UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

PROYECTO FIN DE CARRERA

COMPUTACIÓN EVOLUTIVA APLICADA A LA SIMULACIÓN DE ENTORNOS P2P PUROS

Autor: Abel Fermosel Álvarez
Tutor: Almudena Alcaide Raya
Co-Tutor: Esther Palomar González Septiembre, 2009

AGRADECIMIENTOS

En primer lugar quisiera agradecer a Almudena Alcalde Raya y a Esther Palomar González la oportunidad que me han brindado para realizar este proyecto y aprender de ellas, y al Departamento de Informática el permitirme realizarlo.

A mis padres, porque cuando me pongo cabezón ellos saben orientarme, y sus consejos siempre me han ayudado. Y por supuesto al resto de mi familia… por vuestra confianza en mí.

A todos mis profesores, desde el colegio hasta la universidad; a todos mis compañeros y amigos de la universidad, porque sin todos vosotros, vuestros resúmenes, vuestros consejos, y vuestra ayuda seguro que no estaba escribiendo estas líneas.

A todos mis amigos de siempre, y a todos los demás que siempre estáis ahí.

Gracias.

RESUMEN

Las redes *peer-to-peer* totalmente descentralizadas se han difundido rápidamente porque no necesitan una gestión centralizada, sino que cualquier nodo de la red actúa como cliente y servidor, proporcionando de esta manera autosuficiencia a la red. Un ejemplo de este tipo de redes son las redes para compartir archivos, como por ejemplo Kademlia, Ares Galaxy, Gnutella, etc…

En este trabajo fin de carrera se han realizado importantes avances enfocados a analizar el comportamiento de los nodos dentro de una red P2P. La aportación se basa fundamentalmente en el uso de dos paradigmas, la Computación Evolutiva y la Teoría de Juegos.

Se podría decir que Teoría de Juegos es un área de la matemática aplicada que utiliza modelos para estudiar interacciones en estructuras formalizadas de incentivos (los llamados juegos) y llevar a cabo procesos de decisión. El juego en cuestión es "Piedra-Papel-Tijeras". Este juego se compone de tres acciones, y cada una de estas acciones gana a otra, formado un ciclo en el cual una acción gana a una y pierda contra la otra, es decir, para toda acción posible en el juego existe la posible victoria o derrota, dependerá de la acción elegida por el oponente.

Por otra parte, a partir de la computación evolutiva han surgido principalmente tres corrientes: la programación evolutiva, las estrategias evolutivas y los algoritmos genéticos. En este proyecto se ha implementado un algoritmo genético para poder lograr los resultados esperados. Los algoritmos genéticos fueron propuestos por John H. Holland en 1975 y su motivación inicial fue la de proponer un modelo general de proceso adaptable.

La implementación del algoritmo genético se basa en la evolución de la población, es decir, se genera la siguiente población con nodos que contengan jugadas mejores, de esta manera en cada iteración se mejora la población y se obtiene la mejor solución al cabo de un número de iteraciones, aunque en nuestro caso buscaremos una estrategia evolutiva estable en la población. Para ello, se ha diseñado e implementado una aplicación en .NET, con la cual se puedan realizar simulaciones que dependerán de los parámetros elegidos por el usuario y obtener las conclusiones más relevantes.

Índice de contenidos

Índice de tablas, figuras e imágenes

Acrónimos

P2P:	Peer-to-Peer
TJE:	Teoría de Juegos Evolutiva
TJ:	Teoría de Juegos
AG:	Algoritmos Genéticos
IA:	Inteligencia artificial
DP:	Dilema del prisionero
RPS:	Rock- Paper-Scissors

Glosario de términos

IP: es un número que identifica de manera lógica y jerárquica a una interfaz de un dispositivo (habitualmente una computadora) dentro de una red que utilice el protocolo IP (Internet Protocol), que corresponde al nivel de red del protocolo TCP/IP.

VoIP: voz sobre Protocolo de Internet, también llamado Voz sobre IP, VozIP, VoIP (por sus siglas en inglés), es un grupo de recursos que hacen posible que la señal de voz viaje a través de Internet empleando un protocolo IP (*Internet Protocol*).

Cortafuegos: es una parte de un sistema o una red que está diseñado para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicaciones autorizadas. Se trata de un dispositivo o conjunto de dispositivos configurados para permitir, limitar, cifrar, descifrar, el tráfico entre los diferentes ámbitos sobre la base de un conjunto de normas y otros criterios.

NAT: (Network Address Translation - Traducción de Dirección de Red) es un mecanismo utilizado por routers IP para intercambiar paquetes entre dos redes que se asignan mutuamente direcciones incompatibles.

Hub: es un dispositivo que permite centralizar el cableado de una red y poder ampliarla. Esto significa que dicho dispositivo recibe una señal y repite esta señal emitiéndola por sus diferentes puertos

Algoritmo Genético: es un procedimiento inspirado en la evolución (Charles Darwin), programado en computadoras (ordenadores) y orientado a producir soluciones a problemas donde los tratamientos clásicos encuentran dificultades.

Perfil estratégico: un perfil de estrategia define un plan de contingencia de las acciones de todos los jugadores durante el juego. Un perfil de estrategia por lo tanto determina la salida de un juego. Un perfil de estrategia constituye un equilibrio de Nash de un juego cuando ningún jugador de puede mejorar su rentabilidad de manera unilateral por

desviarse de la estrategia asignada.

Equilibrio de Nash (formulado por John Forbes Nash): se define como un modo de obtener una estrategia óptima para juegos que involucren a dos o más jugadores. Si hay un conjunto de estrategias tal que ningún jugador se beneficia cambiando su estrategia mientras los otros no cambien la suya, entonces ese conjunto de estrategias y las ganancias correspondientes constituyen un equilibrio de Nash.

CAPÍTULO 1. INTRODUCCIÓN

1.1. VISIÓN GENERAL

En la actualidad, el mundo de las telecomunicaciones es una de las áreas de la sociedad en la que se está produciendo una mayor actividad. Esta actividad se debe a la facilidad de comunicación que proporcionan las telecomunicaciones sin importar la distancia que exista entre los puntos a comunicar, principalmente Internet es la actividad mayor y con el crecimiento más alto. Gracias a este incremento, surgen novedades tecnológicas, las cuales hacen cada vez más fácil la comunicación entre distintos nodos de la red (Internet).

El objeto de estudio en este proyecto son las redes *peer-to-peer* (P2P) descentralizadas. Éstas son uno de los tipos de redes que se pueden encontrar en Internet que están teniendo más éxito en los últimos años. Esto se debe a que no necesitan una gestión centralizada, es decir, un nodo central que actúe de coordinador entre todos los demás nodos.

Un ventaja fundamental para demostrar el éxito de estas redes descentralizadas es la tolerancia a fallos; si se cae cualquier nodo, toda la red sigue funcionando, en cambio en las redes centralizadas en cuanto deja de funcionar correctamente el nodo que realiza la función de servidor toda la red se queda incomunicada sin poder hacer nada, solo queda la opción de esperar a que el nodo-servidor vuelva a funcionar correctamente.

Este tipo de redes normalmente se asocia a programas para la descarga de archivos. Para realizar la descarga de archivos entre dos nodos, un nodo actúa como *provider* (proveedor o servidor) y otro nodo como *requester* (solicitante). Intuitivamente para cada rol podemos identificar dos clases básicas de comportamiento: cooperar, por un lado, y no-cooperar por otro lado. Sin embargo, en este proyecto se han definido tres acciones posibles, denominadas como sigue: *Loner*, *Non-Cooperator* y *Cooperator*. Representan las posibles jugadas de los nodos, ya sean nodos *provider* o *requester*.

Lo que se quiere conseguir con este trabajo es modelar un algoritmo con el cual consigamos una estrategia evolutiva estable en la red P2P, dicho en otras palabras, que

no exista ninguna estrategia invasora o dominante en dicha red. Para ello se ha utilizado una disciplina novedosa, la “Teoría de Juegos Evolutiva” (TJE), fruto de la combinación de la “Teoría de Juegos” (TJ) clásica y los “Algoritmos Genéticos” (AG). El origen de la TJE radica en el intento de aplicar el conocimiento adquirido en la evolución biológica en otras áreas, y viene siendo usada desde hace unas décadas en entornos tan diversos como la economía, la sociología, etc. — todas ellas ciencias sociales para las que ha mostrado un excelente comportamiento al ser capaz de modelar con gran capacidad los problemas en los varios agentes interactúan con el fin de conseguir una optimización de sus resultados.

Lo primero que debe hacer es definir el juego a usar para la TJ, en este caso se ha elegido el juego de Piedra-Papel-Tijeras, este es el más indicado ya que se han identificado tres acciones para los posibles roles del nodo. Además se cumple que para cualquiera de las acciones posibles, hay una de las otras dos acciones a la que gana y la otra de las dos acciones perderá, de esta forma se pretende conseguir un equilibrio en la población de nodos P2P, ya que el nodo que pierda tendera a cambiar al comportamiento que le ha ganado formado así un ciclo estable.

Una vez elegido el juego debemos definir la matriz de pagos (véase Tabla 1). Esta matriz representa la valoración de la jugada de un nodo, es decir, en este caso si un nodo juega por ejemplo como piedra y su nodo con el que se relaciona juega como papel, entonces el primer nodo obtendrá la puntuación de -1 mientras que el segundo nodo al haber ganado obtendrá un 1.

		Jugador 2		
		Paper	*Rock*	*Scissors*
Jugador 1	*Paper*	(0, 0)	(1, -1)	(-1, 1)
	Rock	(-1, 1)	(0, 0)	(1, -1)
	Scissors	(1, -1)	(-1, 1)	(0, 0)

Tabla 1: Matriz de pagos en el juego RPS

El algoritmo (AG) a implementar debe ser acorde a las características del modelo a diseñar. Un AG tiene unas fases (ej: cruce, mutación, evaluación, etc...), pero cada una de estas fases puede diseñarse de distinta manera para obtener resultados distintos, por lo que habrá que encontrar los mejores diseños para estas fases. Más

adelante se entrará en detalle en cada una de estas fases.

Y por último se explicaran y se evaluaran los resultados obtenidos. Para una mayor facilidad en el análisis de los resultados, se crearan graficas gracias al simulador, de esta manera de un simple vistazo se podrá observar la evolución de los datos, y así sacar conclusiones.

1.2. OBJETIVOS

El objetivo de este proyecto es observar el comportamiento de una red P2P aplicando la computación evolutiva, demostrando que la red se mantiene y evoluciona sin contar con mecanismos de incentivos y en presencia de nodos no cooperadores. El estudio de dicho comportamiento ha sido realizado mediante la ejecución de numerosas simulaciones.

1.3. ESTRUCTURA DEL DOCUMENTO

La estructura del documento se compone de las cuatro partes siguientes:

1. La segunda parte "*Estado de la Cuestión*" detalla el planteamiento formal para obtener los resultados esperados, se definirán todas las características de la red peer-to-peer y el algoritmo genético aplicado.

2. En el tercer capítulo "*Marco teórico basado en teoría de juegos evolutiva para el análisis de redes P2P puras*", describe el modelo formal desarrollado en este proyecto para demostrar que ausencia de incentivos una red P2P puede ser estable.

3. En la cuarta sección, "*Diseño e implementación de un simulador de redes P2P basado en computación evolutiva*", se muestra el diseño creado para la creación de la aplicación, además de las decisiones tomadas, antes y durante la implementación.

4. En el quinto capítulo, "*Análisis de los resultados*", se expondrán los resultados

obtenidos según la aplicación implementada, se analizaran las graficas generadas por el simulador.

5. En la sexta parte del documento se encuentra el "*Presupuesto*" del proyecto, como en cualquier presupuesto se describen tanto los recursos humanos como los recursos materiales.

6. En última sección, se detallarán las "*Conclusiones y el trabajo futuro*" a seguir en esta vía de investigación, para permitir ampliar y/o modificar este proyecto fin de carrera de una manera fácil y cómoda.

CAPÍTULO 2. ESTADO DE LA CUESTIÓN

2.1. REDES PEER-TO-PEER

Una red P2P o red de iguales, es una red de computadoras en la que todos o algunos aspectos de ésta funcionan sin clientes ni servidores fijos, estando compuesta por una serie de nodos que se comportan como iguales entre sí. Actúan simultáneamente como clientes y servidores respecto a los demás nodos de la red. Como hemos mencionado anteriormente, las redes P2P aprovechan, utilizan y optimizan el uso del ancho de banda de los demás usuarios de la red por medio de la conectividad entre los mismos, obteniendo más rendimiento en las conexiones y transferencias que con algunos métodos centralizados convencionales, donde una cantidad relativamente pequeña de servidores provee el total del ancho de banda y recursos compartidos para un servicio o aplicación.

Dichas redes son útiles para diversos propósitos. A menudo se usan para compartir ficheros de cualquier tipo (por ejemplo, audio, video o software). Este tipo de red es también comúnmente usado en telefonía VoIP para hacer más eficiente la transmisión de datos en tiempo real. La eficacia de los nodos en el enlace y transmisión de datos puede variar según su configuración local (cortafuegos, NAT, ruteadores, etc.), la velocidad de proceso, la disponibilidad de ancho de banda de su conexión a la red y su capacidad de almacenamiento en disco.

Generalmente, el paradigma P2P se basa principalmente en la filosofía e ideales de que todos los usuarios deben compartir sus recursos. Conocida como filosofía P2P, es aplicada en algunas redes en forma de un sistema enteramente basado en méritos, en donde "el que más comparta, más privilegios tiene y más acceso dispone de manera más rápida a más contenido". Con mecanismos de incentivos y de reputación se pretende asegurar la disponibilidad del contenido compartido, ya que de lo contrario no sería posible la subsistencia de la red. Aquellos usuarios que no comparten contenido en la red se les denominan *free-riders*; los cuales muchas veces representan una amenaza para la disponibilidad de recursos en una red P2P debido a que únicamente consumen recursos sin reponer lo que consumen, por tanto podrían agotar los recursos compartidos

y atentar contra la estabilidad de la misma.

2.1.1. CLASIFICACIÓN

Las redes P2P según su grado de centralización se clasifican en:

I. Redes P2P centralizadas

Este tipo de red P2P se basa en una arquitectura monolítica en la que todas las transacciones se hacen a través de un único servidor que sirve de punto de enlace entre dos nodos y que, a la vez, almacena y distribuye los nodos donde se almacenan los contenidos. Poseen una administración muy dinámica y una disposición más permanente de contenido. Sin embargo, está muy limitada en la privacidad de los usuarios y en la falta de escalabilidad de un sólo servidor, además de ofrecer problemas en puntos únicos de fallo, situaciones legales y enormes costos en el mantenimiento así como el consumo de ancho de banda.

Una red de este tipo reúne las siguientes características:

- Se rige bajo un único servidor que sirve como punto de enlace entre nodos y como servidor de acceso al contenido, el cual distribuye a petición de los nodos.
- Todas las comunicaciones (como las peticiones y encaminamientos entre nodos) dependen exclusivamente de la existencia del servidor.

Algunos ejemplos de este tipo de redes son Napster y Audiogalaxy.

II. Redes P2P puras o totalmente descentralizadas

Las redes P2P de este tipo son las más comunes, siendo las más versátiles al no requerir de una gestión central de ningún tipo, lo que permite una reducción de la necesidad de usar un servidor central, por lo que se opta por los mismos usuarios como nodos de esas conexiones y también como almacenes de esa información. En otras palabras, todas las comunicaciones son directamente de usuario a usuario con ayuda de

un nodo (que es otro usuario) quien permite enlazar esas comunicaciones. Las redes de este tipo tienen las siguientes características:

- Los nodos actúan como cliente y servidor.
- No existe un servidor central que maneje las conexiones de red.
- No hay un enrutador central que sirva como nodo y administre direcciones.

Algunos ejemplos de una red P2P pura son: Kademlia, Ares Galaxy, Gnutella, Freenet y Gnutella2.

III. Redes P2P híbridas, semi-centralizadas o mixtas

En este tipo de red, se puede observar la interacción entre un servidor central que sirve como *hub* y administra los recursos de banda ancha, enrutamientos y comunicación entre nodos pero sin saber la identidad de cada nodo y sin almacenar información alguna, por lo que el servidor no comparte archivos de ningún tipo con ningún nodo. Tiene la peculiaridad de funcionar (en algunos casos como en *Torrent*) de ambas maneras, es decir, puede incorporar más de un servidor que gestione los recursos compartidos, pero también en caso de que el o los servidores que gestionan todo dejen de estar operativos, el grupo de nodos sigue en contacto a través de una conexión directa entre ellos mismos con lo que es posible seguir compartiendo y descargando más información en ausencia de los servidores. Este tipo de P2P sigue las siguientes características:

- Tiene un servidor central que guarda información en espera y responde a peticiones para esa información.
- Los nodos son responsables de hospedar la información (pues el servidor central no almacena la información), que permite al servidor central reconocer los recursos que se desean compartir, y para descargar esos recursos compartidos a los nodos que lo solicitan.
- Las terminales de enrutamiento son direcciones usadas por el servidor, que son administradas por un sistema de índices para obtener una dirección absoluta.

Algunos ejemplos de una red P2P híbrida son BitTorrent, eDonkey2000 y Direct Connect.

En este proyecto fin de carrera se estudiara las redes P2P puras o totalmente descentralizadas, ya que todos los se comportan de igual manera, es la red P2P más tolerante a fallos y la que en un futuro se basaran la mayoría de las redes que están implementadas en la red (Internet).

2.2. SISTEMAS BASADOS EN LA COOPERACIÓN

La necesidad de auto-organización de un entorno descentralizado y extremadamente dinámico como las redes P2P actuales, impone la necesidad de cooperación entre los miembros de dicha red.

El diseño de un entorno cooperativo se basa en el conjunto de procedimientos que los participantes aceptan como válidos para lograr sus objetivos. Esos procedimientos de cooperación pueden describirse como un "conjunto de reglas del juego", que integran el aporte de cada uno de los integrantes y refleja los puntos de cooperación. Se formulan a priori, por tanto, los criterios con los que el grupo está dispuesto a colaborar.

La mayoría de los sistemas P2P basados en cooperación usan sistemas de reputación [1] en los que se suelen utilizar seudónimos para identificar a los usuarios. Esto se hace para asegurar el anonimato del usuario. En este tipo de implementaciones de redes P2P se plantean varios problemas, el primer problema surge porque es fácil conseguir una nueva identidad mediante la creación de un nuevo seudónimo (también conocido como ataque Sybil), esto implica que los usuarios que tengan mala reputación podrían crearse un nuevo usuario cada vez que su reputación este considerada como perjudicial para la comunidad, de esta forma tan sencilla conseguirían mejorar su reputación cuando lo necesitasen, perjudicando a la comunidad. Otro problema acerca de los seudónimos es que un usuario le resulta fácil crear múltiples seudónimos, esto se puede utilizar para transferir archivos entre los seudónimos creados por dicho usuario, así consigue una alta reputación en la red sin merecerla.

Otro problema es asignar una nueva reputación, por ejemplo, supongamos que todos

los miembros de la comunidad tienen una reputación alta, entonces cuando se integre un nuevo usuario a la red, a este nuevo nodo se le asignara una reputación inicial que es inferior comparada con la reputación de los individuos cooperativos que componen la red P2P. Ahora, el nuevo usuario tiene un problema, ya que, al tener una baja reputación no podrá descargar archivos si los demás miembros también desean conseguir los mismos archivos, porque en los sistemas de reputación tienen prioridad los nodos que mejor reputación tienen, esta metodología complica la obtención de buena reputación para los nuevo usuarios. Este problema se incrementa en la descarga de ficheros, se seleccionan antes los ficheros de usuarios con alta reputación, por lo que para los usuarios con baja reputación les resulta difícil conseguir buena reputación al no poder proporcionar ficheros a la comunidad para conseguir buena reputación.

