UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

TESIS PARA OBTENER EL TÍTULO DE LICENCIATURA:
ING. EN TELECOMUNICACIONES.

'ALGORTIMO GENÉTICO PARA MEJORAR EL RENDIMIENTO PROMEDIO DE REDES IEEE 802.11g'

Presenta:
Carlos Alberto Marín Ceballos
Director de tesis:
Dr. Javier Gómez Castellanos

Octubre – 2013 **Ciudad Universitaria – México, D.F.**

‘ALGORTIMO GENÉTICO PARA MEJORAR EL RENDIMIENTO PROMEDIO DE REDES IEEE 802.11g’

Presenta:

Carlos Alberto Marín Ceballos

Director de tesis:

Dr. Javier Gómez Castellanos

AGRADECIMIENTOS.

A mi familia por nunca dejar de creer en mí y siempre apoyarme en cada paso.
Madre, cada día de mi vida me esforzaré por hacerte sentir orgullosa.
A mi Abuela cuyo amor me moldeo en el hombre que soy hoy.
Mis amigos y amigas. Me quedo con una pequeña parte de todos ustedes.
A la Facultad de Ingeniería que me ha llenado de experiencias inolvidables.
Al Dr. Javier Gómez y al Dr. Bhaskar Krishnamachari por guiarme en mi trabajo de investigación.
Esta tesis se realizó en parte con el apoyo de los proyectos: CONACyT 105117, PAPIIT IN106609 y PAPIIT 114813.

INDICE.

ABSTRACT.

En el presente trabajo de tesis se planteó, se desarrolló y se simuló una nueva aproximación, en forma de algoritmo genético, que es capaz de encontrar configuraciones más eficientes para construir la ventana de contención del mecanismo 'binary exponential backoff' en redes IEEE 802.11g. Se utilizaron los principios de la programación genética para diseñar una aplicación en Java capaz de simular redes IEEE 802.11g con N estaciones contendientes, misma que logró mejorar el rendimiento promedio en 'kbps' al utilizar los valores obtenidos con el algoritmo genético propuesto. La búsqueda de valores óptimos se hizo emulando el principio de selección natural. Los resultados obtenidos de distintos experimentos fueron comparados en gráficas de dos dimensiones y las conclusiones generadas al final del documento demuestran el éxito obtenido con la aplicación diseñada, se lograron mejoras en el rendimiento promedio de hasta el 60% con 160 terminales contendientes para redes simuladas IEEE 802.11g.

I - INTRODUCCIÓN.

1.1 Presentación.

1.1.1 Objetivos.

El presente trabajo de investigación tiene como objetivos los siguientes puntos:

I. Proveer un marco teórico, incluyendo el estado del arte, en torno al estándar IEEE 802.11g y al uso de algoritmos genéticos como herramientas para tratar problemas de optimización.
II. Plantear y justificar un nuevo modelo para mejorar el rendimiento promedio de redes que operan bajo el estándar IEEE 802.11g.
III. Diseñar el algoritmo genético y el pseudocódigo que logren simular una mejora notable para redes simuladas IEEE 802.11g.
IV. Implementación del algoritmo genético propuesto en una aplicación de lenguaje Java.
V. Proporcionar, de forma ordenada, los resultados obtenidos a partir de simulaciones bajo múltiples escenarios propuestos.
VI. Entregar un cuadro comparativo que corrobore el éxito de la aplicación diseñada.
VII. Consolidar las conclusiones del trabajo presentado en forma de un borrador para evaluarse como Articulo de Investigación de alto nivel.

1.1.2 Alcance.

El trabajo de investigación presentado a continuación abarca el diseño, implementación y simulación del algoritmo genético propuesto.

1.1.3 Estructura.

El desarrollo del presente trabajo se puede considerar en tres etapas. Una primera etapa para abarcar el marco teórico que nos llevará a la propuesta del algoritmo genético, una segunda etapa que consistirá en la implementación del algoritmo genético propuesto en una aplicación en lenguaje Java, y finalmente, el análisis y la comparación de resultados obtenidos para concluir con las lecciones aprendidas y el trabajo futuro recomendado al lector.

1.1.4 Contribuciones.

Las contribuciones generadas por el presente trabajo de investigación son:

I. Un sólido marco teórico para futuras generaciones que deseen desarrollar soluciones sobre redes IEEE 802.11g o que deseen implementar algoritmos genéticos como herramientas para lidiar con problemas de optimización.
II. Conclusiones contundentes acerca de la posibilidad de mejorar el estándar IEEE 802.11g con configuraciones dinámicas que se adaptan al entorno.
III. Las bases para trabajo futuro propuesto. Diseñar un algoritmo genético que funcione de igual manera para estándares de la familia 802.11b/n.
IV. Un borrador para presentarse como Artículo de Investigación de alto nivel. **ANEXO I.**

1.2 Prefacio.

Las redes inalámbricas se han convertido en una de las principales herramientas en el mundo de las telecomunicaciones. Los usuarios están viviendo la rápida evolución de las comunicaciones globales, y ahora el estar conectado se ha vuelto un bien indispensable para cualquier área, sea personal, académica o de negocios.

Una de las formas preferidas por los usuarios para estar en línea es mediante la conexión sin cables. El acceso inalámbrico participa hoy en día en muchas formas de comunicaciones a distancia y su crecimiento sigue perfilando positivamente. El siglo pasado la humanidad fue testigo de las primeras transmisiones de radio transatlánticas, y hoy en día ya podemos encontrar terminales móviles donde realizar videollamadas de alta calidad con sólo un par de comandos.

En un mundo globalizado, donde estar bien comunicado es esencial para el crecimiento de todos los sectores, las redes inalámbricas deben ser lo más eficientes posibles. El usuario no desea tener que esperar más del tiempo necesario para descargar un archivo o para revisar su correo electrónico. Para actividades de entretenimiento, inclusive, el usuario agota su paciencia si intenta ver un video en línea y éste interrumpe su reproducción a causa de tiempos de espera elevados.

Pero entonces, ¿cómo podemos maximizar el uso eficiente de las redes inalámbricas? Existen muchas formas de hacerlo. Algunas aproximaciones son propuestas para aumentar la capacidad del canal con soluciones como: agrandar el ancho de banda o subir la potencia de transmisión, otras ideas tienen que ver con técnicas de modulación más eficaces para el uso del espectro radioeléctrico. Sin embargo, dichas propuestas tienen sus limitaciones en la práctica.

Dicho esto, podemos mejorar la eficiencia de las redes inalámbricas si optimizamos su funcionamiento, pero ¿acaso no una forma de mejorar el desempeño de cualquier sistema es logrando que falle lo menos posible? así no perdemos recursos para la corrección de errores.

Las redes inalámbricas presentan, en gran medida, bajo rendimiento por problemas del canal. El espacio libre es un medio difícil de controlar y presenta cambios constantes en sus propiedades por factores de distintas índoles que no podemos predecir de forma totalmente acertada. El medio inalámbrico presentará, en todo momento, interferencias de otras señales y ruido electromagnético que degradarán la calidad de nuestro mensaje a transmitir o recibir. De modo que, en cualquier sistema de comunicaciones inalámbricas, siempre tendremos una eficiencia condicionada directamente por la pérdida de datos en el medio, entre otras situaciones. Dichas pérdidas pueden deberse a distintos fenómenos como atenuación, distorsión o interferencia por señales no deseadas (por otros mensajes en el medio o por ruido electromagnético) [1].

Una vez comprendido esto, podemos afirmar que una forma segura para aumentar la eficiencia de una red inalámbrica es evitando la perdida de datos. Existen múltiples propuestas para lidiar con cada uno de los fenómenos descritos.

En las redes de datos inalámbricas, y más en específico, en las redes inalámbricas de área local (WLAN), hay múltiples causas que provocan la perdida de datos, además de las correspondientes a

características del canal. Existe mucha investigación acerca de las propiedades físicas del medio inalámbrico para optimizar las transmisiones y lidiar con fenómenos de atenuación, interferencia y distorsión. La mayoría sugieren nuevos métodos de modelar la capa física (PHY layer) con estrategias innovadoras de aprovechamiento del medio [2-4].

Sin embargo, éstas no son las únicas formas en que se puede perder nuestro mensaje antes de ser recibido correctamente por el destinatario. Si nos movemos una capa arriba del modelo OSI (LINK layer) para ver lo que concierne a la subcapa de control de acceso al medio (MAC por sus siglas en inglés) [5], veremos que nuestro mensaje se enfrentará a dificultades adicionales a los fenómenos en la capa física.

Cuando nuestro medio de transmisión es el espacio libre, se vuelve complicado en el receptor diferenciar señales que vienen de distintos puntos de transmisión, y que estos mensajes no se interfieran entre sí. En redes de datos inalámbricas es necesario un control u 'orden' en las transmisiones de todas las estaciones involucradas en la misma; de lo contrario todas las estaciones están sujetas a interferirse entre ellas mismas, y la pérdida de datos sería tan grande que la eficiencia de la red bajaría drásticamente.

Dicha organización y jerarquización de mensajes para todas las estaciones involucradas en una red de datos inalámbrica, donde se comparte el mismo canal, le concierne al protocolo de acceso al medio (MAC), en la capa de enlace. Ésta entidad la podemos encontrar tanto en redes alambradas como en redes inalámbricas. El control de acceso al medio se compone de reglas y mecanismos que tienen como objetivo hacer un uso eficiente del canal para tener un control sobre todas las transmisiones de la red, y evitar que estas se interfieran entre sí. Análogo a una mesa de debate, necesitamos un 'moderador' que les indique a los participantes cuándo pueden hablar o cuándo deben esperar. La relación de la subcapa MAC en las redes inalámbricas se ilustra en la Figura 1.1.

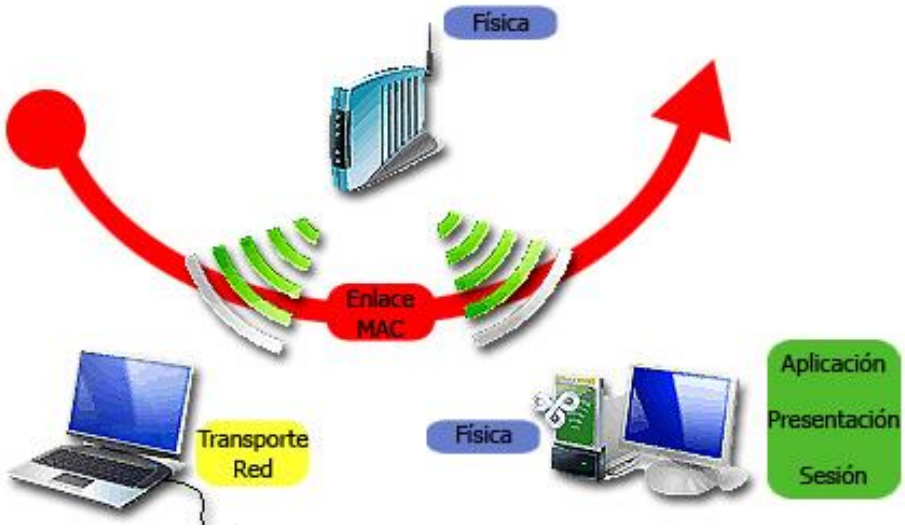


Figura 1.1 - Capa física en los módulos de RF de las estaciones inalámbricas y la capa de enlace (subcapa MAC) encargada del control de acceso al medio inalámbrico.

La asociación mundial del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE por sus siglas en inglés), ha dictaminado las funciones de las capas física y de enlace así como la sub-capa MAC en los artículos correspondientes al estándar 802.11g [6] para redes que deseen trabajar bajo este entorno. No está de sobra mencionar que éste es uno de los estándares para redes de datos inalámbricas de área local más populares a nivel mundial.

En las siguientes páginas se describirá el modo de operación de la subcapa MAC y su función en redes inalámbricas IEEE 802.11g en específico.

1.3 El estándar IEEE 802.11g.

La familia de protocolos para redes inalámbricas de área local (WLANs) de la IEEE son comúnmente llamadas las redes 802.11. Sin embargo, existen muchas versiones estandarizadas que difieren en los parámetros que definen el comportamiento de la red misma.

Por ejemplo, entre los más populares están los protocolos 802.11b, 802.11g y 802.11n. Cada uno de estos posee sus propias características de funcionamiento en la capa física (PHY); tipo de modulación, manejo del espectro, frecuencia de operación, locación de canales, etc., así como las características definidas de la capa de enlace. Por lo tanto tienen diferentes capacidades de red en cuanto a tasas de transmisión, eficiencia espectral y alcance en metros de RF. Una tabla comparativa de dichas tres versiones se muestra en la Tabla 1.1.

Versión	Frecuencia de operación	Máxima tasa de transmisión	Tipo de modulación	Alcance (exteriores)
802.11b	2.4 GHz	11 Mbps	DSSS/CCK	140 m.
802.11g	2.4 GHz	54 Mbps	OFDM/CCK	140 m.
802.11n	2.4 y 5 GHz	200-600 Mbps	OFDM/MIMO	250 m.

Tabla 1.1 - Cuadro comparativo[1] de las características generales para tres versiones de la familia de estándares IEEE 802.11.

En lo que compete a este trabajo, nos concentraremos en la versión IEEE 802.11g por tratarse de un protocolo popular y lo suficientemente maduro, con casi una década en el mercado, para poder hacer un análisis bien justificado de su desempeño y estudiar las múltiples proposiciones que ya existen para su mejora (Capítulo 2).

El estándar IEEE 802.11g está especificado con hasta cuatro tipos de configuraciones para la capa física (PHY) por su compatibilidad con redes tipo IEEE 802.11b[2] y para lograr un mejor desempeño con distintas técnicas de modulación y de manejo del espectro. Dichas especificaciones son:

[1] http://en.wikipedia.org/wiki/Template:802.11_network_standards

[2] Tras la liberación de las técnicas de modulación OFDM en la banda libre de 2.4GHz y 5GHz por la FCC, el grupo de trabajo de la IEEE para el estándar 802.11g decidió introducir una compatibilidad de retroceso con redes 802.11b al incorporar soporte para CCK. [7]

DSSS/CCK, OFDM, DSSS/PBCC y DSSS-OFDM [8]. Para nuestro trabajo utilizaremos las características del estándar IEEE 802.11g que trabajan con OFDM sencillo para la PHY en la banda de los 2.4GHz y tiene una velocidad teórica máxima de transmisión de 54 Mbps. Dicha especificación incluye la posibilidad virtual de detección en el canal (por medio de la NAV) y la fragmentación de paquetes MSDU.[3]

Es importante mencionar que existe una estrecha relación entre la capa física PHY y la subcapa inferior de la capa de enlace llamada MAC. El mecanismo MAC controlará la información proveniente de capas superiores y que las estaciones desean transmitir a un receptor. Sin embargo, al pasar este mensaje por el medio inalámbrico, es fundamental estar bien comunicado con la capa física que especificará los detalles de la transmisión al medio en forma de señales electromagnéticas con un esquema de modulación específico en un canal asignado en la distribución del espectro utilizable. Si los paquetes de control o de datos provenientes de la MAC no son correctamente recuperados e interpretados en la capa PHY, toda la transmisión fallará sin importar el buen desempeño de capas superiores.

En lo que concierne a la capa de enlace, más en específico, al papel de la subcapa de control de acceso al medio MAC del estándar IEEE 802.11g, se presente la siguiente sección.

1.3.1 Especificaciones MAC del estándar IEEE 802.11g.

El estándar IEEE 802.11g define las características de la subcapa MAC, donde se incluye, entre otras cosas, la necesidad de compatibilidad con los equipos de la red que operen con versiones 802.11g y 802.11b para ser capaces de recibir paquetes de control del tipo ACK, RTS y CTS en una tasa de transmisión base común.

Del mismo modo se especifican dos técnicas diferentes para la coordinación del canal pero que pueden trabajar en conjunto sin interferirse. Dichas técnicas son la DCF y la PCF. Estas funciones de coordinación regularán el comportamiento de las transmisiones en la red. En la primera, la Función de Coordinación Distribuida (DCF por sus siglas en inglés) se trabaja con un sistema que escucha el canal priori a la transmisión para proceder a una contienda continua y aleatoria por el derecho de transmitir. En la segunda, la Función de Coordinación Puntual (PCF por sus siglas en inglés), no se maneja una contienda por el canal, en su lugar existe una entidad inteligente que estará encargada de asignar los turnos correspondientes a cada estación que desee transmitir. En la Figura 1.2 se puede apreciar la jerarquización de estas dos funciones de coordinación.

[3] Características más detalladas, donde se incluyen el procedimiento físico y virtual de detección, así como las condiciones para una fragmentación de MSDU se pueden encontrar en la documentación oficial del grupo IEEE. [9]

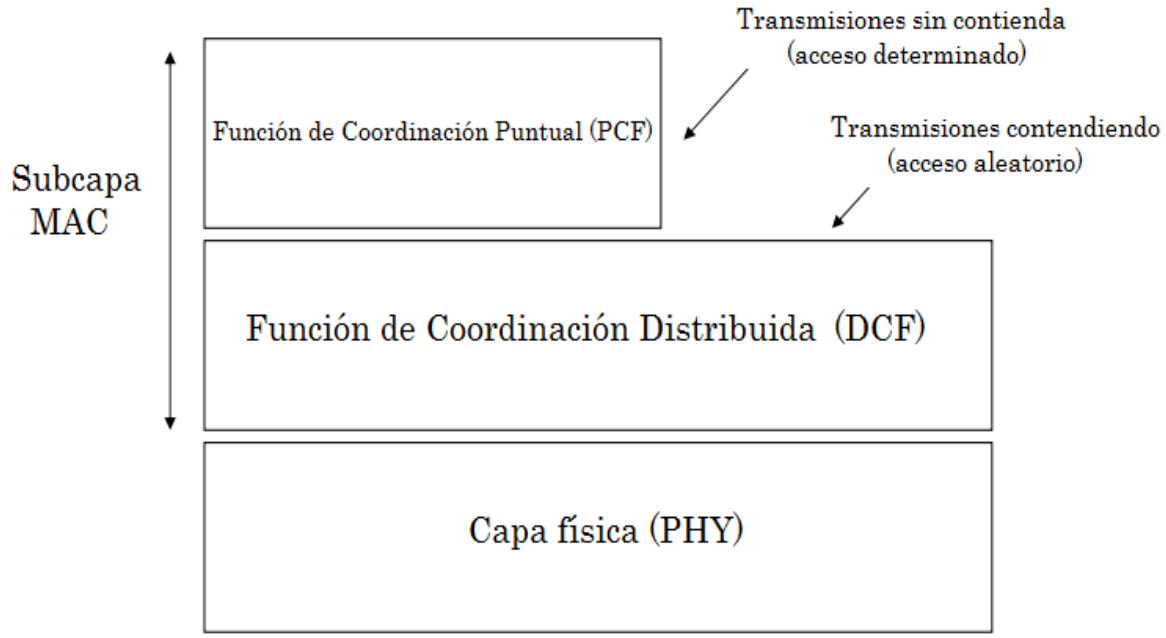


Figura 1.2 - Arquitectura lógica MAC del estándar IEEE 802.11g. [10]

Para los propósitos de este escrito, nos concentraremos en la DCF cuya función es la de determinar cuándo una estación puede transmitir o recibir unidades de datos de la subcapa MAC (sean tramas de control o MSDU de datos) desde el medio inalámbrico. La característica principal de la DCF es la de armar un escenario de contienda por el canal. La red implementará un algoritmo capaz de determinar de forma eficiente, rápida y uniformemente aleatoria (justa), a que estación le corresponde ocupar el canal inalámbrico para que las demás esperen.

Para lograr dicho objetivo, la DCF ocupa un sistema de escucha del canal de acceso múltiple que implementa una técnica para evitar colisiones, conocido como CSMA/CA (por sus siglas en inglés). Una colisión puede entenderse como la pérdida de dos o más paquetes de datos que llegaron al mismo tiempo al receptor y éste no fue capaz de diferenciarlos ni procesarlos a capas superiores.

Con una DCF se pretende reducir tiempos de espera en la red al hacer más dinámico el acceso al medio. La idea es que las estaciones por sí mismas (escuchando constantemente el canal), puedan estar enteradas del estado de la red y administrar cierto orden en sus transmisiones para evitar, en la mayor medida de lo posible, colisiones de paquetes que tienen como consecuencia desaprovechar el canal con tiempos de espera muy grandes.

1.4 Detección en el canal CSMA/CA.

La DCF de la subcapa MAC está basada en un procedimiento por el cual se escucha el canal antes de que una estación intente transmitir algún paquete (sea de control o de datos). Las colisiones pueden evitarse de manera significativa con técnicas de detección en el canal como el CSMA. [11]

En las redes alambradas se utiliza una estrategia muy simple por la cual las estaciones conectadas 'escuchan' al canal y si se detecta libre, inmediatamente pueden empezar su transmisión con bajas probabilidades de que ocurra una colisión. En principio esto es posible porque todas las estaciones en la red pueden escuchar al medio alambrado sin importar la distancia de separación entre ellas.[4]

Desafortunadamente para las redes inalámbricas de área local, las estaciones tienen un rango de detección limitado, marcado por las propiedades físicas de radiofrecuencia, y no lograrían 'escuchar' al medio en su totalidad con una técnica de detección simple como el CSMA usada en redes cableadas. En redes cableadas es posible implementar la detección de colisiones (CSMA/CD) pero en redes inalámbricas no es posible la 'detección' por el problema conocido como near/far[5] y porque en el medio RF de las redes de área local es complicado transmitir y escuchar al mismo tiempo; se requerirían antenas full-duplex que son mucho más costosas.

Esto nos lleva a mencionar dos de los principales problemas que podemos encontrar en las redes inalámbricas, dichos escenarios son conocidos como 'Terminal Oculta' y 'Terminal Expuesta'.

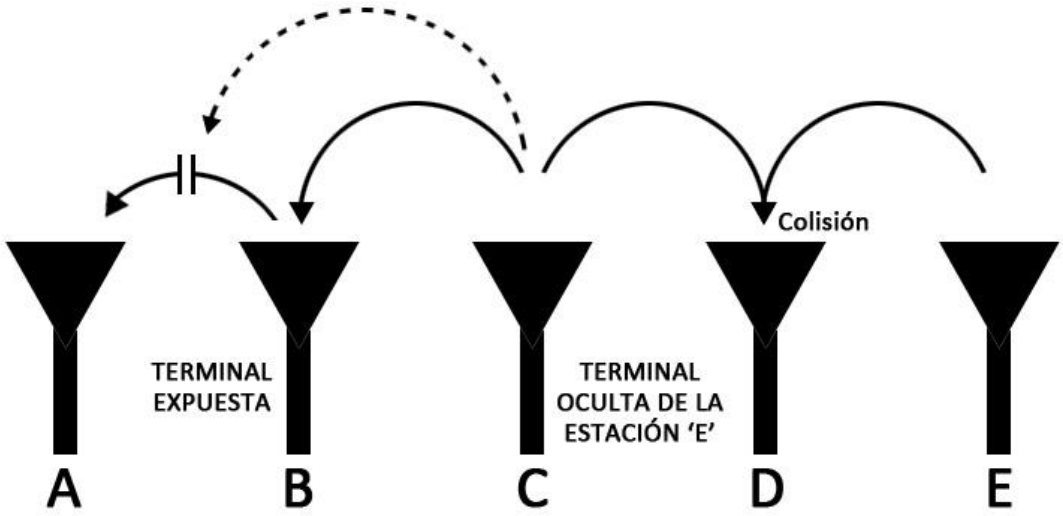


Figura 1.3 – Terminal oculta y terminal expuesta.

Se dice que una terminal es oculta de otra, cuando su rango de escucha no logra detectar sus transmisiones. Como se aprecia en la Figura 1.3, la estación C intenta transmitir a la estación D, detectará el canal libre al no lograr escuchar la transmisión actual de la estación E hacia D, de modo que, tras asumir el canal 'libre', enviará sus paquetes a D y colisionarán con los paquetes de E. Se dice entonces, que la terminal C está oculta de la estación E.

[4] Existe, por supuesto, un límite en las propiedades físicas del conductor, marcado por un retardo en las señales que viajan a través, sin embargo se desprecia para las distancias que se utilizan en redes LAN.
[5] El problema near/far acontece cuando un receptor se concentra en la señal recibida más fuerte y discrimina a las señales más débiles; bloqueando así la posibilidad de detectar oportunamente interferencias potenciales.

Se dice que una terminal está expuesta cuando ésta se abstiene de transmitir porque detecta una transmisión en el medio que originalmente no afectaría a su mensaje inicial. Como se aprecia en la Figura 1.3, la estación B está expuesta porque cuando intenta transmitir un mensaje a la estación A, se abstiene de hacerlo por que escucha la transmisión de la estación C hacia la estación D. Sin embargo, la transmisión de la estación C no interferiría con la estación A, de modo que sin ser necesario, la estación B no transmite y desperdicia tiempo bajando así el rendimiento de la red.

Para lidiar con estas dos situaciones y al mismo tiempo compensar la falta de un mecanismo infalible de detección en la red inalámbrica, se implementa un sistema de evasión de colisiones (CA por sus siglas en inglés). Las redes bajo el estándar IEEE 802.11g usan entonces un protocolo en conjunto denominado CSMA/CA como parte de su DCF. Dicho mecanismo consta de un saludo de cuatro etapas (*four-way handshake*) y aunque no se trata de un sistema totalmente infalible, sí ayuda en gran medida a combatir el problema de múltiples colisiones en el canal inalámbrico, y por lo tanto, mejorará directamente el rendimiento de la red. Existe aún la posibilidad de que dos estaciones detecten libre el canal y manden una petición exactamente al mismo tiempo ocasionando una colisión, de ser el caso la subcapa MAC debe ser capaz de identificarla para retransmitir el paquete por si misma difiriendo de las capas superiores. [12]

Esta secuencia está definida por el siguiente orden de tramas: RTS-CTS-DATA-ACK. Para que se considere una transmisión completa y exitosa, estas cuatro tramas deben ser entregadas satisfactoriamente. Primero se requiere de un Request-to-Send (petición para enviar), después de un Clear-to-Send (libre para enviar), se envía el mensaje (DATA) y se verifica la correcta recepción con un paquete de Acknowledgment (reconocimiento).

El saludo de cuatro vías empieza cuando una estación desea transmitir un paquete que viajará por el medio inalámbrico; primero la estación escucha el canal, si el canal permanece libré por un periodo de tiempo DIFS[6], la estación entrará en un proceso de retroceso ('backoff' explicado en la siguiente sección) en el caso de contienda aleatoria por el medio y al llegar la cuenta regresiva a cero (en unidades de SlotTime), transmitirá una trama de control RTS. Se da un espacio de tiempo SIFS[7] para recibir una trama de control CTS de la estación receptora, si se recibe correctamente entonces la estación emisora esperara un tiempo SIFS y en seguida enviará su paquete de DATA, esperará finalmente un tiempo SIFS por un paquete de control ACK de la estación receptora para concluir la transmisión del mensaje. De ser el caso en que no se reciba a tiempo alguno de estas tramas, se considerará que ocurrió una colisión y se solicitará la retransmisión completa del paquete, las retransmisiones pueden seguir solicitándose hasta un límite donde el paquete es desechado y se notifica a las capas superiores.

[6] Distributed Inter-Frame Space, es el intervalo de tiempo definido que una estación debe esperar como mínimo antes de iniciar una transmisión. Los intervalos de tiempo entre tramas (IFS) ayudan a darle un orden y jerarquización a las transmisiones en la red. [13]

[7] Short Inter Frame Space, es el intervalo de tiempo definido que se debe esperar para obtener respuesta de una trama de control.

Durante el proceso descrito, las demás estaciones deberán permanecer en silencio y sus contadores de retroceso backoff no deben disminuirse. Esto se logra con un mecanismo virtual de escucha del canal que actualiza una tabla NAV en las estaciones contendientes [14]. Cuando una estación escucha un paquete RTS[8], actualizara su NAV para esperar hasta la finalización de la transmisión completa (cuando se reciba el ACK de confirmación), lo mismo ocurre si lo que se escuchará fuera un paquete CTS. De esta forma garantizamos solucionar en la mayor medida de lo posible el problema de la terminal oculta.

El procedimiento de las cuatro vías así como la actualización de las tablas NAV se ilustra en la Figura 1.4.

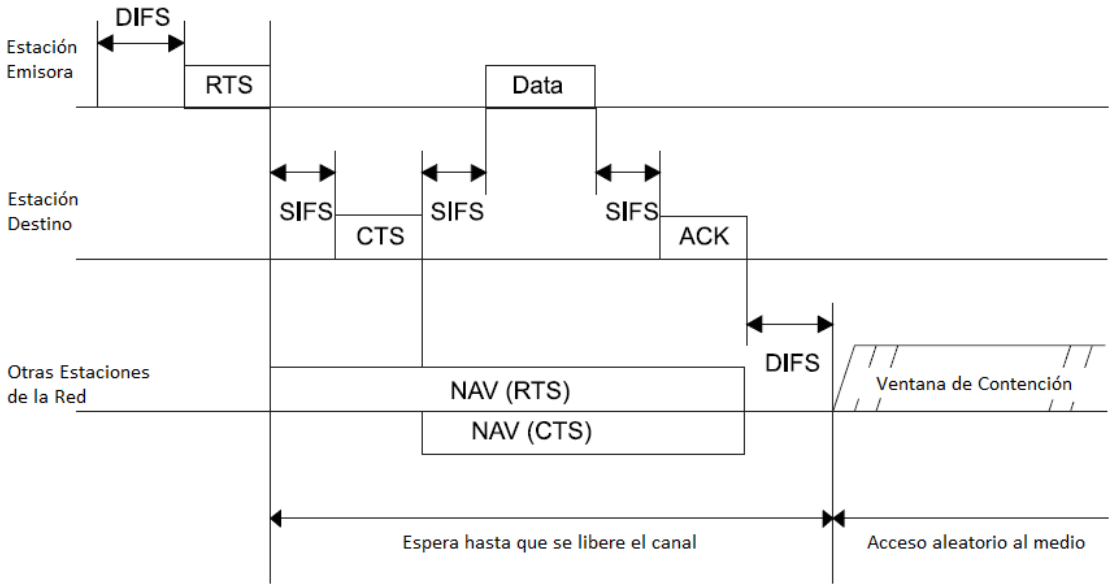


Figura 1.4 - Saludo de cuatro vías para transmitir un paquete bajo el protocolo CSMA/CA. IEEE 802.11 Std.[9]

Cabe señalara que el uso de el saludo de cuatro vías RTS-CTS-data-ACK es opcional en las redes IEEE 802.11g, esto porque puede comprometer el rendimiento de la red en los casos donde el tráfico es bajo y no tenemos gran número de estaciones contendiendo por el canal. Es recomendable su uso sólo cuando el tamaño de los paquetes MSDU es significativamente mayor al tamaño predefinido de las tramas RTS y CTS.

Cuando utilizamos paquetes de control como RTS y CTS logramos lidiar con el problema de la terminal oculta, pero al mismo tiempo estamos agregando tiempo 'muerto' de encabezados en bytes y desperdicio de tiempo de transmisión (SIFSrts+SIFScts) en la red inalámbrica. Cabe señalar que las redes que implementan el acceso aleatorio al medio con CSMA/CA como su protocolo

[8] Las tramas RTS y CTS poseen un fragmento especial en sus encabezados que especifican la duración la transmisión en curso. Este dato es particularmente útil para actualizar la tabla NAV.
[9] IEEE Std. 802.11, 'Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications', ANSI/IEEE, 1999. Figure 53, pp. 79.

MAC, no logran superar un 60% de rendimiento de la red sobre la tasa cruda de transmisión en el canal y cuando su tráfico aumenta, esta eficiencia se degrada exponencialmente. [15]

Hay mucha investigación al respecto del rendimiento que puede lograr este mecanismo de CSMA/CA con el saludo de cuatro vías, y del mismo modo hay diversas aproximaciones y propuestas acerca de cómo mejorarlo [16-25]. Misma información nos ayudará a justificar el planteamiento del objetivo de nuestro trabajo de tesis más adelante.

