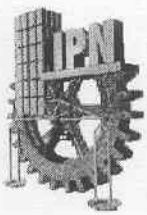


Instituto Politécnico Nacional

Centro de Investigación en Computación

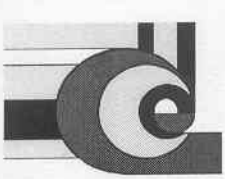
Maestría en Ciencias de la Computación

"Una herramienta para el estudio de sistemas representados por redes"

TESIS QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS PRESENTA EL
ING. RICARDO MENCHACA MÉNDEZ

Asesor: **Dr. Jesús Figueroa Nazuno**

Centro de Investigación en Computación
México D.F.
Febrero del 2005



Contenido

1. Motivación

Muchos de los sistemas que nos rodean son muy complicados, por ejemplo, el cerebro, la célula, el sistema inmunológico, un ecosistema, una sociedad, etc. A pesar de la complejidad y la variedad de estos sistemas ¿es posible la existencia de leyes o fenómenos universales aplicables a cada uno de ellos?. Si tales leyes existieran, nos ayudarían a dar respuesta a muchas de las preguntas de la actualidad, como las siguientes: ¿Cómo surge el pensamiento? ¿Cómo están relacionadas las características fenotípicas con el genotipo? ¿Cómo se pueden desarrollar sistemas distribuidos descentralizados para la solución eficiente, robusta y flexible de un problema dado? Por otro lado, las herramientas utilizadas para su estudio, por ejemplo las ecuaciones diferenciales, han mostrado grandes limitaciones para modelar muchas de sus características como las interacciones no lineales y la estructura de interacción, entre otras. El uso de modelos computacionales ha resultado muy útil para cubrir muchas de estas limitaciones.

El estudio de la estructura de interacción entre los elementos de un sistema es de gran importancia, ya que la estructura siempre afecta a la función, por ejemplo, la topología de una red social afecta en la propagación de información o enfermedades, y la topología de una red metabólica afecta en la robustez y estabilidad de la célula. En términos de sistemas dinámicos, sería interesante entender como una enorme red de sistemas dinámicos interactuando ya sean neuronas, proteínas o personas, se comportan colectivamente dada su dinámica individual y su arquitectura de interconexión.

2. Objetivos

El objetivo principal de este trabajo es el desarrollo de herramientas computacionales que permitan estudiar diferentes aspectos de los sistemas complejos (SC). De manera más precisa, los objtivos particulares en este trabajo son los siguientes:

- Identificar las características más importantes de los SC. Así cómo también sus comportamientos característicos.
- Definir un modelo de cómputo que permita expresar en un lenguaje computacional las características de los SC previamente descritas.
- Diseñar e implementar una nueva arquitectura de software (CSDK) para el desarrollo eficiente de simulaciones de SC basadas en el modelo de cómputo descrito. Utilizar CSDK para implementar algunos de los modelos que han sido utilizados para ejemplificar distintos aspectos de los sistemas complejos.
- Desarrollar un nuevo algoritmo de optimización (denominado Optimización Extrema Distribuida) utilizando el conocimiento sobre redes complejas.
- Modificar e implementar la metodología para la extracción de Motifs de redes[**72**] –subgrafos (tres y cuatro nodos en este estudio) que ocurren en la red bajo estudio en cantidades estadísticamente significativas comparadas con redes aleatorias con las mismas propiedades globales.- en redes complejas.
- Desarrollar un algoritmo para encontrar la estructura organizacional de las redes complejas, en particular, un algoritmo que encuentre patrones de interconexión entre Motifs.
- Por último utilizar CSDK para estudiar el papel dinamico que tienen los Motifs de redes en la red de regulación de la bacteria *E coli*.

3. Estructura de la Tesis

A continuación se describe brevemente el contenido de cada capítulo. De manera general, los experimentos y resultados de esta tesis estan en las secciónes finales de cada capítulo.

En el capítulo 1, se identifican las características generales de los SC. En el capítulo 2 se define un modelo de cómputo que permite estudiar desde un punto de vista computacional el papel que juegan cada una de las características previamente identificadas en el comportamiento global o la función de los SC.

En el capítulo 3, se desarrolla una arquitectura de software (denominada CSDK), cuyo objetivo es facilitar el desarrollo de simulaciones basadas en el modelo de cómputo descrito. Para mostrar el uso de la herramienta se implementan los modelos que han servido para explicar diferentes aspectos de los SC.

La estructura o topología de interacción (identificada en el capítulo 2) entre los elementos de los diferentes SC es una característica esencial, porque ésta determina fuertemente el comportamiento global del sistema. En el capítulo 4 se hace una revisión general de los principales avances que se han hecho en esta línea, en particular se estudia cómo afectan las características estadísticas globales de las redes reales en los diferentes procesos dinámicos que tienen lugar en ellas. Se propone un nuevo algoritmo de optimización denominado Optimización Extrema Distribuida (OED) utilizando dichos procesos dinámicos.

En el capítulo 5 también se estudian las redes complejas, pero no en base a características estadísticas, sino a un nivel local. Se describe, implementa y modifica la metodología presentada en [**72**] (referida en esta tesis como MBM -Metodología para la búsqueda de Motifs de redes-) para entender los principios de diseño en las redes complejas a nivel local. Dicha metodología consiste en encontrar los subgrafos que ocurren en las redes de los sistemas complejos en un número significativamente mayor, comparados con redes aleatorias con las mismas características estadísticas. Se utiliza para estudiar redes complejas reales, por ejemplo, la red de regulación genética de la bacteria *E coli*, del hongo *S cerevisiae* y redes de ecosistemas entre otras.

En el capítulo 6 se describe una nueva metodología para estudiar la forma en que están interconectados los Motifs de redes (denominada MIM -Metodología para el estudio de la topología de interacción entre Motifs), esto es, se estudia un nivel organizacional mayor al previemente estudiado. A pesar de que diferentes sistemas tienen los mismos Motifs de redes, la forma en como éstos estan estructurados es muy distinta. En particular se observa una gran diferencia entre las redes naturales y artificiales. Finalemnte se utiliza la metodología para la identificación de módulos en la red de regulación transcripcional de *E coli*.

Por último, en el capitulo 7 se utiliza CSDK para modelar el comportamiento de uno de los Motifs de redes en el contexto de la red de regulación transcripcional.

En la Fig. 1 se muestra la estructura de la tesis. Las flechas indican dependencias entre cada uno de los capítulos.

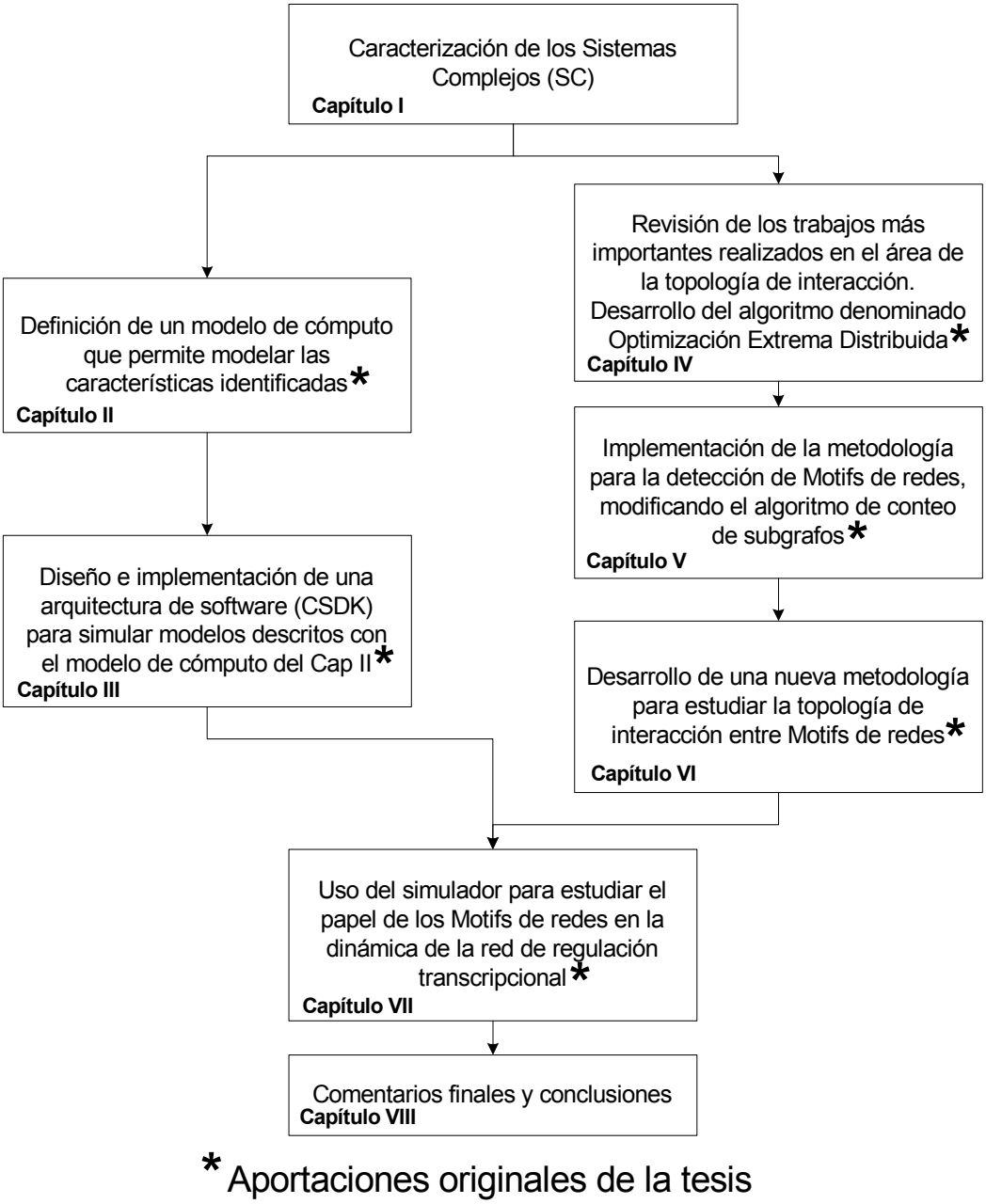


FIGURA 1. Estructura de la tesis. Se indican los objetivos principales de cada capitulo. Las flechas indican dependencias entre cada capitulo.

CAPÍTULO 1

Sistemas Complejos

Resumen. Un sistema Complejo SC se define como un sistema conformado por un conjunto de entidades, cada una de ellas descrita por muchos grados de libertad. En este capitulo se identifican y describen las características más importantes de las entidades de dichos SC: muchos grados de libertad, interacciones no-lineales, interacciones locales, interacciones globales, interacciones estocásticas, elementos heterogéneos, topología de interacción compleja y dinámicas individuales cambiantes. Así cómo también los comportamientos característicos de los SC: transiciones de fase, correlaciones de largo alcance y auto-organización.

1. Caracterización de los Sistemas complejos

A pesar de que se han realizado avances en el área de los Sistemas Complejos (SC) y se tienen buenas aproximaciones sobre el tipo de fenómenos que ocurren en los mismos, no hay una teoría que englobe todo este estudio. De hecho, aún no se ha llegado a una definición generalmente aceptada de sistemas complejos. Algunas de las definiciones más utilizadas son: 1) La palabra complejidad se refiere al estudio de sistemas que operan en el 'borde del caos' (de por si, un concepto pobremente definido); para inferir estructura en las propiedades complejas de los sistemas que están entre el perfecto orden y el perfecto desorden[**1**], 2) la reafirmación del cliché de que el comportamiento de algunos sistemas como un todo, puede ser más que la suma de sus partes[**2**], 3) Un sistema complejo es aquél cuyas propiedades no pueden ser completamente explicadas por el entendimiento de sus partes componentes (emergencia de comportamiento)[**3**]. Si bien estas definiciones de Sistemas Complejos expresan de alguna forma la importancia en el estudio de los mismos, no son muy útiles como base para su estudio, ya que no describen cuales son las caracteristicas del sistema como tal, sino que sólo expresan (de manera muy general) cómo debe ser su comportamiento global. Por esta razón, en este trabajo se toma la definición dada por N. B. Tufillaro[**98**], en la que expresa a los participantes que conforman un SC.

Definición (Sistema Complejo)[98]: Es un sistema conformado por un conjunto de entidades, cada una de ellas descrita por "muchos" grados de libertad.

A continuación se da una caracterización de los Sistemas Complejos utilizando como base la definición anterior. La caracterización se realiza desde dos perspectivas diferentes. Primero se caracteriza a un SC desde un punto de vista de las propiedades que tienen las entidades dadas en la definción. Por ejemplo, una propiedad de los Sistemas Complejos es que sus entidades interactuan de manera no-lineales.

También se caracterizaran a los sistemas complejos en base a las propiedades de su comportamiento (su dinámica) a nivel global, utilizando diferentes tipos de herramientas

matemáticas y/o computacionales. El objetivo es contar con un marco de trabajo, que nos permita encontrar relaciones entre las propiedades de modelado y las propiedades del comportamiento de los Sistemas Complejos (Fig 1).

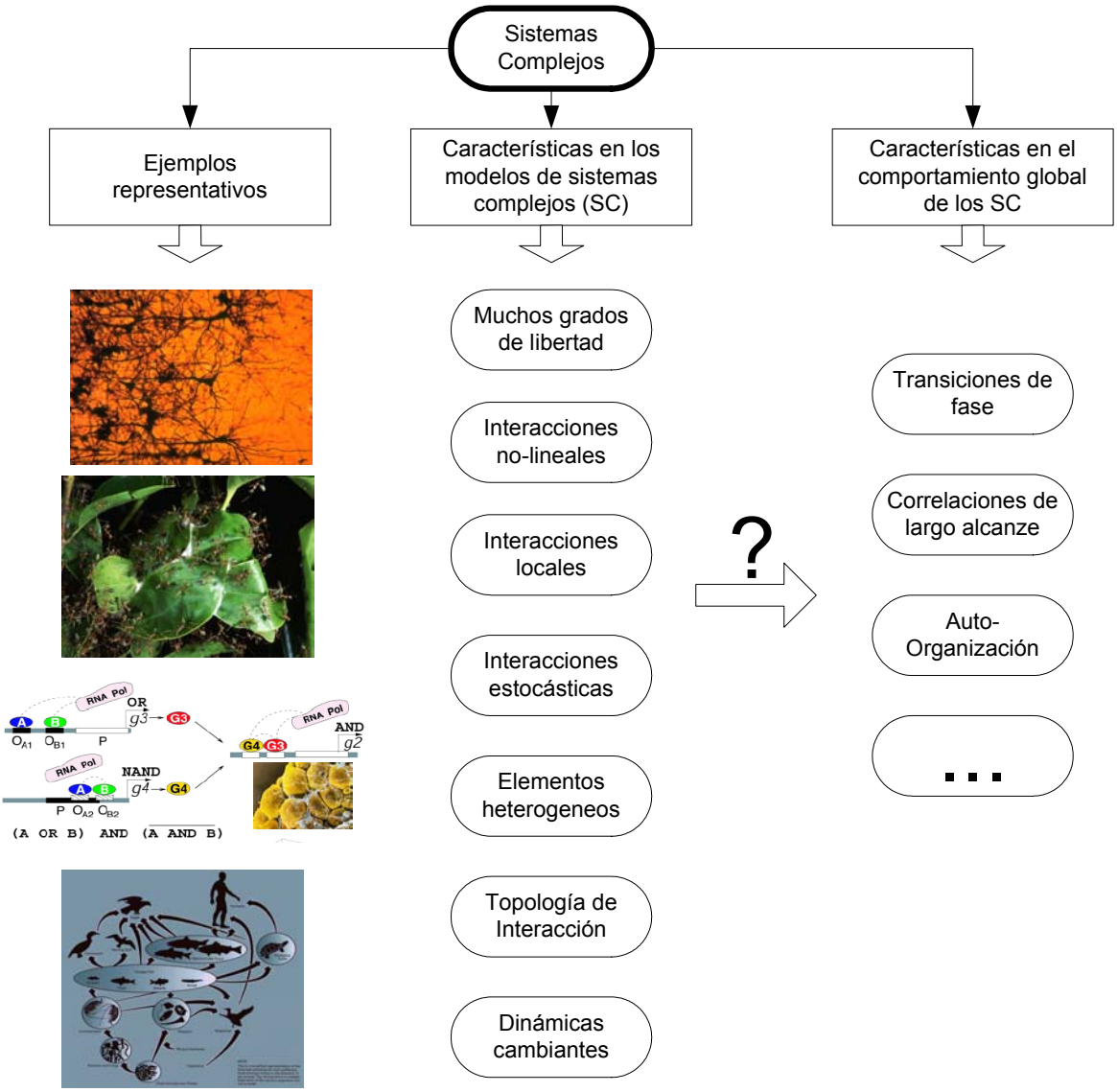


FIGURA 1. Principales propiedades de los sistemas complejos. Junto con algunos ejemplos de los mismos

1.1. Características en la estructura interna de los Sistemas Complejos.

1.1.1. ***"Muchos" grados de libertad.*** Para describir el comportamiento dinámico de un sistema se utiliza el concepto de espacio de estados (o espacio de fase). Por ejemplo, el movimiento oscilatorio de una masa en un resorte ideal a lo largo del eje x (ley de Hooke), se describe mediante la segunda ley de Newton $\mathbf{F} = m\mathbf{a}$ de la siguiente forma

$$Fx = m\frac{d^2x}{dt^2} = -kx$$

donde k es la constante del resorte y m es la masa de la partícula. El movimiento de este sistema esta determinando especificando la posición y la velocidad en cada instante de tiempo. Tradicionalmente, se escoge $t = 0$ para el tiempo inicial, y $x(t = 0)$ y $dx/dt(t = 0) \equiv v_0$ son las condiciones iniciales para el sistema. El movimiento está gobernado por la ecuación,

$$x(t) = x_0 \cos wt + \frac{v_0}{w} \sin wt \tag{1.1}$$

donde $w = \sqrt{k/m}$ es la frecuencia (angular) de la oscilación. Diferenciando 1.1 con respecto al tiempo, se encuentra la ecuación para la velocidad,

$$v(t) = -wx_0 \sin wt + v_0 \cos wt$$

Debido a que el conocimiento de $x(t)$ y $v(t)$ especifica completamente el comportamiento de este sistema, entonces se dice que el sistema tiene **"dos grados de libertad"**. Para cualquier instante de tiempo se puede especificar el estado del sistema con un punto en una gráfica v contra x. A esta gráfica es a la que se le llama el espacio de estados para el sistema. En este caso, el espacio de estados es de dos dimensiones.

De esta manera se puede definir el **número de grados de libertad** como el número de variables independientes necesarias para determinar el estado dinámico de un sistema o de manera equivalente, el número de condiciones iniciales independientes que tienen que ser especificadas para el sistema[**19**]. Esta definición no sólo se utiliza para sistemas continuos, sino también para sistemas discretos.

En el área de sistemas complejos se busca estudiar las leyes que gobiernan el comportamiento de entidades macroscópicas, por ejemplo, economías, el cerebro, nichos ecológicos, etc. Esto es, entidades formadas por un gran número de entidades microscópicas como empresas, neuronas, animales, etc. cada una con sus propios grados de libertad (ver definición).

Se puede decir, que conforme el número de entidades microscópicas aumenta (y por lo tanto, también el número de grados de libertad) también aumenta el grado de irregularidad en el comportamiento de la entidad macroscópica. Sin embargo, esto no es así, y se vera como cuando el número de entidades microscópicas es muy grande, aparecen nuevos tipos de regularidades. Estas nuevas propiedades, resultado de la presencia de un gran número de entidades microscópicas no pueden de ninguna forma ser reducidas a las leyes que rigen a las entidades microscópicas. Por ejemplo, cuando se aplican a sistemas mecánicos con un número pequeño de grados de libertad estas leyes dejan de tener sentido. Así, aunque el movimiento de un sistema con muchos grados de libertad sigue las mismas leyes de la mecánica de la misma manera que el sistema

compuesto con pocos grados de libertad, la existencia de muchos grados de libertad resulta en leyes de diferente tipo.

1.1.2. ***Interacciones No-Lineales***. Las ecuaciones diferenciales describen la evolución de un sistema en tiempo continuo. Han sido ampliamente utilizadas en la ciencia y la tecnología. Existen dos tipos principales de ecuaciones diferenciales: las ordinarias y las parciales. Por ejemplo, la ecuación de la dinámica de un oscilador armónico amortiguado,

$$m\frac{d^2x}{dt^2} + b\frac{dx}{dt} + kx = 0,$$

es una ecuación diferencial ordinaria, porque sólo involucra derivadas ordinarias dx/dt y d^2x/dt^2. Esto es, sólo existe una variable independiente, el tiempo t. En contraste, la ecuación de calor

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

es una ecuación diferencial parcial ya que tiene tanto al tiempo t como al espacio x como variables independientes. Un marco generalizado, para el estudio de las ecuaciones diferenciales ordinarias está dado por el siguiente sistema[**19**],

$$\begin{aligned}
\frac{dx_1}{dt} &= f_1(x_1, x_2, ..., x_n) \\
\frac{dx_2}{dt} &= f_2(x_1, x_2, ..., x_n) \\
&... \\
\frac{dx_n}{dt} &= f_n(x_1, x_2, ..., x_n)
\end{aligned}$$

Las variables $x_1, x_2, ..., x_n$ son las variables de estado del sistema, por ejemplo, las diferentes concentraciones de químicos en un reactor, las poblaciones de diferentes especies en un ecosistema, la posición y velocidad de los planetas en un sistema solar, etc. Las funciones $f_1, f_2, ..., f_n$ modelan el comportamiento del sistema especifican como es la tasa de cambio de cada una de las variables en función de todas las demás.

Se dice que un sistema es **lineal** si las funciones f_i son de la forma $f_i = a_{1i}x_1 + a_{2i}x_2 + ... + a_{ni}x_n$ es decir, todas las x_i sólo aparecen como la primera potencia. De otra manera el sistema es **no-lineal**.

¿Por qué los problemas no-lineales son tan difíciles de analizar? La mayoría de los sistemas no lineales son imposibles de resolver analíticamente. ¿Por qué son mucho más difíciles de analizar que los sistemas lineales? La diferencia esencial es que los sistemas lineales pueden ser descompuestos en partes, después, cada parte puede ser resuelta por separado, y finalmente combinar todas las respuestas para obtener el resultado final. Esta idea permite una simplificación fantástica de problemas muy complicados y es la base para métodos como la transformada de Laplace, argumentos de superposición, etc. En este sentido, un sistema lineal es precisamente igual a la suma de sus partes.

Sin embargo, la mayoría de los fenómenos de la naturaleza no actúan de esta forma. Las partes del sistema interfieren o cooperan o compiten entre sí es decir, siempre hay interacciones no lineales, y el principio de superposición falla espectacularmente. Si

alguien escucha sus dos canciones favoritas, no obtiene el doble de placer. En física la no-linealidad es vital para la operación de un láser, la formación de turbulencia en un fluido, etc.

1.1.3. ***Interacciones locales.*** Se ha observado que las reglas que especifican las interacciones entre los componentes de un sistema son ejecutadas en base a pura información local, sin ninguna referencia al comportamiento global. Por ejemplo, en modelos de colonias de hormigas, se observó como los miembros de la población pueden construir grandes estructuras (como nidos), cada uno actuando unicamente en base a reglas locales[**5**]. Es intrigante como la naturaleza puede construir estructuras complicadas satisfactoriamente sin el "conocimiento" de lo que está construyendo.

El cerebro es otro ejemplo, en donde cada uno de sus componentes (las neuronas) sólo interactúan de manera local. Es importante señalar que se está hablando de interacciones locales directas, por ejemplo, por una cercanía espacial o un enlace físico. Sin embargo, como se verá más adelante el sistema puede evolucionar a un punto en el que existen interacciones de largo alcance entre los elementos del sistema. Es decir el comportamiento de una entidad afecta directamente al de otra, aunque no exista una interacción directa entre las mismas.

Desde otra perspectiva, una razón por la cual ha sido posible encontrar leyes físicas tan simples, se basa en una posible propiedad de localidad. Si cualquier cosa en el universo interactuara con todas las demás cosas de una manera altamente no-local, no se podría entender una parte sin entenderlo todo[**6**].

1.1.4. ***Interacciones globales.*** En muchos sistemas, además de las interacciones locales también existen interacciones globales que afectan a todas las entidades del sistema, por ejemplo, en los modelos de Ising existe un campo magnetico externo H que afecta con la misma intensidad a todos los momentos magnéticos o spines del sistema. El comportamiento cólectivo de este sistema es completamente depndiente del valor de H. De hecho puede observarse una transición de fase (más adelante se dará una breve explicación) cuando se varía H.

1.1.5. ***Interacciones estocásticas.*** En muchos problemas de física moderna, los procesos ruidosos (aleatorios) pueden resultar en un estado altamente ordenado. El orden inducido por el ruido puede tomar muchas formas, desde la separación de diferentes materiales en un estado final heterogéneo, hasta la formación de una gran variedad de patrones regulares, incluyendo cuadrados, hexágonos y espirales[**7**]. La herramienta matemática que se ha utilizado para modelar los procesos son las ecuaciones diferenciales estocásticas (caso continuo), las cuales se caracterízan por incluir variables aleatorias dentro de sus definiciones, por ejemplo, la ecuación diferencial $dx/dt = a(x,t)$, se transforma en una ecuación diferencial estocástica si se le agrega un componente aleatorio: $dx/dt = a(x,t) + b(x,t)\eta(t)$, donde $\eta(t)$ es una variable aleatoria que depende del tiempo.

El uso del ruido para propósitos productivos tiene una larga e interesante historia. Desde hace un siglo, los marineros han reportado que las precipitaciones pluviales desordenadas que caen en el océano pueden calmar al mar[**8**]. El ruido en forma de pequeñas perturbaciones provoca inestabilidades que llevan a oscilaciones regulares en sistemas que van desde instrumentos musicales hasta colapsos estructurales catastróficos.

Es útil diferenciar entre sistemas en los cuales el ruido mejora o inicia un proceso existente y sistemas cualitativamente diferentes en los cuales el orden no aparecería sin la presencia de ruido. Por ejemplo, en la búsqueda de rutas por las colonias de hormigas, la aleatoriedad es crucial, porque permiten el descubrimiento de nuevas soluciones, y las fluctuaciones pueden actuar como semillas a partir de las cuales comienzan a crecer estructuras. Lo que se pretende estudiar es la forma en que el ruido (o las interacciones aleatorias) produce estados ordenados.

1.1.6. ***Elementos heterogéneos.*** Una característica de los sistemas complejos es que están constituidos por entidades estructuralmente (constitución interna) y dinámicamente (comportamiento externo) diversas aunado que, durante la evolución de cada entidad tanto la constitución como la dinámica de la misma, puedan estar cambiando continuamente. La heterogeneidad de las entidades en ambos niveles es una de las principales causas para que el comportamiento global del sistema no pueda ser explicado simplemente por la comprensión de los comportamientos individuales de las entidades.

Aun así, existe un comportamiento característico global (estado ordenado) debido a la diversidad de las entidades.

1.1.7. ***Topología de interacción (redes complejas).*** El estudio de las redes se extiende en muchas áreas de investigación, desde economía hasta física estadística. El tema básico es estructural: ¿Cómo se caracteriza la red de interconexión de una red de alimentos o la Internet o la red metabólica o la bacteria Escherichia coli? ¿Existen principios universales basados en la topología? Desde la perspectiva de dinámica ¿Cómo una red enorme de sistemas dinámicos interactuando ya sean neuronas, estaciones de poder o lasers se comportarán colectivamente, dado su dinámica individual y su arquitectura de acoplamiento?

La estructura es importante, porque ésta siempre afecta a la función, por ejemplo, la estructura de una red social afecta en la propagación de la información y de las enfermedades, y la topología de una red de energía afecta en su robustez y estabilidad[**9**].

1.1.8. ***Dinámicas individuales cambiantes.*** Durante la evolución de un sistema complejo, éste presenta un cierto comportamiento característico que debe ser mantenido la mayor parte del tiempo. Dado que el sistema puede interaccionar a su vez con otros sistemas externos estar sujeto a perturbaciones o fluctuaciones este debe tener un mecanismo que le permita mantener su comportamiento característico o responder de forma controlada al ambiente en el que se encuentra (adaptación).

Dado que este comportamiento es el resultado tanto de los comportamientos individuales de las entidades como de las interacciones entre si, el mecanismo de adaptación debe permitir que cada entidad mantenga un modelo interno del ambiente que se encuentra y en base a ello cambiar su dinámica como respuesta a los eventos externos[**10**]. Esto implica que al remover un tipo de entidad o presentarse una fluctuación externa, se originen una serie de cambios para mantener el comportamiento global característico.

El mecanismo de adaptación que sigue cada entidad requiere una componente de memoria en donde se codifica la información obtenida del ambiente. Esta información es analizada, identificando patrones los cuales son comprimidos en un modelo. En un instante pueden existir otros modelos que se adaptan mejor a la situación actual, por lo que el mejor modelo (o la combinación de modelos) debe ser utilizado y el peor adaptado

debe ser eliminado. Esto conlleva a un proceso de retroalimentación en donde la función de adaptación esta bien definida y como resultado se obtiene un modelo reequipado con la mejor adaptación. Finalmente, esta actualización del modelo provoca que la dinámica de la entidad cambie.

1.2. Características del comportamiento de los sistemas complejos. El objetivo general en el estudio de los sistemas complejos es explicar el comportamiento global de un sistema dadas las características específicas del modelo. Si bien no se ha identificado una relación directa entre las características antes mencionadas con el comportamiento global, sí se han encontrado comportamientos característicos. A continuación se describen brevemente dichos comportamientos.

1.2.1. ***Transiciones de fase.*** El término transición de fase se asocia a cualquier cambio abrupto en la dinámica del sistema. Aunque en el sentido estricto, la transición de fase requiere que exista una singularidad en un potencial termodinámico (Como la energía libre de Gibbs), el cual requiere que el estado del sistema se pueda caracterizar en el límite termodinámico de un tamaño de sistema infinito. Pero la abstracción puede ser muy útil, en el sentido de que las transiciones de fase no sólo ocurren cuando un líquido se evapora o congela, sino también en sistemas económicos, biológicos, sociales, computacionales, matemáticos, etc. Por ejemplo, en el estudio de redes aleatorias, existe una probabilidad crítica p (En este caso si depende del tamaño del sistema) que describe el cambio de una red no-conexa a una red conexa[**11**]. El punto es que la transición de fase es global y abrupta.

Uno de los aspectos más importantes de las transiciones de fase es que muestra aspectos comunes entre diferentes sistemas. Por ejemplo, el comportamiento crítico de un gas a líquido y el comportamiento de ciertos magnetos en su punto de Curie (La temperatura sobre la cual pierden su ferromagnetismo) tienen exponentes críticos numéricamente iguales y ambos son modelados con el modelo de Ising. Esto nos da una idea de universalidad es decir, existen modelos genéricos que describen una variedad de aparentemente diferentes sistemas de muchas entidades[**12**].

1.2.2. ***Correlaciones de largo alcance.*** El estudio de las correlaciones y sus consecuencias es una parte esencial en el análisis de un sistema, debido a que las correlaciones nos dan indicios de los posibles mecanismos subyacentes. Una característica que se presenta en sistemas a los que hemos llamado complejos es la presencia de correlaciones de largo alcance, a pesar de que la dinámica de cada una de las entidades depende sólo de interacciones locales. Estas correlaciones de largo alcance son una huella del comportamiento colectivo de un sistema[**13**].

En los sistemas complejos, no sólo existen correlaciones entre los elementos del sistema sino también en agregados de elementos, generalmente representado como correlaciones entre sumas de variables aleatorias. Esto nos indica que los sistemas complejos tienen muchos *niveles de organización* o *niveles jerárquicos*, entidades de un nivel sirven como bloques de construcción para entidades de un nivel mayor. Por ejemplo, un grupo de proteínas, lípidos, etc forman una célula, un grupo de células forman un tejido, un grupo de tejidos forman un órgano, una asociación de órganos forman un organismo completo y un grupo de organismos forman un ecosistema. De esta manera, los modelos que esperamos encontrar para la representación general de un sistema complejo a un

cierto nivel servirán como las entidades básicas para la construcción de otro sistema complejo, de modo que en cada nivel se encontrarán nuevas interacciones.

1.2.3. ***Auto-organización.*** El término auto-organización se ha utilizado por muchos años para describir la habilidad de ciertos sistemas en no equilibrio para desarrollar estructuras y patrones en la ausencia de control o manipulación por un agente externo[**5**]. Ejemplos incluyen, el crecimiento de patrones en reacciones químicas y el desarrollo de estructuras en los sistemas biológicos.

El punto de vista más pesimista para tratar de entender este mecanismo para la generación de estructuras es que se tienen que definir todas y cada una de las estructuras en la naturaleza, caso por caso. Ciertamente, tal aproximación (colección de estampillas) ha prevalecido en las ciencias como la biología y la geofísica, aunque ha habido excepciones, como la teoría de platos tectónicos, o las redes de regulación de Kauffman. Tal vez, la naturaleza no necesita inventar una gran cantidad de mecanismos, uno para cada sistema. El punto de vista de que un número limitado de mecanismos o principios generen estructuras en todas sus manifestaciones (desde galaxias hasta, células y moléculas) está basado en la observación de las regularidades que aparecen en la descripción estadística de los sistemas complejos[**14**].

2. Herramientas para el estudio de sistemas complejos

Se han desarrollado muchas técnicas para el estudio de los sistemas complejos, las cuales se puede dividir en dos clases: aquellas que analizan datos u observables tomadas del fenómeno complejo y aquellas para construir y entender modelos (sin utilizar datos). Una introducción a muchas de estas herramientas puede encontrarse en[**15**].

En la primera clase, se han utilizado ideas de *aprendizaje estadístico* y *minería de datos*. Los desarrollos en estas áreas han extendido el análisis de los métodos estadísticos más allá de su dominio tradicional de pocas dimensiones e independencia en los datos, y han sido utilizadas para la construcción automática de modelos a partir de los datos y el desarrollo de otras formas de representación más útiles para la comprensión del fenómeno. Estas herramientas son parte del *análisis de series de tiempo*, en donde también están incorporados conceptos de dinámica no lineal[**16**]. Los objetivos principales de estas herramientas son la predicción y la extracción de relaciones casuales de las variables de los fenómenos. En la Fig. 2, se muestra como están relacionadas algunas de estas herramientas para cumplir con estos objetivos.

La segunda clase, son herramientas para la construcción de modelos de sistemas complejos. Entre las principales herramientas están la teoría de sistemas dinámicos para la construcción de modelos continuos, como los sistemas de ecuaciones diferenciales, y discretos como las ecuaciones de diferencias; la teoría de la computación para la construcción de modelos de autómata celular que reflejan de una manera muy adecuada las interacciones locales presentes en los sistemas naturales; las herramientas de mecánica estadística, entre otras. El tema principal de esta tesis trata la clase de modelos computacionales, en particular una extensión al modelo de Automata Celular (AC), que permite estudiar de una manera más directa el papel que juega cada una de las características de los sistemas complejos en su comportamiento global. En específico, se estudia a fondo el papel que tiene la topología de interacción de las entidades de un sistema complejo (redes complejas), en su dinámica global. Además, las observables

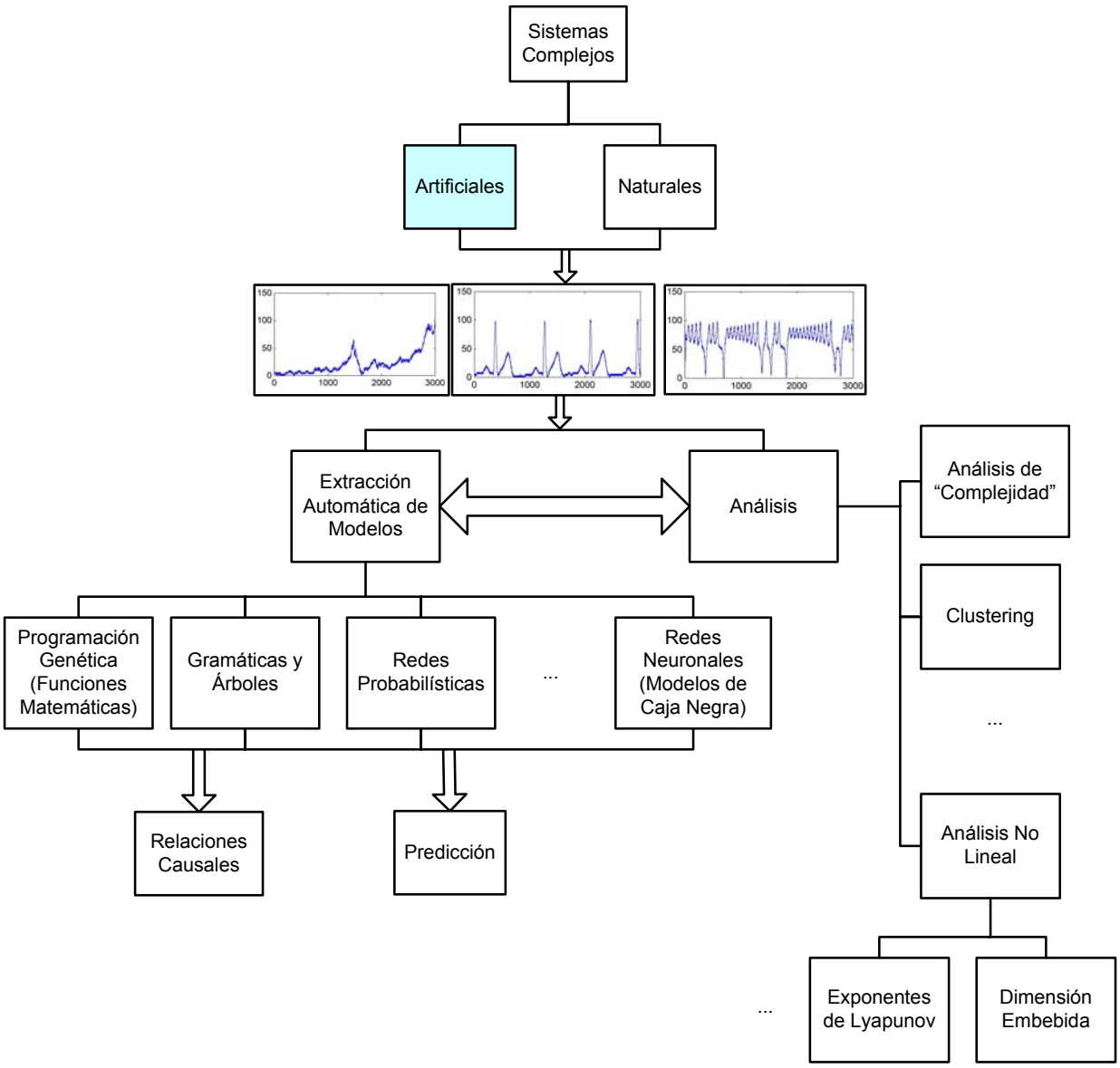


FIGURA 2. Análisis de series de tiempo para el estudio de los sistemas complejos.

tomadas de este modelo, así como de otros, pueden considerarse como observables artificiales (Fig. 2), las cuales pueden ser estudiadas con todas las herramientas del análisis de series de tiempo.