En los sistemas P2P descentralizados que usan reputación tienen una complejidad añadida, ¿quién asigna la reputación a los miembros de la comunidad? En esta clase de sistemas no hay un tercero que supervise el sistemas, por tanto, deberán ser los demás miembros quienes asignen la reputación al nodo que ha interactuado con la comunidad. Aquí volvemos a tener el problema anterior que si quienes califican la reputación conocen o no al nodo a calificar, ya que si le conocen le podrán dar una reputación mejor de la que realmente se ha merecido.

El modelo descrito en este proyecto se basa en un sistema de cooperación sin reputación, por lo que la infraestructura, el almacenamiento, los mensajes y la complejidad de computación se ven notablemente reducidas, además de no tener los problemas descritos anteriormente se simplifica la implementación. También se consigue el buscado anonimato y desaparecen todos los problemas anteriormente descritos.

2.3. TEORÍA DE JUEGOS

La teoría de juegos (TJ) es un área de la matemática aplicada que utiliza modelos para estudiar interacciones en estructuras formalizadas de incentivos (los llamados juegos) y llevar a cabo procesos de decisión. Sus investigadores estudian las estrategias óptimas así como el comportamiento previsto y observado de individuos en

juegos. Tipos de interacción aparentemente distintos pueden, en realidad, presentar estructuras de incentivos similares y, por lo tanto, se puede representar mil veces conjuntamente un mismo juego.

Desarrollada en sus comienzos como una herramienta para entender el comportamiento de la economía, la TJ se usa actualmente en muchos campos, desde la biología a la filosofía. Experimentó un crecimiento sustancial y se formalizó por primera vez a partir de los trabajos de John von Neumann y Oskar Morgenstern, antes y durante la Guerra Fría, debido sobre todo a su aplicación a la estrategia militar. Desde los setenta, la teoría de juegos se ha aplicado a la conducta animal, incluyendo el desarrollo de las especies por la selección natural. A raíz de juegos como el dilema del prisionero (DP) [2], en los que el egoísmo generalizado perjudica a los jugadores, la TJ se ha usado en economía, ciencias políticas, ética y filosofía. Finalmente, ha atraído también la atención de los investigadores en informática, usándose en inteligencia artificial y cibernética.

Aunque tiene algunos puntos en común con la teoría de la decisión, la TJ estudia decisiones realizadas en entornos donde interaccionan. En otras palabras, estudia la elección de la conducta óptima cuando los costes y los beneficios de cada opción no están fijados de antemano, sino que dependen de las elecciones de otros individuos. Un ejemplo muy conocido de la aplicación de la TJ a la vida real es el DP, popularizado por el matemático Albert W. Tucker, el cual tiene muchas implicaciones para comprender la naturaleza de la cooperación humana.

Los analistas de juegos utilizan asiduamente otras áreas de la matemática, en particular las probabilidades, las estadísticas y la programación lineal, en conjunto con la TJ. Además de su interés académico, la TJ ha recibido la atención de la cultura popular. La vida del matemático teórico John Forbes Nash, desarrollador del Equilibrio de Nash [3] y que recibió un premio Nobel.

Los juegos estudiados por la TJ están bien definidos por objetos matemáticos. Un juego consiste en un conjunto de jugadores, un conjunto de movimientos (o estrategias) disponible para esos jugadores y una especificación de recompensas para cada combinación de estrategias. Hay dos formas comunes de representar a los juegos:

1) Forma normal de un juego: la forma normal (o forma estratégica) de un juego es una matriz (Tabla 1) que muestra los jugadores, las estrategias, y las recompensas.

Hay dos tipos de jugadores; uno elige la fila y otro la columna. Cada jugador tiene tres estrategias, que están especificadas por el número de filas y el número de columnas. Las recompensas se especifican en el interior. El primer número es la recompensa recibida por el jugador de las filas (el Jugador 1 en nuestro ejemplo); el segundo es la recompensa del jugador de las columnas (el Jugador 2 en nuestro ejemplo). Si el jugador 1 elige lo acción de arriba y el jugador 2 elige la acción de la derecha entonces sus recompensas son -1 y 1, respectivamente.

Cuando un juego se presenta en forma normal, se presupone que todos los jugadores actúan simultáneamente o, al menos, sin saber la elección que toma el otro. Si los jugadores tienen alguna información acerca de las elecciones de otros jugadores el juego se presenta habitualmente en la forma extensiva. También existe una forma normal reducida. Ésta combina estrategias asociadas con el mismo pago.

2) Forma extensiva de un juego: la representación de juegos en forma extensiva modela juegos con algún orden que se debe considerar. Los juegos se presentan como árboles. Cada vértice o nodo representa un punto donde el jugador toma decisiones. El jugador se especifica por un número situado junto al vértice. Las líneas que parten del vértice representan acciones posibles para el jugador. Las recompensas se especifican en las terminaciones de las ramas del árbol.

Los juegos en forma extensiva pueden modelar también juegos de movimientos simultáneos. En esos casos se dibuja una línea punteada o un círculo alrededor de dos vértices diferentes para representarlos como parte del mismo conjunto de información (por ejemplo, cuando los jugadores no saben en qué punto se encuentran).

La forma normal da al matemático una notación sencilla para el estudio de los problemas de equilibrio, porque desestima la cuestión de cómo las estrategias son calculadas o, en otras palabras, de cómo el juego es jugado en realidad. La notación conveniente para tratar estas cuestiones, más relevantes para la teoría combinatoria de juegos, es la forma extensiva del juego.

2.3.1. CLASIFICACIÓN

Las categorías comunes incluyen:

1) Juegos simétricos y asimétricos: un juego simétrico es un juego en el que las recompensas por jugar una estrategia en particular dependen sólo de las estrategias que empleen los otros jugadores y no de quién las juegue. Si las identidades de los jugadores pueden cambiarse sin que cambien las recompensas de las estrategias, entonces el juego es simétrico. Muchos de los juegos 2×2 más estudiados son simétricos. Las representaciones estándar del juego de la gallina, el DP y la caza del ciervo son juegos simétricos.

Los juegos asimétricos más estudiados son los juegos donde no hay conjuntos de estrategias idénticas para ambos jugadores. Por ejemplo, el juego del ultimátum y el juego del dictador tienen diferentes estrategias para cada jugador; no obstante, puede haber juegos asimétricos con estrategias idénticas para cada jugador.

2) Juegos de suma cero y de suma no cero: en los juegos de suma cero el beneficio total para todos los jugadores del juego, en cada combinación de estrategias, siempre suma cero (en otras palabras, un jugador se beneficia solamente a expensas de otros). El juego piedra-papel-tijera (RPS), el *go*, el ajedrez, el póker y el juego del oso son ejemplos de juegos de suma cero, porque se gana exactamente la cantidad que pierde el oponente. Como curiosidad, el fútbol dejó hace unos años de ser de suma cero, pues las victorias reportaban 2 puntos y el empate 1 (considérese que ambos equipos parten inicialmente con 1 punto), mientras que en la actualidad las victorias reportan 3 puntos y el empate 1.

La mayoría de los ejemplos reales en negocios y política, al igual que el DP, son juegos de suma no cero, porque algunos desenlaces tienen resultados netos mayores o menores que cero. Es decir, la ganancia de un jugador no necesariamente se corresponde con la pérdida de otro. Por ejemplo, un contrato de negocios involucra idealmente un desenlace de suma positiva, donde cada oponente termina en una posición mejor que la que tendría si no se hubiera dado la negociación. La matriz de pagos de un juego es una forma conveniente de representación.

3) Juegos cooperativos: un juego cooperativo se caracteriza por un contrato que

puede hacerse cumplir. La TJ cooperativos da justificaciones de contratos plausibles. La plausibilidad de un contrato está muy relacionada con la estabilidad.

Dos jugadores negocian qué tanto quieren invertir en un contrato. La teoría de la negociación axiomática nos muestra cuánta inversión es conveniente para nosotros. Por ejemplo, la solución de Nash para la negociación demanda que la inversión sea justa y eficiente.

4) Simultáneos y secuenciales: los juegos simultáneos son juegos en los que los jugadores mueven simultáneamente o en los que éstos desconocen los movimientos anteriores de otros jugadores. Los juegos secuenciales (o dinámicos) son juegos en los que los jugadores posteriores tienen algún conocimiento de las acciones previas. Este conocimiento no necesariamente tiene que ser perfecto; sólo debe consistir en algo de información. Por ejemplo, un jugador 1 puede conocer que un jugador 2 no realizó una acción determinada, pero no saber cuál de las otras acciones disponibles eligió.

La diferencia entre juegos simultáneos y secuenciales se recoge en las representaciones discutidas previamente. La forma normal se usa para representar juegos simultáneos, y la extensiva para representar juegos secuenciales.

5) Juegos de información perfecta: un subconjunto importante de los juegos secuenciales es el conjunto de los juegos de información perfecta. Un juego es de información perfecta si todos los jugadores conocen los movimientos que han efectuado previamente todos los otros jugadores; así que sólo los juegos secuenciales pueden ser juegos de información perfecta, pues en los juegos simultáneos no todos los jugadores (a menudo ninguno) conocen las acciones del resto. La mayoría de los juegos estudiados en la TJ son juegos de información imperfecta, aunque algunos juegos interesantes son de información perfecta, incluyendo el juego del ultimátum y el juego del ciempiés. También muchos juegos populares son de información perfecta, incluyendo el ajedrez y el *go*.

La información perfecta se confunde a menudo con la información completa, que es un concepto similar. La información completa requiere que cada jugador conozca las estrategias y recompensas del resto pero no necesariamente las acciones.

En los juegos de información completa cada jugador tiene la misma

"información relevante al juego" que los demás jugadores. El ajedrez y el DP ejemplifican juegos de información completa. Los juegos de información completa ocurren raramente en el mundo real, y los teóricos de los juegos, usualmente los ven sólo como aproximaciones al juego realmente jugado.

6) Juegos de longitud infinita (superjuegos): por razones obvias, los juegos estudiados por los economistas y los juegos del mundo real finalizan generalmente tras un número finito de movimientos. Los juegos matemáticos puros no tienen estas restricciones y la teoría de conjuntos estudia juegos de infinitos movimientos, donde el ganador no se conoce hasta que todos los movimientos se conozcan.

2.4. ALGORITMOS GENÉTICOS Y EVOLUTIVOS

Un algoritmo es una serie de pasos organizados que describe el proceso que se debe seguir, para dar solución a un problema específico.

En los años 1970, de la mano de John Henry Holland, surgió una de las líneas más prometedoras de la inteligencia artificial (IA), la de los algoritmos genéticos (AG) [4]. Son llamados así porque se inspiran en la evolución biológica y su base genético-molecular. Estos algoritmos hacen evolucionar una población de individuos sometiéndola a acciones aleatorias semejantes a las que actúan en la evolución biológica (mutaciones y recombinaciones genéticas), así como también a una selección de acuerdo con algún criterio, en función del cual se decide cuáles son los individuos más adaptados, que sobreviven, y cuáles los menos aptos, que son descartados. También es denominado algoritmos evolutivos, e incluye las estrategias de evolución, la programación evolutiva y la programación genética.

Un AG es un método de búsqueda dirigida basada en probabilidad. Bajo una condición muy débil (que el algoritmo mantenga elitismo, es decir, guarde siempre al mejor elemento de la población sin hacerle ningún cambio) se puede demostrar que el algoritmo converge en probabilidad al óptimo. En otras palabras, al aumentar el número de iteraciones, la probabilidad de tener el óptimo local en la población tiende a uno.

Los AG establecen una analogía entre el conjunto de soluciones de un problema,

llamado fenotipo, y el conjunto de individuos de una población natural, codificando la información de cada solución en una cadena, generalmente binaria, llamada cromosoma. Los símbolos que forman la cadena son llamados los genes. Cuando la representación de los cromosomas se hace con cadenas de dígitos binarios se le conoce como genotipo. Los cromosomas evolucionan a través de iteraciones, llamadas generaciones. En cada generación, los cromosomas son evaluados usando alguna medida de aptitud o utilidad. Las siguientes generaciones (nuevos cromosomas), llamada descendencia, se forman utilizando dos operadores genéticos, de cruzamiento y de mutación.

Un AG puede presentar diversas variaciones, dependiendo de cómo se aplican los operadores genéticos (cruzamiento, mutación), de cómo se realiza la selección y de cómo se decide el reemplazo de los individuos para formar la nueva población. En general, el algoritmo consistiría en los siguientes pasos:

- **Inicialización**: Se genera aleatoriamente la población inicial, que está constituida por un conjunto de cromosomas los cuales representan las posibles soluciones del problema. En caso de no hacerlo aleatoriamente, es importante garantizar que dentro de la población inicial, se tenga la diversidad estructural de estas soluciones para tener una representación de la mayor parte de la población posible o al menos evitar la convergencia prematura.
- **Evaluación**: A cada uno de los cromosomas de esta población se aplicará la función de aptitud o utilidad para saber qué tan "buena" es la solución que se está codificando.
- **Condición de término**: El AG se deberá detener cuando se alcance la solución óptima, pero ésta generalmente se desconoce, por lo que se deben utilizar otros criterios de detención. Normalmente se usan dos criterios: ejecutar el AG un número máximo de iteraciones (generaciones) o detenerlo cuando no haya cambios en la utilidad total de la población. Mientras no se cumpla la condición de término se hace lo siguiente:
    - **Selección**: Después de saber la aptitud de cada cromosoma se procede a elegir los cromosomas que serán cruzados en la siguiente generación. Los

cromosomas con mejor aptitud tienen mayor probabilidad de ser seleccionados.

- **Cruzamiento** El cruzamiento es el principal operador genético, representa la reproducción sexual, opera sobre dos cromosomas a la vez para generar dos descendientes donde se combinan las características de ambos cromosomas padres.
- **Mutación** modifica al azar parte del cromosoma de los individuos, y permite alcanzar zonas del espacio de búsqueda que no estaban cubiertas por los individuos de la población actual.
- **Reemplazo** una vez aplicados los operadores genéticos, se sustituye la población anterior por la de nueva creación para conformar la generación siguiente.

En la Figura 1. Se muestra el diagrama de flujo de un algoritmo genético.

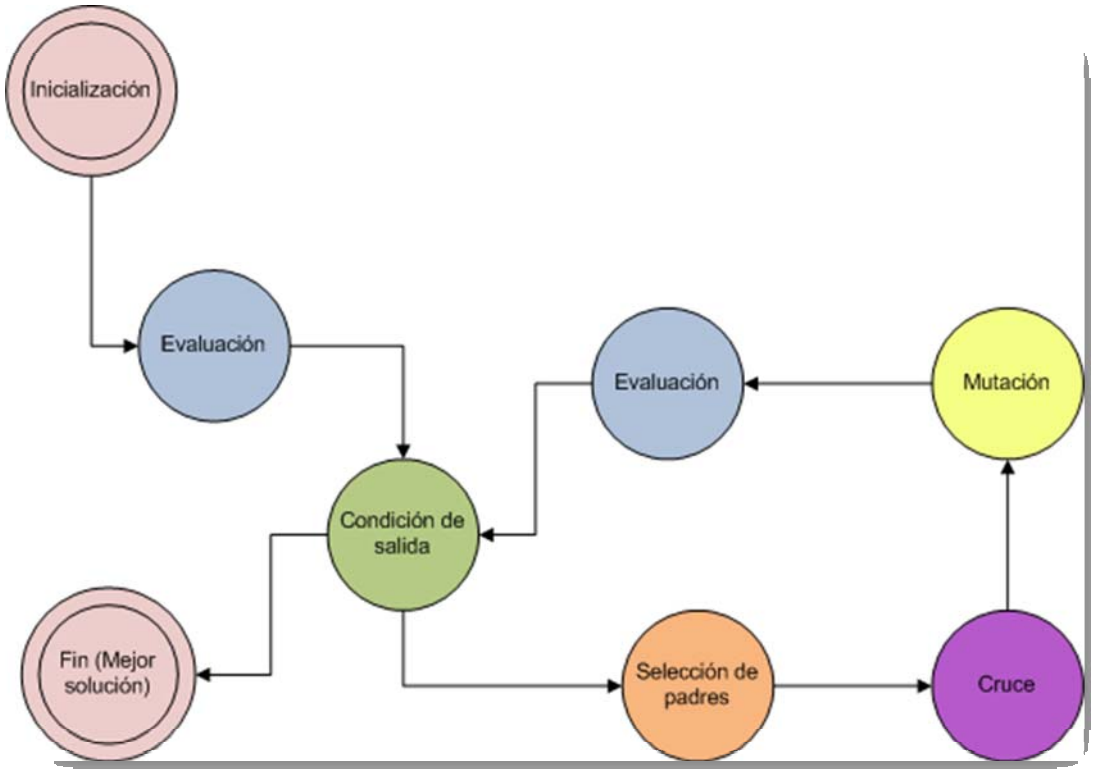


Figura 1: Dibujo de un algoritmo genético

2.4.1. SELECCIÓN, CRUCE Y MUTACIÓN

Como se menciono anteriormente, en el proceso de **Selección** se eligen individuos a partir de una población, con el objeto de ser evolucionados. Existen varios mecanismos de selección. Los más frecuentemente utilizados son presentados a continuación.

1) Selección de ruleta (*roulette-wheel selection*): es también conocida como selección proporcional a la función de utilidad (*fitness proportionate selection*).

Sea N el número de individuos existentes y f_i la utilidad o *fitness* del i-ésimo individuo. La probabilidad asociada a su selección está dada por:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

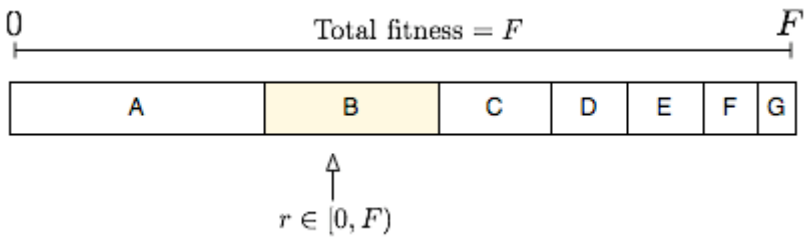


Figura 2: Selección de un individuo

En la Figura 2 se muestra el sumatorio total de los *fitness* de cada nodo, como se puede observar el *fitness* A es el mayor, por lo que tendrá mayor probabilidad de ser seleccionado que cualquier otro nodo. Para seleccionar los nodos se genera un número aleatorio, el cual indicara que nodo se debe seleccionar, de esta manera es más probable que los nodos con un *fitness* mayor sean seleccionados.

Esta selección permite que los mejores individuos sean elegidos con una mayor probabilidad, pero al mismo tiempo permite a los peores individuos ser elegidos, lo cual puede ayudar a mantener la diversidad de la población, en contraste con la selección por truncamiento.

Un problema de la selección de ruleta se presenta cuando existe una pequeña

fracción de la población (en el límite, sólo un individuo) que posee una medida de desempeño excesivamente superior al resto. Esto provoca pérdida de diversidad y puede conducir a convergencia prematura pues la mayor parte de los individuos seleccionados será una copia de los pocos predominantes. En este caso es preferible utilizar selección basada en ranking o selección por torneo.

2) Selección por truncamiento: en esta selección las soluciones candidatas son ordenadas según su función de utilidad, y una proporción p (por ejemplo = $\frac{1}{2}, \frac{1}{3}, \frac{1}{4}$,...) de los individuos con mejor *fitness* es seleccionada y reproducida $\frac{1}{p}$ veces. Esta selección es menos sofisticada que la mayoría de los métodos de selección, y generalmente no es usada en la práctica.

3) Selección basada en ranking: en esta selección los individuos se ordenan según su medida de utilidad y luego son asignados con una segunda medida de *fitness*, inversamente proporcional a su posición en el ranking (esto es, otorgando una mayor probabilidad a los mejores). Los valores de esta segunda asignación pueden ser lineales o exponenciales. Finalmente, los individuos son seleccionados proporcionalmente a esta probabilidad.

