos etiquetados con distancias

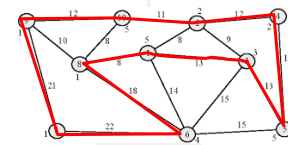


“Geográfico”: puntos = posiciones (x,y)
y distancias euclidianas



Parte 1: EAs para TSP

- TSP: Travelling Salesman Problem



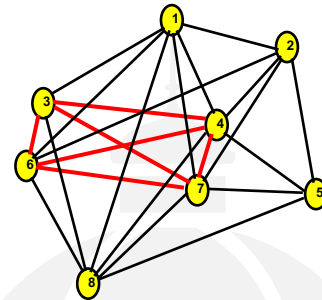
1	3	5	4	2	0	9	7	6	8
---	---	---	---	---	---	---	---	---	---

- Representación (codifica): **Permutaciones**
- Que función de fitness ? → # de nodos del subgrafo - penalidad
- Que operadores?
- Que parámetros?



Parte 2: EAs para MAXCLIQUE

- Clique: subconjunto de nodos **completamente conexo**



1	2	3	4	5	6	7	8
0	0	1	1	0	1	1	0

- Representación (codifica): **Cadena de bits**
- Que función de fitness ? → Longitud del ciclo
- Que operadores?
- Que parámetros?



LibGA100

- Librería de funciones en lenguaje C
- Fácil uso
- Fichero de configuración (.txt)
+
- Función de fitness (.c)



```
#include "ga.h"
main()
{
    GA_Info_Ptr ga_info;
    int i;

    /*--- Initialize the genetic algorithm ---*/
    ga_info = GA_config("GAconfig_ejemplo.txt", obj_fun);

    /*--- Run the GA ---*/
    GA_run(ga_info);

    printf("\nBest chrom: ");
    for(i=0;i<ga_info->chrom_len;i++)
        printf("%5.4f ",ga_info->best->gene[i]);

    printf("    (fitness: %g)\n\n",ga_info->best->fitness);
}
```



```
#include "ga.h"
main()
{
    GA_Info_Ptr ga_info;
    int i;

    /*--- Initialize the genetic algorithm ---*/
    ga_info = GA_config("GAconfig_ejemplo.txt", obj_fun);

    /*--- Run the GA ---*/
    GA_run(ga_info);

    printf("\nBest chrom: ");
    for(i=0;i<ga_info->chrom_len;i++)
        printf("%5.4f ",ga_info->best->gene[i]);

    printf("    (fitness: %g)\n\n",ga_info->best->fitness);
}
```

Estructura con info sobre el GA



```
#include "ga.h"
main()
{
    GA_Info_Ptr ga_info;
    int i;

    /*--- Initialize the genetic algorithm ---*/
    ga_info = GA_config("GAconfig_ejemplo.txt", obj_fun);

    /*--- Run the GA ---*/
    GA_run(ga_info);

    printf("\nBest chrom: ");
    for(i=0;i<ga_info->chrom_len;i++)
        printf("%5.4f ",ga_info->best->gene[i]);

    printf("    (fitness: %g)\n\n",ga_info->best->fitness);
}
```

Carga el fichero de configuración

Especifica la función para
el calculo de la fitness



```
#include "ga.h"
main()
{
    GA_Info_Ptr ga_info;
    int i;

    /*--- Initialize the genetic algorithm ---*/
    ga_info = GA_config("GAconfig_ejemplo.txt", obj_fun);

    /*--- Run the GA ---*/
    GA_run(ga_info);

    printf("\nBest chrom: ");
    for(i=0;i<ga_info->chrom_len;i++)
        printf("%5.4f ",ga_info->best->gene[i]);

    printf("    (fitness: %g)\n\n",ga_info->best->fitness);
}
```

Ejecuta el algoritmo



```
#include "ga.h"
main()
{
    GA_Info_Ptr ga_info;
    int i;

    /*--- Initialize the genetic algorithm ---*/
    ga_info = GA_config("GAconfig_ejemplo.txt", obj_fun);

    /*--- Run the GA ---*/
    GA_run(ga_info);

    printf("\nBest chrom: ");
    for(i=0;i<ga_info->chrom_len;i++)
        printf("%5.4f ",ga_info->best->gene[i]);

    printf("    (fitness: %g)\n\n",ga_info->best->fitness);
}
```

Uso de la struct **ga_info**
Para acceder a la información



```
typedef struct {
    /*--- Basic info ---*/
    int rand_seed; /* Seed for random number generator */
    int datatype; /* Data type flag */
    int pool_size; /* Pool size (IP_RANDOM) */
    int chrom_len; /* Chromosome size (IP_RANDOM) */
    int iter, max_iter; /* Number of iterations for ga */
    int minimize; /* Minimize EV_fun? */
    int converged; /* Has ga converged? */
    float x_rate; /* Crossover rate */
    float mu_rate; /* Mutation rate */
    Chrom_Ptr best; /* Best chromosome */

    ...
} *GA_Info_Ptr;
```




```
GA_Info_Ptr ga_info;  
  
for(i=0;i<ga_info->chrom_len;i++)  
  
ga_info->best->...
```



```
typedef struct {  
    Gene_Ptr    gene;        /* Encoding */  
    int         length;      /* Length of gene */  
    double      fitness;     /* Fitness value of chromosome */  
  
    ...  
} Chrom_Type, *Chrom_Ptr;  
  
X=ga_info->best->fitness;  
  
printf(...,ga_info->best->gene[i]);
```

gene: **array** del tipo
especificad (bits,
enteros, float...)