CAPÍTULO 2

Sistemas complejos y computación

Resumen. Las herramientas matemáticas tradicionales, por ejemplo, los sistemas de ecuaciones diferenciales han fallado en explicar tanto cuantitativa como cualitativamente varias de las propiedades globales presentes en los SC. En este capítulo se define un modelo de cómputo que permite estudiar desde un punto de vista computacional el papel que juegan cada una de las características previamente identificadas en el comportamiento global o la función de los SC.

Se puede ver la relación entre la teoría de la computación y las ciencias naturales (y aún las ciencias sociales) desde dos puntos de vista:

Primero, cualquier sistema físico puede ser visto como un proceso computacional. Se puede especular que los sistemas físicos corresponden a un tipo de cómputo muy específico; con un modelo de cómputo particular. Desde este punto de vista es absolutamente razonable investigar los sistemas físicos con conceptos y métodos desarrollados en la ciencia de la computación[**21**]. El cómputo mediante DNA[**90**] y el cómputo cuántico[**35**] son ejemplos de cómo se utiliza la dinámica particular de un fenómeno natural para llevar a cabo el cómputo de algún problema.

En un sentido más amplio, se puede decir que cualquier teoría, como la mecánica cuántica o una posible teoría de sistemas complejos, puede ser utilizada como base para una teoría de procesamiento de información, que nos permita resolver de manera eficiente problemas para los cuales sólo se cuenta con soluciones no factibles utilizando teorías tradicionales. Uno de los objetivos de este capítulo es clarificar esta aseveración.

En este capítulo se utiliza la caracterización de los sistemas complejos, para desarrollar un modelo de cómputo que nos permita, tanto entender los fenómenos complejos desde una perspectiva computacional, como utilizar las propiedades de tales sistemas para solucionar problemas como los NP-completos.

1. Modelado computacional de los sistemas complejos

Con la caracterización dada en el capítulo anterior, se buscaron aspectos comunes en sistemas tan diversos como el cerebro, un nicho ecológico o una economía, con el objetivo de encontrar patrones abstractos que sean aplicados a muchos de ellos.

Para la descripción de cada patrón, se debe buscar una representación (modelo) adecuada. Por ejemplo, la forma más apropiada para describirle a alguien el camino hacia algún lugar desconocido para él es mediante un mapa; un texto sería mucho menos apropiado. En este mismo sentido, se requieren de diferentes formas de representación para capturar diferentes características generales de los sistemas complejos. Desafortunadamente, aún no se cuenta con las herramientas adecuadas para la descripción de muchos aspectos de los mismos. Por ejemplo, el uso de sistemas de ecuaciones diferenciales para modelar un sistema complejo resulta inadecuado, porque para la mayoría de

los sistemas no-lineales (casi todos) no se cuenta con una solución analítica[**19**]. Se ha tratado de aproximar la solución mediante la integración numérica de tales sistemas de ecuaciones, pero esto falla, porque las entidades presentes en las ecuaciones diferenciales (entidades continuas) no pueden ser representadas en los registros de una computadora (entidades discretas). Este problema, se ve amplificado debido al fenómeno de divergencia de trayectorias cercanas, presentado por sistemas caóticos, en los cuales diferencias mínimas en las condiciones iniciales pueden generar comportamientos completamente diferentes[**19**], impidiendo así la posibilidad de una predicción a largo plazo.

Sin embargo, el concepto abstracto de no-linealidad ($f(a+b) \neq f(a)+f(b)$) es de mucha importancia, por lo que se tiene que encontrar una forma (lenguaje) de representación que sea más útil que el de las ecuaciones diferenciales.

En el estudio de los sistemas complejos sería útil tener patrones abstractos de comportamiento, diferente al de razón de cambio utilizado en ecuaciones diferenciales. Para esto un modelado computacional (algorítmico) sería el candidato natural. Un algoritmo es un procedimiento que puede ser aplicado a cualquier tipo de entradas simbólicas, y el cual eventualmente producirá para cada entrada una correspondiente salida simbólica[**20**]. Por ejemplo, si se está modelando una hormiga mediante un algoritmo, las entradas simbólicas pueden ser la representación del ambiente en el cual está inmerso, como la cantidad de feromonas que hay a su alrededor, si hay alimento, etc. Y las salidas simbólicas pueden ser la representación de los efectos causados por tales condiciones ambientales, como el movimiento de la hormiga hacia donde hay más cantidad de feromonas. Así, el algoritmo modela el comportamiento de la hormiga. Ahora, lo que se busca es describir propiedades abstractas en tales programas de la misma manera como se hace en las ecuaciones diferenciales: "si la ecuación es de segundo orden y los coeficientes que acompañan a las funciones son constantes... entonces...".

Además, con el modelado computacional se tiene un mecanismo de inferencia de "teoremas" automático inherente: la ejecución del algoritmo. Así, por ejemplo, en sistemas complejos, el resultado de una simulación, ya sea la convergencia a un atractor particular o un comportamiento característico, se puede ver como una propiedad global que tiene un sistema en el cual las entidades (algoritmos) tienen ciertas propiedades específicas, en donde la simulación misma es la demostración de tal afirmación. Es decir, un sistema S compuesto por algoritmos con propiedades $p_1, \ldots, p_n$ implica una propiedad global P_x si y sólo si el resultado de la simulación (o ejecución) de S produce como salida a P_x. La propiedad global P_x puede ser expresada con cualquier lenguaje de modelado, por ejemplo, puede ser el resultado del análisis de una secuencia de valores (símbolos) generados por la simulación. Un problema con este tipo de modelado es que la demostración es específica de la condición inicial dada. Se debe encontrar la manera de generalizar los resultados obtenidos para conjuntos de condiciones iniciales cada vez más grandes.

Al igual que todas las diferentes formas de modelado, el modelado computacional y los resultados de las simulaciones son sólo abstracciones, por lo que para hacer una asociación a cualquier fenómeno real, se debe de corroborar con experimentos. De igual forma, se debe de analizar las limitaciones de esta forma de modelado como la no computabilidad es decir ¿existen fenómenos no computables así como funciones no computables? Un argumento utilizado comúnmente es la naturaleza continua de los fenómenos naturales. Sin embargo, se esta desarrollando una nueva teoría física,

llamada teoría de lazos gravitacionales cuánticos, la cual predice que tanto el espacio como el tiempo son discretos[**91**].

2. Modelo computacional (discreto) de sistemas complejos

En base a la caracterización de los SC antes descrita, se establece un modelo computacional que permite el cómputo universal y el desarrollo de modelos de SCs. El objetivo es representar en un lenguaje computacional las características generales de los SC, de manera que se pueda estudiar el papel que juega cada una de éstas en el comportamiento global de los SC. Este modelo también puede utilizarse para dar una definición computacional de Sistema Complejo.

Definición (**Sistema Multi-Entidad**) Se suponen ciclos de tiempo discretos, etiquetados $0, 1, 2, 3, ...$ Un SME es una 3-tupla $SME = (M, H, T)$ donde $M = \{M_1, M_2, ..., M_n\}$ con $n \in \mathbf{N}$ es un conjunto de Máquinas RAM aleatorias, conectadas localmente en una topología especificada por una función de vecindad $H : M \to P(M)$, donde $P(M)$ denota al conjunto potencia de S. La evolución temporal, puede ser síncrónica, asincrónica u otra y está dada por una función $T : \mathbf{N} \to P(M)$ que especifica para cada ciclo de tiempo t, el conjunto de máquinas que se ejecutarán hasta parar.

Cada máquina RAM M_i, tiene el siguiente conjunto de registros especiales:

- Un conjunto de *registros públicos* R_i, que pueden ser accesados por las máquinas vecinas. Estos registros mantienen su valor en cada ejecución de la máquina.
- Un conjunto de *registros privados* I_i que sólo pueden ser accesados por M_i y que mantienen su valor en cada ejecución de la máquina. Estos se pueden ver como la memoria de la entidad.
- Un conjunto de *registros locales* L_i que se utilizan para los cálculos de M_i. Estos siempre se inicializan en cero en cada ejecución de la máquina.

En cada ciclo de tiempo t las máquinas especificadas por T computarán su algoritmo A_i el cual toma como entrada los valores en el tiempo t de los registros públicos de las máquinas vecinas especificadas por H, y los registros públicos y privados propios. El conjunto posible de entradas se denota por V_i. En general, los resultados de tal cómputo se pueden almacenar en cualquiera de los registros R_i o I_i. El conjunto de posibles salidas se denota por O_i.

A continuación, se da una explicación más detallada de cada uno de los componentes del SME y como se relacionan con las características de los sistemas complejos.

2.1. Entidades. Con la teoría de las funciones recursivas, se puede especificar un algoritmo A, por la función $f_A : V_i \to O_i$ que éste computa. De este modo es posible asociar propiedades a tales algoritmos en términos de propiedades de la función que el algoritmo computa. Por ejemplo, el algoritmo que calcula cada entidad (máquina RAM) puede ser lineal o no lineal, si la función que computa es lineal o no lineal respectivamente. Una función D_i es no lineal si dados $x_1, x_2 \in V_i$ (asumiendo que se define una operación suma sobre los vectores en V_i) entonces,

$$D_i(x_1 + x_2) \neq D_i(x_1) + D_i(x_2)$$

De este modo, a diferencia de los autómatas celulares es posible introducir la no linealidad de una manera más directa. Además, los algoritmos pueden ser aleatorios, es decir, un algoritmo que hace elecciones aleatorias durante su ejecución.

La capacidad de cómputo de cada entidad permite hacer modelos jerárquicos: una vez conocida la función que calcula un sistema o subsistemas, en principio es posible implementarla con cualquier otro modelo computacional. De esta forma, aunque un subsistema esté compuesto por muchas entidades es posible substituirlo por una sola entidad que compute su función.

2.2. Función de vecindad. La función de vecindad H, indica qué entidades interactuan entre sí directamente. La forma más directa para especificar la interaxión entre entidades es mediante un grafo de interconexiones $G = (V, E)$, en donde la función H devuelve para cada M_i los vecinos especificados por el grafo $H(M_i) = \{v|(v, M_i) \in E\}$. Por ejemplo, en una red neuronal, un grafo define la estructura de esta red, es decir, los vértices son las neuronas y las aristas son enlaces sinápticos entre ellas. La dinámica global del sistema depende fuertemente de las características estructurales del grafo[**32**]. Además de la representación de grafo, H puede tomar formas distintas, por ejemplo, supongamos que algunos de los registros públicos de las entidades se utilizan para modelar una posición en el espacio, y la función de vecindad se define de manera tal que para cada entidad M_i se obtengan las entidades que están dentro de una esfera de radio r con centro en M_i.

Una de las líneas de investigación más fértiles en el estudio de los SC ha sido, precisamente, el estudio de las propiedades estructurales en las redes de interacción que hay en diferentes SC y como estas propiedades afectan directamente en la dinámica de los mismos. En capítulos posteriores se estudian los avances más importantes que se han hecho en esta área, y los resultados obtenidos en esta línea.

2.3. Dinámica temporal. La forma de esta función se utiliza para representar el tiempo de cómputo de cada entidad. Es decir, con qué velocidad computan su algoritmo una entidad con respecto a la otra. Por ejemplo si todas las entidades son exactamente iguales (átomos), entonces todas las entidades computan su función en el mismo tiempo y por tanto de manera sincrona, es decir, para cada instante t, $T(t) = M$. Un caso particular de este modelo es el AC en el cual la evolución es siempre síncrona. El caso contrario es que no se conoce la relación entre los tiempos de cómputo de cada entidad, es decir, el sistema es asíncrono. Para modelar tal asincronía, la función T se maneja como una variable aleatoria uniforme U que puede tomar los valores de $\{1, 2, ..., n\}$. El valor que toma para cada tiempo discreto t especifica la entidad que va a evaluarse.

La necesidad de la función T, se debe a que no se conoce de manera precisa los tiempos en que cada entidad debe computar una función, cuando se modela un sistema dado. Esto es porque se está implementando el algoritmo que cada entidad lleva a cabo de manera distinta a como está lo hace en realidad. Este es uno de los costos de la abstracción.

2.4. Ejemplo: propagación de enfermedades. Como ejemplo, se muestra un modelo de propagación de enfermedades en una población. El modelo más simple de propagación de enfermedades sobre una red es el modelo SIR[**66**]. Este modelo divide la población en tres clases: susceptibles (S), lo cual significa que no tienen la enfermedad de interés, pero que pueden contagiarse si se exponen a alguien que la tiene, infeccioso (I), significa que tienen la enfermedad y que pueden transmitirla, y recuperado (R), significa que se ha recuperado de la enfermedad o que ha muerto y por tanto tiene

una inmunidad permanente, de manera que no puede contagiarse de nuevo, ni tampoco transmitir la enfermedad.

En matemática epidemiológica tradicional[**66**] se supone que cualquier individuo susceptible tiene una probabilidad uniforme β por unidad de tiempo de ser contagiado por cualquier agente infeccioso, y que dichos agentes se recuperan y se vuelven inmunes en alguna tasa estocástica constante γ. Las fracciones s, i y r de individuos en los estados S, I y R están gobernadas por las ecuaciones diferenciales,

$$\begin{aligned} \frac{ds}{dt} &= -\beta is \\ \frac{di}{dt} &= \beta i - \gamma i \\ \frac{dr}{dt} &= \gamma i \end{aligned}$$

Los modelos de este tipo se conocen como modelos completamente mezclados, y aunque nos han enseñado mucho acerca de la dinámica básica de las enfermedades, obviamente, las suposiciones que se hacen son demasiado irreales. Las enfermedades sólo pueden ser propagadas entre aquellos individuos que tienen contacto físico, y por lo tanto la estructura de la red de contactos es importante para el patrón de desarrollo de la enfermedad.

A continuación se describe el modelo utilizando el formalismo de los sistemas multientidad.

Sea $I = (P, H, T)$, donde $P = \{M_1, M_2, ..., M_n\}$ es un conjunto de n máquinas RAM aleatorias que representan a acada uno de los individuos de la población, cada M_i tiene un registro público $R_0^i \in \{0, 1, 2\}$ donde $0, 1, 2$ corresponden a los estados S, I y R respectivamente. Las máquinas no tienen registros privados. La función de vecindad H está definida en base a un grafo $G = (V, E)$ que denota la estructura de la población, $V = P$, y existe una arista entre un par de máquinas M_1 y M_2 si existe contacto entre ellos. De modo que $H(m) = \{v|(v, m) \in E\}$. La dinámica f_i que sigue cada entidad es muy simple y la misma para todas $f_i(R_0^i, H(m_i), p) \rightarrow \{0, 1, 2\}$ se define como sigue: si $R_0^i = 1$ entonces $f_i = 2$ (el individo adquiere inmunidad o muere, en un interavlo de tiempo), si $R_0^i = 2$ entonces $f_i = 2$ (sigue inmune en el siguiente intervalo de tiempo) y si $R_0^i = 0$ entonces $f_i = 1$ con probabilidad $P = 1 - p^{I(H(m_i))}$, donde $I(H(m_i))$ es el número de vecinos que están en el estado infeccioso y $f_i = 0$ con probabilidad $1 - P$. Si en el tiempo $t = 0$ es introducido un individuo infectado a la red, conforme el sistema evoluciona, la enfermedad se propaga a lo largo de las aristas del grafo, hasta que infecte a toda la población o termine infectando a alguna fracción de la misma. Se pretende estudiar como afecta una topología particular en la fracción de la población que resulta infectada o la velocidad en que se infecta cierta fracción de la población.

En el siguiente capítulo se describe el diseño y la implementación de una arquitectura de software, denominada CSDK, para implementar modelos especificados con el módelo de cómputo de los sistemas multi-entidad. En particular se implementa el modelo de esparcimiento de enfermedades antes descrito

3. Sistemas complejos como paradigma computacional

Se ha planteado la forma en que los sistemas complejos pueden ser vistos como sistemas computacionales que computan una función específica. Desde esta perspectiva es necesario preguntarse ¿qué tan eficiente es el cálculo de esta función? es decir, ¿cuál es la cantidad de recursos que utiliza el sistema complejo (por ejemplo, recursos temporales) para el cálculo de tal función? Se ha visto que sistemas complejos como las colonias de insectos, el sistema inmunológico, desde luego el cerebro, etc. pueden resolver problemas considerados como intratables con los modelos de cómputo tradicional. Además de esta eficiencia, tales sistemas tiene otras ventajas muy importantes como la robustez: el sistema sigue funcionando (calculando la función) aunque algunas entidades fallen y la flexibilidad: el sistema sigue funcionando en ambientes cambiantes. Una mayor comprensión de los sistemas complejos en general, nos ayudará a encontrar soluciones eficientes, robustas y flexibles a muchos problemas tecnológicos.

3.1. Complejidad computacional. En la ciencia de la computación, se desarrolló la teoría de la complejidad computacional para estudiar los recursos mínimos que necesita un modelo especifico para resolver un problema[**33**]. Desde un punto de vista de diseño, a veces es necesario reducir algún tipo de recurso en particular, aunque generalmente el recurso más importante es el tiempo, una situación específica puede requerir la reducción de otro recurso como el espacio o la cantidad de comunicaciones.

Diferentes modelos computacionales requieren diferentes recursos para resolver el mismo problema. Por ejemplo una máquina de Turing multi-cintas puede resolver muchos problemas substancialmente más rápido que una máquina de Turing de una sola cinta. Aunque la capacidad computacional de ambos modelos es equivalente, la eficiencia en la solución de problemas es distinta. En este sentido, se ha visto, que el modelo de cómputo basado en sistemas complejos posee características que le permiten aproximar soluciones eficientes (polinomiales en tiempo), robustas y flexibles a problemas NP-completos[**33**] para los cuales hasta ahora sólo se cuenta con soluciones exponenciales en el tamaño de la entrada con el modelo de MT. Es importante resaltar que en muchas de las soluciones basadas en modelos de sistemas complejos, en realidad no resuelven de manera exacta el problema, sino que encuentra una solución muy aproximada a la exacta. Este tipo de soluciones han sido llamadas *soluciones aproximadas*[**34**].

Muchos problemas de gran relevancia práctica, son NP-completos. Aunque hasta ahora no hay ninguna prueba de que los problemas NP-completos no puedan ser resueltos mediante algoritmos basados en MT que corran en tiempo polinomial, hay una fuerte evidencia de que tales algoritmos no existen, y por lo cual una solución polinomial sólo puede encontrarse en otros modelos computacionales como el modelo de automata celular, el modelo de cómputo de sistemas complejos o con el modelo de cómputo cúantico[**35**]. Más aún, tal vez las únicas soluciones polinomiales sean sólo aproximaciones. Entonces el reto es, como garantizar que tales aproximaciones están dentro de una cota establecida. Lo cual es aún muy lejano para el modelo de sistemas complejos.

CAPÍTULO 3

Simulación de sistemas complejos

Resumen. Se diseña e implementa una nueva arquitectura de software (CSDK) para el desarrollo eficiente de simulaciones de SC basadas en el modelo de cómputo descrito. Con el objetivo de ejemplificar y evaluar la arquitectura de software, se utiliza CSDK para simular dos de los modelos que han sido más importantes en el desarrollo del área de los Sistemas Complejos (SC): se simula el modelo "vida" y el modelo de esparcimiento de enfermedades en redes complejas.

1. Entorno de desarrollo para la simulación de sistemas complejos (CSDK)

En este capítulo se presenta una arquitectura de software (CSDK) basada en el modelo de cómputo descrito en el capítulo anterior. La arquitectura está compuesta por un conjunto de bibliotecas orientadas a objetos de componentes reusables[**43**] para la construcción, análisis y control de experimentos de sistemas multi-entidad, de manera que se logre un desarrollo rápido de simulaciones y un análisis eficiente de los diferentes modelos. En el CD que acompaña a esta tesis se pueden consultar todas las bibliotecas del sistema. La arquitectura se presentó en [**G**] (Las referencias con letras se refieren a trabajos presentados por el autor de esta tesis).

1.1. Introducción. Como se ha planteado anteriormente, los modelos computacionales (programas de computadora) han venido a jugar un papel importante como herramienta científica. Las simulaciones en computadora son dispositivos experimentales construidos en software que se han utilizado en comparación a los dispositivos experimentales en física. Los modelos computacionales tienen muchas ventajas sobre los métodos de experimentación tradicionales, pero también tienen muchos problemas:

- La investigación sobre sistemas complejos está basada fuertemente en el uso de herramientas computacionales. Se utiliza mucho tiempo en la construcción de aparatos de software para cada experimento; es equivalente a construir un microscopio para cada nuevo experimento en química o biología. Muchos de tales modelos son similares, sin embargo, éstos se codifican una y otra vez por diferentes investigadores, por lo que se desperdicia mucho tiempo.
- Un problema delicado con los modelos de cómputo personalizados es que el software final tiende a ser muy específico, el código es muy confuso y por lo tanto difícil de entender, después de un tiempo, aún para la persona que lo escribió.
- Estas simulaciones contienen un gran número de suposiciones implícitas, accidentes en la forma en que el código fue escrito, que pueden modificar el comportamiento del modelo que se intenta estudiar, por lo que es difícil evaluar y reproducir los resultados para otros investigadores.

Se pretende construir una arquitectura de software que fomente el uso de herramientas de software bien definidas que se puedan reutilizar para la construcción de una amplia variedad de modelos, además de que esté basada en un modelo de cómputo específico, de manera que todos los parámetros de la simulación sean conocidos. CSDK pretende dar estas herramientas, con el objetivo de que los científicos se centren en sus investigaciones y no en la construcción de las herramientas de software.

Se han desarrollado otros simuladores como StarLogo[**47**][**49**] y Swarm[**48**]. Sin embargo StarLogo no puede manejar defierentes topologias de interconexión y Swarm no esta basado en un modelo de cómputo específico. En la última sección de este capítulo se hace una comparación más detallada entre los simuladores.

1.2. Simulación basada en el formalismo de los sistemas multi-entidad. El formalismo de modelado que CSDK utiliza, está basado en el modelo computacional de los sistemas multi-entidad descrito en el capítulo anterior. Consiste en una colección de entidades que interactúan mediante una topología (ambiente) definida en tiempo discreto. La arquitectura de software no hace suposición alguna acerca del fenómeno en particular que quiere ser implementado; no hay requerimientos específicos del ambiente espacial ni temporal de los fenómenos a modelar, de manera tal que se puedan implementar modelos de distintas áreas como biología, química, economía, física, ecología, etc, por ejemplo, los AC son un caso particular del modelo de SME, se debe poder simular cualquier modelo de AC (cualquier dimensión, cualquier vecindario, condiciones de frontera, etc).

ProcessingCore es la clase de los objetos básicos en el sistema, y representa las entidades que se quieren simular, por ejemplo, partículas de agua y paredes en el modelo de un fluido u hormigas, alimentos y feromonas en el modelo de una colonia de hormigas, etc. El comportamiento de estos objetos se debe implementar en el método *nextState*. Este método será llamado en diferentes instantes de tiempo, dependiendo de la dinámica temporal (especificada en la clase *TemporalDynamics*) que se haya establecido, por ejemplo, en el caso de un AC para modelar fluidos, todos los *ProcessingCore* deben ejecutar su método *nextState* en cada instante de tiempo (síncronamente), otros modelos de cómputo distribuido requieren dinámicas temporales asíncronas, en donde para cada instante de tiempo se ejecuta el método *nexState* de *ProcessingCore*'s tomados aleatoriamente. La ejecución de *nextState* para cada *ProcessingCore* depende del estado actual del ambiente, es decir, el estado actual de los *ProcessingCore* vecinos (el valor de sus registros públicos), éstos son obtenidos a través de una instancia de la clase *Environment*.

Debido a que se está simulando un sistema paralelo y distribuido en una máquina secuencial es necesario simular situaciones de concurrencia. Cuando la función T especifica el conjunto de *ProcessingCore*'s que se ejecutarán en el tiempo t (en paralelo), cada uno de éstos debe leer de manera concurrente los valores de sus vecinos, antes de que se realice cualquier procesamiento. En un sistema secuencial esto no es posible, cada núcleo de procesamiento es ejecutado uno a la vez, por lo que cambia el valor de sus registros antes de que el siguiente *ProcessingCore* se ejecute. Esto, obviamente, no simula adecuadamente el sistema multi-entidad. La solución que se adoptó, fue almacenar los resultados de la ejecución del algoritmo en registros privados y una vez que todos los *ProcessingCore*'s se hayan ejecutado, pasar tales valores a sus registros

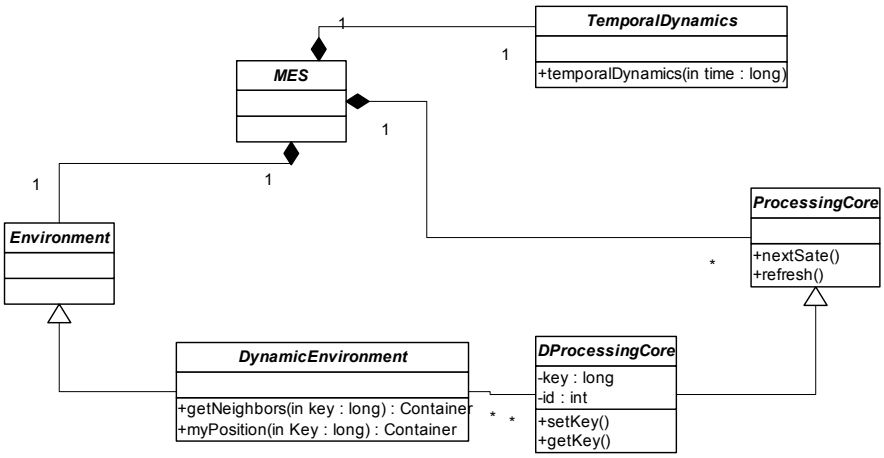


FIGURA 1. Estructura dinámica para la simulación de Sistemas Multi-Entidades

públicos respectivos. Esta actualización se realiza al llamar el método *refresh* de *ProcessingCore*.

De esta manera, para implementar un sistema multi-entidad, se requiere de tres elementos:

(1) Un conjunto de entidades (*ProcessingCore*), en donde se especifica el comportamiento y el estado de cada entidad (registros públicos y privados).
(2) Establecer una dinámica temporal T (*TemporalDynamic*) que especifique que entidades ejecutarán su transición de estado en cada instante de tiempo
(3) Establecer un ambiente en el cual interactuen las entidades (*Environment*), mediante el cual, las entidades podrán conocer el estado actual de sus vecinos.

1.2.1. *Estructura Dinámica.* El objetivo de la arquitectura es poder hacer cada uno de estos tres elementos intercambiables, de forma tal que se pueda modificar uno sin la necesidad de modificar los otros. Por ejemplo, si se tiene un modelo de propagación de enfermedades, que consiste:

- De un conjunto de núcleos de procesamiento que representan entidades con cierto grado de infección,
- Una dinámica temporal síncrona,
- Se quiere estudiar como afecta la estructura del ambiente a la propagación de la enfermedad, entonces, debemos poder modificar el ambiente sin tener que modificar los otros dos componentes.

Para lograr lo anterior, se diseñó la arquitectura mostrada en la Fig. 1, la cual se basa únicamente en clases abstractas (interfaces), con el objetivo de que los clientes de tales interfaces, puedan utilizar de la misma manera a cada una de las subclases que implementan dicha interfaz.

Por ejemplo, el código de la implementación del método *temporalDynamics* de la interfaz *TemporalDynamics* tiene la forma en lenguaje C++ mostrada en el código 3.1, se supone que ya se había inicializado el conjunto de entidades que se quiere simular. Para este método es transparente la instancia particular de la entidad que se está simulando, sólo utiliza el método *nextState,* el cual está definido para todos los *ProcessingCore*'s.

Código 3.1

```
void SynchronousDynamics::temporalDynamics(long int t)
{
	int i;

	for(i=0;i<N;i++){
		E[i]->nextState();
	}

	for(i=0;i<N;i++){
		E[i]->refresh();
	}
}
```

FIGURA 2. Código para la implementación de una dinámica temporal síncrona

De la misma manera, cada instancia de *ProcessingCore* puede obtener a sus vecinos utilizando el método *getNeighbors* de la interfaz *DynamicEnvirinment* independientemente de si el ambiente está implementado como una reja de dos dimensiones o un grafo con arquitectura de mundo pequeño.

1.2.2. *Estructura estática.* Existen arquitecturas de ambientes sobre las cuales se han hecho bastantes estudios. Las más comunes son las latices cuadriculares y latices hexagonales que se utilizan comúnmente en los autómatas celulares. Por otro lado, la flexibilidad que da la arquitectura dinámica para poder interconectar diferentes componentes tiene su costo en términos de la eficiencia en la simulación. Por esta razón, una parte de la arquitectura está diseñada de manera estática, en donde las entidades están fuertemente acopladas con la estructura sobre la cual van a interactuar. En la Fig. 3 se muestra la arquitectura estática para una latice cuadricular, en donde *GridProcessingCore* mantiene en su estado las coordenadas (x, y) de su posición en la latice, por lo que ya no necesita utilizar el método *getNeighborns* como en el caso dinámico. Por otro lado, aún se mantiene la flexibilidad de utilizar diferentes dinámicas temporales y se puede implementar diferentes modelos sobre esta misma arquitectura, por ejemplo, muchas de las implementaciones de los métodos de latice de gas[**44**].

1.3. Bibliotecas de CSDK.

1.3.1. *Bibliotecas de simulación.* Las bibliotecas de CSDK para la simulación de SME se pueden utilizar de dos formas. En la primera, las bibliotecas son utilizadas como un conjunto de clases que los constructores de modelos pueden utilizar mediante instanciación directa. Es muy probable que muchas clases comunes como latices rectangulares, dinámicas temporales síncronas y asíncronas sean utilizadas de esta manera. La segunda forma de utilizar las bibliotecas de CSDK es utilizando el mecanismo de herencia para especializar clases particulares, con el fin de cumplir con las necesidades de un modelo particular.

En la Figura 4 se muestran las bibliotecas del CSDK. La figura muestra como es la estructura de archivos del CSDK y así es como se encontrarán en un sistema de directorios en cualquier plataforma. El número de bibliotecas de CSDK es variable, con el tiempo se irán añadiendo, por ejemplo, nuevas clases que incorporen dinámicas temporales particulares o nuevos ambientes.

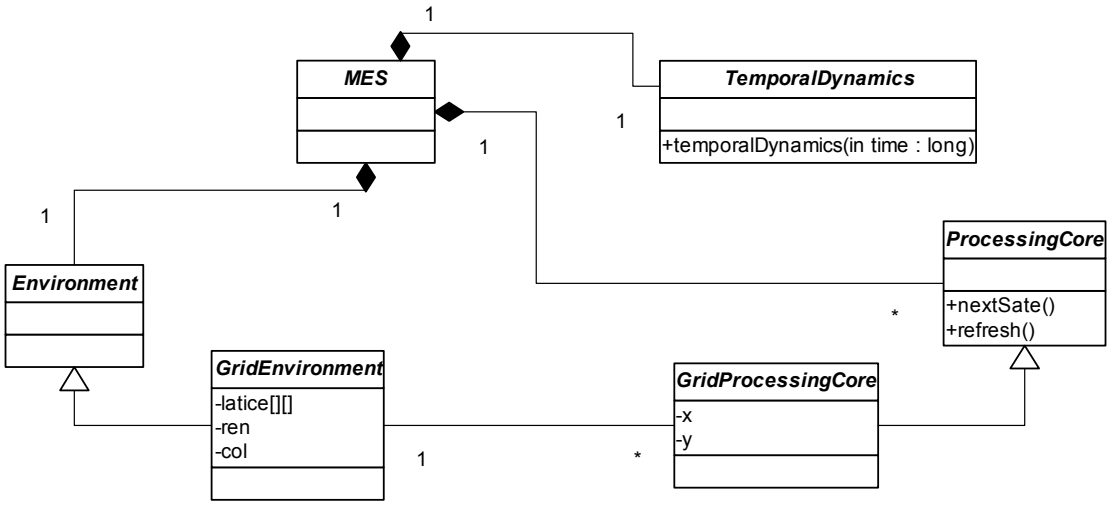


FIGURA 3. Ejemplo de una estructura estática, para el caso de una latice cuadricular

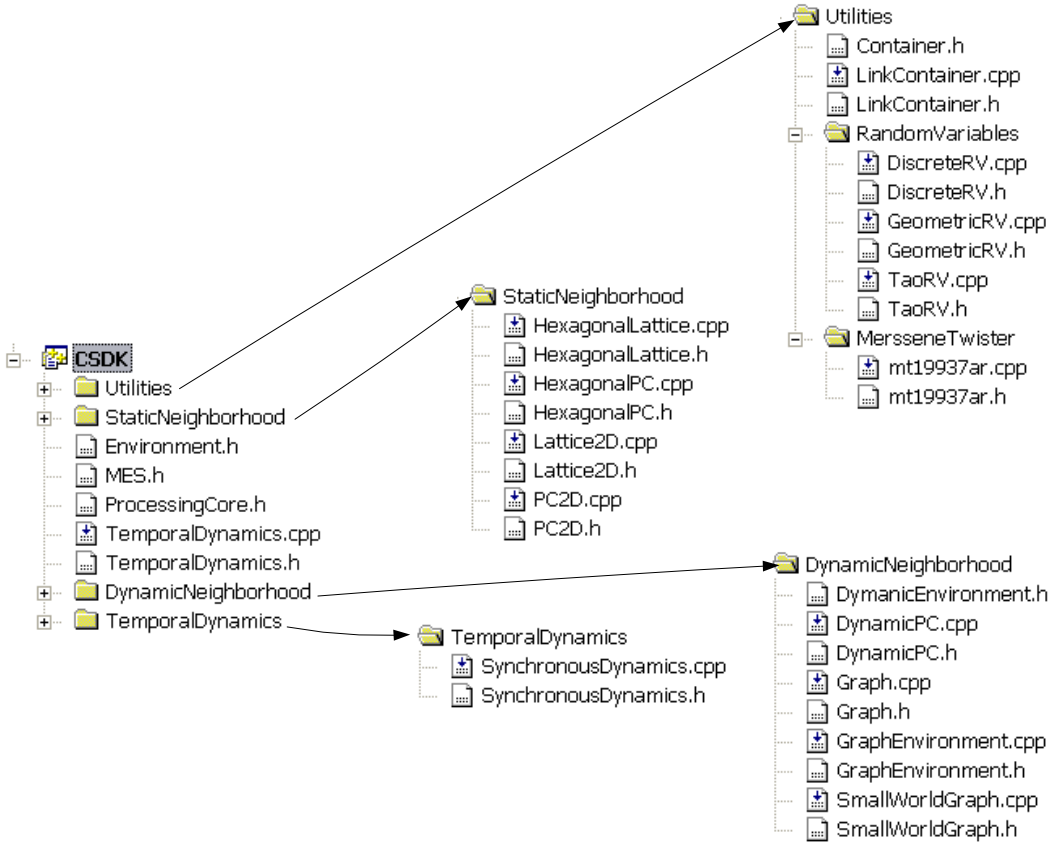


FIGURA 4. Librerías de CSDK

A continuación, se de una breve explicación de algunas de las librerias mostradas en la Figura 4,

- `SynchronousDynamics.h`. Se define la clase *SynchronousDynamics.* Implementa una dinámica temporal síncrona, todas las entidades ejecutan su estado siguiente en cada paso de tiempo.
- `AsynchronousDynamics.h.` Se define la clase *AsynchronousDynamics*. Implementa una dinámica temporal asíncrona. En cada instante de tiempo se escoge aleatoriamente que entidades ejecutan su estado siguiente.
- `HexagonalLattice.h.` Se define la clase *HexagonalLattice.* Implementa un ambiente de latice hexagonal, con contenedores en cada punto.
- `HexagonalPC.h`. Se define la clase *HexagonalPC* que será la clase a partir de la cual heredarán todas las entidades que habiten en una ambiente de latice hexagonal. Esta clase proporciona los servicios para conocer los vecinos en 0, 60, 120, 180, 240 y 300 grados.
- `Lattice2D.h.` Se define la clase *Lattice2D*. Implementa un ambiente de latice en dos dimensiones mediante una matriz de núcleos de procesamiento. Esta clase es muy utilizada para implementar Autómatas Celulares.
- `PC2D.h`. Se define la clase *PC2D* que será la clase a partir de la cual heredarán todas las entidades que habiten en una ambiente de latice en dos dimensiones. Esta clase mantiene variables (x,y) que indican la posición en la latice, a partir de la cual se pueden conocer los vecinos.
- `DynamicEnvironment.h`. Define la interfaz *DynamicEnvironmet*, la cual tienen que implementar todos los ambientes en los que puedan habitar todo tipo de núcleos de procesamiento. Se pueden utilizar los ambientes estáticos, para implementar ambientes dinámicos.
- `Container.h.` Define la clase *Container*, que representa un contenedor de entidades. Además, especifica la interfaz del patrón Iterador[**45**] para recorrer los elementos que se encuentran en el contenedor.
- `ProcessingCore.h.` Define la interfaz *ProcessingCore*. Esta interfaz debe de ser implementada por todas las entidades que se quieran utilizar.
- `TemporalDynamics.h.` Define la interfaz *TemporalDynamics*. Esta interfaz debe de ser implementada por todas las clases que quieran implementar una dinámica temporal.

1.3.2. *Bibliotecas de análisis.* El análisis de los modelos de sistemas multientidad simulados se hace en dos formas. En la primera, el análisis se hace en tiempo de ejecución es decir, el análisis se hace junto con la ejecución del programa. Esta forma de análisis es apropiada, por ejemplo, cuando se estén utilizando Sistemas Multientidad para realizar tareas computacionales específicas, el analizador debe detectar cuando se resolvió la tarea. La clase Observer es la encargada de tal tarea, la cual tiene acceso a los registros de todos los ProcessingCore's. Hay observadores especializados en mostrar un representación gráfica del sistema.

La segunda forma de análisis es a través del estudio de las series de tiempo generadas a partir del sistema multi-entidad. Este análisis se hace después de la ejecución de la simulación, por lo que CSDK sólo debe encargarse de generar las series de tiempo que se quieran analizar. Después de esto, se puede utilizar cualquiera de las herramientas para analizar, modelar, graficar y predecir series de tiempo con que se cuenta en la actualidad.

Por otro lado, también se deben desarrollar técnicas de análisis de series de tiempo que utilicen el conocimiento que se tiene del sistema multientidad a partir del cual las series fueron generadas, por ejemplo, las series de tiempo tomadas de cada celda de un autómata celular de una dimensión guardan una relación de vecindad que no se puede expresar en las técnicas generales para el análisis de series de tiempo. Sería interesante encontrar analizadores que encontraran esta relación de vecindad automáticamente.

2. Ejemplos del uso de CSDK.

A continuación, con el objetivo de ejemplificar y evaluar la arquitectura de software, se utiliza CSDK para simular los modelos que han sido más importantes en el desarrollo del área de los SC. Primero se describe la implementación del sistema "vida" una autómata celular inventado por John Conway para mostrar la formación de estructuras a diferentes niveles jerárquicos (estructuras emergentes). Y el modelo de esparcimiento de enfermedades descrito en el capítulo anterior.

