

PRACTICA 5

Objetivo General

Desarrollar un par de versiones de un Algoritmo Genético para la simulación de redes porosas de conectividad 4 mediante el Modelo Dual de Sitios y Enlaces.

Objetivos Particulares:

- Que el alumno comprenda el funcionamiento de una simulación discreta, en el que se modela un fenómeno natural que se presenta en la mayoría de los materiales que nos rodean.
- Comprender con prácticas paulatinas los fenómenos capilares que se presentan en la naturaleza, mediante la generación de redes porosas aleatorias con distribución normal o gaussiana.
- Constatar lo aprendido sobre apuntadores para emular entidades que necesiten de un tipo de dato abstracto elaborado.
- Ratificar la comprensión sobre el tema de la generación de algoritmos usando la bifurcación de procesos, usando la arquitectura de maestro y esclavo, o de servidor – cliente, de pende de las necesidades de este algoritmo.

Documentos para entregar

- La práctica puede realizarse de forma individual o en equipo sin incorporar a nuevos integrantes a los equipos ya formados.
- Programas en C que se hayan desarrollado para lograr los objetivos establecidos en la práctica (50 % de la calificación).
- Informe de cómo se abordaron cada uno de los problemas planteados (50 % de la calificación). El informe debe contener:
 - Caratula, con nombre de la materia, nombre propio (empezando por apellidos), matrícula, profesor y fecha de entrega.
 - Introducción de lo que se realizó a grandes rasgos en la práctica.
 - Una sección de Desarrollo en el cual se expliquen usando parcialidades de código y algún otro recurso (pantallazos, esquemas, etc) como se trataron cada uno de los objetivos particulares que conforman la práctica.
 - Conclusiones generadas de la Práctica (**MUY IMPORTANTE**).
 - EL INFORME NO ES SOLO PONER CÓDIGO, DEBEN ESFORZARSE POR PONER ALGO MÁS QUE ESO.

Plazo de entrega

La hora y fecha límite para entregarla será el martes 25 de marzo del 2025 antes de que inicie la sesión de laboratorio y todo será enviado al correo de siempre.

Nota. No se recibirá ninguna práctica fuera de ese horario, sin ninguna excepción.

Especificaciones del programa para entregar:**ConstructorRedes2D_C4_Genetico_Secuencial.c**

En esta parte de la práctica se pretende realizar un algoritmo genético secuencial para la generación de redes porosas, para ello se les proporcionará una plantilla con la cual podrán generar una red 2D con distribución Gaussiana o Normal. Vamos a explicar primero en qué consiste la estructura principal de esta plantilla, la cual se muestra en el Código 1.

```
typedef struct Nodo_Red {  
    double r_Sitio;  
    double r_EIzq;  
    double r_EArr;  
    int errT1;  
    //Para Sitios, valores: 0 Pequeños, 1 Medianos, 2 Grandes  
    int tipo_S;  
    //Para Enlaces Izquierdos, valores: 0 Pequeños, 1 Medianos, 2 Grandes  
    int tipo_EIzq;  
    //Para Enlaces Arriba, valores: 0 Pequeños, 1 Medianos, 2 Grandes  
    int tipo_EArr;  
} NODO_BSM;
```

Código 1. Estructura principal para el modelado de la red porosa.

Como se puede observar en el Código 1, se tienen tres variables de tipo double para modelar cada uno de los nodos propuestos en el Modelo Dual de Sitios y Enlaces, los Sitios son esferas que a su vez están rodeadas de un conjunto de cilindros que modelan a los enlaces, físicamente tendríamos un nodo tal y como se muestra en la Figura 1.



Figura 1. Nodo desde una vista 3D para tener un mejor entendimiento de lo que se va a modelar en la práctica

De la Figura 1 podemos observar que al Sitio solo lo rodean 2 enlaces, sin embargo, vamos a manejar una conectividad 4, esto es que cada Sitio debe estar rodeado de 4 enlaces, los dos que mostramos en la Figura 1, y dos más que van a ser compartidos por sus vecinos derechos e inferiores, logrando de esta manera una red 2D que presentamos en la Figura 2.