1.4.1 Binary Exponential Backoff.

La parte que se encarga de la evasión de colisiones (CA) se desarrolla fundamentalmente con un conteo decreciente exponencial binario conocido como el Binary Exponential Backoff [25]. Dicho procedimiento forma parte del saludo de cuatro vías cuando se presenta una contienda aleatoria por el medio en la DCF.

El proceso BEB tomará lugar cuando una estación detecte el canal ocupado en un intervalo de tiempo DIFS priori a su intento de transmisión o cuando sea interrumpida durante la misma, después de una retransmisión y después de una transmisión exitosa. La estación escogerá un valor entero positivo aleatorio dentro de un rango conocido como ventana de contención (CW por sus siglas en inglés). Dicho valor representa un tiempo de espera en unidades de SlotTime[10] que deberá 'consumir' antes de intentar transmitir su mensaje. La ventana de contención, de donde la estación escogerá un número al azar, esta acotada de 0 hasta CWmin-1.

Las estaciones que estén compitiendo por acceso al canal, podrán disminuir en una unidad el valor del CW que hayan escogido mientras detecten el canal libre y sólo mientras este libre. De manera que cuando su conteo regresivo BEB llegue a cero, podrán entonces acceder al canal y transmitir su mensaje. Sin embargo, las estaciones no pueden disminuir su contador de CW mientras haya una transmisión en proceso, esto incluye todas las etapas del four-way handshake.

Este proceso ayuda a que la contienda por el canal sea un proceso más azaroso y por tanto se disminuye la probabilidad de conflictos en el medio como bajo rendimiento o un gran número de colisiones. Y sin embargo, aún con nuestro saludo de cuatro vías y el mecanismo BEB en proceso, las colisiones aún pueden ocurrir. Por ejemplo, en el caso de que dos estaciones detecten el canal libre exactamente al mismo tiempo y obtengan un CW igual (situaciones perfectamente posibles), el contador de ambas estaciones llegará a cero e intentarán transmitir su mensaje al mismo tiempo provocando una colisión inminente. O en otra situación, si el rango de escucha de una estación no logra distinguir una transmisión en proceso y por tanto disminuye su CW como si el canal estuviera libre, cuando llegue su contador a cero transmitirá y colisionará con la transmisión en proceso.

[10] La duración del SlotTime variará dependiendo de las características de la capa PHY. Para el estándar IEEE 802.11g con OFDM/CCK el valor de SlotTime es de 9µs. [26]

Cuando se da el caso de una colisión, las estaciones involucradas doblarán su CWmin y escogerán un nuevo valor para su CW. Aquí es donde se visualiza un crecimiento exponencial del BEB. De modo que si dos estaciones o más colisionan, escogerán un valor de CW entre 0 y 2*CWmin-1. Esta duplicación de la ventana de contención es acumulativa hasta un límite CWmax. Y cada vez que existan colisiones consecutivas, la ventana seguirá duplicándose hasta llegar a un punto donde la estación escogerá su nuevo valor CW entre 0 y CWmax-1. Dicho comportamiento se ilustra en la Figura 1.5 donde se distingue un CWmin de 8 y un CWmax de 256.

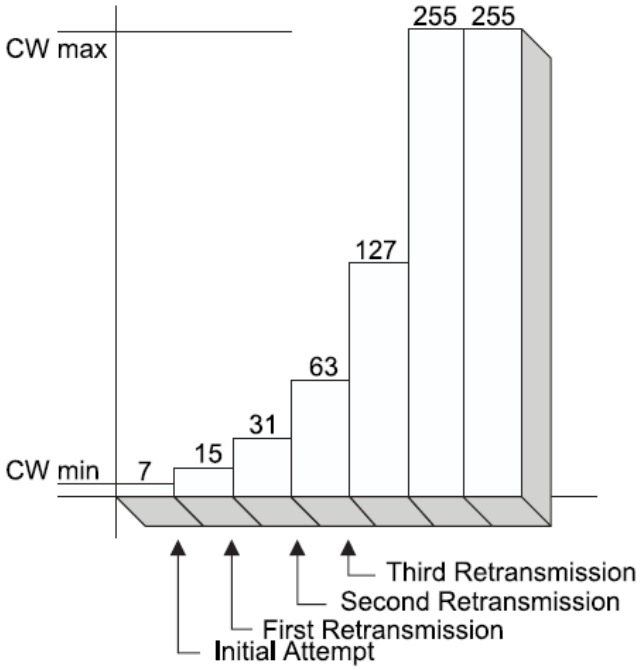


Figura 1.5 – Crecimiento exponencial de la ventana de contención iniciando en CWmin y llegando hasta CWmax.[11]

Los valores para CWmin y CWmax suelen ser múltiplos binarios y cada versión de los estándares IEEE 802.11 especifica su propia configuración. Otra manera de referirse al valor de CWmax es como el número de niveles que la ventana incrementará de forma binaria, por ejemplo; para un CWmin = 8 y que llegará a un CWmax = 256 (Refiérase a la Figura 1.5), se tiene un valor de niveles igual a 5. Suele referirse a este valor con la letra 'm' (mStages en inglés).

Desafortunadamente, tanto las tramas de control del saludo de cuatro vías como el proceso BEB agregarán tiempo de espera a las transmisiones desaprovechando el uso de la red. De modo que su rendimiento se verá directamente comprometido. Las colisiones pueden ser prevenidas, pero como consecuencia se degradará la tasa de transmisión efectiva en nuestra red inalámbrica.

Este es un factor muy importante respecto a los tiempos de espera que introduce la ventana de contención en el proceso de BEB. Es muy importante estar consientes de que una ventana inicial

[11] IEEE Std. 802.11, 'Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications', ANSI/IEEE, 1999. Figure 50, pp. 76.

(CWmin) estrecha es igual de negativa que una ventana muy amplia. Por ejemplo, si el valor inicial CWmin es muy pequeño, la probabilidad de que dos estaciones obtengan el mismo valor de CW es muy grande y por tanto la probabilidad de una colisión aumenta. Por otro lado, si la ventana es muy grande, los tiempos de espera serán mayores e innecesarios. Ambos casos (gran número de colisiones y tiempo de red desperdiciado en espera), son perjudiciales para el rendimiento de la red que es función directa del tiempo que se utilice para transmitir paquetes de datos[12].

Explicado entonces el mecanismo completo que llevan a cabo las redes IEEE 802.11g en la subcapa MAC por medio de su DCF con CSMA/CA que incorpora el saludo de cuatro vías con BEB; resta sólo calcular el tiempo total que tomará a una estación enviar un solo paquete de datos. Dicho tiempo lo podemos calcular con la Expresión 1.1.

$$t[s] = BO_{SlotTime} + DIFS_{sec} + 4PLCP_{sec} + \frac{RTS_{bits} + CTS_{bits} + DATA_{bits} + ACK_{bits}}{R_{bits/sec}} + 3SIFS_{sec}$$

Expresión 1.1 – Cálculo del tiempo total para transmitir un paquete de datos. [27]

En donde BO (backoff) es la espera en unidades de SlotTime que la estación debe esperar en el proceso de disminuir su CW y el PLCP (Physical Layer Convergence Protocol) es el preámbulo que demorará la capa física para hacer su transmisión, esta involucra tiempos de propagación y sincronización en el medio inalámbrico.

Como podemos concluir, la tasa de transmisión real por estación, será menor que la velocidad cruda del canal (R).

1.5 Algoritmos genéticos.

Los algoritmos genéticos se han convertida en una de las formas más inteligentes para encontrar la mejor solución a problemas de optimización donde el espacio de posibles soluciones es muy amplio o no es factible analizarlas en su totalidad [28]. Se trata de un modelo de búsqueda heurístico[13]. Desde sus fundamentos por el Dr. John Henry Holland en los años '70s [29] los algoritmos genéticos han demostrado un futuro prometedor para el campo de la inteligencia artificial, cómputo avanzado y optimización de sistemas.

Dichos algoritmos son llamados 'genéticos' por que emulan el principio de la Selección Natural de Charles Darwin [30] en el cual se habla acerca de una evolución y mejora de la especie por mutaciones genéticas, y al mismo tiempo, la sobrevivencia sólo de los más aptos. Cada individuo tiene una cadena única de cromosomas dentro de su material genético que lo representa como tal, sus fortalezas y sus debilidades; dicho de otra manera, su grado de aptitud para continuar existiendo en la naturaleza; como es de esperarse y se ha comprobado científicamente, los individuos con más debilidades y con fortalezas que no son significantes, terminan por extinguirse si no presentan mutaciones que recaigan en una evolución parcial o completa de la especie.

[12] Dicha afirmación nos ayudará a plantear nuestro objetivo en la sección 1.5 de este mismo capítulo.
[13] Un modelo de búsqueda heurístico es usado cuando una búsqueda exhaustiva por todo el campo de posibles soluciones es impráctica.

Pero, ¿cómo se relaciona esto con un algoritmo que pretende buscar la mejor solución a un problema de optimización? La respuesta es que el algoritmo va a interpretar el conjunto de posibles soluciones como una población de individuos, cada uno con sus fortalezas y debilidades que representan que tan 'aptos' son para solucionar el problema inicial. En lugar de hacer una búsqueda lineal o estática para hallar la mejor solución; los algoritmos genéticos realizan una búsqueda dinámica donde seleccionarán a los individuos más aptos para su reproducción, recombinación o mutación que aseguren una evolución para obtener mejores sujetos, de esta forma se creará una mejor generación de individuos conforme avanza el tiempo (totalmente análogo a la evolución natural de las especies). Dado que el objetivo principal es encontrar la mejor solución, se espera que cada generación sea mejor que la anterior hasta llegar a un punto de convergencia donde se puede declarar al mejor sujeto de la población. Este punto de convergencia dependerá de un criterio de terminación establecido dentro del algoritmo genético [31]. El criterio de terminación puede ser cuando se alcance un cierto número de generaciones, o cuando no se observa un cambio significativo en el sujeto más fuerte de la generación anterior en comparación con la actual.

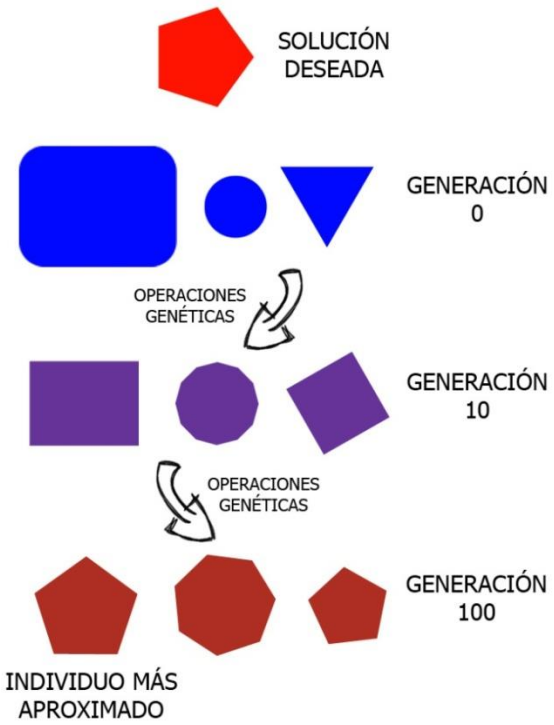


Figura 1.6 - Iteración de generaciones hasta llegar a una población con individuos lo más parecidos a la solución deseada.

En la Figura 1.6 se ilustra un ejemplo sencillo en donde la solución deseada es un polígono mediano de cinco lados. Las características (genes) que definen a los sujetos de la población son el número de lados y el tamaño de la figura. Se mutará una generación cero o población inicial cuantas veces sea necesario hasta alcanzar la solución deseada o al menos una muy buena aproximación. En la generación cero no necesariamente tenemos individuos cercanos a la solución

buscada; sin embargo, observamos que en la generación cien, los individuos ya están muy cerca de la solución deseada. Los individuos que sean más aptos (aquellos que representen una mejor solución al problema inicial) tendrán mejores posibilidades de sobrevivir y mutar en una mejor solución en las futuras generaciones.

Un algoritmo genético está diseñado de tal forma que evaluará a todos los individuos de una población inicial (generación cero) con la ayuda de una 'función de aptitud' (*fitness function* en inglés) que en la mayoría de los casos nos da una aproximación cuantitativa de que tan 'bueno' es ese individuo dentro del espacio de posibles soluciones. El algoritmo debe ser capaz de identificar de esta manera a los individuos más aptos para entonces armar la siguiente generación con base a los más fuertes. Existe un criterio muy popular dentro de los algoritmos genéticos conocido como 'elitismo' que consta en copiar sin cambio alguno al individuo más fuerte de la generación actual en la siguiente generación.

1.5.1 Programación genética.

Una de las principales interrogantes cuando se pretende programar un algoritmo genético es: ¿cómo va a mutar el algoritmo a los individuos de mi población de soluciones iniciales? La respuesta reside en la correcta codificación de los mismos.

Es cierto que los algoritmos genéticos se pueden aplicar a casi cualquier tipo de problema de optimización; sin embargo funcionan mejor con problemas donde la solución es fácilmente cuantificable, es decir, en términos numéricos o dimensionales con magnitudes lineales porque se pueden manejar de forma más sencilla a los 'individuos' que representarán una magnitud escalar en el mejor de los casos. Es decir, la solución a mi problema inicial es una cantidad o cantidades numéricas perfectamente computables.

Cuando se trabaja en el diseño de un programa genético se debe ser muy cuidadoso en la codificación que se usará para los individuos. Las principales codificaciones son la binaria, octal, hexadecimal, números reales, por valor y en modo de árbol [32]. La idea es expresar a cada individuo en cadenas numéricas que puedan ser fácilmente manipulables y representen los 'cromosomas' del material genético de cada uno. Estas formas de codificar a los individuos como cifras numéricas nos ayudarán a que las operaciones que debe realizar el algoritmo genético sean más evidentes. Dichas operaciones, de nuevo, son análogas a los fenómenos de la biología.

Dependiendo del programador, y bajo su criterio de lo que considere conveniente para su problema inicial; se pueden usar las operaciones de: Reproducción, Recombinación, Mutación y Evolución (Recombinación + Mutación).

La Reproducción requiere de un solo individuo de la población que será copiado sin cambio alguno a la siguiente generación. La Mutación requiere de un solo individuo que sufrirá cambios arbitrarios en sus cromosomas (en algunas cadenas aleatorias de su material genético). La Recombinación requiere de dos individuos, un padre y una madre, que darán fruto a uno o dos hijos que contendrán cierta porción del material genético del padre y otra porción de la madre. La

Evolución requiere de dos individuos que primero llevarán a cabo una Recombinación y después una Mutación en cada uno de los hijos.

Cada operación genética tiene una probabilidad de ocurrir más que otra. Las más populares, por haber demostrado mejores resultados, son la Reproducción y la Evolución; que se sugiere tengan mayores posibilidades de ocurrir durante nuestro algoritmo que las otras dos. De esta forma garantizamos la supervivencia de los más fuertes y la mejora de la especie.

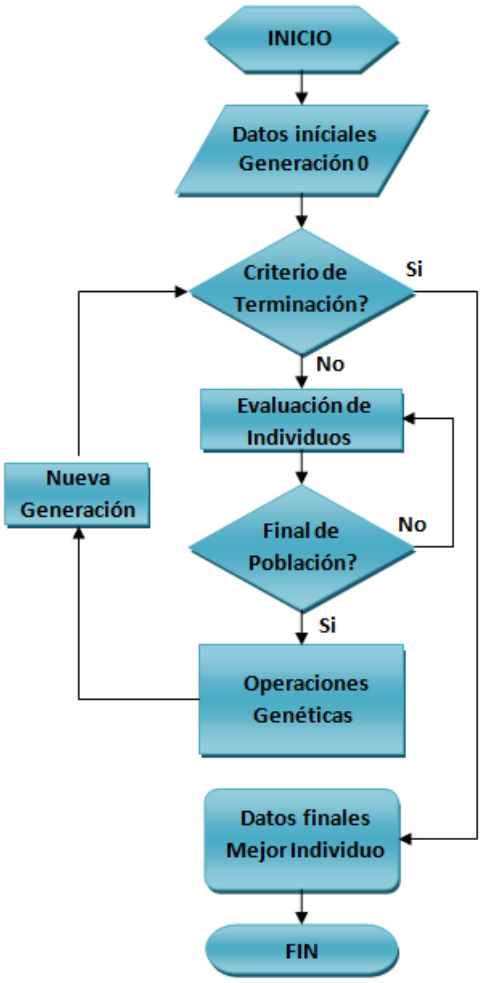


Figura 1.7 - Diagrama de flujo general para un programa genético.

Dicho esto; un programa genético es entonces una serie de instrucciones autónomas que colectan datos iniciales concernientes a la población inicial o generación cero y al criterio de terminación; codificará a los individuos, los evaluará con una función de aptitud y ejecutará operaciones genéticas con los más fuertes para crear una nueva generación. Repetirá este proceso hasta que se alcance una convergencia o se cumpla el criterio de terminación. Y finalmente expondrá los datos finales que debe ser la información del individuo más fuerte de la última generación iterada. En la Figura 1.7 encontramos un sencillo diagrama de flujo que representa un programa genético.

1.6 Justificación. Planteamiento del problema.

El propósito formal del presente trabajo de investigación es el de proponer una forma más eficiente para manipular el acceso al medio y con esto evitar pérdida de datos (colisiones); mejorando así el uso de la capacidad en las redes inalámbricas y todo con el objetivo final de maximizar el rendimiento de la misma, en específico de las redes que funcionan bajo el estándar IEEE 802.11g. Esto se pretende lograr mediante una aproximación de búsqueda inteligente diseñada para problemas de optimización conocida como 'algoritmos genéticos' y mediante una aplicación desarrollada en Java que evaluará diferentes opciones que sugieren mejores configuraciones para valores del *Binary Exponential Backoff* en el *Carrier-Sense Multiple Access with Collision Avoidance* en la subcapa del *Medium Access Control* (MAC). Es importante señalar que mejorar la eficiencia del BEB sería altamente gratificante dado que es uno de los métodos de *Backoff* más populares en mecanismos MAC utilizados actualmente [33].

Como hemos mencionado con anterioridad y los estudios previos demuestran [17-25], el rendimiento de las redes inalámbricas que trabajan bajo el estándar IEEE 802.11g sin ningún tipo de cambio en las características que se especifican para la capa PHY y lo relacionado al protocolo MAC en la capa de enlace, disminuye exponencialmente cuando aumenta el tráfico o el número de terminales en la red. El rendimiento de la WLAN disminuye drásticamente por un desaprovechamiento en la utilización del canal cuando el mecanismo BEB asigna tiempos de espera muy amplios o muy cortos para que las terminales inicien sus transmisiones; por ejemplo en redes públicas donde existen múltiples usuarios que desean conectarse a un *access point*, existen múltiples colisiones y por lo tanto los tiempos de contención aumentan exponencialmente.

Gran parte del problema recae en la forma en que la función de BEB trabaja en el proceso de CSMA/CA. El punto crítico está en saber seleccionar adecuadamente los valores para la ventana de contención, díganse los valores para CWmin y CWmax (mStages).

En las redes convencionales estos valores son estáticos y no varían conforme varían las condiciones de tráfico. Las redes inalámbricas bajan su rendimiento enormemente cuando los valores de la CW no se adaptan a la situación cambiante de la misma.

Por ejemplo, cuando el tráfico de red se vuelve mucho más pesado, convendría disminuir ligeramente la ventana de contención inicial CWmin, para compensar cargas pesadas con tiempos de espera más cortos; por supuesto esto sólo cuando el número de estaciones sea pequeño. Por otro lado, si tenemos tráfico ligero de red pero existen múltiples estaciones contendientes, convendría elevar el valor de CWmax o lo que es lo mismo, subir el número de niveles mStages.

Fundamentada la importancia de la correcta configuración de valores para la ventana de contención en el proceso de Binary Exponential Backoff que utiliza CSMA/CA; y el problema al que nos enfrentamos cuando dicha configuración permanece estática frente a cambios en las condiciones de la red; argumentamos que, existe trabajo de optimización que se puede desarrollar para ser capaces de escoger una mejor configuración en el protocolo MAC que maximice el uso eficiente del canal y por tanto aumente el rendimiento general de la red. Se vuelve necesario

modificar el mecanismo de acceso al medio que maneja el estándar IEEE 802.11 para reducir los tiempos de espera y aumentar la utilización del canal al mismo tiempo que se eviten colisiones, en conjunto todo esto nos dará como resultado un aumento notable en el rendimiento de la red.

Justificamos de esta manera el trabajo de investigación que se presenta en los siguientes capítulos y que consta de una nueva aproximación, más inteligente, para mitigar en gran medida el problema mencionado. Dicha solución se propone por medio de un algoritmo genético programado para arrojar valores de configuración BEB más adecuados a diferentes escenarios propuestos.

II - ESTADO DEL ARTE.

2.1 Algoritmos dinámicos, adaptivos y evolutivos para BEB.

En las redes inalámbricas de área local es el mecanismo de acceso al medio (MAC) el que determina en gran medida la eficiencia de la red y como se controlan las transmisiones que desean ocupar el canal. El máximo valor de rendimiento expresado como una tasa de transmisión real en bits por segundo, medida en función del tiempo que la red está transfiriendo paquetes de manera exitosa sobre la cantidad de tiempo total transcurrido; es conocido como la capacidad neta del protocolo.

Como se ha discutido en el Capítulo I del presente trabajo, el rendimiento neto del estándar IEEE 802.11g puede mejorarse de manera significativa con cambios en la DCF y más en específico, en la manera en que opera el BEB. A continuación se presentan trabajos sobresalientes que confirman esta afirmación con distintos modelados, modificaciones y aproximaciones para la obtención de los parámetros en el BEB de la DCF en redes IEEE 802.11 y sus resultados obtenidos respectivamente.

2.1.1 A new backoff method for IEEE 802.11. [34]

En este trabajo de investigación, presentado por Hadi Minooei y Hassan Nojumi, se argumenta la necesidad de un método de autocontrol inteligente por parte de las estaciones contendientes en una red inalámbrica IEEE 802.11 regidas bajo una DCF, esto dado que es una de las formas más populares para el acceso al medio y ha demostrado tener rendimientos muy por debajo del límite máximo teórico[14].

Del mismo modo se menciona la exhaustiva tarea de muchos investigadores del área para mejorar propiedades del mecanismo BEB como son el tiempo de demora entre transmisiones y la efectividad del control de colisiones.

El modelo propuesto por los autores se base en el mismo mecanismo original del estándar IEEE 802.11 para el algoritmo BEB, pero con dos cambios importantes que se analizan para concluir si en verdad mejoraría el rendimiento de la red o no.

El primer cambio es evitar la contienda de estaciones cuando la red está en una situación no saturada, es decir, cuando se tienen pocas terminales o el tráfico en la red es bajo. Esto sugiere escoger valores mucho más pequeños de la ventana de contención (CW) y por tanto maximiza el uso eficiente del canal al reducir tiempos de espera innecesarios.

El segundo cambio sugerido aplica cuando se transmite un paquete satisfactoriamente, se disminuirá la ventana de contención en un nivel (mStages -1) en lugar de reiniciarla hasta CWmin.

[14] G. Bianchi, 'Performance analysis of the IEEE 802.11 distributed coordination function', IEEE Journal on Selected Areas in Communications, 18 (3) (2000), pp. 535-547.

Esto representa un gran alivio para problemas como el de 'inanición' (starving[15]) en la red inalámbrica y podría maximizar de igual manera el rendimiento.

Con la representación de los posibles estados en la red inalámbrica por medio de un diagrama discreto en tiempo de Markov, se obtienen ecuaciones que podrán calcular el rendimiento promedio de la red por medio de una media matemática (el valor esperado del tamaño de los paquetes transmitidos, dividido entre el valor esperado del tamaño de un SlotTime).

Después de un análisis para distintas condiciones de red con 5, 10, 20 y 50 estaciones contendientes, se presentan los resultados y conclusiones del nuevo método BEB propuesto. En la Tabla 2.1 se observa la comparación de los rendimientos obtenidos con el nuevo método frente a los obtenidos con el estándar IEEE 802.11.

Maximum throughput of new method

N	Max throughput (exact)	Max throughput (approximated)
5	0.8312	0.8323
10	0.8280	0.8281
20	0.8260	0.8260
50	0.8247	0.8248

Maximum throughput of BEB (IEEE 802.11)

N	Max throughput (exact)	Max throughput (approximated)
5	0.8328	0.8327
10	0.8283	0.8283
20	0.8261	0.8261
50	0.8248	0.8248

Tabla 2.1 - Valores máximos normalizados de rendimiento de red. [16]

Como se puede apreciar, los valores no varían significativamente y el estándar IEEE 802.11 por sí mismo, supera el rendimiento máximo de la red frente al nuevo método BEB para los cuatro casos. A partir de dicha información, las conclusiones obtenidas por los autores del nuevo método BEB sugerido, gracias al análisis matemático y a pruebas comparativas; son las siguientes.

El nuevo método BEB presenta mejores resultados frente a estaciones contendientes en un escenario no saturado de red. En condiciones ordinarias, el máximo valor de rendimiento conseguido es prácticamente el mismo como se aprecia en la Tabla 2.1.

[15] Se dice que ocurre un problema de 'starving' en la red cuando una estación acapara el canal con sus transmisiones, de manera que las demás colisionan y aumentan su CW volviendo más complicado para ellas obtener acceso al medio.

[16] Derechos al autor. Tablas 2 y 3 del documento citado en [34].

Aunque no se logró mejorar el desempeño de la red inalámbrica, si se logró ilustrar la forma en que el valor teórico máximo es representado como una relación de valores esperados de las transmisiones exitosas frente al tiempo invertido en los intentos.

Quedó demostrado que es imperativo mejorar la forma en que la red inalámbrica maneja los 'cuellos de botella' para el acceso al medio. Futura investigación puede desarrollarse para simulaciones en tiempo real del método propuesto en condiciones saturadas y no saturadas.

2.1.2 Dynamic tuning of the backoff mechanism in IEEE 802.11. [35]

En este trabajo de investigación, presentado por Raffaele Bruno, Marco Conti y Enrico Gregori, se estudia el desempeño de un novedoso mecanismo dinámico para ajustar el BEB con una simple estimación de las condiciones de la red en tiempo real. En especial, el mecanismo sugerido aprovecha la información que obtiene al escuchar el canal (CSMA). En general, cuando el tráfico y/o condiciones de red cambien, el mecanismo dinámico también cambiará para ajustar el BEB y obtener un mejor rendimiento máximo de aprovechamiento del canal.

Dicho trabajo aprovecha muchas de las características estudiadas en una investigación previa[17] que maneja el concepto de un protocolo IEEE persistente (*p-persistent IEEE protocol*).

El protocolo persistente explica el uso de una distribución geométrica con parámetro 'p' para definir el tamaño de la CW en lugar del incremento binario original utilizado en el BEB. El resto de las reglas correspondientes al funcionamiento de la DCF son iguales al estándar IEEE 802.11.

El valor de 'p' en esta distribución geométrica será calculado a partir de las condiciones de tráfico en la red inalámbrica. Esto asegura el moldear la ventana de contención CW de forma dinámica frente a cambios en las características de contienda basándose en la información de detección CSMA. De esta forma, cada estación implementará un algoritmo distribuido para ajustar el conteo regresivo (backoff) de su CW antes de iniciar una transmisión.

El presente trabajo está basado en el protocolo IEEE dinámico (Dynamic IEEE protocol), el cual es mucho más complejo al requerir la estimación de estaciones en la red. Sin embargo, se sugiere una aproximación mucho más sencilla denominado Protocolo Dinámico Simple IEEE (SDP por sus siglas en inglés). Siendo su principal característica el no necesitar de estimar el número M de estaciones contendientes en la red inalámbrica.

El SDP trabajará únicamente con los datos concernientes al tiempo en que el canal esta libre, ocupado por colisiones y ocupado en transmisiones exitosas. De esta forma se puede calcular una estimación muy acertada de las condiciones generales del medio en tiempo real. A continuación de describe de forma resumida el funcionamiento de SDP.

[17] F. Calí, M. Conti, E. Gregori, 'IEEE 802.11 wireless LAN: capacity analysis and protocol enhancement', Proceedings of INFOCOM 98, San Francisco, IEEE/ACM Trans on Networking, December 2000.

Primeramente las estaciones calcularan un valor esperado del tiempo en que el canal esta libre y el tiempo en que el canal se desperdicia por colisiones con la ayuda del mecanismo de detección CSMA. Seguido se calculará un nuevo valor de 'p' para el protocolo persistente y se aplicará un factor de amortiguamiento α tomando en cuenta el valor 'p' anterior. Dicho valor será calculado para cada periodo de transmisión (compuesto por la suma del tiempo libre del canal más el tiempo de transmisión exitosa o tiempo de transmisión terminada en colisión) y por cada estación de forma independiente. Dicho algoritmo se puede apreciar sin modificaciones de la obra original en la Figura 2.1.

Begin

step 1: $Idle\_p_n$ = measure of the n-th the idle period;

step 2: $Coll_n$ = measure of the n-th collision cost;

step 3: $E[Idle\_p]_n = \alpha \cdot E[Idle\_p]_{n-1} + (1-\alpha)_n \cdot Idle\_p$

step 4: $E[Coll]_n = \alpha \cdot E[Coll]_{n-1} + (1-\alpha) \cdot Coll_n$

step 5: $p_{comp} = p_{n-1} \cdot \sqrt{\frac{E[Idle\_p]_n \cdot t_{slot}}{E[Coll]_n}}$

step 6: $p_n = \alpha \cdot p_{n-1} + (1-\alpha) \cdot p_{comp}$

End.

Figura 2.1 - Algoritmo para el ajuste de *backoff* por medio del protocolo SDP. [18]

En la obra exhaustiva presentada en [35] se hacen análisis para diferentes escenarios de red, tomando en cuenta tráfico estable, tráfico cambiante, para distintos tamaños de paquetes de datos y múltiples estaciones contendientes, se analizan inclusive cambios bruscos en las condiciones de la red para determinar el tiempo de convergencia del protocolo SDP y finalmente se estudia el rendimiento bajo condiciones realistas de red (donde propiedades físicas del canal, tales como desvanecimientos, afectan a las transmisiones).

Las conclusiones arrojadas por esta obra señalan al SDP como una significativa mejora frente al BEB original del estándar IEEE 802.11. El protocolo simple dinámico basado en una distribución 'p' geométrica para el cálculo de la CW mostró elevar el rendimiento de red para todos los casos explicados con condiciones cambiantes de red; arrojando siempre tiempos de convergencia en alrededor de apenas un par de segundos.

Las únicas desventajas encontradas del SDP son un ligero aumento en el tiempo necesario para una transmisión y que su rendimiento es menor que el original del estándar IEEE 802.11 para redes con bajo tráfico.

[18] Derechos al autor. Figura 2 del documento citado en [35].

2.1.3 Adaptive backoff algorithm for IEEE 802.11. [36]

Para el presente trabajo de investigación de Maali Albalt y Qassim Nasir, se propone un mecanismo de BEB basado en el historial de transmisiones exitosas de cada estación en lugar de únicamente el último intento por ocupar el canal como lo hace el estándar original IEEE 802.11. Con esto, se pretende mejorar la calidad de servicio (QoS), el rendimiento de la red inalámbrica y disminuir el tiempo de retardo por transmisión. Se presentan los resultados para simulaciones del nuevo mecanismo adaptivo propuesto frente a los resultados obtenidos por el estándar IEEE 802.11 y su implementación en una red MANET (Mobile Ad-Hoc Network) con la ayuda de hardware especializado bajo una plataforma Linux. Al final del documento se concluyen posibles mejoras en el rendimiento de hasta 8.56% y en retardo por transmisión de hasta un 34.3%.

Los autores hacen mención de la gran variedad de trabajos previos que existen respecto al mejoramiento del algoritmo BEB para redes IEEE 802.11 y hacen hincapié en el hecho de que la gran mayoría de aproximaciones son apenas ligeras modificaciones y no cambian la forma de decremento en la ventana de contención (CW).

Los mejores resultados se presentaron para aquellos modelos prepuestos que disminuyen la CW de forma gradual, en lugar de un reinicio hasta CWmin después de una transmisión exitosa. De forma que este importante factor es tomado en cuenta para proponer su propia variante del BEB, los autores le llaman *History-Based Adaptive Backoff* (HBAB).