2.1. Autómata celular "Vida". "Vida" es un autómata celular inventado por John Conway en 1960. Es un AC de dos dimensiones con estados binarios, en donde el vecindario de cada celda está definido según la vecindad de Moore. Las reglas de transición son muy simples:

- Si el estado actual es 1; entonces el estado siguiente será 1, si y sólo si 2 ó 3 de los vecinos están en el estado 1, de otra manera el siguiente estado será 0.
- Si el estado actual es 0; entonces el estado siguiente será 1, si y sólo si 3 de los vecinos están en el estado 1, de otra manera el siguiente estado será 0.

"Vida" es bien conocido, porque estas simples reglas de transición llevan a patrones complicados e interesantes en el espacio celular. Por ejemplo, es muy fácil construir configuraciones iniciales que produzcan estructuras simples, localizadas y con movimiento, llamadas "deslizadores", las cuales fueron utilizados para demostrar la capacidad de cómputo universal de este AC[**46**]. Se implementará este AC como ejemplo de cómo se emplea el CSDK, utilizando la forma estática.

2.1.1. *Implementación.* Para implementar "Vida", lo primero que se debe hacer es definir el archivo LifePC.h en donde se declara la clase *LifePC* (Código 3.2). Esta clase debe heredar de la clase *PC2D* e implementar los métodos *nextState* y *refresh* con las reglas mencionadas anteriormente. Se declara la variable *bin,* utilizada para representar el estado, y *newbin* utilizada para almacenar temporalmente el estado siguiente calculado, hasta que todos los vecinos hayan leído el valor de *bin* (depende de la dinámica temporal establecida).

La implementación de esta clase se muestra en el código de la figura 6. Como la clase *LifePC* hereda de *PC2D* entonces tiene una referencia a una latice de dos dimensiones cuadricular. la cual es utilizada para obtener los vecinos de cada celda. En este ejemplo, se utilizó el método *moore* aprovechando los beneficios de la estructura estática. Después solo se prgraman las reglas de el autómata "vida".

Por último, se tiene que interconectar cada uno de los componentes que se van a utilizar para conformar un SME, lo cual se lleva a cabo en una clase *Observer*. Además, ésta será la encargada de generar los archivos de series de tiempo requeridos y/o de la graficación del sistema. El SME para este caso, se conformará de un ambiente en dos dimensiones cuadricular H, un conjunto de entidades que implementan las reglas del

Código 3.2

```
#include "..//ComplexSystemV2.0//StaticNeighborhood//PC2D.h"
#include "..//ComplexSystemV2.0//StaticNeighborhood//Lattice2D.h"

class LifePC : public PC2D{
public:
      void nextState(); //Se computa el estado siguiente
      void refresh(); //Se transmite la información hacia el nuevo
      //estado
public:
      unsigned int bin; //Se encuentra el estado actual
      unsigned int newbin; //Se encuentra el estado siguinte
};
```

FIGURA 5. Archivo de cabecera LifePC.h

Código 3.3

```
void LifePC::nextState()
{
      int i=0;
      int black, white;
      LifePC *fp;
      ProcessingCore *p;

      black = white = 0;

      ProcessingCore **moore;
      moore = new ProcessingCore*[8];
      lattice->moore(&moore,x,y);

      for(i=0;i<8;i++){
            fp = (LifePC*) moore[i];
            if(fp->bin == 0)
                  white++;
            else
                  black++;
      }
      if(bin == 1){
            if(black == 2 || black == 3)
                  newbin = 1;
            else
                  newbin = 0;
      }

      if(bin == 0){
            if(black == 3)
                  newbin = 1;
            else
                  newbin = 0;
      }

}
```

FIGURA 6. Implementación del método *nextState* de la clase *LifePC*

modelo "Vida" E y una dinámica temporal síncrona tD, cada uno declarado en la clase *LifeObserver* ver código de la figura 7. Además la clase también declara el método *step* para ejecutar un paso de la simulación del sistema.

La principal actividad de la clase LifeObserver, además de cualquier tarea de análisis que se quiera agregar, es la de interconectar e inicializar cada uno de los componentes mencionados anteriormente (código de figura 8).

Código 3.4

```
class LifeObserver{
private:
      CDC *dc;
      int alto,ancho;
public:
      long int time;
private:
      int xren,yren;
public:
      long int N; //Número de entidades (NP's) en el sistema
      LifePC **E; //Puntero hacia los NP's
      ProcessingCore **eP;
      Lattice2D *H; //Ambiente que define la función de vecindad
      SynchronousDynamics *tD; //Se define la dinámica
      //temporal de las entidades

      LifeObserver(CDC *window, int alto, int ancho);
      void simulate(int steps);
      void step();
      void graphActual();
      void setGuilder(int x,int y);
};
```

FIGURA 7. Declaración de la clase *LifeObserver.h*

En la figura 9 se muestra la simulación del sistema "vida" para la condición inicial dada. Se resalta un arreglo de celdas conocido como "deslizador", esta estructura sirvió para mostrar la formación de estructuras a un nivel jerárquico mayor. En la figura se observa como el deslizador sigue una trayectoria bien definida dentro del ambiente. Dichas estructuras fueron utilizadas (mediante colisiones entre ellas) para mostrar la capacidad de computo universal del sistema "vida".

Código 3.5

```
LifeObserver::LifeObserver(CDC *window, int alto, int ancho)
{
	int i,j, w = 0;
	int x,y;float aux;
	dc = window;
	this->alto= alto;
	this->ancho = ancho;
	//Se inicializa el tamaño de la cuadrícula y el
	//número de Nucleos de Procesamiento que se vana simular
	xren = yren = 50;
	N = xren * yren;
	//Se inicializa el tiempo a 0
	time = 0;
	//Se crean los NP que implementan las reglas de vida
	E = new LifePC*[xren];
	for(i=0;i<xren;i++)
		E[i] = new LifePC[yren];
	//Se declara un arreglo de apuntadores a NP para cada
	//una de las entidades
	eP = new ProcessingCore*[N];
	//Se crea una latice cuadricular de dos dimensiones
	H = new Lattice2D(xren,yren);
	//Se inicializan las partículas de procesamiento
	for(i=0;i<yren;i++){
		for(j=0;j<xren;j++){
			//Se le indica a cada NP el ambiente en el que viven
			E[i][j].setEnv(H,i,j);
			//Se inserta el NP en el ambiente
			H->lattice[i][j] = &E[i][j];
			//Se inicializa su estado a 0
			aux = rand();
			aux = aux/RAND_MAX;
			if(aux < 0.95)
				E[i][j].bin = 0;
			else
				E[i][j].bin = 1;
			//Se gurada la refrencia de cada NP en el
			//arreglo de apuntadores
			eP[w] = &E[i][j];
			w++;
		}
	}
	//Se construye un conjunto de gulders en las
	//coordenadas especificadas
	this->setGuilder(7,7);
	this->setGuilder(20,30);
	this->setGuilder(40,7);
	this->setGuilder(30,10);
	//Se crea una dinámica temporal, y se le pasan
	//los NP que se van a simular.
	tD = new SynchronousDynamics(eP,N);
}
```

FIGURA 8. Interconexión e inicialización de cada uno de los componentes que forman el SME "Vida"

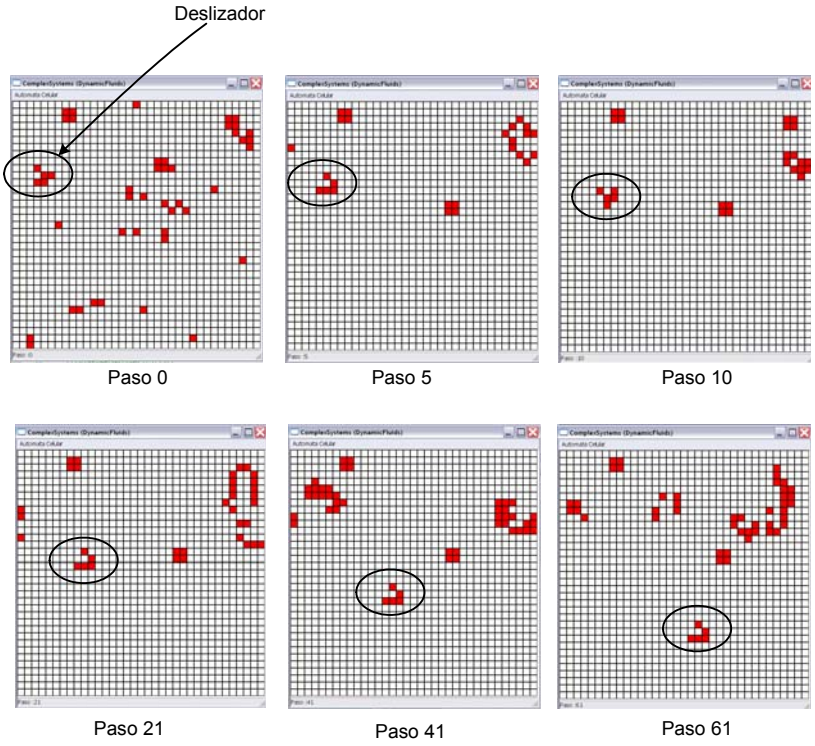


FIGURA 9. Simulación del sistema "vida". En el óvalo se resalta un arreglo de cinco celdas conocido como "deslizador". Los deslizadors fueron utilizados como ejemplos de estructuras de un nivel jerárquico mayor. En la simulación se observa como el deslizador se mueve a través del ambiente.

Este ejemplo mostró la capacidad de CSDK para simular uno de los modelos más importantes en el área de los SC. Al mismo tiempo se describió como es posible la implementación de cualquier modelo planteado en un Autómata Celular.

2.2. Propagación de enfermedades en redes de "Mundo Pequeño". A continuación se presenta un modelo simplificado de propagación de enfermedades en sistemas representados con un grafo. Ahora, sólo se utiliza este modelo como ejemplo de un sistema con estructura dinámica. En el próximo capítulo se estudia en detalle el comportamiento dinámico de sistemas discretos en redes de interacción que tienen características estadísticas específicas.

En el tiempo $t = 0$ se introduce un individuo infeccioso a la población. Los individuos infectados, se eliminan permanentemente del sistema por inmunidad o por muerte después del periodo de enfermedad, el cual dura una unidad de tiempo. Durante este periodo de tiempo, un individuo puede infectar a cada uno de sus vecinos saludables con probabilidad r. Durante el paso del tiempo la enfermedad se propaga a través de la red, infectando a toda la población o se detiene habiendo infectado sólo una fracción de ésta.

2.2.1. *Implementación.* Para implementar este modelo, primero se crea el archivo DiseasePC.h en donde se declara la clase *DiseasePC* Código de la figura 10. Esta clase debe heredar de la clase *DynamicPC* e implementar los métodos *nextState* y *refresh* con las reglas mencionadas anteriormente. Se declara la variable *sick*, utilizada para representar su estado: saludable (0), infeccioso (1) o inmune (-1).

Código 3.6

```
class DiseasePC: public DynamicPC{
public:
      static GeometricRV gRV;
public:
      void nextState(); //Se computa el estado siguiente
      void refresh(); //Se transmite la información hacia el nuevo
      //estado
public:
      DiseasePC(){}
      DiseasePC(float p);
      void setP(float p);
public:
    int sick;
      int newSick;
};
```

FIGURA 10. Archivo de cabecera *DiseasePC.h*

La implementación de esta clase se muestra en código de la figura 11. Como la clase *DiseasePC* hereda de *DynamicPC* entonces tiene una referencia a un objeto de la clase *DynamicEnvironment*. Esta referencia es utilizada para obtener los vecinos de cada entidad. El código de la clase *DiseasePC* es independiente de la clase ambiente que se utilice. Además de que utiliza un objeto de la clase *GeometricRV* para representar una variable aleatoria geométrica con parámetro r. Ésta, como otras clases de utilería, puede consultarse en el manual de usuario del sistema.

Por último, se tienen que interconectar cada uno de los componentes que se van a utilizar para formar un SME en una sola clase *Observer*. Generalmente esta clase, además, será la encargada de generar los archivos de series de tiempo requeridos y/o de la graficación del sistema. Nuestro SME en este caso se formará de

- un ambiente especificado por un grafo H, el cual está implementado en la clase *GraphEnvironmen*. Dicha clase es inicializada con el grafo que se va a utilizar,

Código 3.7

```
void DiseasePC::nextState()
{
	int i,n;
	LinkContainer *neighborn;
	DiseasePC *dp;	ProcessingCore *p;

	if(sick == 1 || sick == -1){
		newSick = -1;
		return;
	}

	neighborn = new LinkContainer();
	n = dyEnv->getNeighbors(neighborn,this->key);

	neighborn->initialize();
	for(i=0;i<n;i++){
		neighborn->iterate(&p);
		dp = (DiseasePC *)p;
		if(dp->sick == 1){
			if(gRV.random()==1){
				newSick = 1;
				delete neighborn;
				return;
			}
		}
	}
	newSick = 0;
}
```

FIGURA 11. Implementación de las reglas del modelo de propagación de enfermedades

el cual en este caso está basado en el modelo de Watts y está implementado en la clase *SmallWorld* la cual hereda de la clase *Graph*,
- un conjunto de núcleos de procesamiento que implementan las reglas del modelo E,
- una dinámica temporal síncrona tD,

cada uno declarados en la clase *DiseaseObserver* Código de la figura 12. Además, la clase también declara el método *printRateInfectiusness* para generar la serie de tiempo del porcentaje de la población infectada.

La principal actividad de la clase *DiseaseObserver* es la de interconectar e inicializar cada uno de los componentes mencionados anteriormente. Además de generar la serie de tiempo que mide el porcentaje de la población que ha sido infectada hasta el tiempo t.

En la figura 14 se muestra el porcentaje de individuos infectados con respecto al tiempo para un sistema con $N = 1000$ individuos. En a) se estudia como varia el porcentaje de individuos infectados cuando $r = 0.25$ y para diferentes valores del parámetro p en el modelo de Watts [**51**] (se describe en detalle en el siguiente capítulo). Observese que cuando p varia de 0 (red regular) a 0.01 (red de mundo pequeño), aun cuando el cambio en p es pequeño, los cambios en el porcentaje de individios infectados es muy grande, esto se debe a que solo se necesitan aleatorizar muy pocas aristas (1%) para

Código 3.8

```
class DiseaseObserver{
public:
      long int time;
      float p;
public:
      long int N; //Número de entidades (NP's) en el sistema
      DiseasePC *E; //Puntero hacia los NP's
      ProcessingCore **eP;
      //Ambiente que define la función de vecindad
      GraphEnvironment *H;
      //Se define la dinámica temporal de las entidades
      SynchronousDynamics *tD;
      SmallWorldGraph *sw;
      RandomizeGraph *rg;
      DiseaseObserver(float p);
      void simulate(int steps);
      void step();
      void printActual();
      void printRateInfectiusness();
private:
      ofstream out;
};
```

FIGURA 12. Archivo de cabecera *DiseaseObserver.h*

Código 3.9

```
DiseaseObserver::DiseaseObserver(float p)
{
      int i;

      N = 1000;
      time = 0;
      E = new DiseasePC[N];
      eP = new ProcessingCore*[N];

      //rg = new RandomizeGraph(N,.02);
      sw = new SmallWorldGraph(N,10,0.07);

      H = new GraphEnvironment(sw);

      //Se inicializan las partículas de procesamiento
      E[0].setP(p);
      for(i=0;i<N;i++){
            E[i].setEnv(H,i);
            E[i].setP(p);
            H->insertPC(&E[i],E[i].key);
            E[i].sick = 0;
            eP[i]= &E[i];
      }

      E[0].sick = 1;
      E[N/2].sick = 1;


      tD = new SynchronousDynamics(eP,N);
      out.open("Sickness.txt");
}
```

FIGURA 13. Interconexión e inicialización de cada uno de los componentes que integran el SME de propagación de enfermedades

hacer a la red un mundo pequeño. Cuando r es lo suficientemente grande para que se infecte a toda la población independientemente de su topologia, entonces se estudia como la topología afecta el tiempo en que tarda la enfermedad en propagarse totalmente, ver figura 14 b). Se observa un resultado análogo al caso anterior.

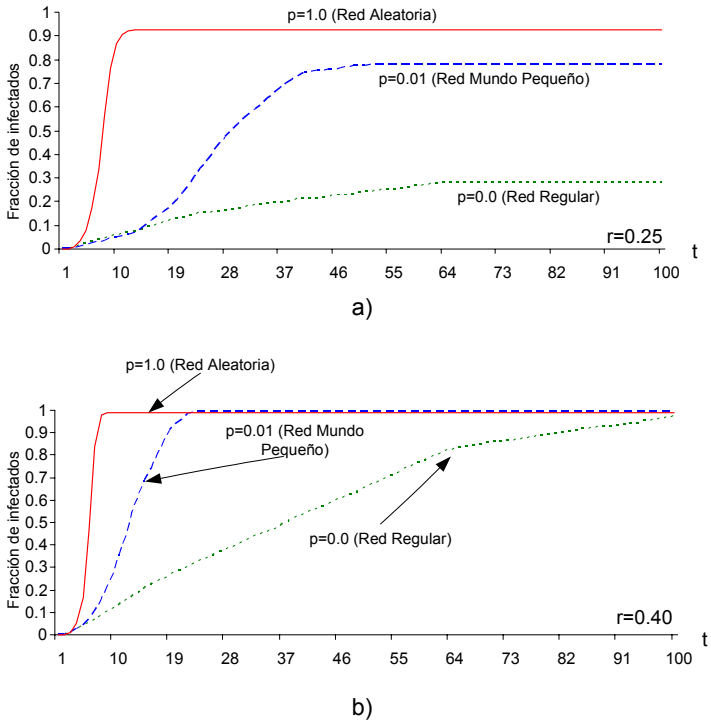


FIGURA 14. Simulación del modelo de propagación de enfermedades con numero de individuos $N = 1000$ para diferentes topologias de interacción. a) La probabilidad de infección r no es suficiente para infectar a toda la población ($r = 0.25$). Se estudia el porcentaje de la población infectada para diferentes valores del parámetro p. b) La probabilidad de infección r infecta a toda la población independiente de la topologia ($r = 0.40$). Se observa la velocidad con la que se infecta la población para diferentes valores del parámetro p.

El modelo de esparcimiento de enfermedades en redes complejas fue el modelo con el que se empezó a mostrar la importancia de la topología de interacción. Aquí se reprodujeron los resultados haciendo uso de CSDK, mostrando así la capacidad de éste para simular sistemas con una topología de interacción cualquiera.

TABLA 1. Comparación entre simuladores de Sistemas Complejos: Swarm, StarLogo y CSDK

Propiedad	StarLogo	CSDK	Swarm
GradosLibertad	Si	Si	Si
No-Linealidad	Si	Si	Si
Topologia de Interacción	Cuadricular	Cualquier tipo	Cualquier tipo
Aleatoriedad	Si	Si	Si
Tiempo	Discreto	Discreto	Discreto/Continuo
Heterogeneidad	No	Si	Si
Adaptación	No	Si	Si
DinámicaTemporal	Sincrona	Cualquier tipo	Cualquier tipo

3. Comparación entre ambientes para la simulación de sistemas complejos

Durante la investigación en los sistemas complejos, se han desarrollado diferentes herramientas computacionales que ayuden a una mejor comprensión de los mismos. Entre las herramientas más importantes dentro del área estan: Swarm[**48**] y StarLogo[**47**][**49**]. Las cuales utilizaremos para medir la capacidad de CSDK. La comparación de los entornos de simulación se realiza en base a los siguientes aspectos: capacidad de modelado, rendimiento del sistema y la facilidad para el desarrollo y estudio de los SC.

3.1. Capacidad de Modelado. Existe un viejo proverbio que dice: "Dale un martillo a una persona, y todas las cosas en el mundo le parecerán un clavo". Ciertamente, la forma en que nosotros vemos al mundo esta profundamente influenciado por las herramientas que utilizamos. En esta tesis se propone ver a los fenómenos complejos como sistemas computacionales. Las herramientas computacionales que se analizan aquí están diseñadas especialmente para ayudar a la gente a crear modelos de sistemas descentralizados esto es, sistemas en los que el comportamiento surge de la interacción entre muchos objetos individuales. En el capítulo uno se identificaron las propiedades que afectan directamente el comportamiento de los SC, y por tanto las herramientas de modelado (simuladores) deben expresar tales propiedades de la manera más sencilla posible. A continuación se enumeran algunas de las propiedades descritas en el capítulo uno (el listado no es exhaustivo):

- Grados de libertad
- ¿Se pueden implementar interacciones No-Lineales?
- ¿Se pueden representar diferentes arquitecturas de interconexión?
- ¿Se pueden implementar interacciones estocásticas?
- El tiempo en la simulación ¿es continuo o discreto?
- ¿Se pueden implementar elementos heterogéneos?
- ¿Se pueden implementar dinámicas cambiantes?
- La dinámica ¿es sincrona, asíncrona u otras?

En la Tabla 1, se muestra la comparación entre lo tres ambientes de simulación. Tanto CSDK como Swarm tienen una capacidad de modelado similar.

3.2. Rendimiento del Sistema. De nada sirve tener una herramienta computacional en la que se pueda modelar cualquier fenómeno de interés si el tiempo que va a tardar en ejecutarse es demasiado, al grado de ser impráctico. Por lo tanto, el rendimiento es uno de los parámetros más importantes que se debe tener en cuenta para comparar los distintos sistemas.

3.3. Facilidad de desarrollo y estudio de SC. ¿Cuánta facilidad brinda cada entorno para la construcción de modelos y el estudio de los mismos una vez que se conoce el sistema? Aunque la comparación en este punto es subjetiva, se puede decir que CSDK tiene la ventaja de estar basado en la descripción formal de un modelo de cómputo universal, lo cual, en primera instancia sirve como guía para la construcción de los modelos, pero más importante, sirve de marco teórico para ayudar a hacer generalizaciones: se puede hablar de propiedades abstractas en los programas y no en propiedades de modelos particulares.

CAPÍTULO 4

Redes complejas

Resumen. La estructura o topología de interacción entre las entidades de los diferentes SC es una característica esencial, porque ésta determina fuertemente el comportamiento global del sistema. En este capítulo se hace una revisión general de los principales avances que se han hecho en esta línea, en particular se estudia cómo afectan las características estadísticas globales de las redes reales en los diferentes procesos dinámicos que tienen lugar en ellas. Después, se describe un nuevo algoritmo de optimización, denominado optimización extrema distribuida (OED). Esté es una modificación sustancial del algoritmo de optimización extrema (OE) que contempla la topología de interacción. OE resulta sólo como el caso particular en que se tiene una red completamente conexa. Se utiliza el algoritmo para resolver el problema de coloreado de grafos y se compara con el algoritmo de optimización extrema tradicional.

Las redes complejas describen un amplio rango de sistemas en la naturaleza y la sociedad. Ejemplos citados frecuentemente, incluyen la célula: una red de químicos enlazados por reacciones químicas y el Internet: una red de ruteadores y computadores conectados por enlaces físicos, entre otros. Tradicionalmente estos sistemas han sido modelados como grafos aleatorios, pero con el tiempo se ha ido reconociendo que la topología y la evolución de las redes reales están gobernadas por robustos principios organizacionales. En este capítulo se revisan los avances recientes en el campo de las redes complejas, enfocándose en las propiedades estadísticas de la topología de la red, su dinámica en base a estas propiedades y las diferentes formas en que estas redes pueden crecer o evolucionar.

1. Introducción

Uno de los aspectos más importantes que se necesita para el entendimiento de los sistemas complejos es el estudio de la estructura de las redes de interacción existentes entre los elementos que conforman el sistema, y cómo esta estructura afecta directamente a la dinámica o capacidad funcional del mismo.

En este capítulo se hace una revisión de las características estructurales globales que existen en diferentes redes complejas y algunas de las consecuencias que tienen éstas en propiedades globales específicas como la robustez ante fallas aleatorias, el flujo de información, entre otras. En capítulos posteriores se estudian características estructurales locales. Primero se hace una introducción a la teoría de grafos, herramienta matemática utilizada para la descripción de las redes complejas.

1.1. Introducción a la teoría de grafos. La teoría de grafos se inició en 1736 en un artículo publicado por el matemático suizo Leonhard Euler (1707-1783). La idea principal subyacente al estudio se desarrolla a partir de una situación, ahora popular, conocida como los siete puentes de Königsberg.

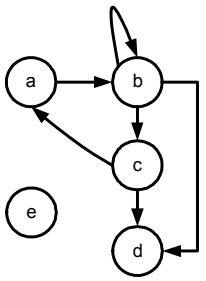


FIGURA 1. Grafo dirigido

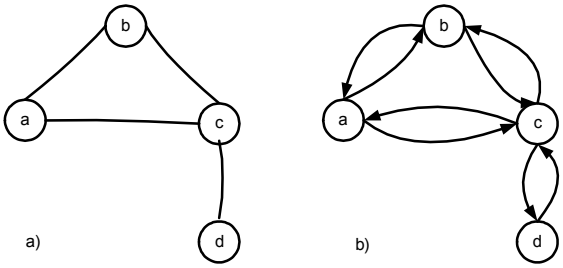


FIGURA 2. a) Grafo no dirigido. b) Grafo dirigido equivalente al grafo no dirigido del inciso b).

1.1.1. *Aspectos generales.* **Definición.** Sea V un conjunto no vacio y $A \subseteq V \times V$. Entonces el par (V, A) se denomina grafo dirigido en V, o *digrafo*, donde V es el conjunto de vértices o nodos y A es un conjunto de aristas. Se escribe $G = (V, A)$ para denotar dicho grafo.

La Figura 1 muestra una grafo dirigido en $V = \{a, b, c, d, e\}$ con aristas $A = \{(a, b), (b, b), (b, d), (b, c), (c, d), (c, a)\}$. La dirección de la arista se indica colocando una flecha con dirección en ella. Para cualquier arista como (b, c), se dice que la arista es incidente en los vértices b y c; b es adyacente a c. Además b se denomina origen o fuente de la arista, mientras que c es el destino o término. La arista (b, b) es un ejemplo de lazo; el vértice e no es incidente con ninguna arista y se denomina vértice aislado.

Cuando no importa la dirección de una arista, el grafo se denomina *no dirigido* Figura 2 a). Este grafo es una manera más compacta de escribir el grafo de la Figura 2 b). En un grafo no dirigido hay aristas no dirigidas como $\{a, b\}, \{b, c\}, \{a, c\}, \{c, d\}$ de la Figura 4.2 a). Una arista como $\{a, b\}$ significa $\{(a, b), (b, a)\}$.

En general, cuando no se especifica si un grafo G es o no dirigido, se supone que es dirigido. Si no contiene lazos, se denomina grafo sin lazos.

Definición. Sea $G = (V, A)$ un grafo no dirigido. Para, $x, y \in V$, se dice que hay un camino de x a y, si existe una sucesión no vacia finita de aristas distintas de A, como $\{x, x_1\}\{x_1, x_2\}\{x_2, x_3\}...\{x_{n-1}, x_n\}\{x_n, y\}$. Cuando $x = y$, el camino se denomina *ciclo*. El número de aristas de un camino o ciclo se denomina longitud del camino.

En la Figura 3, $\{a, b\}\{b, c\}\{c, f\}$ proporcionan un camino de a a f. También lo proporciona la sucesión $\{a, b\}\{b, c\}\{c, e\}\{e, d\}\{d, c\}\{c, f\}$. Sin embargo en el último camino se ha pasado dos veces por el vértice c. Cuando un camino entre dos vértices sólo pasa una vez por cualquiera de ellos, el camino se denomina *simple*. Esto se aplica

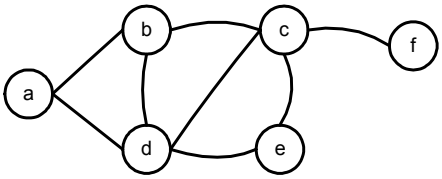


FIGURA 3. Grafo de ejemplo

también a un ciclo, excepto que el primer y último vértice es el mismo. Para un grafo dirigido, se utilizan los términos caminos dirigidos y ciclos dirigidos.

Definición. Sea $G = (V, A)$ un grafo no dirigido. G se denomina *conexo* si existe un camino entre cualesquiera dos vértices distintos de G. Sea G un grafo dirigido. Su grafo no dirigido asociado es el obtenido de G, ignorando la dirección de las aristas. G se considera conexo si su grafo asociado es conexo. Un grafo que no es conexo se le denomina *no conexo*.

1.1.2. *Subgrafos, e isomorfismo de grafos.* **Definición**. Si $G = (V, A)$ es un grafo dirigido o no dirigido, $G' = (V', A')$ se denomina subgrafo de G si $\phi \neq V' \subseteq V$ y $A' \subseteq A$, donde cada arista de A' es incidente con vértices de $V\prime$.

Definición. Sea $G_1 = (V_1, A_1)$ y $G_2 = (V_2, A_2)$ dos grafos dirigidos. Una función $f : V_1 \to V_2$ se denomina isomorfismo de grafos si: a) f es uno a uno y suprayectiva, y b) para todo $a, b \in V_1$, $(a, b) \in A_1$ si, y sólo si $(f(a), f(b)) \in V_2$. Cuando existe dicha función, G_1 y G_2 se denominan grafos *isomorfos*.

La correspondencia de vértices de un isomorfismo de grafos mantiene las adyacencias. Es decir, se mantiene la estructura de los grafos. Pero el problema de encontrar tal correspondencia es NP-completo[**34**]. A pesar de esto, se observa que si un isomorfismo preserva adyacencias, entonces preserva subestructuras de grafos como caminos y ciclos estas observaciones ayudan al desarrollo de algoritmos más eficientes para la búsqueda de un isomorfismo.

1.2. Caracterización global de las redes complejas. Los conceptos y medidas que más se han utilizado para describir la estructura global de las redes son las siguientes[**32**]:

Distancia de camino característica y mundos pequeños. El concepto de mundo pequeño, describe en simples términos el hecho de que a pesar de su frecuente gran tamaño, en la mayoría de las redes hay un camino pequeño entre cualquier par de nodos. La distancia de camino característica de una red se define como el promedio de las longitudes de camino más cortos existentes entre todos los pares de nodos de una red.

Aunque intrigante, el concepto de mundos pequeños, no es una indicación de un principio organizativo en particular. Ciertamente, como Erdos y Renyi han demostrado[**50**], la longitud típica entre cualquier par de nodos en un grafo aleatorio se escala como el logaritmo del número de nodos. De esta forma los grafos aleatorios son mundos pequeños también.

Agrupamiento. Una propiedad común de las redes sociales es la formación de grupos de gente, representando círculos de amigos donde cada miembro se conoce uno a otro. Esta tendencia al agrupamiento es cuantificada por el coeficiente de agrupamiento[**51**], éste se calcula para cada nodo como

$$C_i = \frac{2E_i}{k_i(k_i - 1)}$$

donde k_i es el número de vecinos del nodo i y E_i es la cantidad de aristas que existen en en el subgrafo formado por el nodo y los vecinos. El coeficiente de agrupamiento es 1 cuando E_i vale $k_i(k_i - 1)/2$, que es el número máximo de aristas que pueden existir en el cluster de un nodo con k_i vecinos. Este indice también se puede interpretar como la probabilidad de que exista un enlace entre los nodos b y c dado que existen enlaces entre a y b y a y c. Para el caso de los grafos aleatorios el coeficiente de agrupamiento es $C = p$. Sin embargo, en la mayoría, si no en todas las redes verdaderas, el coeficiente de agrupamiento es típicamente más grande que el que se encuentra en un grafo aleatorio.

Distribución del grado. No todos los nodos en la red tienen el mismo número de aristas (mismo grado de nodo). la propagación en el grado de los nodos esta caracterizado por una función de distribución $P(k)$, que da la probabilidad de que un nodo seleccionado aleatoriamente tenga exactamente k aristas. En las redes aleatorias, la mayoría de los nodos tiene aproximadamente el mismo grado, cercano al grado promedio de la red $\langle k \rangle$. La distribución del grado de una red aleatoria es una distribución de Poisson con pico en $\langle k \rangle$. Uno de los descubrimientos más interesantes en el desarrollo de las redes complejas es que en la mayoría de las grandes redes, la distribución se desvía significativamente de la distribución de Poisson. La WWW, Internet y la red metabólica siguen una distribución de ley de potencia $P(k) \propto k^{-\lambda}$. Tales redes son conocidas como redes libres de escala.

1.3. Ejemplos de redes complejas. El objetivo final en el estudio de las redes complejas es el de entender los sistemas reales, desde las redes de comunicaciones, hasta las redes ecológicas. Existen bases de datos disponibles de diferentes disciplinas, a partir de las cuales se calcularon las propiedades antes mencionadas: distancia de camino característica, coeficiente de agrupamiento, y la distribución del grado. Existen otras medidas que se describirán en secciones posteriores.

Para la identificación de principios organizacionales dentro de una red, se han comparado cada una de las propiedades medidas en la red real, con los resultados obtenidos para una red aleatoria con el mismo número de nodos y aristas que la red real. En [4.1] se resumen los resultados obtenidos para los diferentes tipos de redes. A continuación se describe el modelo de red para sistemas de diferente índole.

WWW. La WWW representa la red, para la cual se cuenta con mayor información topológica. Los nodos de la red son los documentos (páginas web) y las arístas son las hiper-ligas (URL) que apuntan de un documento a otro. Debido a que la red es dirigida está caracterizada por dos distribuciones de grado: la distribución de aristas de salida $P_{out}(k)$, y la distribución de aristas de entrada $P_{in}(k)$.

Internet. La Internet es una red de enlaces físicos entre computadoras y otros dispositivos de comunicaciones. La topología de Internet se estudia en dos niveles diferentes. En el nivel de ruteador, los nodos son los ruteadores, y las aristas son los enlaces físicos entre ellos. A nivel de sistema autónomo, cada dominio, compuesto por cientos de ruteadores y computadoras es representado por un sólo nodo, y se establece una arista entre dos dominios si existe al menos una ruta que los conecte.

Red de colaboración de actores de películas. Esta red fue tomada de la Base de datos de películas en internet, la cual contiene todas las películas y su reparto desde

1940. En esta red los nodos son los actores. Dos nodos tienen una arista común si los dos actores han trabajado juntos en una película. Ésta es una red que se esta expandiendo continuamente, con 225226 nodos en 1998[**51**].

Red de contactos sexuales humanos. Muchas de las enfermedades transmitidas sexualmente, incluyendo el SIDA, se esparcen en una red de relaciones sexuales. En [**52**], Lijeros construyó una red a partir de las relaciones sexuales de 2810 individuos, basada en una extensiva encuesta realizada en Suecia en 1996. Debido a que las aristas en esta red son dinámicas y duran generalmente poco tiempo, ellos analizaron la distribución de parejas durante un año.

Redes celulares. En [**53**], Jeong et al. estudiaron el metabolismo de 43 organismos representando los tres dominios de la vida, reconstruyéndolos en redes donde los nodos son los substratos (como ATP, ADP, H2O) y las aristas representan reacciones químicas predominantes directas en las que los substratos pueden participar.

Otra caracterización importante de red en la célula, son las interacciones proteína – proteína, donde los nodos son proteínas. Existe una arista entre un par de proteínas si se ha demostrado experimentalmente que éstas se ligan una a la otra.

Redes ecológicas. Las redes ecológicas se utilizan para cuantificar las interacciones existentes entre diferentes especies. En una red alimenticia, los nodos son las especies y las aristas representan una relación predador-presa entre dos especies[**54**].

Redes en lingüística. La complejidad del lenguaje humano, ofrece muchas posibilidades para definir y estudiar las redes complejas. En [**55**], Ferrer et al, construyeron una red para el lenguaje inglés, basada en el "British National Corpus", con las palabras como nodos y existe una arista entre dos nodos si éstos aparecen juntos o a una palabra de distancia uno del otro en las sentencias.

Redes neuronales. En la pequeña red neuronal del gusano nemátodo C. elegans, los nodos son las neuronas. Dos neuronas están conectadas si existe una sinapsis o una juntura 'gap' entre las dos.

Desdoblamiento de proteínas. Durante su desdoblamiento, una proteína toma conformaciones consecutivas. Cada nodo representa un estado posible de las configuraciones de la proteína. Existe una arista entre dos estados si uno se puede obtener a partir del otro mediante un movimiento elemental.

Además, en un sistema existen diferentes estructuras de red funcionando al mismo tiempo, por ejemplo, en la célula existe la red de regulación genética, la red de interacción entre proteínas y la red de interacción metabólica. En el cerebro, además de las redes eléctricas existentes entre las neuronas, también existen redes químicas. De esta manera, también se debe estudiar el comportamiento de sistemas en donde coexisten diferentes tipos de redes.

La influencia que tienen estas redes en nosotros es increíble. Son parte de nuestra vida y de nuestro mundo. Nuestro presente y nuestro futuro son imposibles sin ellas. Sin embargo, sabemos muy poco de su estructura y su organización jerárquica, su topología global y sus propiedades locales, y de los varios procesos que ocurren en ellas. Este conocimiento nos hará hacer un uso más eficiente de Internet y de la WWW, para garantizar seguridad, nos ayudará a desarrollar planes para detener la propagación de enfermedades en una red social y nos ayudará a comprender los maravillosos mecanismos que dan lugar a la vida.

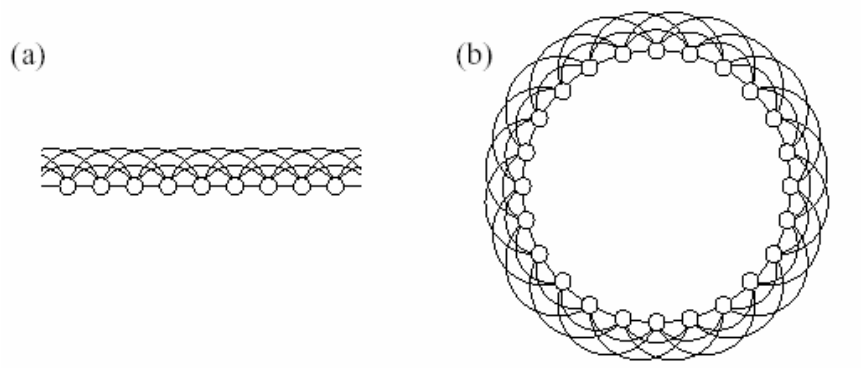


FIGURA 4. a) Latice regular unidimensional b) Latice regular unidimensional con condiciones de frontera periódicas.

En secciones posteriores se analizará cómo las propiedades globales medidas en cada una de estas redes están relacionadas directamente con el funcionamiento global de cada sistema.

2. Modelos de redes complejas

Se han propuesto modelos de redes que explican los parámetros medidos en las redes reales. Estos modelos pueden ayudar a entender la dinámica en la formación de las redes reales y pueden predecir algunas propiedades dinámicas de las mismas. En esta sección se describirán los modelos de redes más importantes que se han desarrollado hasta el momento.