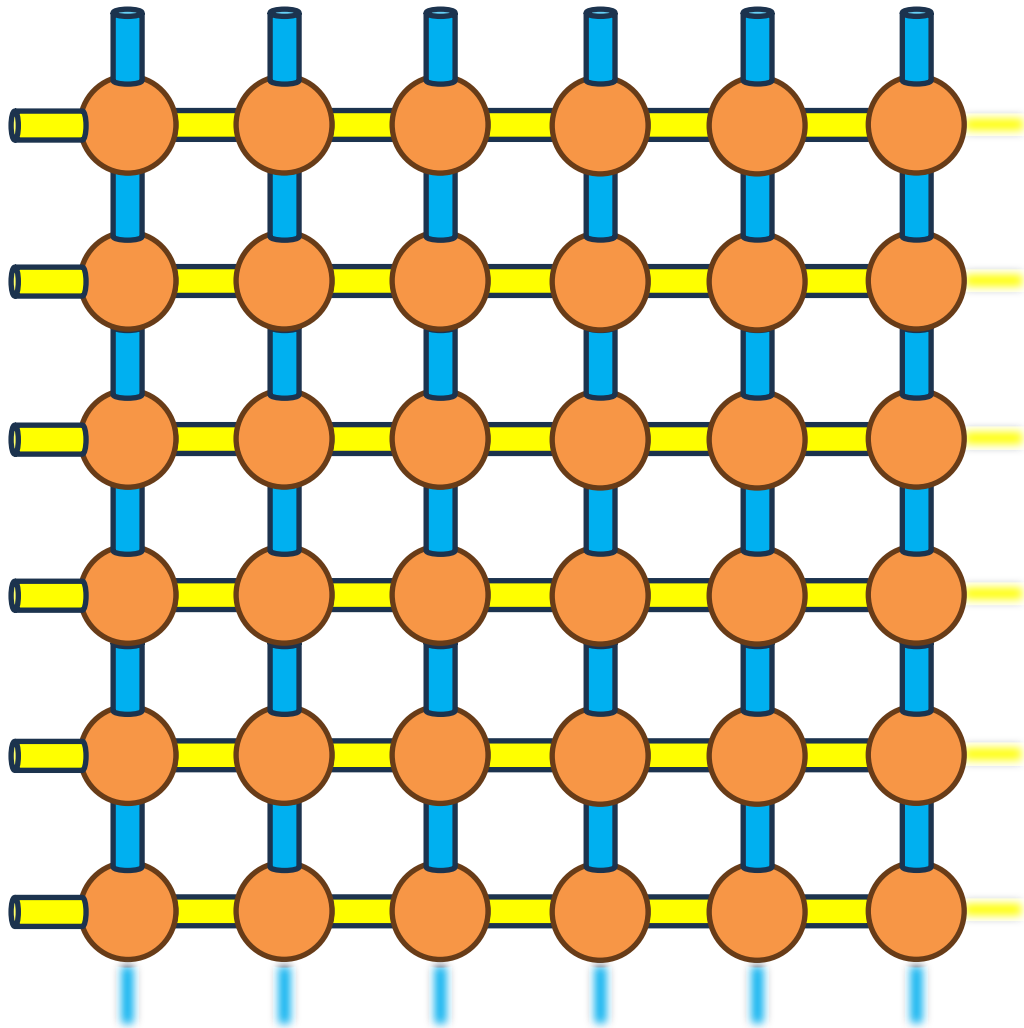


Figura 2. Red de conectividad 4, compartiendo los enlaces con sus demás vecinos.

De la Figura 2 podemos visualizar que todos los Sitios de color Naranja tienen 4 enlaces que lo rodean, resaltando que en la última columna y la última fila, hemos difuminado los enlaces izquierda y abajo con su color respectivo (amarillo y azul), para simular que los tienen. Sin embargo, sabemos que estos enlaces son compartidos, los enlaces izquierdos de la primera columna son los enlaces derechos para la última columna, mientras que los enlaces arriba de la primera fila son los enlaces debajo de la última fila.