Este nuevo mecanismo para manipular la CW, basado en el historial de transmisiones en el medio inalámbrico, se caracteriza por no requerir cálculos complicados ni agregar una carga en el encabezado MAC que podría disminuir el rendimiento de la red. En pocas palabras, el algoritmo propuesto tomará en cuenta las recientes condiciones de la red para actualizar el valor de la CW en lugar de hacerlo de manera estática como se maneja en el BEB del estándar IEEE 802.11 en donde la ventana se duplica tras colisión y se reinicia a CWmin tras una transmisión exitosa.

El algoritmo HBAB trabaja con tres variables importantes, el valor actual de la CW, un factor multiplicador 'α' y un indicador de estado del medio 'ChannelState'. El factor para multiplicar puede ser cualquier entero positivo mayor a 1 y hace la función del pre multiplicativo fijo como 2 en el BEB original, sólo que en este caso el valor de α puede variar conforme a las condiciones de la red y se utilizará no solo en caso de colisión sino también en el caso de transmisión exitosa.

El indicador *ChannelState* sirve para guardar los últimos dos estados de la red, se trata de un contador que marcará como libre (1 binario) o como ocupado (0 binario) de modo que sus posibles valores son: 01 para un estado ocupado y luego libre, 10 para un estado libre y luego ocupado, 11 para dos estados libres consecutivos y 00 para dos estados ocupados consecutivos.

HBAB sugiere actualizar el valor de la CW en tres formas distintas dependiendo si hubo una colisión y si el estado de la red se ha marcado libre y ocupado; esto se expresa en la siguiente fórmula:

$$CW = \begin{cases} CW \cdot \alpha, transmission\ \ failure \\ CWmin, \text{transmission success, ChannelState} \neq 00 \\ \frac{CW}{\alpha}, \text{transmission success, ChannelState} = 00 \end{cases}$$

Figura 2.2 - Algoritmo sugerido HBAB para manipular la CW. [19]

Como podemos notar, la principal diferencia es la de no reiniciar la ventana CW en CWmin tras una transmisión exitosa en todos los casos, si no sólo en los casos en que la red no marque dos estados ocupados seguidos; esto porque es más probable que ocurran colisiones. En su lugar, se sugiere dividir la ventana CW entre el factor α cuando la red se detecta congestionada (con la variable ChannelState = 00).

A su vez, la ventana no necesariamente tiene que duplicarse cada que haya un intento fallido de transmisión, sino que se multiplica por el factor α que toma el valor de cualquier entero positivo mayor que 1.

Según los datos de simulación que los autores presentan, el nuevo algoritmo HBAB logra alcanzar hasta un 8.56% de mejora en el rendimiento de la red con un valor de α = 1.1. La información pertinente a la simulación y su implementación se puede encontrar en la Tabla 2 del documento original. [36]

Tras la simulación, los autores presentan los datos obtenidos de una implementación en hardware de dicho algoritmo; sin embargo hubo dificultades para su total realización debido a limitaciones en la manipulación de las características de las tarjetas de red. Aún con estas desventajas, los resultados presentados sugieren una mejora en el rendimiento, en el retardo por transmisión y en efectividad frente al estándar IEEE 802.11 por sí solo.

Aunque el aumento en el rendimiento de la red inalámbrica no fue mucho, el nuevo método HBAB estudiado sí presentó ser mejor que el estándar IEEE 802.11 en todos los casos analizados y bajo distintas condiciones de red así como múltiples valores de α. Con esto confirmamos nuevamente que existen distintas aproximaciones para mejorar el algoritmo BEB de la DCF en redes IEEE 802.11.

Los autores concluyen afirmando que su método puede ser mejorado para modificar el valor de α en tiempo real y que tomar en cuenta el histórico de transmisiones mediante la variable ChannelState fue de gran utilidad. La forma sugerida de actualizar la ventana CW mediante incrementos o decrementos controlados por un factor pre multiplicativo ha demostrado ser más óptimo que el incremento binario del BEB original.

[19] Derechos al autor. Ecuación 4 del documento citado en [36].

2.1.4 RegionDCF: self-adapting CSMA/Round-Robin media access protocol. [37]

El trabajo de investigación presentado por Alberto Riggi y Javier Gómez propone un mecanismo para el control de acceso al medio basado en la DCF del estándar IEEE 802.11 que actuará a la vez como CSMA y con una técnica Round-Robin[20] adaptándose siempre a las condiciones de tráfico en la red. La idea de esta adaptación es la de mejorar la utilización del canal disminuyendo los tiempos de espera que introduce el mecanismo de BEB en la DCF del estándar.

Dado que los tiempos de espera definidos por el *Exponential Backoff* en la DCF pueden provocar un impacto negativo en el rendimiento de la red para escenarios con múltiples estaciones contendiendo por el canal; se propone un mecanismo que se adapte a las condiciones para comportarse como una técnica puramente CSMA cuando existen pocas estaciones o como una técnica *'RegionDCF'* en escenarios con alta congestión de red y múltiples estaciones.

El mecanismo propuesto y denominado RegionDCF utiliza las ventajas de redes estructuradas (como la existencia de un nodo central denominado *access point*) y en primera instancia no está diseñado para redes ad-hoc. La principal característica de esta nueva aproximación, es la del concepto de *regiones*. De manera que, en lugar de tener un escenario donde cada estación contienda por ocupar el canal inalámbrico, tendremos regiones constituidas por múltiples estaciones deseando transmitir sus mensajes.

De esta forma se inicia una competencia por el canal entre regiones que deseen transmitir y cuando una región gana el acceso al medio, se implementa una técnica Round-Robin para verificar cuales estaciones de dicha región están listas para enviar su mensaje. Para ganar acceso al medio inalámbrico se utiliza la técnica original de la DCF conocida como BEB. Todas las estaciones entran en dicha competencia y la primera que gana acceso al canal le otorga automáticamente el derecho para transmitir a su región completa[21] y así se entra en un modo de acceso al medio libre de competencia donde cada estación de la región puede transmitir un paquete si es que lo requiere. Este proceso iniciado se conocerá como *Region Burst* (RB).

Para poder implementar dicha idea se sugiere el uso de un nuevo encabezado en los paquetes de control denominado *'Region Header'*. Dicho encabezado contendrá la información necesaria para distinguir entre regiones, para actualizar la tabla NAV de las estaciones no pertenecientes a la región actual y para actualizar el valor de un contador especial denominado *TOp* (transsmision oportunities) en base al campo *Reserved Slots*.

Parte del Region Header es denominado RSH (*Region Subheader*) que servirá para reservar el canal en slots de tiempo (análogo a una técnica TDMA) y cada estación de una región tendrá su

[20] La técnica *Round-Robin* consta en seleccionar un recurso de una lista que irá rotando cada elemento para que todos sean candidatos y en algún momento sean utilizados, de esta manera no existirá un problema de 'hambruna'.

[21] Cada estación poseerá dos identificadores, el *MemberId* para distinguirse de las demás y el *RegionId* para saber a qué región pertenecen. Las regiones están delimitadas por la distancia máxima de separación entre terminales donde se permita una comunicación en una tasa de transmisión básica.

oportunidad de transmitir en base a un contador TOp[22] que determinará cuantas transmisiones podrá soportar el RB actual. Una vez concluido un RB, el *access point* podrá emitir un paquete de control hacia todos los miembros de dicha región denominada como *Region ACK* (RA).

Un punto clave para este nuevo mecanismo es el de poseer dos contadores de espera distintos y que asegurarán un mejor aprovechamiento del canal. El primer contador es el utilizado para ganar acceso al medio e iniciar el RB, dicho contador es el original de la DCF el cual está basado en el BEB. Pero, una vez en el RB, cada estación tendrá su propio contador regresivo basado en el TOp inicial, dicho contador se actualizará en base a las transmisiones de los demás miembros de la región y con una separación SIFS entre mensajes, de esta forma si existiera una colisión o si una terminal de la región actual no tuviera mensaje a transmitir, el máximo desaprovechamiento sería de un periodo SIFS. Durante todo este mecanismo Round-Robin las demás regiones deben considerar el canal ocupado hasta que finalice el RB. Una vez finalizado, se empieza una competencia BEB nuevamente para ganar acceso al medio e iniciar un nuevo RB.

Para las transmisiones de descarga provenientes del AP (*access point*) existe un periodo especial denominado *'AP burst'*. La idea es darle cierta ventaja al AP cuando se inicie una contienda por acceso al canal, dicha ventaja proviene de darle un periodo de espera SIFS en lugar de un DIFS utilizado por las demás estaciones en el procedimiento BEB. Extra a este cambio, es necesario implementar un algoritmo para ordenar los paquetes que el AP desea transmitir en base a la región a la que corresponden, esto porque el *AP burst* finaliza hasta que una región completa ha sido atendida.

Una vez explicado el funcionamiento del método propuesto, se prosiguió a realizar simulaciones que demuestren la efectividad de la técnica *RegionDCF*. Los resultados finales están basados en simulaciones con el software NS2 y se compararon los resultados obtenidos de dicha técnica frente a los obtenidos con el estándar IEEE 802.11b/g. Los criterios evaluados fueron: rendimiento (medido en bytes/seg), tiempos de espera, utilización del canal, número de colisiones y retardos. Dichos elementos reflejarán directamente la eficiencia real de la red.

Las pruebas realizadas fueron para una red espacial de 200x200 metros con un AP en el centro y con 20 terminales distribuidas aleatoriamente con paquetes de una aplicación FTP deseando ser transmitidos todo el tiempo (alta congestión). TCP se encargaría de fragmentar los paquetes FTP que serían enviados a una terminal conectada a una subred perteneciente al AP para hacer las pruebas más realistas.

En la Tabla 1 del documento citado, se demuestran mejoras en todos los criterios analizados. Dichos datos corresponden a una simulación con paquetes FTP de 512 bytes. Y en la Figura 2.3 se demuestra una notable mejora en el rendimiento promedio de la red.

[22] Los slots definidos entre TOp son de duración SIFS.

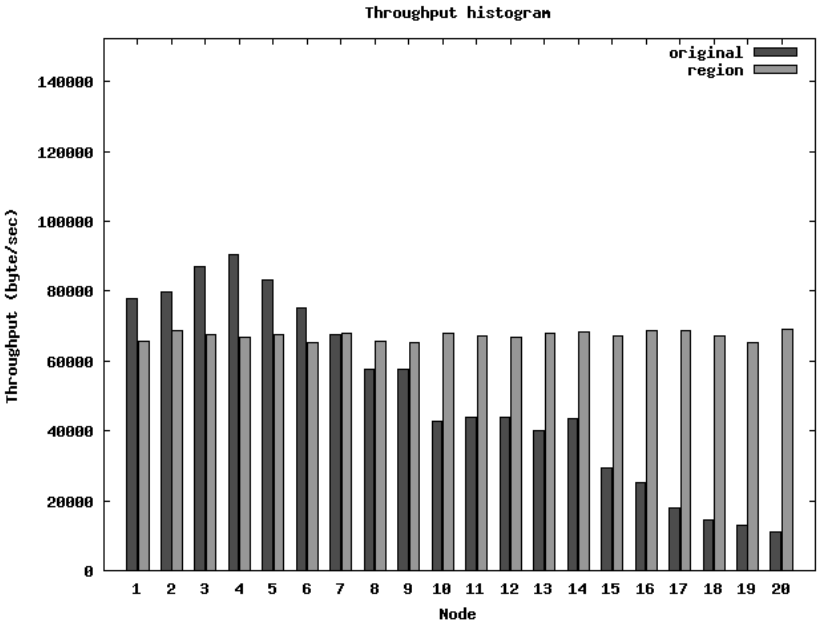


Figura 2.3 - Rendimiento individual para 20 terminales en regiones de 5 miembros. [23]

Como se aprecia en la gráfica, *RegionDCF* muestra una mejor distribución en la utilización del canal y demuestra ser más 'justo' que el estándar por sí solo. En el original, algunas estaciones acaparan gran parte del canal mientras que otras se quedan con poca capacidad de enviar sus mensajes. En *RegionDCF* las veinte estaciones presentan un buen valor distribuido de rendimiento (en bytes/seg). Según los análisis presentados por los autores, *RegionDCF* funciona mejor con regiones de mayor número de miembros, logrando una mejora de hasta el 81.26% en el rendimiento de la red, con paquetes de 64 bytes en regiones con 20 estaciones.

En conclusión, *RegionDCF* demostró ser una novedosa técnica con mejoras en el rendimiento de las redes inalámbricas frente al estándar IEEE 802.11b/g; se disminuyen los tiempos de contienda y los retardos, mejorándose así la utilización del canal. El resultado final es un aumento en el rendimiento neto de la red. La técnica propuesta reduce los retardos de contención en situaciones de alta congestión de tráfico lo cual nos lleva a una mejora real en la eficiencia de la WLAN.

Las posibles desventajas recaen en la forma en que se definen las regiones, las pruebas no especifican como se debe actuar frente a terminales móviles que cambian de una región a otra y donde pueden existir perdidas parciales o totales de transmisiones en curso. Otra desventaja es la necesidad del AP para definir las regiones, de forma que se aumenta la carga de trabajo a la unidad central y puede aumentar la complejidad del mecanismo.

[23] Derechos al autor. Figura 5 del documento citado en [37].

III - PROPUESTA.

3.1 Planteamiento inicial.

Dado que nos enfrentamos a un problema de optimización, cuyo principal objetivo es el de aumentar el rendimiento de la red al escoger valores CWmin y CWmax lo más apropiados posible frente a situaciones cambiantes de la misma; podríamos entonces implementar un algoritmo genético que evalúe las condiciones de la red y sea capaz de arrojar un resultado de configuración BEB idóneo que maximice, en la medida de lo posible, el rendimiento a nivel de capa de enlace; esto se verá reflejado en tiempos de espera por contención menores al igual que un número disminuido de colisiones.

Se propone entonces, el desarrollo de un programa genético en plataforma Java, que sea capaz de simular diferentes escenarios en una red inalámbrica IEEE 802.11g con N estaciones contendientes y arrojar valores sugeridos de BEB (*CWmin y mStages*) que maximicen la utilización del canal. El programa genético realizará una búsqueda inteligente en un campo de soluciones amplio y que garantizará mejorar el rendimiento de la red frente a una configuración BEB estática (como lo es el caso del estándar IEEE 802.11g). La calidad de los resultados arrojados estará en función de la información proporcionada por el usuario[24] respecto a las condiciones de la red a simular y a la información pertinente al desarrollo del algoritmo genético, como son, tamaño de población y criterio de terminación.

El programa genético será capaz de simular una WLAN con N estaciones contendientes que poseerán siempre un paquete deseando ser transmitido (alta congestión) y que evaluará el rendimiento promedio de la red en función de los paquetes entregados satisfactoriamente (bits) entre el tiempo total consumido (segundos). Este valor corresponde a la tasa de transmisión real por nodo conocida popularmente como el *'Throughput'*[25]. Dicha función será utilizada entonces para cada par de valores arrojados CWmin y mStages sugeridos por la búsqueda genética de cada generación hasta llegar a un criterio de terminación (punto de convergencia deseado).

El usuario especificará el criterio de terminación que desee para detener la iteración de generaciones y el tamaño de la población del algoritmo genético para ampliar o reducir el campo de búsqueda. De la misma manera, el programa necesitará partir de valores iniciales que producirán la generación cero. Estos valores pueden ser los pertinentes al estándar IEEE 802.11g o los que el usuario considere pertinentes.

Se busca que el programa sea diseñado como una aplicación en Java con una interfaz gráfica de usuario amigable (GUI por sus siglas en inglés) que colecte de forma sencilla datos de entrada para generar valores sugeridos de salida. Los datos de salida consisten en el par de valores escogidos para la ventana de contención CW y que, según la búsqueda inteligente y la función de evaluación

[24] Se entiende que el usuario que utilice dicho programa genético, conozca las características y condiciones de la red para mejorar así su rendimiento.

[25] Throughput: Es la tasa promedio de paquetes entregados satisfactoriamente a través de un canal de comunicaciones en un periodo de tiempo medido.

del rendimiento de la WLAN, son los valores óptimos que permitirán a la red inalámbrica alcanzar su máximo rendimiento con las N estaciones contendientes.

Dicho esto, se asume que los 'individuos' de la población de tamaño M serán evaluados, mutados e iterados durante un número fijo de generaciones especificado en la información del criterio de terminación siempre conservando al más fuerte de la generación anterior (elitismo). Al final, el programa arrojará los valores correspondientes al individuo más fuerte de la última generación.

Un individuo del programa genético está conformado por el par de valores CWmin y mStages, ambos enteros positivos mayores a cero. Las operaciones genéticas se llevarán a cabo con estas cantidades numéricas de modo que sean evaluadas el mayor número de combinaciones posibles y se logré obtener el par más adecuado con el fin siempre, de maximizar el rendimiento de la red.

Los individuos de la población serán mucho más afines entre más elevado sea el valor de 'throughput' que arrojen y contarán entonces, con mejores posibilidades de ser elegidos y así perdurar entre generaciones.

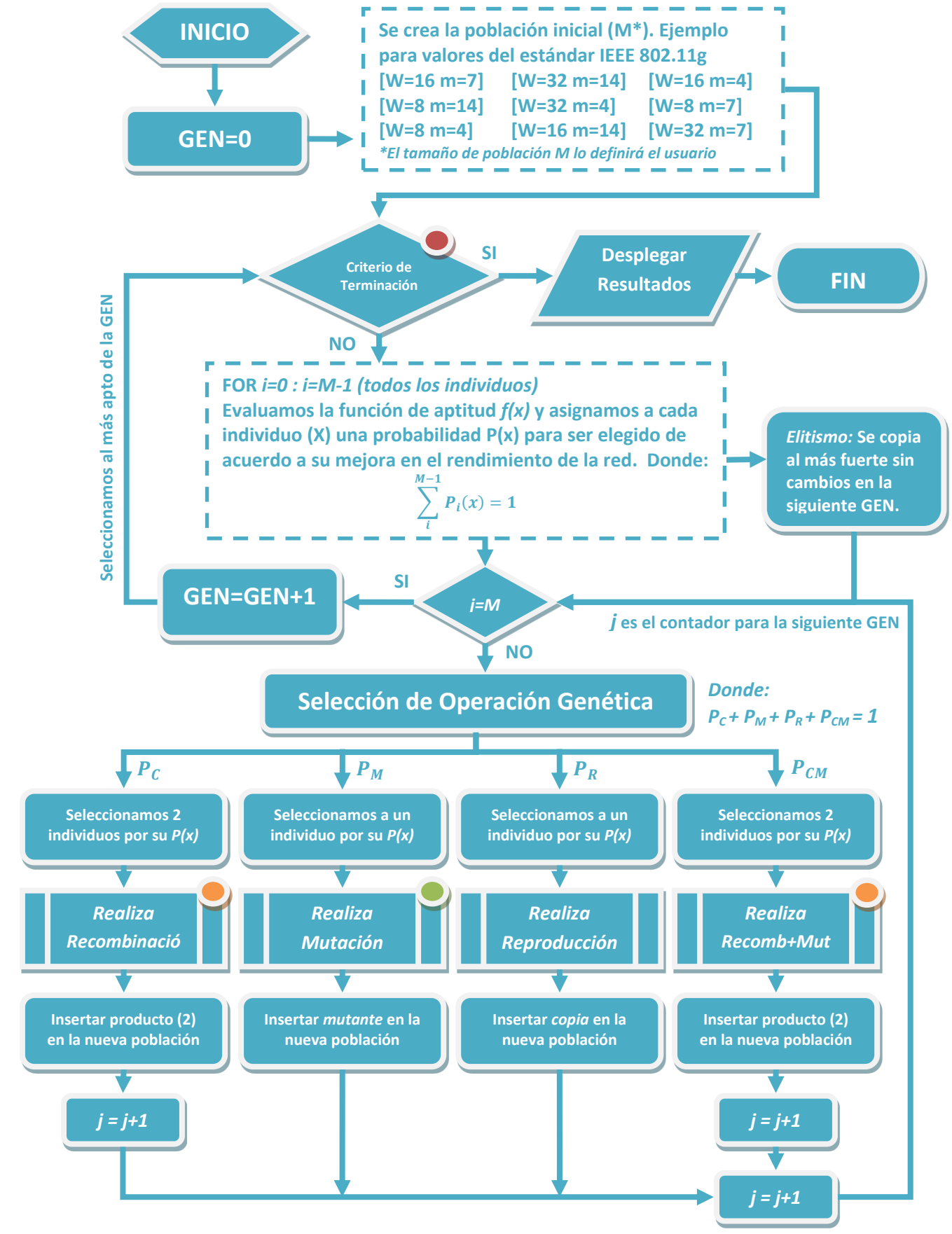
Los individuos que arrojen un valor de rendimiento de red muy bajo, esto es, valores de CWmin y mStages que aumenten el tiempo de espera para transmitir y/o generen un gran número de colisiones en la simulación; serán rápidamente desechados o mutados de tal forma que mejoren el *'throughput'* simulado.

A continuación se presenta el trabajo realizado para el desarrollo del programa genético que buscará la mejor configuración BEB en una red IEEE 802.11g de N estaciones contendientes. Primeramente se explica a detalle el algoritmo y después se presenta el pseudocódigo que nos llevará más tarde a la programación formal en Java.

A partir de este momento, se hará referencia al programa genético presentado en esta tesis como la **GBEBapp** (*Genetic Binary Exponential Backoff Application*) tratándose de una aplicación en Java que encuentra valores óptimos de BEB por medio de una búsqueda inteligente por medio de la resolución de un algoritmo genético.

3.2 Algoritmo.

En la siguiente página se presenta el algoritmo diseñado para nuestro programa GBEBapp que pretende encontrar la mejor configuración BEB en redes inalámbricas IEEE 802.11g que posean N estaciones contendiendo por acceso al canal en condiciones de alta congestión.



El criterio de terminación puede definirse con un # de generaciones o a un valor óptimo de aptitud alcanzado.

Si tenemos j=M-1, las operaciones Recombinación o Recomb+Muta no pueden ser elegidas.

La operación de Mutación no debe arrojar valores cero o negativos para W o para mStages.

El propósito de este algoritmo es encontrar la mejor configuración para valores de *Wo* y *mStages* que prometen mejorar el rendimiento en una red de N terminales. Partimos de una generación cero con múltiplos de los valores iniciales proporcionados por el usuario, por ejemplo valores típicos del mecanismo MAC en el estándar IEEE 802.11g. Seguido a esto, se pregunta si se satisface o no un criterio de terminación, de no ser así se evalúan los individuos y se aplican las operaciones genéticas para transformar a la población en una nueva generación. El tamaño de la población M será escogido por el usuario para hacer más estricta o más liviana la búsqueda.

Los GA (algoritmos genéticos) trabajan con un espacio de búsqueda de M elementos dentro del cual se encuentran las mejores soluciones. La idea es producir nuevas generaciones de soluciones candidatas a través de operaciones genéticas que podrían mejorar la calidad y la aptitud de cada individuo. Análogo al proceso Darwiniano de selección natural, estas operaciones genéticas desarrollarán una nueva generación más fuerte y más apta, mientras se garantice la conservación de los mejores genes y la creación de nuevas especies por recombinación de dos o más individuos. Como en la naturaleza, las operaciones genéticas son: la recombinación, la mutación, la reproducción y una operación en pareja llamada recombinación + mutación (evolución). Para cada una de estas operaciones se necesita uno o dos individuos de la población actual que serán seleccionados de acuerdo con su probabilidad P(x) que nos indica quienes son los más fuertes.

La población será mejorada por cada iteración en la que se evaluará la función de aptitud por individuo y se llevarán a cabo las operaciones genéticas seleccionadas con el fin de conseguir una nueva generación. En este caso, la población es un conjunto de soluciones a un problema de optimización, y así, al obtener la mejor generación, nos conduce de la misma manera, a la probable mejor solución a nuestro problema.

Para encontrar la mejor configuración BEB en redes IEEE 802.11g debemos encontrar una función de aptitud adecuada. En nuestro caso vamos a evaluar cada solución individual (W, mStages) de la población por su mejora efectiva en el rendimiento de la red. Con esta evaluación por medio de la función de aptitud, se asignará a cada individuo una probabilidad de ser elegido para una operación genética más adelante. A los individuos evaluados y asignados con una probabilidad P(x) le llamamos el espacio de evaluados. La suma de las probabilidades del espacio de evaluados (E) debe de ser uno.

$$Población\ (espacio\ de\ busqueda) \rightarrow\ S = \{X_o, X_1, X_2, \dots, X_{M-1}\}$$

$$Individuo \rightarrow X = [W, m]\quad donde: W, m \in \mathbb{Z}^+ = \{1,2,3, \dots, \infty\}$$

$$Función\ de\ aptitud \rightarrow f(X) = \max(RENDIMIENTOred[W, m])$$

$$Espacio\ de\ Evaluados \rightarrow E = \{P_0(X), P_1(X), P_2(X), \dots, P_{M-1}(X)\}$$

$$De\ donde: \sum_{i=0}^{M-1} P_i(X) = 1$$

Garantizamos la supervivencia del individuo más apto a través de cada generación con la estrategia conocida como selección *elitista*. Aunque la probabilidad de ser elegido para ser reproducido (sin modificaciones) es alta para los más aptos; hay un punto particular en el algoritmo en el que el más apto de la generación actual se copia en la nueva generación antes de que todas las operaciones genéticas se lleven a cabo. De esta forma, las operaciones genéticas se producirán en los M individuos de la generación actual, donde ya se incluye al más fuerte, y así obtenemos la próxima generación.

Cada operación genética tendrá su propia probabilidad de ser elegida, es decir, P_M es la probabilidad de seleccionar la operación de mutación. Las operaciones de reproducción y recomb+muta (evolución) son más propensas a tener una alta probabilidad de ser escogidas, ya que éstos son los principios fundamentales de la selección natural. Explicamos ahora cómo se lleva a cabo cada operación:

Recombinación. Dos padres son seleccionados y se recombinan sus cromosomas para la creación de dos nuevos individuos (descendencia), que pueden incluso ser una copia exacta de los padres, si los padres son idénticos entre sí. Una parte del padre, llamado el fragmento de cruce, se combina con un resto de la madre. La descendencia debe tener la misma longitud de configuración cromosomática que los padres. El punto de cruce (donde el fragmento de cruce del padre termina y comienza el resto de la madre) debe ser uniformemente seleccionado al azar. Dado que sólo tenemos dos elementos en nuestras cadenas de cromosomas (W y mStages), el cruce será tan simple como el intercambio de estos dos parámetros en la descendencia.

Ejemplo de Recombinación:

$$Recombinación\ (X_0 = [W_0, m_0]\ , X_1 = [W_1, m_1]) \rightarrow hijos:\ X_0' = [W_0, m_1] \& X_1' = [W_1, m_0]$$

Mutación. La mutación es un pequeño cambio aleatorio en los cromosomas de un individuo. Este cambio único en el individuo es poco probable que suceda por sí mismo como un evento aislado, es mucho más probable que ocurra cuando se desarrolla una operación de evolución (recombinación + mutación). El producto (mutante) es una copia exacta del individuo seleccionado con un cambio en su configuración cromosomática. Es posible más de un solo cambio. El punto de mutación debe seleccionarse de manera uniformemente aleatoria. En nuestro caso particular, sólo hay dos posibles puntos de mutación; los valores de *W* o *mStages*.

Ejemplo de Mutación:

$$Mutación\ (X_0 = [W_0, m_0])\ \rightarrow\ mutante: X_0' = [W_0 \pm A, m_0 \pm B]$$

$$Donde:\ A, B\ \in \{0,1,2,3,4,5\}$$

Limitamos el rango de valores para la mutación (hasta un valor de 5) para prevenir mutaciones invertidas que nos conducen a un fenómeno conocido como *convergencia prematura*.[26]

Valores de cero o negativos de W y mStages no son válidos para crear un nuevo individuo en la población. Si la operación de mutación produce valores negativos o de cero en W o mStages, una nueva iteración de la operación mutación deberá tener lugar hasta obtener valores enteros positivos mayores a cero.

Reproducción. La reproducción es una copia idéntica que se crea a partir de un individuo seleccionado. Dado que éste individuo se selecciona de acuerdo a su aptitud, medida en el espacio de evaluados conforme a su probabilidad P(x), logramos entonces una forma particular para garantizar la supervivencia de los más fuertes en cada generación (selección natural). La operación de Reproducción deberá tener una alta probabilidad de llevarse a cabo para garantizar un óptimo desempeño de nuestro algoritmo genético.

Ejemplo de Reproducción:

$$Reproducción\ (X_0 = [W_0, m_0])\ \rightarrow\ copia : X_0' = [W_0, m_0]$$

Recombinación+Mutación (Evolución). Esta es una operación en conjunto que opera sobre dos individuos de la población (los padres) en la que una recombinación ocurre primero y luego una mutación será ejecutada en la descendencia. Esto garantiza una evolución en las próximas generaciones.

Ejemplo de Recombinación+Mutación:

$$Recombinación\ (X_0 = [W_0, m_0]\ , X_1 = [W_1, m_1]) \rightarrow X_0' = [W_0, m_1] \& X_1' = [W_1, m_0]$$

$$Mutación\ (X_0' = [W_0, m_1], X_1' = [W_1, m_0])$$

$$descendencia : X_0'' = [W_0 \pm A, m_1 \pm B]\ \&\ X_1'' = [W_1 \pm C, m_0 \pm D]$$

$$Donde:\ A, B, C, D\ \in \{0,1,2,3,4,5\}$$

3.3 Pseudocódigo.

A partir del algoritmo para nuestra GBEBapp y en base a los primeros análisis pertinentes a la implementación de nuestro programa, se presenta a continuación el pseudocódigo para su implementación en Java así como una explicación general del funcionamiento del mismo.

[26] Convergencia de una búsqueda inteligente antes de un tiempo óptimo que arrojará soluciones de baja calidad que perjudicará nuestros resultados globales.

PSEUDOCÓDIGO Y CLASES PRINCIPALES DEL PROGRAMA EN JAVA "GBEBapp"

***public class* geneticBEB**

Datos de Entrada | **Datos de Salida**

SOLVE | **CLEAR** | **EXIT**

- ✓ Reinicia variables globales.
- ✓ Limpia áreas de texto. [*textArea.setText("")*]
- ✓ Limpia datos de entrada y salida.

System.exit(0)

public class GABEB_Frame

static class GBEB

int Wo, mStages, tput; float selecP; //Objeto de clase GBEB.

***public* GBEB ()** //Constructor inicializa Wo, mStages, tput, selecP = 0;

***public* GBEB (int index)**

Crea la población inicial GEN 0 (Switch) de acuerdo a los Datos de Entrada (M, Wo, m). Regresa los datos iniciales de W y mStages para el elemento 'index' del arreglo.