Este método disminuye el riesgo de convergencia prematura que se produce cuando se utiliza selección de ruleta en poblaciones con unos pocos individuos con valores de utilidad muy superiores a las del resto.

4) Selección por torneo: esta selección se efectúa mediante un torneo (comparación) entre un pequeño subconjunto de individuos elegidos al azar desde la población.

Los beneficios de este tipo de selección son la velocidad de aplicación (dado que no es necesario evaluar ni comparar la totalidad de la población) y la capacidad de prevenir, en cierto grado, la convergencia prematura. La principal desventaja es la necesidad de establecer el parámetro correspondiente al tamaño del subconjunto.

El operador de **Cruce** es un operador genético utilizado en los AG para generar variación en la programación de un cromosoma o cromosomas de una generación a la siguiente. Es análogo a la recombinación de la reproducción sexual biológica, en la que los algoritmos genéticos se inspiran. Existen muchas técnicas de cruzamiento para organismos que utilizan diferentes estructuras para almacenar los datos. Las enumeramos a continuación:

1) Cruzamiento en un punto: se selecciona un punto en el vector del primer parental. Todos los datos más allá de este punto en el vector cromosoma del organismo se intercambiarán entre los dos organismos parentales. Los organismos resultantes son los hijos:

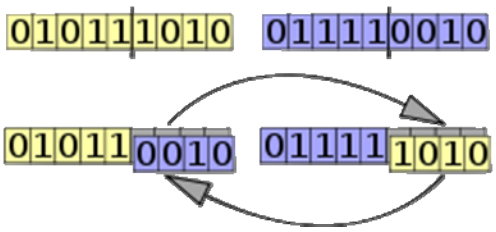


Figura 3: Cruzamiento en un punto

2) Cruzamiento en dos puntos: el cruzamiento en dos puntos requiere seleccionar dos puntos en los vectores de los organismos parentales. Todos los datos entre los dos puntos se intercambian entre los organismos parentales, creando dos cromosomas hijos:

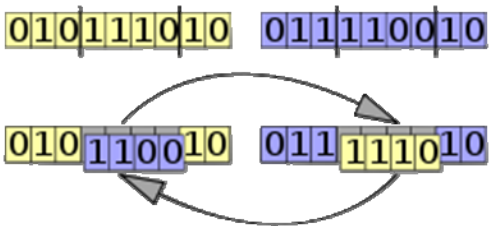


Figura 4: Cruzamiento en dos puntos

3) Corte y empalme: otro variante de cruzamiento, el enfoque "cortar y empalmar", ocasiona un cambio de la longitud de los vectores de los hijos. La razón

para esta diferencia es que se selecciona un punto de corte diferente para cada vector parental:

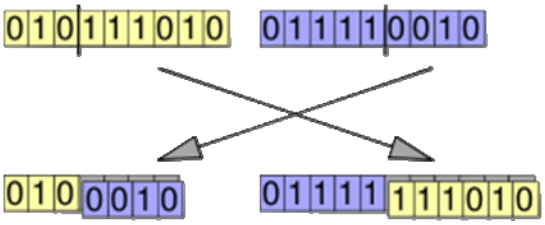


Figura 5: Corte y empalme

4) Cruzamiento uniforme y uniforme medio: en ambos de estos esquemas los dos padres se combinan para producir los descendientes. En el esquema de cruzamiento uniforme (UX por el inglés *Uniform Crossover*), los bits del vector del cromosoma se comparan individualmente entre ambos padres. Los bits se intercambian con una probabilidad fijada, usualmente 0.5.

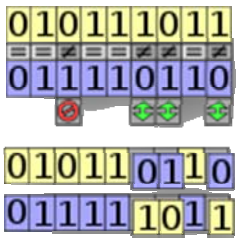


Figura 6: Cruzamiento uniforme

En el esquema de cruzamiento uniforme medio (HUX del inglés *Half Uniform Crossover*), exactamente la mitad de los bits que son diferentes se intercambian. Por esto se necesario calcular la distancia de *Hamming* (número de bits diferentes). Este número se divide entre dos, y el número resultante es la cantidad de bits diferentes que tiene que ser intercambiada entre los parentales.

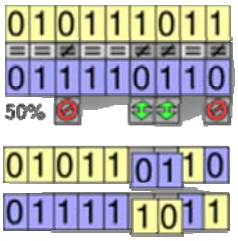


Figura 7: Cruzamiento uniforme medio

5) Cruzamiento de cromosomas ordenados: dependiendo de cómo representa la solución el cromosoma, un cambio directo puede no ser posible. Un caso de ello se da cuando el cromosoma es una lista ordenada, como la lista ordenada de las ciudades visitadas en el problema del viajero. Un punto de cruzamiento se selecciona en los padres. Puesto que el cromosoma es una lista ordenada, un cambio directo introduciría duplicados y extraería candidatos necesarios de la lista. En cambio, el cromosoma hasta el punto de cruzamiento se retiene para cada padre. La información después del punto de cruzamiento se ordena como está ordenada en el otro padre. Por ejemplo, si nuestros dos padres son ABCDEFGHI e IGAHFDBEC y nuestro punto de cruzamiento se establece después del cuarto carácter, entonces los hijos que resultan serían ABCDIGHFE e IGAHBCDEF.

La **Mutación** es un operador genético usado para mantener la diversidad genética de una población. Es análogo a la mutación biológica. Un ejemplo clásico de operador de mutación consiste en la modificación de un bit del cromosoma de algunos individuos. Esta modificación se efectúa con una probabilidad preestablecida, llamada probabilidad de mutación.

El propósito de la mutación es proveer un mecanismo para escapar de los óptimos locales, así como desplazar a los individuos hacia zonas del espacio de búsqueda que no pueden ser alcanzadas por medio de otros operadores genéticos.

CAPÍTULO 3. MARCO TEÓRICO BASADO EN TEORÍA DE JUEGOS EVOLUTIVA PARA EL ANÁLISIS DE REDES P2P PURAS

El DP representa uno de los problemas más populares y estudiados de la TJ [5]. El juego es estratégicamente muy complejo, al mismo tiempo que las reglas son muy sencillas. El DP consiste en dos jugadores que al mismo tiempo tienen que elegir entre dos acciones, denotadas por D (deserción) y por C (cooperación). Ambos jugadores son conscientes de la perdida y el beneficio que cada acción les reporta y como estos dependen de lo que el otro jugador elige. La Tabla 2 describe los valores de pagos obtenidos en un juego de DP, donde los componentes de primero y segundo de cada vector de pagos corresponden a la utilidad obtenida por el jugador 1 y 2 respectivamente. Fijándose en la tabla, es evidente observar la estrategia que se define por las acciones (D, D) (ambos jugadores juegan con la acción D cada pago obtenido con valor P), constituye un equilibrio de Nash. También se ve claro que jugando (C, C) habría reportado a cada jugador una mejor recompensa, ya que R > P. En el caso de que el juego DP se repita muchas veces (DP iterado [6]), si el número de rondas se conoce antes por los dos jugadores, la teoría de juegos prueba que los dos jugadores por defecto (juegan D) una y otra vez, no importa cuántas veces se repita el juego. Solamente cuando los jugadores juegan un número indefinido de veces o al azar, la cooperación puede surgir.

El dilema de iguales: las diferentes tentativas se han creado para definir un marco formal en el cual se analizara los niveles de cooperación descentralizada en los sistemas P2P. Varios de ellos se han basado en el juego clásico de los conceptos teóricos. Al respecto, se ha identificado dos enfoques diferentes: (enfoque 1) cuando el sistema en su conjunto se considera y analiza como un juego de N-jugadores en el que los nodos/jugadores tienen la opción de ser colaborador o *free-rider* [7]; y (enfoque 2) cuando los protocolos individuales que los nodos utilizan para interactuar son analizados como juegos individuales ([8]). En este último, las conclusiones sobre el análisis completo del sistema se derivan del análisis individual de las interacciones de igual a igual. Por otra parte, en la mayoría de los formalismos, el análisis del comportamiento de iguales ha sido basado en los jugadores enfrentándose en un dilema

social entre cooperar o *free-ride*, equivalente en muchos casos al el DP iterado descrito anteriormente. Además, para analizar formalmente la aparición de un comportamiento de cooperación entre jugadores, eran un conjunto similar de pagos como los de las Tabla **2**, en algunos casos, ha sido utilizado para representar el pago obtenido por el jugador cooperador/*free-rider*. Sin embargo, una observación clave es que los pagos se basaron aplicando incentivos, en sistemas de remisión de reputación y en redes particulares, formalmente justifican la aparición de conducta de cooperación entre iguales ([9] [10] [11]) en sistemas P2P. Por último, todos los formalismos anteriores consideran la participación en cada tipo de dilema de interacciones como obligatorio, en otras palabras, a los nodos solo se les permitió cooperar o *free-rider*, asique todas las estrategias consistieron en una secuencia de estas dos acciones.

La sociedad mundial de piedra-papel-tijeras: el club *Paper Scissors Stone* fue fundado en 1842, en Inglaterra. En 1925, el club llego a más de 10.000 miembro y el nombre fue cambiado a *The Worl RPS Society*. De hecho, el juego RPS es otro juego clásico con reglas muy sencillas pero complejas para analizar estratégicamente. El juego consiste en dos jugadores quienes tienen que elegir simultáneamente entre tres acciones diferentes, la consecución de diferentes pagos para cada uno de ellos, también depende de la acción del otro jugador. La Tabla **3** resume los vectores de pago para los jugadores 1 y 2 (los valores numéricos son elegidos para ilustrar mejor el resultado del dilema).

En este caso, la salida del juego RPS no se puede predecir con certeza al no haber una acción que ofrezca las mejores garantías. Por lo tanto, se alcanza un equilibrio cuando cada jugador elige todas las posibles acciones con una probabilidad de $\frac{1}{3}$. En cuando a la versión iterada del juego RPS, los resultados de la TJ de juegos establecen las mismas estrategias (e. $\frac{1}{3}$ piedra, $\frac{1}{3}$ papel, $\frac{1}{3}$ tijeras) para cada jugador como la mejor respuesta a la estrategia de cualquier otro jugador, alcanzando un equilibrio. Esto puede ser interpretado como jugadores cooperativos en repetidos escenarios, para lograr un resultado justo del juego.

		Jugador 2	
		Cooperate	Defeat
Jugador 1	Cooperate	(R, R)	(S, T)
	Defeat	(T, S)	(P, P)

Tabla 2: Tabla de pagos para el juego DP

		Jugador 2		
		Paper	Rock	Scissors
Jugador 1	Paper	(0, 0)	(1, -1)	(-1, 1)
	Rock	(-1, 1)	(0, 0)	(1, -1)
	Scissors	(1, -1)	(-1, 1)	(0, 0)

Tabla 3: Tabla de pagos para el juego RPS

La propuesta está basada en introducir una tercera opción dentro del tradicional dilema de iguales, de ahí la introducción de un nuevo tipo de nodo: *loner*. Un *loner* es un nodo que decide desconectarse del sistema en varias rondas. Los *loners* tienen el riesgo de desconectarse estratégicamente de la red buscando maximizar sus propios beneficios. El modelo presentado en este proyecto servirá para demostrar que la participación voluntaria de los jugadores en el Dilema Social *Cooperate/Free-rider* es suficiente para mantener la cooperatividad entre ellos, y que asegura niveles aceptables de rendimiento de la red. La participación voluntaria puede impedir la no-cooperatividad de los nodos y para superar el dilema social habrá que crear patrones de cooperación [12]. La propuesta se basa en describir una serie de los experimentos llevados a cabo para el analizar el efecto de que los *loners* tienen en la aparición de la cooperación. Por lo tanto, los jugadores tienen tres distintas acciones en cada interacción individual. Ahora, ellos deben elegir entre cooperar, no-cooperar o temporalmente desconectarse. Se muestra la matriz de pagos en la Tabla 3, que sirve como modelo de una comunidad P2P descentralizada y que este mecanismo simple pero efectivo, no requieren de incentivos, sin basarse en un sistema de confianza/reputación y operar de manera eficiente bajo el anonimato. La única hipótesis es la racionalidad de

los nodos, es decir, los jugadores actúan por su propio interés.

3.1. EL NUEVO DILEMA RPS ENTRE IGUALES

En general, en las comunidades P2P puras, los cooperadores son expuestos a la explotación y la invasión de los nodos no-cooperativos. A menos que se proporcione algún tipo de incentivo a los nodos que cooperan, los no-cooperativos siempre producen un mayor beneficio en contra de los comportamientos cooperativos. Por ejemplo, en un intercambio de archivos en la comunidad P2P, los *free-rider* solicitaran y descargaran archivos de los recursos de nodos cooperativos y sin dar nada a cambio. Para evitar el *free-riding*, una solución común es aplicar una reputación para castigar y aislar este tipo de comportamiento antisocial. Lamentablemente, esto representa un desafío y grandes gastos generales en sistemas P2P grandes y descentralizados. Sin embargo, en el supuesto de que el objetivo de los nodos es maximizar su pago individualmente, si a los nodos se les da la opción de desconectarse cada cierto periodo de tiempo (esto parece una hipótesis razonable), el fenómeno se comporta de la siguiente forma:

- Una vez que los no-cooperativos incrementan y comienzan la invasión de jugadores que cooperan, la opción *loners* se convierte cada vez más atractiva para los jugadores honestos. Por lo tanto, previamente los cooperadores trataran de escapar de la explotación, temporalmente se desconectaran de la red.

- Sin embargo, tan pronto como los *loners* dominan el sistema, los nodos no-cooperativos comienza a morir de hambre y la opción de la cooperación es la más rentable.

- Por último, cuando los cooperadores dominan la comunidad, los nodos tienen la tentación de no-cooperar lo que se convierte de nuevo en la estrategia más rentable.

Por lo tanto, se alterna siguiendo la siguiente secuencia: cooperativo » *loners* » no-cooperativo » cooperativo, etc.... De esta manera, la interacción subyacente entre nodos puede ser modelado como un dilema RPS, en la que cada jugador tiene tres posibles acciones: *loner* (tijeras), no-cooperativo (papel) y cooperativo (piedra). La Tabla 3 resume el escenario cuando:

- El comportamiento no-cooperativo contra el rendimiento de los nodos cooperativo tienes más valor que cooperativo contra cooperativo.

- El comportamiento no-cooperativo contra los nodos *loners* incurre en una

perdida para los nodos no-cooperativo.

- El comportamiento simétrico ofrece el mismo pago para cualquiera de los nodos.

Por otra parte, la Figura 8 representa el dominio cíclico de las tres estrategias posibles en los escenarios iterados que produce patrones auto-organizados. Los mismos pagos de la Tabla 3 son considerados como, que cada estrategia juega con la probabilidad de $\frac{1}{3}$, que conduce a una convivencia estable entre cooperativo, no-cooperativo y *loners*. Aunque unas pocas variantes son introducidas, esto constituirá la base del enfoque: los *loners* evitan formar una comunidad invadida por los no-cooperativo.

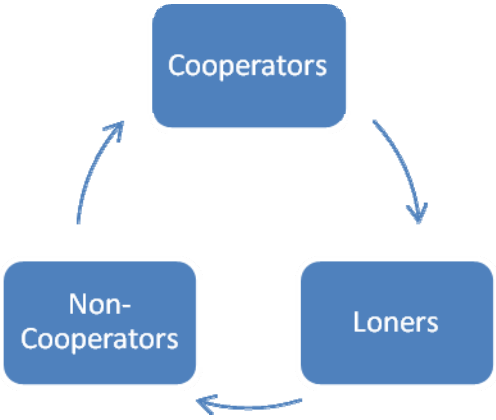


Figura 8: Ciclo de acciones para el juego RPS

3.2. MARCO TEÓRICO DEL JUEGO

El tradicional juego RPS ha sido ampliamente estudiado dentro del ámbito de la TJE y clásica [13]. Al respecto, en esta sección presentaremos los resultados más significativos evitando siempre que sea posible cualquier notación matemática. Estos resultados, más adelante, nos ayudan a razonar sobre la aparición de cooperatividad entre nodos anónimos no incentivados sin el marco de análisis descrito en la sección 4.

Nosotros denotamos $\sigma_{RPS} = (\sigma_1, \sigma_2)$ el perfil estratégico en el que el jugador 1 juega la estrategia σ_1 y el jugador 2 juega la estrategia σ_2 en el juego RPS. Por lo tanto, tal perfil estratégico σ_{RPS} dictamina como juegan los jugadores 1 y 2 y por lo tanto determinan los resultados del juego.

En el juego RPS el perfil estratégico que constituye el equilibrio de una única interacción RPS, es aquel en el que $\sigma_1 = \sigma_2 = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. En otras palabras, cada jugador elige entre las opciones posibles (piedra, papel, tijeras) con una probabilidad de $\frac{1}{3}$. Este perfil estratégico de denota como σ^*_{RPS}. En la versión iterada del juego RPS el mismo perfil estratégico σ^*_{RPS} surge como la solución más estable con algunas propiedades interesantes:

- σ^*_{RPS} es una estrategia evolutiva débilmente estable: no protege a cooperativos de pequeñas incursiones de los jugadores no-cooperativos pero previene la invasión total.

- σ^*_{RPS} es un equilibrio adecuado: el perfil protege a todos los jugadores individuales de incurrir en el costo fatal y letal, aun cuando los jugadores con la estrategia de nunca cooperar son parte del juego.

- σ^*_{RPS} permite a los jugadores estar más alerta a posibles desviaciones del perfil, aunque permite pequeñas variaciones.

Nota Importante: se va a demostrar que los resultados recién descritos sobre el tradicional juego RPS pueden ser aplicados al dilema RPS de iguales definido en la sección 3.1. Esto es: la cooperación puede subsistir, aunque las interacciones no se repitan, en la presencia de anónimos no-cooperativos, donde los nodos no tienen memoria y no son incentivados a cooperar. Sin embargo, los siguientes dos aspectos nos han impedido directamente derivar a las mismas conclusiones:

1) Si bien la dinámica de nodos dentro de la comunidad P2P puede ser modelada usando la matriz de pago simétrica (Tabla 3), en este caso el dilema RPS de iguales los roles de los dos jugadores (*provider* y *requester*) no son simétricos. De hecho, un nodo podría convertirse en *provider* de un *requester* en la próxima interacción, y viceversa. Así, la evolución de una comunidad P2P solo depende de la proporción de *requester* y *provider*.

2) El trabajo experimental previo ha revelado que el modelado de el conjunto de posibles acciones de los jugadores (*provider*-honesto, *provider*-deshonesto, *loner*, *requester*-honesto, *requester*-deshonesto) no conduce a un reconocible ciclo de estrategias dominantes, por lo que no se ha podido extraer mas conclusiones. La propuesta, preserva el juego de tres acciones RPS principal al mismo tiempo que se crea

perfiles estratégicos para todos los roles posibles de los jugadores (sección 4.1).

CAPÍTULO 4. DISEÑO E IMPLEMENTACIÓN DE UN SIMULADOR DE REDES P2P BASADO EN COMPUTACIÓN EVOLUTIVA

4.1. MODELO FORMAL

En esta sección se describe los fundamentos del formalismo propuesto. A continuación enumeramos las suposiciones técnicas del sistema P2P:

1) Los jugadores: en el modelo los nodos son racionales (nunca se comportan en contra de su propio interés) y no incentivados. Además, las comunidades modelo tienen un límite máximo de participantes N y con diferentes proporciones de cooperación, ej. 25% y 75% de nodos cooperativos. Un número de experimentos se llevaron a cabo escalando miles de nodos.

2) Los enlaces y conexiones: una extensa experimentación ha puesto de manifiesto como la estructura de red afecta al nivel de cooperación en comunidades P2P descentralizadas [14]. En el modelo se considera una red débilmente al azar, es decir, generada aleatoriamente la red P2P, de manera que se ven pocos cambios durante la vida de la comunidad. Por otra parte, para poder hacer frente al dinamismo de la red en forma de nodos uniéndose y dejando la red, diferentes tasas de cambio son consideradas durante la vida de la comunidad. Las desconexiones de nodos son representadas como estrategias *loners*.