Esto se logra implementando el operador aritmético modulo sobre el contador de filas, normalmente i , y sobre el contador de las columnas, normalmente j . Esto se modela como una matriz 2D, por lo tanto, si queremos acceder a los diferentes enlaces podríamos utilizar el Código 2, suponiendo que L es el tamaño fijado por el usuario.

```
//Para acceder al enlace izquierdo
RED2D[i][j].r_EIzq
//Para acceder al enlace derecho
RED2D[i][(j+1) % L].r_Eizq
//Para acceder al enlace arriba
RED2D[i][j].r_Earr
//Para acceder al enlace abajo
RED2D[(i+1) % L][j].r_Earr
```

Código 2. Modo para acceder a un nodo de la matriz 2D de la red porosa.

Como podemos observar en el Código 2, para acceder al enlace derecho y abajo utilizamos el modulo L , y de esa manera accedemos a los enlaces que no son propios del Sitio, si no de su vecino próximo. Solo tener cuidado en la generación de sus poblaciones ya que como podrán observar en la plantilla esto es una matriz 3D, definida como:

```
NODO_BSM ***Poblacion = malloc(numCromosomas * sizeof(NODO_BSM **));
```

Y para ello el acceso a cada elemento se hará como lo muestra el Código 3:

```
//Para acceder al enlace izquierdo
Poblacion[k][i][j].r_EIzq
//Para acceder al enlace izquierdo
Poblacion[k][i][(j+1) % L].r_EIzq
//Para acceder al enlace izquierdo
Poblacion[k][i][j].r_Arr
//Para acceder al enlace izquierdo
Poblacion[k][(i+1) % L][j].r_Arr
```

Código 3. Modo para acceder a un nodo de una matriz 3D de la red porosa para una población del algoritmo genético.

Teniendo esto aclarado, en el Código 1. tenemos un atributo para contabilizar los Errores de Tipo 1, Mayagoitia y Kornhauser en su trabajo seminal titulado Capillary Processes in Porous Networks: 1. Models of Porous Structures; Principles and Applications of Pore Structural Characterization establecieron el siguiente Principio de Construcción (PC):

El radio de un Sitio debe ser más grande o igual que el radio de cada uno de los Enlaces que lo rodea.

Para la simulación de redes porosas, se generan un conjunto de números aleatorios con una distribución Gaussiana, los cuales representan los radios de Sitios y Enlaces, lo cual viene incorporado en sus respectivas plantillas. Con esta distribución se presenta un elemento que dificulta la construcción de las redes porosas llamado Traslape. El traslape es el punto interacción entre las dos campanas de Gauss, entre más traslape la construcción resulta mucho más difícil. Estos elementos también son parte de la plantilla con la cual contarán para el desarrollo de esta práctica y podrán profundizar más cuando entiendan este apartado llamado traslape.

Es por ello por lo que cuando se tengan traslapes cercanos a 1, la construcción aleatoria tendrá presente los Errores Elementales o de Tipo 1, que serán acumulados en la variable `errT1`. Estos errores se presentan cuando el radio de los Sitios es más pequeño que alguno de los radios de los Enlaces que lo rodean, tal y como lo podemos observar en la Figura 3