Switch(M)

case 3 $[2W \mid \frac{1}{2}W, m]$

case 9 $[2W \mid \frac{1}{2}W, 2m \mid \frac{1}{2}m]$

case 15 $[2W|\frac{1}{2}W|4W|\frac{1}{4}W, 2m|\frac{1}{2}m]$

case 25 $[2W|\frac{1}{2}W|4W|\frac{1}{4}W, 2m|\frac{1}{2}m|4m|\frac{1}{4}m]$

```
int[] populationW, populationm = new int[M];
populationW[0]=W;
populationm[0]=m;
...
populationW[M-1]=n*W;
populationm[M-1]=n*m;          donde: n=1,2,4
```

***public* Throughput (GBEB obj)**

```
//N estaciones contendiendo en una red IEEE 802.11g
this.tput = (int) result;
// result depende del rendimiento por individuo (W,m).
```

***public* SelectProb (int sum)**

```
// Asignamos probabilidades P(x) a cada individuo en función de su previa evaluación con Throughput().
// sum es la suma total del rendimiento medido de todos los individuos.
this.selecP = (float) this.tput/sum;
// De esa forma estamos usando un modelo de selección 'roulette wheel'.
```

public GBEB[] Crossover (*GBEB a, GBEB b*)
return offspring

public GBEB Mutation (*GBEB a*)
return a2

public GBEB Reproduction (*GBEB a*)
return a2

public GBEB[] CrossoverMutation (*GBEB a, GBEB b*)
return offspring

Las operaciones genéticas están definidas como *métodos para la clase GBEB*. Cada operación requerirá de uno o dos individuos de la población actual (currentGen), estos son objetos GBEB; y de esta forma realizarán cambios que nos llevaran a armar la siguiente generación (nextGen). Los métodos están diseñados para regresar objetos iguales de la clase GBEB que son nuestros individuos a evaluar y son guardados en la siguiente población.

public void solveGBEBactionPerformed()

1) **INICIO de corrida.**
2) **GBEB[] currentGen = new GBEB [M];** //Creamos el arreglo currentGen
3) **FOR i:M-1 currentGen[i] = GBEB(i);** //Vaciamos W y m para la currentGen
4) **Termination Criteria? "no" ir al paso 5, "si" ir al paso 19**
5) **FOR i:M-1 currentGen[i] = Throughput();** //Evaluamos el *throughput* de cada individuo
6) **sum=$\sum_{i=0}^{M-1}$ currentGen[i]. tput;**
7) **FOR i:M-1 currentGen[i] = SelectProb(sum);** //Asignamos probabilidades P(x)
8) **GBEB[] nextGen = new GBEB [M];** //Preparamos la nueva generación
9) **nextGen[0] = MAX(currentGen.selecP);** //Elitismo
10) **j=1; j<M? "no" ir al paso 11, "si" ir al paso 16.**
11) **Función de probabilidad para seleccionar una Operación Genética.**
12) **Función de probabilidad para seleccionar uno/dos individuos para la operación genética.**
13) **Realizamos [Recombinación, Mutación, Reproducción o Recomb+Muta]**
14) **nextGen[j] = descendencia de la operación genética realizada;**
15) **j=j+1 en Mutación y Reproducción j=j+2 en Recombinación y Recomb+Muta. Ir al paso 10**
16) **GEN=GEN+1;** //Contador de generaciones
17) **currentGen=nextGen;** //Completamos el ciclo y usamos la siguiente generación
18) **Ir al paso 4.**
19) **Datos de salida. Fin de corrida.**

Notas

La sentencia Switch creará la población inicial en base a premultiplicativos doble, triple, cuádruple, etc. Depende de M.

Las operaciones genéticas se explican mejor en las siguientes páginas. Operan aritméticamente sobre los valores W y mStages de los individuos seleccionados.

La función de probabilidad para seleccionar una operación genética se explica en las siguientes páginas y depende de constantes para cada una en función de su utilidad.

La función de probabilidad para seleccionar uno o dos individuos se explica en las siguientes páginas y depende del valor *selecP* de cada uno en función de su aptitud.

La aplicación Java descrita en el pseudocódigo funciona con la clase GBEB dentro del botón *solveGBEB*, éste será el botón que desencadenará la solución del algoritmo en base a los datos de entrada. Se crean objetos de la clase GBEB, cada uno tiene cuatro atributos, los valores enteros de *Wo, mStages, tput* y el valor flotante para *selecP*. En orden, estos son los valores para: el intervalo inicial de la ventana de contención CWmin, el número de etapas para escalar de forma binaria la ventana hasta un CWmax, el rendimiento promedio medido en una simulación de red inalámbrica IEEE 802.11g con N estaciones y finalmente la probabilidad (de 0 a 1) de ser seleccionado para una operación genética de acuerdo con lo obtenido en la función de aptitud (*Throughput*), por lo tanto, cuanto mayor sea el rendimiento simulado por pareja de valores W y mStages, mayor será el valor *selecP*. Esto se conoce como un modelo de selección de ruleta (*roulette wheel model*).[27]

A continuación, cada generación (GEN) sobre una corrida (RUN) de la aplicación Java creará una matriz tamaño M de objetos GBEB. M es el tamaño de la población elegida por el usuario en los datos de entrada y el constructor GBEB inicializará los cuatro atributos a cero de cada uno, más tarde el método GBEB asignará valores entero a W y mStages basados en lo propuesto por el usuario en los datos de entrada (es decir, la generación cero siempre será una población con individuos que son múltiplos o submúltiplos de los datos iniciales para Wo y mStages). La corrida empieza con la clase principal. En este punto, el primer paso es crear la población inicial o GEN 0. Llamamos "población" sólo al conjunto de parejas de valores W y mStages de cada objeto, ya que estos son los parámetros que evaluaremos del BEB para hallar nuestras posibles soluciones en el espacio de búsqueda (S). Esta población inicial se crea a partir de cuatro casos posibles en la sentencia *Switch()* en función del valor de M (tamaño de la población escogida por el usuario en los datos de entrada). Se puede hacer trabajo posterior con el fin de alcanzar poblaciones más amplias y determinar un límite superior efectivo.

La sentencia Switch creará la población inicial pata todas las posibles combinaciones no repetitivas de W y mStages dadas por dos factores premultiplicativos en la forma 1/n y n, donde n ∈ {1,2,4} Por lo tanto, la población se define como:

$$Población\ Inicial\ (M) = \left[\frac{1}{n}W \text{ ó } n * W\ , \frac{1}{k}m \text{ ó } k * m\right]$$

$$Donde: n, k\ \in \{1,2,4\}$$

Estas posibles combinaciones podrán generar poblaciones de tamaño 3, 9, 15 o 25. Para fines prácticos suponemos que una población de 25 será suficiente para evaluar de forma eficiente nuestro espacio de búsqueda (S), el cual se centrará en torno a los valores iniciales de W y mStages que son proporcionados por el usuario, estos valores iniciales por ejemplo, podrían ser los valores fijos del protocolo MAC en el estándar IEEE 802.11g con el fin de comparar los resultados de nuestro algoritmo genético con los obtenidos en una red inalámbrica sin cambio alguno en su ventana de contención.

[27] También conocido como '*fitness proportionate selection*' es un modelo para asignar probabilidades de selección a los individuos de una población en función directa de su aptitud o fortaleza. Su característica principal es que la suma de las probabilidades de todos los individuos es uno.

El siguiente paso del programa será preguntar si el criterio de terminación se ha cumplido, si es así, entonces el programa debe lanzar los datos de salida (resultados) y finalizar la corrida, caso contrario, entonces el proceso del algoritmo genético comenzará por sí mismo. Queremos 'evaluar' cada individuo de la generación actual de tamaño M (arreglo *currentGen*) para verificar la aptitud de cada posible solución. Esto se hará por el método de *Throughput* que está diseñado para simular una red inalámbrica IEEE 802.11g de N estaciones contendiendo por el canal y con alta congestión de tráfico. La simulación obedecerá la definición RTS-CTS que especifica el estándar IEEE para reducir las colisiones introducidas por el problema de la *terminal oculta* (explicado en el Capítulo 1). Vamos a utilizar una velocidad de transmisión cruda de 12Mbps. La simulación calculará entonces un rendimiento promedio (throughput) después de transmitir satisfactoriamente 10,000[28] paquetes MSDU de 1500 bytes. También se planea introducir un criterio para evitar el problema de 'inanición'[29] que sucede cuando una estación monopoliza el tráfico de la red. La simulación usará los valores iniciales W y mStages para calcular el rendimiento promedio de cada objeto (individuo) en la población (currentGen) y así producir un valor entero de rendimiento en kbps (*throughput)*. Esta información se almacenará en el atributo *tput* de cada individuo de la currentGen.

Ahora que cada individuo ha sido evaluado, es necesario asignarle su probabilidad de ser seleccionado para una operación genética de acuerdo a su mejora en el rendimiento de la red. Aquí el criterio de aptitud es tan simple como que el más alto rendimiento medido debe recibir la más alta probabilidad de selección. El programa utilizará el modelo de ruleta para asignar a cada individuo una probabilidad de selección, es decir, la suma de todas las probabilidades de selección de los M individuos en la generación debe ser la unidad. Estas probabilidades se obtienen a través de una sencilla relación, el rendimiento individual medido dividido entre la suma de los rendimientos de toda la currentGen (sum) invocando al método *SelectProb*. De esta forma garantizamos que entre más alto sea el rendimiento obtenido por un individuo, mayor será su probabilidad de selección. Se guardará entonces esta probabilidad en el atributo *selecP* de cada individuo de la población y se iterarán dichos cálculos hasta que la currentGen sea completamente recorrida. Ahora, cada objeto (individuo de la población) tiene sus cuatro atributos listos.

Ya que se ha evaluado completamente la currentGen, el siguiente paso es inicializar la próxima generación (*nextGen*) con una población del mismo tamaño M para luego recoger el individuo más fuerte de la currentGen y copiarlo en el nextGen sin ningún cambio (elitismo). Los M-1 individuos restantes se generarán a partir de operaciones genéticas hasta completar la nueva generación.

El programa ahora comenzará un bucle repetitivo hasta que llegue a *j = M*. Así, durante el ciclo, una operación genética tendrá lugar a la vez y nuevos individuos se copiarán en nextGen hasta que

[28] El valor de 10,000 paquetes se ha elegido en función de simulaciones similares realizadas en el trabajo de investigación en [38]. Del mismo modo, se justifica la utilización de no más de 10,000 simulaciones de contención para lo que puede soportar el programa en Java (tiempos de procesado muy altos).

[29] El fenómeno de inanición también conocido como '*starving problem'* en redes inalámbricas ocurre cuando una estación se apodera del canal para sus propias transmisiones dejando a las demás estaciones contendientes sin oportunidad de utilizarlo.

se copie el último. Primero, el programa escogerá una operación genética (OG) en función de las probabilidades asignadas. Cada OG tendrá su propia probabilidad de suceder en la población. Las operaciones de Reproducción y Recombinación+Mutación (evolución) son más idóneas de poseer una alta probabilidad, ya que éstos son los principios fundamentales de la selección natural. Las probabilidades para cada OG se establecerán como valores constantes dentro de la aplicación:

$$Recombinación \rightarrow P_C = 0.05$$
$$Mutación \rightarrow P_M = 0.05$$
$$Reproducción \rightarrow P_R = 0.45$$
$$Recombinación + Mutación \rightarrow P_{CM} = 0.45$$

Los sufijos C, M, R y CM corresponden a las iniciales de sus equivalentes en inglés: *Crossover, Mutation, Reproduction y Crossover+Mutation*. Trabajos de investigación sugieren valores de probabilidad bajos para las operaciones de Mutación y Recombinación, es por eso que les asignamos apenas un 5% de ocurrencia. [39]

Con el fin de seleccionar una OG para que se lleve a cabo, se realizó un sencillo algoritmo de selección aleatorio que utiliza la función de Java *Math.random* para garantizar una 'competencia' que le dé prioridad a las operaciones con mayor probabilidad. Dicho algoritmo se puede implementar fácilmente siguiendo la fracción de pseudocódigo siguiente:

```
     decision = (int) Math.round(100*Math.random());
     //se genera una decisión aleatoria del 0 al 100
     if (decision>=0 && decision<5)        GO=1;
     if (decision>=5 && decision<10)       GO=2;
     if (decision>=10 && decision<55)      GO=3;
     if (decision>=55 && decision<=100)    GO=4;
     switch(GO)
          case 1:    realiza Recombinación
          case 2:    realiza Mutación
          case 3:    realiza Reproducción
          case 4:    realiza Recombinación+Mutación
     end switch
```

Cada OG se declara como un método independiente de uno o dos argumentos (objetos de la clase GBEB). Y su función es la de devolver uno o dos objetos con diferentes valores para W y mStages (descendencia). Los individuos devueltos se copian en nextGen. La única OG que no cambia ningún atributo (valores de W y mStages) es la operación de Reproducción. La OG de Mutación está diseñado para hacer un solo cambio (en W ó en mStages) y no se le permite regresar valores negativos o de cero para los mismos. El pseudocódigo para las cuatro OG es el siguiente:

```
//Operación genética RECOMBINACIÓN
public GBEB[] Crossover (GBEB a, GBEB b)
     GBEB offspring = new GBEB[2];
     offspring[0].Wo = a.Wo;
     offspring[0].mStages = b.mStages;
     offspring[1].Wo = b.Wo;
     offspring[1].mStages = a.mStages;
return offspring;
```

```
//Operación genética MUTACIÓN
public GBEB Mutation (GBEB a)
     a2 = new GBEB();
     n = (int) Math.round(5*Math.random());   //Mutación
±{0,1,2,3,4,5}
     decision = (int) Math.round(100*Math.random());
     //Decisión aleatoria para cuatro tipos distintos de Mutación
     if (decision>=0 && decision<25)  a2.Wo=a.Wo+n;
     a2.mStages=a.mStages;
     if (decision>=25 && decision<50)a2.Wo=a.Wo-n;
     a2.mStages=a.mStages;
     if (decision>=50 && decision<75)a2.Wo=a.Wo;
a2.mStages=a.mStages+n;
     if (decision>=75 && decision<=100) a2.Wo=a.Wo;
a2.mStages=a.mStages-n;
     if (a2.Wo<=0)   a2.Wo=1;        if (a2.mStages<=0)
     a2.mStages=1;
return a2;
```

```
//Operación genética REPRODUCCIÓN
public GBEB Reproduction (GBEB a)
     a2 = new GBEB();
     a2.Wo = a.Wo;
     a2.mStages = a.mStages;
/* asegurando la supervivencia de los más fuertes al copiarlos sin
cambios en la siguiente generación */
return a2;
```

```
//Operación genética RECOMBINACIÓN+MUTACIÓN
public GBEB[] CrossoverMutation (GBEB a, GBEB b)
     GBEB offspring = new GBEB[2];
     offspring[0].Wo = a.Wo;
     offspring[0].mStages = b.mStages;
     offspring[1].Wo = b.Wo;
     offspring[1].mStages = a.mStages;
return Mutation(offspring[0]), Mutation(offspring[1]);
//Evolución
```

Como podemos notar, cada OG realizará operaciones aritméticas simples en los objetos seleccionados para generar nuevos objetos de la misma clase (individuos para la nextGen). Pero, ¿cómo se seleccionarán estos individuos en primer lugar? La respuesta es tan simple como una función de selección probabilística (paso 12 de nuestro pseudocódigo). Debido a que cada individuo de la currentGen tiene una probabilidad de ser elegido (selecP), y sabemos que la suma de estas probabilidades es la unidad (100%), tenemos que crear entonces una función que respete las probabilidades y que tenga una preferencia para recoger a los individuos más aptos (que tienen un valor de *selecP* mayor) obedeciendo al modelo de 'selección de ruleta' como se plantea en [40].

El pseudocódigo para la ejecución de la función de selección probabilística es el siguiente:

```
public GBEB Selection (GBEB[] a)
     //Recibirá como argumento el arreglo de la población
currentGen
     int selection = 0;
     float lowLim = 0, upLim = 0;
     selec = (float) Math.random();
     for (i=0;i<M;i++)
          upLim = a[i].selecP+lowLim;
          if (selec>=lowLim && selec<upLim)  selection = i;
          lowLim = a[i].selecP;
     end for
     //devolverá sólo un individuo de la currentGen
return a[selection];
```

De esta manera, la función de selección probabilística debe ser invocada cada vez que una OG solicita uno o dos individuos para operar, (la función de selección probabilística será invocada dos veces en el caso de Recombinación y Recombinación+Mutación). Un ejemplo de cómo implementar los pasos 10 al 15 del pseudocódigo para nuestro programa genético esta dado por las siguientes líneas:

```
for (j=1; j<M; j++)
     decision = (int) Math.round(100*Math.random());
     if ((j<M-1) && decision>=0 && decision<5)            GO=1;
     if (decision>=5 && decision<10)                      GO=2;
     if (decision>=10 && decision<55)                     GO=3;
     if ((j<M-1) && decision>=55 && decision<=100)        GO=4;
     switch(GO)
          case 1:
          GBEB nextGen[j],nextGen[j+1] = (GBEB)

     Crossover(Selection(currentGen),Selection(currentGen)).clone();
          j++;
          case 2:
          GBEB nextGen[j] = (GBEB)
Mutation(Selection(currentGen)).clone();
          case 3:
          GBEB nextGen[j] = (GBEB)
Reproduction(Selection(currentGen)).clone();
          case 4:
          GBEB nextGen[j],nextGen[j+1] = (GBEB)

     CrossoverMutation(Selection(currentGen),Selection(currentGen)).clon
e();
          j++;
     end switch
end for
```

Cabe señalar que se ha añadido una condición adicional en la instrucción *IF* para las operaciones *Crossover* y *CrossoverMutation* esto para ser coherentes con el tamaño de la población M. Estas dos OG producen una descendencia que consta de dos individuos, de modo que, sólo se podrá

realizar cuando el tamaño de la nextGen sea menor que M-1. Tengamos en cuenta también que, dado que no hay ningún operador en Java para duplicar un objeto, se utiliza el método *'clone'* para copiar el contenido de los resultados arrojados de las OG al arreglo nextGen. Debemos recordar que es necesario actualizar el valor del índice contador *j* en las operaciones que dan lugar a una descendencia de dos individuos.

El bucle para las operaciones genéticas continuará hasta que la población nextGen alcance el elemento M y la nueva generación esté lista para ser evaluada. Actualizamos nuestro contador GEN (paso 16 del pseudocódigo) para saber cuántas generaciones han sido creadas, éste valor se puede utilizar más adelante para el criterio de terminación y forma parte de los datos de salida.

El siguiente paso consiste en sobrescribir la población currentGen con los individuos generados en nextGen (debemos utilizar el método clone) con el fin de estar preparados para las evaluaciones de rendimiento y la asignación de las probabilidades de selección (en este punto, si no se ha cumplido el criterio de terminación, estamos repitiendo todo el proceso ahora una generación adelante). Comenzamos de nuevo desde el paso 4 del pseudocódigo. Vamos a repetir este gran bucle (pasos 5 al 17 de nuestro pseudocódigo) hasta que el criterio de terminación se cumpla. El criterio de terminación será especificado por el usuario en los datos de entrada antes de correr el programa.

Hay dos formas posibles para especificar el criterio de terminación, la primera es cuando se llega a un cierto número de generaciones (con ayuda del contador GEN) y la segunda es cuando el programa detecta que el individuo más apto de las últimas generaciones no cambia más allá de un pequeño porcentaje (convergencia).

Finalmente estamos listos para mostrar los datos de salida y terminar la corrida.

Esperamos que a partir de una corrida de dicha aplicación GBEBapp se puedan arrojar los mejores valores de configuración (W y mStages) para el mecanismo BEB y que será finalmente el individuo más apto de la corrida ó lo que es lo mismo, el individuo más fuerte de la última generación. Dichos valores serán arrojados con el fin de lograr una mejora visible en el rendimiento de la red inalámbrica de N terminales. El algoritmo genético pondrá en marcha una búsqueda inteligente sobre todas las soluciones posibles (campo de búsqueda S) hasta encontrar la mejor y que ésta garantice mejoras en el desempeño siempre por arriba de el rendimiento obtenido con los valores estáticos para el BEB del estándar IEEE 802.11g.

En los siguientes capítulos se hace la implementación de la GBEBapp en Java y se realizan pruebas pertinentes para corroborar nuestras hipótesis.

IV - APLICACIÓN EN JAVA (GBEBapp).

4.1 Funcionamiento. Datos de entrada y de salida.

En el presente capítulo se explicará el funcionamiento de la aplicación desarrollada en Java que llamamos GBEBapp. Dicha aplicación obedece al algoritmo y el pseudocódigo desarrollados en el Capítulo 3. La aplicación está diseñada con las ventajas que obtenemos gracias a la programación orientada a objetos (POO). Los objetos que trabajarán dentro de las clases diseñadas emularan un escenario de contienda por el canal en una red inalámbrica IEEE 802.11g.

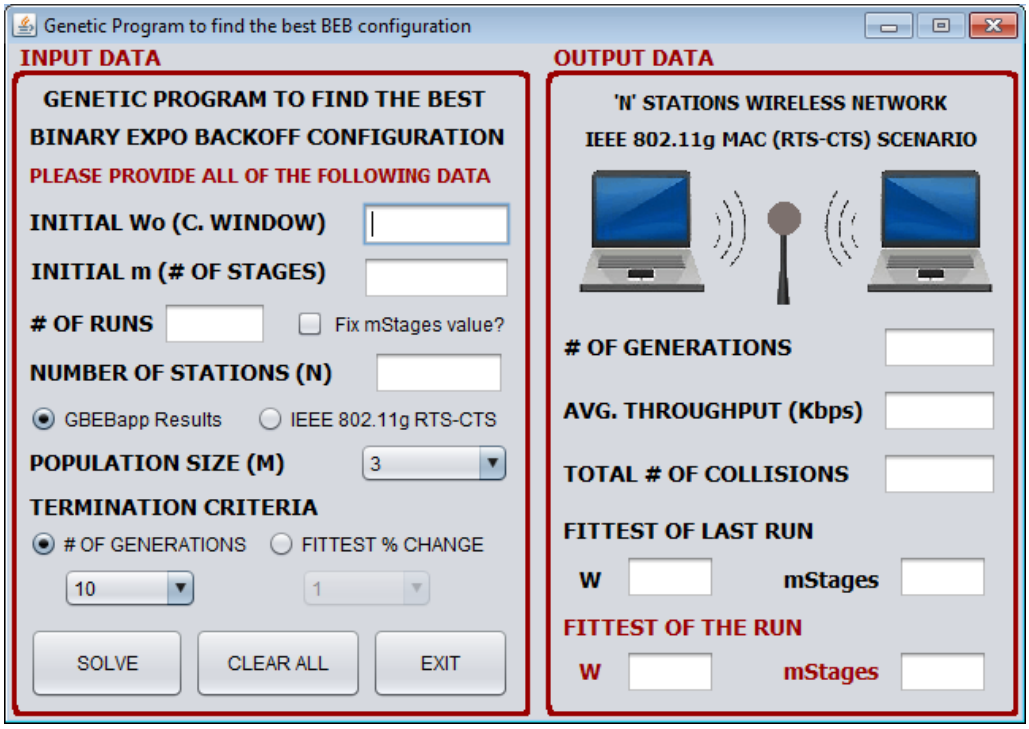


Figura 4.1 - Interfaz gráfica de usuario para la aplicación GBEBapp.

El programa genético para encontrar la mejor configuración de *Binary Exponential Backoff* (GBEBapp) es una aplicación gráfica en Java (Figura 4.1) que pedirá valores iniciales de Wo y mStages para una red inalámbrica de N estaciones operando bajo el estándar IEEE 802.11g con RTS-CTS activado. Hemos elegido el modelo RTS-CTS para hacer frente al problema de la terminal oculta. La aplicación va a realizar una simulación para evaluar el rendimiento real (en kbps) de las múltiples posibles soluciones (valores variados de W y mStages) en el escenario descrito donde todas las estaciones tienen siempre un paquete pendiente para transmitir. Así, mediante la aplicación del algoritmo genético diseñado, se realizarán operaciones genéticas a través de múltiples generaciones de M individuos; siempre con la intención de mejorar la población del espacio de búsqueda (S). El programa iterará múltiples generaciones de M individuos y evaluará a cada uno con una función de aptitud (f) hasta que un criterio de terminación se cumpla. Los criterios de terminación se deben centrar en la obtención de la mejor solución para el problema. También, el usuario debe especificar el tamaño de la población (M) de todas las generaciones a

iterar, así como los criterios de terminación deseados. El usuario proporcionará los datos necesarios para la simulación mediante los datos de entrada. Los resultados del programa genético se mostrarán en la sección de datos de salida.

A continuación se explican los datos de entrada necesarios para la correcta ejecución de la aplicación y los datos de salida que arrojará la misma. Esta información será analizada a fondo durante el desarrollo de los capítulos posteriores.

DATOS DE ENTRADA (Input Data).

El usuario debe proporcionar toda la información solicitada en la sección de Datos de Entrada. Primeramente los datos iniciales necesarios de Wo y mStages, que pueden ser por ejemplo, la configuración predeterminada del estándar IEEE 802.11g para CWmin y CWmax que son Wo = 16 y mStages = 6. Esta sugerencia es para que el programa parta de una configuración inicial real y busque la forma de mejorarla; lo cual es nuestro objetivo principal.

Además, el usuario escogerá valores para las variables del programa genético, valores como el tamaño de la población (M) y el criterio de terminación. El tamaño de la población está fijo a cuatro valores posibles (3, 9, 15, 25) y esto determinará la amplitud del espacio de búsqueda, se ha determinado que un tamaño de población controlado arrojará mejores resultados, justificación encontrada en [41]. El criterio de terminación puede ser elegido de entre dos opciones diferentes; la primera alcanzando un número fijo de generaciones o la segunda cuando se logre un porcentaje de cambio en el individuo más fuerte (convergencia [42]).

En los datos de entrada tenemos dos opciones extras que el usuario puede elegir dependiendo del tipo de corrida que se desee. Primeramente se cuenta con la opción de dejar fijo el valor de mStages durante todo el desarrollo del programa, esta opción nos servirá para ver los efectos que tiene el valor de CWmax en la solución de nuestro algoritmo. Y la segunda opción es la de una corrida con el GBEB activado o con el estándar IEEE 802.11g fijo, es decir, podemos utilizar el programa para simular una red inalámbrica de N terminales que operan bajo las características del estándar IEEE 802.11g con RTS-CTS sin modificaciones; dicha opción es para efectos de obtener datos que se usarán en gráficas comparativas.

Los valores para Wo y mStages iniciales deben ser enteros positivos. La aplicación tiene funciones de validación incorporadas y avisará al usuario si los datos de entrada son incorrectos.

Tenemos tres botones para la interfaz gráfica de la aplicación: SOLVE, CLEAR ALL y EXIT. En ese orden, el primer botón resolverá el algoritmo genético programado con los datos de entrada y desplegara automáticamente los resultados en los datos de salida, el segundo limpia la información en pantalla y vacía los datos almacenados previamente, y el tercero sale de la aplicación (cierra GBEBapp). Cada uno comienza un método *actionPerformed* cuando se hace clic en él. Estos métodos se describen en la siguiente sección. Cuando se presiona el botón de SOLVE se validan todos los datos de entrada y de existir algún error, notificará al usuario.

DATOS DE SALIDA (Output Data).

En los Datos de Salida encontraremos el número de generaciones totales de la corrida (una corrida es una ejecución completa de la aplicación mediante el botón de SOLVE), encontraremos también el rendimiento promedio (en kbps) alcanzado por los individuos más fuertes de la corrida. Debajo de estos datos, encontraremos las soluciones encontradas en la corrida; estos son, los valores de W y mStages del individuo más fuerte y los mismos valores del individuo más fuerte de la corrida anterior (para propósitos de comparación entre ellos). El objetivo final del programa genético GBEBapp es encontrar al mejor individuo (solución) que logre mejorar el rendimiento de la red inalámbrica simulada, dicho individuo se conforma de los valores de W y mStages ó lo que es lo mismo, CWmin y CWmax.

La aplicación tiene también una función incorporada para exportar todas las soluciones iteradas (valores de W, mStages, rendimiento promedio en kbps y número de colisiones) a un archivo de texto CSV[30] (*OUTPUT.TXT*). Esto con el fin de permitir su uso en gráficas o manipulación con software de simulación de redes para comparar los resultados obtenidos.

4.1.1 Contenido. Descripción detallada del programa en Java GBEBapp.

En esta sección se describirán de forma detallada las principales clases y métodos utilizados dentro de la programación Java que conforman nuestro programa genético GBEBapp.

public class GABEB_Frame extends javax.swing.JFrame

Clase principal del proyecto. Esta clase tiene dos métodos que crearán toda la interfaz gráfica de usuario para nuestra aplicación Java.

public GABEB_Frame ()

Constructor. Llama al método initComponents.

initComponents private void ()

Más de 300 líneas de código generadas automáticamente por el software NetBeans IDE que se encargan de crear toda la interfaz gráfica para nuestro programa GBEBapp. Este método contiene múltiples llamadas a objetos de la clase *javax.swing* que se usan para aplicaciones gráficas en Java.

private void ExitActionPerformed.

Método llamado desde el botón EXIT en la aplicación. Esta es una salida inmediata de la aplicación con cero segundos de retraso.

[30] CSV: Comma-Separated Values. Archivo de texto especial con datos de formato similar separados por comas. Suelen utilizarse cuando existen múltiples datos de salida que desean utilizarse de forma automática en otras aplicaciones.

private void ClearAllActionPerformed.

Método llamado desde el botón CLEAR ALL en la aplicación. Este método limpiará todo el contenido en la interfaz gráfica de usuario. En él se establece una cadena "" (null) para ser colocada en cada área de texto y también devuelve todas las listas desplegables a sus valores por defecto. Esta es una forma rápida de iniciar una nueva corrida con diferentes datos de entrada para la simulación.

private void solveGBEBActionPerformed.

Método llamado por medio del botón SOLVE en la aplicación. Este es el método más importante de la aplicación GBEBapp pues es el que resolverá el algoritmo genético, es aquí donde todo el programa genético se llevará a cabo por sí mismo. Aquí se encuentra la mayor parte de la validación que contiene la aplicación para los datos de entrada (no valores nulos, evitar que se congele la aplicación, etc.).

Primero se recogen todos los datos de entrada ya validados proporcionados por el usuario y entonces se procederá a crear la generación cero mediante el método GBEB (que se explica en detalle más adelante). Después comenzará la iteración principal de la GBEBapp, dicho ciclo iterativo comenzará siempre preguntando si se ha cumplido el criterio de terminación de la corrida. Dentro de esta iteración vamos a realizar las operaciones genéticas sobre los individuos de la generación actual para desarrollar a los individuos de la próxima generación (la cual debe ser mejor que la anterior). Esto se repetirá hasta que el criterio de terminación se cumpla[31].

En este método se define la forma de guardar a cada individuo dentro de las generaciones. Cada población, la actual y la próxima, se guardará en una matriz de objetos especiales; así, cada generación se encuentra en una matriz de tamaño M (llamadas currentGen y nextGen respectivamente). Los individuos de dichas generaciones tienen cuatro parámetros al ser objetos de la clase GBEB.

Para las operaciones genéticas es necesario evaluar (a través de una función de aptitud) a toda la población M. Esto se hará mediante el método 'Throughput' que asignará un rendimiento promedio medido en kbps, para cada individuo de la población y después el método 'SelectProb' asignará a cada individuo una probabilidad de ser elegido en función de su aptitud (probabilidad de selección P(x)). Los métodos Throughput y SelectProb se describen a detalle más adelante.

También se incluye dentro de nuestro método *solveGBEB*, una función que asigna probabilidades a las cuatro operaciones genéticas para ser elegidas y que se lleven a cabo sobre la generación actual (currentGen). Sin embargo, es necesario un método de selección para recoger a uno o dos individuos de la generación actual, esto siempre en función de su probabilidad de selección, para ser usados en una operación genética determinada; dicho método se explica más adelante como 'Selection'.

[31] La forma para seleccionar el criterio de terminación más adecuado se explicará en la siguiente sección del presente capítulo.

Se incluyeron sentencias de validación especial en las operaciones de Crossover y CrossoverMutation para evitar rebasar el tamaño de la generación siguiente (nextGen) del valor actual M.

Al final del código pertinente al botón SOLVE encontramos las sentencias que escriben los datos de salida (W y mStages, con su rendimiento medio en kbps obtenido correspondiente) de cada serie consecutiva de la GBEBapp en el archivo de texto CSV (*OUTPUT.TXT*).

public class GBEB.

Esta clase creará los objetos GBEB (individuos de las poblaciones para cada generación). Cada objeto tiene cuatro parámetros, tres números enteros, *Wo, mStages* y *tput*, y un valor flotante *selecP*. Esta clase tiene que implementar *'Cloneable'* a fin de poder utilizar un método de clonación como se describe a continuación. La clase GBEB tiene tres métodos:

public GBEB ()

Constructor. Inicializa los cuatro parámetros del objeto GBEB a cero.

public GBEB (int index)

Constructor sobrecargado. Este método tiene la responsabilidad de crear la generación cero de tamaño M, a partir de los valores iniciales W y mStages dados en los datos de entrada por parte del usuario.

public Object clone ()

Método específico para clonar objetos. Dado que las clases Java no poseen un método de clonación por defecto, es necesario forzar un método clone () dentro de la clase GBEB con el fin de ser capaz de duplicar objetos GBEB[32].

public boolean terminationCrtieria(int GENs, int TPUT, int lastTPT)

Este método está diseñado para arrojar un valor verdadero ó falso (true/false) cuando el criterio de terminación es evaluado. La GBEBapp tiene dos opciones diferentes para seleccionar un criterio de terminación y el usuario es capaz de seleccionar la que mejor se adapte a sus propósitos. Las dos opciones son; alcanzar un cierto número de generaciones dentro de la GBEBapp ó iterar múltiples generaciones hasta que el individuo más apto de la generación actual muestre un cierto cambio porcentual respecto al individuo más apto de la generación pasada. Así, cuando el rendimiento promedio (en kbps) del individuo más apto de la generación actual, difiere no más de un cierto porcentaje (los valores a elegir son 1%, 0,5%, 0,1%, 0,05%, 0,01%, 0,005% y 0,001%) del valor obtenido por el individuo más apto de la generación pasada, entonces el criterio de terminación se cumple puesto que se logró una convergencia.