2.1. Redes aleatorias[56]. Las redes con una topología compleja y principios de organización desconocidos, parecen grafos aleatorios, es decir, grafos en los que las aristas están distribuidas aleatoriamente, por lo que el primer modelo para tratar de entender las redes reales, fue un modelo de red aleatoria. Erdös y Renyi[**50**] desarrollaron una teoría de grafos aleatorios en donde se estudian las propiedades del espacio de probabilidades asociados con grafos de N nodos conforme $N \to \infty$. Muchas de las propiedades de los grafos aleatorios pueden ser determinadas utilizando argumentos probabilísticos. En este sentido Erdos y Renyi utilizaron la definición de que cualquier grafo tenía la propiedad Q si la probabilidad de tener Q se aproxima a 1 cuando $N \to \infty$.

Del tipo de propiedades que se investigan son ¿un grafo determinado es conexo? o ¿como se escala la longitud característica conforme el tamaño de la red crece?, entre otras. El principal objetivo de la teoría de los grafos aleatorios es determinar, para que probabilidad de conexión una propiedad particular del grafo es más probable que ocurra. Uno de los resultados más interesantes es que muchas propiedades importantes aparecen de repente,es decir, hay una transición de fase.

2.2. Redes de mundo pequeño.

2.2.1. *Modelo de redes de mundo pequeño de Watts y Strogatz.* Los grafos aleatorios muestran el efecto de mundo pequeño debido a que la longitud de camino característico se incrementa sólo logaritmicamente con el número total de vértices, pero no tienen un alto índice de agrupamiento como las redes reales.

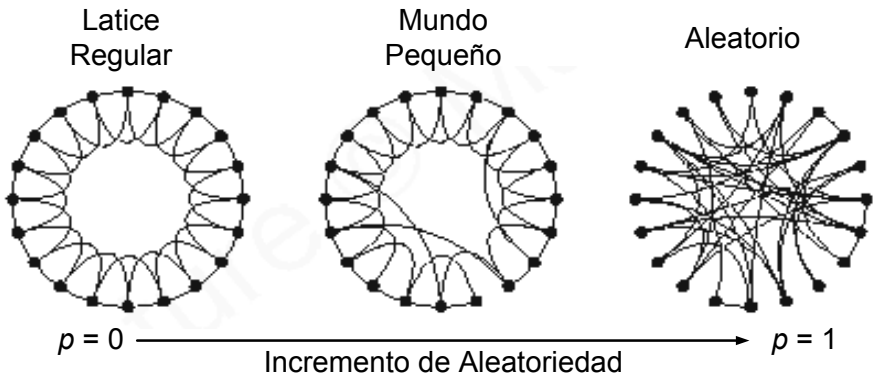


FIGURA 5. Procedimiento de recableado aleatorio, para interpolar entre una latice regular y una red aleatoria, sin modificar el número de vértices o aristas en el grafo.

El opuesto a un grafo aleatorio, en cierto sentido es una latice completamente ordenada, el ejemplo más simple es una latice unidimensional -un conjunto de vértices acomodados en una línea recta. Si se toma tal latice y se conecta cada vértice con los z vértices más cercanos a él, como se muestra en la Fig. 4 a) es fácil observar que la mayoría de los nodos inmediatos de cualquier vértice, son vecinos uno de otro, y por lo tanto esta red muestra la propiedad de agrupamiento. Normalmente se aplican condiciones de frontera periódicas a la latice Fig. 4 b). Para tal latice se puede calcular el coeficiente de agrupamiento de manera exacta. Si $z \leq \frac{2}{3}N$, se encuentra que[**57**]

$$C_r = \frac{3(z-2)}{4(z-1)}$$

El coeficiente de agrupamiento tiende $3/4$ cuando z crece. También se pueden crear latices de más dimensiones, como cuadradas o cúbicas, las cuales también muestran la propiedad de agrupamiento. El valor de C para cualquier dimensión d es

$$C_r = \frac{3(z-2d)}{4(z-d)}$$

Las latices de pocas dimensiones, sin embargo, no muestran el efecto de mundo pequeño. En una latice regular de d dimensiones, con la forma de cuadrado o hypercubo de lado L, y por lo tanto con $N = L^d$ vértices, la longitud de camino característico se incrementa con L, o equivalentemente con $N^{1/d}$. Para valores pequeños de d no se tiene la característica de mundo pequeño. Por ejemplo, en una dimensión, la longitud característica se incrementa linealmente con el tamaño del sistema.

Watts y Strogatz propusieron un modelo de redes que se puede ajustar entre un modelo completamente regular y un modelo completamente aleatorio: las redes regulares se "realambraran" para introducir desorden incrementalmente[**51**]. Para interpolar entre redes aleatorias y regulares, se utiliza el siguiente procedimiento de recableado (Fig. 5). Comenzando de una latice regular con n vértices y k aristas por vértice, se recablea cada arista aleatoriamente con probabilidad p. Esta construcción nos permite 'ajustar' el grafo entre la regularidad ($p = 0$) y el desorden ($p = 1$) y por lo tanto probar la región intermedia ($0 < p < 1$) en donde se tiene muy poco conocimiento.

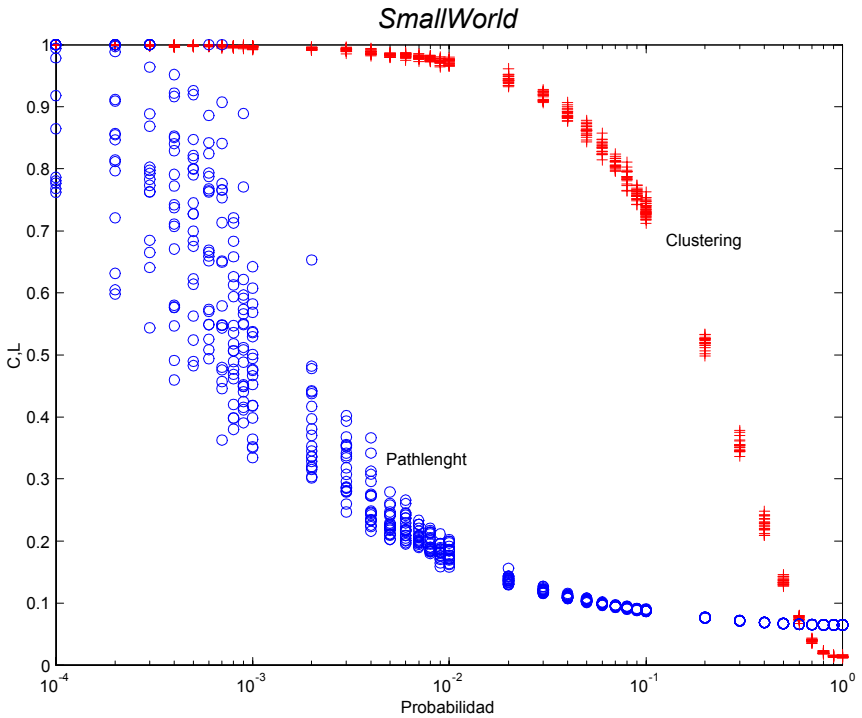


FIGURA 6. Longitud de camino característico $L(p)$ y coeficiente de agrupamiento $C(p)$ para distintos valores de p. Los valores mostrados son 20 realizaciones aleatorias del proceso de recableado descrito en el texto y han sido normalizados con los valores $C(0)$ y $L(0)$ de una latice regular. Todos los grafos son de $N = 1000$ y un grado promedio de $k = 10$ aristas por vértice. Se ha utilizado una escala horizontal logaritmica para resolver la rápida caída en $L(p)$.

Las redes de interés aquí, tienen muchos vértices con conexiones escasas, pero no lo suficientemente escasas como para que el grafo este desconectado. Específicamente se requiere que $n >> k >> \ln(n) >> 1$ donde $k >> \ln(n)$ garantiza que el grafo aleatorio este conectado[**56**].

Haciendo un análisis de los casos extremos, se observa que un C grande siempre esta asociado con una L grande. Y una C pequeña con una L pequeña. Sin embargo en la Fig. 6 se muestra que hay un intervalo amplio en p sobre el cual $L(p)$ es tan pequeña como L_{rand} y $C(p) >> C_{\text{rand}}$. Estas redes de mundo pequeño resultan por la caída inmediata en $L(p)$ causada por la inserción de unas pocas aristas de largo alcance o "atajos". Para p pequeña, cada 'atajo' tiene un efecto no lineal muy grande sobre L, contrayendo la distancia, no sólo entre el par de vértices que conecta, sino también entre vecinos inmediatos, vecinos de vecinos y así sucesivamente.

En contraste, para una vecindad agrupada un 'atajo' tiene sólo un efecto lineal sobre C; por lo tanto $C(p)$ permanece casi sin cambio para p pequeñas. La implicación importante aquí es que a nivel local, la transición de mundo pequeño no es detectable.

De esta manera, las redes basadas en el modelo de Watts y Strogatz, tienen la propiedad de mundo pequeño y la propiedad de alto agrupamiento al mismo tiempo, lo cual corresponde a los valores encontrados en las redes reales. En [**58**] se hace un

estudió analítico de las propiedades en modelos de mundos pequeños como el descrito anteriormente.

2.3. Redes libres de escala. Otro resultado empírico es que muchas redes son libres de escala[**32**] esto es, la distribución del grado sigue una ley de potencia. Ni los grafos aleatorios, ni los modelos de mundo pequeño reproducen esta característica. Se pueden construir grafos aleatorios con cualquier distribución predefinida[**59**]. Sin embargo la construcción de estos grafos, sólo pospone una pregunta más importante: ¿Cuál es el mecanismo responsable de la emergencia de las redes libres de escala? Para responder a esta pregunta se necesita hacer un cambio, de modelado de la topología de la red, a un modelado del ensamblado y la construcción de las mismas. Hay una diferencia fundamental entre la aproximación que se tomó en el modelado de las redes aleatorias y los modelos de redes de mundo pequeño, y la aproximación de ensamblado para reproducir las distribuciones de leyes de potencia: mientras el objetivo de los primeros es construir una arquitectura con las características topológicas correctas, en los modelos libres de escala lo que se busca es capturar la dinámica de la red esto es, la suposición subyacente detrás de la evolución y la dinámica de las redes es que si se pueden capturar correctamente los procesos de ensamblaje de las redes reales, entonces también se obtendrá la topología correcta. La dinámica toma el papel principal, y la topología es sólo un producto de esta forma de modelado.

2.3.1. *El modelo de Barabási y Albert.* Barabási y Albert[**60**] argumentan que la naturaleza libre de escala de las redes reales tiene sus raíces en dos mecanismos genéricos. Los modelos de redes discutidos con anterioridad suponen que se comienza con un número fijo de vértices. En contraste, la mayoría de las redes reales describen sistemas abiertos que crecen por la adición continua de nuevos nodos. En este modelo se comienza por un pequeño núcleo de nodos, el número de nodos se incrementa durante el tiempo de vida de la red mediante la adición continua de nuevos nodos. Por ejemplo, la WWW crece exponencialmente en tiempo, mediante la adicción de nuevas páginas web, y la literatura científica crece constantemente, mediante la adición de nuevos artículos.

En este modelo se toma en cuenta el grado de los nodos al momento de establecer nuevas conexiones. Se dice que las redes reales exhiben un enlace preferencial, de manera que la probabilidad de conexión hacia un nodo depende del grado del mismo. Por ejemplo, en la WWW es más probable, que los enlaces de una página web vayan hacia documentos populares, con un grado actual alto.

El modelo de Barabasi-Albert es etonces:

- *Crecimiento*: Se comienza con un número pequeño m_0 de nodos, y durante cada paso de reloj se añade un nuevo nodo con $m(\leq m_0)$ aristas, que enlazarán al nuevo nodo a m nodos diferentes que ya estaban presentes en el sistema.
- *Enlace preferencial*: Cuando se escogen los nodos a los que el nuevo nodo se enlazará, se supone que la probabilidad de que el nuevo nodo sea conectado a un nodo i depende del grado k_i del nodo i, de la siguiente forma

$$\Pi(k_i) = \frac{k_i}{\sum_j k_j}$$

Después de t pasos este procedimiento resulta en una red con $N = t + m_0$ nodos y mt aristas. Las simulaciones indican que la red evoluciona a un estado invariante

de escala en donde la probabilidad de que un nodo tenga k aristas sigue una ley de potencia con exponente 3. Este exponente es independiente de m el único parámetro del modelo.

3. Dinámica en redes complejas

El objetivo final en el estudio de la estructura de las redes complejas es comprender y explicar el funcionamiento de los sistemas construidos sobre estas. Nos gustaría, por ejemplo, entender cómo la topología de la WWW afecta la navegación de los motores de búsqueda, cómo la estructura de las redes sociales afecta la propagación de la información, cómo la estructura de una red de alimentos afecta la dinámica de poblaciones, etc. De esta manera, ahora que se estudiaron algunos modelos estructurales de las redes, se analizará el comportamiento de procesos físicos, biológicos y sociales que tienen lugar en estas redes. Los avances en esta área han sido lentos, sin embargo, han habido estudios importantes en la tolerancia a fallas, procesos epidémicos y navegación en redes.

3.1. Tolerancia a fallas y ataques en las redes complejas. Muchos sistemas complejos muestran un sorprendente grado de tolerancia a fallas. Por ejemplo, organismos relativamente simples crecen, persisten y se reproducen a pesar de drásticas intervenciones farmacéuticas o ambientales esta tolerancia a errores se atribuye a la robustez de la red metabólica subyacente. Las redes de comunicación también muestran un sorprendente grado de robustez: aunque algunos componentes funcionen mal, raramente llevan a la perdida del manejo de información global de la red. La estabilidad de estos y otros sistemas complejos frecuentemente se atribuye al cableado redundante en la red funcional definida por los componentes del sistema. Sin embargo, Albert et al.[**64**] encontraron que la tolerancia a errores no era una propiedad compartida por todos los sistemas redundantes: sólo se presenta en redes que están enlazadas de manera no homogénea, tal como las redes libres de escala.

3.2. Procesos epidemiológicos. Una de las razones originales y de mucha importancia para el estudio de las redes, era entender los mecanismos mediante los cuales las enfermedades, la información, los rumores y los virus de computadora, se propagan sobre éstas. Por ejemplo, el objetivo principal en el estudio de las redes de contactos sexuales[**52**] es para ayudar a entender y tal vez controlar la propagación de enfermedades de transmisión sexual. De manera similar se estudian las redes de e-mail[**65**] para aprender la forma en que se propagan los virus de coputadora.

El modelo más simple de propagación de enfermedades sobre una red es el modelo SIR[**66**]. Este modelo divide la población en tres clases: susceptibles (S), lo cual significa que no tienen la enfermedad de interés, pero que pueden contagiarse si se exponen a alguien que la tenga, infectado (I), significa que tienen la enfermedad y que pueden transmitirla, y recuperado (R) significa que se ha recuperado de la enfermedad y tiene una inmunidad permanente, de manera que no puede contagiarse de nuevo, ni tampoco transmitirla.

En matemática epidemiológica tradicional[**66**] se supone que cualquier individuo susceptible tiene una probabilidad uniforme β por unidad de tiempo de ser contagiado por cualquier agente infeccioso, y que los agentes infectados se recuperan y se vuelven inmunes en alguna tasa estocástica constante γ. Las fracciones s, i y r de individuos en los estados S, I y R están gobernadas por las ecuaciones diferenciales

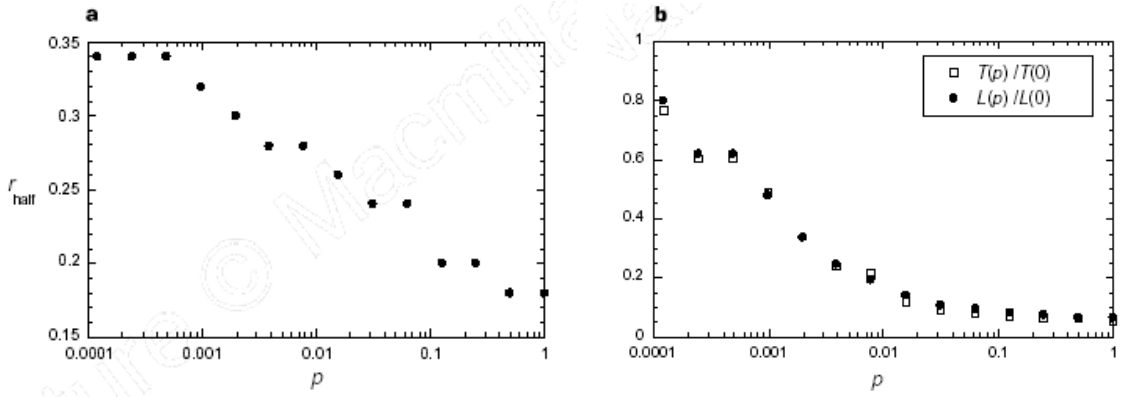


FIGURA 7. Tomada de [**51**]. Propagación de enfermedades en redes de mundo pequeño.

$$\begin{aligned} \frac{ds}{dt} &= -\beta is \\ \frac{di}{dt} &= \beta i - \gamma i \\ \frac{dr}{dt} &= \gamma i \end{aligned}$$

Los modelos de este tipo se conocen como modelos completamente mezclados, y aunque nos han enseñado mucho acerca de la dinámica básica de las enfermedades, obviamente, las suposiciones que se hacen son demasiado irreales. Las enfermedades sólo pueden ser propagadas entre aquellos individuos que tienen contacto físico, y por lo tanto la estructura de la red de contactos es importante para el patrón de desarrollo de la enfermedad.

El modelo SIR puede ser generalizado de manera directa a una epidemia que tenga lugar en una red. En [**51**], Watts estudió la propagación de información en su modelo de redes de mundo pequeño. En el tiempo $t = 0$ es introducido un individuo infeccioso a la red. Estos agentes infecciosos pasan al estado recuperado, en un intervalo de tiempo que dura una unidad de tiempo adimensional. Durante este intervalo de tiempo, cada individuo infeccioso puede infectar a cada uno de sus vecinos susceptibles con una probabilidad r. Conforme el sistema evoluciona, la enfermedad se propa a lo largo de las aristas del grafo, hasta que infecta a toda la población, o muera, infectando a alguna fracción de la población en el proceso.

Se encontraron dos resultados: primero, la infección crítica $r_{1/2}$ en la cual la enfermedad infecta a la mitad de la población, decrece rápidamente para p pequeño (Fig. 7a) y segundo, para una enfermedad que es lo suficientemente infecciosa, como para infectar a la población entera sin importar su estructura, el tiempo $T(p)$ requerido para una infección global es muy similar a la curva de camino característico, para diferentes valores de p (Fig 7b). Así, las enfermedades infecciosas se esparcen mucho más fácil y rápidamente en un mundo pequeño; el punto alarmante es cuántos 'atajos' son necesarios para hacer un mundo pequeño.

Una observación que ha permitido hacer estudios analíticos de la propagación de enfermedades en redes es que el modelo puede ser trasladado a un problema de precolación por enlace en la misma red[**67**]. Las predicciones acerca de las epidemias en el modelo de precolación es simple: la distribución de componentes de precolación (i.e. componentes conectados por aristas ocupadas) corresponde a la distribución de los tamaños de las epidemias que comienzan con un portador inicial, la transición a precolación corresponde al umbral epidemiológico, arriba del cual una epidemia es posible (i.e., una que infecta a una fracción de la población diferente de cero en el límite del tamaño del sistema) y el tamaño de este componente gigante arriba de la transición corresponde al tamaño de la epidemia. La solución analítica fue dada en[**68**]. Una de las conclusiones más importantes fue para el caso de las redes libres de escala, para las cuales, en el caso de percolación por sitio, hay un umbral epidémico distinto de cero cuando el exponente de la ley de potencia es menor que 3. Debido a que la mayoría de las redes satisfacen esta condición, se espera que las enfermedades siempre se propaguen en estas redes, sin importar la probabilidad de transmisión entre individuos.

Un aspecto epidemiológico que se puede tratar ahora es la estrategia de vacunación de una población para prevenir la propagación de enfermedades. La vacunación, puede ser vista como la eliminación de un conjunto particular de vértices de la red. Los modelos predicen que las redes tienden a ser vulnerables a la eliminación de los vértices con grado más alto, por lo que se espera que la vacunación de estos individuos sea efectiva en la prevención de la propagación de la enfermedad.

3.3. Navegación y búsqueda en redes complejas. El fenómeno de mundo pequeño tiene dos componentes fundamentales: primero, la existencia de caminos cortos entre cualquier par de vértices y segundo, individuos operando únicamente con información local tienen la capacidad de encontrar estos caminos cortos. En [**69**], Kleinberg modela cómo los individuos pueden encontrar caminos cortos en grandes redes sociales, y que es más fácil encontrar estos caminos en unas redes que en otras.

Se encontró que las pistas necesarias para descubrir caminos cortos emergen en un modelo de red muy sencillo. Este modelo de red es similar al de Watts y Strogatz[**51**]. Las conexiones de largo alcance son insertadas a una latice bidimensional, y estan controladas por un exponente de agrupamiento α, que determina la probabilidad de conexión entre dos vértices como una función de su distancia de latice (Figura 8).

Se estudian algoritmos descentralizados para transmitir mensajes: en cada paso, el poseedor de un mensaje debe pasarlo a través de alguna de sus conexiones; es importante notar que el poseedor del mensaje no conoce las conexiones de largo alcance que no han tocado el mismo. El rendimiento de estos algoritmos se mide con el tiempo de entrega esperado T que representa el número esperado de pasos necesarios para dirigir un mensaje de una fuente aleatoria a un objetivo específico. Es crucial restringir el algoritmo a que solamente utilice información local –con un conocimiento global de todas las conexiones en la red, los caminos cortos pueden encontrarse de una manera relativamente sencilla[**70**].

Como ya se vio, una de las características de las redes de mundo pequeño es que su diámetro está limitado por un polinomio en el $\log N$. Sin embargo esto no implica que un algoritmo descentralizado sea capaz de descubrir tales caminos cortos. Uno de los aspectos más interesantes del modelo de Kleinberg es que hay un único valor de α para el cual esto es posible. Cuando $\alpha = 2$, existe un algoritmo descentralizado que logra un

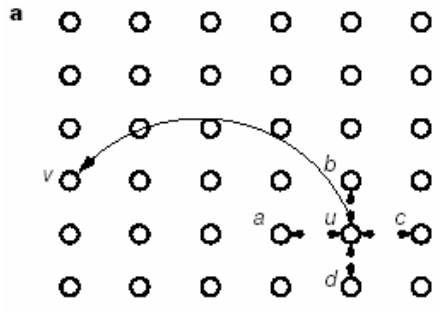


FIGURA 8. El modelo de red es derivado de una latice de $n \times n$. Cada vértice u, tiene una conexión de corto alcance con sus vecinos más cercanos (a, b, d y c). Y una conexión de largo alcance a un vértice v escogido aleatoriamente con probabilidad $r^{-\alpha}$, donde r es la distancia Manhattan entre u y v.

tiempo de entrega muy rápido, T está acotado por $O(\log^2 N)$. El algoritmo que logra esta cota es una heurística 'glotona': cada poseedor dirige su mensaje a la conexión que lo lleva al vértice más cercano posible al objetivo en una distancia de latice. Más aún, $\alpha = 2$ es el único exponente para el cual cualquier algoritmo descentralizado puede lograr un tiempo de entrega acotado por $\log N$.

Estos resultados indican, que la navegación eficiente es una propiedad fundamental de sólo algunas estructuras de mundo pequeño. Si bien este modelo es muy idealizado, hay una idea general muy importante: la correlación entre la estructura local y los enlaces de largo alcance dan pistas para encontrar caminos cortos a través de la red.

Un modelo alternativo, menos idealizado que el de Kleinberg y que muestra un modelo plausible de la estructura, ha sido propuesto por Watts et al.[**71**]. En este modelo se utiliza la idea de que se navega a través de las redes sociales buscando por características comunes entre sus conocidos y el objetivo, tales como localización geográfica u ocupación. Esto siguiere un modelo en el que los individuos están agrupados en categorías de acuerdo, por ejemplo, a sus trabajos. Estas categorías pueden ser agrupadas en supercategorías y así sucesivamente, creando una organización tipo árbol que define una distancia social entre cualquier par de gentes: la distancia social entre dos individuos es medida como la altura del nivel más bajo en el árbol al cual están conectados los dos individuos (Fig. 9).

Este árbol no es la red es sólo una construcción mental que afecta la forma en que la red crece. La probabilidad de que exista una arista entre dos vértices es mayor mientras más cercanos socialmente estén los individuos. Watts asumió que esta probabilidad caía exponencialmente con la distancia social. Se utiliza el mismo algoritmo 'glotón', pero ahora se busca al vecino con la menor distancia social al objetivo. Se mostró mediante simulaciones por computadora que el algoritmo se comporta eficientemente para varios parámetros del modelo en un tiempo $O(n)$.

Aunque el modelo se creó para estudiar las búsquedas en redes sociales, se puede utilizar como una forma para diseñar redes. Si se agrupan de manera jerárquica los elementos en una base de datos distribuida, en base a algunas características identificables, el algoritmo 'glotón' que esté consciente de tales características, podrá buscar los elementos de manera eficiente.

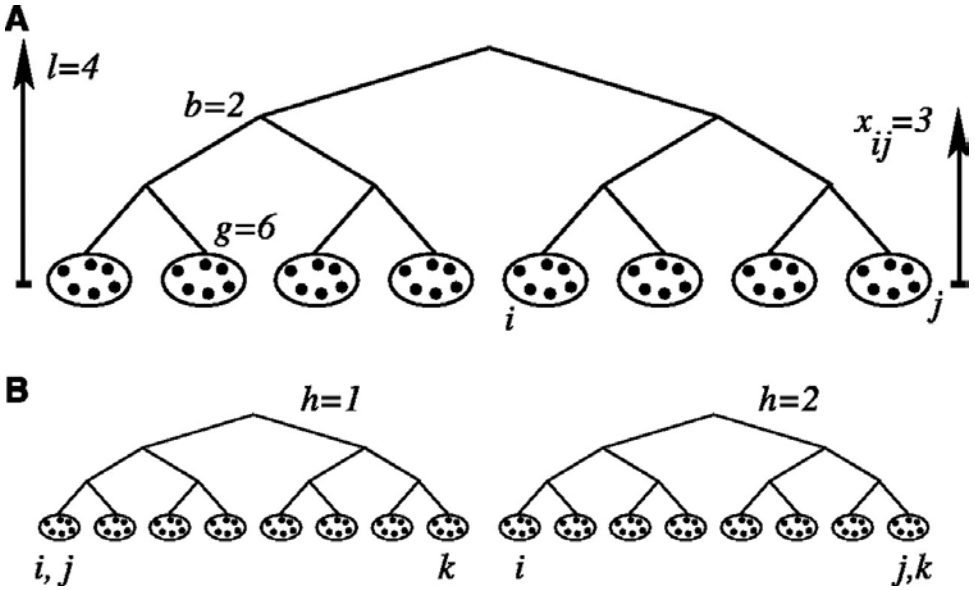


FIGURA 9. Jerarquía arborescente, para definir una distancia social

4. Optimización extrema distribuida

Se describe un nuevo algoritmo de optimización, denominado optimización extrema distribuida. Esté es una generalización del algoritmo de optimización extrema (OE) que contempla la topología de interacción. OE resulta como el caso particular en que se tiene una red completamente conexa. Se utiliza el algoritmo para resolver el problema de coloreado de grafos y se compara con el algoritmo de optimización extrema tradicional.

La optimización extrema[**40**] es una heurística de propósito general para encontrar soluciones de alta calidad para problemas de optimización difíciles en un tiempo proporcional a un polinomio del tamaño de la entrada. La optimización extrema se basa en un modelo que presenta auto-organización crítica[**41**], una propiedad que ha sido utilizada ampliamente para explicar el comportamiento de muchos sistemas complejos, en particular modelos evolutivos. A continuación se describe tal heurística.

En los modelos de evolución, las especies biológicas son acopladas en un proceso comparativo que continuamente elimina al menos adecuado. En este proceso estructuras altamente adaptadas surgen inadvertidamente. De esta manera la adaptación óptima surge de la dinámica, simplemente a través de la selección contra el extremadamente más malo. Este proceso previene la inevitable inflexibilidad de la cultivación controlada de una buena solución como en el caso de los Algoritmos genéticos.

Bak et. al.[**42**] presentó un modelo basado en este principio. En su modelo, las especies están localizadas en una cuadrícula y tienen asociado un valor de "adaptabilidad" entre 0 y 1. En cada paso de tiempo, la especie con el valor más pequeño (la peor adaptada) es seleccionada para una actualización aleatoria, pero el cambio en la adaptación de una especie impacta en la adaptabilidad de las especies interrelacionadas. Así todas las especies de la vecindad también reemplazan su valor de adaptabilidad por otro aleatorio. Después de un número suficiente de pasos, el sistema alcanza un estado altamente correlacionado, conocido como auto-organización crítica. En este estado de alta correlación, una perturbación (la modificación aleatoria de una variable) no sólo afecta localmente al sistema, sino que se propaga a través del sistema completo[**41**]. Esta actividad co-evolucionaria da lugar a una cadena reacciones llamada "avalancha", largas fluctuaciones que reconfiguran la mayor parte del sistema, haciendo potencialmente accesible cualquier configuración . De esta forma la mayoría de las especies han alcanzado una adaptación por arriba de un cierto límite.

Se ha utilizado este modelo como motivación para una nueva aproximación a los problemas de optimización. La heurística que se ha introducido, llamada optimización extrema (OE)[**40**], solamente actualiza aquel elemento que tiene el "peor" valor en la solución y se reemplaza por un valor aleatorio sin buscar ninguna mejora explícita sobre ellos. OE añade una nueva heurística de aproximación a los problemas de optimización duros, utilizando largas fluctuaciones inherentes al sistema manejado en el estado de auto-organización crítica. Es decir, cuando se explora el espacio de estados, se utilizan las largas fluctuaciones, para hacer un muestreo eficiente del mismo.

El planteamiento general de un problema de optimización es como sigue: se tiene un conjunto X de n-variables (grados de libertad) $x_1, x_2, ..., x_n$ donde cada una tiene asociado un dominio X_i, los dominios pueden ser distintos y una función $C : \Omega \rightarrow \mathbf{R}$ que se quiere optimizar, donde $\Omega \subseteq X_1 \times X_2 \times ... \times X_n$ representa el conjunto de valores permitidos (soluciones correctas) para un problema dado. El objetivo es entonces,

encontrar la configuración $S \in \Omega$ que optimice (maximice o minimice) C. Sin embrago, la búsqueda de S es muy complicada, debido a que la cardinalidad del espacio de soluciones crece exponencialmente con el número de grados de libertad del problema $|\Omega| \approx |X_1| \cdot |X_2| \cdot ... \cdot |X_n|$ (por ejemplo, aunque las variables sean binarias, el espacio de búsqueda sería aproximadamente de 2^n).

Para utilizar la heurística de optimización extrema en los problemas de optimización se necesita reformular el problema, de forma tal que se pueda asociar a cada variable una función de adaptación $f_i : \Omega \to \mathbf{R}$, que indique que tan adecuado (adaptado) es el valor actual de la variable x_i para tener una solución óptima. Con las funciones de adaptación se construye otra función O Ec. 4.1, de manera tal que O tenga una correlación positiva con C.

$$O(S) = \sum_{i=1}^{n} f_i(S), S \in \Omega \tag{4.1}$$

La forma de sumatoria de la Ec. 4.1, nos dice que para utilizar OE es necesario que las función de costo pueda ser descompuesta en contribuciones de cada uno de los grados de libertad individuales. Para muchos problemas de optimización, las funciones de adaptación que cumplen esta característica se obtienen de una manera muy directa a partir del planteamiento del problema, sin embargo para otros no.

El algoritmo de OE procede a buscar en el espacio de soluciones, mediante el cambio secuencial de la variable con la peor adaptación x_b. De manera tal, que después de cada actualización, el valor de la variable x_b sea actualizado a un nuevo valor tomado aleatoriamente de su dominio. Es importante mencionar que para muchos problemas la modificación de una sola variable puede llevar a una solución incorrecta, una solución $S \notin \Omega$, en estos casos (por ejemplo, el problema de bipartición de grafos (PBG) que se analiza en párrafos posteriores) es necesario modificar también algunas de las otras variables de manera que durante todos los pasos del algoritmo siempre se tenga una solución correcta $S \in \Omega$. Se utiliza una heurística para la selección de las variables que se van a modificar para mantener esta condición. Esta heuristica consiste en ordenar de menor a mayor todas las variables en base a su valor de adaptación actual, de manera tal que se tenga $f_{\pi(1)} \leq f_{\pi(2)} \leq ... \leq f_{\pi(n)}$ donde $\pi(i)$ es la etiqueta de la variable k–sima en la lista ordenada. Entonces la nueva variable se escogerá de acuerdo a la distribución de probabilidad de la Ec. 4.2. Así, para que OE resulte eficiente es necesario que el problema de llevar una configuración incorrecta a una válida se haga en tiempo polinomial con respecto al tamaño de la entrada.

$$P(k) \propto k^{-\mu}, 1 \leq k \leq n \tag{4.2}$$

El algoritmo básico de OE es como sigue:

(1) Plantear una configuración $S \in \Omega$ como se desee; establecer $S_{best} = S$.
(2) Para la configuración actual S, hacer
    (a) Evaluar f_i para cada variable x_i.
    (b) Encontrar la j tal que se cumpla $f_j \leq f_i$, $\forall i$ y por lo tanto x_j tiene la peor adaptación.

(c) `Escoger una configuración` $S' \in \Omega$ `que se alcance a partir de la modificación de la variable` x_j `de la configuración actual` S.
(d) $S = S'$
(e) `si` $O(S) \geq O(S_{best})$ `entonces establecer` $S_{best} = S$.
(3) `Repetir el paso 2 las veces que se desee.`

4.0.1. *Ejemplo: Problema de "bipartición de grafos".* El problema de la bipartición de grafos (PBG) se formula de la siguiente manera: Dado un grafo de N vértices, donde N es par, el problema consiste en separar los vértices del grafo en dos conjuntos de cardinalidad $N/2$, de manera tal que el número de aristas que conecta ambos conjuntos, el "tamaño del corte" m, sea mínimo. Este problema esta dentro de la clase de problemas NP-Completos[**34**].

Para aproximar este problema utilizando OE, asociamos a cada vértice v_i una variable x_i que puede tomar los valores de 0, 1, que indican la pertenencia a uno de los dos conjuntos. La función de adaptación de cada variable se define como sigue:

$$f_i = \frac{g_i}{g_i + b_i}$$

donde g_i es el número de aristas "buenas" que conectan a v_i con otros vértices dentro del mismo conjunto, y b_i es el número de aristas "malas" que conectan a v_i con vértices a través de la partición. Para nodos no conectados asociamos una $f = 1$. Está claro que optimizar la función implica minimizar el tamaño de corte del grafo.

Con respecto al inciso c) del algoritmo se da el caso en el que la solución puede caer fuera de Ω, debido a que se tiene que cumplir con la restricción de que el número de variables que tengan asignado el valor de 0 debe ser igual a las variables que tienen asignado un 1, por lo que se modifica una de las variables que están en el otro conjunto en base a la heurística descrita anteriormente.

4.0.2. *Optimización extrema distribuida.* Se propone una modificación al modelo Bak-Sneppen[**42**] de manera que se base solamente en interacciones locales, pero manteniendo la idea de desechar la peor solución (peores): las entidades que tengan un grado de adaptación menor al grado de adaptación de todos sus vecinos, desaparecerá y se generara una nueva especie con adaptabilidad aleatoria como en el caso anterior y también se generará un nuevo valor aleatorio para sus vecinos. Este nuevo modelo también presenta la característica de auto organización crítica, en donde la mayoría de las especies tiende a superar cierto umbral de adaptabilidad y depende fuertemente de la red en la que estan interaccionando los elementos del sistema.

El planteamiento general para el problema de optimización es el mismo que para el caso anterior. Ahora, debido que la base para esta nueva heurística de OE es un modelo de sistemas complejos con sólo interacciones locales. Cada una de las variables tiene asociada una vecindad $V : X \to P(X)$, (donde $P(X)$ es el conjunto potencia de X), que define el conjunto de variables con las que interactúa directamente (localmente) [6]. El algoritmo de OED procede entonces, a buscar en el espacio de soluciones, mediante el cambio distribuido de las variables con la peor adaptación con respecto a sus vecinos. Una variable v_i esta "peor" adaptada si se cumplen las dos condiciones siguientes:

(1) f_i es menor o igual que todas las f_j de las variables en la vecindad de v_i.
(2) Existe una variable de vecindad v_k tal que f_k es mayor que f_i.

Como en el caso de OE , si la modificación de la variable lleva a la solución a caer fuera de Ω, entonces se modifica una de las variables vecinas utilizando la heurística dada anteriormente, sin embargo, como este algoritmo es distribuido, la modificación de las variables se debe de hacer de manera atómica. El algoritmo básico de OED queda como sigue:

Algoritmo (Optimización Extrema Distribuida)
1. Plantear una configuración $S \in \Omega$ como se desee; establecer $S_{best} = S$
2. Para la configuración actual S, hacer
   Evaluar f_i para cada variable x_i.
   Para todas las i tales que se cumpla $f_{i/} \leq f_j \ \forall j \in V(x_i)$
   y $\exists j \ f_i < f_j$ y por tanto todos los x_i tienen peor adaptación hacer
      Escoger una configuración $S' \in \Omega$ que se alcance a partir
      de la modificación de la variable x_i de la configuración actual S
      $S = S'$
      si $O(S) \geq O(S_{best})$ entonces establecer $S_{best} = S$
3. Repetir el paso 2 las veces que se desee.

(1)

Esta nueva heurística tiene la ventaja de que se puede paralelizar muy fácilmente debido a que se basa sólo en interacciones locales.

4.0.3. *Ejemplo: Coloreado-k.* El problema Coloreado-k se formula de la siguiente manera: dados k diferentes colores utilizados para etiquetar los vértices de un grafo, debemos encontrar el coloreado que minimice el número de aristas "monocromáticas": aristas conectando a vértices del mismo color. Este problema se encuentra en la clase de los problemas NP-completos[**34**]. Para utilizar OED, cada vértice tiene asociada una variable que representa el color actual, de esta manera la función de adaptación se define como -1/2 el número de aristas monocromáticas conectadas a el.

4.0.4. *Vecindad de mundo pequeño.* El algoritmo OED cambia en base a como se defina la función de vecindad V, de hecho, el algoritmo básico de OE es un caso particular de OED cuando $V(x) = X$ es decir todas las variables estan conectadas con todas. Aunque en primera instancia pareciera que esta arquitectura resultaría más eficiente no es así. En los resultados se muestra cómo con una vecindad basada en modelos de mundo pequeño se puede mejorar la búsqueda, al mismo tiempo de que se facilita la implementación del sistema en un ambiente distribuido.

Una red de mundo pequeño mejora el rendimiento en la búsqueda debido a que se pueden encontrar buenos óptimos locales gracias al alto agrupamiento C y al mismo tiempo tener una relación con todo el sistema para comparar entre estos mínimos debido a la longuitud de camino característico L pequeña. Para probar la heurística se utilizaron diferentes tipos de grafo: grafos aleatorios; grafos geométricos y grafos de mundo pequeño. Los grafos aleatorios se construyen en base a una probabilidad p, que es la probabilidad de que exista un enlace entre cualquier par de vértices. Los grafos geométricos se construyen lanzando N puntos aleatoriamente con distribución uniforme a una cuadrícula de área unitaria; existe un enlace entre cada par de vértices si la distancia euclidiana que hay entre los dos es menor a un radio r. La construcción de los grafos de mundo pequeño se puede encontrar en[**51**].