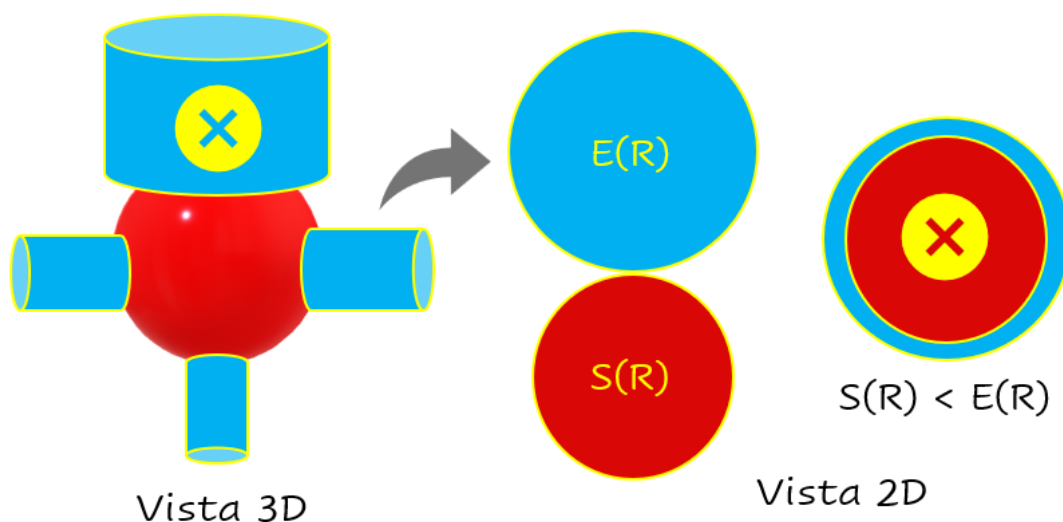


Figura 3. Escenario pictórico en el que se presenta un error elemental o tipo 1.

En la Figura 3 se tiene representado la violación al principio de construcción con el enlace de arriba del nodo mostrado en la imagen, el cual claramente se puede ver que es mayor al radio del Sitio.

Esto será lo más interesante de la práctica ya que el Algoritmo Genético tendrá que reducir los errores iniciales a 0 y ese es el estado para que este converja, esto quiere decir, que el reto será que el Algoritmo Genético termine su funcionamiento cuando un contador global de errores en la red llegue a cero. A principio de cuentas parece algo sencillo, pero no lo es, ya que en la ejecución del Algoritmo Genético no se deben disminuir drásticamente (puede variar un poco) el número de Sitios y Enlaces para una clasificación de pequeños, medianos y grandes, de allí el verdadero reto de esta práctica y más aún el despertar de su ingenio para resolver este problema.

De aquí que finalmente en el Código 1, tengamos los atributos `tipo_S`, `tipo_EIzq` y `tipo_EArr`, ya que cada uno de los nodos tendrá una etiqueta para cada elemento, 0 para Sitios y Enlaces pequeños, 1 para Sitios y Enlaces medianos y 2 para Sitios y Enlaces grandes, los intervalos para que queden definidos estos valores se encuentran en los procedimientos:

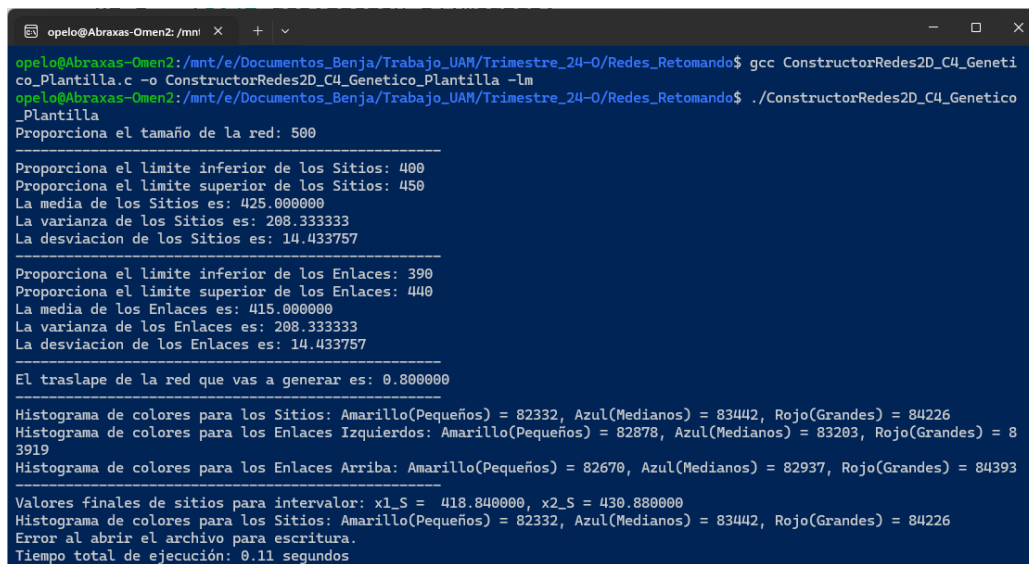
```
void inicializarPoblacion(NODO_BSM ***, int , int , int , int , int, int);
void clasificaTamano(int , NODO_BSM **);
```