3) Conocimiento y comunicaciones de nodos: las comunicaciones entre nodos son inherentemente probabilísticas. Por otra parte, los nodos no tienen conocimientos iniciales sobre el tipo de sus oponentes, así podría ayudar en el proceso de elegir la estrategia con mejor respuesta. Del mismo modo, el modelo no depende de ningún jugador que disponga de capacidad para reconocer a otros jugadores o recordar acciones pasadas. En un nivel intuitivo, ya que no hay memoria y no hay referencias a nodos anteriores, no se puede aplicar un ataque.

El modelo se define a partir de los siguientes fundamentos:

1) Rondas: en el modelo, una ronda "r" se considera como un punto en el tiempo en donde los nodos de una comunidad interactúan simultáneamente en parejas, jugando al dilema RPS de "iguales" descrito en la sección 3.1. Dado un número fijo de jugadores N, se utiliza un grafo dirigido $G_r = (V, E)$ para denotar la interacción de la red en la ronda "r". El conjunto de vértices $V = \{1,.., N\}$ representa los N jugadores y el conjunto de aristas E representa sus conexiones, donde $e_{ij} = 1$ si hay una conexión establecida entre los nodos i y j, y sino $e_{ij} = 0$. Tras la asunción de una red débilmente al azar, el conjunto de aristas E cambia en cada ronda con una pequeña probabilidad.

2) Estrategia: se define A = {no-cooperativo, cooperativo, *loner*} como el conjunto de posibles acciones para cada nodo en cualquier ronda "r". Esto es representado como una cadena codificada de bits, equivalente al código genético o cromosoma. Una estrategia para el nodo "i" es una matriz $s^i - [s_{ir}] \in \mathcal{M}_{1xk}$ (Eq.1) que dictamina las acciones escogidas en cualquiera de las "k" rondas consecutivas, donde l determina el rol de los nodos, tal que: $\forall 1 \leq r \leq k$, $s_{1r} \in A$ es la acción adoptada en la ronda "r" si los nodos juegan como *provider* y $s_{2r} \in A$ es la acción adoptada en el ronda "r" si los nodos interactúan como *requester*.

$$s^i = \begin{pmatrix} s_{11} & \cdots & s_{1k} \\ s_{21} & \cdots & s_{2k} \end{pmatrix} donde\ s_{ir} \in A$$

3) Los pagos de los nodos: en cada ronda "r", los pagos son obtenidos por competir en el juego RPS contra otros miembros de la sociedad definido en el grafico G_r. El grafo dirigido G_r define las dos interacciones de nodos y el rol que cada uno debe adoptar. La Tabla 3 describe los pagos individuales donde el primer componente de cada vector de pago representa el pago para el jugador *provider* y *requester*. Como se menciono antes el conjunto de aristas E en G_{r+1} difiere poco de G_r. El pago obtenido por la estrategia se calcula como la suma aritmética total de cada pago individual encontrado en cada ronda.

4.2. ENTORNO DE DESARROLLO

El entorno de desarrollo se ha elegido Visual Studio 2008, ya que es una herramienta novedosa, muy completa, fácil de manejar y además potente en la creación de aplicaciones. Esta herramienta es la proporcionada por Microsoft para realizar aplicaciones de todo tipo.

Se ha decidido por esta plataforma de desarrollo por dos razones, la primera razón es por la facilidad que existe a la hora de crear aplicaciones con una interfaz gráfica agradable y con una buena presentación, de esta manera se pueden presentar los resultados con gráficas para poder obtener las conclusiones de forma fácil y sencilla, y la segunda es por la facilidad que hay en comunicar o interactuar con librerías de C++, ya que este lenguaje crea librerías bastante rápidas comparados con otros lenguajes, esta característica se puede considerar un requisito ya que en el algoritmo implementado ejecuta gran cantidad de bucles que aumentan el tiempo de ejecución y la complejidad del algoritmo.

El desarrollo de la aplicación se ha realizado un Intel (R) Core (TM)2 Duo de 1,66 GHz, con una memoria RAM de 2 Gb. El sistema operativo elegido ha sido Windows Vista Business SP1 de 32 bits.

Para poder realizar el simulador se ha necesitado el Visual Studio 2008 y su plataforma MS .NET Framework 3.5 SP1, y sus correspondientes paquetes que se instalan conjuntamente. Para la realización de graficas se ha instalado un paquete llamado MS Chart, este paquete es proporcionado de forma gratuita Microsoft ya que complementa a .NET Framework 3.5. El .NET Framework es el conjunto de librerías que usa Visual Studio para ayudar a crear aplicaciones, es similar a Java Runtime Environment, pero en vez de ser propiedad de Sun Microsystems, Framework .NET es propiedad de Microsoft.

Una vez instalado el Visual Studio y el paquete MS Chart se puede abrir el proyecto de la aplicación para poder ver el código. En Visual Studio el código generado en librerías y ejecutables se divide en proyectos, la aplicación creada se divide en tres proyectos:

- El ejecutable que contiene la interfaz grafica, este proyecto se está escrito en el

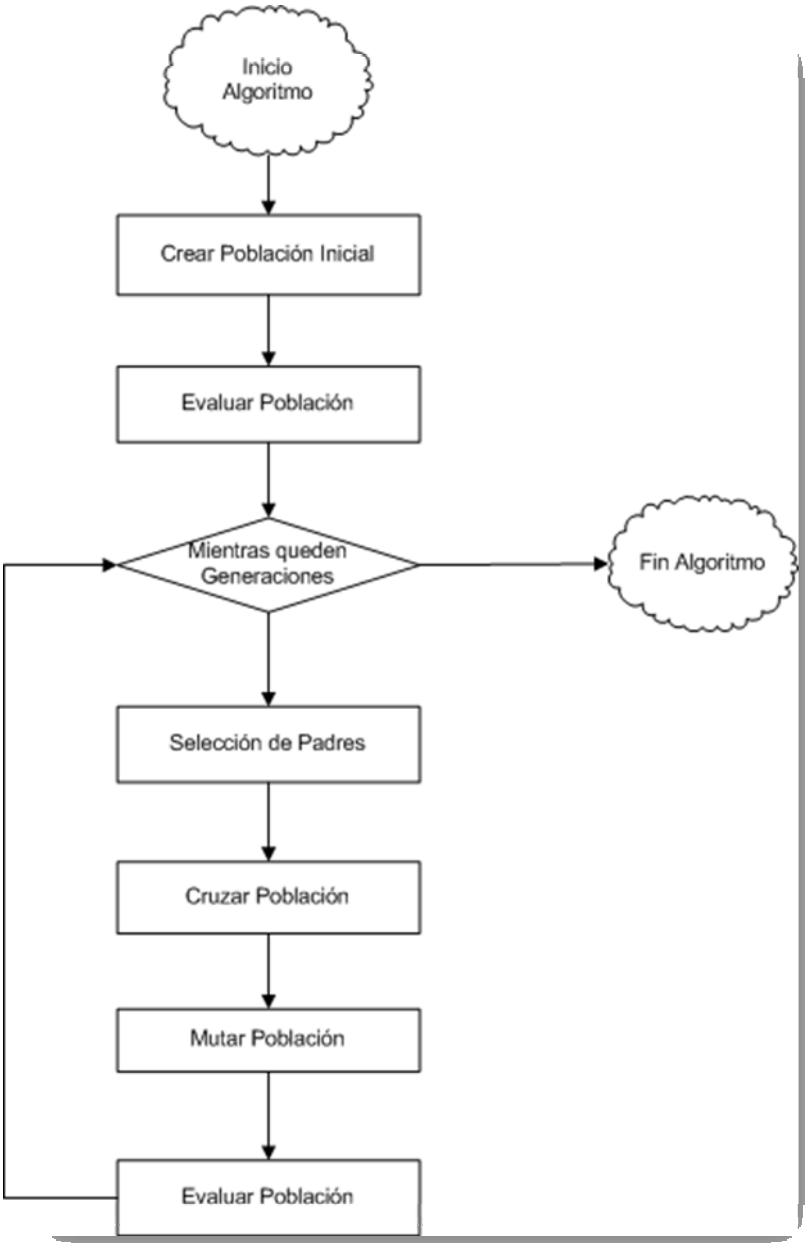
leguaje C#, este lenguaje es parecido a C pero sin la posibilidad de usar punteros para que resulte más fácil la creación de aplicaciones, además de otras ventajas significativas que hacen más fácil el trabajo del programador para crear una aplicación atractiva.

- Interfaz: este proyecto contiene las dos interfaces y la clase simulación que usa el algoritmo. La primera interfaz se usa para comunicar los parámetros de entrada al algoritmo genético, y la segunda interfaz se usa para notificar de los resultados de una generación al ejecutable para que este dibuje los datos recibidos en la grafica. La clase simulación contiene básicamente los parámetros de entrada y la población, así se consigue tener toda la información de la simulación en una clase.
- Algoritmo genético: y por último este proyecto contiene toda la implementación del algoritmo genético, es la librería (.dll) que acompaña al ejecutable, en esta librería se está codificado todo el algoritmo mediante funciones con significado dentro del algoritmo, es decir, como se ha podido ver en la sección anterior hay funciones de creación de población, de evaluación, generación de relaciones, cruce, mutación, etc... estas funciones tienen su homóloga en el algoritmo implementado.

4.3. DIAGRAMA DE FLUJO

El diagrama de flujo principal corresponde básicamente al AG, como se puede observar contiene las fases principales que todo AG debe implementar para la consecución de los objetivos perseguidos, el diagrama de flujo (Figura 9) es el siguiente:

Figura 9: Diagrama de flujo del Algoritmo Genético



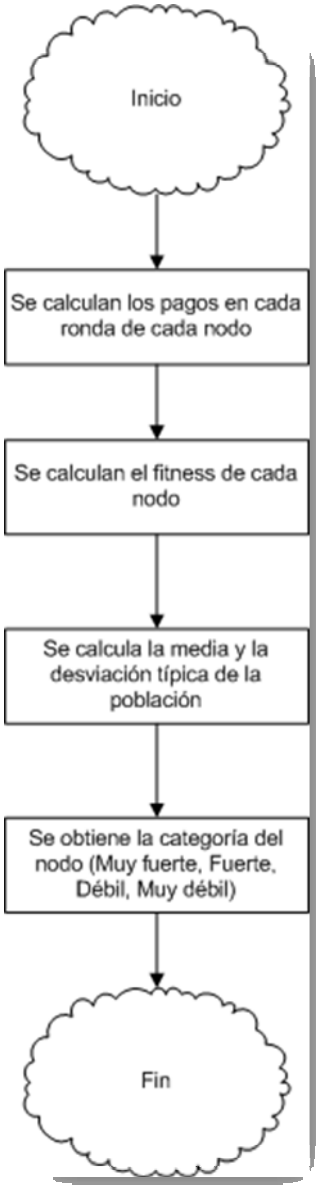
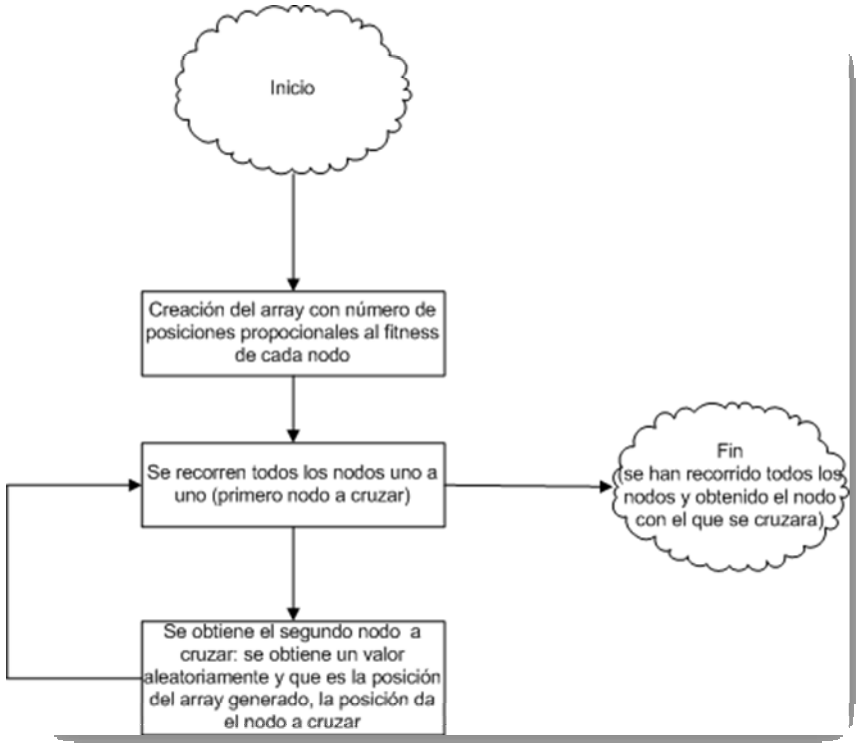


Figura 10: Proceso de evaluación

A continuación se van a detallar los diagramas para los procesos de selección (Figura 11), cruce (Figura 12) y evaluación (Figura 10), ya que son las partes del AG que más influyen en los en la consecución de los objetivo previsto del algoritmo.

Figura 11: Proceso de selección de padres



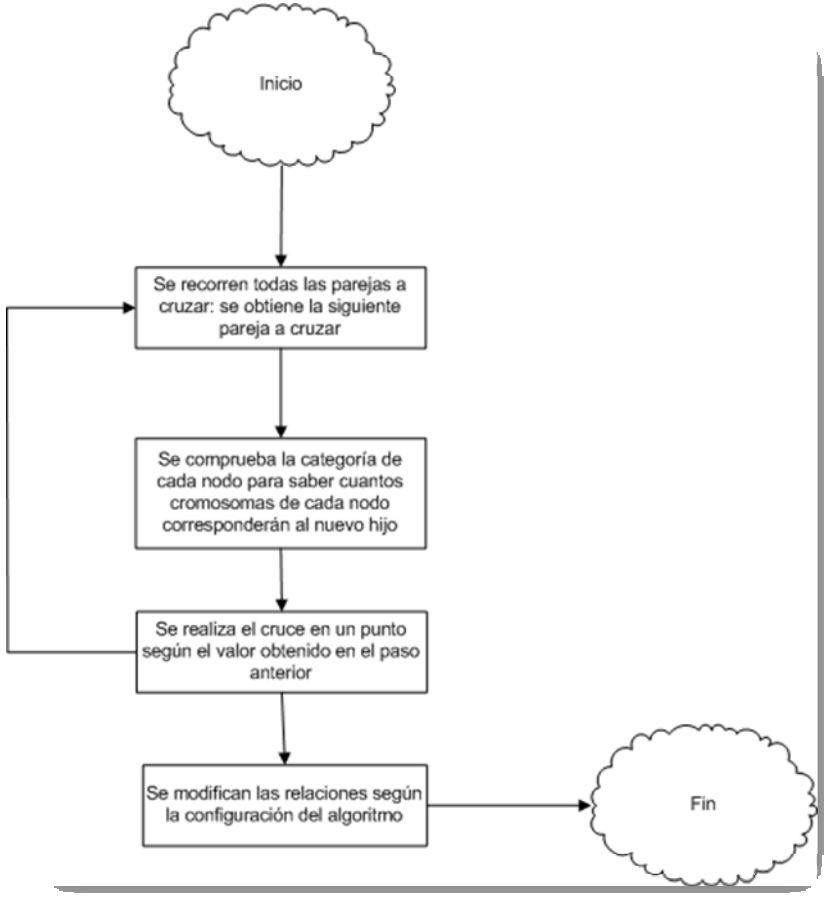


Figura 12: Proceso de cruce

4.4. PROCEDIMIENTOS Y ESTRUCTURAS PRINCIPALES

A continuación se enumeran las clases principales usadas así como los procedimientos y funciones usados para el desarrollo del algoritmo:

1.- CreatePopulation (int nInitialLoners, int nInitialNonCooperators, int nInitialCooperators)

El primer paso en esta función es obtener el número de cada clase de nodos (*Loners*, *NonCooperators* y *Cooperators*, esta clasificación se realiza observando cual la mayoría de acciones que contiene el nodo, estas acciones son las que el nodo ejecuta en cada una de sus jugadas, estas acciones también son *Loner*, *NonCooperator* y *Cooperator*) a crear, se consigue gracias a los parámetros (nInitialLoners, nInitialNonCooperators, nInitialCooperators) pasados en la función.

Después se generan uno a uno aleatoriamente los nodos, una vez creado un nodo se obtiene su clase (*Loner*, *NonCooperator*, *Cooperator*), y se aumenta el contador correspondiente.

Si el contador es mayor al número de nodos a crear de dicha clase se repite la generación de este nodo. Este proceso se repite hasta crear todos los nodos de la población.

2.- GenerateMovesRandom ()

Una vez creada toda la población, se generan las relaciones entre los nodos aleatoriamente. Para cada nodo se genera su relación *Provider* y su relación *Requester*, es decir, por cada uno de los nodos de la población se generan dos relaciones, de tal manera que la relación *Provider* se relaciona con la relación *Requester* de otro nodo, y viceversa. También se tiene en cuenta que ningún nodo se quede sin relación y que no se relacione consigo mismo.

Este procedimiento genera aleatoriamente las relaciones entre los nodos. En cada jugada se relacionan todos los nodos entre sí (*Rol Provider - Rol Requester*), es decir, cada nodo tiene dos relaciones en una jugada una por cada rol.

GenerateMovesMaintainPopulation ()

Esta función mantiene el número de nodos de la población, es decir, que durante la ejecución del algoritmo no se elimina ningún nodo por baja que sea su *fitness*. De esta manera se va modificando la población sin alterar el número de nodos.

Esta función genera las jugadas para la siguiente generación, en este caso se aplica un porcentaje de mutación de las relaciones (*Mutation rate for relations*) el cual se define en la interfaz gráfica.

El primer paso es obtener los nodos a mutar su relación. Después se generan las relaciones para cada nodo a mutar. Todo este proceso se repite para cada jugada que contienen los nodos.

GenerateMovesNotMaintainPopulation (ArrayList^ aNodesDeleted)

En esta función se utiliza cuando la población ha podido disminuir el número de nodos. La población disminuye cuando se marca en la interfaz grafica la opción *Keep even population size*, además de esta opción hay que seleccionar una de las dos acciones a realizar cuando la población resultante sea impar, estas dos acciones son “Añadir un nodo de categoría Débil” (*Adding a low fitness node*) y “Eliminar un nodo de categoría Muy Débil” (*Eliminating a very low fitness node*). Se eliminan nodos cuando se ejecuta el algoritmo con esta opción y después de la evaluación de la población se obtienen nodos de categoría Muy Débil, estos nodos se eliminaran para la siguiente población.

Las categorías de los nodos se obtienen gracias a su *fitness*, se realiza la media de los *fitness* de cada nodo, después se halla la desviación típica, y por último se van clasificando uno a uno los nodos de la población, la clasificación es (X = media, σ = desviación típica):

- Categoría Muy fuerte: *Fitness* > X + σ.
- Categoría Fuerte: X – σ < *Fitness* < X + σ.
- Categoría Débil: X – 2 * σ < *Fitness* < X – σ.
- Categoría Muy débil: *Fitness* < X – 2 * σ.

Este procedimiento genera las relaciones pero antes reajusta las relaciones existentes, ya que es posible que se hayan eliminado nodos y por esto es posible que los

nodos que siguen en la población estén relacionados con algún nodo eliminado, por esta razón se reajustan las relaciones de los nodos.

Una vez reajustadas las relaciones, se generan las relaciones que faltan. Después se mutan las relaciones según el parámetro de la interfaz gráfica, de la misma forma que en la función anterior.