	Grafos Aleatorios , 1000 vertices			*Grafos Geometricos , 1000 vertices*		
Prob / Radio	**OE**	**OED**	**OEDsw**	**OE**	**OED**	**OED sw**
0.01	84	132	**60**	**0**	1	**0**
0.03	452	411	**254**	35	31	**4**
0.05	718	667	**464**	73	110	**45**
0.07	1017	961	**730**	177	213	**126**
0.09	1294	1232	**988**	308	337	**260**
0.11	1665	1470	**1317**	510	535	**380**
0.13	1918	1794	**1674**	681	694	**596**
0.15	2252	2095	**1941**	930	920	**824**
0.17	2523	2336	**2234**	1183	1144	**987**
0.19	2873	2661	**2512**	1458	1399	**1361**
0.21	3149	2929	**2911**	1688	1680	**1613**
0.23	3472	**3243**	3279	2015	2007	**1921**
0.25	3828	**3562**	3543	2354	2335	**2262**
0.27	4060	**3813**	3838	2723	**2495**	2612
0.29	4464	**4152**	4112	3163	**3034**	3058
0.31	4750	**4401**	4450	3634	**3347**	3370

FIGURA 10. Comparación entre OE, OED y OEDsw, para distintos valores de p y r.

4.0.5. *Resultados: Coloreado-k.* Para este problema la heurística de OED resultó ser mejor (se encontró un valor de adaptación mas alto) para la mayoría de los grafos. En la tabla10 se muestra la grafica de adaptabilidad con respecto al tiempo (número de iteraciones) para grafos aleatorios con 1000 vértices y probabilidades .01, .a.0.31. Como se puede observar la arquitectura de mundo pequeño es mejor para una amplio rango de valores de p y de r, más importante es mejor para los valores de p y r donde se encuentran las instancias del problema más dificiles. Los resultados se presentaron en [**E**]

Con el algoritmo de Optimización Extrema Distribuida presentado en este trabajo se mostró el uso de la herramienta y la teoría de las redes complejas para el desarrollo de un algoritmo original para la solución de problemas de optimización.

CAPÍTULO 5

Motifs de redes

Resumen. En el capítulo anterior se vió como las redes complejas que ocurren en la naturaleza comparten características globales estadísticas. Para ir más allá de estas características globales se requiere una comprensión de los elementos estructurales básicos locales particulares a cada clase de red. Con este objetivo, en este capítulo se modifica la metodología para la extracción de Motifs de redes[**72**] –subgrafos (tres y cuatro nodos en este estudio) que ocurren en la red bajo estudio en cantidades estadísticamente significativas comparadas con redes aleatorias con las mismas propiedades globales.- en redes complejas. Se prueba la metodología modificada con las redes de regulación transcripcional de la bacteria E coli, el hongo S cerevisiae, dos circuitos electrónicos y la red alimenticia de la reserva Sn. Martín, todas estas utilizadas en la metodología original. Con la modificación descrita en este trabajo se identifica un nuevo Motif en el caso de las redes de regulación transcripcional y más instancias para cada uno de los Motifs previamente identificados.

1. Motifs de redes complejas

Se ha mostrado que muchas de las redes complejas que ocurren en la naturaleza comparten características globales estadísticas. Para ir más allá de estas características globales se requiere una comprensión de los elementos estructurales básicos locales particulares a cada clase de red. En [**72**] se desarrolló una metodologia (MBM) para detectar dichas estructuras locales denominadas "Motifs de redes":

El termino Motif se ha utilizado durante mucho tiempo en diferentes áreas para referirse a patrones recurrentes, por ejempl$_o$, en musica se utiliza para referirse a un fragmento perceptible que ocurre muchas veces dentro de una pieza musical, en bioquímica es una subsecuancia de simbolos (nucleotidos o amino acidos) que aparece recurrentemente en la codificación de un gene. El término Motif de red no necesariamnete es un subgrafo recurrente en sentido absoluto, sino recurrente en comparación con la aparción en estructuras aleatorias con propiedades específicas.

A continuación se describe y se dan los algoritmos que implementan los elementos de la metodologia MBM. Y se describe la modificación realizada en este trabajo.

1.1. Metodología modificada para la detección de Motifs de redes. Primero se describe, de manera resumida, la metodología seguida para la detección de los Motifs de redes presentada en [**72**], la cual en sí misma define él termino Motif de red. En secciones posteriores se detallan cada uno de los elementos de esta metodología y la modificación realizada a uno de estos elementos.

Se comienza con una red en donde las interacciones están representadas mediante aristas dirigidas. La red es analizada para encontrar **todos** los posibles subgrafos de tamaño n (en el presente estudio, $n = 3$ y 4), a diferencia de la metodología original

en la que se cuentan todos los subgrafos de tamaño n que no estan contenidos en otros subgrafos de tamaño n. El número de ocurrencias de cada subgrafo es almacenado en una tabla. Cada red contiene una cantidad numerosa de cada tipo de subgrafo de tamaño n. Para fijarse en aquellos que son probablemente más importantes, se compara la red real, con un conjunto de redes aleatorias adecuadas, de manera que sólo se seleccionen los patrones que aparecen en la red real en un número significativamente mayor que el que aparece en las redes aleatorias. Para que la comparación sea rigurosa, se utilizan redes aleatorias que tienen las mismas características a nivel de nodo que la red real: cada nodo en la red aleatoria tiene el mismo número de aristas de entrada y de salida que las que tiene su correspondiente nodo en la red real. Eso es para que los patrones no aparezcan, simplemente por las características estadísticas de la red, por ejemplo, la existencia de nodos con un gran número de aristas. En la Fig. 1 se muestra una vista esquemática de tal metodología y en el siguiente recuadro se describe formalmente.

Metodología modificada para la detección de Motifs de redes
Entrada: Grafo $RR = (V, E)$ con la red real a estudiar

1. Contar todo los subgrafos de tamaño n en RR, y almacenar dicho valor en el vector $\mathbf{S}_n^{RR}$. $\mathbf{S}_n^{RR}[k]$ es el número de subgrafos del tipo k contenidos en RR. (Paso modificado del original)
2. Generar un conjunto de redes aleatorias (50 en este estudio). Cada red aleatoria RA^i debe cumplir la propiedad de que $\forall i\ P_{in,RR}(k) = P_{in,RA^i}(k)$ y $P_{out,RR}(k) = P_{out,RA^i}(k)$ (donde $P_{in}(k)$, $P_{out}(k)$ son las distribuciones de grado de entrada y de salida respectivamente)
3. Para cada red aleatoria i hacer el mismo conteo que en el punto dos y almacenar los resultados en el vector $\mathbf{S}_n^{RA^i}$.
4. Seleccionar como Motifs de redes aquellos subgrafos con identificador k tales que: $Z = \frac{\mathbf{S}_n^{RR}[k] > med(\mathbf{S}_n^{RA}[k])}{\sqrt{var(\mathbf{S}_n^{RA}[k])}} > 3.03$

Así, se definen exactamente a los "**Motifs de redes**" como sigue:

Definición (Motifs de Redes)[72]: Dado un grafo $RR = (V, E)$. Un Motif de tamaño n en RR se define como un subgrafo de n vertices cuya probabilidad P de aparecer en una red aleatoria RA, con la propiedad de que $P_{in,RR}(k) = P_{in,RA}(k)$ y $P_{out,RR}(k) = P_{out,RA}(k)$ (donde $P_{in}(k)$, $P_{out}(k)$ son las distribuciones de grado de entrada y de salida respectivamente), en un número igual o mayor de veces que en la red real es menor a $P = 0.001$, es decir dos desviaciones estandar alejado de la media obtenida en el conjunto de las redes aleatorias.

Es importante mencionar que pueden existir subgrafos que son funcionalmente importantes, pero que no son estadísticamente significativos, los cuales serían olvidados por esta aproximación.

1.1.1. ***Algoritmo para la detección de subgrafos de tamaño*** n. El algoritmo debe de contar todos los subgrafos de tamaño n que estan contenidos dentro de un grafo G. El algoritmo descrito a continuación, no es el utilizado en [**72**], más aun, no resuelve el mismo problema computacional: en [**72**] se cuentan los subgrafos de tamaño n *que no*

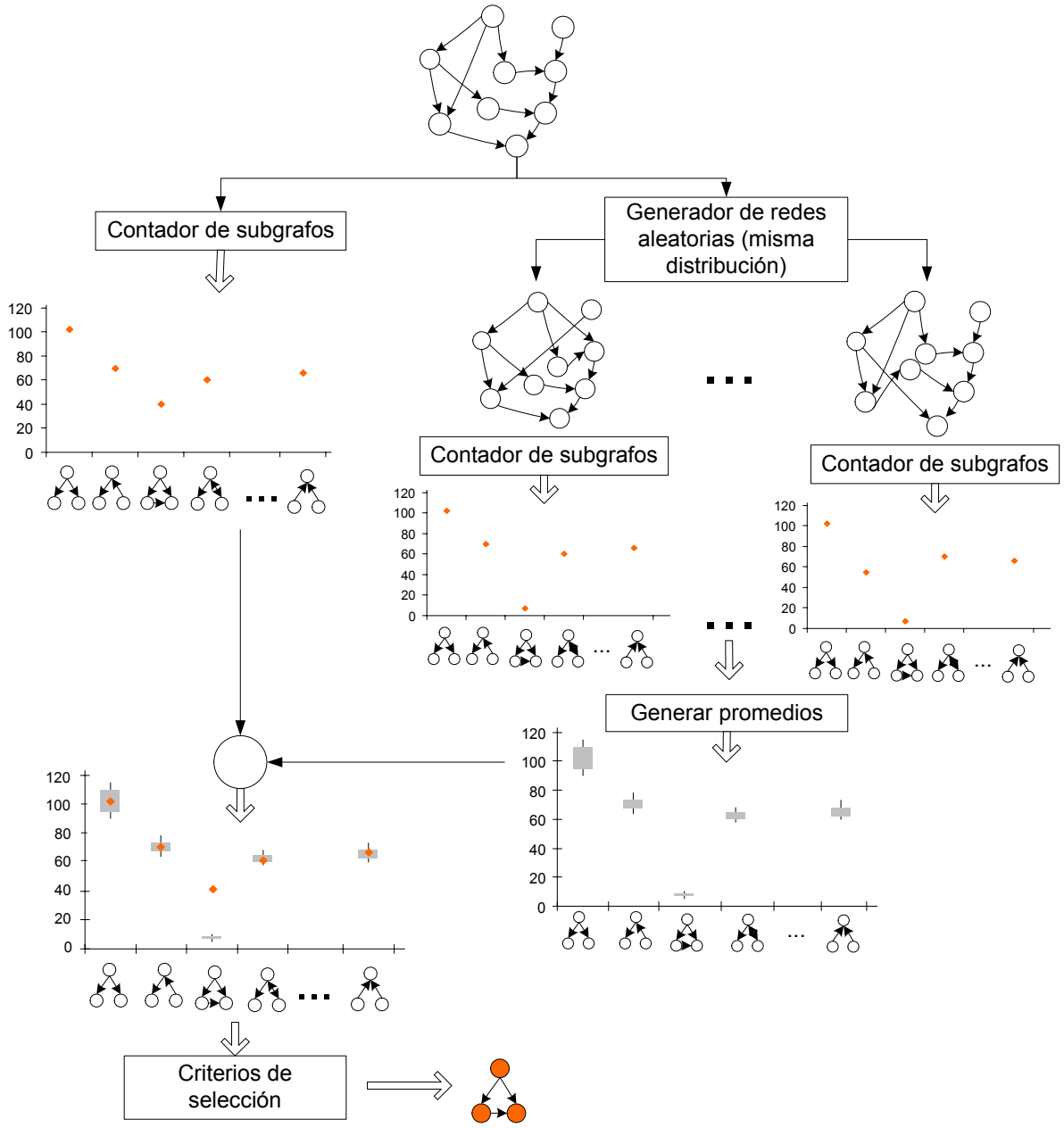


FIGURA 1. Metodología general para la detección de Motifs. 1) Se cuentan todos los subgrafos de tamaño n (en este estudio $n = 3, 4$) contenidos en la red bajo estudio y se alamacena el resultado en una tabla. 2) Se generan redes aleatorias con las mismas propiedades estadísticas que la red bajo estudio (50 en este análisis). 3) Se cuentan todos los subgrafos de tamaño n en cada una de las redes aleatorias. 4) Se comparan los conteos entre la red bajo estudio y las redes aleatorias. Se seleccionan los subgrafos cuyá probabilidad de aparición en una red aleatoria en el mismo número de instancias que en la red real es menor a 0.001

son subgrafos de algún otro subgrafo también de tamaño n. Esta diferencia es muy sutil, pero importante, y no se menciona en [**72**], de hecho los resultados que se presentan en [**72**] son en base a un algoritmo que resuelve el problema con restricción (no se da el código) cuando en la metodología simepre se habla del problema de identificar todos los subgrafos conexos de tamaño n (ver figura 2). Como son dos problemas computacionales distintos no hay uno que sea el correcto, sino que depende de la red que se este estudiando. En la sección de resultados se hablará de este punto con respecto a cada red en particular. A continuación se describe un algoritmo para resolver el problema de contar **todos** los subgrafos de tamaño n que hay en la red bajo estudio. También se diseño un algoritmo para el segundo, que puede consultarse en el CD que acompaña a la tesis.

El grafo G se representa a través de una matriz de conectividad $\mathbf{M}^G$, en donde $M^G_{ij} = 1$ indica que existe una arista dirigida del nodo i al nodo j.

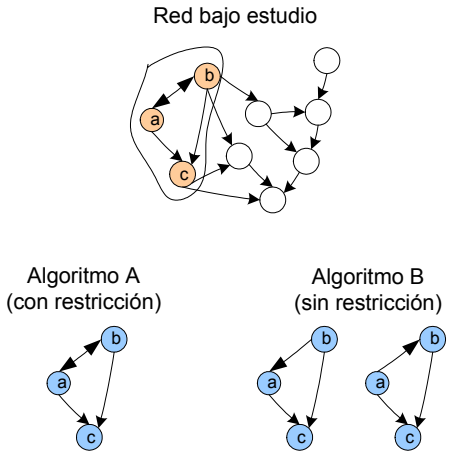


FIGURA 2. Comparación entre algoritmos para la detección de subgrafos de tamaño n. Cuando se analiza los nodos a, b y c (marcados en la figuara) de la red real, el algoritmo con restricción sólo contabilizará el subgrafo que contempla a todas las aristas que conecta cualquiera de estos nodos. Por otro lado el algoritmo sin restricción contará todos los subgrafos conexos de este subgrafo de tamaño tres. En la figura se muestran dos de estos subgrafos.

El algoritmo recorre todas las aristas de la matriz $\mathbf{M}^G$, para cada arista (i, j) se deben buscar todos los subgrafos de tamaño n que la contengan. Para hacer esto, se construye un árbol cuya raíz es la arista (i, j) (Fig. 3). Después se añade el conjunto de aristas que se puede alcanzar a través de la arista (i, j) esto es: $M^G_{ik} = 1$, $M^G_{kj} = 1$, $M^G_{ki} = 1$, y $M^G_{jk} = 1$. En este conjunto pueden existir aristas que forman ciclos respecto a este árbol, tales aristas se convertirán en hojas (Fig. 4). Esto se repite recursivamente con los elementos (i, k), (k, i), (j, k), y (k, j) que no son hojas (aristas que no forman ciclos) hasta que se obtiene un subgrafo de tamaño n denominado g_x. No se pueden agregar aristas al árbol que ya han sido utilizadas en el mismo nivel o en un nivel inferior. Se dice que una arista se ha utilizado si se expandió o es una arista formadora de ciclos. Por ejemplo, en la Fig. 3 se utilizó la arista $(0, 1)$ en el nivel 0, por lo que

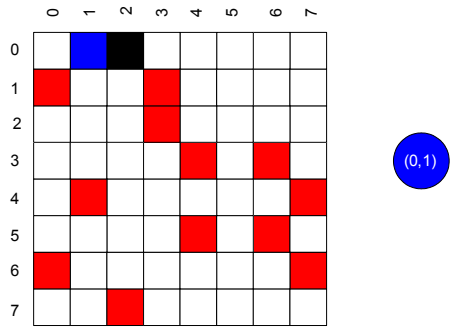


FIGURA 3. Matriz de conectividad $\mathbf{M}^G$. Se escoge la arista $(0,1)$ para buscar todos los subgrafos de tamaño n que la contengan. Después el algoritmo escogerá la arista $(0,2)$ para buscar los subgrafos que la contengan, pero que no incluyan a la arista $(0,1)$.

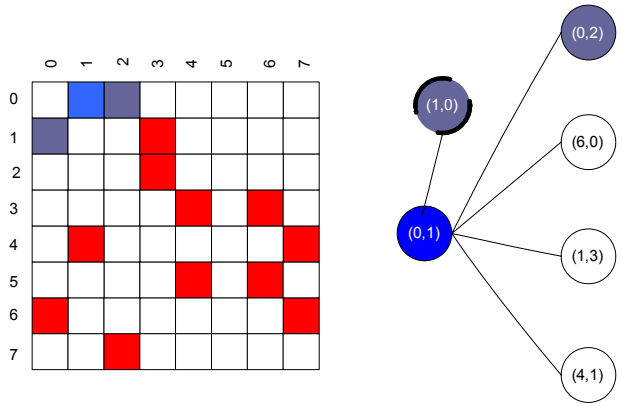


FIGURA 4. Algoritmo para la extracción de subgrafos de tamaño n. La arista $(1,0)$ forma un ciclo, por lo que se convertirá en hoja. Las aristas $(0,2)$, $(6,0)$, $(1,3)$ y $(4,1)$ se pueden alcanzar a través de (0,1) y por tanto son expandidas en el árbol.

ya nunca va a ser expandida en ningún árbol. Esto es se debe a que el algoritmo busca todos los subgrafos de tamaño n que tienen la arista $(0,1)$, después, durante el recorrido de la matriz busca todos los subgrafos de tamaño n que tienen la arista $(0,2)$, pero sólo aquellos que no tienen a $(0,1)$, y así sucesivamente. El mismo argumento se aplica para los niveles siguientes.

Se tiene una tabla que almacena el número de instancias de cada tipo de subgrafo que hay en la red. Para esto, una vez que se obtiene g_x a partir del árbol, se busca su tipo y se aumenta su contador en la tabla. Un grafo G_1 es de tipo G_x si G_1 y G_x son isomorfos. La prueba de si dos grafos son isomorfos es un problema NP-completo[**34**]. Sin embargo, como se está trabajando con problemas de tamaño 3 y 4, el problema aún es tratable. En este trabajo se utilizó el algoritmo presentado en [**74**].

Además del grafo g_x, también se agregan los subgrafos que se obtienen a partir de g_x y la adición de todas las posibles combinaciones, de las aristas formadoras de ciclos presentes en las aristas que generaron a g_x. Esto es, si se tienen c aristas formadoras

de ciclos, se tendrán $2^c - 1$ nuevos subgrafos. Por ejemplo, en el árbol de la Fig. 5, el subgrafo g_x que se formó es $g_x = \{(0,1),(0,2),(1,3)\}$, pero las aristas $(0,1)$ y $(1,3)$ expandieron las aristas formadoras de ciclos $(1,0)$ y $(2,3)$ respectivamente, de modo que también se agregan a la tabla los grafos $g_x^1 = \{(0,1),(0,2),(1,3),(1,0),(2,3)\}$, $g_x^2 = \{(0,1),(0,2),(1,3),(1,0)\}$, y $g_x^2 = \{(0,1),(0,2),(1,3),(2,3)\}$. El pseudocódigo resultante se muestra a continuación:

```
BusquedaMotivos(Grafo RR, int n)
    G_x = ({}, {})
    Para cada arista e_i ∈ E(RR) hacer
        Marcar e_i como utilizada
        E(G_x) = E(G_x) ∪ {e_i}
        Subgrafos(G_x, n − 1)
```

```
Subgrafos(G_x, n)
    Poner en la lista A^n todas las aristas no marcadas que estan
    conectadas al subgrafo G_x y que no forman ciclos
    y en A_c^n las que si forman ciclos

    si(n==1)
        P = 2^{∪A_c^n}
        Para cada p_i ∈ P
            G = G_x; E(G) = E(G) ∪ p_i
            Aumentar el contador del subgrafo isomorfo
            a G
        regresar

    Para cada e_i ∈ A^n hacer
        Marcar e_i como utilizada
        E(G_x) = E(G_x) ∪ {e_i}
        Subgrafos(G_x, n − 1)

    Desmarcar aristas marcadas durante esta funcion
```

1.1.2. ***Generación de redes aleatorias***. Se utilizaron dos algoritmos, para generar una red aleatoria con las mismas características a nivel de nodo que la red real. Esto es, cada nodo en la red real con k_i aristas de entrada y k_o aristas de salida, tiene un nodo correspondiente en la red aleatoria con las mismas k_i aristas de entrada y k_o aristas de salida. Los dos algoritmos dan estadísticas similares en el número de subgrafos de cada tipo.

El primer algoritmo empleado es el algoritmo de cadena de Markov presentado en [**75**]. Este algoritmo comienza con la red real y repetidamente, intercambia pares de conexiones aleatoriamente: $\{(x_1,y_1),(x_2,y_2)\}$ es remplazado por $\{(x_1,y_2),(x_2,y_1)\}$. Este procedimiento continua hasta que la red esté bien aleatorizada. Si cualquiera de las conexiones (x_1,y_2) o (x_2,y_1) ya existían en la red, entonces el intercambio se cancela.

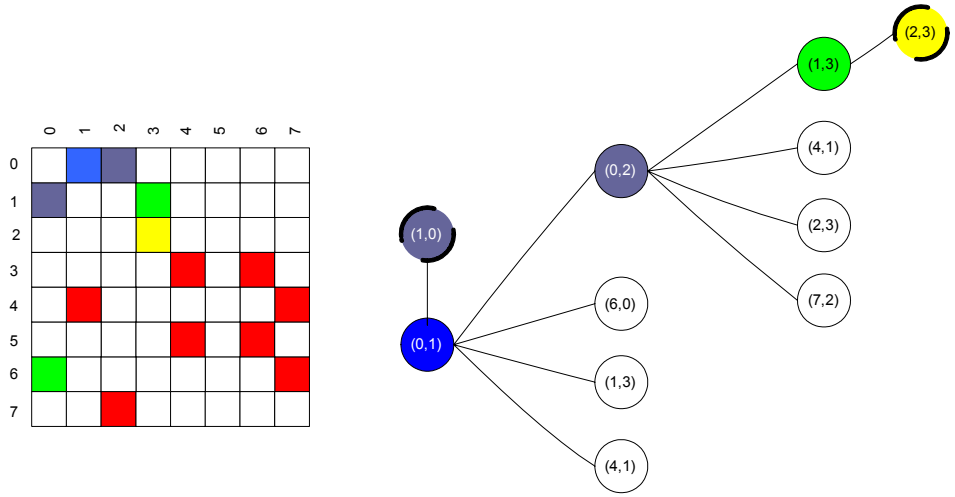


FIGURA 5. Algoritmo para la extracción de subgrafos de tamaño n. Las aristas $(0,1)$, $(0,2)$ y $(1,3)$ ya forman un subgrafo de tamaño n. Si se agregan las aristas $(1,0)$ y $(2,3)$ en cualquier combinación, se obtienen nuevos grafos de tamaño n.

El segundo algoritmo es una modificación del presentado en [**59**]. Cada red se representa por una matriz de conectividad $\mathbf{M}$. El objetivo es crear una matriz de conectividad $\mathbf{M}_A$ de una red aleatoria que tenga el mismo número de elementos distintos de cero que cada renglón y cada columna de la matriz $\mathbf{M}$:

$$R_i = \sum_j M_{Aij} = \sum_j M_{ij}, \text{ y } C_j = \sum_i M_{Aij} = \sum_i M_{ij}$$

Para generar la red aleatoria, se comienza con una matriz vacia $\mathbf{M}_A$. Repetidamente, se escoge de manera aleatoria un renglón r con una densidad de probabilidad $p(r) = R_r / \sum_i R_i$ y una columna c con una densidad de probabilidad $p(c) = C_c / \sum_i C_i$. Si $M_{A,rc} = 0$, se establece $M_{A,rc} = 1$. Se actualizan los valores $R_r = R_r - 1$ y $C_c = C_c - 1$. Si $r = c$, se vuelven a generar nuevos valores sin modificar la matriz $\mathbf{M}_A$. Este proceso continua hasta que $R_i = 0$ y $C_j = 0$, $\forall i, j$. Si el algoritmo no encuentra una solución en un número determinado de iteraciones se vuelve a correr desde el inicio.

1.1.3. ***Control de la aparición de Motifs de n-1 nodos***. En [**72**], una vez que se tiene la red aleatoria con las mismas características por nodo, se modifica para que preserve el número de apariciones de todos los subgrafos de $n-1$ nodos que tiene la red real. El objetivo es, asegurar que no se asigne una significancia alta a un patrón sólo porque está compuesto por un sub-patrón altamente significativo. En este trabajo se siguió un camino diferente. Se considera importante conocer los patrones en los que aparecen los Motifs significativos de tamaño $n-1$. Por otro lado es posible definir un orden parcial para el conjunto de todos los grafos de tamaño n, utilizando la relación de subgrafo. Así, pueden haber estructuras de tamaño n, que en base al criterio utilizado en [**72**], no se consideren como significativas porque están formadas también por estructuras significativas de tamaño n. Por tal motivo, no se modificó el grafo para que preservara el número de subgrafos de $n-1$ nodos. En vez de eso, una vez obtenidos los

Motifs significativos, se extrae la parte del diagrama de Hasse que contiene a los Motifs encontrados. A continuación se describen las dos aproximaciones.

Para tener una red aleatoria de hipótesis nula, como base para la detección de Motifs de tres nodos, se preserva el grado de entrada y de salida de cada nodo, así como el número de aristas mutuas ($x \leftrightarrow y$) para cada nodo. Esto se implementa modificando el algoritmo de cadena de Markov para que trate a las aristas mutuas y a las simples de manera separada. Una arista mutua sólo es intercambiada con otra arista mutua ($x_1 \leftrightarrow y_1$, $x_2 \leftrightarrow y_2$ se intercambia por $x_1 \leftrightarrow y_2$, $x_2 \leftrightarrow y_1$). Este intercambio se hace, sólo si ambos, (x_1 y y_2) y (x_2 y y_1) no están conectados por ningún tipo de arista. De manera similar, las aristas simples se intercambian ($x_1 \rightarrow y_1$, $x_2 \rightarrow y_2$ se remplaza por $x_1 \rightarrow y_2$, $x_2 \rightarrow y_1$) sólo si no forman aristas dobles.

Para tener una red aleatoria de hipótesis nula, como base para la detección de Motifs de cuatro nodos, se genera una red aleatoria que tiene el mismo número de subgrafos de tamaño tres que la red real. Para construir tal grafo, se utiliza una aproximación Monte-Carlo. Sea $V_{\text{real},k}$, para $k = 0...13$ el número de apariciones de cada uno de los trece subgrafos de tamaño tres en la red real y $V_{\text{rand},k}$ el vector correspondiente para la red aleatoria. Se define una energía $E = \sum_k |V_{\text{real},k} - V_{\text{rand},k}| / (V_{\text{real},k} + V_{\text{rand},k})$. La energía E es igual a cero sólo cuando el número de cada tipo de subgrafo de tamaño tres en la red real es igual que en la red aleatoria. Se comienza con una red aleatoria generada con el algoritmo de cadena de Markov. Después, se genera un cambio aleatorio ($x_1 \rightarrow y_1$, $x_2 \rightarrow y_2$ se remplaza por $x_1 \rightarrow y_2$, $x_2 \rightarrow y_1$, y de manera similar para las aristas mutuas), si el intercambio disminuye E, entonces se acepta, si no, se acepta con probabilidad $e^{-\Delta E/T}$, donde ΔE es la diferencia de energía antes y después del cambio y T es una temperatura efectiva. Este proceso se continua bajo el control del recocido simulado[**76**], para disminuir T lentamente, hasta que se obtiene una solución con $E = 0$. Este procedimiento se puede generalizar para tener una red aleatoria de hipótesis nula de $n - 1$ nodos, como base para la detección de Motifs de n nodos.

1.1.4. ***Diagramas de Hasse de Motifs de redes***. Una de las hipótesis más importantes acerca de los Motifs de redes es que pueden ser los bloques de construcción básicos de una red. Por lo tanto, es importante estudiar como se estructuran los Motifs dentro de las redes. Una primera aproximación (en la siguiente sección se presentan los Motifs jerárquicos) es olvidarnos de la hipótesis nula, y generar todos los Motifs de n nodos, independientemente de si están formados por Motifs de $n-1, n-2, ..., 3$ nodos, después, analizar como se relacionan estructuralmente unos con otros.

Se puede demostrar (se hace un mapeo, al orden parcial de conjuntos de cardinalidad finita), que la relación de subgrafo define un orden parcial en el conjunto de todos los posibles subgrafos de una red y también en el conjunto de todos los posibles grafos de tamaño n (en la Fig.6 se muestra el diagrama de Hasse para los grafos de tamaño tres). Por lo que se puede crear un diagrama de Hasse con los Motifs obtenidos en donde se observe su relación de contención. Por ejemplo, un Motif puede estar compuesto no sólo por uno, sino por dos o más Motifs. Los Motifs que no tengan padres en este diagrama son aquellos que no están compuestos por otros Motifs significativos y por lo tanto aquellos que se busca obtener mediante los grafos aleatorios de hipótesis nula.

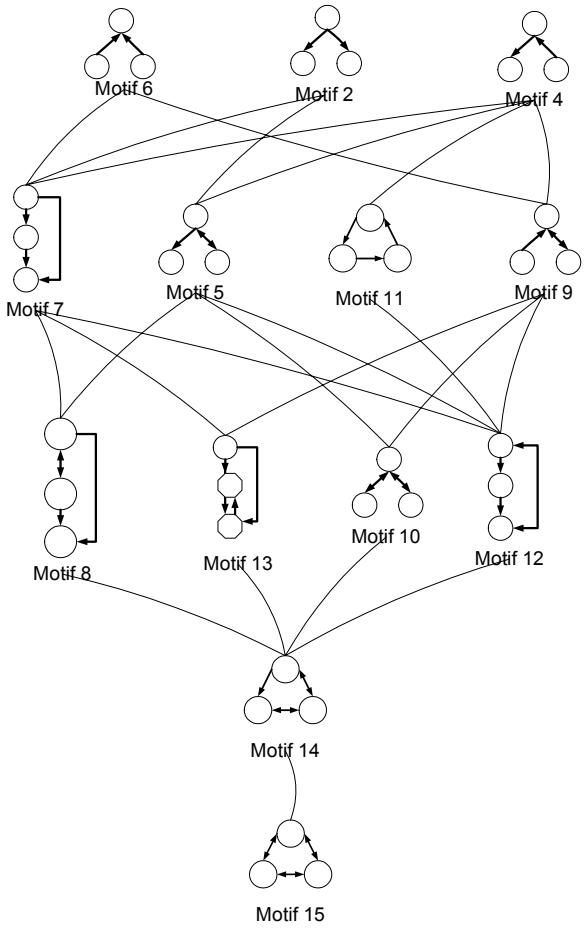


FIGURA 6. Diagrama de Hasse para los grafos de tamaño tres.

1.1.5. ***Criterios para la selección de Motifs.*** Para el presente estudio, los Motifs de redes son subgrafos que cumplen con los siguientes criterios.

(1) La probabilidad de que aparezca en una red aleatoria en un número igual o mayor que el número de veces que aparece en la red real es menor que $P = 0.001$ (dos desviaciones estandar alejadod e la media), esto es, la significancia estadística Z es

$$Z = \frac{nM - nR}{SD} \geq 3.03$$

donde:
   (a) nM es el número de instancias del Motif, encontradas en la red real.
   (b) nR es el número promedio de instancias del Motif, encontradas en un conjunto de 50 redes aleatorias con las mismas características por nodo que la red real.
   (c) SD es la desviación estandar del número de instancias del Motif, encontradas en el mismo conjunto de 50 redes aleatorias.

(2) El numero de apariciones nM en la red real es significativamente mayor que en las redes aleatorias:

$$p = \frac{nM - nR}{nR} \geq 1$$

Esto se hace para no detectar como Motifs a los subgrafos que tienen sólo una ligera diferencia en el número de instancias, pero que la distribución de las redes aleatorias es muy 'estrecha'.

En [**72**] se utiliza un criterio más: el número de veces que aparece un Motif en la red real con un conjunto completamente distinto de nodos es de al menos $U = 4$. En este trabajo se consideró al parámetro U como un elemento extra de información acerca de como están estructurados Motifs, pero no como criterio para la selección de los mismos. Por ejemplo, en la Fig. 5.6 se observa claramente la importancia estructural de las 16 instancias del Motif FFL (en el CD que acompaña a la tesis en el archivo `\IdMotifs.doc` está el listado de todos los subgrafos de tamaño tres y cuatro, junto con su identificador, en este caso FFL) que aparecen en una subred de *E coli* independientemente del valor $U = 1$. De hecho, más adelante se ve que U es muy diferente en los Motifs encontrados en redes naturales y artificiales. Generalmente, U es mayor en las redes artificiales, y por lo cual puede dar pistas de las diferencias que hay entre los sistemas naturales y los artificiales, como la robustez o los mecanismos de su formación: las redes naturales se forman mediante un proceso evolutivo y las artificiales mediante un proceso de diseño.

En la tabla 2, se muestran de manera resumida los criterios para la selección de Motifs utilizados en este trabajo y los utilizados en [**72**].

2. Resultados

Se aplicó la metodología a redes que representan sistemas de diferentes áreas. Se analizaron redes bioquímicas (redes de regulación genética), redes ecológicas (redes alimenticias) y de ingeniería (circuitos electrónicos). En la figura 8 se muestran los resultados obtenidos para las distintas redes utilizando las dos versiones para el conteo de subgrafos de tamaño n. En las siguientes secciones se describen los resultados obtenidos para cada tipo de red.

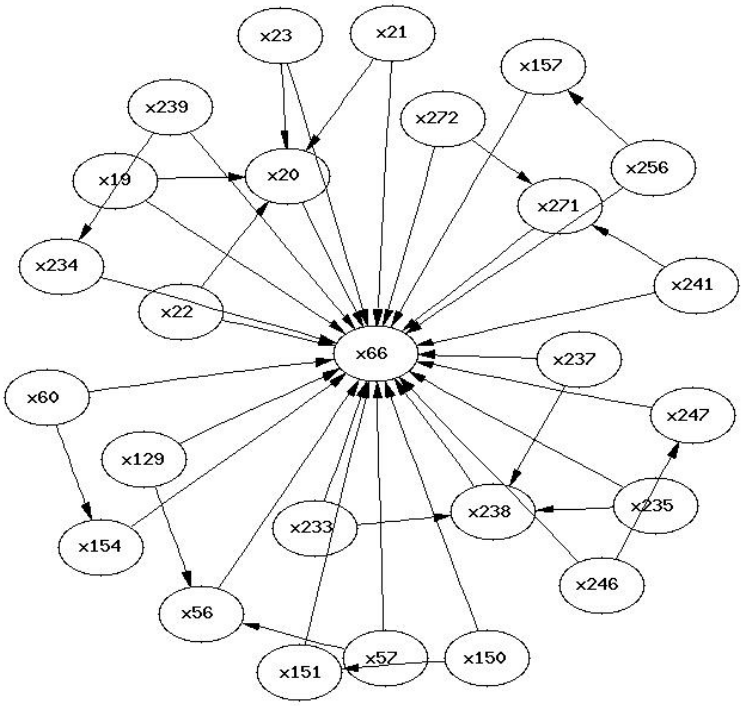


FIGURA 7. Subred de *E coli*, donde aparecen 16 instancias del subgrafo (FFL) con un valor $U = 1$.

TABLA 1. Criterios para la selección de motifs

	Menchaca-Méndez	**Milo et. al.**
Z	> 3.03	> 2.32
p	> 1	> 0.1
nM	> 4	-
U	-	> 3

TABLA 2

2.1. Red de trascripción genética. Las redes de trascripción genética son redes bioquímicas responsables de la regulación de la expresión de los genes en la célula. Estos son grafos dirigidos en los que los nodos representan genes. Las aristas son dirigidas desde el gen que codifica el factor de trascripción (proteína) hacia el gen regulado por dicho factor de trascripción. Se analizan las dos redes de regulación genética más caracterizadas en la actualidad, correspondientes a organismos de dos diferentes reinos: un eucarionte, el hongo de levadura *Saccharomyces cerevisiae* y una bacteria *Escherichia coli*.

Los datos de la red de *E. coli* fueron obtenidos de RegulonDB[**77**] y son los mismos que los utilizados en [**72**]. Los datos de *S. cerevisiae* fueron obtenidos de la página de los autores de [**72**]. Como se puede observar en la figura 8 se obtuvo un Motif mas cuando se utilizó el metodo de conteo diseñado en esta tesis. Éste nuevo Motif

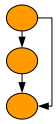
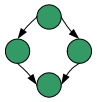
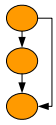
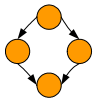
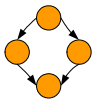
Red	Nodos	Aristas	nReal	nAleatorio±SD	z-score	U	nReal	nAleatorio±SD	z-score	U	nReal	nAleatorio±SD	z-score	U
Redes de transcripción genética			 FFL				 BF				 BP			
E. coli	424	519												
Milo, et al			40	7 ± 3	10.4	7	203	47 ± 12	13	11	No se considera como un motif			
Esta Tesis			40	8 ± 3	10.1	7	247	64 ± 12	14.8	11	6	1.36 ± 1	4.05	2
S. cerevisiae	685	1052												
Milo, et al			70	11 ±4	14	14	1812	300 ± 40	41	12	No se considera como un motif			
Esta Tesis			73	14 ± 3	16	14	2026	333 ± 39	43	13	9	2 ± 1.78	3.7	5
Circuito de lógica hacia adelante			 FFL				 BF				 BP			
s9234														
Milo, et al	5844	8197	211	2 ± 1	140	211	754	1 ± 1	1050	60	209	1 ± 1	200	137
Esta Tesis	5844	8074	211	1 ±1	160	211	515	0.3 ± 0.7	818	55	184	1 ± 1	173	130
Multiplicador fraccional digital			 FBL				 BF				 FBL4			
s838														
Milo, et al	512	819	40	1 ± 1	38	40	22	1 ± 1	20	9	23	1 ± 1	25	23
Esta Tesis	512	764	32	0.8 ±0.9	34	32	22	1.2 ± 1	20	8	23	0.5 ± 0.6	32	23
Red alimenticia			 BP											
Sn. Martin														
Milo, et al	42	205	382	130 ± 20	12	4								
Esta Tesis	45	224	1080	406 ±51	13	8								

FIGURA 8. Motifs identificados para distintos tipos de redes utilizando la metodologia presentada en[**72**]. Se muestran los resultados utilizando las dos variantes de conteo de subgrafos de tamaño n.

denominado BP es el mismo Motif que se encontró en las redes de circuitos electrónicos y en la red neuronal de *C. elegans*, reforzando así la hipótesis de que estos tres tipos de redes comparten (ahora si) exactamente los mismo Motifs por que todos estos sistemas manipulan información. El algoritmo de conteo sin restricción siempre encontrará más instancias de un Motif dado debido a que cuenta este subgrafo áun cuando este contenido en otro subgrafo de tamaño n.