Y el fragmento que se utiliza para establecer los intervalos sale de un razonamiento estadístico que podemos encontrar en el Código 4.

```
//Modulo para calcular los intervalos de división de los Sitios
x1_S = (-0.44*desviacion_S)+mediaS;
x2_S = (0.42*desviacion_S)+mediaS;
//Modulo para calcular los intervalos de división de los Enlaces Izquierdos
x1_EIzq = (-0.44*desviacion_E)+mediaE;
x2_EIzq = (0.42*desviacion_E)+mediaE;
//Modulo para calcular los intervalos de división de los Enlaces Arriba
x1_EArr = (-0.44*desviacion_E)+mediaE;
x2_EArr = (0.42*desviacion_E)+mediaE;
```

Código 4. Fragmento de código para establecer los intervalos de los Sitios y Enlaces.

Al ejecutar la plantilla tendrán una salida similar a la que se muestra en la Figura 4.



```
opelo@Abraxas-Omen2: /mnt/ x + v
opelo@Abraxas-Omen2:/mnt/e/Documentos_Benja/Trabajo_UAM/Trimestre_24-0/Redes_Retomando$ gcc ConstructorRedes2D_C4_Geneti
co_Plantilla.c -o ConstructorRedes2D_C4_Genetico_Plantilla -lm
opelo@Abraxas-Omen2:/mnt/e/Documentos_Benja/Trabajo_UAM/Trimestre_24-0/Redes_Retomando$ ./ConstructorRedes2D_C4_Genetico
_Plantilla
Proporciona el tamaño de la red: 500

Proporciona el limite inferior de los Sitios: 400
Proporciona el limite superior de los Sitios: 450
La media de los Sitios es: 425.000000
La varianza de los Sitios es: 208.333333
La desviacion de los Sitios es: 14.433757

Proporciona el limite inferior de los Enlaces: 390
Proporciona el limite superior de los Enlaces: 440
La media de los Enlaces es: 415.000000
La varianza de los Enlaces es: 208.333333
La desviacion de los Enlaces es: 14.433757

El traslape de la red que vas a generar es: 0.800000

Histograma de colores para los Sitios: Amarillo(Pequeños) = 82332, Azul(Medianos) = 83442, Rojo(Grandes) = 84226
Histograma de colores para los Enlaces Izquierdos: Amarillo(Pequeños) = 82878, Azul(Medianos) = 83203, Rojo(Grandes) = 8
3919
Histograma de colores para los Enlaces Arriba: Amarillo(Pequeños) = 82670, Azul(Medianos) = 82937, Rojo(Grandes) = 84393

Valores finales de sitios para intervalo: x1_S = 418.840000, x2_S = 430.880000
Histograma de colores para los Sitios: Amarillo(Pequeños) = 82332, Azul(Medianos) = 83442, Rojo(Grandes) = 84226
Error al abrir el archivo para escritura.
Tiempo total de ejecución: 0.11 segundos
```

Figura 4. Compilación y ejecución de la plantilla inicial para la implementación del Algoritmo Genético.

Como podemos observar en la Figura 4, se solicitará al usuario el límite inferior de los sitios y los enlaces, automáticamente mostrará la media, la varianza y desviación estándar de ambos. Después mostrara el traslape de la red que se pretende construir, que para este ejemplo es de 0.8, una red ideal para empezar, así que prueben con esos valores.

Parte de la salida de la Figura 4, muestra tres histogramas, que no son otra cosa que un conjunto de tres arreglos de tipo int, con tres casillas para depositar el conteo de los sitios y enlaces iniciales. El Histograma de colores para los Sitios tiene: Amarillo (Pequeños) = 82,332, Azul (Medianos) = 83,442, Rojo (Grandes) = 84,226, que si se observa es un número casi equitativo, que es lo que se pretende buscar. Para el caso del Histograma de los Enlaces Izquierdos se tiene: Amarillo (Pequeños) = 82,878, Azul (Medianos) = 83,203, Rojo (Grandes) = 83,919 y para el caso del Histograma de colores para los Enlaces Arriba se tiene: Amarillo (Pequeños) = 82,670, Azul (Medianos) = 82,937, Rojo (Grandes) = 84,393, respetando el número casi igual para cada uno de ellos.