3.- EvaluatePopulation ()

La primera parte de esta función es calcular los pagos de cada jugada para cada nodo, una vez obtenidos los pagos se obtiene el *fitness* total de cada nodo.

A continuación, se calcula la media y la desviación típica a partir de los *fitness* de cada nodo. Finalmente se clasifican los nodos en: Muy Fuerte, Fuerte, Débil y Muy Débil.

4.- CrossPopulationMaintainPopulation ()

El primer paso en esta función es crear el array con selección proporcional al *fitness* de cada nodo. De esta manera, se elegirán los mejores individuos con una mayor probabilidad, así se generan la siguiente población tendrá más probabilidad de contener mejores individuos (Método de selección “ruleta rusa”).

El segundo paso es realizar el cruce, el método de cruce elegido es el de cruzamiento en un punto. El punto para el cruce es elegido según la categoría del nodo (muy fuerte, fuerte, débil y muy débil), el punto de cruce se define en la interfaz grafica en la sección *Crossover and Mutation Parameters*, apartado *Fitness related crossover* por medio de porcentajes.

La selección de padres se produce de la siguiente manera, se recorren todos los nodos, para cada nodo se elige un nodo aleatoriamente a partir del array con selección proporcional al *fitness*. Así, todos los nodos se cruzan con el nodo que por probabilidad es mejor, mejorando la población en cada generación.

Y por último se realiza la generación de jugadas según haya elegido el usuario en la interfaz gráfica.

CrossPopulationNotMaintainPopulation(bool fActionForPopulationOdd, int nPercentMutationMoves)

La primera parte en este procedimiento es obtener la población a cruzar, es decir, hay que obtener los nodos que se cruzaran. También se tiene en cuenta si la población es impar, para poder obtener una población par, se procederá según haya elegido el usuario en la sección *Population Data* en el apartado *Keep even population size*.

Una vez obtenida la población par a cruzar se crea el array con selección proporcional al *fitness* de cada nodo, y se siguen los pasos explicados en la función anterior.

5.- MutatePopulationActions ()

Esta función recorre todas las acciones de todos los nodos, y realiza una obtención aleatoria para saber si hay que mutar la acción del nodo correspondiente. En el caso de que haya que mutar, se elige aleatoriamente una jugada que no sea la que tenia inicialmente.

Por ejemplo, si la acción 12 del nodo 24, es *Loner*, y hay que mutar esta acción, se elije aleatoriamente una de las otras dos posibles (*Non-Cooperator* ó *Cooperator*).

4.5. SIMULACIÓN

Para la realización de la simulación, se ha creado una clase que contiene toda la información necesaria para la ejecución de algoritmo, esta clase contiene las siguientes propiedades:

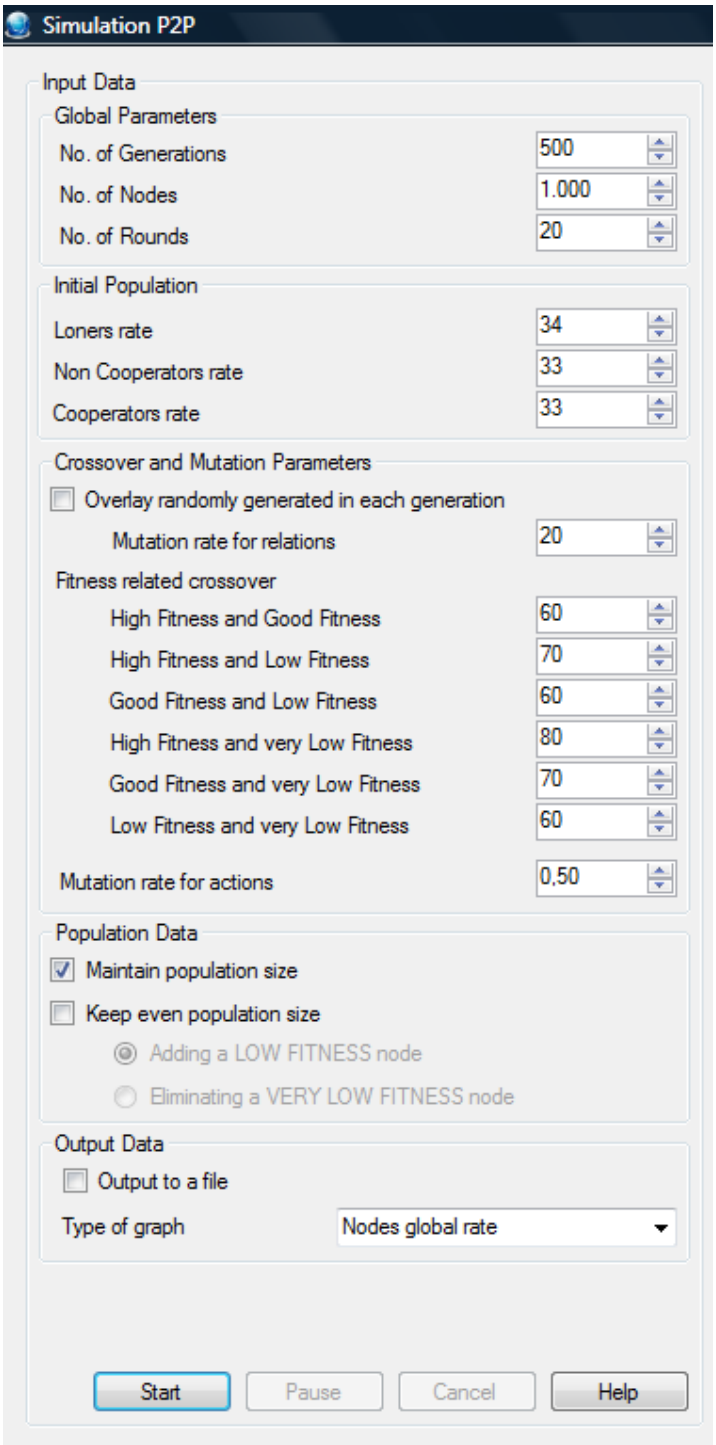


Imagen 1: Interfaz para la configuración de la simulación

1. *NumberGenerations*: define el número de generaciones a ejecutar por el algoritmo.
2. *CounterGenerations*: es el contador de generaciones, cuando este valor sea igual a la anterior el algoritmo finalizara.
3. *NumberNodes:* número inicial de nodos que contiene la población en la simulación.
4. *NumberMoves:* número de jugadas que tiene cada nodo de la población.
5. *GenerateMovesRandom*: indicador para generar las relaciones en cada generación aleatoriamente.
6. *NumberMutationMoves:* porcentaje de mutación de las relaciones que se aplica en cada generación de las relaciones de los nodos pertenecientes a la población.
7. *NumberMovesVeryStrongWithStrong:* porcentaje de jugadas a cruzar cuando los nodos seleccionados son de categoría muy fuerte y fuerte.
8. *NumberMovesVeryStrongWithWeak:* porcentaje de jugadas a cruzar cuando los nodos seleccionados son de categoría muy fuerte y débil.
9. *NumberMovesVeryStrongWithVeryWeak:* porcentaje de jugadas a cruzar cuando los nodos seleccionados son de categoría muy fuerte y muy débil.
10. *NumberMovesStrongWithWeak:* porcentaje de jugadas a cruzar cuando los nodos seleccionados son de categoría fuerte y débil.
11. *NumberMovesStrongWithVeryWeak:* porcentaje de jugadas a cruzar cuando los nodos seleccionados son de categoría fuerte y muy débil.
12. *NumberMovesWeakWithVeryWeak:* porcentaje de jugadas a cruzar cuando los nodos seleccionados son de categoría débil y muy débil.
13. *PecentMutationActions:* porcentaje de mutación para los cromosomas de cada nodo.
14. *Average:* media de todos los *fitness* calculados en una generación.
15. *StandardDeviation:* desviación típica de la generación calculada conjuntamente con la media (propiedad anterior).
16. *CountVeryStrong:* contador de nodos muy fuertes en una generación.
17. *CountStrong:* contador de nodos fuertes en una generación.
18. *CountWeak:* contador de nodos débiles en una generación.

19. *CountVeryWeak:* contador de nodos muy débiles en una generación.
20. *GenerateFile:* indicador para generar un fichero de salida con la información relevante de cada generación.
21. *InterfaceResults:* interfaz para notificar los resultados a la interfaz grafica y así poder crear la grafica correspondiente y la salida en modo de texto.
22. *Population:* población de la simulación, esta población contiene un conjunto de nodos, cada nodo contiene el *TotalFitness* del nodo, la categoría, y las jugadas que tiene el nodo. Cada jugada contiene la acción *Provider*, la relación *Provider* (es el nodo con el que está relacionado, por ejemplo si el nodo 1 se relaciona con el nodo 34, en el nodo 1 la relación *Provider* será 34, mientras que en el nodo 34 la relación *Requester* será 1), y el pago para la acción *Provider*, también contiene estas propiedades para la acción *Requester*.

CAPÍTULO 5. ANÁLISIS DE LOS RESULTADOS

El objetivo es simular la evolución de una comunidad P2P por medio de computación evolutiva (en particular, un algoritmo genético). Se ha implementado el concepto de competencia RPS como una manera de estudiar la evolución de la estrategia de iguales. Los objetivos son:

1.- Identificar las estrategias estables evolutivas.

2.- Identificar *free-riders* cuya estrategia consiste en jugar como no-cooperativo cuando se asigna el rol de un *provider* y cooperativo cuando se asigna el rol de un *requester*.

3.- Identificar *free-riders* cuya estrategia consiste en jugar como *loners* para la mayoría de las rondas y se convierten en *requester* cooperativos al final.

4.- Identificar los *free-riders* cuya estrategia consiste en jugar primero como *requester* cooperativos y como *loners* al final.

5.- Medir la esperanza de vida de los nodos con diferentes estrategias.

A continuación, se describe brevemente el marco experimental (ver Imagen 2):

5.1. PARAMETROS EXPERIMENTALES

1) Parámetros globales: una población inicial aleatoria de N individuos (Imagen 2 – *No. of Nodes*) se creara de forma que, cada individuo de la población está compuesto por una matriz de estrategias. Además, un grafo aleatorio G_1 = (V, E) se definirá al inicio. En cada ronda "r" (Imagen 2 – *No. of Rounds*), los nodos interactúan en parejas siguiendo sus estrategias asignadas y de acuerdo con las conexiones establecidas en G_r; cada interacción consiste en una dilema RPS de iguales. El marco también permite definir que en cada iteración se genere aleatoriamente la matriz de interacciones o que se modifique según un porcentaje (Imagen 2 – *Mutation rate for relation or at random*). Se estudia la evolución de estrategias sobre un número de generaciones (Imagen 2 – *No.*

of Generations).

2) Función *Fitness*: la función *fitness* necesita ser definida acorde con el problema a resolver. En cualquier generación el valor *fitness* de los individuos se calcula con el fin de clasificarlos en, por ejemplo, la siguiente clasificación de cuatro: muy fuerte, fuerte, débil y muy débil *fitness*.

3) Reproducción: el funcionamiento del sistema en gran medida depende de dos operadores genéticos básicos, es decir, cruce y mutación. Del mismo modo, la aplicación de estos operadores se basa en el problema específico [15]. Las respuestas a las siguientes preguntas son necesarias: (a) ¿Cómo elegir los padres?, (b) ¿Cómo se combinan?, y por último (c) ¿Cómo la tasa de mutación afecta a la generación?. Los mejores cromosomas, que contribuyen con sus genes heredados para la siguiente generación, deben ser seleccionados de la población de padres a cruzar, asumiendo "La supervivencia del más apto" [16]. Aquí se aplica usando el *Fitness Proporcionate Selection* (FPS), también conocido como proceso de selección de ruleta [17]. En este esquema, la probabilidad de selección de cada padre es proporcional a su valor *fitness*.

Existen muchas técnicas de cruce que combinan el código genético de los padres. La técnica usada ha sido cruzamiento en un punto, que consiste en la selección de un punto de cruce en los cromosomas de ambos padres, este punto de cruce depende del *fitness* de los padres (Imagen 2 – *Fitness related crossover*). El código genético del hijo resultante se obtiene copiando los primeros cromosomas del primer padre hasta el punto de cruce y a continuación se copian los cromosomas del segundo padre comenzando en el punto de cruce calculado. Por último, se considera una tasa de mutación de acciones muy baja (Imagen 2 – *Mutation rate for actions*), por ejemplo, 0,05%.

4) El *fitness* de los individuos se calcula como la suma aritmética de los pagos obtenidos por los nodos en cada ronda. La media y la desviación típica nos ayuda a medir el estado de la comunidad (ver Figura 13). Por ejemplo, los individuos, cuyo *fitness* es una desviación estándar por encima de la media, son considerados más adecuados para la reproducción que individuos que tienen un *fitness* dos veces la desviación típica por debajo de la media.

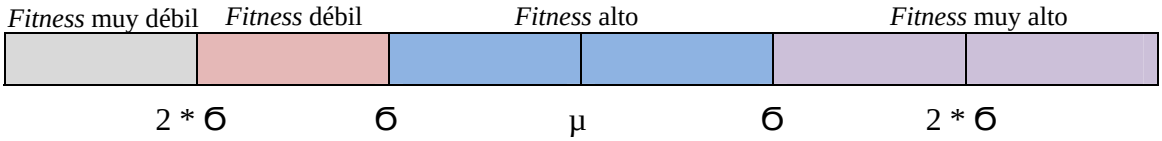


Figura 13: Sistema de evaluación aplicado

5) Criterio de la evolución: dado que el objetivo es llegar a la estabilidad del sistema contra estrategias invasoras (dominantes), no debemos concentrarnos en encontrar el valor *fitness* más satisfactorios, pero si en demostrar la cooperatividad de los nodos en el nuevo dilema RPS sin mecanismos de incentivos.

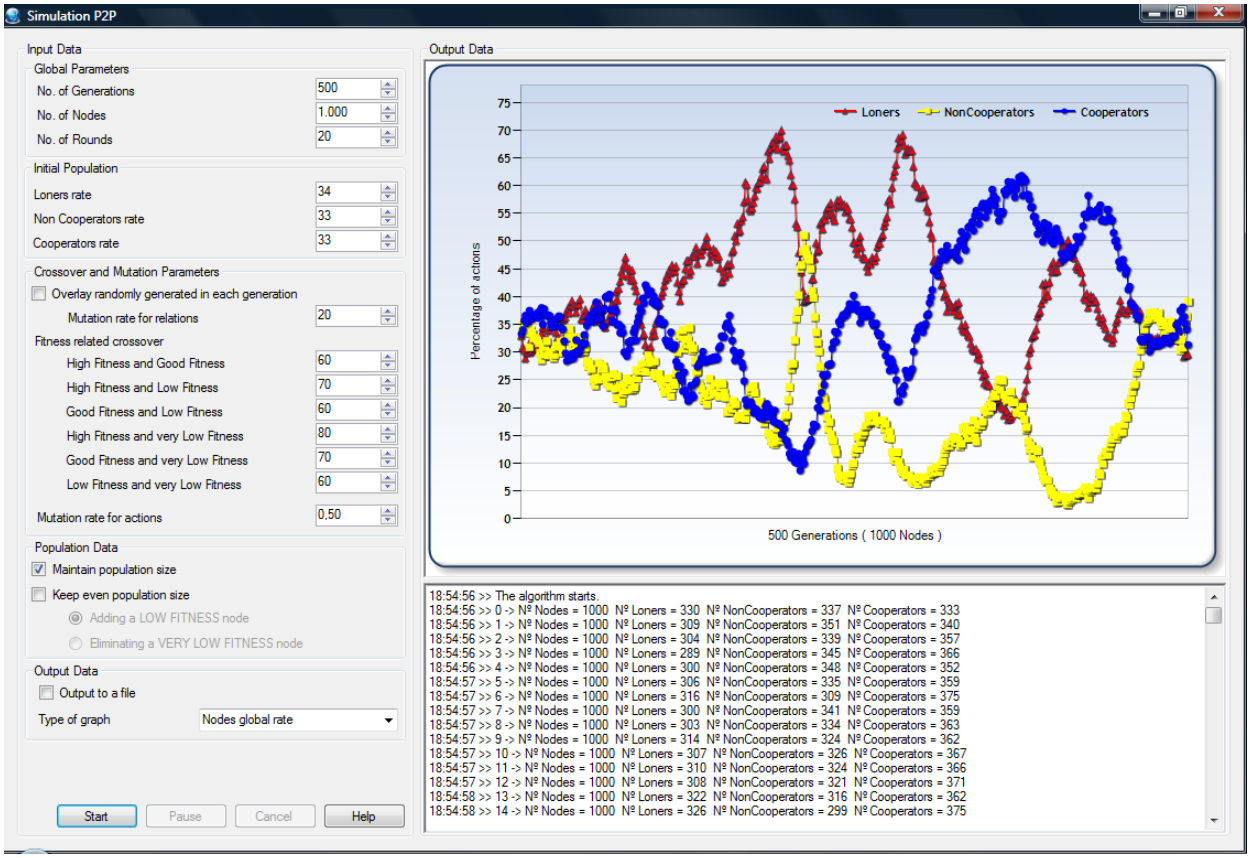


Imagen 2: Imagen de la aplicación completa

5.2. RESULTADOS EXPERIMENTALES

Los experimentos se realizaron con un procesador Intel (R) Core (TM)2 Duo de 1,66 GHz, con 1 GB de RAM y el sistema operativo es Windows Vista Business SP1 de 32 bits, y usando la plataforma de desarrollo MS .NET Framework 3.5 SP1.

Las siguientes figuras resumen un conjunto de resultados de la experimentación para probar que los nodos opcionalmente participan en el Dilema Social *Cooperate/Free-ride* basta para mantener la cooperatividad entre nodos y garantiza los niveles de rendimiento de la red. Durante todos los experimentos el tamaño de la población se mantuvo en 1.000 nodos, el número de generaciones se fijo en 500, y el número de rondas en 20, en todas las simulaciones se han mantenido la población (es decir, no nodos muy débiles no mueren), se han elegido estos parámetros porque es necesario un numero grande de generaciones para observar a donde tiende la población, y un número de nodos grande ya que es lo que se puede encontrar en cualquier red P2P descentralizada. En todas las figuras la línea con triángulos son individuos actuando como *loners*, la línea con cuadrados son nodos no-cooperativos y la línea con círculos para los cooperativos.

El primer grafico (**Imagen 3**) a analizar es el de una población estable, es decir, al menos el 33% para cada estrategia. Las interacciones de los nodos cambian con una probabilidad de 0,2, mientras que la tasa de mutación de acciones se establece en un 0,5%. De este grafico se puede sacar la conclusión de que partiendo de una estrategia estable no hay estrategia que invada a las otras dos estrategias, por lo tanto se puede decir, que se ha producido una evolución estable de la red, aunque no ha sido una evolución perfecta, es decir, no se han ido alternando cada vez una las estrategias que sería lo ideal.