[32] No basta con una operación de asignación puesto que así sólo se copian las direcciones de los apuntadores de los atributos del objeto y no los valores como tal.

El método *terminationCriteria* necesita tres valores para poder emitir un valor booleano verdadero o falso, dichos indicadores son, el número de las generaciones actuales, el rendimiento actual (kbps) y el mejor rendimiento promedio anterior logrado (para poder comparar).

public class STA

Esta es una clase diseñada para definir un objeto especial llamado STA, emulando a una estación, para su uso en la simulación de la red inalámbrica que se utiliza en el método de 'Throughput'. Cada objeto STA (estación) tiene cuatro parámetros enteros para operar que son: un contador *exponential backoff* (expBoff), un contador de colisiones consecutivas (*cCollisions*), un contador de paquetes entregados con éxito (*data*) y un valor de rendimiento promedio en kbps (*tput*).

public STA ()

Constructor. Inicializa los cuatro parámetros del objeto STA a cero.

public int expBO (int w, int m, int col)

Método que genera aleatoriamente un contador de *exponential backoff* para ser asignado a un objeto STA. Aquí se especifica que el número para el *exponential backoff* será seleccionado uniformemente aleatorio de entre 0 a $w_o * 2^{\mathrm{col}} - 1$. Donde, 'col' es el número de colisiones consecutivas ocasionadas por el objeto. La ventana de contención tendrá un límite superior hasta llegar a un valor de mStages, es decir, si el individuo actual tiene W = 16 y mStages = 7, el valor del *exponential backoff* se seleccionará de [0 - 1023] si han ocurrido siete o más colisiones consecutivas.

public int Throughput(GBEB obj)

Este método es el corazón de la aplicación pues es el responsable de la simulación de la red inalámbrica con N estaciones contendiendo en un escenario IEEE 802.11g con RTS-CTS. Cada una de las N estaciones tendrá siempre un paquete listo para ser enviado y se simplificará el problema asumiendo un canal libre de interferencia y ruido. Por lo tanto, la única razón de pérdida de paquetes es por colisiones. La simulación iterará una contienda para transmitir en el canal inalámbrico hasta que se entreguen 10,000 paquetes satisfactoriamente (en suma total para las N estaciones)[33].

En primer lugar, se definen las duraciones para cada paquete de control necesario (*SLOTTIME, DIFS, SIFS, EIFS, RTS, CTS, ACK*), teniendo en cuenta la duración del preámbulo de la capa *PHY* en las tramas de RTS, CTS y ACK [43]. Se asume una tasa de transmisión de datos bruta de 12 Mbps (OFDM-12)[34]. La simulación tendrá un contador de tiempo (tiempo de emulación) y un contador de paquetes entregados con éxito para cada estación con el fin de hacer más sencillo el cálculo del rendimiento de cada estación. El calculo del rendimiento se calcula con la siguiente expresión:

[33] En un escenario ideal, si tenemos 10 estaciones (N = 10) la simulación se terminará cuando las diez estaciones hayan transmitido 1000 paquetes cada una. Condición uniforme.
[34] Recomendada en [43] para compatibilidad con sistemas DSSS de menor capacidad como IEEE 802.11b.

$$Tput = \frac{Datos\ enviados\ éxitosamente\ (bits)}{Tiempo\ total\ transcurrido\ (seg)}$$

Al igual que en cualquier escenario práctico, cada estación obtendrá una tasa de transmisión real de datos inferior a la tasa de transmisión de datos en bruto; esto debido al retardo introducido por colisiones, por los encabezados y las tramas de control.

El método Throughput opera con objetos de la clase STA y es capaz de simular un escenario de contienda por el canal. Tenemos algunas banderas booleanas importantes como las de *'collision'* y *'mediumIdle'*, que indican cuando ocurrió una colisión y cuando el canal está libre u ocupado respectivamente. Hemos cubierto la mayoría de las posibles situaciones en las que una colisión puede ocurrir y también se incluyen los siguientes criterios fundamentales para garantizar una simulación eficiente.

El primer criterio lidia con el problema de que la aplicación se congele, esto puede ocurrir cuando hay demasiadas colisiones y la simulación es incapaz de entregar 10,000 paquetes con éxito. La GBEBapp no puede manejar la pérdida de tantos paquetes con colisiones que tienden a infinito y por lo tanto no arroja datos de salida. Para evitar este problema, fijamos la tasa de error de pérdida de paquetes a un máximo de 10%, entonces, si más de 100,000 paquetes se pierden, entonces el método 'Throughput' muestra una ventana de mensaje de error y sale de la simulación.

El segundo criterio evitará una situación de *'starving'*[35]. Esto sucede cuando una de las N estaciones acapara el canal inalámbrico con sus transmisiones y deja al resto de las terminales sin posibilidad de enviar sus paquetes debido a una ventana de contención muy elevada debido a muchas colisiones producidas. Nuestro criterio es evitar que una estación logre enviar exitosamente más de cien veces los paquetes transmitidos que cualquier otra estación. Si este fenómeno llegara a ocurrir, entonces el método 'Throughput' empezará la simulación completamente de nuevo.

public float SelectProb(int totalT, int tput)

Método que asigna una probabilidad de selección a cada individuo para ser elegido en una operación genética durante las iteraciones, esta probabilidad será un valor flotante entre 0 y 1 almacenado en el parámetro *selecP* de un objeto GBEB. Utilizamos un modelo de selección ruleta.

public GBEB Selection (GBEB[] a)

Este método es responsable de elegir al azar a un individuo de una población (currentGen) de acuerdo a su aptitud o fortaleza (mediante la probabilidad de selección que es asignada con el método *SelectProb*). Entonces, los individuos con una alta probabilidad de selección son más propensos a ser elegidos por este método. La suma de las probabilidades de selección de todos los

[35] El fenómeno de '*starving*' (inanición) se explicó en el Capítulo 2 del presente documento.

individuos de la población M es la unidad. El método de selección devuelve un único objeto (un individuo) seleccionado de un arreglo de objetos GBEB.

public GBEB[] Crossover (GBEB a, GBEB b)

Operación genética de Recombinación. En la operación de recombinación o cruce se mezclarán los 'genes' (valores de W y mStages) de los padres (dos individuos seleccionados de la población *currentGen*) para crear una descendencia de dos individuos. Cada hijo tendrá información del 'padre' y de la 'madre'. El método devuelve un arreglo de dos objetos GBEB.

public GBEB Mutation (GBEB a)

Operación genética de Mutación. La operación de mutación realizará una ligera 'mutación' (cambio) en un gen del individuo de la generación actual. Primeramente se genera un valor entero aleatorio (n) de 0 a 5, y entonces se selecciona al azar una posible forma de aplicar la mutación ($W \pm n$ ó $mStages \pm n$). Al final de este método tenemos una sentencia de precaución para no permitir valores negativos o cero en W o en mStages. Si fuera el caso, el valor se sustituye por 1. El método devuelve un objeto único GBEB (mutante).

public GBEB Reproduction (GBEB a)

Operación genética de Reproducción. La operación de reproducción se aplica sobre un individuo de la generación actual (seleccionado de acuerdo a su aptitud o fortaleza) y es clonado a la siguiente generación sin cambio alguno. Este método imita el principio de la selección natural 'la supervivencia del más apto'. El método devuelve un objeto único GBEB.

public GBEB[] CrossoverMutation (GBEB a, GBEB b)

Operación genética de Recombinación + Mutación (evolución). La operación de Recombinación + Mutación necesita dos individuos de la generación actual para aplicar los métodos de Recombinación y Mutación a ellos en ese orden. Primero, genera una descendencia de dos individuos con la operación de recombinación y luego muta a cada hijo obtenido. El método devuelve un arreglo de dos objetos GBEB (descendencia mutada).[36]

El código Java más relevante para la aplicación GBEBapp se puede encontrar en el APÉNDICE para la versión electrónica de este documento.

4.2 Pruebas iniciales para una WLAN de dos terminales.

Una vez explicado el funcionamiento de la aplicación GBEBapp desarrollada, estamos en condiciones de iniciar las pruebas pertinentes para corroborar su funcionamiento y la efectividad de los resultados arrojados. Para dichas pruebas iniciales vamos a simular la red inalámbrica más

[36] Las cuatro operaciones genéticas modificarán los "genes" de sólo dos parámetros de los objetos GBEB, estos son los parámetros que definen el espacio de búsqueda, W y mStages.

sencilla posible con sólo dos terminales. Las condiciones de tráfico son las mismas descritas anteriormente y se operará bajo el estándar IEEE 802.11g con RTS-CTS activado.

Haremos una comparación de los resultados obtenidos para una red convencional operando con los valores CWmin y CWmax especificados por el estándar y los valores sugeridos por nuestra aplicación, lo que esperamos ver es una mejora significativa en el rendimiento promedio (medido en kbps) de la red. Se analizarán tres escenarios distintos y se concluirán los resultados obtenidos en cada caso. Para cada uno se muestra una gráfica comparativa entre los resultados obtenidos de la aplicación GBEBapp contra el rendimiento promedio obtenido con el estándar IEEE 802.11g.

Primer caso (Figura 4.2). Las condiciones fueron las siguientes. Veinte corridas con el programa genético GBEBapp activado con valores iniciales W = 16 y mStages = 6, se seleccionó una población de M = 15 con un criterio de terminación de 500 generaciones. Y para la simulación con IEEE 802.11g RTS-CTS activado se usaron los valores del estándar sin modificaciones que son W = 16 y mStages = 6, es decir, CWmin = 15 y CWmax = 1023. [44]

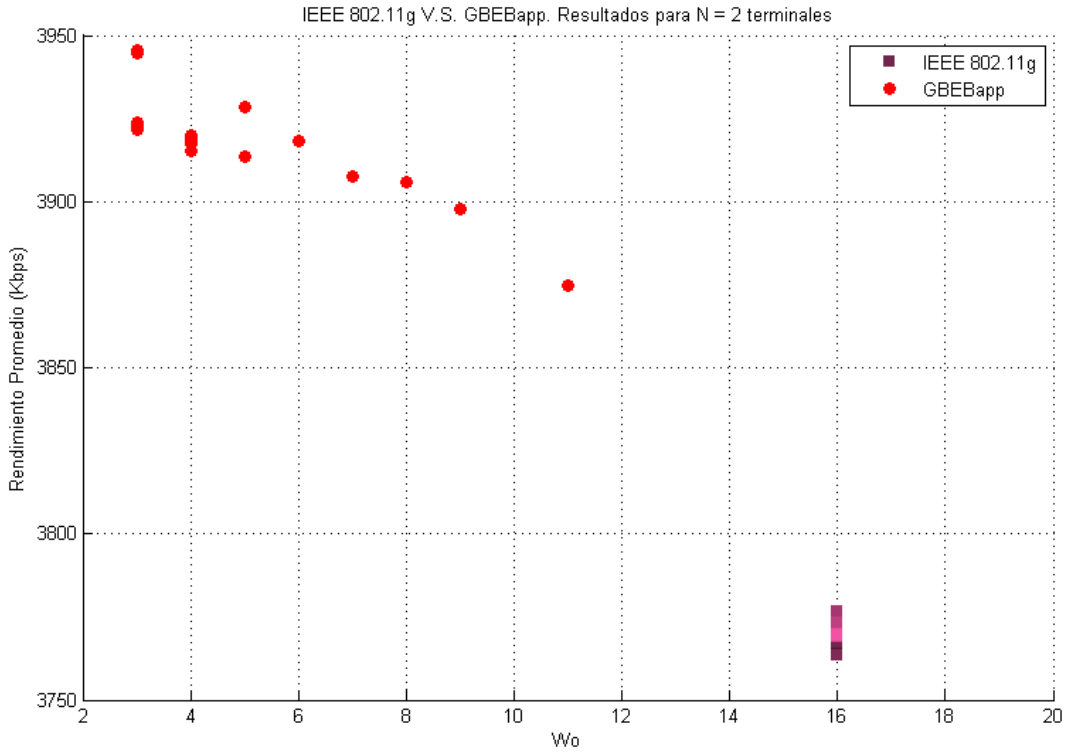


Figura 4.2 - Desempeño de pruebas iniciales para la aplicación GBEBapp. Red de dos terminales.

La simulación corrió con ambas terminales contendiendo por acceso al medio, cabe señalar que el algoritmo es independiente del número de terminales y el único factor que determina su procesamiento es el Criterio de Terminación elegido. Como podemos observar, se nota una mejora en el rendimiento promedio alcanzado con las soluciones arrojadas por la GBEBapp. En una primera aproximación, nuestra aplicación obtiene con el mejor resultado sugerido, una mejora de alrededor del 5% por sobre el estándar IEEE 802.11g con RTS-CTS activado.

Gracias a los resultados de nuestra primera prueba podemos notar un comportamiento peculiar en los valores de W y mStages sugeridos; el valor de CWmax reflejado en mStages no parece cambiar por mucho, sin embargo, nuestra aplicación GBEBapp nos recomienda bajar el valor W de 16 hasta valores de entre 4 y 8 lo que nos lleva a concluir que para redes de pocas terminales, un valor alto de CWmin disminuirá el rendimiento promedio obtenido por un desperdicio de utilización en el canal, esto porque estamos manejando tiempos de espera en el proceso de *backoff* que son innecesarios al tener apenas dos estaciones contendiendo.

Segundo caso (Figura 4.3). Las condiciones fueron exactamente las mismas que para el caso primero pero con valores iniciales de W = 160 y mStages = 60. Intencionalmente alejamos los valores iniciales diez veces de lo sugerido por el estándar IEEE 802.11g para así verificar la capacidad de nuestro programa de hallar siempre los mejores resultados, aun partiendo de valores muy lejanos a los apropiados.

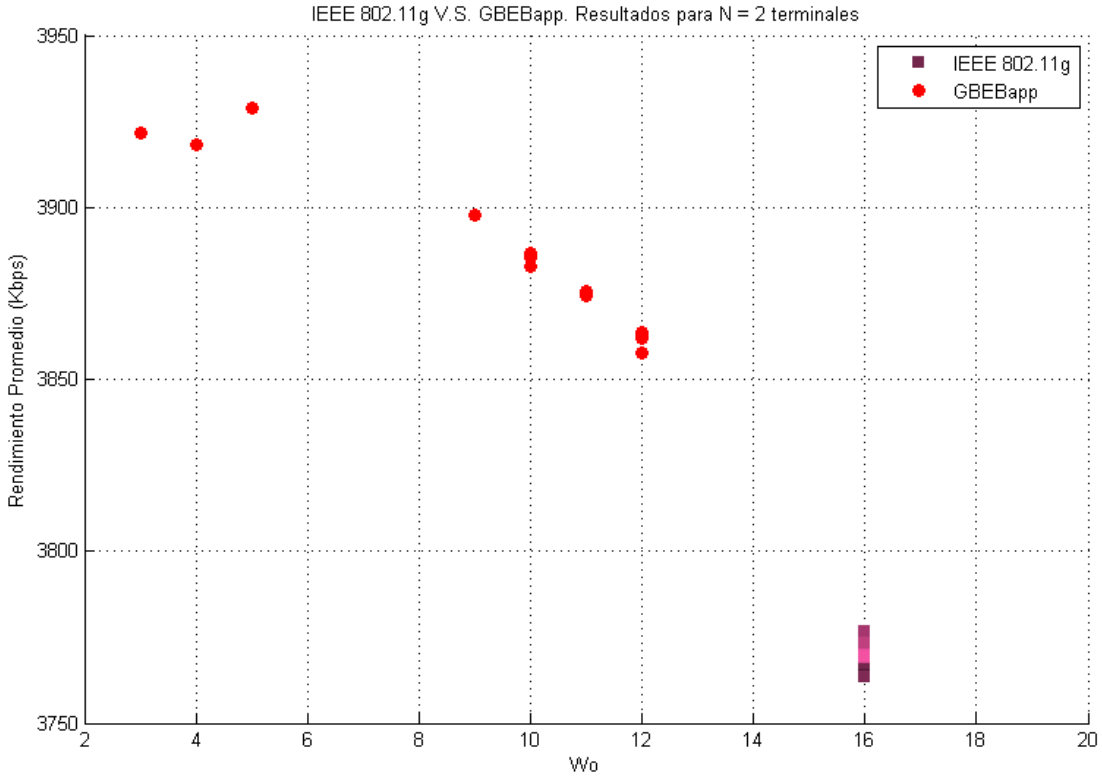


Figura 4.3 - Mejora en el rendimiento por la GBEBapp con valores iniciales alterados.

Como podemos observar, a pesar de que modificamos drásticamente los valores iniciales, los resultados arrojados por la GBEBapp muestran aún una mejora en el rendimiento promedio alcanzado con las soluciones arrojadas de alrededor del 4% por sobre el estándar IEEE 802.11g con RTS-CTS activado. Notablemente no se lograron los mismos resultados que con la primera prueba, pero si podemos confirmar que la GBEBapp tiene la capacidad de sugerir mejores resultados de BEB que el estándar por sí solo.

Con esta segunda prueba, podemos visualizar otra situación peculiar. Los valores arrojados para W bajaron desde un valor inicial de 160 hasta valores alrededor de 10, mientras que los valores para mStages no parecieron mostrar relación alguna. Podemos concluir que en una red de dos terminales, el valor de mStages no es tan crítico pues no se esperan muchas colisiones, lo que se desea es un valor inicial de CWmin bajo para garantizar tiempos de contención justos y que al mismo tiempo no sea tan grande que se desperdicie el uso del canal. Los mejores valores de rendimiento (medido en kbps) se obtuvieron con los valores de W más pequeños. GBEBapp ha demostrado nuevamente su correcto funcionamiento dado que podemos ver los puntos rojos (resultados para GBEBapp) siempre por encima de los cuadrados morados (resultados para el estándar IEEE 802.11g).

Finalmente intentamos justificar nuestra conclusión anterior. La aplicación GBEBapp cuenta con una opción para dejar fijo el valor de mStages, si lo hacemos así, podremos medir su capacidad de mejorar el rendimiento de la red con dos terminales variando únicamente el valor inicial de W. En este tercer caso (Figura 4.4) podemos corroborar que la tendencia es hacia valores de CWmin pequeños.

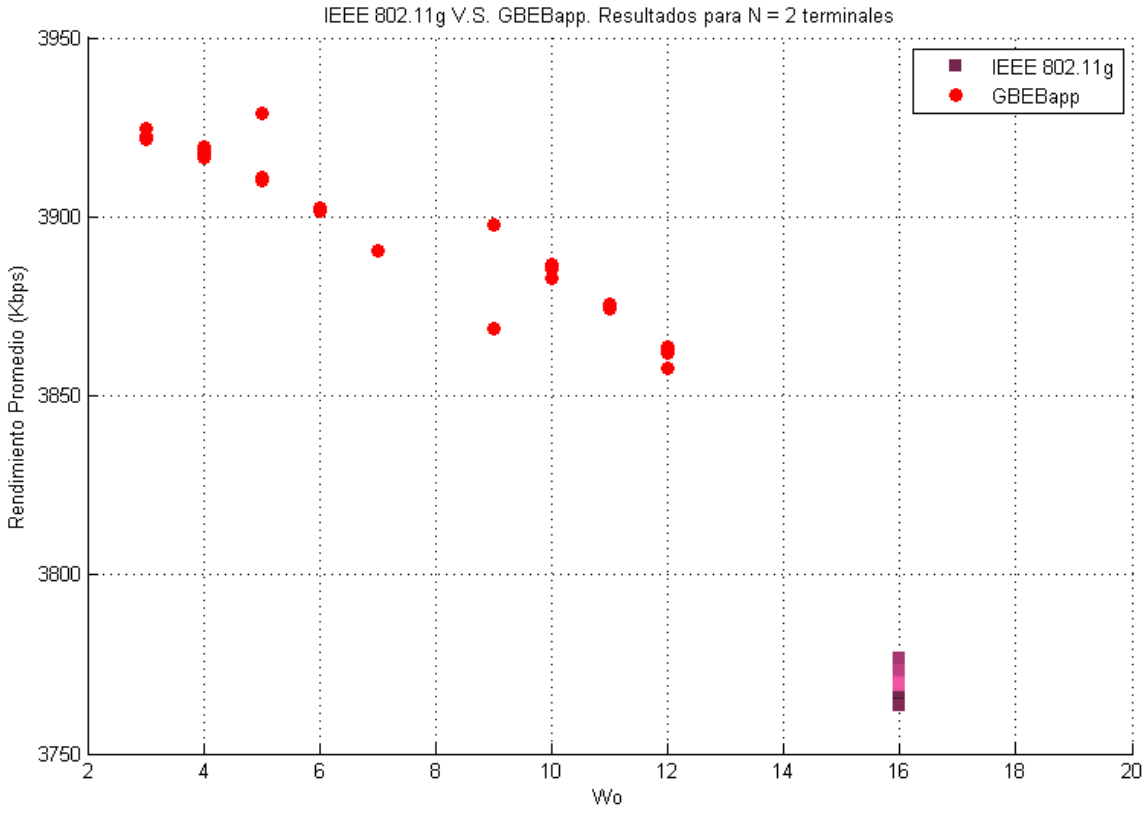


Figura 4.4 - Desempeño de los resultados arrojados por la GBEBapp con el valor mStages fijo. Red de dos terminales.

A pesar de dejar mStages fijo al valor sugerido de 6, los resultados arrojados por la GBEBapp son mejores que los obtenidos con IEEE 802.11g. Notamos nuevamente la preferencia y los mejores rendimientos obtenidos para valores pequeños de Wo cuando se tiene N = 2.

Para los tres escenarios planteados en las pruebas iniciales, el algoritmo tardó menos de 1 minuto en converger y logró una mejora en el rendimiento promedio de la red de alrededor del 5% con los valores de la ventana de contención sugeridos.

4.2.1 Parámetros recomendados para obtener resultados óptimos con la aplicación GBEBapp.

Basándonos en las tres pruebas iniciales, donde los parámetros del programa genético fueron siempre los mismos con un tamaño de población M y un criterio de terminación fijo; nos corresponde ahora evaluar el comportamiento de la GBEBapp variando dichos valores.

Como lo explica la teoría de los algoritmos genéticos, el tamaño de la población y el criterio de terminación son parámetros que determinan el éxito o fracaso de su implementación [45]. Un tamaño de población muy pequeño es tan malo como una población muy extensa, y un criterio de terminación muy estricto será tan inconsistente como uno muy simple.

En esta sección realizaremos experimentos sobre nuestra GBEBapp de modo que encontremos los parámetros óptimos para obtener los mejores resultados. Las pruebas se realizarán para una red de dos terminales y nuestro objetivo siempre será el de maximizar el rendimiento de la WLAN.

En esta sección variaremos, de manera individual, los parámetros fundamentales del algoritmo genético para visualizar su impacto en las sugerencias obtenidas y al final concentraremos la información recolectada en la Tabla 4.1. Los parámetros de la ventana de contención inicial son: W = 16, mStages = 6 y terminales N = 2.

Las primeras tres pruebas se realizaron modificando el valor del tamaño de población M. Tras mostrar los resultados de tres casos con valores de M distintos (Figuras 4.5, 4.6 y 4.7) compararemos el impacto reflejado de dicha configuración en la GBEBapp.

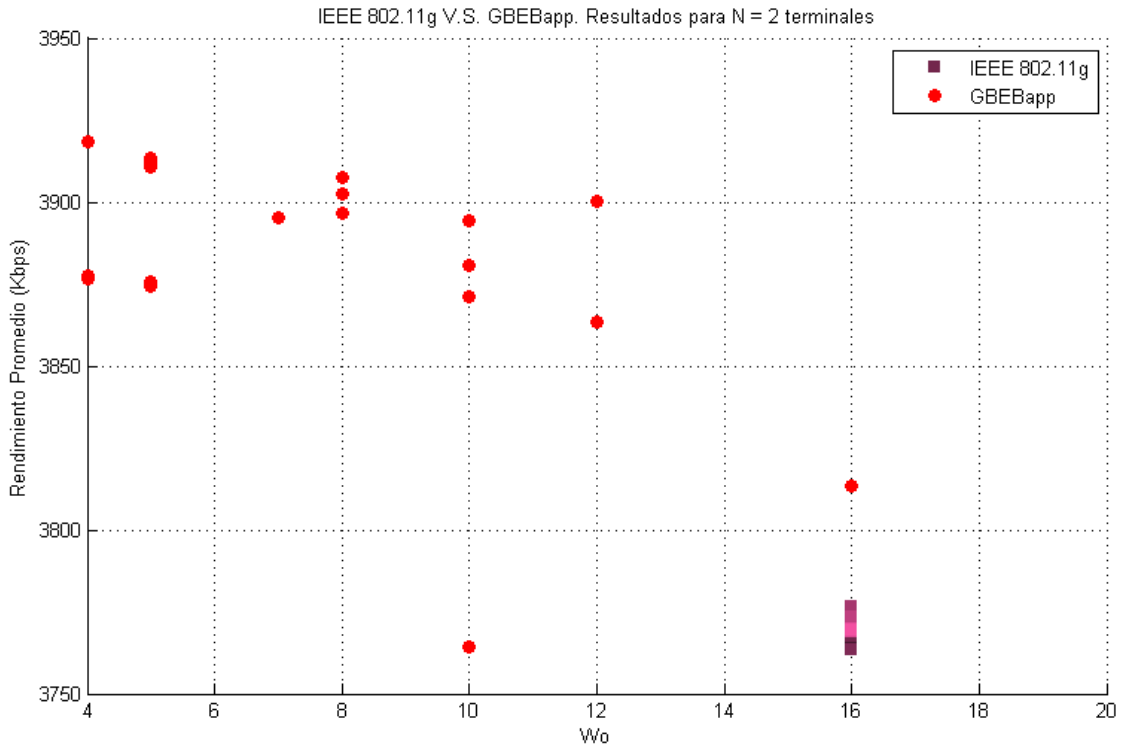


Figura 4.5 - Resultados arrojados para M = 3. Red de dos terminales. Experimento 1.

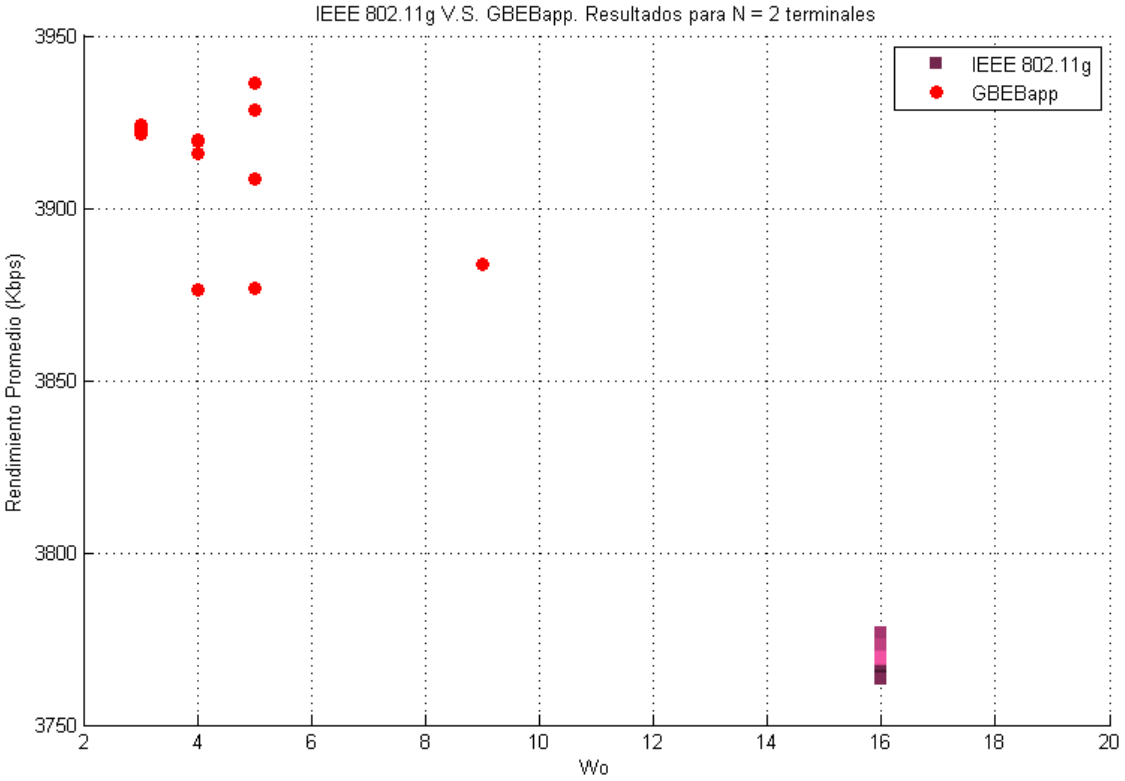


Figura 4.6 – Resultados arrojados para M = 15. Red de dos terminales. Experimento 2.

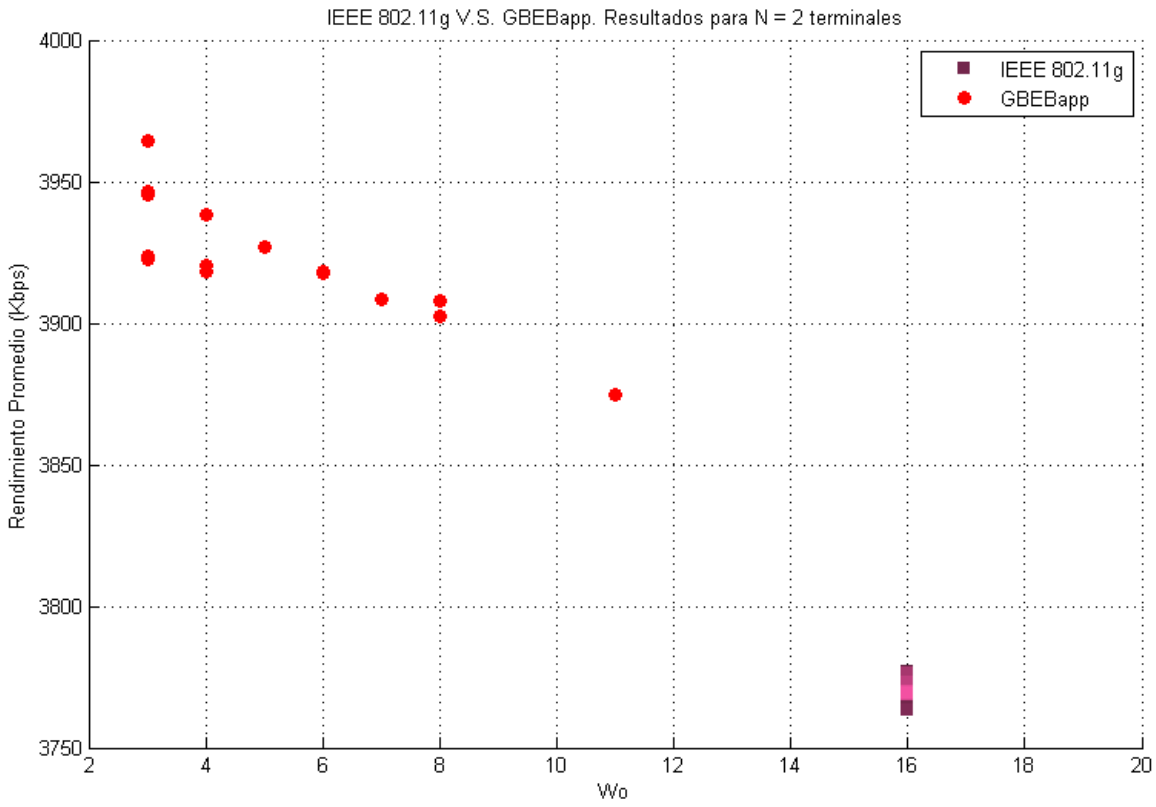


Figura 4.7 – Resultados arrojados para M = 25. Red de dos terminales. Experimento 3.