Esto es muy importante ya que como se menciono anteriormente, el Algoritmo Genético debe respetar la misma distribución con algunas variaciones, pero muy mínimas, por ejemplo, para los sitios es válido que cambie a: Amarillo (Pequeños) = 82,297, Azul (Medianos) = 83,352, Rojo (Grandes) = 84,351, la variación no afecta la distribución y se tienen 250,000 sitios en total para una red de tamaño 500, lo mismo aplicaría para los Enlaces Izquierdos y Arriba.

Al finalizar la plantilla se tendrá un procedimiento llamado:

```
void exportarRedConColores(NODO_BSM **, int , const char *);
```

Que permitirá generar un archivo Excel para guardar la red construida con el patrón de colores que mencionamos en los párrafos anteriores, lo cual es la salida esperada de esta práctica, lo cual se muestra en la Figura 5.

	A	B	C	D
1	x	y	r_Sitio	color
2		0	0 387.370806	yellow
3		0	1 384.023177	yellow
4		0	2 524.220577	red
5		0	3 425.832288	blue
6		0	4 424.266643	blue
7		0	5 386.758243	yellow
8		0	6 386.948163	yellow
9		0	7 508.163774	red
10		0	8 427.953736	blue
11		0	9 516.036147	red
12		0	10 384.551713	yellow
13		0	11 383.513271	yellow
14		0	12 426.51295	blue
15		0	13 454.795046	red
16		0	14 474.860272	red
17		0	15 386.678944	yellow
18		0	16 559.162965	red
19		0	17 422.613347	blue
20		0	18 415.250329	blue
21		0	19 483.302597	red
22		0	20 428.554796	blue
23		0	21 429.378531	blue
24		0	22 385.699724	yellow
25		0	23 383.3866	yellow
26		0	24 386.938451	yellow
27		0	25 427.559969	blue
28		0	26 428.185686	blue
29		0	27 427.306238	blue
30		0	28 387.308555	yellow
31		0	29 523.883953	red
32		0	30 428.853352	blue
33		0	31 430.485945	blue
34		0	32 485.997815	red
35		0	33 461.550876	red
36		0	34 428.033014	blue
37		0	35 428.007749	blue

< > red_colores +

Listo Accesibilidad: No disponible

Figura 5. Salida de la plantilla inicial para construir el Algoritmo Genético.

La idea es continuar con sus propias implementaciones, logrando incorporar los siguientes procedimientos y funciones, más algunas adicionales que consideren pertinentes, los cuales muestro en el Código 5.

```
int evaluarFitness(NODO_BSM **RED_2D, int L);
void cruzarCromosomas(int tamPoblacion, int L, NODO_BSM ***Poblacion, double mediaS,
                     double mediaE, double desviacion);
void mutarCromosoma(int numCromosomas, int L, NODO_BSM ***POBLACION);
void seleccionRuleta(int tamPoblacion, int L, NODO_BSM ***Poblacion);
```

Código 5. Prototipos básicos para generar el funcionamiento del Algoritmo Genético.

La función `evaluarFitness()` permite calcular los errores de cada nodo de la red, el procedimiento `cruzarCromosomas()` permite recombinar los elementos de cada uno de los miembros de las poblaciones, teniendo en cuenta que en la inicialización se replica la población inicial. El procedimiento `mutarCromosoma()` de la misma forma es un operador que permite recombinar los genes de cada cromosoma de la población. Finalmente, en el procedimiento `seleccionRuleta()`, una vez que la cruce y la mutación termina, se debe aplicar un método para seleccionar cuáles son los individuos más aptos para pasar a la siguiente generación, se recomienda que se implemente el método de Ruleta.

ConstructorRedes2D_C4_Genetico_Procesos.c

Básicamente es tomar como base el `ConstructorRedes2D_C4_Genetico_Secuencial.c` para proponer la primera versión concurrente usando el esquema de procesos con la nativa `fork()`, sincronización con el `wait()` y el `exit()` y compartir datos entre procesos como se ha desarrollado en la Semana 4 y 5, de no hacerlo de esta forma, la práctica no será tomada en cuenta.