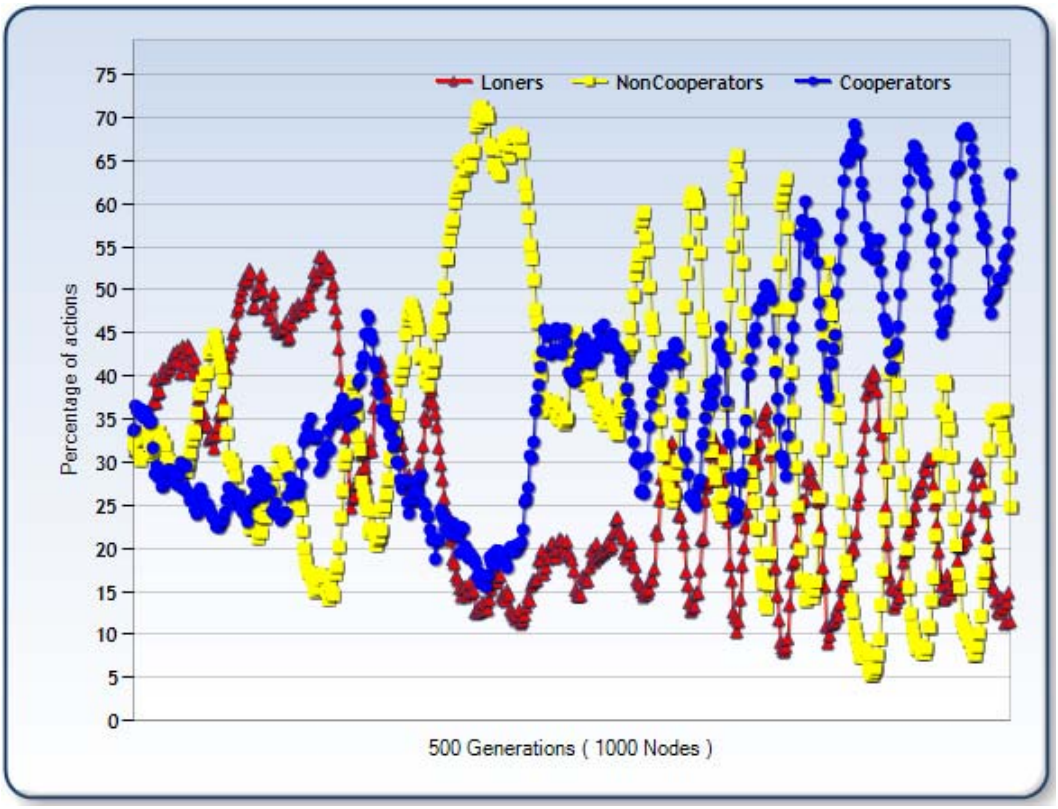


Imagen 3: Simulación de ejemplo

Las siguientes tres simulaciones se han realizado buscando una invasión de los nodos una estrategia sobre las demás, para intentar demostrar que en tal caso no subsistiría la red. Para ello se iniciara una ejecución con un porcentaje mayor de estos respecto a los otros dos. La primera configuración (**Imagen 4**) va a ser: 50% *loners*, 25% de no-cooperativos, y 25% de cooperativos, la segunda ejecución (**Imagen 5**): 50% de no-cooperativos, 25% *loners* y 25% de cooperativos, y la última configuración (**Imagen 6**): 25% *loners*, 25% de no- cooperativos, y 50% de cooperativos, los demás parámetros se han mantenido iguales que en la ejecución anterior.

Como se puede observar en los gráficos, un alto índice en una estrategia determinada no indica que la red este en peligro, ya que ninguna estrategia ha conseguido invadir la población en ninguna de las tres simulaciones, sino que después de unas generaciones se han ido alternando dando estabilidad a la población. Como conclusión a estas tres graficas se puede decir que partiendo de una población donde una estrategia es bastante superior a las otras dos, la red después de unas iteraciones se estabiliza proporcionando la subsistencia a la red.

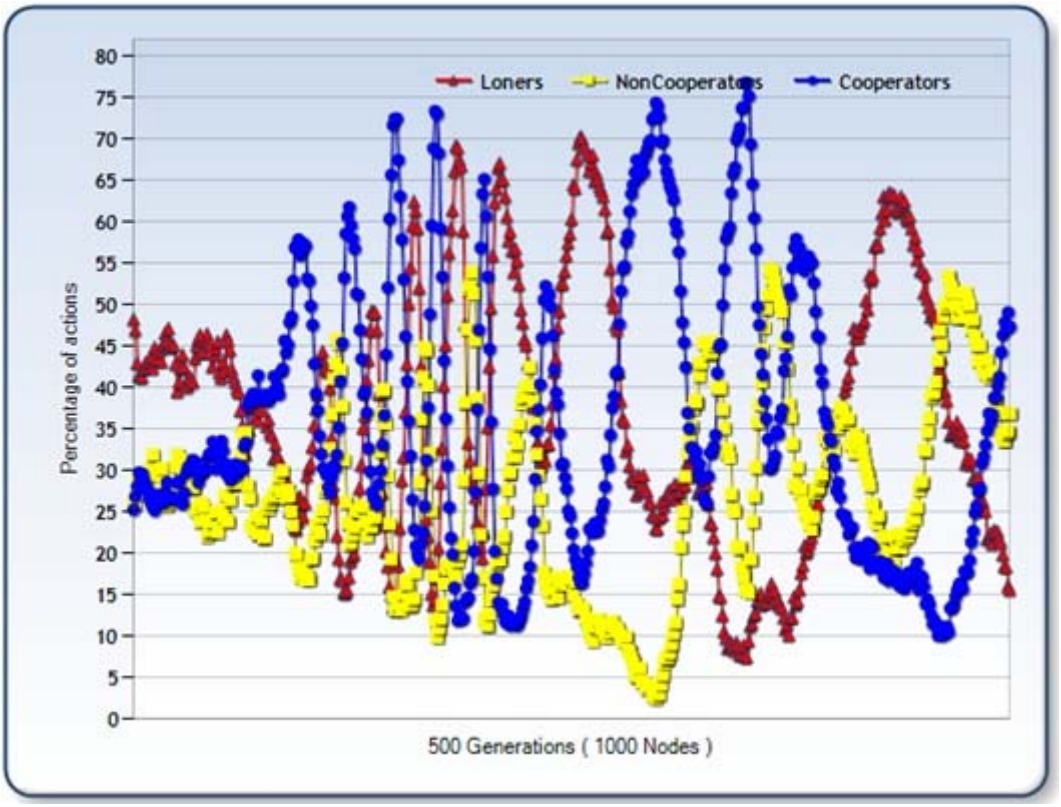


Imagen 4: Simulación de ejemplo

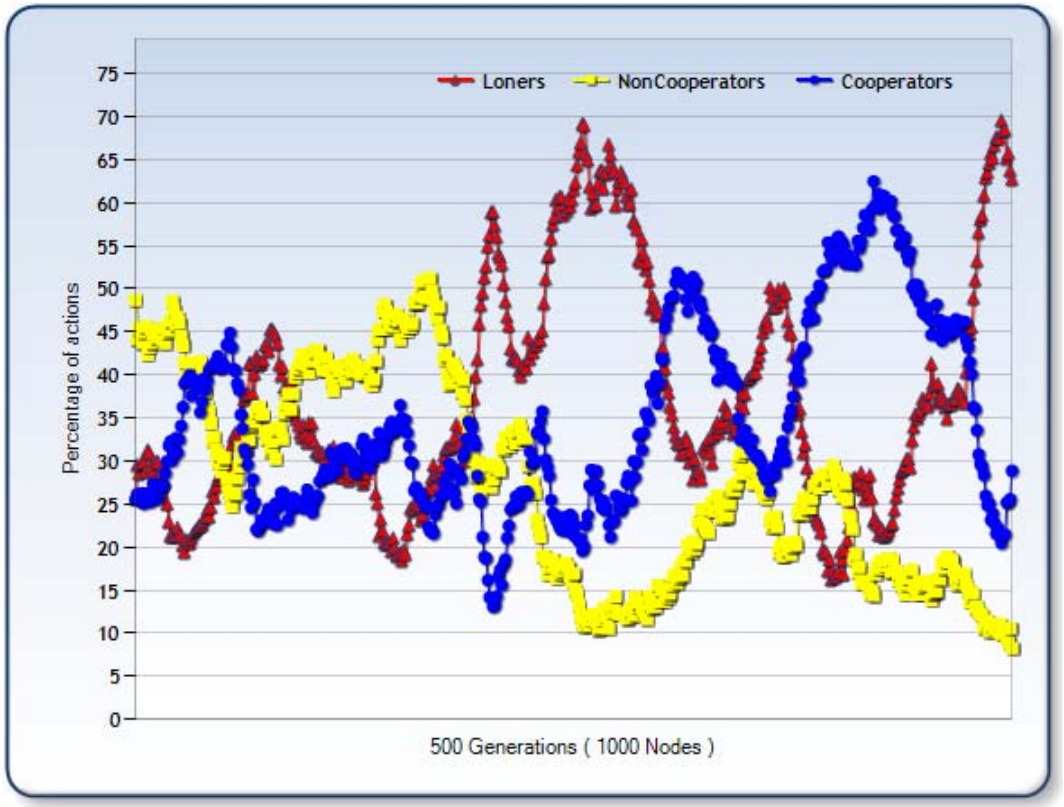


Imagen 5: Simulación de ejemplo

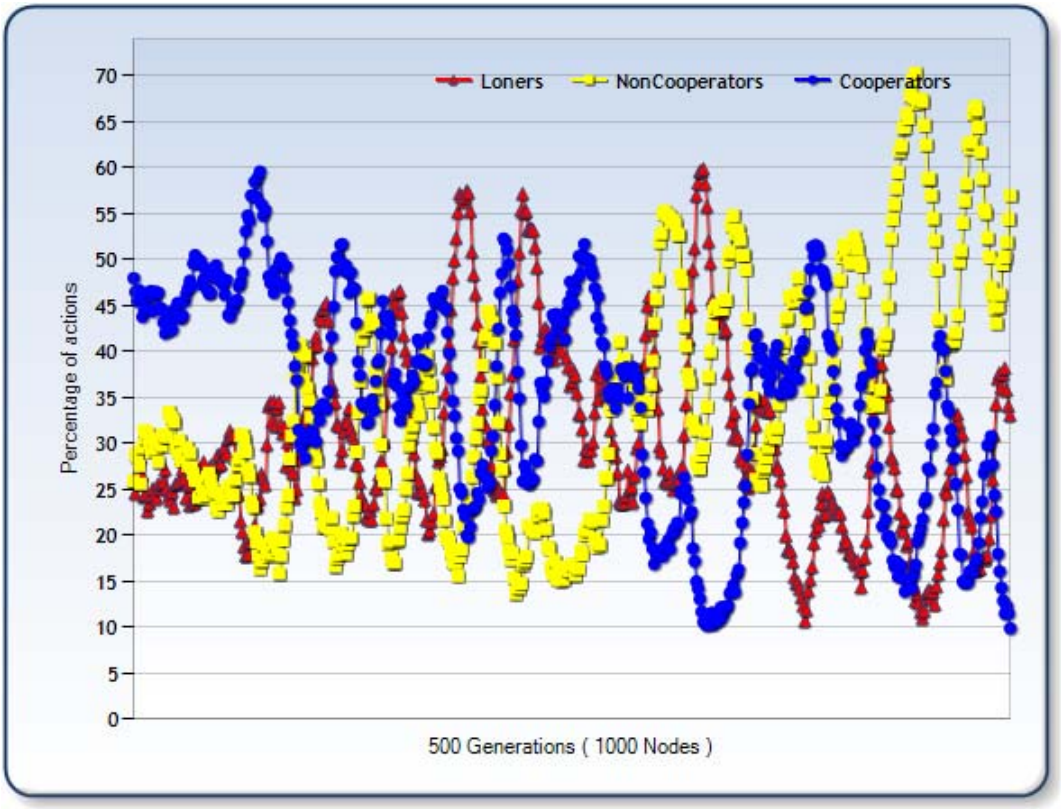


Imagen 6: Simulación de ejemplo

A continuación, se muestran cuatro ejecuciones que se genera en cada generación la matriz de relaciones de forma aleatoria, esto no debería de ser así y como se puede observar en los dibujos la red no se estabiliza en ningún caso, por lo tanto si queremos obtener conclusiones útiles, no debemos usar esta configuración. Y de estas cuatro ejecuciones podemos sacar la conclusión de que si las relaciones se generan aleatoriamente en cada generación, no se consigue la estabilidad y la red terminaría desapareciendo.

Esta ejecución (**Imagen 7**) parte de una población estable en la que las relaciones se generan aleatoriamente en cada iteración.

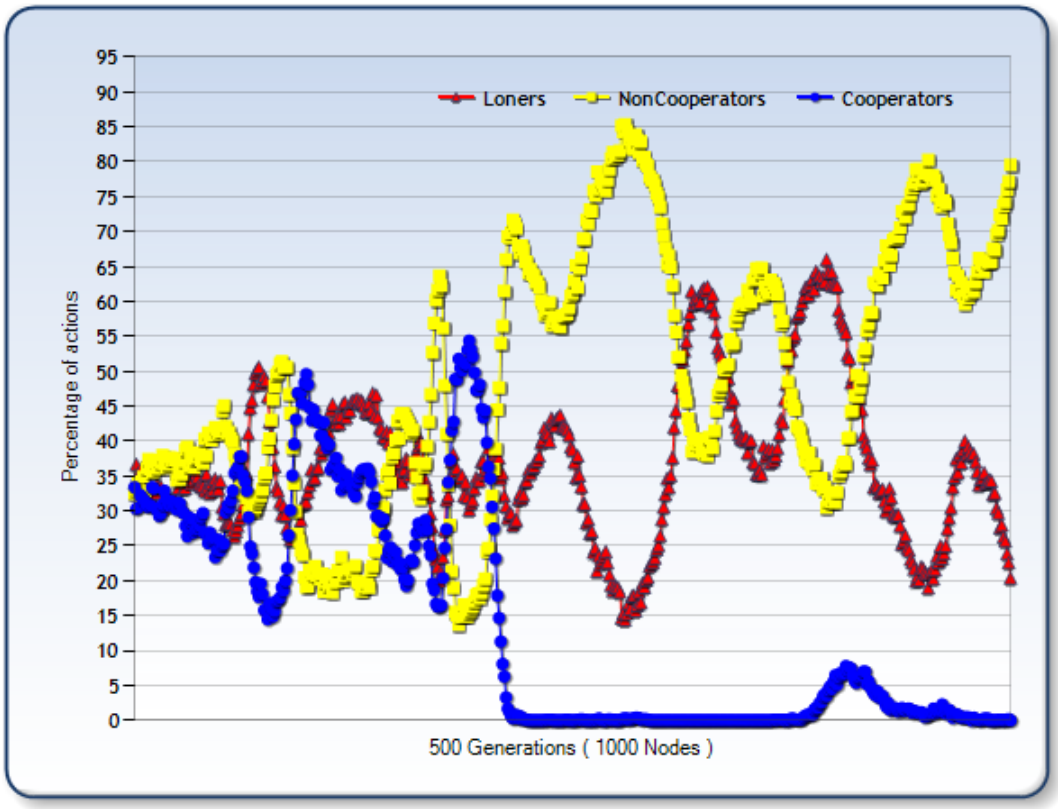


Imagen 7: Simulación de ejemplo

Las simulaciones siguientes parten de una población que la mayoría corresponde a una estrategia, en el primer caso (**Imagen 8**): 50% loners, 25% de no-cooperativos, y 25% de cooperativos, la siguiente (**Imagen 9**): 25% loners, 50% de no-cooperativos y 25% de cooperativos, y la última configuración (**Imagen 10**): 25% loners, 25% de no-cooperativos, y 50% de cooperativos.