Como se habia mencionado anteriormente, ninguna forma de conteo es incorrecta. Sin embrago, en el caso de la regulación transcripcional, los enlaces descritos por la red sólo reflejan posibilidad y por lo tanto en cada instante de tiempo sólo algunos de ellos están presentes, de modo que para cualquier subgrafo dado sólo una fracción de estos genes puede estar operando. Uno de estos subgrafos puede ser la istancia de un Motif que no se habia contabilizado porque estaba contenida dentro de otro subgrafo y por lo tanto se puede perder información valiosa. Este nuevo algoritmo fue utilizado en [**A**] para estudiar como estaban distribuidos los Motifs de *E coli* en modulos identificados previamente utilizando información topológica y biológica.

2.2. Redes ecológicas. A continuación, se utilizó el algoritmo para estudiar redes alimenticias, en estas, cada nodo representa un grupo de especies. El término especies

se refiere a especies tróficas que son grupos funcionales que comparten los mismos predadores y presas en una red de alimentos. Las aristas son dirigidas de un nodo representando al predador a un nodo representando su presa. Se analizó una de las redes alimenticias empíricas mas grande y más utilizadas en investigación[**54**], éste es un hábitat terrestre en la isla de San Martín. El Motif encontrado es el Motif BP (m66). Este Motif indica que dos especies que son presas de un mismo predador tienden a compartir las mismas presas.

Además de los Motifs, en las redes ecológicas también se encontraron anti-Motifs: subgrafos que aparecen en un número significativamente mayor en las redes aleatorias que en las redes reales. Los anti-Motifs representan restricciones impuestas en la topología de la red. En la figura **??** se muestran los anti-Motifs encontrados en la red ecológica del habitat de San Martín. El anti-Motif FBL(m11) indica que en las redes ecológicas se prohibe el hecho de que una especie sea predada por la presa de una de sus presas. En las redes ecológicas el flujo de energía es en un sólo sentido. El anti-Motif FFL4 (m52) indica que se prohiben las interacciones entre especies que están a más de dos niveles de separación (los omnívoros son especies que tienen presas en dos niveles distintos).

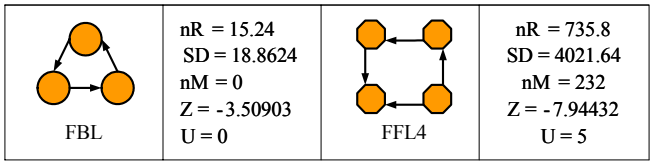


FIGURA 9. Anti-Motifs -subgrafos que aparecen en un número significativamente mayor en las redes aleatorias que en las redes reales- encontrados en la red ecológica de Sn. Martin

2.3. Redes electrónicas. Se analizaron redes creadas artificialmente, con el objetivo de contrastar con las redes creadas naturalemnte. Se analizó el estándar de comparación ISCAS89[**78**] de un conjunto de circuitos de lógica secuencial con una función específica. Los nodos en los circuitos representan compuertas lógicas y flip-flops. Estos nodos están enlazados por aristas dirigidas. Las redes se encuentran en [**79**].

Los Motifs encontrados para el circuito s9234 se muestran en la también en 8. Observese que los Motifs básicos, son los mismos que en la red de *E coli*, lo cual, posiblemente se debe al hecho de que ambas redes están construidas para procesar información. Aunque es cierto que ambos sistemas manejan información, es díficil pensar que los principios de diseño utilizados para ambos sea el mismo. Es por esto que en el siguiente capítulo de este trabajo se describe una nueva metodología (MIM) para estudiar como estan organizados estos Motifs en niveles organizacionales mayores.

Le metodología modificada fue utilizada en [**A**] para estudiar como estan organizados los Motifs dentro de modulos identificados métodos distintos.

Con la modificación a la metodología de identificación de Motifs descrita en este trabajo se identifica un nuevo Motif en el caso de las redes de regulación transcripcional, el nuevo Motif encontrado es el mismo que se encuentra en las redes electrónicas reforzando así la hipótesis de que ambas redes manejan información. Además se identificaron más instancias para cada uno de los Motifs previamente encontrados.

CAPÍTULO 6

Topología de interconexión entre Motifs de redes

Resumen. Se desarrolla una metodología completamente nueva para estudiar como estar organizados los Motifs de redes dentro de las redes complejas. Para probar la metodología se analizan las redes de regulación transcripcional de la bacteria E coli, el hongo S cerevisiae, las redes de los circuitos electrónicos y la red neuronal del gusano C elegans. Se escogieron estas redes porque todas comparten los mismos Motifs de redes. Se observa que la forma en como están organizados los Motifs dentro de cada red es muy distinto, por ejemplo, la diferencia entre redes naturales y redes artificiales es muy clara.

Se mencionó que una de las hipótesis más importantes acerca de los Motifs de redes es que estos pueden ser los bloques de construcción básicos de una red, y por lo tanto reflejan la funcionalidad de la misma: redes de circuitos electrónicos, redes neuronales y redes géneticas comparten los mismos Motifs: FFL, BF y BP, debido a que la funcionalidad de estas redes es el manejo de información, a diferencia de las redes ecológicas en las que su funcionalidad consiste en facilitar el flujo de energía desde abajo hacia arriba en las cadenas alimenticias[**72**]. Además, se ha mostrado que los Motifs en las redes de regulación genética, no han surgido por el mecanismo de duplicación genética, por lo que se cree que existe una presión evolutiva para la formación de tales estructuras[**92**].

Sin embargo, sabemos que las redes naturales tienen propiedades diferentes a las redes artificiales:

- Las primeras son robustas ante fallas de alguno de sus componentes y ante ciertas modificaciones en el ambiente, mientras que el funcionamiento de las redes artificiales se pierde si uno de los componentes falla.
- Los mecanismos para la formación de las redes naturales y artificiales es muy distinto: en las redes naturales es mediante un proceso evolutivo, mientras que en las artificiales es un proceso de diseño.

Por otro lado, existe una controversia entre una vista localista en la función de las redes complejas, contra una vista holística. La primera enfatiza en la especificidad y modularidad de la organización de las redes, mientras que la última acentúa la función global. Esta controversia refleja propiedades contrastantes que, posiblemente, no son excluyentes sino que coexisten en las redes complejas, por ejemplo, en el cerebro de los grandes vertebrados: la segregación funcional de diferentes regiones del mismo y su integración en la percepción y el comportamiento[**80**]. Así, la organización en las redes complejas no puede terminar en un nivel de Motifs.

El siguiente paso es estudiar como se estructuran los Motifs dentro de las redes. En este trabajo se describe una nueva metodología para estudiar la forma en que están interconectados los Motifs de redes. Se observa una gran diferencia entre las redes

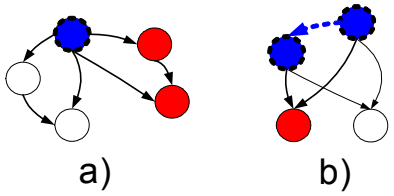


FIGURA 1. Superposición entre pares de FFL. a) Superposición débil b) Superposición fuerte

naturales y artificiales. Por último se utiliza la metodología para la identificación de módulos en redes complejas.

1. Metodología para estudiar la topología de interacción entre Motifs

Hemos visto que la herramienta matématica más conocida y poderosa para estudiar topoligia de interacción es la teoria de grafos. Así la idea básica para el estudio de la topología de interacción entre Motifs es visulizar a estos como elementos básicos o nodos de un nuevo grafo en el que las aristas reflejen la forma en que los Motifs estan organizados en la red bajo estudio. La aproximación seguida para estudiar los patrones de interconexión entre Motifs es entonces la siguiente:

Metodología para estudiar la topologia de interacción entre Motifs
Entrada: Grafo $RR = (V, E)$ con la red real a estudiar

1. Se utiliza la metodología MBM (descrita en el capítulo anterior) para identificar los Motifs de redes.
2. Se construye una nueva red RI, donde los nodos son cada una de las instancias de los Motifs identificados y existe una arista entre cada par de nodos si las instancias de los Motifs que definen a los nodos estan superpuestas fuertemente. Definimos la superposición entre cada par de Motifs (sub-grafos) $G_A = (V_A, E_A)$ y $G_B = (V_B, E_B)$ como el sub-grafo $S_{AB} = G_A \cap G_B = (V_A \cap V_B, E_A \cap E_B)$; con esta definición se puede identificar dos tipos de superposición: una superposición débil o superposición de nodos es cuando $E_A \cap E_B = \varnothing$ y una superposición fuerte o de aristas es cuando $E_A \cap E_B \neq \varnothing$. En la Figura 2 b) se muestra la red de Motifs obtenida a partir de una sub-red tomada de la red de regulación de E coli que consiste de cuatro Motifs FFL mostrados en la Figura 2 a).
3. Calcular el coeficiente de agrupamiento, el camino característico e identificar los componentes principales de RI.

En la figura 3 se muestra de manera esquemática la metodología para estudiar la topología de interacción entre Motifs.

2. Resultados

En las Figuras 4, 5 y 6 se muestran las redes de interacción entre Motifs FFL para la red de regulación de E coli, la red neuronal de C elegans y la red de circuitos

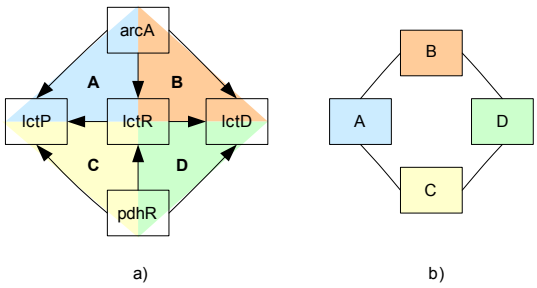


FIGURA 2. a) Se muestra la sub-red de E coli que contiene a los genes arcA, lctP, lctR, lctD y pdhR. Esta subred esta conformada por cuatro instancias del motivo FFL etiquetadas A, B, C, y D. b) Red de interacción entre motivos. La instancia A interactúa con la instancia B y C por que hay una superposición fuerte entre ellas (A y C comparten la arista lctR→lctP y A y B comparten la arista arcA→lctR), pero no interactúa con la instancia D por que no hay superposición fuerte (No comparten ninguna arista).

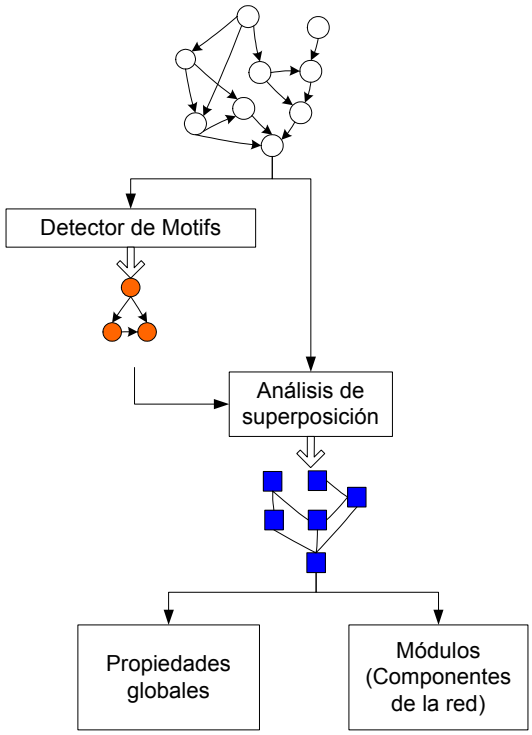


FIGURA 3. Metodología para estudia la topología de interacción entre motifs de redes.

electrónicos s9234 respectivamente. Es evidente la gran diferencia que existe entre las redes naturales y las redes artificiales: en las redes artificiales Figura 6 no existe ninguna interacción entre ninguno de los Motifs identificados en la red a diferencia del gran número de interacciones que hay en las redes biológicas. Probablemente, esto se debe a la forma en que se construyen estas redes: en las redes electrónicas existe un diseño

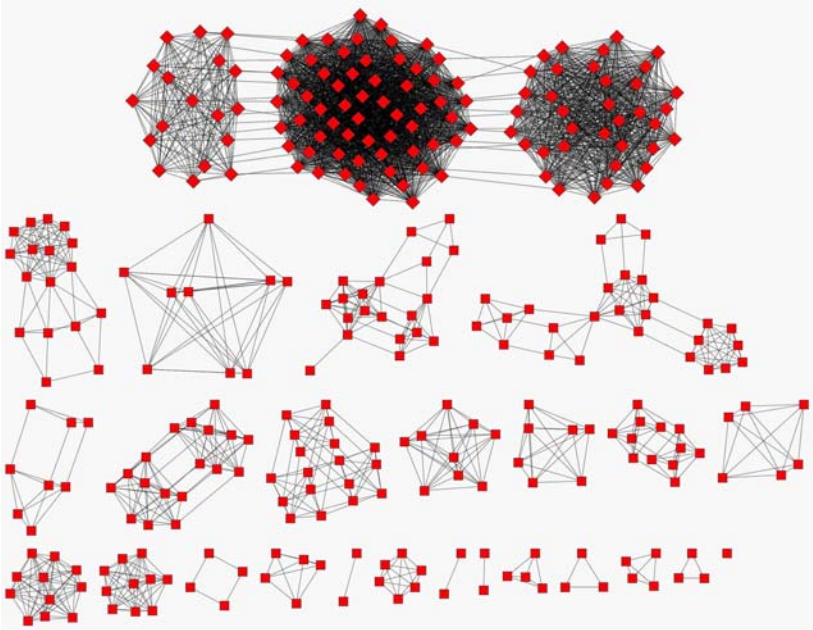


FIGURA 4. Red de interacción entre motivos FFL para la red de transcripción genética de E coli.

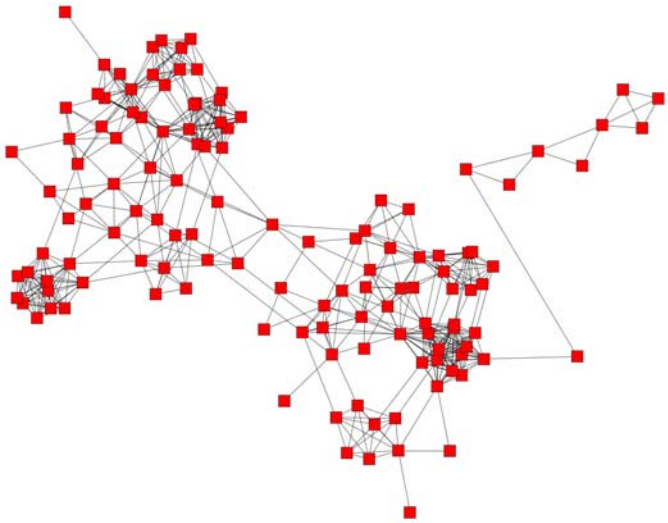


FIGURA 5. Red de interacción entre motivos FFL para la red neuronal del nematodo C elegans.

previo, y se sabe de manera precisa el papel que ocupará cada uno de los componentes, si se necesita una nueva función, se puede insertar a la red una sub-estructura con elementos completamente nuevos predefinidos para dicha función; por el contrario en una red de regulación, no se sabe de antemano el papel que desempeñara cada uno de los genes, y en este caso, si se necesita una función nueva, es más fácil desde un punto de vista evolutivo, tratar de reutilizar alguna parte de la red que extender una sub-red completamente nueva.

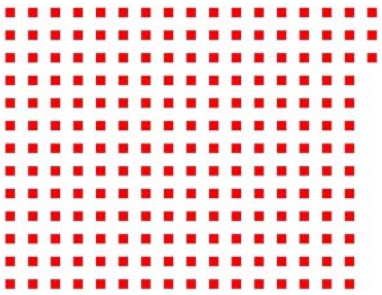


FIGURA 6. Red de interacción entre motivos FFL para la red del circuito electrónico s9234

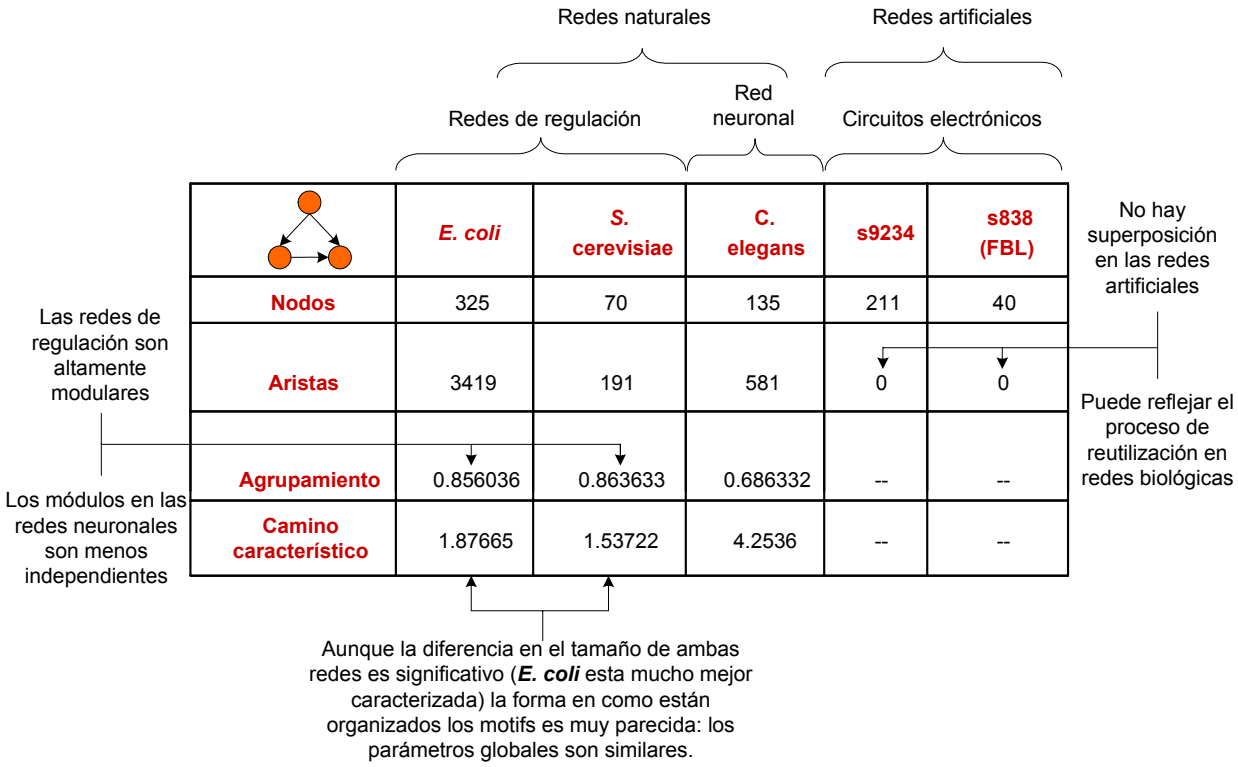


FIGURA 7. Se muestran las propiedades topologicas calculadas en cada una de las redes de interacción entre el motif FFL.

Por otro lado, también se observan muchas diferencias en la interacción de Motifs entre la red de regulación y la red neuronal, por ejemplo, el coeficiente de agrupamiento en las redes de interacción de Motifs de regulación es mayor que en la red de Motifs de red neuronal, ver tabla de la figura 7. Esto se debe a que en la primera los Motifs están en grupos aislados (la red de interacción entre Motifs esta formada por muchos componentes) y por tanto altamente modulares, a diferencia de la red neuronal en la que los Motifs están interaccionando en una red conexa de un sólo componente significativo. La metodologia y resultados parciales fueron publicados en [**B**] Así vemos que los posibles módulos son más independientes en las redes de regulación que en las redes neuronales.

2.1. Crecimiento de redes. Los procesos de crecimiento en las redes naturales y en las redes artificiales son muy distintos. En los sistemas artificiales o de ingeniería, la lógica se impone. Por razones económicas obvias, el diseño es por mucho, el mayor componente en la mayoría de los esfuerzos técnicos de la sociedad moderna. En general, un ingeniero supone que la interacción entre componentes debe de ser lo más simple posible, que no hay interacciones innecesarias o no planeadas, que hay una asignación explícita o función para cada parte del mecanismo, y que la corrección de errores se lleva a cabo por retroalimentación u otros paradigmas de la teoría de control. La protección puede ser lograda mediante una redundancia planeada, pero una compensación no planeada no es esperada ni usual. La irrelevancia es desechada desde el principio.

En contraste, el proceso en la formación de las redes naturales, como las redes genéticas es el proceso evolutivo, en donde no hay un diseño, el término irrelevante no tiene un significado a priori. Es posible que cualquier cambio en alguna parte, contribuya a la función global, las mutaciones pueden incitar una compensación, las interacciones estocásticas con el ambiente pueden dar lugar a la selección de una parte del sistema, frecuentemente no hay una asignación fija o exclusiva para una función dada, y a diferencia del caso de ingeniería, las interacciones se hacen cada vez más complejas[**82**]. En este sentido, se dice que el proceso evolutivo actúa como un 'ajustador'[**83**] que sabe lo que quiere producir, pero que esta limitado por la organización biológica de la red, así como también de los accidentes históricos.

Cuando se analiza la forma en que están estructurados los Motifs dentro de las redes biológicas se observa un re-uso de diferentes partes en la formación de los diferentes Motifs. Por ejemplo, cuando se analizó *Ecoli*, muchas veces se observó que uno o más genes participan en la formación de diferentes Motifs. En la Fig. 2 se muestra como los genes crp y fis participan en la formación muchas instancias distintas del Motif FFL. Por el contrario, en las redes electrónicas este tipo de superposición rara vez fue observado. Para cuantificar el grado de superposición para cada Motif en una red, se utilizo la siguiente medida:

$$S = 1 - \frac{U}{nM}$$

donde U es el número de veces que aparece un Motif en la red real con un conjunto completamente distinto de nodos y nM es el número total de Motifs. Así, cuando $U = nM$ ningún elemento de la red se reutiliza o no hay superposición $S = 0$. El valor de S se aproxima a 1 conforme el número de Motifs superpuestos aumenta. Sin embargo, S no toma en cuenta las diferentes formas de superposición. Por tal motivo, también se utilizó la medida:

$$S_C = 1 - \frac{nC}{nM}$$

donde nC es el número de componentes obtenidos en la red de interacción entre Motifs, por lo cual esta influenciado por la forma de superposición. En la Fig 8 se muestra el valor de S y SC utilizando "superposición fuerte" para los Motifs encontrados en las redes analizadas anteriormente. La diferencia en S en las redes naturales y las redes artificiales es clara.

Motif	E. coli (Genes)		E. coli (operones)		s838		s9234	
	S	S_C	S	S_C	S	S_C	S	S_C
 Motif 7	0.95	0.92	0.88	0.71			0.49	0.0
 Motif 19	0.997	0.995	0.998	0.993	0.63	0.32	0.89	0.88
 Motif 11					0.0	0.0		
 Motif 66								0.26

FIGURA 8. Superposición en redes complejas

2.2. Modularidad. La identificación y caracterización de los aspectos a nivel de sistema en la organización de los diferentes sistemas complejos es indispensable, por ejemplo en la biología celular post-genómica [**84**][**85**]. El concepto de modularidad, supone que la funcionalidad de los sistemas complejos puede ser particionada en una colección de módulos. Cada módulo es una entidad discreta compuesta de muchos componentes elementales y ejecuta una tarea identificable, separable de las funciones de otros módulos [**84**]. Esta noción de módulo es útil, sólo si involucra a una pequeña fracción de todos los componentes de un sistema para completar una función relativamente autónoma. Sin embrago, la identificación de los elementos que forman un módulo no está clara. Una indicación de esto es la naturaleza libre de escala en la distribución del grado en redes complejas[**32**]. Una característica de tales redes es la existencia de pocos nodos altamente conectados. Con un gran número de enlaces estos 'hubs' integran a todos los elementos en una sola red integrada en la que la existencia de módulos totalmente separados es prohibida por definición[**81**]. Sin embargo también se ha visto que muchas redes poseen un alto coeficiente de agrupamiento, propiedad que sugiere organización modular.

Con los Motifs se pretende encontrar dichos módulos funcionales. Una primera aproximación es utilizar a los diferentes componentes del grafo de interacción entre Motifs como diferentes modulos del sistema. En las figuras de 1 a 11 del Apéndice A se muestran los módulos encontrados en la red de regulación genética de la bacteria E coli. Para evaluar la significancia biológica de cada uno de los módulos identificados, se buscó la anotación funcional biológica conocida para cada uno de los genes que conforman los diferentes módulos. Se observa que cada uno de los módulos corresponde a una función biológica bien conocida, reforzando la hipótesis de que los Motifs de redes son los bloques básicos para la funcionalidad de la red compleja. Para hacer un estudio mas detallado se pueden utilizar técnicas de clustering al interior de cada uno de estos módulos. Estos resultados se presentaron en [**B**].

La nueva metodología presentada en este capítulo mostró que aunque las redes de regulación trascripcional de la bacteria E. coli y del hongo S. cerevisiae, la red neuronal del gusano C. elegans y la red de circuitos electrónicos tiene exactamente los mismos Motifs, la forma en que estos están organizados dentro de la red es completamente distinta. Primero, en las redes electrónicas cada una de las instancias del Motif FFL esta completamente aislada una de la otra, a diferencia de las redes biologías en la cual los mismos Motifs FFL están interactuando unos con otros. Y segundo, en las redes de regulación se encuentran muchos grandes aglomerados de Motifs (reflejado por el alto índice de agrupamiento), a diferencia de la red neuronal donde solo se observa un solo componente.

CAPÍTULO 7

Simulación de la red de regulación transcripcional

Resumen. Se utiliza CSDK para estudiar el papel dinámico que tienen los Motifs de redes en la red de regulación de la bacteria E coli. Si simula el Motif FFL utilizando el modelo de regulación tradicional y asumiendo que las proteínas hacen una caminata al azar dentro de la célula. Los resultados de las simulaciones siguieren que una de las funciones de la topología FFL es la de ayudar a aumentar el nivel de expresión de un gene cuando este se encuentra distante de su regulador dentro de la célula. Para probar la hipótesis sugerida se estudian las distancias entre gene regulador y gene regulado en el genoma real de la bacteria E coli, observando que la probabilidad de que el par de genes este cerca uno del otro es muy grande, como se esperaría por la hipótesis.

En esté capítulo se utilizan CSDK para estudiar el papel de el Motif FFL en la dinámica de las redes de regulación genética.

Los organismos biológicos desde las bacterias hasta los humanos poseen un enorme repertorio de respuestas genéticas a combinaciones de señales celulares y del ambiente. A grandes rasgos, este repertorio esta codificado en redes complejas de genes, cada uno regulando la actividad de otro. Sin embargo, a diferencia de los circuitos integrados que procesan la información a través de cascadas sincronizadas de muchos nodos sencillos y rápidos y donde la topología de interconexión es la principal fuente de complejidad dentro de los circuitos, una red de regulación genética típicamente consiste de sólo unos cuantos centenares de genes cuya expresión es lenta y asíncrona. Pero estos "nodos" son muy sofisticados en su capacidad para integrar señales: la función de control regulatoria puede ser extremadamente complicada[**95**]. A continuación se describirá un modelo, utilizando CSDK, para comenzar a estudiar el papel dinámico de las diferentes estructuras significativas (Motifs de redes) dentro de una red de regulación.

1. Modelo

A continación, se decriben a grandes razgos los principales componentes en los mecanismo de la regulación (Para una descripción detallada de lso ménismos de regulación conocidos cunsultar [**96**]). La actividad de un gene es regulada por otros genes a través de la concentración de sus productos genéticos -proteínas originadas a partir de la expresión de cierto gene- denominados factores de transcripción (FT). Esto es logrado mediante la interacción mecánica de los FT con la proteina conocida como RNA polimeraza (RNAP) en la región regulatoria del gen regulado. Estos FT pueden ser activadores o represores de transcripción, o incluso ambos [**96**]. Estudios recientes han mostrado que la función de una proteína regulatoria esta determinada principalmente por la posición del sitio de pegado en el DNA relativo al inicio de la transcripción[**97**]. Los activadores estan lejos del inicio de transcripción, mientras que los represores está

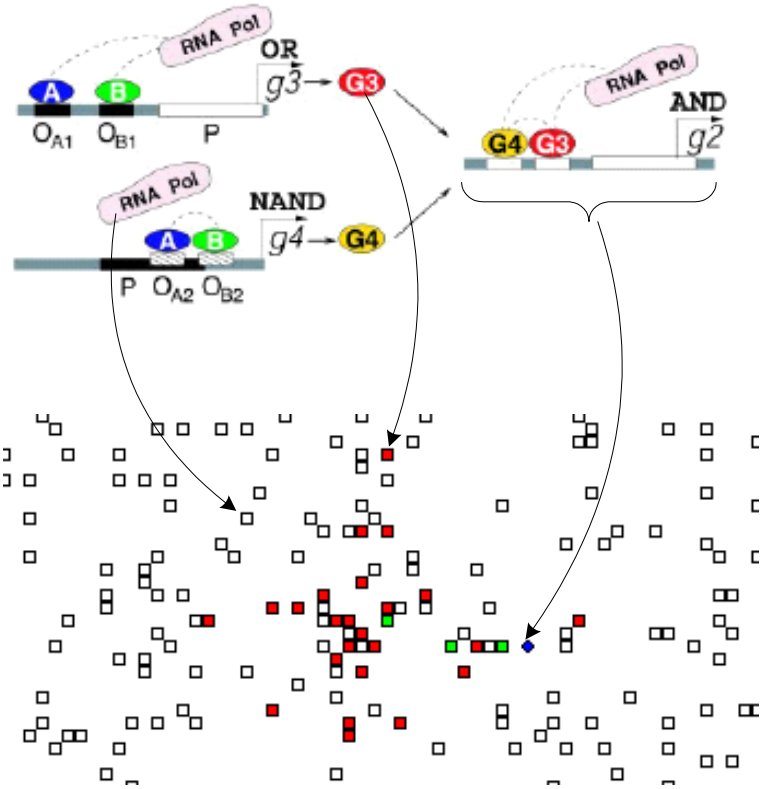


FIGURA 1. Regulación transcripciónal. Las flechas indican los elelmetos en la simulación que representan a cada uno de los participantes en la regulación transcripcional

cerca del inicio de transcripción, en la figura 1 se muestran cada uno de estos componentes. A continuación se describen más detalladamente cada uno de estos componentes junto con su implementación en CSDK.

1.1. Gene. Un gen esta caracterizado por la secuencia codificadora –secuencia del DNA que contiene la codificación para la expresión de una proteína- y por la secuencia promotora –secuencia del DNA adyacente a la secuencia codificadora donde se encuentra codificados sitos de pegado para factores de transcripción y para la RNAP. Uno de los problemas más importantes en biología molecular es determinar la estructura terciaria de una proteína a partir de la secuencia codificadora; otro problema igualmente importante es determinar el sitio que reconoce una proteína dada su estructura terciaria. En este modelo se especifican los mapeos arbitrariamente de manera que se cumplan las relaciones de regulación especificadas en un archivo de entrada y que en este estudio son las relaciones identificadas experimentalmente.

En la figura 2 se muestra el código de la clase Gene. Las variables *sec* y *secProm* almacenan las secuencias codificadora y promotora respectivamente. La clase tiene métodos para generar ambas secuencias aleatoriamente (*generateRandomSecuence* y *generateRandomPromoter*) para agregar un sitio de pegado en la secuencia promotora que sea activador o represor (*setBindingSite*). La variable *dT0* es la variable que indica el tiempo de degradación que tienen las proteínas codificadas por éste gen. La variable

Código 6.1

```
class Gene : public PC2Dc{
private:
      static UniformRV base;
public:
      Gene();
      ~Gene();
      Gene(char *sec, GeneProteinMap *ma);
      void setPromoter(char *sp, int ip);
      void setMap(GeneProteinMap *m);
public:
      void nextState(); //Se computa el estado siguiente
      void refresh();
public:
      int l; //lenght of the operator secuence
      char *secProm; //secuense
      bool *useProm;
      int p; //position where the promoter begins
      int id; //identificator of the Gene
      bool active; //The gene is active or pasive
      char *sec; int lsec;
      int dT0; //Tiempo de degradación de las proteinas
public:
      int match(Protein *p); //maximal match in the
      void express();
      GeneProteinMap *m;
public:
      void generateRandSecuence(int lenght);
      void generateRandomPromoter(int len,int pr);
      void generateFixPromoter(int len,int pr);
      void setBindingSite(char *bind, int k, bool kind);
public:
      void printPrometerUse(char *file);
      void setGarbage(LinkContainer *gar);
      static LinkContainer *garbage;
};
```

FIGURA 2. Codigo con la declaración de la clase *Gene*

binaria *active* determina si el gen puede expresar proteínas sólo con la RNAP, cuyo valor es uno, o si necesita de al menos un activador par su expresión, cuyo valor es cero. La variable p indica la posición que determina si un sitio de pegado es activador (cuando el sitio esta antes de ésta posición) o represor (cuando el sitio de pegado esta después de la posición). Además la clase tiene métodos para asignar valores arbitrarios a cada uno de las variables mencionadas anteriormente. Finalmente, la clase contienen los métodos *nextState* y *refresh* especificados por CSDK.

En las figuras 3 y 4 se muestra la implementación del método *nextState*. La primera parte del método figura 3 se encarga de determinar cuales son los elementos que se encuentran en el gen actualmente, es decir si hay activadores o represores pegados, y si hay una RNAP. La segunda parte, figura 4, utiliza la información obtenida para determinar si el gen expresara una proteína o no untilizando el método express. Un gen con $active = 1$ se activa si no hay represores y hay una RNAP presente, un gen con $active = 0$ se activa sólo si hay activadores presentes, no hay represores y hay una RNAP presente. Finalmente en la figura 5 se muestra la implementación del método match que es utilizado por los elementos de la clase *Protein* para determinar si se van a pegar al gen o no. En el ciclo se busca por sitios de pegado, una proteína puede tener varios sitios de pegado dentro de la misma región promotora, es por esta razón que un gen puede ser tanto activador como represor. En la segunda parte del método se escoge al azar uno de los sitios de pegado encontrados y se indica a la proteína la

Código 6.2

```
void Gene::nextState()
{
      int i=0,j,k;
      Protein *fp;
      bool mrna = false;      bool repress = false;
      bool activator = false;
      ProcessingCore *p;
      double s;
      int nP;     int cond=0;
      int **dM;
      Protein **Ps;

      nP=lattice->lattice[x][y].card()-1;
         Ps = new Protein*[nP];

      k=0;
      lattice->lattice[x][y].initialize();
      for(i=0;!lattice->lattice[x][y].end();i++){
            lattice->lattice[x][y].iterate(&p);
        if(p != this){
                  if(typeid(Protein) == typeid(*p)){ //proteina
                        Ps[k] = (Protein*) p;
                        if(Ps[k]->tempd>=0){
                              if(Ps[k]->tempd+Ps[k]->r < this->p)
                                    activator = true;
                              else
                                    repress = true;
                        }

                  }
               if(typeid(mRNA) == typeid(*p)){ //ve un gen
                        mrna = true;
                  }
               }
      }
```

FIGURA 3. Primera parte de la implementación del método *nextState* de la clase *Gene*

Código 6.3

```
//Falta mecanismo de cooperación
      if(mrna){
            if(active && !repress)
                  express();
            if(!active && !repress && activator)
                  express();
      }

      delete[] Ps;
}
```

FIGURA 4. Segunda parte de la implemetación del metodo *nextState* de la clase *Gene*

fueraza de pegado (es proporcional al numero de pares de base que son reconocidos por la proteina).

1.2. Proteína. Una vez que una proteína es expresada esta comienza a hacer una caminata al azar en el ambiente especificado. La clase *Protein* figura 7 hereda de la clase *RandomWalk* figura 6 está utiliza las bibliotecas de numeros aleatorios contenidas en CSDK para realizar la caminata al azar. La clase *Protein* contiene variables para especificar el tiempo de degradación de la proteina el cual se va disminuyendo en uno con cada paso de tiempo, y para determinar el tiempo en que se mantiene pegado a uno de los genes.

Código 6.4

```
int Gene::match(Protein *p)
{
      int i,j; int k;
      BindA ba;
      int minB = p->r; int cT = 10;
      GenericList<BindA> Bs;

      for(i=0;i<l-p->r;i++){
            k=0;
            for(j=0;j<p->r;j++){
                  if(secProm[j+i]==p->binds[j] && !useProm[j+i])
                        k++;
            }
            if(k>=minB){
                  ba.p=i; ba.st=k;
                  Bs.insert(ba);
            }
      }
      if(Bs.size()!=0){
            UniformRV sel;
            sel.setN(Bs.size());
            sel.normalize(0,Bs.size()-1);

            k = sel.random();
            Bs.initialize();
            for(i=0;i<=k;i++)
                  Bs.iterate(&ba);

            p->tempd = ba.p;
            for(i=0;i<p->r;i++)
                  useProm[p->tempd+i]=true;
            p->stick = this;
        return ba.st*10;
      } else
            return 1;
}
```

FIGURA 5. Implementación del método *match* de la clase *Gene*

Código 6.5

```
class RandomWalk: public PC2Dc
{
public:
      RandomWalk();
      virtual void nextState()=0;
      virtual void refresh()=0;
protected:
      static UniformRV Walk;
      void randomWalk();
      bool isThereWall(int xp,int yp);

};
```

FIGURA 6. Declaración de la clase *RandomWalk*

En la figuras 8 y 9 se muestra la implementación del método *nextState*. Lo primero que hace el método es decrementar el contador del tiempo de vida de la proteína, después se verifica si la proteína ya estaba pegada a un gen, en tal caso se determina si seguirá pegada o no en base a la variable k figura 8. Si la proteína se va a despegar entonces se llama a la función randomWalk para comenzar la caminata al azar. En la segunda parte del método figura 9 se verifica si durante la caminata, la proteína llego a un gen en el que hay un sitio de pegado para ella.