Como podemos observar, el promedio de las sugerencias obtenidas con GBEBapp (puntos rojos) siempre está por encima de los resultados obtenidos con el estándar IEEE 802.11g por sí solo (cuadros morados). Sin embargo, obtenemos mejores o peores rendimientos promedios dependiendo del tamaño de la población M (exp. 1 al 3).

Ahora realizaremos tres pruebas modificando el valor del criterio de terminación. Tras mostrar los resultados de tres casos con criterios de terminación flojos, medios y estrictos (Figuras 4.8, 4.9 y 4.10) compararemos el impacto reflejado de dicha configuración en la GBEBapp.

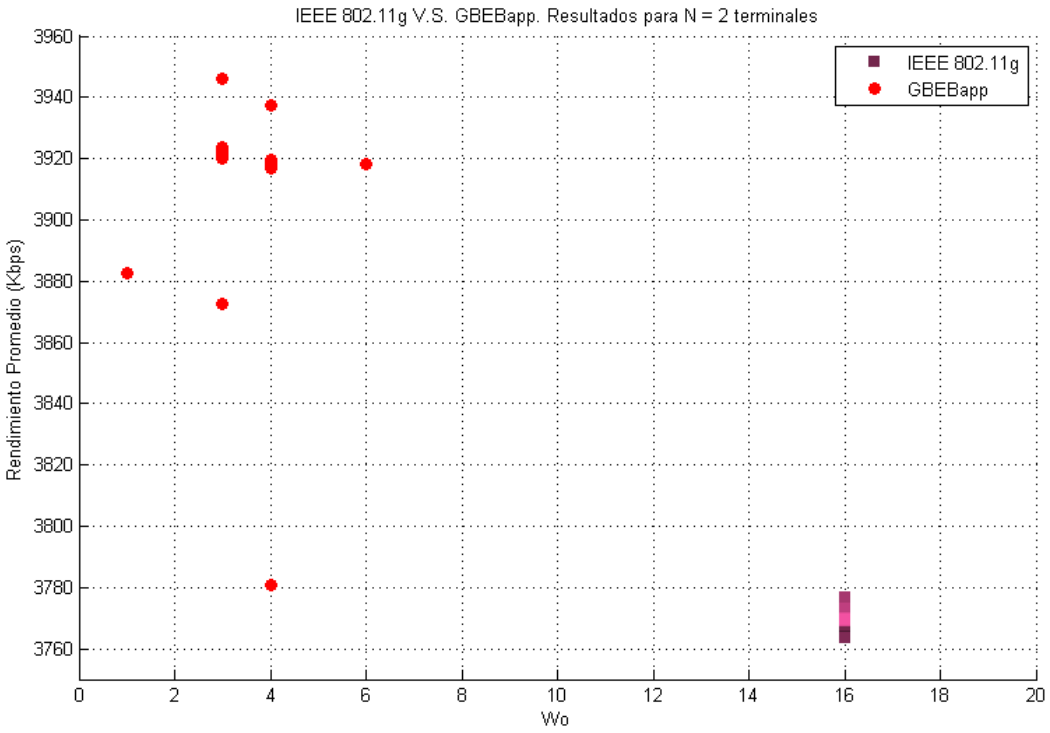


Figura 4.8 - Resultados con un criterio de terminación BAJO: # generaciones = 20. Red de dos terminales. Experimento 4.

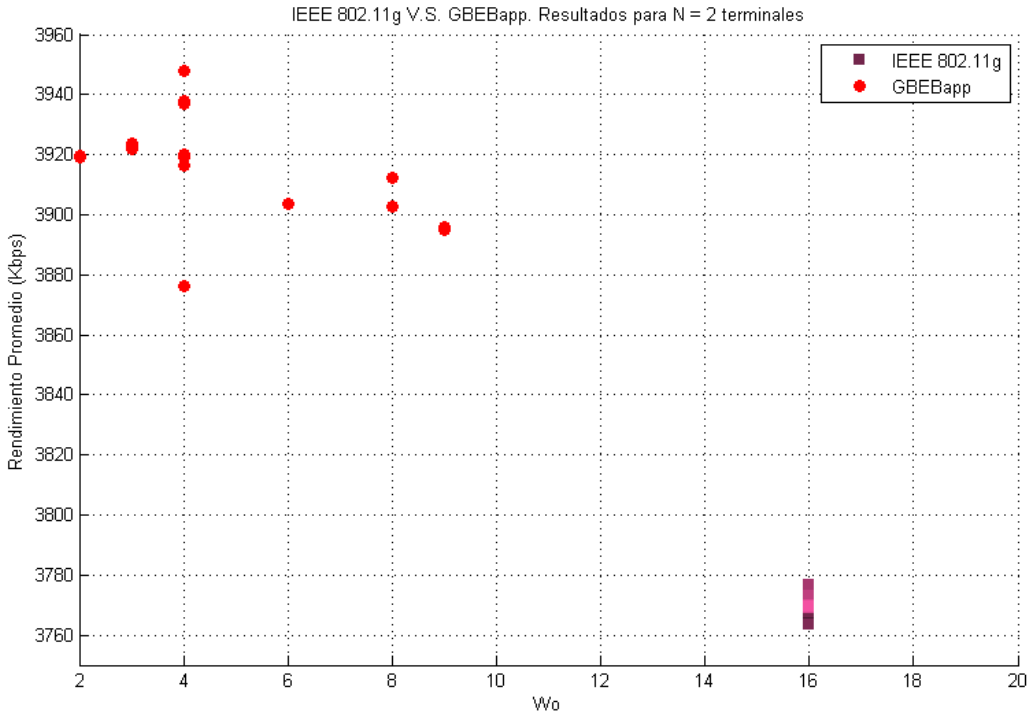


Figura 4.9 – Resultados arrojados con un criterio de terminación MEDIO: # generaciones = 200. Red de dos terminales. Experimento 5.

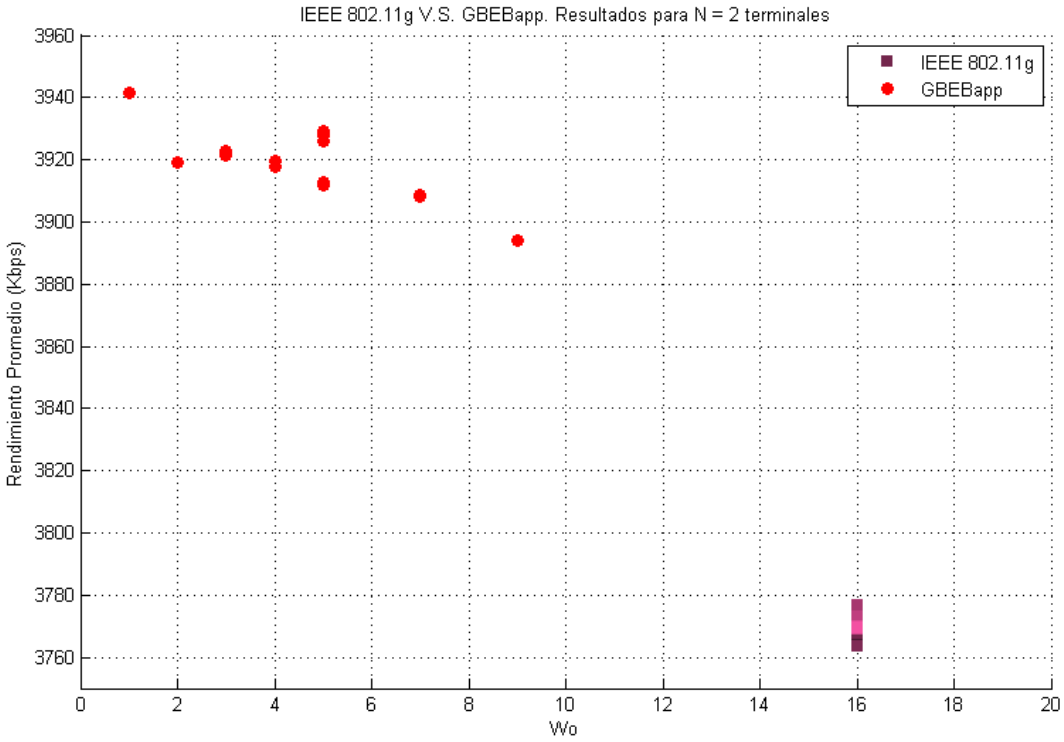


Figura 4.10 – Resultados arrojados con un criterio de terminación ALTO: # generaciones = 1000. Red de dos terminales. Experimento 6.

De manera similar podemos observar que el promedio de las sugerencias obtenidas con GBEBapp (puntos rojos) siempre está por encima de los resultados obtenidos con el estándar IEEE 802.11g. Sin embargo, obtenemos mejores o peores rendimientos promedios dependiendo de la severidad del criterio de terminación utilizado, al igual que menores o mayores tiempos de procesamiento por la GBEBapp.

Finalmente realizaremos una prueba utilizando la otra modalidad del criterio de terminación por medio de una variación porcentual en el individuo más fuerte de la población. Nuestro objetivo es comparar los resultados obtenidos con alguno de los tres casos anteriores para poder estimar su relación. En la Figura 4.11 usaremos un criterio de terminación MEDIO con un valor de 0.01% de variación en el individuo más fuerte y analizaremos a que caso se asemeja más.

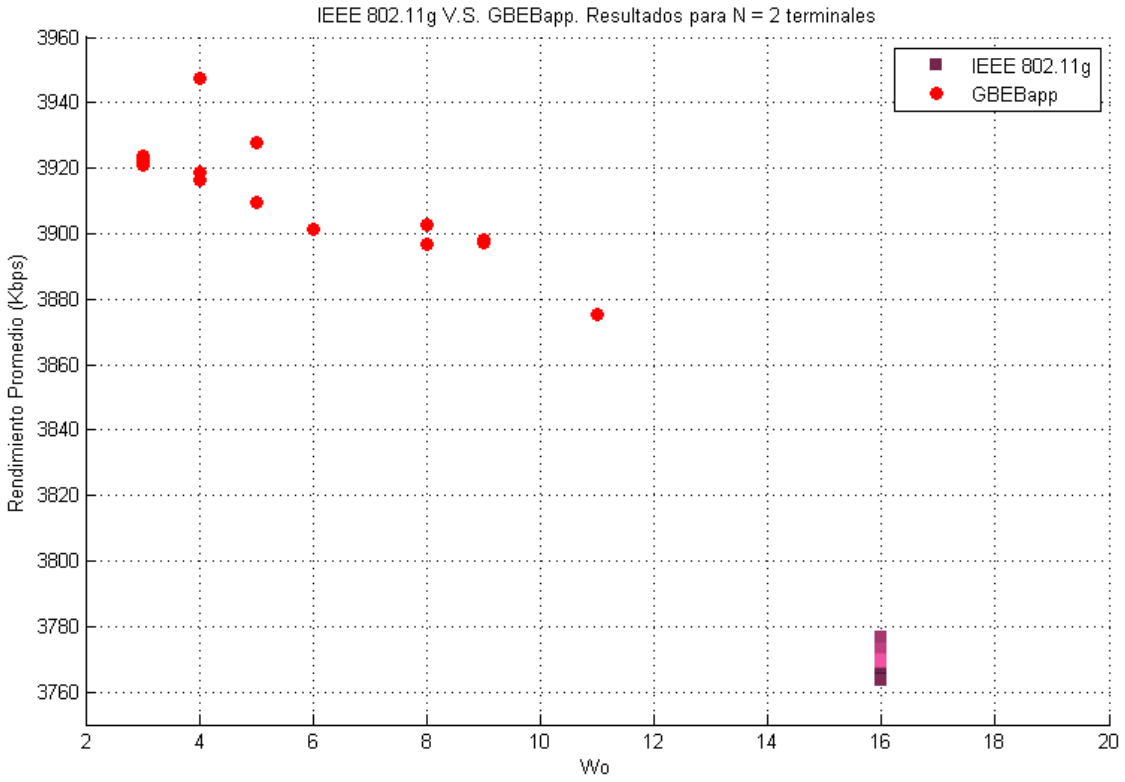


Figura 4.11 – Resultados arrojados con criterio de terminación MEDIO: cambio en el individuo más fuerte = 0.01%. Red de dos terminales. Experimento 7.

Como se observa, los resultados obtenidos en el exp. 7 con un porcentaje mediano (0.01%) son muy similares a los obtenidos en la Figura 4.10 (exp. 6), es decir, se aproximan a las sugerencias de la GBEBapp cuando usamos 1000 generaciones para terminar el algoritmo.

A continuación se presenta el concentrado de los datos recabados (Tabla 4.1) con estos experimentos y se pretende dar una idea de primer plano acerca de los parámetros usados en el algoritmo genético y su impacto en los resultados generados, podemos comparar entonces, tiempos de procesamiento, rendimiento promedio alcanzado y número de colisiones simuladas.

DATOS DE ENTRADA: GBEBapp							RESULTADOS OBTENIDOS (PROMEDIOS)					
#exp.	Wo	m_o	#RUNS (corridas)	N (terminales)	M (población)	Criterio de Terminación	W	mStages	Rendimiento (bps)	#Colisiones	Tiempo procesamiento	Mejora sobre IEEE 802.11g
1	16	6	20	2	3	# gens = 500	7.6	9.7	3881219.8	2437	Bajo	2.96%
2	16	6	20	2	15	# gens = 500	3.9	5.95	3915746.4	3281	Medio	3.88%
3	16	6	20	2	25	# gens = 500	4.7	9.05	3923324.8	**652**	**Alto**	**4.08%**
4	16	6	20	2	15	# gens = 20	3.4	6	3911340.5	1266	Bajo	3.76%
5	16	6	20	2	15	# gens = 200	4.5	6.85	3916979.0	**5660**	**Bajo**	**3.91%**
6	16	6	20	2	15	# gens = 1000	4.1	6.45	3919656.0	**772**	**Alto**	**3.98%**
7	16	6	20	2	15	% más fuerte = 0.01%	5	7.65	3914694.5	532	Alto	3.84%

Tabla 4.1 – Concentrado de la información obtenida con los experimentos realizados para la sección 4.2.1.

Como se muestra en la Tabla 4.1, la mejora del rendimiento (medida porcentualmente sobre lo obtenido con el estándar IEEE 802.11g por si solo) varía según se modifiquen los parámetros del algoritmo genético con la aplicación GBEBapp. Aunque parecieran diferencias mínimas; el tiempo de procesamiento, las colisiones generadas y los valores promedio sugeridos para la ventana de contención Wo (CWmin) y mStages (CWmax) son notablemente distintos. De la misma manera debemos recordar que estas son pruebas realizadas en una primera aproximación para poner a prueba el programa y se simuló una red de únicamente dos terminales inalámbricas. En el siguiente capítulo se hará el análisis de rendimiento para N terminales.

La teoría de los algoritmos genéticos ha sido demostrada con las pruebas realizadas. Los mejores resultados se obtuvieron para una población superior y un criterio de terminación estricto, sin embargo, esto no significa que sea la mejor forma de hacer trabajar la GBEBapp. Como podemos notar, la diferencia no es tan extraordinaria entre los resultados obtenidos para una población superior (exp. 3) y un criterio de terminación alto (exp. 6) contra los resultados obtenidos con una población mediana (exp. 2) y criterio de terminación medio (exp. 5). Con esto podemos concluir que el óptimo funcionamiento de la aplicación reside entre criterios medios-altos, dependiendo del tiempo de procesamiento que se quiera invertir y el fino margen de optimización deseado.

En una red de dos terminales contendiendo por acceso al canal inalámbrico, no se espera que ocurran muchas colisiones; de modo que el rendimiento global de la red dependerá más de los tiempos de espera que van relacionados directamente con el tiempo que se aprovecha el medio (utilización) en lugar de tiempo desperdiciado por retransmisiones. Por esta razón para los experimentos realizados en esta sección, los valores sugeridos de Wo (CWmin) para la ventana de contención son mucho menores que el valor de 16 que establece el estándar IEEE 802.11g. Con esto se demuestra el valor agregado de nuestro programa GBEBapp que demuestra en pruebas iniciales su capacidad adaptable al variar los resultados tomando en cuenta el número de terminales contendientes en la red inalámbrica.

V - RESULTADOS.

5.1 Análisis para simulaciones con 'N' terminales.

La verdadera eficiencia de la aplicación GBEBapp presentada en este documento será medida en función de la mejora alcanzada en una red inalámbrica de N terminales; escenario mucho más aproximado a la realidad, y en el cual se presenta una de las dolencias principales del estándar IEEE 802.11g. El rendimiento de las redes WLAN populares como las conocemos, disminuye significativamente cuando crece el número de terminales contendiendo por ocupar el medio inalámbrico. Esto debido a la ineficiente asignación de 'turnos' por medio del sistema de *Backoff* utilizado en el procedimiento CSMA/CA actualmente.

El trabajo de investigación aquí presentado no sugiere modificación alguna al modo de operación del BEB, sino que ofrece una solución adaptable para mejorar su rendimiento en función de características cambiantes de la red que administremos, como puede ser el número actual de terminales contendientes.

El estándar IEEE 802.11g presenta una alta tasa de colisiones conforme aumenta el número de terminales presentes, y lo que es peor, este crecimiento es símil-cuadrático (Figura 5.1). Como sabemos, si aumenta el número de colisiones, se solicitarán más retransmisiones y por lo tanto se desperdiciara tiempo de uso en el canal, lo cual se refleja en una pérdida de igual proporción en el rendimiento neto de la red (kbps).

Dicho fenómeno ocurre, entre otras razones, por el hecho de que los valores para la ventana de contención en el mecanismo BEB del estándar IEEE 802.11g permanecen estáticos; es decir, no se adaptan a los cambios en la red misma [45]. El mecanismo para asignar los tiempos de espera es útil cuando existe un número moderado de terminales, pero es ineficiente cuando existen pocas o muchas contendiendo.

Para un escenario donde múltiples estaciones tienen un gran *buffer* de paquetes por transmitir y pretenden acaparar el medio inalámbrico todo el tiempo; una incorrecta implementación de *Backoff* degradará enormemente la utilización del canal y por lo tanto el rendimiento de la red, conforme más terminales se presenten, más cuidadoso se debe de ser, pues aumentará la complejidad de los cálculos de los procesos estocásticos que el BEB representa.

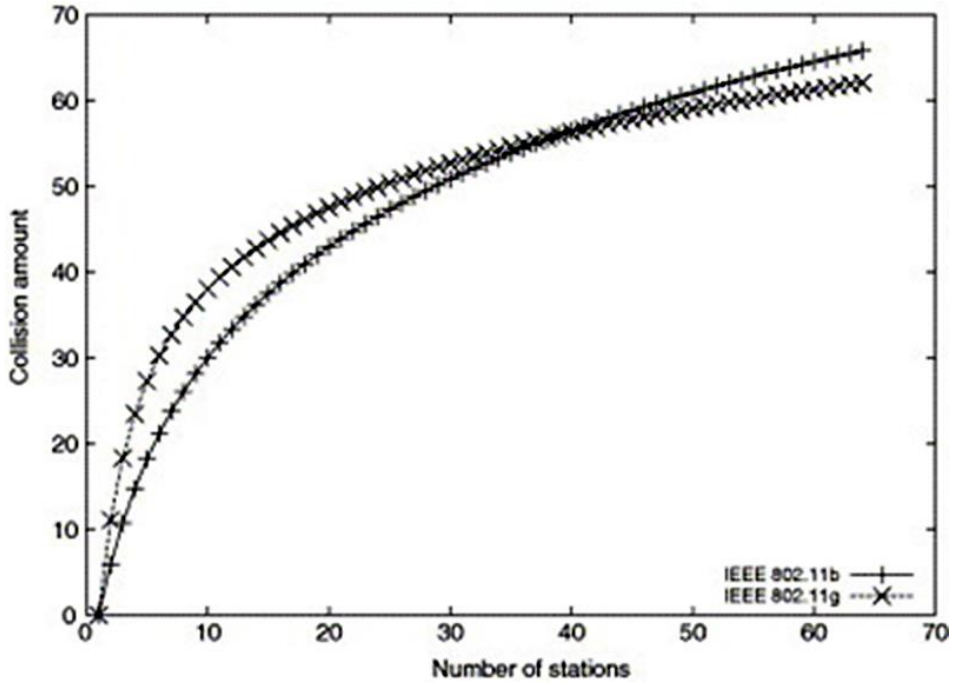


Figura 5.1 – Incremento símil-cuadrático de colisiones con el incremento de terminales. [37]

En las siguientes secciones se simularán diversos escenarios con N terminales y se compararán los rendimientos promedio alcanzados con el estándar IEEE 802.11g por sí solo y los obtenidos con los resultados arrojados por la GBEBapp. Hay que recordar que estos valores cambiarán la idea original de valores estáticos (Wo = 16 y m = 6) por valores dinámicos que cambien dependiendo de la simulación de contienda en una red con N terminales inalámbricas.

Nuestro objetivo será el de poner a prueba la GBEBapp para demostrar su capacidad de procesar los datos y simular escenarios donde existan muchas terminales y donde el estándar IEEE 802.11g no demuestre ser óptimo. Los valores sugeridos encontrados en los siguientes experimentos, así como su comparativa final se presentaran en las siguientes páginas.

Esperamos demostrar la utilidad y veracidad de toda la investigación y todos los cálculos involucrados en la elaboración de este documento y de la aplicación en Java GBEBapp.

5.2 Simulaciones con N terminales.

En esta sección realizaremos experimentos para N = 4, 8, 16, 32 y 64 terminales. Compararemos los resultados obtenidos, y al final concentraremos la información en una tabla comparativa (Tabla 5.1). Durante las simulaciones se variarán y se ajustarán los parámetros del algoritmo genético, según se observaron los resultados de las pruebas realizadas en el Capítulo 4 de este documento.

Se utilizarán los parámetros siguientes: M = 25, Criterio de terminación = 500 generaciones y valores iniciales Wo = 16 y mo = 6.

[37] Derechos al autor. Figura 1 del documento citado en [45].

En cada prueba se mostrará la gráfica correspondiente a los valores sugeridos de Wo y mStages contra el rendimiento en kbps obtenido de 20 corridas. Figuras 5.2 – 5.6.

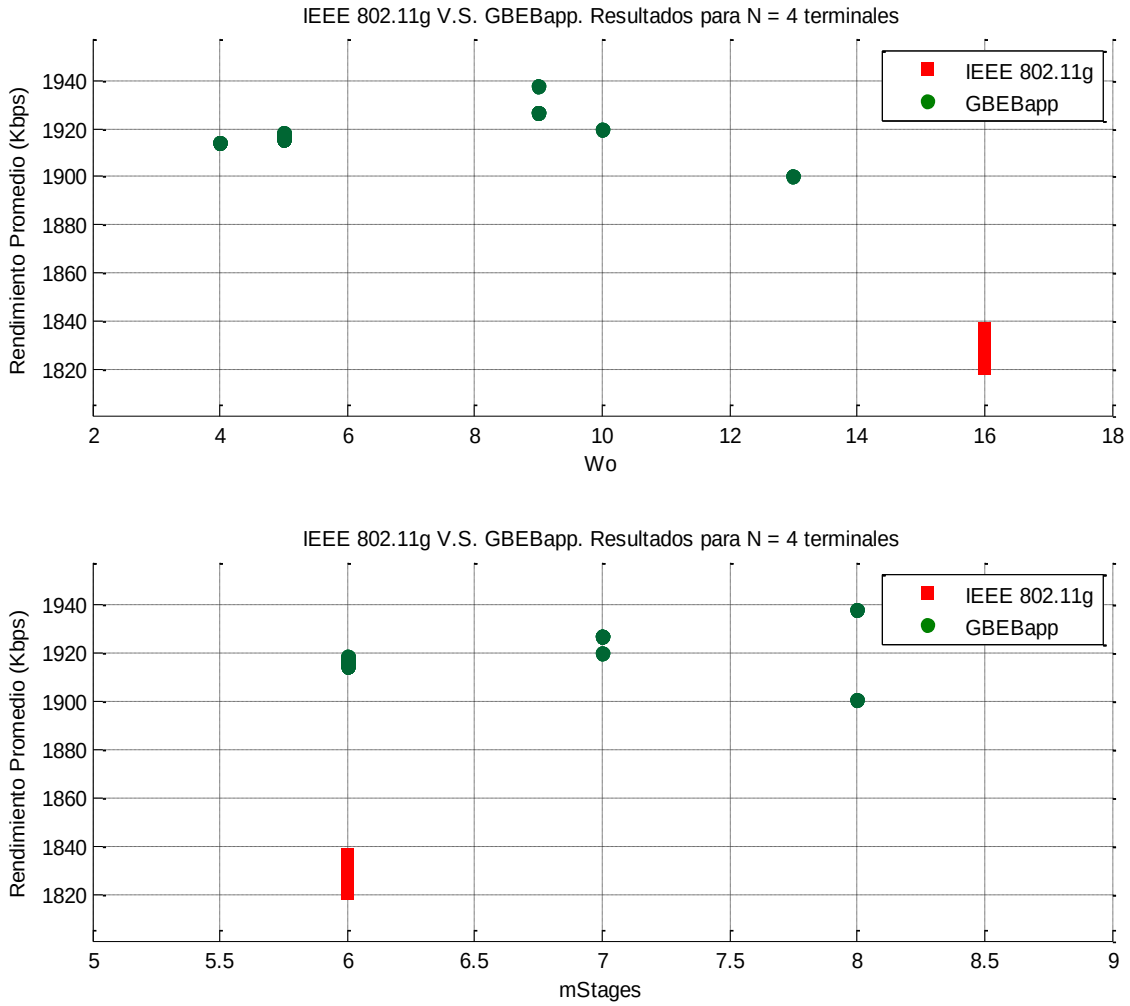


Figura 5.2 – Resultados obtenidos para N = 4. GBEBapp por encima de IEEE 802.11g.

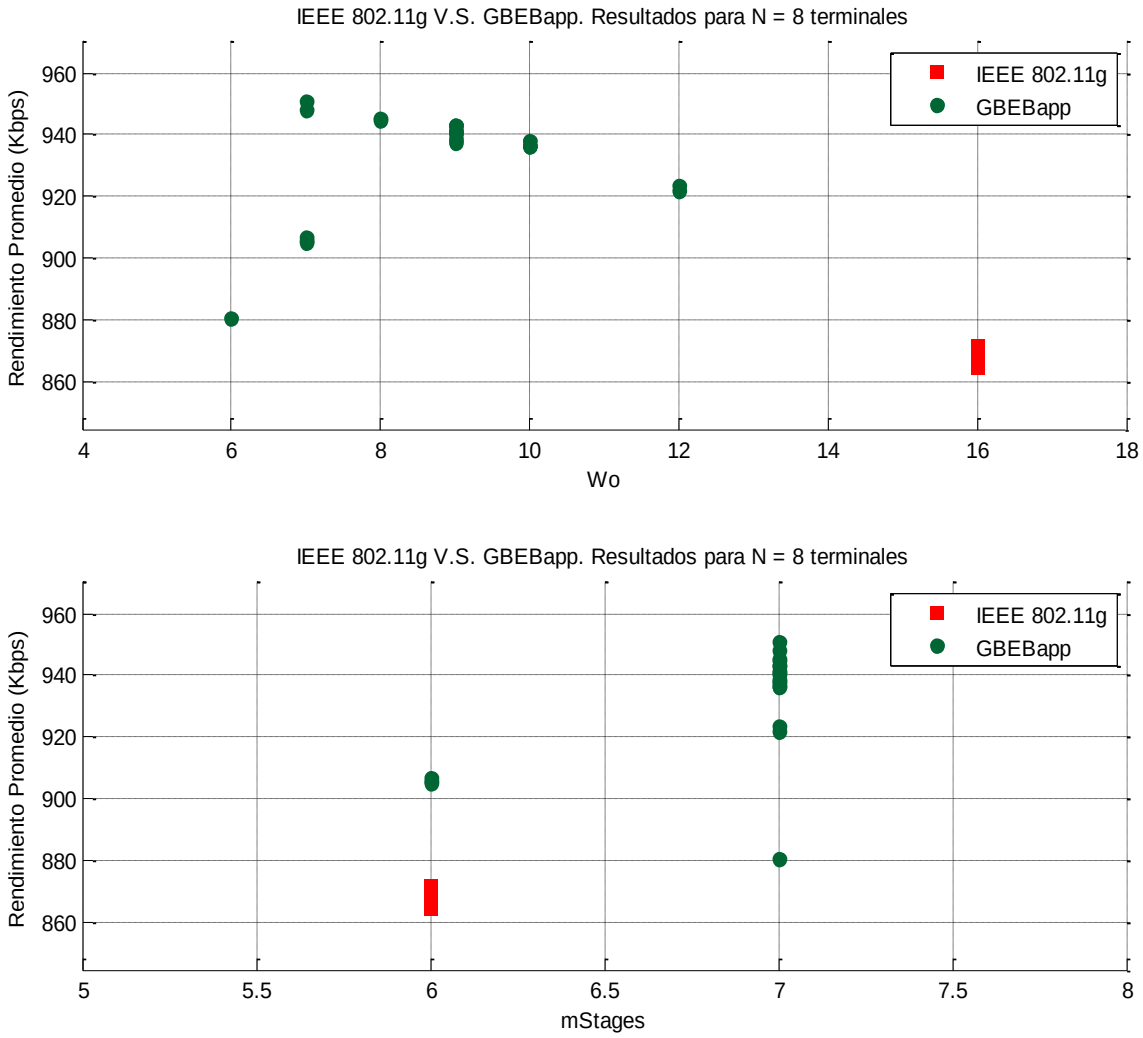


Figura 5.3 – Resultados obtenidos para N = 8. GBEBapp por encima de IEEE 802.11g.

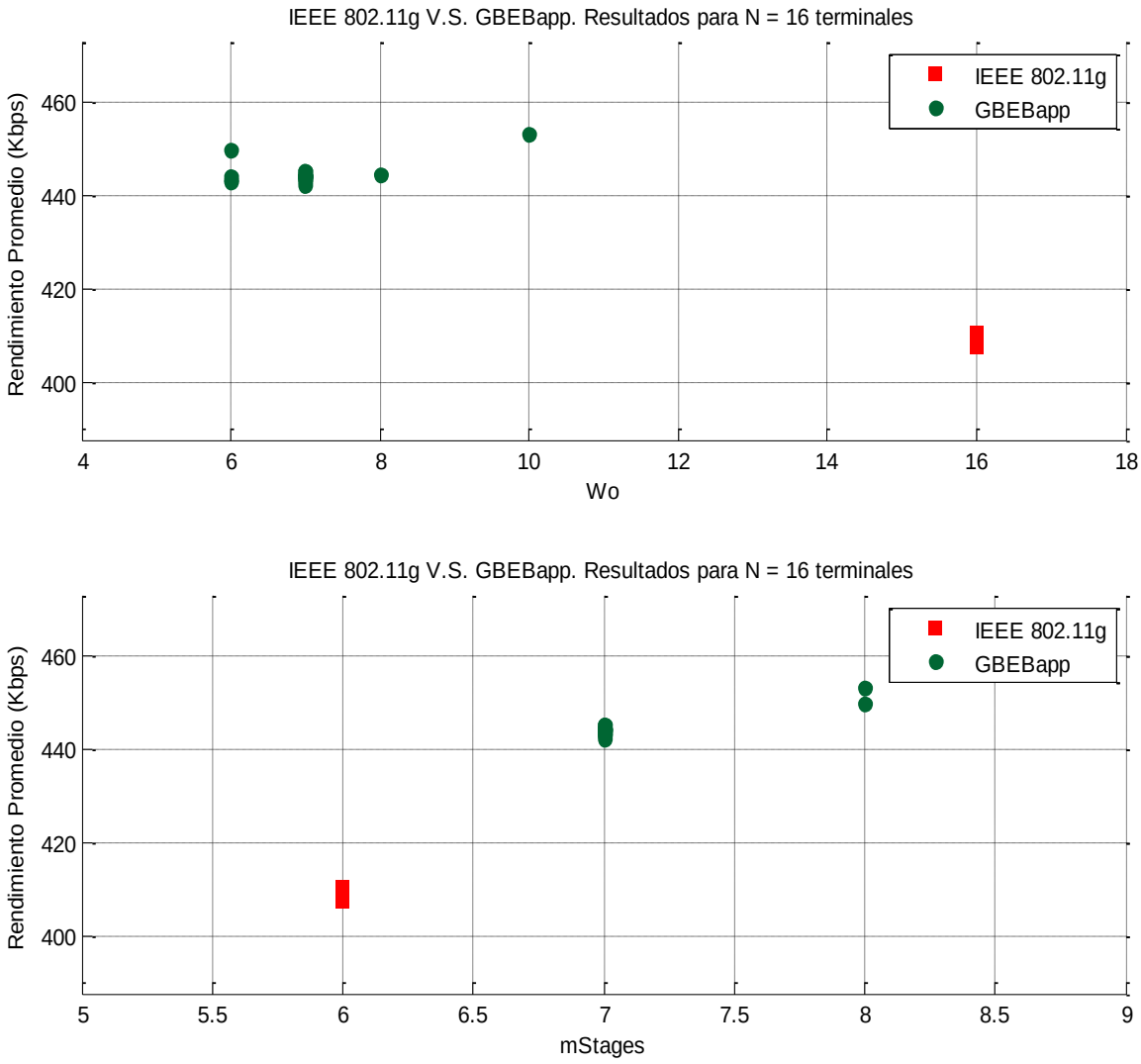


Figura 5.4 – Resultados obtenidos para N = 16. GBEBapp por encima de IEEE 802.11g.