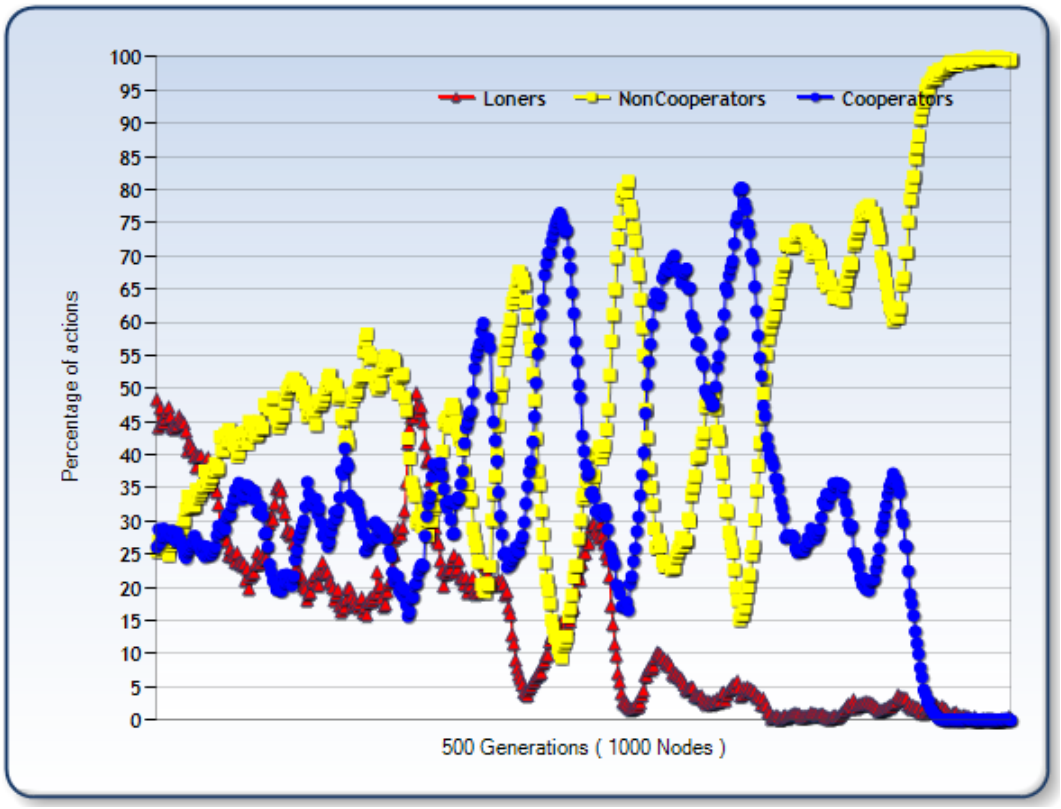


Imagen 8: Simulación de ejemplo

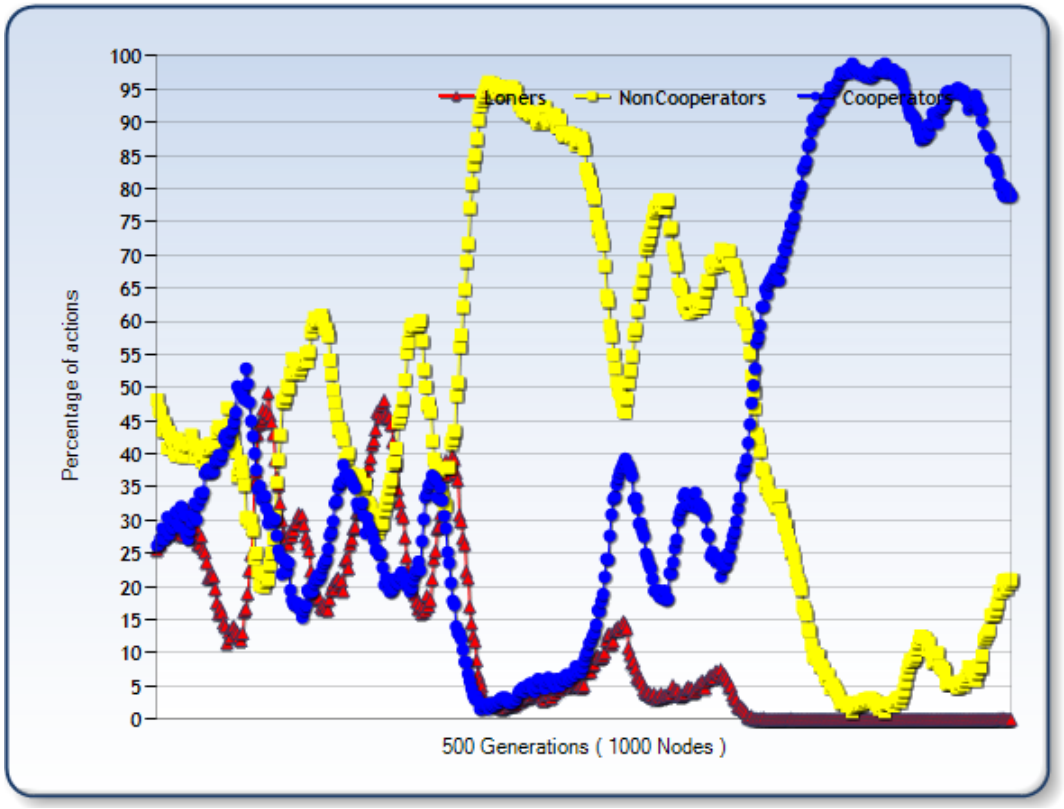


Imagen 9: Simulación de ejemplo

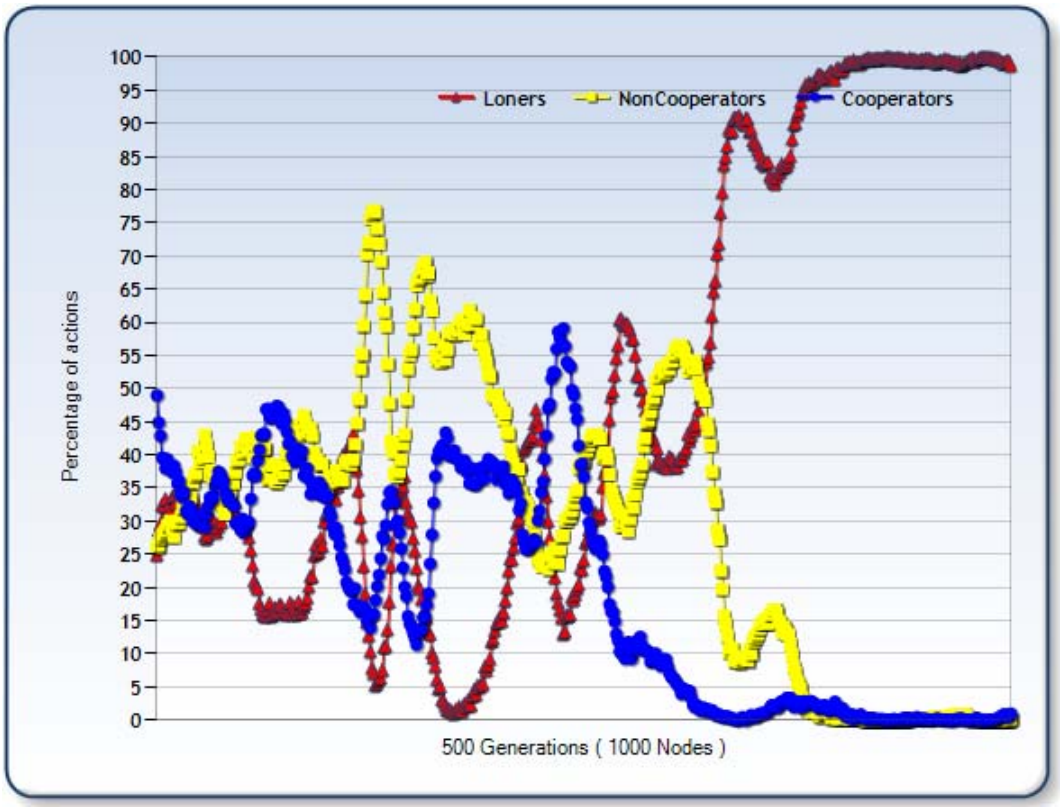


Imagen 10: Simulación de ejemplo

El grafico siguiente (**Imagen 11**) tiene la misma configuración que las anteriores simulaciones con población estable (al menos un 33% para cada estrategia) solo que ahora se ha reducido a la mitad el porcentaje de mutación de las relaciones (al 10%) y se ha duplicado el porcentaje de mutación de las acciones (al 1%). Como se puede observar la población es estable y alternando estrategias. La conclusión de este grafico es que si reducimos las modificaciones en la matriz de relaciones de cada ronda y aumentamos el porcentaje de mutación de acciones, se consigue una población bastante estable (las más estable por ahora), debido a que toda la población de cada estrategia durante toda la ejecución se mantiene en el intervalo de 20-50 %.

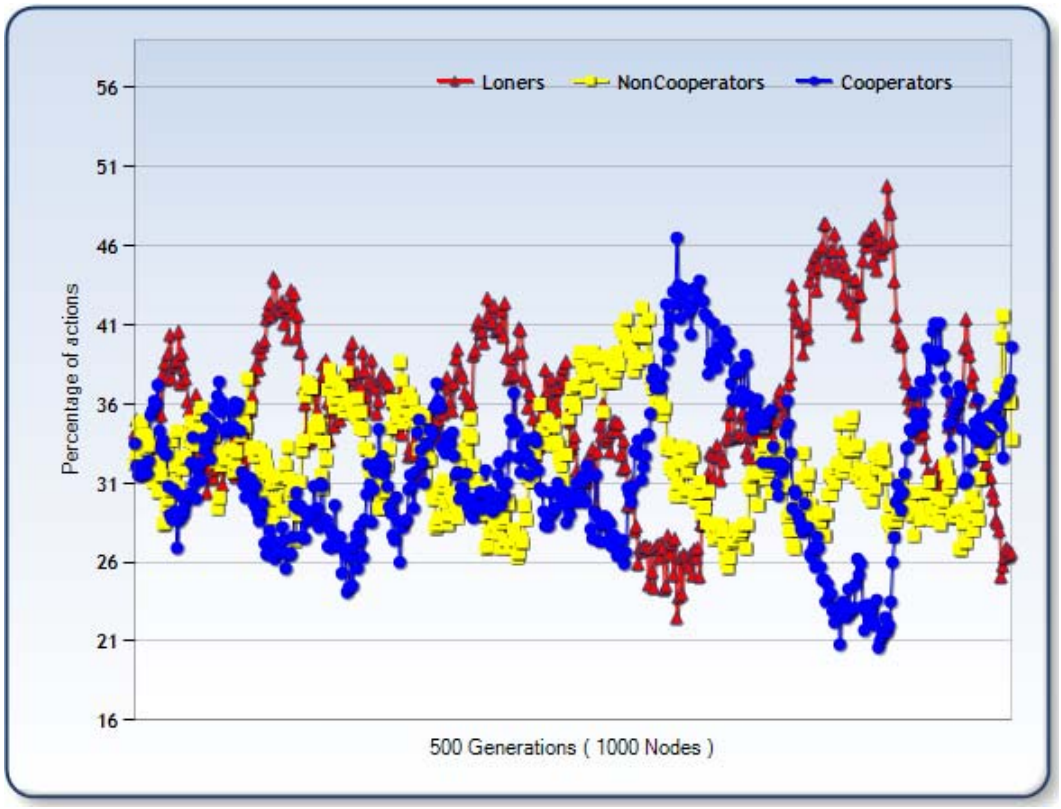


Imagen 11: Simulación de ejemplo

A continuación se muestra la ejecución (**Imagen 12**) más estable que se ha obtenido, se ha conseguido aumentando el porcentaje de mutación de acciones (se ha asignado el valor 5%, parece un valor bajo pero es muy alto para la mutación de acciones), partiendo de una población estable y el porcentaje de mutación de relaciones fijado en 10%, como se puede observar todas las estrategias están prácticamente entre el 40% y 25% de la población total, y cada vez converge mas a 33%. Por lo que se deduce que pocas modificaciones en la matriz de relaciones y alta mutación de acciones, se consigue la población más estable para las redes P2P descentralizadas.

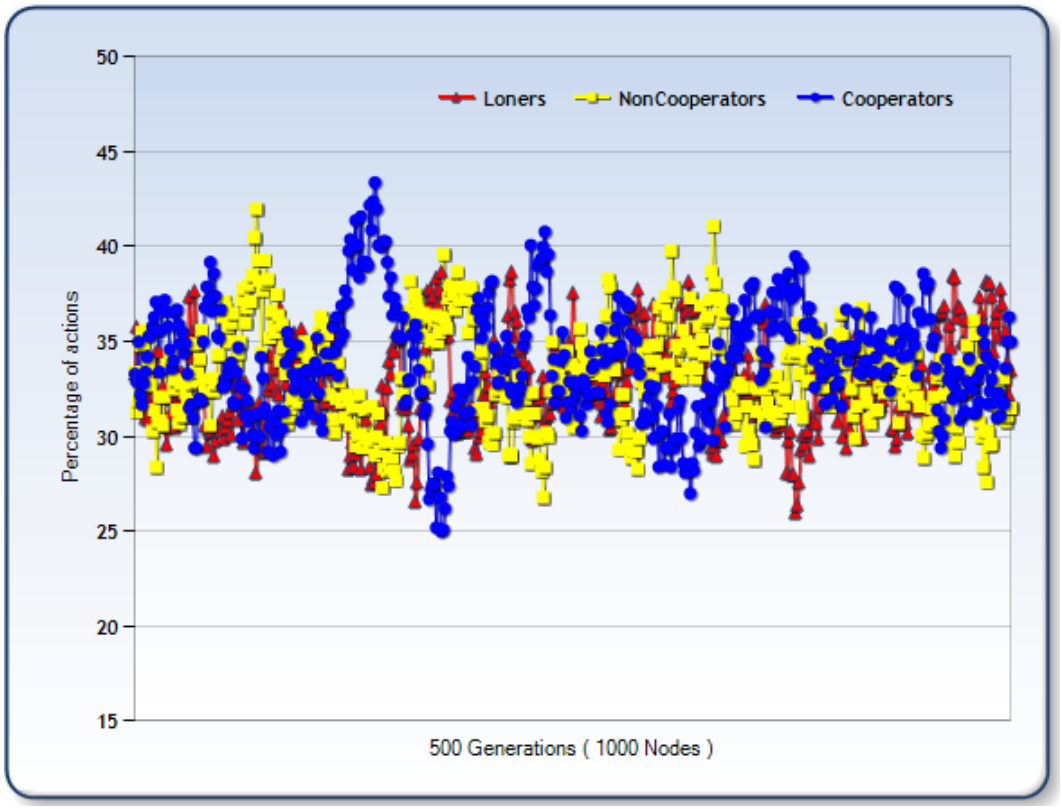


Imagen 12: Simulación de ejemplo

La Imagen 13 tiene la misma configuración que la anterior simulación, lo único que se ha modificado es la población inicial (10% *loners*, 80% no-cooperativos, 10% cooperativos) para intentar demostrar si se estabiliza la comunidad partiendo de una estrategia mayoritaria. Como se puede observar es la configuración más estable, incluso partiendo con una estrategia casi invasora. Como en el caso anterior las estrategias prácticamente se mantienen entre el intervalo 25-40 %. Como conclusión, se obtiene que es la configuración más estable para la comunidad P2P descentralizada.

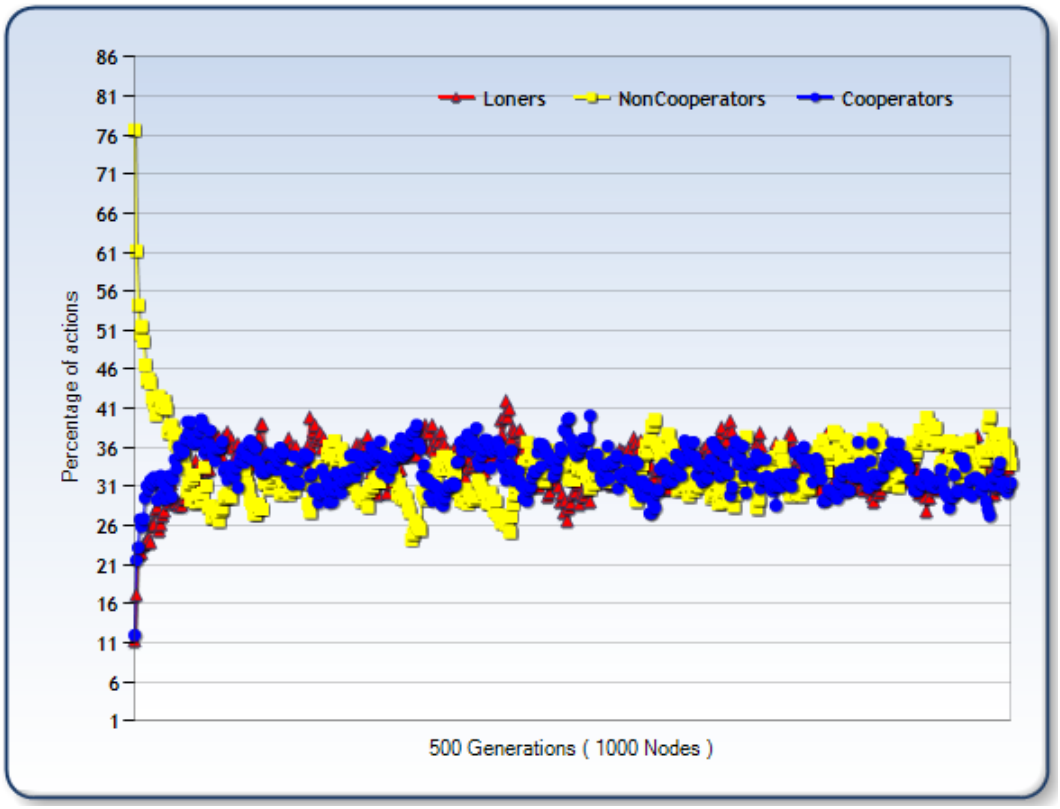


Imagen 13: Simulación de ejemplo

En la próxima figura (**Imagen 14**) se ha restablecido el porcentaje de mutación de acciones a su valor normal en un algoritmo genético, que es 0,5%. Pero se ha dejado el porcentaje de mutación de relaciones en 10%. Como se puede ver en el grafico la población es estable y las estrategias se alternan de forma rápida, siendo el incremento y decremento de la estrategia de los mayores hasta ahora, esto significa que un bajo índice de modificación en las relaciones la población se mantiene estable, todo lo contrario a las simulaciones en las que se generaban aleatoriamente las relaciones. Por lo tanto, si queremos conseguir simulaciones estables deberemos mantener este parámetro en valores bajos, que es como se da en las redes P2P descentralizadas.

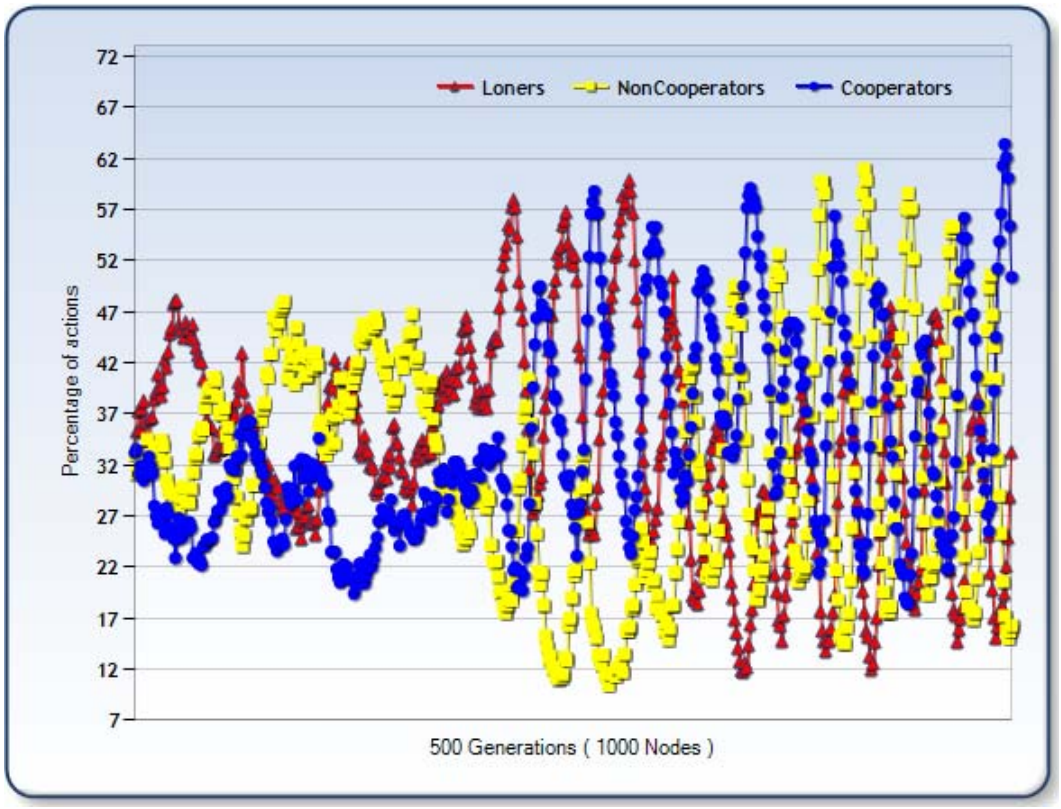


Imagen 14: Simulación de ejemplo

A continuación se muestran la tabla resumen para las configuraciones analizadas hasta el momento, para ver claramente las configuraciones que provocan una evolución estable.

Parámetros \ Imagen	Imagen 3	Imagen 4	Imagen 5	Imagen 6	Imagen 7	Imagen 8	Imagen 9	Imagen 10	Imagen 11	Imagen 12	Imagen 13	Imagen 14
Nº *Loners*	34	50	25	25	34	50	25	25	34	34	10	34
Nº No-Cooperativos	33	25	50	25	33	25	50	25	33	33	80	33
Nº Cooperativos	33	25	25	50	33	25	25	50	33	33	10	33
% Mutación relaciones	20	20	20	20	100	100	100	100	10	10	10	10
% Mutaciones acciones	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	1	5	5	0,5
Estabilidad	Si	Si	Si	Si	No	No	No	No	Si	Si	Si	Si

Tabla 4: Resumen estabilidad de las simulaciones

Como se puede observar (Tabla 4) las configuraciones que se han generado

aleatoriamente las relaciones entre los nodos en cada iteración son las únicas que no consiguen una población equilibrada.

Hasta el momento se ha analizado los parámetros que indican la población inicial, la mutación de relaciones y de acciones en cada iteración, es decir, las características de la población y su evolución.

Ahora estudiaremos como varia la población si modificamos los puntos de cruce del proceso de cruce del AG. Para ello probaremos en una población estable. De esta forma se verá si el punto de cruce influye en la evolución de la población.

La primera grafica (Imagen 15) se ha configurado de la siguiente manera, población estable (*loners* 34%, no-cooperativos 33% y cooperativos 33%), 500 generaciones y 20 rondas por nodo, el porcentaje de mutación de las relaciones de los nodos se ha marcado en 20%, el porcentaje de mutación de acciones está fijado en 0,5 %. Los puntos de cruce dependen de las categorías de los nodos a cruzar, para la primera simulación se ha marcado todos los porcentajes en 50 %. La grafica es la siguiente:

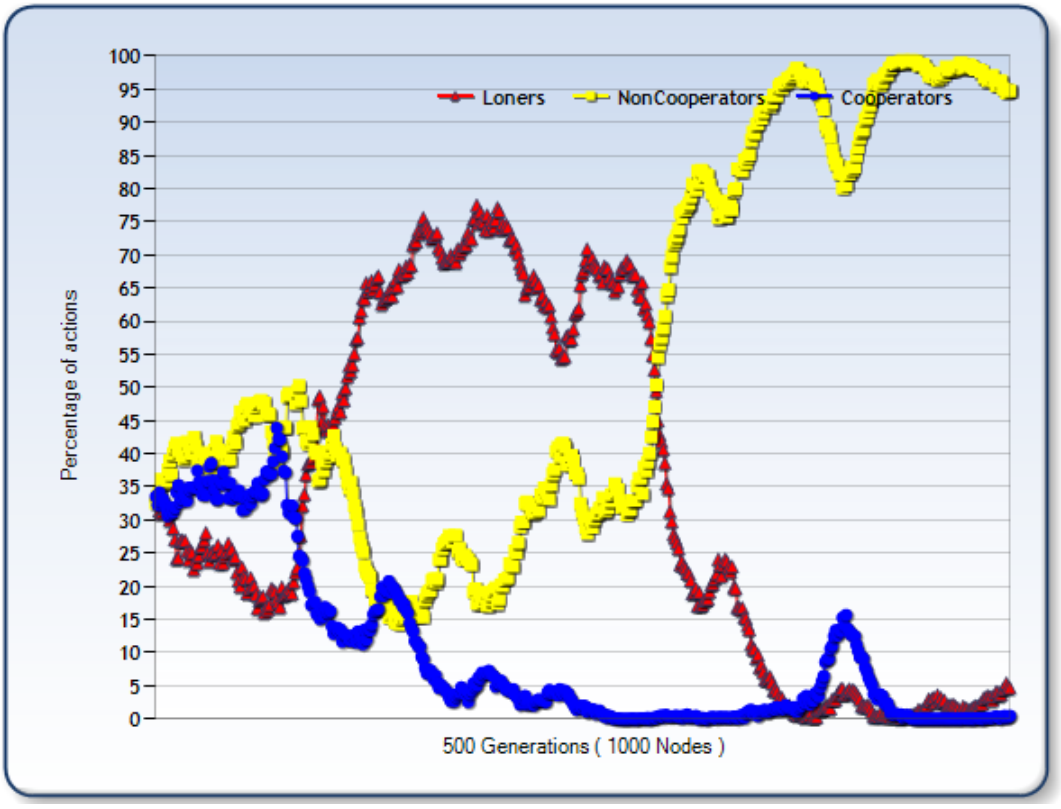


Imagen 15: Simulación de ejemplo

Para las siguientes graficas sean dejado todos los parámetros iguales menos los puntos de cruce, que es lo que estamos analizando, estos puntos de cruce se ha definido de la siguiente manera para la grafica siguiente (Imagen 16):

- Muy fuerte con Fuerte: 30 %.
- Muy fuerte con Débil: 35%.
- Muy fuerte con Muy débil: 40%.
- Fuerte con Débil: 30%.
- Fuerte con Muy débil: 35%.
- Débil con Muy débil: 30%.