Código 6.6

```
class Protein : public RandomWalk{
public:
      void nextState(); //Se computa el estado siguiente
      void refresh(); //Se transmite la información hacia el nuevo
      //estado

public:
      Protein();
      void setDegradationTime(int deg);
public:
      int r;
      char *binds;

      int k; //time while the protein sticks in the gene
      int newk;

      int dT0; //degradation time of the protein
      int dT;
      int newdT;

      //int k0; //Binding strenght
      int id; //identificator of the protein

public:
      int tempd; //es solo una optimizacion de implementacion
      Gene *stick;
public:
      static LinkContainer *garbage;
};
```

FIGURA 7. Declaración de la clase *Protein*

Código 6.7

```
void Protein::nextState()
{
      int i=0;
      int s;
      Protein *fp;
      Gene *g;
      ProcessingCore *p;

      newdT = dT-1;
      if(k>0){
            newk=k-1;
            if(newk==0){
                  if(tempd !=-1){
                        for(i=0;i<r;i++)
                              stick->useProm[tempd+i]=false;
                        tempd = -1;
                        stick = NULL;
                  }
                  randomWalk();
            }
            else{
                  newx = x;
                  newy = y;
            }
            return;
      }
```

FIGURA 8. Primera parte de la implementación del método *nextState* de la clase *Protein*

Código 6.8

```
lattice->lattice[x][y].initialize();
      for(i=0;!lattice->lattice[x][y].end();i++){
            lattice->lattice[x][y].iterate(&p);
        if(p != this){
                  if(typeid(Gene) == typeid(*p)){ //ve un gen
                        g = (Gene*) p;
                        s = g->match(this);
                        newk = s;
                        newx = x;
                        newy = y;
                        return;
                  }
            }
      }
      randomWalk();
}
```

FIGURA 9. Segunda parte de la implementación del método *nextState* de la clase *Protein*

Código 6.9

```
void Protein::refresh()
{
      lattice->lattice[x][y].remove(this);

      if(newdT==0){
            vsd->remove(this);
            if(tempd !=-1){
                  for(int i=0;i<r;i++)
                        stick->useProm[tempd+i]=false;
                  tempd = -1;
                  stick = NULL;
            }
            garbage->insert(this);
            //Poner en cola de basura para su futura eliminacion
            return;
      }
      dT = newdT;

      x = newx;
      y = newy;
      lattice->lattice[newx][newy].insert(this);

      k = newk;
}
```

FIGURA 10. Implementación del método *refresh* de la clase *Protein*

Finalmente en el método *refresh* (figura 10) verifica si el tiempo de vida ha terminado en cuyo caso elimina la proteína del sistema. Sino hace las actualizaciones correspondientes.

La clase *CellObserver* es la encargada de interconectar todos los elementos del sistema. La función de vecindad (el ambiente) será una latice bidimencional regular. La dinámica temporal será sincrona –la asincronía de la red de regulación antes mencionada esta implícitamente en el modelo debido a las caminatas al azar. También es la encargada de inicializar las secuencias promotoras y codificadoras, de modo que cumplan con las características descritas en el modelo a estudiar (En el apendice 1 se muestra el manual de usuario de este modelo en particular junto con el formato del archivo con la especificación del modelo). El codigo con la implementación de los métodos de

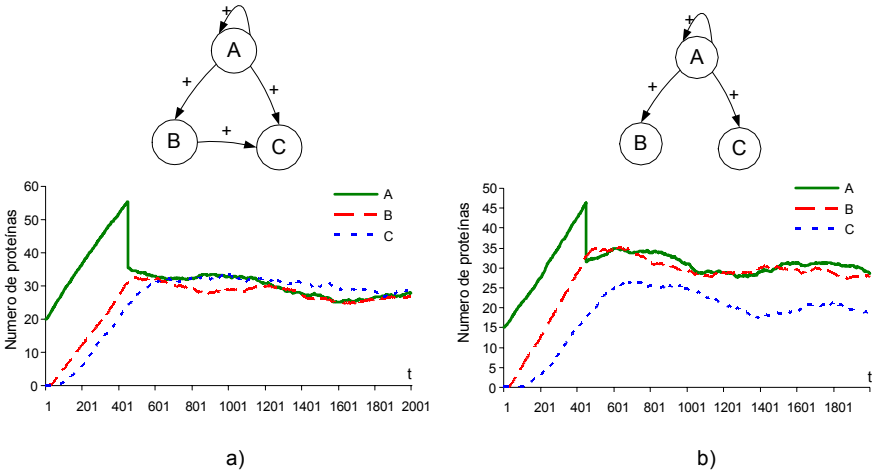


FIGURA 11

CellObserver pueden consultarse en el CD que acompaña a la tesis junto con el código de todos los ejemplos dados anteriormente.

2. Resultados

Se utilizará el simulador para estudiar la dinámica de la red de regulación en general, y el Motif FFL en particular. Para todos los ejemplos estudiados se asume que la tasa de degradación y la afinidad de pegado a cada uno de los sitios es la misma. En la figura 11 a) se muestra una red genética con la topología de un FFL más una relación de auto-regulación en el gen A la cual es encontrada en la mayoría de los FFL identificados en la red de regulación de *E coli*. Todas las interacciones de esta red son positivas, es decir los factores de transcripción activan la expresión de los genes regulados y todos los genes son pasivos (necesitan de al menos un activador para expresarse). Los genes están posicionados en una línea a una distancia de 6 (distancia de latice) uno del otro. Finalmente la densidad de RNAP es de 1/12 de polimeraza por elemento de latice (es decir si se toman 12 elementos de la latice al azar se debe encontrar una RNAP). La condición inicial utilizada es de 20 proteinas del gen A y ninguna de los otros dos. En la grafica de la figura 11 a) se muestra el promedio de 50 realizaciones de dicha simulación. Se observa que los genes B y C alcanzan un nivel de expresión muy similar. En la figura 11 b) se muestra la misma red genética pero sin la interacción que va del gen B al C y la gráfica que muestra el promedio de 50 realizaciones con las mismas condiciones iniciales que en a). En esta a diferencia que a) se observa que el nivel de expresión del gen C es mucho menor que el de B. Este resultado siguiere que una de las funciones de la topología FFL es la de ayudar a aumentar el nivel de expresión de un gen cuando este se encuentra distante de su regulador dentro de la célula.

Se encuentra un resultado análogo para un FFL donde las interacciones hacia el gen C son negativas. Con el FFL es posible la represión del gen activo C, mientras que sin la topología FFL esta represión es menos eficiente (Figuras 12 a) y b) respectivamente).

Estas simulaciones también hacen evidente el hecho de que la dinámica de la red de regulación esta altamente influenciada por factores diferentes al de la topología. Si uno toma los genes de un organismo junto con su secuencia regulatoria y hace una

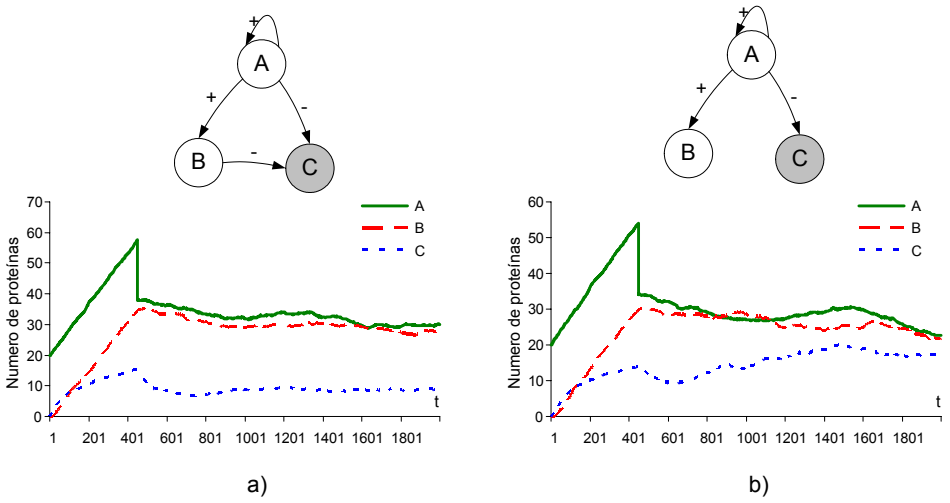
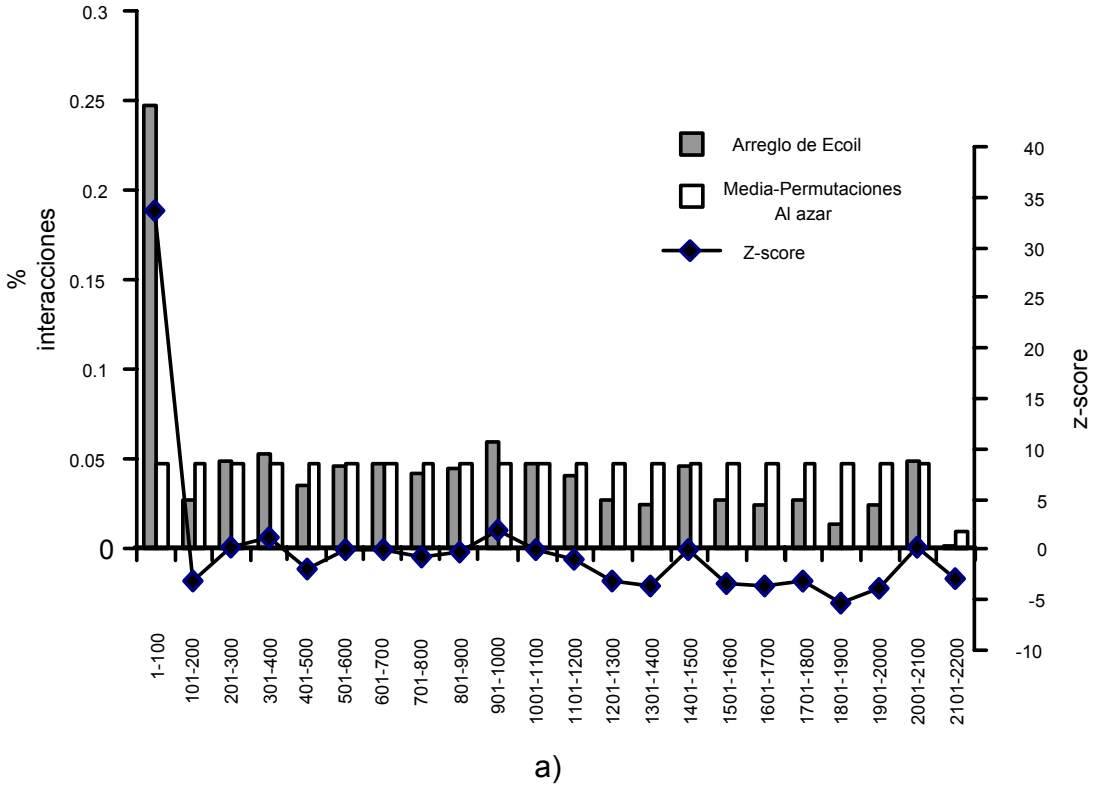


FIGURA 12

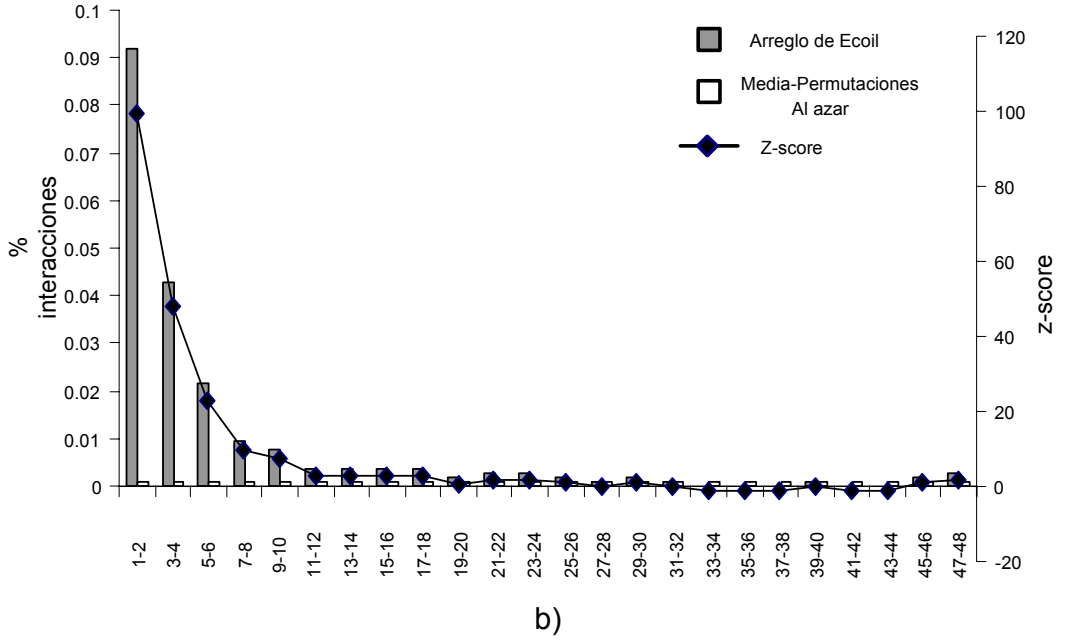
permutación al azar de los mismos, entonces en principio la topología de la red de regulación debe de ser exactamente la misma, sin embargo la dinámica y por lo tanto todo el comportamiento puede cambiar drásticamente debido a que la distancia que tiene que viajar una proteína dentro de la célula para interactuar con el gen regulado cambia con dicha permutación. Así los FFL pueden ser un mecanismo que utiliza la célula para regular genes distantes a través de un gen intermedio.

Si esta hipótesis es correcta entonces la evolución debería de arreglar el orden de los genes en el genoma de manera tal que las relaciones de regulación estén lo más cercano posible. Para probar esta aseveración se obtuvo la distribución de distancias entre factores de transcripción y genes regulados especificados en la red conocida de la bateria *E coli*. Las distancias se calcularon utilizando el orden real de los genes en el genoma de *E coli*. Esta distribución se comparó con distribuciones generadas utilizando la misma red pero con permutaciones al azar del genoma real. Como puede observarse en la figura 13 a) y b) es claro que las distancias son bastante mas cortas en comparación con las obtenidas con las permutaciones aleatorias, mostrando una alta significancia del arreglo del genoma actual. Esta significancia se cuantificó mediante la evaluación del Z-score, mostrado también en las figuras 13 a) y b).

Se mostró el uso de CSDK para estudiar el papel dinámico del Motif FFL en las redes genéticas: una de las posibles funciones del Motif FFL es la de controlar genes distantes uno del otro dentro de la célula.



a)



b)

FIGURA 13. a) Comparación entre la distribución en las distancias intergenicas (para este estudio mediada en número de genes) entre factores de transcripción y genes regulados para la red conocida de *E coli* calculada utilizando el arreglo del genoma real y entre la distribución calculada utilizando 1000 permutaciones al azar de dicho genoma. Se calculó el z-score para cuantificar la significancia entre los valores de ambas distribuciones b) Ampliación del intervalo de distancias entre 1 y 100.

CAPÍTULO 8

Comentarios Finales y Conclusiones

1. Modelado computacional de SC

Se definió un modelo de cómputo que permite estudiar desde un punto de vista computacional el papel que juegan cada una de las características previamente identificadas en el comportamiento global o la función de los SC y se diseñó e implementó una nueva arquitectura de software (CSDK) para el desarrollo eficiente de simulaciones de SC basadas en dicho modelo.

Por otro lado, sólo se ha respondido parcialmente, como un sistema complejo dado desarrolla un comportamiento colectivo particular. Esta búsqueda apenas ha comenzado y el lenguaje computacional será de gran utilidad para describir las relaciones precisas entre un comportamiento colectivo particular y la especificación de un sistema complejo dado, por ejemplo: el algoritmo glotón en una red de mundo pequeño permite que todos los elementos de un sistema complejo se puedan encontrar unos a otros en un número pequeño de pasos, sin embargo, el mismo algoritmo glotón no logra dicho objetivo en una red aleatoria, aunque en dicha red también existan caminos cortos; la regla de mayoría para modelos de votación en una red de mundo pequeño permite al sistema llegar a un estado de decisión global, la misma regla de mayoría falla en llegar a dicho estado en una red regular.

Se ha planteado un modelo computacional para expresar dichas abstracciones (programar la regla de mayoría en cada entidad cuyas interacciones están dadas en una topología de mundo pequeño). Se ha dicho que el modelo computacional es un sistema formal en el que las demostraciones a ciertas afirmaciones son la ejecución del sistema mismo (regla de mayoría + topología de mundo pequeño implica convergencia a estado de decisión global). Sin embargo estas demostraciones son muy particulares porque son válidas sólo para el estado inicial dado. Para poder ampliar la aplicación de un resultado será necesario hablar del modelo de cómputo en términos de conceptos de dinámica no lineal, no en el sentido que se ha utilizado hasta ahora, sino en un sentido más amplio: un atractor no sólo será definido por un conjunto de puntos a los que el sistema llega cuando el tiempo tiende a infinito, también puede ser cualquier punto que cumpla con una propiedad dada, por ejemplo, una colonia de hormigas nunca va a construir un panal exactamente igual, sin embargo si va a construir un panal con ciertas propiedades específicas: la temperatura interna debe de estar entre x y y grados, etc. La propiedad se puede especificar utilizando cualquier lenguaje que hable acerca de los puntos en el espacio de estados.

Con esto en mente también se puede cuantificar la robustez de un sistema complejo para ejecutar una tarea como el tamaño del radio de atracción para el atractor en el cual el sistema realiza la función para la cual fue creado, pero ¿cómo encontrar dichos atractores y sus radios de atracción? para sistemas finitos esto se puede hacer mediante la ejecución del sistema para cada uno de los posibles estados iniciales, sin embargo,

la metodología es totalmente impráctica aún para un sistema relativamente pequeño, por ejemplo, para la red de regulación genética de la bacteria *E coli* (la red genética más pequeña) el sistema se tendría que ejecutar 2^{792} veces . Una alternativa puede ser el desarrollo de un método de Monte Carlo para aproximar los atractores y radios de atracción a partir de los resultados obtenidos de un muestreo aleatorio de condiciones iniciales. En conclusión, el desarrollo de un entorno de simulación basado en un modelo de cómputo diseñado específicamente para modelar sistemas complejos, puede ayudar tanto al desarrollo eficiente de modelos computacionales como al planteamiento de generalizaciones en términos de propiedades especificas del modelo.

2. Optimización Extrema Distribuida

El algoritmo de optimización extrema distribuida mostró como se puede hacer uso del conocimiento sobre Sistemas Complejos, en particular las dinámicas en Redes Complejas, para diseñar mejores algoritmos para la solución de problemas computacionales. Así como este, se pueden mejorar otros algoritmos, por ejemplo, el algoritmo genético seria otro candidato ideal, modificando el operador de selección para que visualice a la población de soluciones no como homogénea, sino con una topología de interacción compleja.

3. Motifs de redes

Los Motifs de redes son patrones de interconexión que ocurren en un número significativamente mayor en las redes reales que en redes aleatorias con la misma distribución por nodo. Con la modificación del algoritmo de conteo se identificaron nuevos Motifs de redes así como también más instancias de los Motifs particulares. Así como se modificó el algoritmo de conteo, también se pudo haber modificado el algoritmo para la generación de redes aleatorias para buscar por regularidades particulares, en este caso se buscaban regularidades que no tuvieran que ver con la distribución de la red, es decir identificar estructuras que no sean consecuencia de la distribución del grado. De la misma manera se podrían establecer diferentes restricciones para las redes aleatorias contra las que se esta comparando la red real, por ejemplo, comparar contra redes con un mecanismo de crecimiento particular, con el objetivo de identificar estructuras que se formaron por medios diferentes a los mecanismos básicos de crecimiento de redes. Es muy importante tomar en cuenta estos factores cuando se interpreta la estructura de los Motifs obtenidos.

4. Interacción entre Motifs

Aunque ha habido muchas expectativas en los Motifs identificados, ni la forma en que están interactuando unas con otros, ni su función en cualquiera de los sistemas es clara. La metodología presentada muestró una forma de como analizar la topología de interacción entre Motifs identificando diferencias claras entre las diferentes clases de sistemas y como estas diferencias topologícas reflejan aspectos que se cree existen en las distintas redes. Sin embargo, deja fuera a los nodos que no forman parte de los Motifs, este es un punto muy importante que debe tomarse en cuenta en un estudio posterior.

5. Simulación de la red de regulación transcripcional

Con respecto a la dinámica, el resultado obtenido simulando Motifs de redes individuales en redes genéticas dio una indicación del papel de estos en la dinámica global del sistema: el Motif FFL se utiliza para controlar genes a distancia; pero para ser más precisos se debería estudiar el papel de los mismos junto con todos los demás elementos de la red bajo estudio. Una aproximación para estudiar dicha dinámica puede ser mediante la comparación del comportamiento de un sistema que tiene Motifs contra otro que no los tenga, cuántos atractores tienen, cuál es el tamaño de los radios de atracción, etc.

Durante ésta tesis, por medio de las discusiones del grupo, los artículos y libros leídos, pero principalmente los resultados obtenidos, se ha desarrollado en mí un marco conceptual de la obtenidos, se ha desarrollado en mí un marco conceptual de la relación entre computación y los fenómenos naturales. Primero cualquier teoría sobre algún fenómeno natural puede ser utilizada como base para el desarrollo de nuevos algoritmos y de nuevos modelos computacionales para la solución de problemas como los NP-completos. El algoritmo de optimización extrema distribuida es un ejemplo de esto. Por otro, igualmente importante, es la visón computacional acerca de los sistemas físicos, por ejemplo, la física mucho tiempo se ha centrado al análisis de sistemas muy simples, sin embargo como vimos muchos fenómenos interesante surgen sólo cuando los sistemas se vuelven más grandes y complicados. Uno de los mensajes de esta tesis es que es posible desarrollar herramientas para cruzar el abismo que hay entre lo pequeño y lo relativamente complejo: el modelado computacional junto con los algoritmos desarrollados provén de una manera sistemática para la construcción y el entendimiento de tales sistemas.

6. Conclusiones Finales

- Se identificaron las características más importantes de los SC. Así cómo también sus comportamientos característicos.

- Se definió un modelo de cómputo que permite expresar en un lenguaje computacional las características de los SC identificadas.

- Se diseñó e implementó una nueva arquitectura de software (CSDK) para el desarrollo eficiente de simulaciones de SC basadas en el modelo de cómputo descrito. Se mostró la capacidad de modelado de (CSDK) mediante la reproducción de los resultados obtenidos en dos de los modelos más importantes en el área de los SC:
  - Se implementó el modelo "vida", reproduciendo el comportamiento de las estructuras conocidas como deslizadores.
  - Se implemento el modelo de esparcimiento de enfermedades sobre redes complejas, mostrando como la topología influye tanto en la velocidad de transmisión de la enfermedad como en el porcentaje de la población que esta abarca. Se reprodujeron los resultados presentados en [**51**].

- Se desarrolló un nuevo algoritmo de optimización, denominado Optimización Extrema Distribuida (OED). Con el cual se mostró el uso de la herramienta y la teoría de las redes complejas para el desarrollo de un algoritmo original para la solución de problemas de optimización. OED es una modificación sustancial del algoritmo de optimización extrema (OE) que contempla la topología de interacción. OE resulta sólo como el caso particular en que se tiene una red completamente conexa. Se utiliza el algoritmo para resolver el problema de coloreado de grafos, un problema NP-completo y se compara con el algoritmo de optimización extrema tradicional. Se mostró que OED obtiene resultados más óptimos que OE para diversas topologías, en particular cuando la red de interacción tiene una topología de mundo pequeño.

- Se modificó e implementó la metodología para la extracción de Motifs de redes[**72**] con el objetivo de estudiar más en detalle la estructura local de las redes complejas. Se probó la metodología modificada con las redes de regulación transcripcional de la bacteria E coli, el hongo S cerevisiae, dos circuitos electrónicos y la red alimenticia de la reserva Sn. Martín, todas estas utilizadas en la metodología original. Con la modificación a la metodología de identificación de Motifs descrita en este trabajo se identificó un nuevo Motif en el caso de las redes de regulación transcripcional, el nuevo Motif es el mismo que se encuentra en las redes electrónicas reforzando así la hipótesis de que ambas redes manejan información. Además se identificaron más instancias para cada uno de los Motifs previamente encontrados.

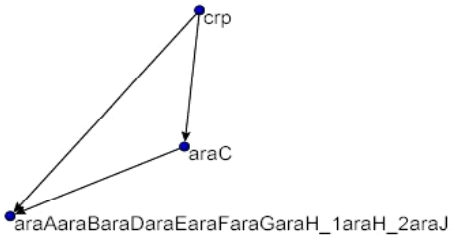
- Se desarrolló una metodología completamente nueva para estudiar como están organizados los motifs de redes dentro de cada sistema. Para probar la nueva metodología se analizarón las redes de regulación transcripcional de la bacteria E coli, el hongo S cerevisiae, las redes de los circuitos electrónicos y la red neuronal del gusano C elegans. Se escogieron estas redes porque todas comparten los mismos Motifs de redes. Se observa que la forma en como están organizados los Motifs dentro de cada red es muy distinto:
  - Primero, en las redes electrónicas cada una de las instancias del Motif FFL esta completamente aislada una de la otra, a diferencia de las redes biologías en la cual los mismos Motifs FFL están interactuando unos con otros. Lo cual muestra el amplio proceso de reutilización que se tiene en las redes biológicas: en estas es mucho más fácil, en términos de crecimiento (p. e. evolutivos), el reutilizar subestructuras ya presentes para la realización de una nueva función
  - Segundo, en las redes de regulación se encuentran muchos grandes aglomerados de Motifs (reflejado por el alto índice de agrupamiento), a diferencia de la red neuronal donde solo se observa un solo componente. Lo cual sugiere que las redes de regulación tienen una arquitectura modular mucho más definida que la arquitectura de la red neuronal. Lo cual concuerda con el conocimiento general que se tiene de ambas redes.

- Por último, se utilizó CSDK para estudiar el papel dinámico que tienen los Motifs de redes en la red de regulación de la bacteria *E coli*. Los resultados de las simulaciones siguieren que una de las funciones de la topología FFL es la de ayudar a aumentar el nivel de expresión de un gene cuando este se encuentra distante de su regulador dentro de la célula. Para probar la hipótesis sugerida se estudian las distancias entre gene regulador y gene regulado en el genoma real de la bacteria E coli, observando que la probabilidad de que el par de genes este cerca uno del otro es muy grande, como se esperaría por la hipótesis.

APENDICE A

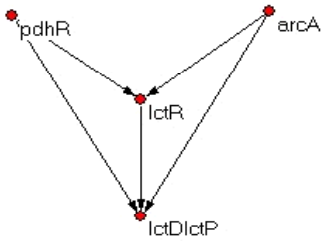
Módulos en *E coli*

Lista de módulos identificados en la red de regulación genética de la bactería *E coli*. En las figuras de 1 a 11 se muestran los módulos encontrados en la red de regulación genética de la bacteria *E coli* utilizando la metodología presentada en el capitulo VI. Para evaluar la significancia biológica de cada uno de los módulos identificados, se buscó la anotación funcional biológica conocida para cada uno de los genes que conforman los diferentes módulos (parte inferior de cada figura). Se observa que cada uno de los módulos corresponde a una función biológica conocida, reforzando la hipótesis de que los Motifs de redes son los bloques básicos para la funcionalidad de la red compleja.



araC	Degradation of small molecules: Carbon compounds
araA	Degradation of small molecules: Carbon compounds
crp	Global regulatory functions: Global regulatory functions
araB	Degradation of small molecules: Carbon compounds
araD	Degradation of small molecules: Carbon compounds
araE	Transport/binding proteins: MFS family
araF	Transport/binding proteins: ABC superfamily (peri_perm)
araG	Transport/binding proteins: ABC superfamily (atp_bind)
araH_1	Transport/binding proteins: Mechanism not stated
araH_2	Transport/binding proteins: Mechanism not stated
araJ	Transport/binding proteins: MFS family

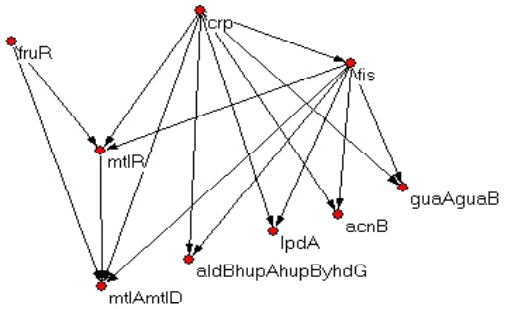
a)



arcA	Global regulatory functions: Global regulatory functions
lctD	Energy metabolism, carbon: Aerobic respiration
lctR	Energy metabolism, carbon: Aerobic respiration
lctP	Transport/binding proteins: Mechanism not stated
pdhR	Energy metabolism, carbon: Pyruvate dehydrogenase

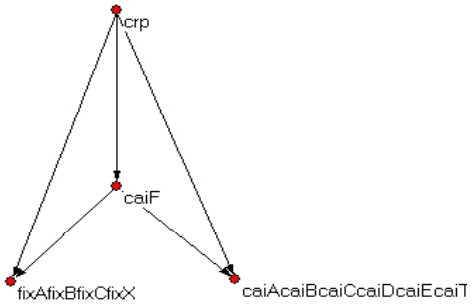
b)

FIGURA 1.



crp	Global regulatory functions: Global regulatory functions
acnB	Energy metabolism, carbon: TCA cycle
fis	Macromolecule synthesis, modification: DNA - replication, repair, restr./ modification
lpdA	Energy metabolism, carbon: Pyruvate dehydrogenase
aldB	Degradation of small molecules: Carbon compounds
guaA	Nucleotide biosynthesis: Purine ribonucleotide biosynthesis
guaB	Nucleotide biosynthesis: Purine ribonucleotide biosynthesis
hupA	Macromolecule synthesis, modification: Basic proteins - synthesis, modification
hupB	Macromolecule synthesis, modification: Basic proteins - synthesis, modification
mtlA	Transport/binding proteins: Sugar-specific PTS system
mtlD	Degradation of small molecules: Carbon compounds
mtlR	Degradation of small molecules: Carbon compounds
fruR	Energy metabolism, carbon: Glycolysis

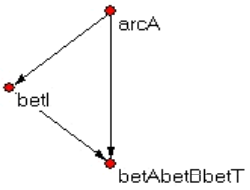
a)



crp	Global regulatory functions: Global regulatory functions
caiA	Degradation of small molecules: Carbon compounds
caiF	Degradation of small molecules: Carbon compounds
caiB	Degradation of small molecules: Carbon compounds
caiC	Degradation of small molecules: Carbon compounds
caiD	Degradation of small molecules: Amines
caiE	Degradation of small molecules: Carbon compounds
caiT	Degradation of small molecules: Carbon compounds
fixA	Degradation of small molecules: Carbon compounds
fixB	Degradation of small molecules: Carbon compounds
fixC	Degradation of small molecules: Carbon compounds
fixX	Degradation of small molecules: Carbon compounds

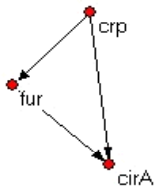
b)

FIGURA 2



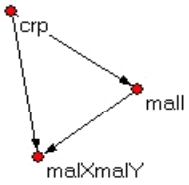
arcA	Global regulatory functions
betA	Adaptation: Osmotic adaptation
betI	Adaptation: Osmotic adaptation
betB	Adaptation: Osmotic adaptation
betT	Transport/binding proteins: BCCT superfamily

a) Módulo 2



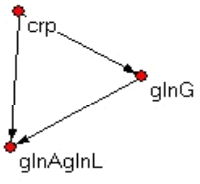
crp	Global regulatory functions
cirA	Cell envelop: Outer membrane constituents
fur	Global regulatory functions

b) Módulo 4



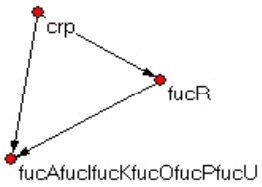
crp	Global regulatory functions
malI	Degradation of small molecules: Carbon compounds
malX	Transport/binding proteins: Sugar-specific PTS system
malY	Degradation of small molecules: Carbon compounds

c) Module 11



crp	Global regulatory functions
glnA	Amino acid biosynthesis: Glutamine
glnG	Amino acid biosynthesis: Glutamine
glnL	Amino acid biosynthesis: Glutamine

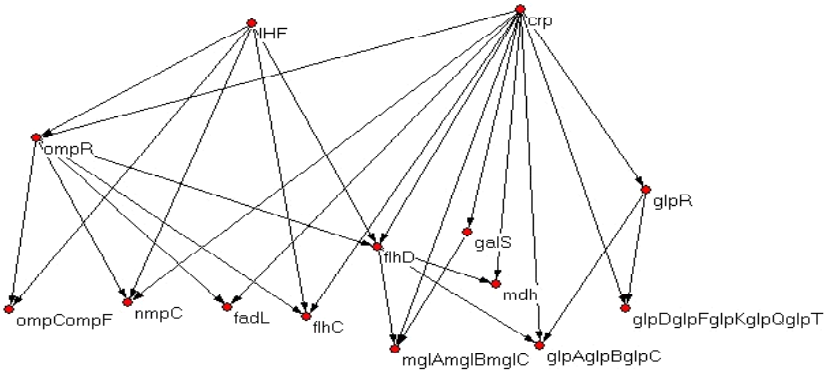
d) Module



crp	Global regulatory functions: Global regulatory functions
fucA	Degradation of small molecules: Carbon compounds
fucR	Degradation of small molecules: Carbon compounds
fucI	Degradation of small molecules: Carbon compounds
fucK	Degradation of small molecules: Carbon compounds
fucO	Degradation of small molecules: Carbon compounds
fucP	Transport/binding proteins: MFS family
fucU	Degradation of small molecules: Carbon compounds

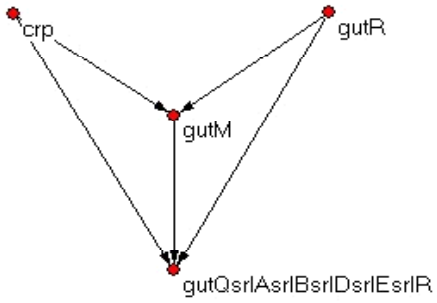
e)

FIGURA 3



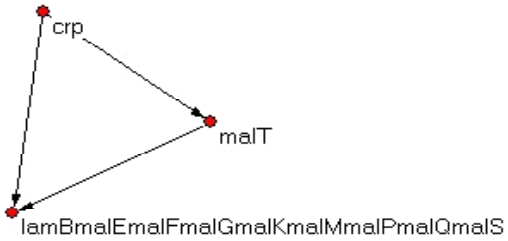
crp	Global regulatory functions: Global regulatory functions
fadL	Transport/binding proteins: Mechanism not stated
ompR	Global regulatory functions: Global regulatory functions
flhC	Cell envelop: Surface structures
flhD	Cell envelop: Surface structures
glpA	Energy metabolism, carbon: Anaerobic respiration
glpR	Energy metabolism, carbon: Anaerobic respiration
glpB	Energy metabolism, carbon: Anaerobic respiration
glpC	Energy metabolism, carbon: Anaerobic respiration
mdh	Energy metabolism, carbon: TCA cycle
mglA	Transport/binding proteins: ABC superfamily (atp_bind)
mglB	Transport/binding proteins: ABC superfamily (peri_perm)
mglC	Transport/binding proteins: ABC superfamily (membrane)
galS	Degradation of small molecules: Carbon compounds
glpD	Energy metabolism, carbon: Aerobic respiration
glpF	Transport/binding proteins: MIP family
glpK	Central intermediary metabolism: Pool, multipurpose conversions of intermed. met'm
glpQ	Central intermediary metabolism: Pool, multipurpose conversions of intermed. met'm
glpT	Transport/binding proteins: Mechanism not stated
nmpC	Laterally acquirred elements: Phage-related functions and prophages
IHF	
ompC	Cell envelop: Outer membrane constituents
ompF	Cell envelop: Outer membrane constituents

FIGURA 4



crp	Global regulatory functions: Global regulatory functions
gutM	Degradation of small molecules: Carbon compounds
gutQ	Unknown proteins, no known homologs: Unknown function
srlA	Transport/binding proteins: Sugar-specific PTS system
srlB	Transport/binding proteins: Sugar-specific PTS system
srlD	Degradation of small molecules: Carbon compounds
srlE	Transport/binding proteins: Sugar-specific PTS system
srlR	Degradation of small molecules: Carbon compounds

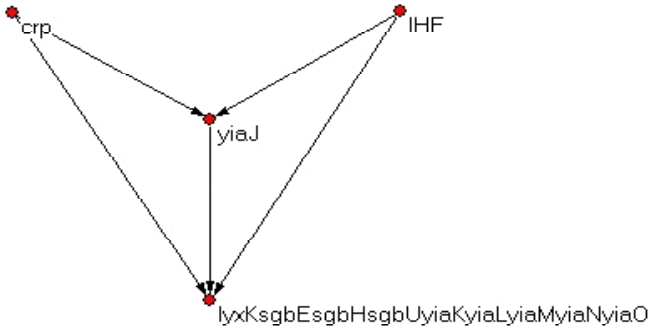
a)



crp	Global regulatory functions: Global regulatory functions
lamB	Transport/binding proteins: Mechanism not stated
malT	Degradation of small molecules: Carbon compounds
malE	Transport/binding proteins: ABC superfamily (peri_perm)
malF	Transport/binding proteins: ABC superfamily (membrane)
malG	Transport/binding proteins: ABC superfamily (membrane)
malK	Transport/binding proteins: ABC superfamily (atp_bind)
malM	Degradation of small molecules: Carbon compounds
malP	Degradation of small molecules: Carbon compounds
malQ	Macromolecule degradation: Degradation of polysaccharides
malS	Macromolecule degradation: Degradation of polysaccharides

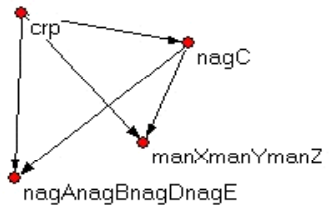
b)

FIGURA 5



crp	Global regulatory functions: Global regulatory functions
lyxK	Degradation of small molecules: Carbon compounds
yiaJ	Some information, but not classifiable: Not classified (included putative assignments)
sgbE	Unknown proteins, no known homologs: Unknown function
sgbH	Central intermediary metabolism: Pool, multipurpose conversions of intermed. met'm
sgbU	Central intermediary metabolism: Pool, multipurpose conversions of intermed. met'm
yiaK	Unknown proteins, no known homologs: Unknown function
yiaL	Unknown proteins, no known homologs: Unknown function
yiaM	Unknown proteins, no known homologs: Unknown function
yiaN	Some information, but not classifiable: Not classified (included putative assignments)
yiaO	Some information, but not classifiable: Not classified (included putative assignments)

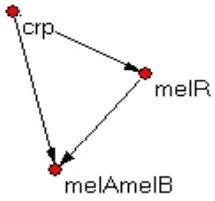
a)



crp	Global regulatory functions: Global regulatory functions
manX	Transport/binding proteins: Sugar-specific PTS system
nagC	Central intermediary metabolism: Amino sugars
manY	Transport/binding proteins: Sugar-specific PTS system
manZ	Transport/binding proteins: Sugar-specific PTS system
nagA	Central intermediary metabolism: Amino sugars
nagB	Central intermediary metabolism: Amino sugars
nagD	Central intermediary metabolism: Amino sugars
nagE	Transport/binding proteins: Sugar-specific PTS system

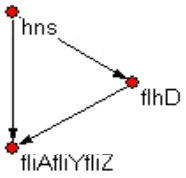
b)