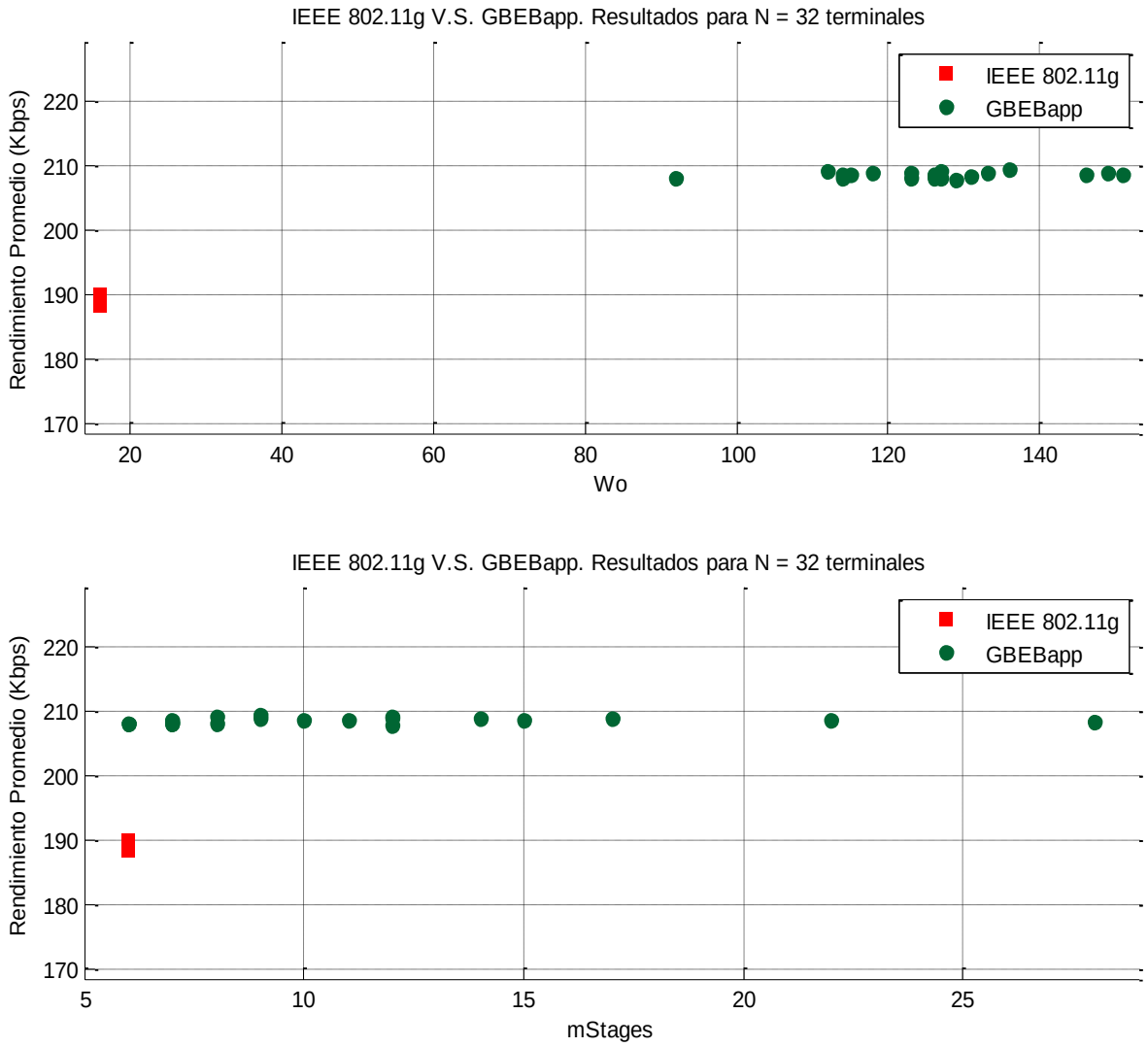


Figura 5.5 – Resultados obtenidos para N = 32. GBEBapp por encima de IEEE 802.11g.

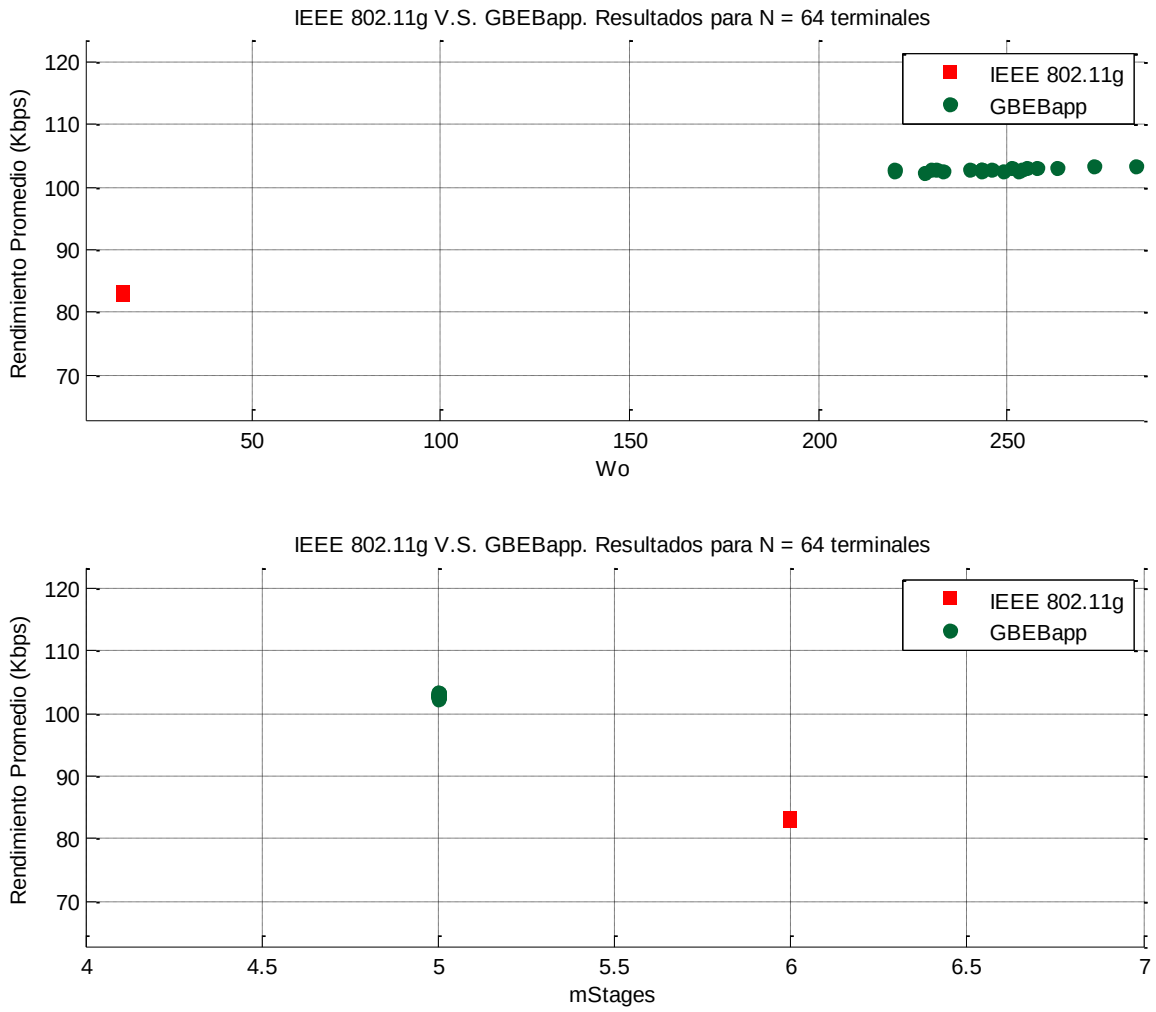


Figura 5.6 – Resultados obtenidos para N = 64. GBEBapp por encima de IEEE 802.11g.

Como podemos observar, en los cinco casos se obtuvieron resultados con la GBEBapp por encima del rendimiento alcanzado con el estándar IEEE 802.11g. La tendencia general para Wo sugerida es de valores pequeños cuando hay pocas terminales (hasta 16), pero salta a valores mayores cuando hay muchas terminales (de 32 en adelante). Correr el programa 20 veces nos ayuda a identificar una zona de convergencia para los resultados sugeridos por la GBEBapp.

Respecto al valor de mStages podemos observar cierta dispersión que no se consolida en valores fijos (a excepción del caso con 64 terminales), de modo que ahora nos proponemos realizar los mismos experimentos pero utilizando la opción de dejar fijo el valor 'mStages' y así analizar como afecta esto los rendimientos logrados por la GBEBapp. Valor mo = 6 (original del estándar).

En las siguientes pruebas (Figuras 5.7 – 5.11) el único valor que podrá modificar la GBEBapp será Wo de modo que podamos corroborar la tendencia para mejorar el rendimiento promedio de la red inalámbrica.

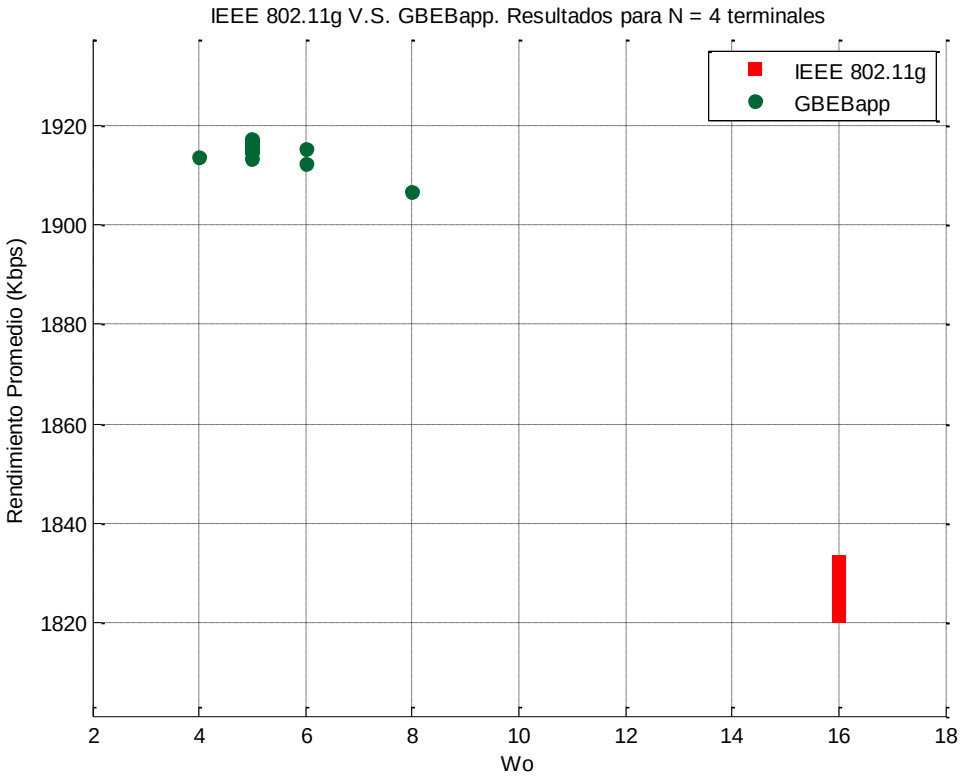


Figura 5.7 – Resultados obtenidos con 'mStages' fijo para N = 4.

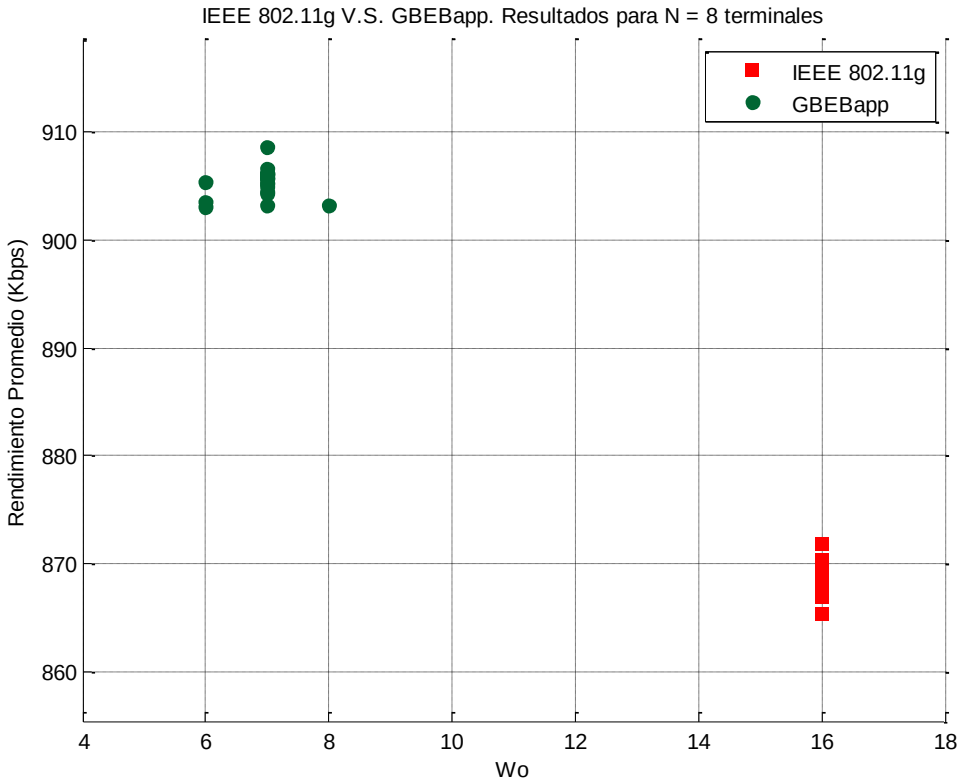


Figura 5.8 – Resultados obtenidos con 'mStages' fijo para N = 8.

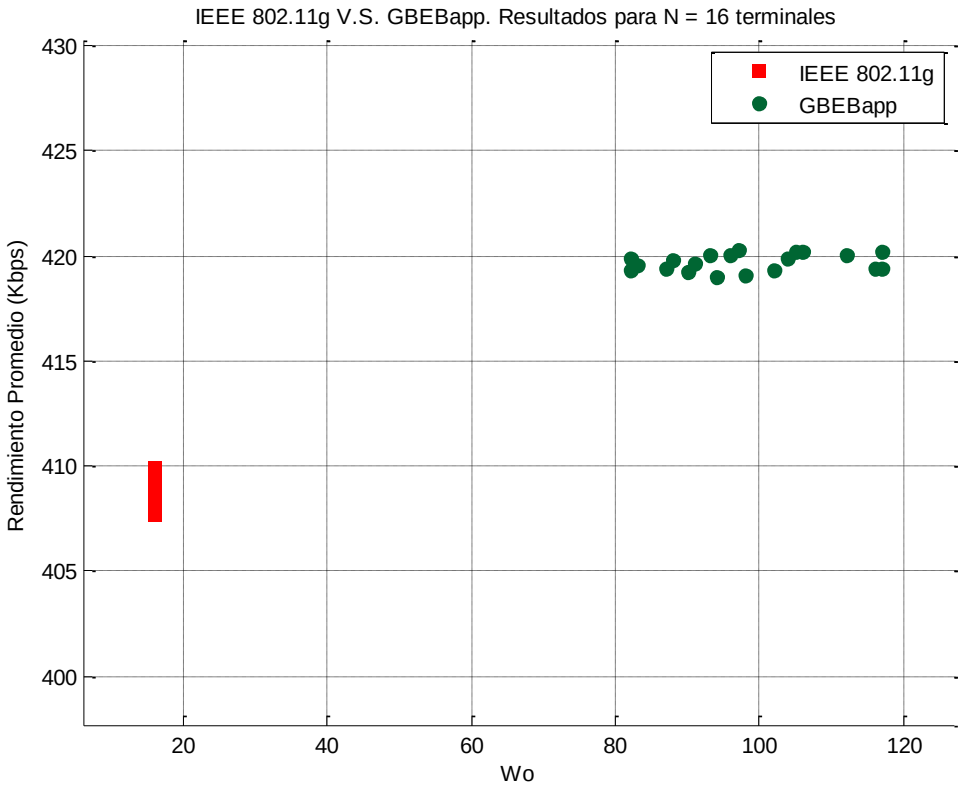


Figura 5.9 – Resultados obtenidos con 'mStages' fijo para N = 16.

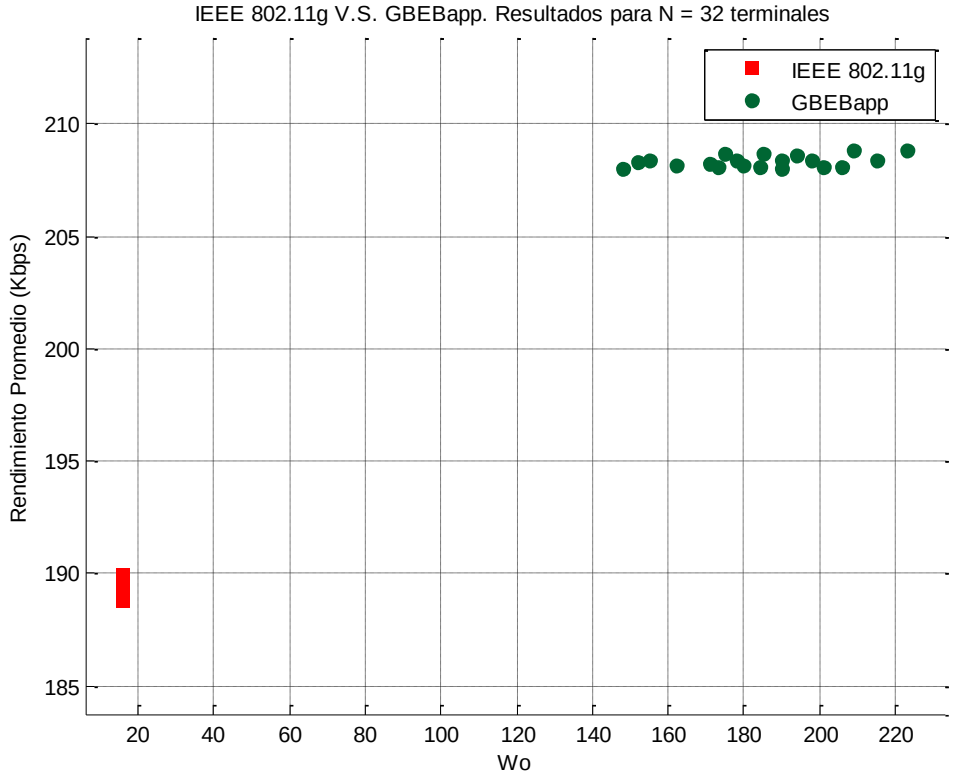


Figura 5.10 – Resultados obtenidos con 'mStages' fijo para N = 32.

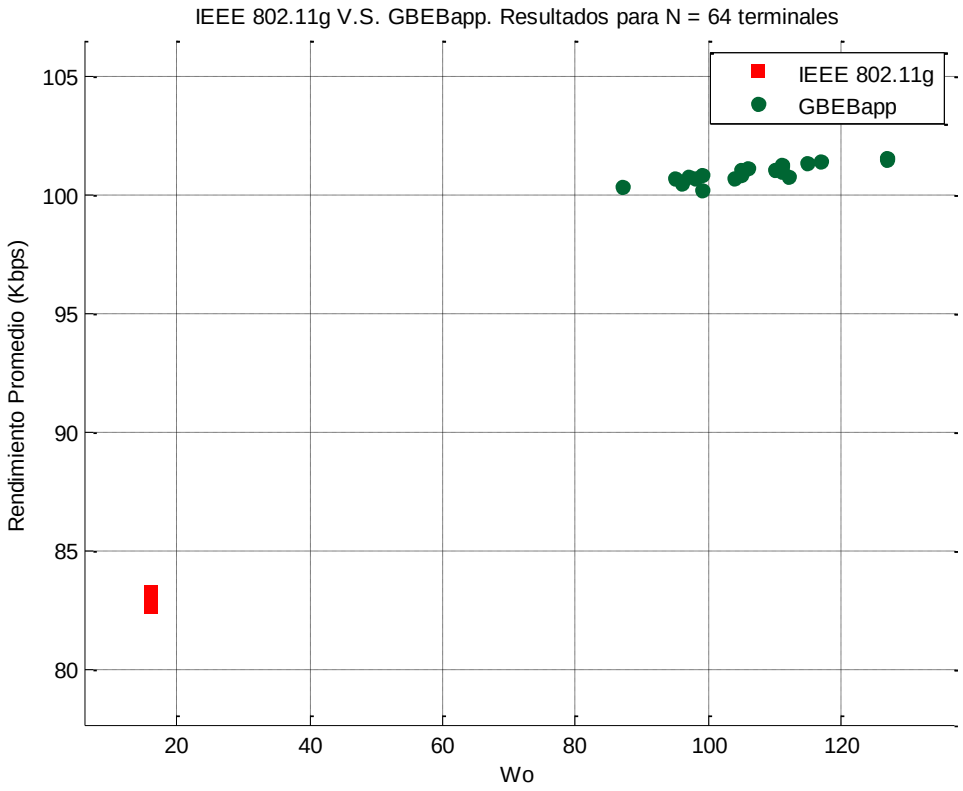


Figura 5.11 – Resultados obtenidos con 'mStages' fijo para N = 64.

Nuevamente podemos observar los resultados obtenidos con la GBEBapp siempre por encima de el rendimiento logrado con el estándar IEEE 802.11g por sí solo. Al dejar fijo el valor de 'mStages', el programa modificará únicamente el valor CWmin (Wo) y observamos en nuestras pruebas un comportamiento peculiar. Para escenarios con pocas terminales (N = 4 y N = 8), las sugerencias de Wo rondan valores bajos, pero dichas sugerencias saltan a valores altos cuando la población en la red inalámbrica crece (N = 16, 32 y 64). Al dejar fijo el valor de 'escalones' CWmax (mStages), la optimización de la contienda por el canal mediante el BEB, dependerá únicamente de un valor apropiado de CWmin (Wo).

Para los últimos tres escenarios se obtuvo una mejora de alrededor del 20% por sobre el estándar IEEE 802.11g. Es importante señalar, que la aplicación GBEBapp es capaz de sugerir valores Wo y mStages mucho más óptimos incluso conforme crece la población de terminales contendientes.

Notablemente también apreciamos la caída significativa del rendimiento promedio de la red conforme aumenta el número de terminales, este fenómeno no se puede eludir dado que seguimos usando un mecanismo CSMA-CA con BEB como mediador de acceso al medio y en la práctica el ancho de banda crudo del canal inalámbrico es dividido entre el número de terminales incluyendo todas las tramas de control.

A continuación se muestra la Tabla 5.1 donde englobamos los resultados encontrados para los diez experimentos en total realizados. Se puede apreciar la tendencia de mejora que la GBEBapp es capaz de lograr.

DATOS DE ENTRADA: GBEBapp						RESULTADOS OBTENIDOS (PROMEDIOS)			
#exp.	Wo	m_o	mStages ¿fijo?	#RUNS (corridas)	N (terminales)	W	mStages	Rendimiento (bps)	Mejora sobre IEEE 802.11g
1	16	6	NO	20	4	6.1	6.35	1917723	**4.91%**
2	16	6	NO	20	8	8.7	6.85	931307	**7.42%**
3	16	6	NO	20	16	7	7.1	444848	**9.03%**
4	16	6	NO	20	32	126	11.35	208567	**10.35%**
5	16	6	NO	20	64	246	5	102845	**23.91%**
6	16	6	SI	20	4	5.2	6	1915024	**4.76%**
7	16	6	SI	20	8	7	6	905296	**4.42%**
8	16	6	SI	20	16	98	6	419705	**2.87%**
9	16	6	SI	20	32	184	6	208360	**10.24%**
10	16	6	SI	20	64	107	6	100925	**21.60%**

Tabla 5.1 – Resumen de pruebas para simulaciones con 'N' terminales.

Interpretando la información conseguida por los diez experimentos podemos fácilmente concluir que se obtienen mejores resultados cuando se deja el valor 'mStages' para su libre manipulación por parte de la GBEBapp que cuando se deja fijo. En el mejor caso, para 64 terminales contendientes y el valor de mStages no fijo, se obtuvo una mejora de casi el 24% por sobre el estándar IEEE 802.11g; con esto queda demostrada la robustez del programa genético frente a cambios drásticos de población en la red inalámbrica.

El éxito de la optimización inteligente en la red y sus recursos (en este caso, el máximo rendimiento posible alcanzado para la utilización de canal) recae firmemente en la correcta pareja de valores para CWmin y CWmax que nuestro programa arroja representados respectivamente por Wo y mStages.

5.2.1 Gráficas comparativas.

Demostrada la eficacia de la GBEBapp con los experimentos anteriores, ahora nos disponemos a generar diversas gráficas comparativas que ayuden a verificar el alcance de optimización que se puede lograr por sobre el estándar IEEE 802.11g.

Separaremos dichas gráficas en tres secciones: análisis de N vs Rendimiento (kbps), N vs Colisiones y análisis de Rendimiento (kbps) variando los parámetros para el algoritmo genético; en esta última sección se mostrarán resultados para diversas modalidades de uso de la GBEBapp. Para cada sección se harán comentarios pertinentes según lo observado en el comportamiento de la ventana de contención sugerida.

A continuación se muestran dos gráficas de los resultados obtenidos para barridos de N estaciones. La finalidad es la de comparar el redimiendo promedio de la red logrado con el estándar y el logrado con la GBEBapp. En todos los casos se utilizaron los siguiente parámetros: Wo = 16, mo = 6, 10 corridas, M = 15, con criterio de terminación a 500 generaciones.

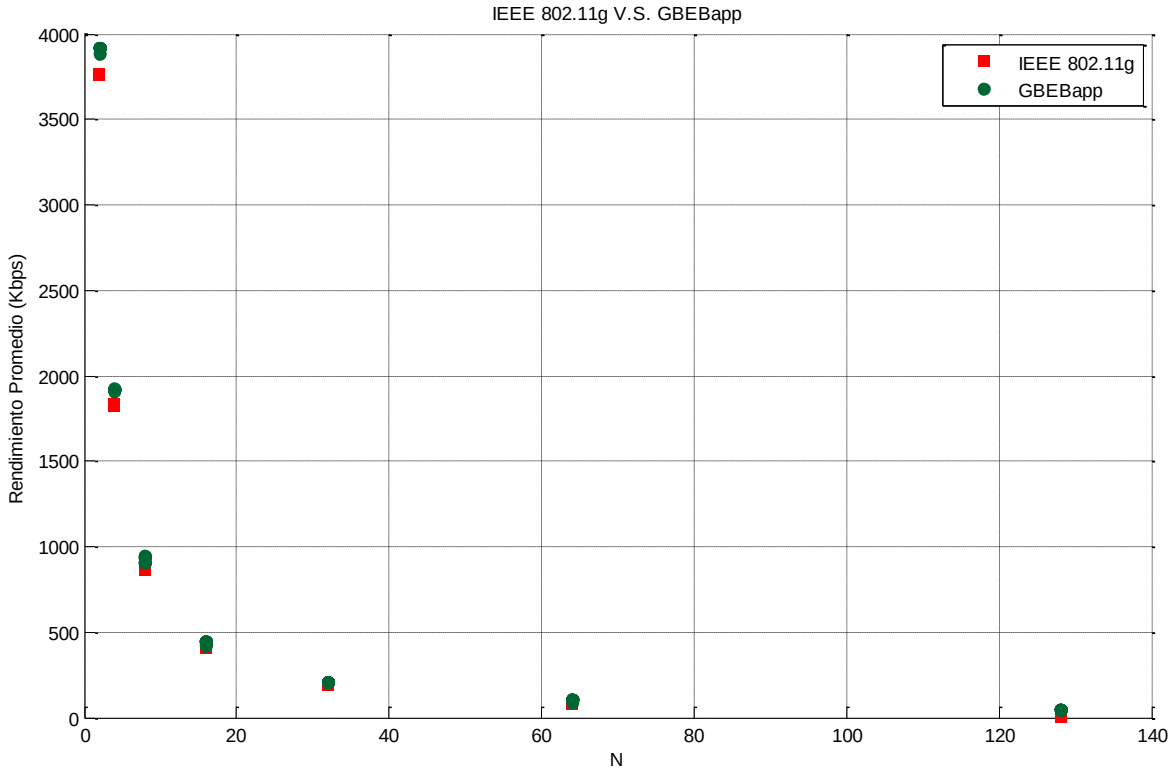


Figura 5.12 – Barrido de N estaciones vs Rendimiento (kbps). N en saltos de potencias de 2.

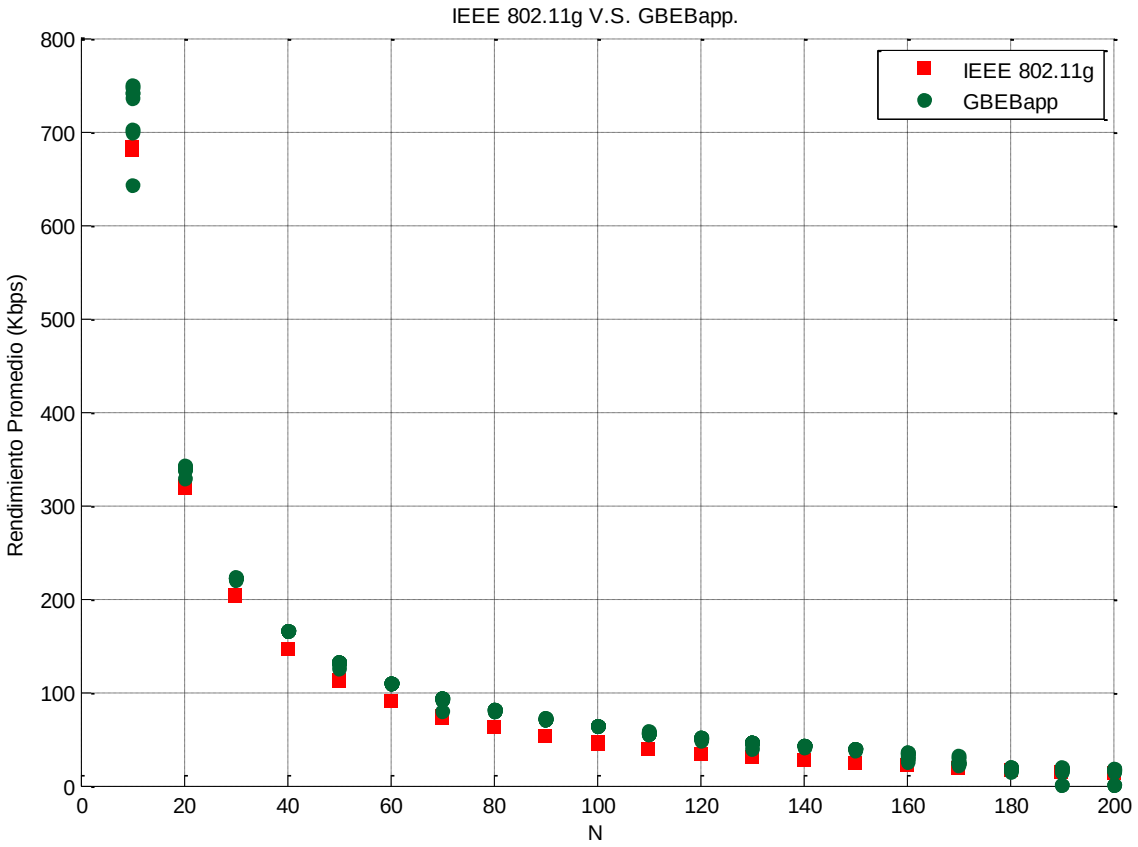


Figura 5.13 – Barrido de N estaciones vs Rendimiento (kbps). N en saltos de 10.