El resultado obtenido es el siguiente:

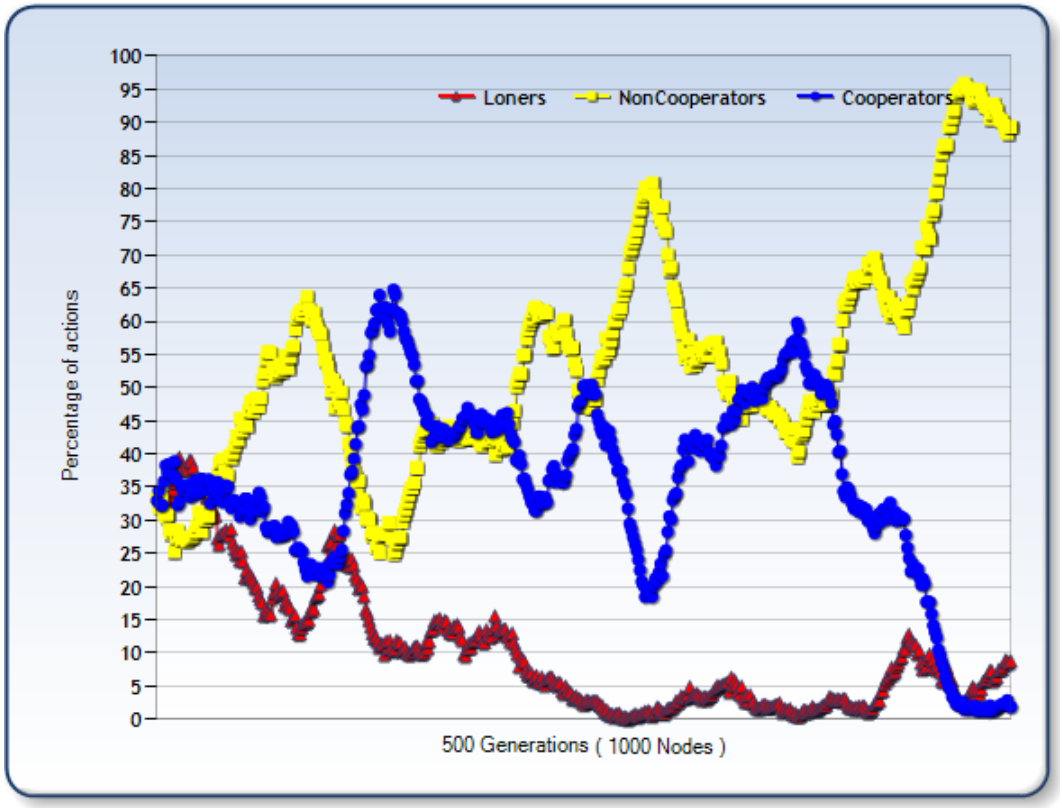


Imagen 16: Simulación de ejemplo

A continuación se describe los puntos de cruce de la siguiente figura (Imagen 17):

- Muy fuerte con Fuerte: 100 %.
- Muy fuerte con Débil: 100%.
- Muy fuerte con Muy débil: 100%.
- Fuerte con Débil: 100%.
- Fuerte con Muy débil: 100%.
- Débil con Muy débil: 100%.

Y la salida del simulador ha sido la siguiente:

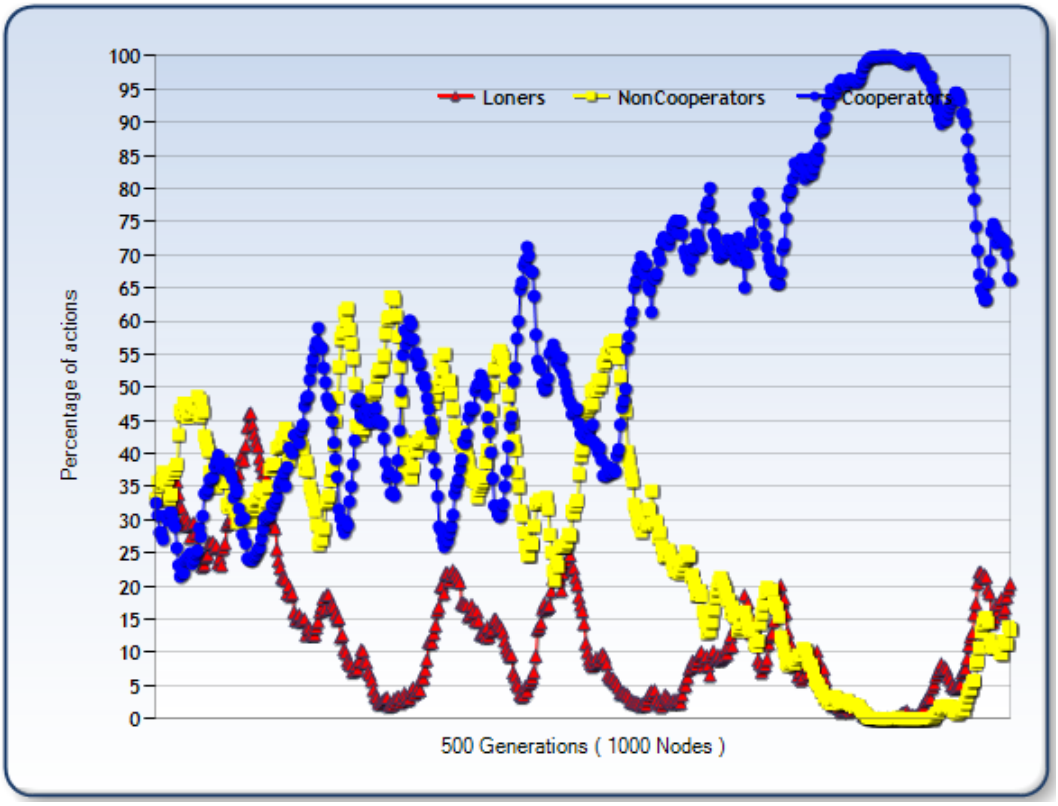


Imagen 17: Simulación de ejemplo

A continuación se describe los puntos de cruce de la siguiente figura (Imagen 18):

- Muy fuerte con Fuerte: 0 %.
- Muy fuerte con Débil: 0%.
- Muy fuerte con Muy débil: 0%.
- Fuerte con Débil: 0%.
- Fuerte con Muy débil: 0%.

- Débil con Muy débil: 0%.

El grafico es el siguiente:

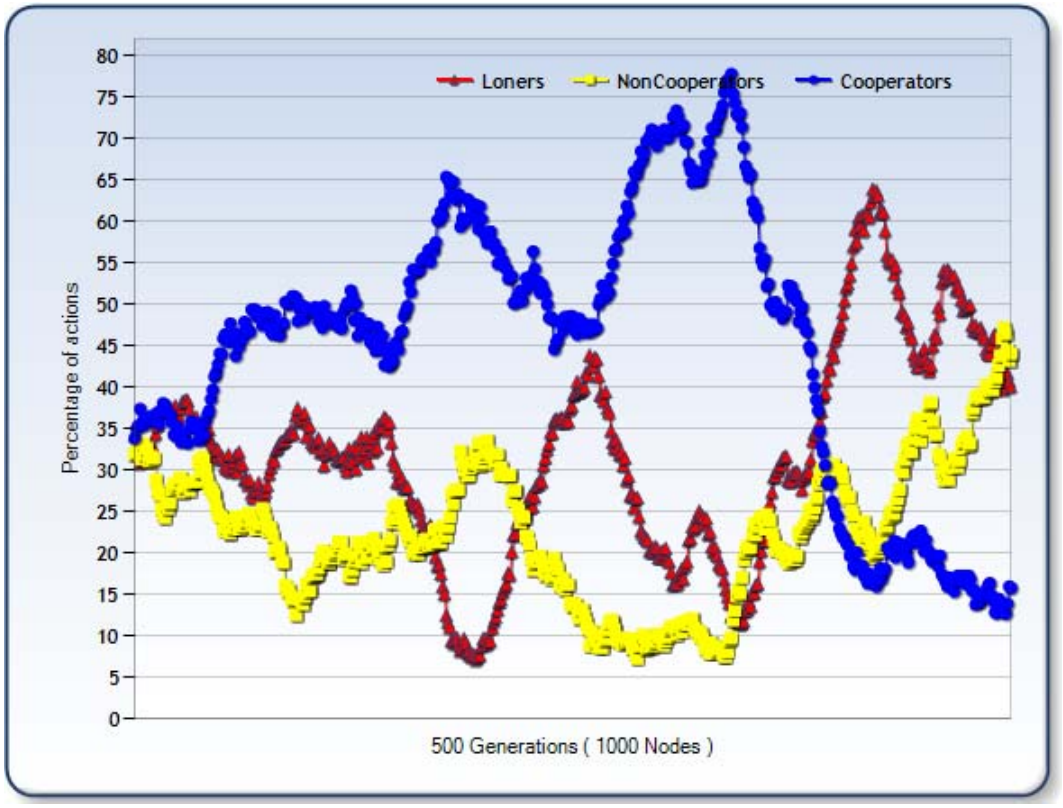


Imagen 18: Simulación de ejemplo

Como conclusión del análisis de los puntos de cruce, se puede que es importante elegir los puntos de cruce, ya que si no la simulación no converge al objetivo buscado.

Esto se debe a que el punto de cruce se usa para generar la nueva generación, entonces debemos elegir un punto de cruce optimo para que la siguiente generación sea mejor que la anterior. Esto se consigue evaluando los nodos, y dependiendo de su categoría se seleccionan más o menos genes. Se seleccionan mas genes cuando el individuo esta mejor capacitado (tiene mayor *fitness*) así mas genes óptimos pasan a la siguiente población, proporcionando a esta población de mayor capacidad que la anterior, y de esta forma nos acercamos más al objetivo deseado en cada generación.

CAPÍTULO 6. PRESUPUESTO

En este capítulo se presenta justificado el presupuesto de la realización de este proyecto fin de carrera.

Para poder elaborar el presupuesto, primero se han identificado cada una de las fases en las que se divide el proyecto, de forma que se puedan estudiar y analizar por separado los recursos utilizados en cada una de dichas fases.

Se va a considerar el modelo clásico de ciclo de vida para el proyecto en cuestión. Este modelo distingue las siguientes fases:

- Análisis.
- Diseño.
- Implementación.
- Pruebas.
- Documentación.

Para cada fase del proyecto se considerarán los recursos humanos utilizados, así como también los recursos materiales empleados (entendiendo como tales, equipos informáticos, material de oficina, etc.).

La siguiente Tabla 5 muestra el coste total estimado que ha supuesto el proyecto respecto a los recursos humanos. Se ha considerado para cada semana, una media de 4 horas al día durante 5 días a la semana, por lo que se contabilizará cada semana como 20 horas de trabajo. Además, se ha estimado para cada fase, un precio/hora correspondiente al perfil mínimo necesario para la elaboración de las tareas.

Fase	Precio/Hora (€)	Semanas	Horas	Total/Fase (€)
Análisis	60	2	40	2.400
Diseño	60	2	40	2.400
Implementación	60	5	100	6.000
Pruebas	60	3	60	3.600
Documentación	60	2	40	2.400
Análisis de Resultados	60	2	40	2.400
Total	**19.200 €**			

Tabla 5: Recursos humanos

Entre los recursos materiales (Tabla 6) se incluye tanto el material fungible y equipos informáticos, como también los servicios requeridos durante la elaboración del mismo.

Debido a que resulta difícil asignar los recursos a las diferentes fases del proyecto, se considerará que éstos se han usado de manera uniforme durante todo el ciclo de vida del mismo. La tabla siguiente los muestra de manera aproximada:

Recurso	Cantidad	Precio	Total
Ordenador portátil	1	950 €	950 €
Conexión a Internet (cantidad considerada en meses)	5	30 €	150 €
Impresión y encuadernación con tapas duras del proyecto	3	48 €	144 €
Impresión y encuadernación en espiral del proyecto	3	9 €	27 €
Microsoft Office 2007			120 €
Material de oficina			15 €
Total	**1.286 €**		

Tabla 6: Recursos materiales

Como se puede observar no el software de desarrollo no se ha incluido en la tabla, porque todo el proyecto ha sido realizado con software gratuito que proporciona Microsoft para jóvenes estudiantes universitarios, dándonos la facilidad que a nivel de empresa que existe en la actualidad. Este software es Visual Studio 2008 y el componente MS Chart Control.

6.1. COSTE TOTAL DEL PROYECTO

El coste total se obtiene sumando el coste total de los recursos humanos y el de los recursos materiales. Por tanto, el coste total del proyecto es de **20.486 €.**

El tiempo empleado para la realización del proyecto ha sido de cuatro meses, por lo que los costes por mes han sido de 20.486/4 = **5.121,5 €**.

CAPÍTULO 7. CONCLUSIONES Y TRABAJO FUTURO

En este proyecto, se ha estudiado el dilema RPS de iguales como un mecanismo para permitir la cooperación a surgir naturalmente, sin incentivos externos. Esto es particularmente adecuado para redes P2P puras para compartir ficheros donde pueden ser implementados mecanismos de cooperación sin ninguna sofisticación. La introducción de nodos *loner* es suficiente para llegar a un equilibrio estable entre nodos anónimos egoístas. La esperanza de este proyecto, directa o indirectamente, es inspirar nuevas direcciones en las áreas de investigación de redes P2P.

Es muy prometedor el enfoque aplicado y los resultados obtenidos, ya que se ha demostrado que no es necesaria la aplicación de incentivos a los nodos en una red P2P para la subsistencia de ésta. De esta manera se podrá reducir los costes en este tipo de redes porque simplifica la implementación de dichas redes, además de reducir el número de mensajes transmitidos por la red (Internet).

Por otra parte, al no haber incentivos, se podrá interactuar con la red P2P descentralizada totalmente en el anonimato, proporcionando seguridad a la red.

Como trabajo futuro, en un primer lugar se deberá implantar el modelo en un entorno real para ver si los sucesos analizados en este proyecto corresponden totalmente con la realidad. En el caso de que el entorno real no se comporte como el esperado se deberá analizar las diferencias entre el modelo y entorno real, y modificar el modelo hasta conseguir el objetivo esperado.

Otra línea de trabajo abierta consiste en analizar el rendimiento y el consumo de recursos respecto a una red P2P descentralizada que use un sistema de reputación o de incentivos, así comprobaremos cuánto se ha reducido la complejidad de un entorno y otro, sin perder un elevado nivel de cooperación dentro de la comunidad.

Y como última línea de trabajo futuro se pueden analizar los genes de los nodos para intentar predecir su evolución e identificar su comportamiento a la vez que se obtienen patrones de comportamiento para los nodos, es decir, partiendo de un nodo *loner* en que momento va a cambiar a no-cooperativo o a cooperativo, ya que de esta forma podremos predecir la evolución de la red P2P descentralizada.

Bibliografía

[1] Jochem van Vroonhoven. *Peer-to-peer security*. Departament of Computer Science. Faculty of Electrical Eengineering, Mathematics and Computer Science. University of Twente.

[2] Robert Axelrod. *The evolution of strategies in the iterated Prisoner's Dilemma*.

[3] Ariel Rubinstein, Asher Wolinsky. *Rationalizable conjectural equilibrium between nash and rationalizable (May 1991).*

[4] David E. Goldberg, Kalyanmoy Deb., James H. Clark. "*Genetic algorithms, noise and sizing of populations (1991)*".

[5] W. Poundstone, *Prisoner's Dilemma*. Doubleday, New York, 1992.

[6] D. Fogel, "*Evolving behaviors in the iterated prisoner's dilemma*" *Evolutionary Computation*, vol. 1, pp. 77–97, Spring 1993.

[7] R. Gupta and A. Somani, "*Game theory as a tool to strategize as well as predict nodes behavior in peer-to-peer networks*" in Proc. of the 11th Int. Conf. on Parallel and Distributed, 2005, pp. 244–249, systems, Fukuoka, Japan, IEEE Computer Society (2005).

[8] E. Palomar, A. Alcaide, J. Tapiador, and J. Hern´andez-Castro, "*Bayesian analysis of secure P2P sharing protocols*" in Proc. of OTM to Meaningful Internet Systems 2007: IS (2), 2007, pp. 1701–1717.

[9] H. Feng, S. Zhang, C. Liu, J. Yan, and M. Zhang, "*P2P incentive model on evolutionary game theory*" in Proc. of the 4th International Conference on Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08., 2008, pp. 1–4.

[10] D. Hales and S. Arteconi, "*Slacer: a self-organizing protocol for coordination in peer-to-peer networks*" Intelligent Systems, IEEE, vol. 21, no. 2, pp. 29–35, 2006.

[11] M. Feldman, K. Lai, I. Stoica, and J. Chuang, "*Robust incentive techniques for peer-to-peer networks*" in Proc. of the 5th ACM conference on Electronic commerce, 2004, pp. 102–111.

[12] C. Hauert, S. D. Monte, J. Hofbauer, and K. Sigmund, "*Volunteering as red queen mechanism for cooperation in public goods game*" Science, vol. 296, pp. 1129–1132, 2002.

[13] R. Ma, S. Lee, J. Lui, and D. Yau, "*A game theoretic approach to provide incentive and service differentiation in p2p networks*" in Proc. of the joint Int. Conf. on Measurement and modeling of computer systems. New York, USA: ACM Press, June 2004, pp. 189–198.

[14] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "*A survey and comparison of peer-to-peer overlay network schemes*" IEEE Communications Surveys & Tutorials, vol. 7, pp. 72–93, 2005.

[15] D. Goldberg, "*Genetic Algorithms in Search, optimization and Machine Learning*". Addison-Wesley, 1989.

[16] R. Hinterding and Z. Michalewicz, "*Your brains and my beauty: parent matching for constrained optimisation*" in Proc. of the IEEE World Congress on Computational Intelligence, Int. Conf. on Evolutionary Computation. Anchorage, AK, USA: IEEE Computer Society, May 1998, pp. 810–815.

[17] P. Hancock, "*Your brains and my beauty: parent matching for constrained optimisation*" in Proc. of the AISB Workshop on Evolutionary Computing. Leeds, UK: Springer, April 1994, pp. 80–94