FIGURA 6



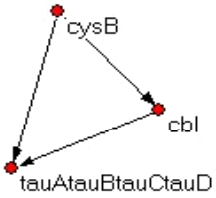
crp	Global regulatory functions
melA	Degradation of small molecules: Carbon compounds
melR	Degradation of small molecules: Carbon compounds
melB	Transport/binding proteins: GPH family

a)



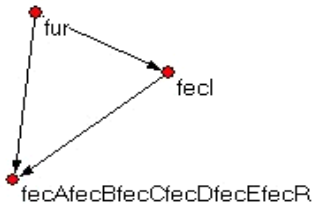
flhD	Cell envelop: Surface structures
fliA	Cell envelop: Surface structures
hns	Macromolecule synthesis, modification: Basic proteins - synthesis, modification
fliY	Transport/binding proteins: ABC superfamily (peri_perm)
fliZ	Not classified

b)



cbl	Amino acid biosynthesis: Cysteine
tauA	Transport/binding proteins: ABC superfamily (peri_perm)
cysB	Amino acid biosynthesis: Cysteine
tauB	Transport/binding proteins: ABC superfamily (atp_bind)
tauC	Transport/binding proteins: ABC superfamily (membrane)
tauD	Degradation of small molecules: Carbon compounds

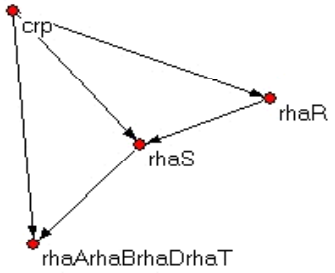
c)



fur	Global regulatory functions
fecI	Global regulatory functions
fecA	Transport/binding proteins
fecB	Transport/binding proteins: ABC superfamily (peri_perm)
fecC	Transport/binding proteins
fecD	Transport/binding proteins: ABC superfamily (membrane)
fecE	Transport/binding proteins: ABC superfamily (atp_bind)
fecR	Transport/binding proteins

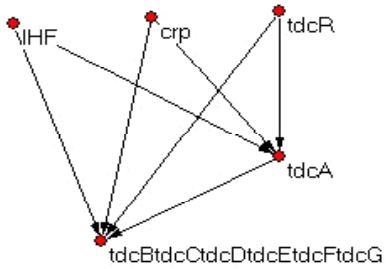
d)

FIGURA 7



crp	Global regulatory functions: Global regulatory functions
rhaA	Degradation of small molecules: Carbon compounds
rhaS	Degradation of small molecules: Carbon compounds
rhaB	Degradation of small molecules: Carbon compounds
rhaD	Degradation of small molecules: Carbon compounds
rhaR	Degradation of small molecules: Carbon compounds
rhaT	Transport/binding proteins: GntP family

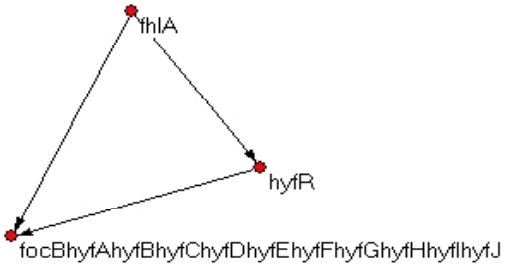
a)



crp	Global regulatory functions: Global regulatory functions
tdcA	Degradation of small molecules: Amino acids
tdcB	Degradation of small molecules: Amino acids
tdcC	Transport/binding proteins: STP family
tdcD	Degradation of small molecules: Amino acids
tdcE	Energy metabolism, carbon: Anaerobic respiration
tdcF	Unknown proteins, no known homologs: Unknown function
tdcG	Degradation of small molecules: Amino acids
tdcR	Degradation of small molecules: Amino acids

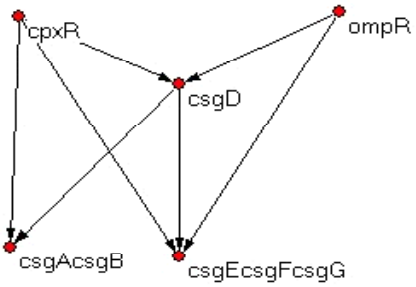
b)

FIGURA 8



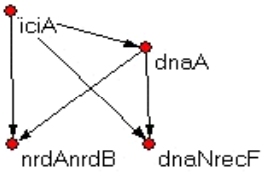
fhlA	Energy metabolism, carbon: Fermentation
focB	Transport/binding proteins: Mechanism not stated
hyfR	Some information, but not classifiable
hyfA	Energy metabolism, carbon: Anaerobic respiration
hyfB	Energy metabolism, carbon: Anaerobic respiration
hyfC	Energy metabolism, carbon: Anaerobic respiration
hyfD	Energy metabolism, carbon: Anaerobic respiration
hyfE	Energy metabolism, carbon: Anaerobic respiration
hyfF	Energy metabolism, carbon: Anaerobic respiration
hyfG	Energy metabolism, carbon: Anaerobic respiration
hyfH	Energy metabolism, carbon: Anaerobic respiration
hyfI	Energy metabolism, carbon: Anaerobic respiration
hyfJ	Some information, but not classifiable

b)



cpxR	Some information, but not classifiable
csgA	Cell envelop: Surface structures
csgD	Some information, but not classifiable
csgB	Cell envelop: Surface structures
csgE	Cell envelop: Surface structures
ompR	Global regulatory functions
csgF	Cell envelop: Surface structures
csgG	Cell envelop: Surface structures

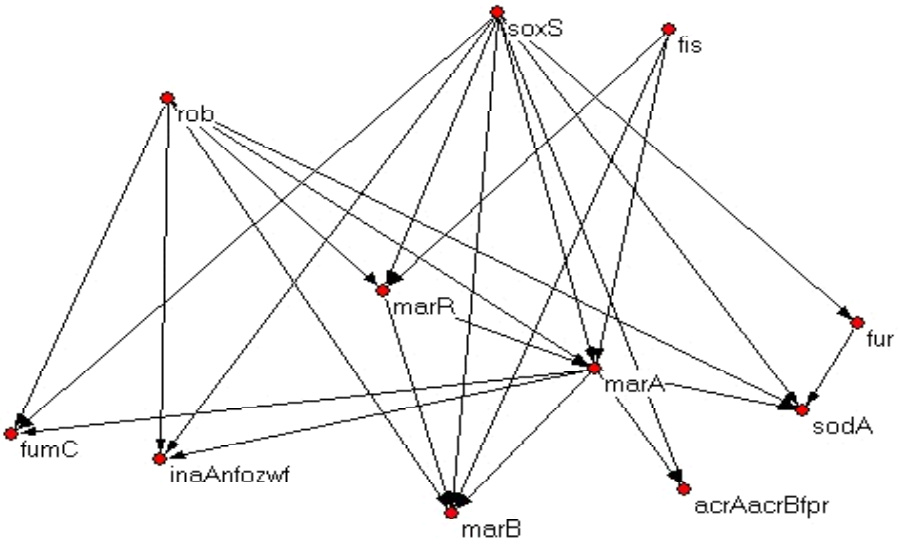
c)



dnaA	Macromolecule synthesis, modification: DNA - replication, repair, restr./ modification
dnaN	Macromolecule synthesis, modification: DNA - replication, repair, restr./ modification
iciA	Macromolecule synthesis, modification: DNA - replication, repair, restr./ modification
nrdA	Central intermediary metabolism: 2'-Deoxyribonucleotide metabolism
nrdB	Central intermediary metabolism: 2'-Deoxyribonucleotide metabolism
recF	Macromolecule synthesis, modification: DNA - replication, repair, restr./ modification

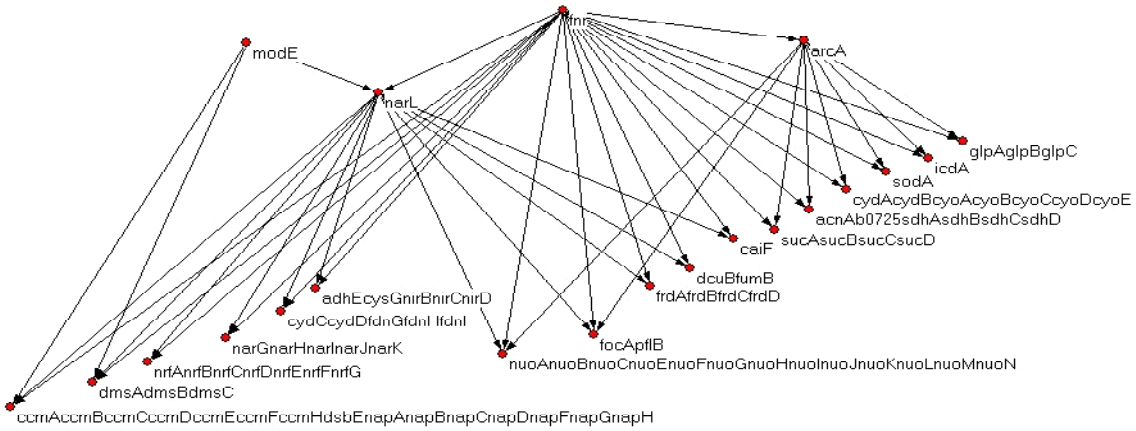
a)

FIGURA 9



marA	Protection responses: Drug/analog sensitivity
fumC	Energy metabolism, carbon: TCA cycle
rob	Macromolecule synthesis, modification: DNA - replication, repair, restr./ modification
soxS	Global regulatory functions: Global regulatory functions
sodA	Protection responses: Detoxification
fur	Global regulatory functions: Global regulatory functions
marB	Protection responses: Drug/analog sensitivity
fis	Macromolecule synthesis, modification: DNA - replication, repair, restr./ modification
marR	Protection responses: Drug/analog sensitivity
acrA	Protection responses: Drug/analog sensitivity
acrB	Transport/binding proteins: RNDfamily
fpr	Central intermediary metabolism: Pool, multipurpose conversions of intermed. met'm
inaA	Adaptation: Adaptations, atypical conditions
nfo	Macromolecule degradation: Degradation of DNA
zwf	Energy metabolism, carbon: Oxidative branch, pentose pathway

FIGURA 10



arcA	Macromolecule synthesis, modification: DNA - replication, repair, restr./ modification
acnA	Fatty acid biosynthesis: Fatty acid and phosphatidic acid biosynthesis
fnr	Some information, but not classifiable:
b0725	Macromolecule synthesis, modification: DNA - replication, repair, restr./ modification
cydA	Macromolecule synthesis, modification: DNA - replication, repair, restr./ modification
cydB	Some information, but not classifiable:
cyoA	Degradation of small molecules: Amino acids
cyoB	Laterally acquirred elements: Phage-related functions and prophages
cyoC	Cell envelop: Outer membrane constituents
cyoD	Central intermediary metabolism: Pool, multipurpose conversions of intermed. met'm
cyoE	Central intermediary metabolism: Pool, multipurpose conversions of intermed. met'm
focA	Energy metabolism, carbon: Electron transport
glpA	Energy metabolism, carbon: Electron transport
glpB	Energy metabolism, carbon: Aerobic respiration
glpC	Energy metabolism, carbon: Aerobic respiration
icdA	Energy metabolism, carbon: Aerobic respiration
nuoA	Energy metabolism, carbon: Aerobic respiration
nuoB	Energy metabolism, carbon: Aerobic respiration
nuoC	Energy metabolism, carbon: Aerobic respiration
nuoE	Degradation of small molecules: Carbon compounds
nuoF	Degradation of small molecules: Carbon compounds
nuoG	Degradation of small molecules: Carbon compounds
nuoH	Degradation of small molecules: Carbon compounds
nuoI	Transport/binding proteins: MFS family
nuoJ	Transport/binding proteins: ABC superfamily (peri_perm)
nuoK	Transport/binding proteins: ABC superfamily (atp_bind)
nuoL	Transport/binding proteins: Mechanism not stated
nuoM	Transport/binding proteins: Mechanism not stated
nuoN	Transport/binding proteins: MFS family
pflB	Global regulatory functions: Global regulatory functions
sdhA	Central intermediary metabolism: Glyoxylate bypass
sdhB	Central intermediary metabolism: Glyoxylate bypass
sdhC	Central intermediary metabolism: Glyoxylate bypass
sdhD	Energy metabolism, carbon: TCA cycle
sodA	Energy metabolism, carbon: TCA cycle
sucA	Energy metabolism, carbon: Fermentation
sucB	Unknown proteins, no known homologs: Unknown function
sucC	Adaptation: Osmotic adaptation
sucD	Adaptation: Osmotic adaptation
narL	Adaptation: Osmotic adaptation
caiF	Transport/binding proteins: BCCT superfamily
dcuB	Energy metabolism, carbon: Electron transport
fumB	Energy metabolism, carbon: Electron transport
frdA	Energy metabolism, carbon: Aerobic respiration
frdB	Energy metabolism, carbon: Aerobic respiration
frdC	Energy metabolism, carbon: Aerobic respiration
frdD	Energy metabolism, carbon: Aerobic respiration
adhE	Energy metabolism, carbon: Aerobic respiration
cysG	Transport/binding proteins: DAACS family
nirB	Degradation of small molecules: Fatty acids
nirC	Degradation of small molecules: Fatty acids
nirD	Transport/binding proteins: FNT family
ccmA	Energy metabolism, carbon: TCA cycle
ccmB	Energy metabolism, carbon: TCA cycle
ccmC	Transport/binding proteins: Mechanism not stated
ccmD	Central intermediary metabolism: Glyoxylate bypass
ccmE	Degradation of small molecules: Carbon compounds
ccmF	Central intermediary metabolism: Pool, multipurpose conversions of intermed. met'm
ccmH	
cydC	Unknown proteins, no known homologs: Unknown function
cydD	Energy metabolism, carbon: Anaerobic respiration
dmsA	Energy metabolism, carbon: Anaerobic respiration
dmsB	Energy metabolism, carbon: Anaerobic respiration
dmsC	Energy metabolism, carbon: TCA cycle
dsbE	Energy metabolism, carbon: TCA cycle
fdnG	Energy metabolism, carbon: Aerobic respiration
fdnH	Transport/binding proteins: Mechanism not stated
fdnI	Energy metabolism, carbon: Aerobic respiration
napA	Energy metabolism, carbon: Pyruvate dehydrogenase
napB	Energy metabolism, carbon: TCA cycle
napC	Energy metabolism, carbon: Aerobic respiration
napD	Energy metabolism, carbon: Aerobic respiration
napF	Energy metabolism, carbon: Aerobic respiration
napG	Energy metabolism, carbon: Aerobic respiration
napH	Energy metabolism, carbon: Aerobic respiration
narG	Energy metabolism, carbon: Aerobic respiration
narH	Energy metabolism, carbon: Aerobic respiration
narI	Energy metabolism, carbon: Aerobic respiration
narJ	Energy metabolism, carbon: Aerobic respiration
narK	Energy metabolism, carbon: Aerobic respiration
nrfA	Energy metabolism, carbon: Aerobic respiration
nrfB	Energy metabolism, carbon: Aerobic respiration
nrfC	Energy metabolism, carbon: Aerobic respiration
nrfD	Energy metabolism, carbon: Anaerobic respiration
nrfE	Energy metabolism, carbon: TCA cycle
nrfF	Energy metabolism, carbon: TCA cycle
nrfG	Energy metabolism, carbon: TCA cycle
modE	Energy metabolism, carbon: TCA cycle

FIGURA 11

APENDICE B

Manuales de Usuario

1. Extracción de Motifs de Redes

Descripción general

El ejecutable **Motifs.exe** es el programa que implementa la metodología presentada en [**72**] para la detección de Motif de redes. Como se menciono en el capitulo V, existen dos variantes para el conteo de subgrafos de tamaño n que dan diferentes resultados una vez que se hace el conteo. En la primera variante, denominada conteo estricto, se hace un conteo estricto de todos los posibles subgrafos de tamaño n, independientemente de cualquier contexto en el que se encuentre cualquier subgrafo. En la segunda variante, denominada conteo sin contención, sólo se cuantan los subgrafos de tamaño n que no son subgrafos de otro subgrafo también de tamaño n. Esta diferencia es muy sutil, pero importante, y no se menciona en [**72**], de hecho sus resultados son en base a la segunda variante cuando en la metodoogía se habla de la primera.

Entrada

El programa **Motifs.exe** lee el archivo Input.txt localizado en el mismo directorio que **Motifs.exe.** Este archivo especifica el nombre de la red que se va a anlizar (NameRed), el formato del archivo que especifica la red (NetFormat. Al final de este tutorial se listan los formatos soportados), el algoritmo de conteo utilizado 0 para conteo estricto y 1 para conteo sin contención (Type) y por último el numero de redes aleatorias que se utilizaran para hacer la estadistica (nRand). Además debe de estar la carpeta `IsoGraph4` (esta contiene los grafos isomorfos de 4 nodos, para que no sean generados por el programa cada vez que este se ejecute) en el mismo directorio que el ejecutable El formato del archivo es el siguiente:

```
NameRed NetFormat Type nRand
```

Salida

Al terminar el programa habra generado un conjunto de archivos que se describen a continuación:

- *Motifk.txt*: Contiene el numero de subgrafos de tamaño k encontrados en la red bajo estudio (NameRed). En el apendice X, se especifica un identificador numerico para cada uno de los diferentes posibles subgrafos de tamaño k. En el archivo aparece una secuencia de numeros, la posición de cada numero en esta secuencia denota el identificador del subgrafo y el numero mismo determina la cantidad en que aparecio dicho subgrafo.
- *Statisticsk.txt:* Contiene el numero de subgrafos de tamaño k encontrados en cada una de las redes aleatorias, cada renglon, con el mismo formato que en el archivo anterior, denota el numero de subgrafos de cada red aleatoria en particular. El renglon $nRand + 1$ denota la varianza y el renglon $nRand + 2$

la media obtenida de el conjunto de redes aleatorias. El renglon $n+4$ es el computo del valor $z = (r-\mu)/\sigma$. Y los renglones $nRand+5$ y $nRand+6$ son los valores de los parametros U y p respectivamente.

Formatos para la especificación de redes

Formato 0

El formato 0 se utiliza para especificar un atributo numerico (un entero) a cada una de las aristas de la red. Este formato se utiliza, por ejemplo, en las redes de regulación, para especificar el tipo de regulación: positiva 0, negativa 1 y dual 2 (El archivo `\DB Graphs\RegulonDB\generegulated.dat` contiene la red de regulación de la bacteria *E coli*). El formato es el siguiente:

```
NameNodeX NameNodeY Attribute
...
```

Formato 1

El formato 1 se utiliza especificamente para redes neuronales. El formato se encuentra en el archivo `\DB Grpahs\CElegans\data base-read me` incluido en el CD que acompaña a la tesis

El programa **Motifs.exe** jnuto con la carpeta `IsoGraph4` y las redes de ejemplo estan en el CD de la tesis.

2. Generación de la red de interación entre Motifs de redes

Descripción General.

El programa **MotifInteraction.exe** es el programa que implementa el algoritmo descrito en el capitulo 5 para el estudio de la topologia de interacción entre los diferentes Motifs de redes.

Entrada

El programa **MotifInteraction.exe** lee el archivo Input.txt localizado en el mismo directorio. Este archivo especifica el nombre de la red que se va a anlizar (NameRed) y su formato (NetFormat), el algoritmo de conteo utilizado 0 para conteo estricto y 1 para conteo sin contención (Type), el numero de Motifs que se utilizaran para generar la red (nMof) y finalmente una lista de 0 hasta nMof con los identificadores de los Motifs que se van a estudiar. El formato para especificar los Motifs es un par x y donde x especifica el tamaño del Motif y y es su identificador. El formato del archivo es el siguiente:

```
NameRed NetFormat Type
nMof
x1 y1
...
xnMof ynMof
```

Salida

Al terminar el programa habra generado un conjunto de archivos que se describen a continuación:

- x-y.txt: Cada uno de estos archivos contine una lista con todas la sinstancias del Motif con identificador x-y.

- nmk.tgf: es la red de interacción entre las instancias del Motif k (k es el indice en el que fue introducido en el archivo de entrada). El formato de este archivo es el utilizado por el programa yED^{TM}, uno de los programas mas utilizados para la graficación de grafos.
- nmkMl.tgf: Es el componente l extraido de la red de interacción entre Motifs de tipo k.

3. Simulador de Redes de Regulación

Descripción General

El programa para windows networkedCell.exe es el ejecutable con la implementación del modelo presentado en el capítulo 6 para la simulación de redes de transcripción genética.

Entrada

Al ejecutar este programa se abre el cuadro de dialogo mostrado en la figura 1. En el cuadro de texto `modelo` se indica el archivo que contienen la especificación del modelo que se va a simular y el cuadro de texto `salida` se indica el nombre del archivo que contendra el resultado de la simulación.

FIGURA 1

El formato del archivo de entrada es el siguiente:

```
NoEnsambles TimeSimulation SizeX SizeY
NoGenes
IdGen0 GeneName DegradationTime Active PosX PosY
IdGen1 GeneName DegradationTime Active PosX PosY
...
IdGenNoGenes GeneName DegradationTime Active PosX PosY
NoEdges
Ida Idb RegulationType
...
NoPerturbations
Time CIdGen0 ...  CIdGenNoGenes
```

Donde,

- *NoEnsambles:* indica el numero de realizaciones del modelo para hacer la estadística.
- *TimeSimulation:* indica el numero de pasos que durara cada una de las realizaciones.
- *SizeX y SizeY:* indican el largo y ancho repectivamente del tamaño de la latice.
- *NoGenes:* indica el número de genes que habrá dentro del sistema
- *IdGeneX:* es un valor entero que se utiliza como identificador de una gen particular
- *GeneName:* especifica el nombre del gen
- *DegradationTime:* especifica el tiempo de degradación de las proteinas expresadas por el gen, es decir, una vez expresada una proteina se mantendra en el sistema durante DegradationTime pasos de tiempo discretos.
- *Active:* es una variable binária que determina si el gen es activo o pasivo.

- *PosX y PosY:* especifican las coordenadas dentro de la latice donde se localiza el gen.
- *NoEdges:* indica el número de interacciónes de regulación que hay entre los genes del sistema.
- *Ida y Idb:* Son numeros que especifian que hay una arista dirigida del gen con identificador *Ida* al gen con identificador *Idb*.
- *RegulationType:* especifica si la regulación es activadora 1, represora 2 o dual 3.
- *NoPerturbations:* indica el número de perturbaciónes que se haran al sistema. El valor minimo de este valor es 1 para especificar la condición inicial.
- *Time:* especifica el tiempo en el que se perturbará al sistema.
- *CIdGenX:* especifica el número de proteinas que se agregaran arbitrariamente al sistema.

Una vez caragdo el archivo con el modelo se puede ver la simulación grafica de una de las realizaciónes presinando la tecla F2 para que el sietma ejecute cada paso de tiempo discreto o se puede correr la estadistica de todas las realización presionando la tecla F3.

Salida

Si se eligió correr la estadistica entonces se generara el archivo especificado en el cuadro de texto `salida`, el cual contiene en el renglon i los valores promedio del numero de proteinas de cada gen en el tiempo i.

Referencias

[1] M. Waldrop **"Complexity.The Emerging science at the edge of order and chaos"** Ed. Touchtone (1993)

[2] R. Gallagher, R. Appenzeller, **"Beyond Reductionism"**, Science Vol. 284 (1999)

[3] R. Ziemelis, **"Nature Insight: Complex Systems"**, Nature Vol. 410 (2001)

[5] E. Bonabeau, M. Dorigo and G. Theraulaz, **"Swarm Intelligence: From Natural to Artificial Systems"** Ed. Oxford University Press (1999)

[6] R. Davies, **"Why is the Physical World so Comprehensible?"** in Complexity, Entropy, and the Physics of Information, SFI Studies in the Sciences of Complexity, vol. VIII, Ed. Zurek W., Ed. Addison-Wesley, (1990)

[7] R. Shinbrot and R. Muzzio, **"Noise to order"**, Nature Vol. 410 (2001)

[8] R. Tsimplis, **"The effect of rain in calming the sea"** J. Phys. Occeanogr. 22, (1993)

[9] H. Strogatz **"Exploring Complex Networks"** Nature Vol. 410 (2001)

[10] J. Holland, **"Hidden Order"**, Perseus Publishing (1996)

[11] B. Bollobás, **"Modern Graph Theory"**, Ed. Springer (1998)

[12] P. Ball **"Transitions still to be made"**, Nature Vol. 402 (1999)

[13] D. Sornette, **"crítical Phenomena in Natural Sciences"**, Ed. Springer (2000)

[14] P. Bak, C. Tang, and K. Wiesenfeld, **"Self-organized críticality: An explanation of 1/f noise"**, Phys. Rev. Lett. 59, 381 (1987)

[15] C. R. Shalizi, **"Methods and Techniques of Complex Systems Science: An Overview"**, Thomas S. Deisboeck, J. Yasha Kresh and Thomas B. Kepler (eds.), Complex Systems Science in Biomedicine, Kluwer (2003)

[16] T. Schreiber **"Interdisciplinary application on nonlinear time series methods"**, Physics Reports (1998)

[21] K. Svozil, **"Randomness & Undecidability in Physics"**, World Scientific (1991)

[17] J. L. Casti **"Would be worlds: How Simulation is Changing the Frontiers of Science"**, Wiley (1997)

[18] S. Lloyd **"Computational capacity of the Universe"**, Phys. Rev. Letters, 88, (2002)

[19] R. C. Hilborn, **"Chaos and Nonlinear Dynamics: An Introduction for Scientists and Engineers"**, Oxford University Press, (1994)

[20] H. Rogers, **"Theory of Recursive Functions and Effective Computability"**, MIT Press (1987)

[22] A. Turing, **"On computable numbers, with an application to the Entscheidungsproblem"** Proceedings of the London Mathematical Society, (Ser. 2, Vol. 42, 1937)

[23] J. E. Hopcroft, R. Motwani, J. D. Ullman **"Introduction to Automata Theory, Languages, and Computation (2nd Edition)"** Addison-Wesley November 2000

[24] D. Deutsch **"Quantum theory, the Church-Turing principle and the universal quantum computer"**, Proc. R. Soc. Lond A 400 (1985)

[25] R. Motwani and P. Raghavan, **"Randomized Algorithms"**, Cambridge Univ. Press (1995)

[26] J. von Neumann, **"Theory of Self-Replication Automata"**, ed by A. W. Burks, University Illinois Press (1966)

[27] T. Toffoli, N Margolus **"Cellular Automata Machines: A new Environment for Modeling"** MIT Press 1987

[28] A. Ilachinski, **"Cellular Automata: A discrete Universe"**, World Scientific (2001)

[29] S. Wolfram, **"A new kind of Science"**, Stephen Wolfram Inc. (2002)

[30] N. Margolus, **"Physics like models of computation"**, Physica D 10 (1984)

[31] M. Sipser, **"Introduction to the Theory of Computation"**, Brooks Cole (1996)
[32] Albert R. and Barabási A.-L. **"Statistical mechanics of complex networks"**, Reviews of Modern Physics, Vol. 74 (2002)
[33] C.H. Papadimitriou, **"Computational Complexity"**, Addison-Wesley (1994)
[34] G. Ausiello, P. Crecenzi, G. Gambosi, V. Kann A. Marchetty-Spaccamela & M. Protasi **"Complexity and Aproximation: Combinatorial Optimization Problems and Their Approximability Properties"** Springer (1999)
[35] M. A. Nielsen and I. L. Chuang, **"Quantum Computation and Quantum Information"**, Cambridge University Press (2000)
[36] E. Bonabeau, M. Dorigo, G. Theraulaz, **"Inspiration for optimization from social insect behavior"**, Nature Vol. 406 (2000)
[37] D. Dasgupta, **"An Overview of Artificial Immune Systems and Their Applications"** on Artificial Immune Systems and Their Applications ed. D. Dasgupta, Springer (1998)
[38] C. A. Janeway and P. Travers. **"Immunobiology: the immune system in health and disease"**. Current Biology Ltd., London, 2nd edition, (1996)
[39] N. K. Jerne, **"Towards a Network theory of the immune system"**, Ann. Immunol. (Inst. Pasteur), 125C (1974)
[40] S. Boettcher and A. G. Percus, **"Optimization with Extremal Dynamics"**, Complexity Vol. 8, No. 2 (2003)
[41] H. Jensen, **"Self-Orginized críticaly: Emergent Complex Behavior in Physical and Biological Systems"** Cambridge University Press (1998)
[42] P. Bak and K. Sneppen, **"Punctuated Equilibrium and críticality in a simple model of Evolution"**, Phys. Rev: Lett 71, 4083 (1997)
[43] B. Mayer, **"Object-Oriented Software Construction"** 2nd edition, Prentice Hall (2000)
[44] Edited by G.D. Doolen, U. Frisch, B. Hasslacher, S. Orszag, and S. Wolfram, **"Lattice gas methods for partial differential equations"**, Addison-Wesley (1990)
[45] E. Gamma, **"Design Patterns"**, Addison-Wesley (1995)
[46] E. Berlekamp, J. H. Conway, and R. Guy, **"Winning Ways for Your Mathematical Plays"** Vol. 2. Academic Press (1982).
[47] V. S. Colella, E. Klopfer, M. Resnick, **"Adventures in Modeling: Exploring Complex, Dynamic Systems with StarLogo"**, Teachers College Press (2001)
[48] http://www.swarm.org/
[49] http://el.www.media.mit.edu/groups/el/Projects/starlogo/
[50] P. Erdös and A. Rényi, **"On random graphs"**, Publicationes Mathematicae, Vol. 6 (1959)
[51] D. J. Watts & S. H. Strogatz **"Collective Dynamics of 'Small World' Networks"** Nature Vol. 393 4-June-1998
[52] Ligeros E., Edling C., Nunes-Amaral L., Stanley E., Aberg Y. **"The web of human sexual contacts"** Nature Vol. 411 (2001)
[53] Jeong H. , Tombor B., Albert R., Oltvai Z. N. & Barabási A.-L. **"The large-scale organization of metabolic networks"** Nature Vol. 407 (2000)
[54] Williams R. and Martinez N. **"Simple rules yield complex food webs"**, Nature Vol. 404 (2000)
[55] R. Ferrer i Cancho, and R. V. Solé, **"The small world of human language"**, Proceedings of The Royal Society of London. Series B, Biological Sciences, 268(1482)
[56] Bollobas B. **"Random Graphs"** Ed. Cambridge University Press; 2nd edition (2001)
[57] M. E. J. Newman **"Models of Small World: A Review"**
[58] M. E. J. Newman and D. J. Watts **"Scaling and percolation in the small-world network model"**, Phys. Rev. E 60 (1999)
[59] M. E. J. Newman, S. H. Strogatz and D. J. Watts **"Random graphs with arbitrary degree distributions and their applications"** Phys. Rev. E 64, 6118 (2001)
[60] A.-L. Barabasi and R. Albert **"Emergence of scaling in random networks"** Science Vol. 286 (1999)
[61] S. N. Dorogostev, J. F. Mendes, A. N. Samukhin **"Structure of growing networks with preferential linking"** Phys. Rev. Lett. 85 4633 (2000)

[62] P. L. Krapivsky, S. Redner and F. Leyvraz **"Connectivity of growing random networks"**, Phys. Rev. Lett. 85, 4629
[63] P. L. Krapivsky and S. Redner **"Organization of growing random networks"**, Phys. Rev. E 63,066123 (2001)
[64] R. Albert, H. Jeong and A. Barabási **"Error and attack tolerance of complex networks"** Nature Vol. 406 (2000)
[65] M. E. Newman, S. Forrest, and J. Balthrop, **"Email networks and the spread of computer viruses"** Phys. Rev. E 66, 035101 (2002)
[66] H. W. Hethcote, **"The mathematics of infectious diseases"** SIAM Review 42, (2000)
[67] M. E. Newman **"Spread of epidemic diseases on Networks"**, Phys. Rev. E 66, 016128 (2002)
[68] D.S. Callaway, M. E. Newman, S. H. Strogatz, and D. J. Watts, **"Networks robustness and fragility: Percolation on random graphs"**, Phys. Rev. Lett. 85 (2000)
[69] J. M. Kleinberg, **"Navigation in a small world"**, Nature Vol. 406 (2000)
[70] T. Cormen, C. Leirserson, R. Rivest, and C. Stein, **"Introduction to Algorithms"**, 2nd Edition, MIT Press (2001)
[71] D. J. Watts, P. S. Dodds and M. J. Newman **"Identity and Search in Social Networks"** Science Vol. 296 (2002)
[72] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii and U. Alon, **"Network Motifs: Simple Building Blocks of Complex Networks"**, Science, Vol. 298, (2002)
[73] S. Shen-Orr, R. Milo, S. Mangan & U. Alon, **"Network Motifs in the transcriptional regulation network of Escherichia coli"**, Nature Genet. 31, 64 (2002)
[74] J. R. Ullmann, **"An Algorithm for Subgraph Isomorphism"** ACM Vol. 23 No. 1 1976
[75] S. Maslov and K. Sneppen, **"Specificity and Stability in Topology of Protein Networks"**, Science Vol. 296 (2002)
[76] Z. Michalewicz, D. B. Fogel, **"How to Solve it: Modern Heuristics"**, Springer (2000)
[77] http://www.cifn.unam.mx/Computational_Genomics/regulondb/
[78] F. Brglez, D. Bryan, and K. Kozminski, **"Combinational Profiles of Sequential Benchmark Circuits"**, Proc. IEEE Int. Symposium on Circuits and Systems, May (1989)
[79] http://www.cbl.ncsu.edu/CBL_Docs/iscas89.html
[80] G. Tononi, O. Sporns, and G. M. Edelman, **"A measure for brain complexity: Relating funcional segregation and integration in the nervous system"**, Proc. Natl. Acad. Sci. USA Vol. 91, (1994)
[81] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabási, **"Hierarchical Organization of Modularity in Metabolic Networks"**, Science Vol. 297 (2002)
[82] G. M. Edelman, J. Gally **"Degenereation and complexity in biological systems"**, Proc. Natl. Acad. Sci. Vol. 98, Nov (2001)
[83] R. V. Solé, R. F. Cancho, S. Valverde, and J. M. Montoya, **"Selection, Tinkering, and Emergence in Complex Networks"**, Complexity Vol. 8 (2002)
[84] L. H. Hartwell, J.J. Hopfield, S. Leibler and A. W. Murray, **"From molecular to modular cell biology"**, Nature Vol 402, C47 (1999)
[85] H. Kitano, **"Systems Biology: A Brief Overview"**, Science Vol 295, 1662 (2002)
[86] M. Mitchell, **"An Introduction to Genetic Algorithms",** MIT Press (1996)
[87] S. Khuri, T Bäck, and J. Heitkötter, **"The Zero/One Multiple Knapsack Problem and Genetic Algorithms"**, ACM Symposium of Applied Computation (SAC'94)
[88] H.M. Weingartner and D.N. Ness **"Methods for the solution of the multi-dimensional 0/1 knapsack problem."** Operations Research, 15 (1967)
[89] A. Frevile and G. Plateau, **"Hard 0/1 multiknapsack test problems for size reduction methods"**, Investigatión Operativa 1 (1990)
[90] L Adleman, **"Molecular computation of solutions to combinatorial problems"**, Science Vol. 266 (1994).
[91] L. Smolin, **"Atoms of space and time"**, SIAM Enero 2004
[92] G. Conant, A Wagner, **"Convergent evolution of gene circuits"**, Nature Genetics Vol. 34 July (2003)

[93] S. Wolfram, **"Undecidability and Intractability in Theoretical Physics"**, Physical Review Letters, Vol. 54, No 8 (1985)

[94] N. Israeli and N. Goldenfeld, **"Computational Irreducibility and the Predicatability of Complex Physical Systems"**, Physical Review Letters, Vol 27, No 7 (2004)

[95] N. E. Buchler, U. Gerland, and T. Hwa, **"On schemes of combinatorial transcription logic"**, PNAS March 2003

[96] B. Alberts et. al., **"Molecular Biology of the Cell"**, Garland Science; 4th Bk&Cdr edition (March, 2002)

[97] M. M. babu and S. A. Teichmann, **"Functional determinants of transcription factors in Escherichia coli: protein families and binding sites"** TRENDS in Genetics Vol. 19 No. 2 2003

[98] N B. Tufillaro, **"Nonlinear Dynamics: Flows in 3-D and beyond"**, Thursday 2PM, 22 June 1995 HP-LABS Bristol, UK.

Publicaciones generadas durante la tesis

[A] O. Resendis-Antonio, J. A. Freyre-González, **R. Menchaca-Méndez**, R. M. Gutiérrez-Ríos, A. Martínez-Antonio, C. Ávila-Sánchez, and J. Collado-Vides. **"Modular analysis of the transcriptional regulatory network of E. coli."** Trends Genet. 2005, in press.

[B] **R. Menchaca Méndez**, S. Janga, C. Ávila-Sánchez, J. Collado-Vides, **"Finding Modules Using Motifs as Building Blocks"**.Poster Session at 12th International Conference on Intelligent Systems for Molecular Biology, ISMB 2004. August 2nd, 2004. Glasgow, UK.

[C] S. Janga, C. Ávila-Sánchez, **R. Menchaca-Méndez**, J. Collado-Vides, **"Gene Ontology Network of Interactions among Operons in E. coli and B.subtilis"**. Poster Session at 12th International Conference on Intelligent Systems for Molecular Biology, ISMB 2004. August 2nd, 2004. Glasgow, UK.

[D] C. Ávila-Sánchez, S Janga, A. Martínez-Antonio, **R. Menchaca-Méndez**, J. Collado-Vides, **"Inference of Biological Modules in the Escherichia coli Regulatory Network"**.Poster Session at 12th International Conference on Intelligent Systems for Molecular Biology, ISMB 2004. August 2nd, 2004. Glasgow, UK.

[E] **R. Menchaca-Méndez** & J. Figueroa-Nazuno, **"Optimización Extrema Distribuida basada en Arquitectura de Mundo Pequeño"**. Poster Session at XLV Nacional Congress of Physics. Guanajuato, Mexico, Noviembre, 2002.

[F] **R. Menchaca-Mendez** and J. Figueroa-Nazuno, **"Solving NP-Complete problems with Cellular Automata"**, XI International Computing Conference. Mexico City, November, 2002.

[G] **Ricardo Menchaca-Mendez** & Rolando Menchaca-Mendez, **"Entorno de Desarrollo para la Simulación de Sistemas Complejos"** XV National Congress y I International Congress of Informatics and Computing of ANIEI. Guadalajara Mexico, Octubre, 2002.

[H] **R. Menchaca-Mendez**, C. Sanchez-Rodriguez & J. Figueroa-Nazuno, **"Predicción de Series de Tiempo mediante Modelado Gramatical"**. Poster Session at XLIV National Congress of Physics. Morelia, Mexico, 2001.

[I] **R. Menchaca-Mendez**, C. Sanchez-Rodriguez, A. Espinoza-Contreras and J. Figueroa-Nazuno. **"Modelado y Caraterización de Series de Tiempo mediante Análisis Gramatical"** Poster Session at XLIII National Congress of Physics. Puebla, Mexico, October-November 2000.

[J] **R. Menchaca-Mendez**, C. Sanchez-Rodriguez and J. Figueroa-Nazuno.**"Dos soluciones distintas al problema de satisfabilidad"**.Poster Session at XLIII National Congress of Physics. Puebla, Mexico, October-November 2000.