INDUSTRIALES
ETSII | UPM

LibGA100

POLITÉCNICA

= comentario

Fichero de configuración


```

#Chromosome type
datatype bit
# datatype int
# datatype int_perm
# datatype real
#-----
# Chromosome length
# Usage: chrom_len length
#-----
chrom_len 100
#-----
pool_size 100
...
          
```

En esta práctica será "bit" o "int_perm"

Depende de la instancia !!!

¡Probar varias combinaciones de parámetros !



INDUSTRIALES
ETSII | UPM

LibGA100

POLITÉCNICA

Fichero de configuración

```

# Mutation Rate
#
# Usage: mu_rate number
#-----
mu_rate 0.9
#-----
# Crossover Rate
#
# Usage: x_rate number
#-----
x_rate 0.7
          
```

Probabilidad de mutación

¡Probar varias combinaciones de parámetros !

Probabilidad de cruce

¡Probar varias combinaciones de parámetros !



Ejemplo de función de fitness

```
/*-----  
| obj_fun() - user specified objective function  
-----*/  
int obj_fun(Chrom_Ptr chrom)  
{  
    int i;  
    double val = 0.0;  
  
    for(i = 0; i < chrom->length; i++)  
    {  
        val += chrom->gene[i];  
    }  
  
    chrom->fitness = val;  
  
    return 0; // no usado  
}
```

```
ga_info = GA_config("GAconfig_ejemplo.txt",  
                    obj_fun);
```

En este ejemplo sencillo, la fitness es el numero de "1" del chrom.

- chrom: estructura
- length: entero
- gene: vector
- fitness: real



Vuestra función /1 (TSP)

```
int obj_fun(Chrom_Ptr chrom)  
{  
    float d_tot=0;  
  
    // dtot += distancia entre ciudad 1 y ciudad n  
    // bucle: de i = 1 a n-1  
    //      d_tot += distancia entre ciudad i y ciudad i+1  
  
    chrom->fitness = d_tot;  
  
    return 0;  
}
```

- chrom: estructura
- length: entero
 - gene: vector
 - fitness: real

¡OJO!

Antes tenéis que cargar el fichero con las distancia en una matriz n x n

	1	2	...	n
1	0	46	...	56
2		0	...	5
...			0	61
n				0



Vuestra función /2 (MAXCLIQUE)

Esto es un ejemplo: podéis probar otras

```
int obj_fun(Chrom_Ptr chrom)
{
    float A, PENALTY=0;

    // A = número de 1 (tamaño del subconjunto)
    // Por todas las parejas de nodos del subconjunto:
    //     si NO hay arista entre ellos, PENALTY++;

    chrom->fitness = A -  $\alpha$ *PENALTY;

    return 0;
}
```

chrom: estructura

- length: entero
- gene: vector
- fitness: real

¡OJO!
Antes tenéis que cargar el fichero
con el grafo en una matriz nxn
("matriz de adyacencia")

	1	2	...	n
1	0	0	...	1
2		0	...	0
...				1
n				0

α es un parámetro que
tenéis que ajustar



Practica 2: EAs

Se proporciona:

- Librería libGA100 (x linux, funciona tb. en VC++)
- Código de ejemplo + funciones para TSP y MAXCLIQUE
- **Estancias TSP (euclidianas!)**
 - Test50 – 50 ciudades → Optimum tour = ???
 - kroA100 – 100 ciudades → Optimum tour = 21282
 - lin318 – 318 ciudades → Optimum tour = 42029
 - lin105 – 105 ciudades → Optimum tour = 14379
- **Estancias MAXCLIQUE**
 - C125.9 (125 nodos) → max clique: 34
 - C250.9 (250 nodos) → max clique: 44
 - Keller4 (171 nodos) → max clique: 11
 - Toy8 (8 nodos) → max clique: 5



Para compilar la librería:

En la carpeta "Practica EAs MUAR 2019"

```
$cd libga
```

```
$make
```

Para compilar vuestro programa: (todo en la misma línea)

```
$gcc <nombre.c> -o <nombre> -L./libga -I./libga  
-lGA -lm
```

O bien, modificar el *makefile* proporcionado