En primera instancia observamos los resultados de la GBEBapp por encima del rendimiento promedio logrado con el estándar IEEE 802.11g, esto sumado a la capacidad del programa genético para arrojar resultados óptimos aumentando el número de estaciones hasta N = 200.

En las Figuras 5.12 y 5.13, que se diferencian únicamente en la escala del eje X (número de terminales), se observan comportamientos similares, conforme aumenta el número de terminales contendientes N, decae el rendimiento promedio de la red (kbps), los resultados arrojados por la GBEBapp intentan que esta caída sea un poco menos drástica.

Para los mejores resultados obtenidos, se logró una mejora de alrededor del 60% sobre el rendimiento alcanzado con el estándar IEEE 802.11g por sí solo. Y se nota que la mejora no decae conforme aumenta el número de terminales. Esto confirma la fiabilidad del programa GBEBapp incluso en entornos de alta congestión.

Para esta sección cabe mencionar dos situaciones observadas durante la recolección de datos.

Primeramente, y como era de esperarse, el programa GBEBapp consume mucho más tiempo de procesamiento conforme se aumenta el número de terminales contendientes, alcanzando hasta 2 horas de procesamiento para lograr las 10 corridas en el caso de N = 200. Dicho fenómeno se debe a la forma en que el programa hace la simulación para cada una de las estaciones y termina hasta

haber transmitido 10,000 paquetes satisfactoriamente, entre más terminales contienden por ocupar el canal inalámbrico, más colisiones se generan y mayor el índice de retransmisión. El mayor porcentaje del total de colisiones generadas se da al inicio de la simulación, donde 200 estaciones compiten por transmitir con un CW inicial de entre 0 a 16 y naturalmente más de una estación llegará a 0 en su *backoff*.

Segundo, cabe destacar las limitaciones del programa respecto al número de terminales contendientes, cuando alcanzamos N = 200, algunos valores eran irregulares y nos indicaban un error global en la función de '*throughput*' de cada individuo en la población. Debido a las mismas limitaciones con la plataforma Java en capacidad de procesamiento y las sentencias de cierre internas incluidas en el programa genético (como un máximo índice de perdida de paquetes del 10%) para evitar un 'loop' infinito; si el programa no es capaz de simular 10,000 paquetes entregados satisfactoriamente en el entorno de red, arrojará entonces un resultado erróneo. Rendimiento = 1 kbps. Dicho fenómeno se observó en algunas corridas con 190 y 200 terminales (Figura 5.13). No es recomendable su uso por encima de esta población de estaciones contendientes. Trabajo adicional se puede realizar para depurar el algoritmo y mejorar la eficiencia de procesamiento con el fin de poder simular un entorno con más de 200 terminales.

Ahora se mostrarán dos gráficas con los resultados obtenidos para barridos de N estaciones contra el número de colisiones simuladas en la red, comparando siempre el estándar y la aplicación GBEBapp. En todos los casos se utilizaron los siguientes parámetros: Wo = 16, mo = 6, 10 corridas, M = 15, con criterio de terminación a 500 generaciones.

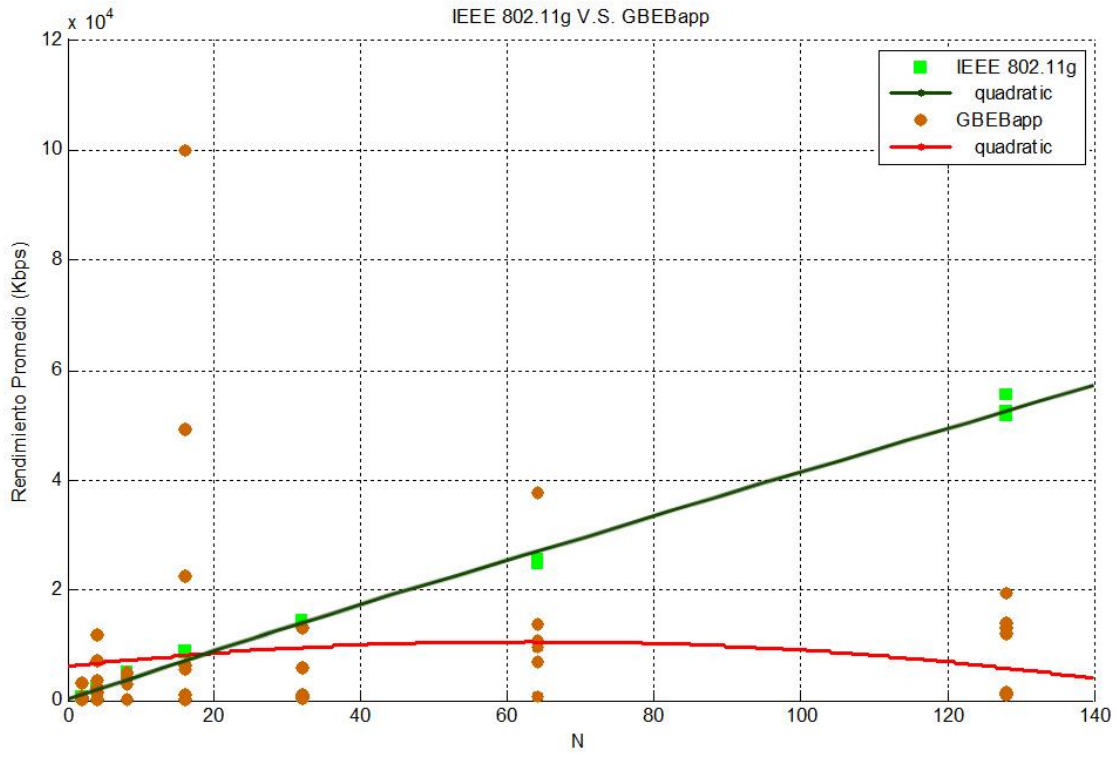


Figura 5.14 – Barrido de N estaciones vs # colisiones. N en saltos de potencias de 2.

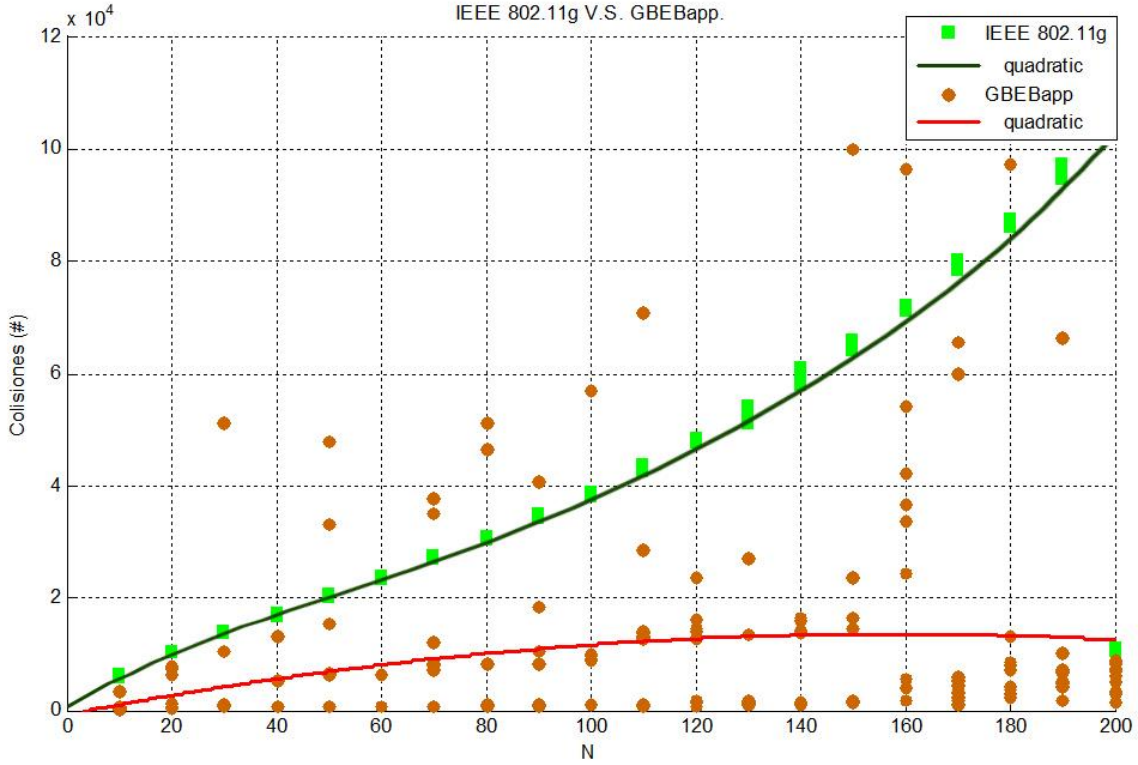


Figura 5.15 – Barrido de N estaciones vs # colisiones. N en saltos de 10.

En las Figuras 5.14 y 5.15 se pueden observar las curvas de aproximación cuadráticas para estimar el promedio del total de colisiones en cada salto de N. Para ambos casos, saltos en potencias de dos y saltos en décadas; se observa la curva del GBEBapp por debajo de la curva del estándar IEEE 802.11g, esto demuestra la capacidad de la aplicación creada para disminuir considerablemente el número de colisiones que ocurren en redes con alta congestión.

El estándar IEEE 802.11g por su naturaleza 'estática' no es capaz de manipular el modo de operación del sistema BEB, por lo tanto, el número de colisiones presentes aumenta casi de manera cuadrática cuando aumenta la población de terminales contendiendo por el canal. Por otro lado, la aplicación GBEBapp trabaja de forma dinámica intentando obtener en cada generación una combinación de valores Wo y mStages capaz de minimizar de gran manera el número de colisiones simuladas.

En ambas figuras son visibles algunos puntos mostrando casos con alto número de colisiones para la GBEBapp, lo cual es evidencia que en algunas iteraciones el algoritmo no converge en la mejor manera posible debido a su naturaleza aleatoria, y por tal motivo es recomendable realizar al menos 10 corridas por cada caso que se desee analizar, tal y como lo hemos realizado en las pasadas pruebas.

Para la tercera y última sección se realizaran diversas pruebas con parámetros del GBEBapp distintos, siempre buscando maximizar el rendimiento promedio de la red en situación de congestión por N terminales, recordemos que la aplicación está diseñada para simular un entorno donde todas las estaciones tienen todo el tiempo un paquete que desean transmitir y utilizan el mecanismo RTS-CTS. Para todos los casos se realizarán diez corridas barriendo desde N = 10 hasta N = 100. Las configuraciones presentadas son las siguientes.

- Débil. W inicial = 16, m inicial = 6, M = 3, # generaciones = 50.
- Estricto. W inicial = 16, m inicial = 6, M = 25, % change = 0.001.
- mStages fijo. W inicial = 16, m inicial = 6, M = 7, # generaciones = 500, valor mStages fijo.
- Disperso. W inicial = 64, m inicial = 18, M = 15, % change = 0.01.

En cada caso se graficarán los valores sugeridos de Wo, mStages y rendimiento promedio para comparar contra el estándar IEEE 802.11g. A continuación los resultados para las simulaciones.

El primer análisis corresponde a corridas con una configuración 'débil' del algoritmo genético.

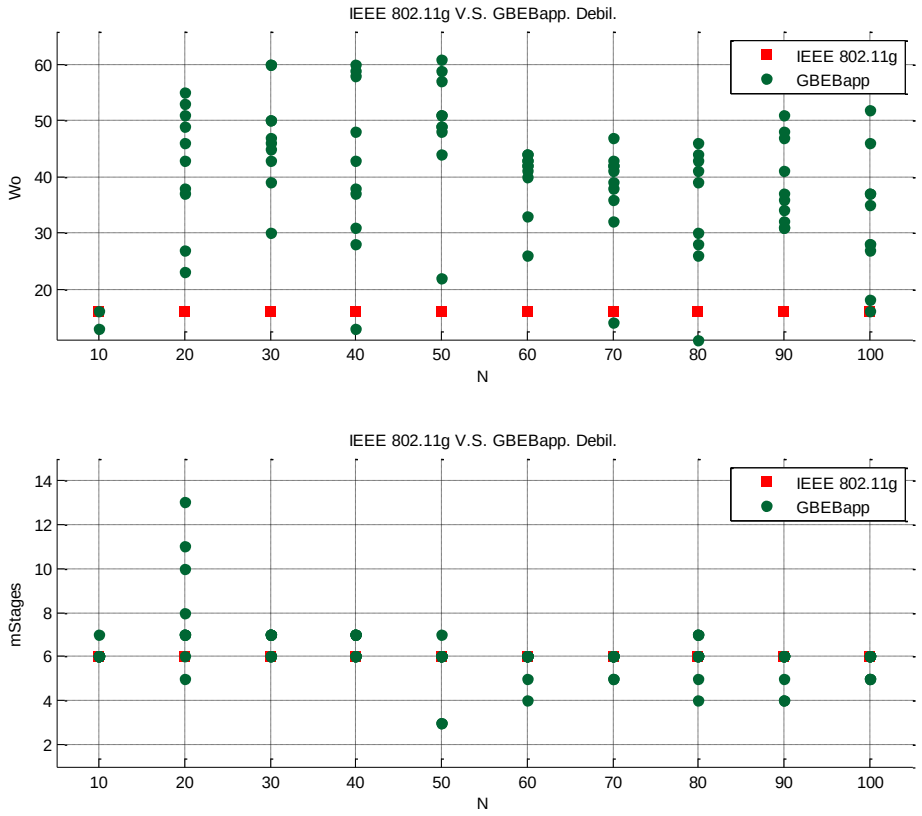


Figura 5.16 – Débil. Se observan los valores W y mStages sugeridos.

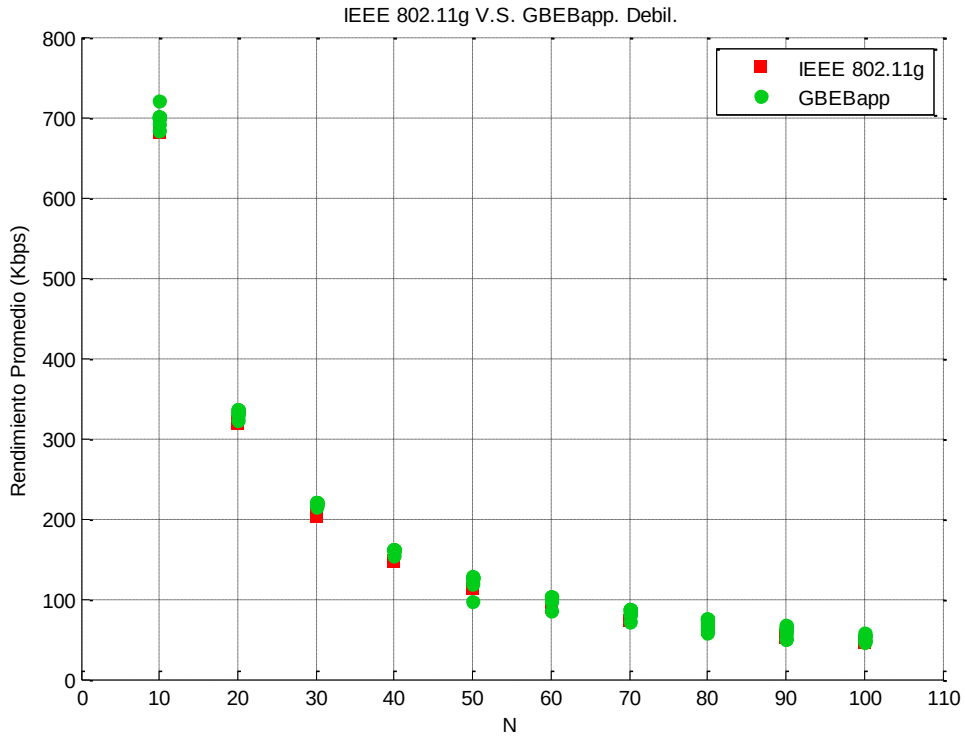


Figura 5.17 – Débil. N vs Rendimiento (kbps).

En las Figuras 5.16 y 5.17 podemos observar el bajo rendimiento demostrado por la aplicación cuando se utilizan parámetros para una búsqueda 'débil', es decir, los criterios para la selección de los mejores individuos en la población de cada generación durante el algoritmo genético son mínimos. Es importante señalar dos puntos; primero, con un criterio débil de búsqueda las iteraciones serán menos exigentes y por tanto se reduce el tiempo de procesamiento de la GBEBapp, sin embargo los resultados son poco óptimos en mejora respecto al estándar IEEE 802.11g. El segundo punto a destacar es la falta de una convergencia para los valores de Wo y mStages sugeridos. La dispersión se muestra mayormente con los valores de Wo.

Sin embargo, y a pesar de utilizar la GBEBapp de la forma menos óptima posible, los valores de rendimiento promedio de la misma, sobrepasan ligeramente los logrados por el estándar IEEE 802.11g por sí solo.

El segundo análisis corresponde a corridas para una configuración 'estricta' del algoritmo genético.

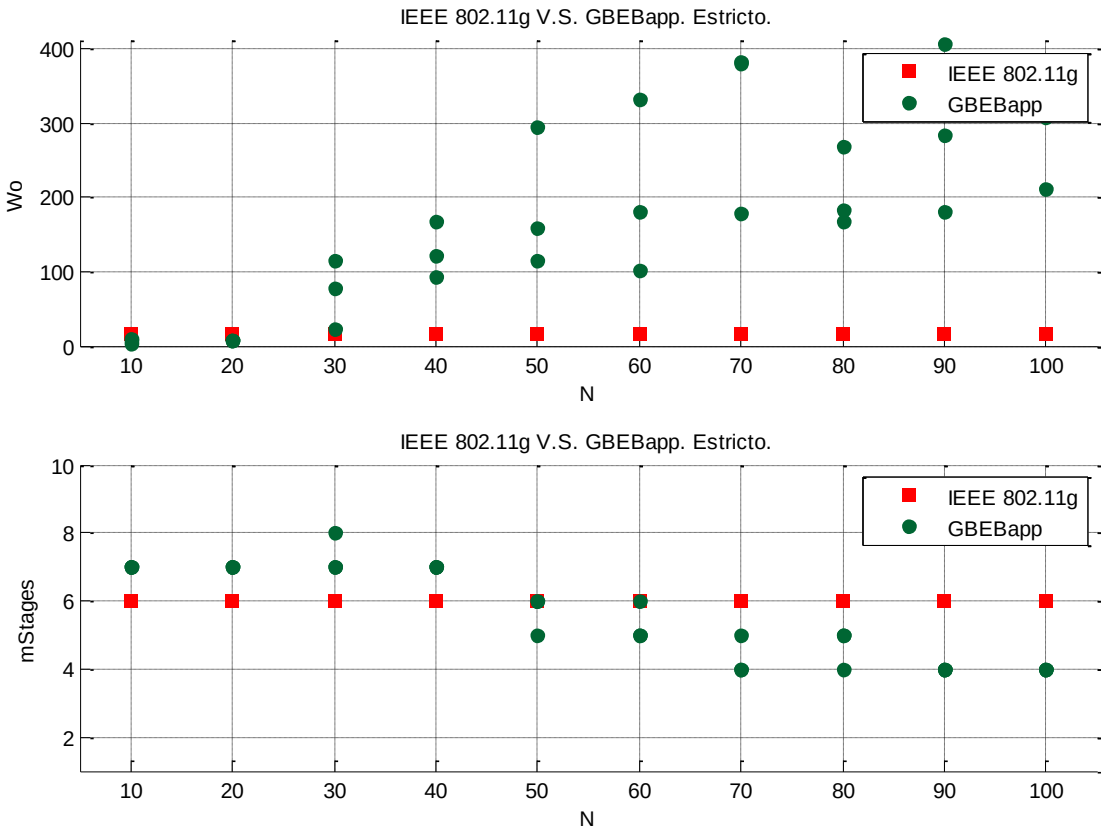


Figura 5.18 – Estricto. Se observan los valores W y mStages sugeridos.

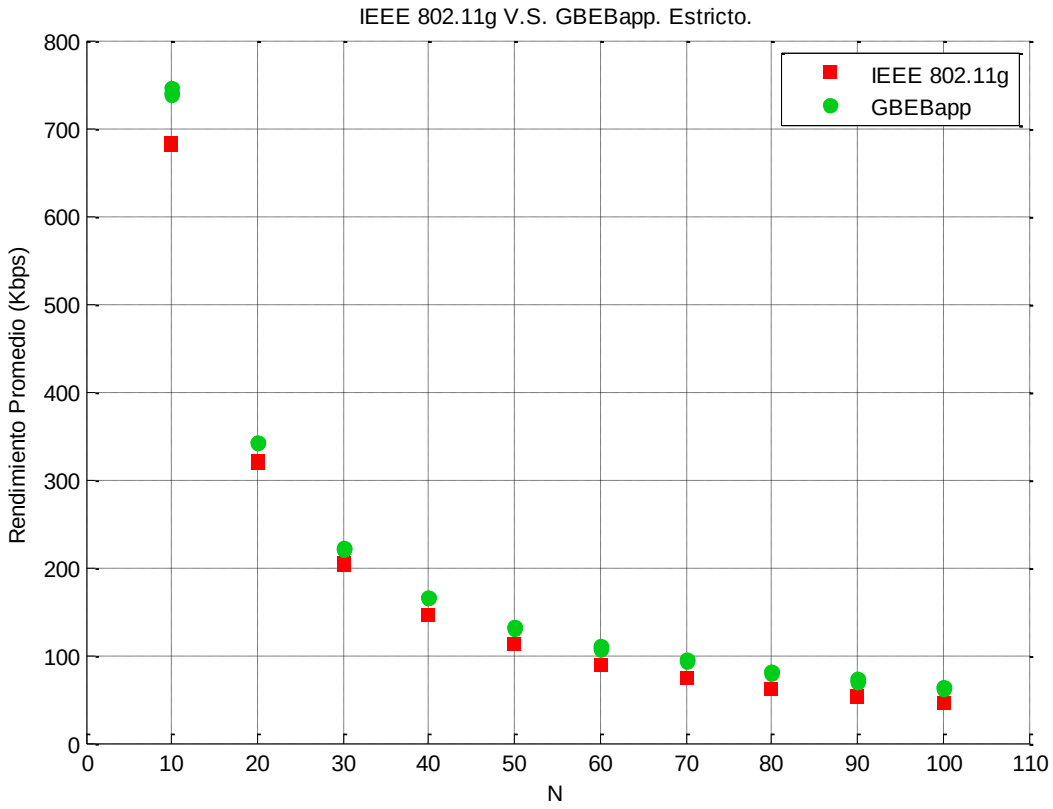


Figura 5.19 – Estricto. N vs Rendimiento (kbps).

En las Figuras 5.18 y 5.19 podemos observar el comportamiento de la aplicación con criterios estrictos para el algoritmo genético, es decir; con valores superiores para encontrar a los mejores individuos de cada generación iterada.

De primera instancia tenemos que destacar que el tiempo de procesamiento fue mucho más elevado que con el experimento pasado (débil) y por lo mismo, sólo se realizaron 3 corridas para cada uno de los saltos de N. Dichas mediciones fueron suficientes para determinar una posible convergencia en los valores sugeridos por la GBEBapp.

Los valores de Wo tendieron a subir conforme aumentaba la población de estaciones contendientes, mientras que el valor de mStages permaneció entre valores de 4 a 8. Finalmente en la Figura 5.19 podemos distinguir fácilmente la mejora lograda de los resultados GBEBapp por sobre le estándar IEEE 802.11g a pesar del alto tiempo de procesamiento por parte de la GBEBapp, los valores de rendimiento promedio de la misma, sobrepasan los logrados por el estándar IEEE 802.11g por sí solo.

A continuación el análisis correspondiente a corridas con criterios promedio del algoritmo genético pero con el valor de mStages fijo.

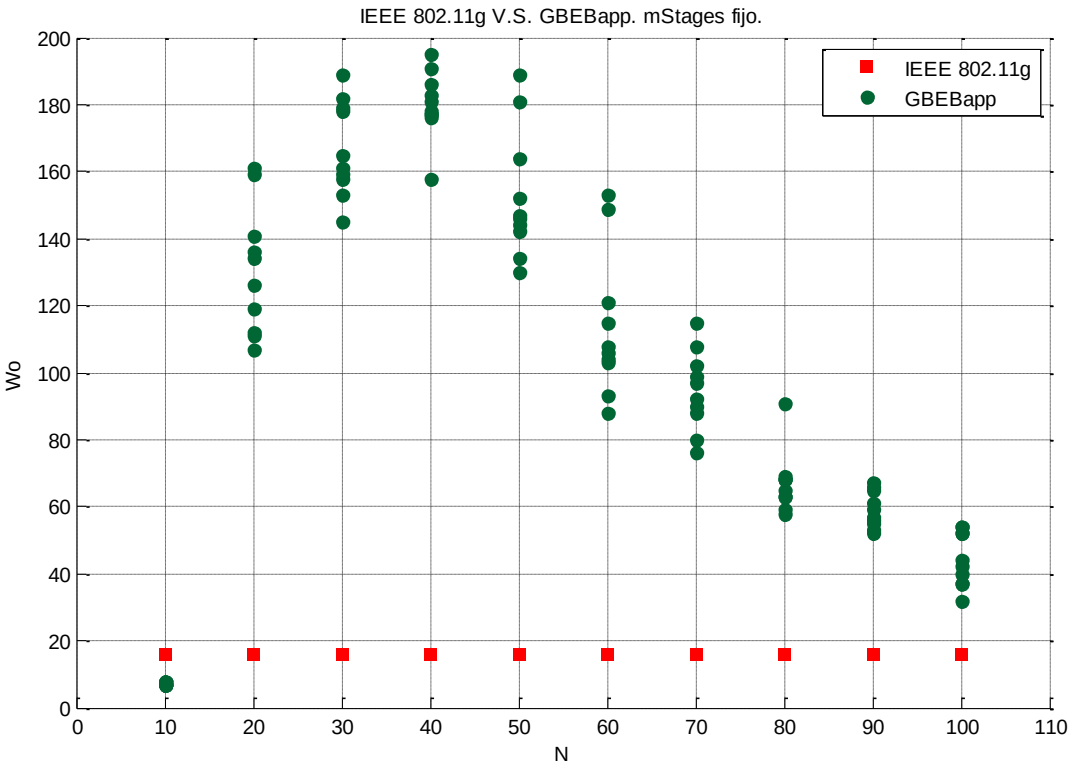


Figura 5.20 – mStages. Valores sugeridos Wo vs Rendimiento (kbps).

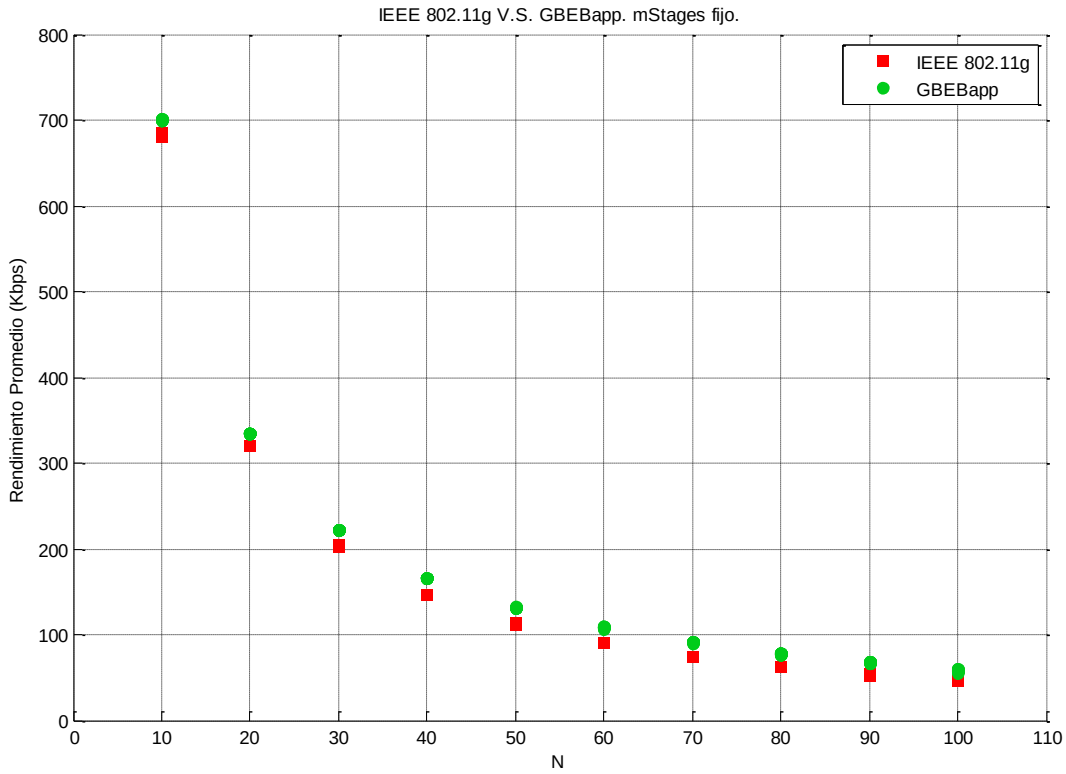


Figura 5.21 – mStages fijo. N vs Rendimiento (kbps).

Para el caso donde dejamos el valor de mStages fijo lo que se pretendía era observar el comportamiento del valor Wo en la ventana de contención inicial, es decir; cómo se comporta el mecanismo BEB frente a poblaciones en aumento de terminales compitiendo por acceso al canal. El valor de mStages implica el número de 'escalones' que la ventana podrá iterarse en potencias de dos, con esto, fijamos un valor de CWmax en función de su valor inicial Wo/CWmin. Suponemos entonces, que el mantener este valor estático debería tener impacto en el rendimiento promedio logrado para la red inalámbrica.

En la Figura 5.20 se puede observar un comportamiento muy peculiar respecto a los valores arrojados de Wo. Primeramente para el caso de N = 10, los valores de Wo oscilan entre 6 y 8, debajo del valor para el estándar que es de 16. Sin embargo, cuando la población de terminales salta de N = 20 en adelante, los valores arrojados de Wo son mayores a 16. Esto puede verse como la forma en que el algoritmo genético pretende compensar el hecho de que no puede variar el valor de mStages y por tanto se reducen sus parámetros para 'mutar' a las generaciones.

En la incapacidad de variar el valor de mStages, el algoritmo sugiere valores elevados y dispersos de Wo que no parecen converger, sin embargo, no necesariamente son valores erróneos. Esto demuestra una cualidad adicional de la GBEBapp frente a situaciones donde algunos parámetros no pudieran ser modificados como el caso del valor de CWmax.

A pesar de la limitante visible que el algoritmo sufre con el valor fijo de mStages, podemos observar en la Figura 5.21 que el rendimiento promedio logrado con la GBEBapp es ligeramente superior en todos los casos a lo simulado con el estándar IEEE 802.11g por sí solo. Este tercer experimento presenta valores de mejora por encima de los logrados con la configuración 'débil' pero aún por debajo de los logrados con la configuración 'estricta'.

Finalmente se presenta el análisis correspondiente a corridas con criterios 'dispersos' para el algoritmo genético en la GBEBapp.

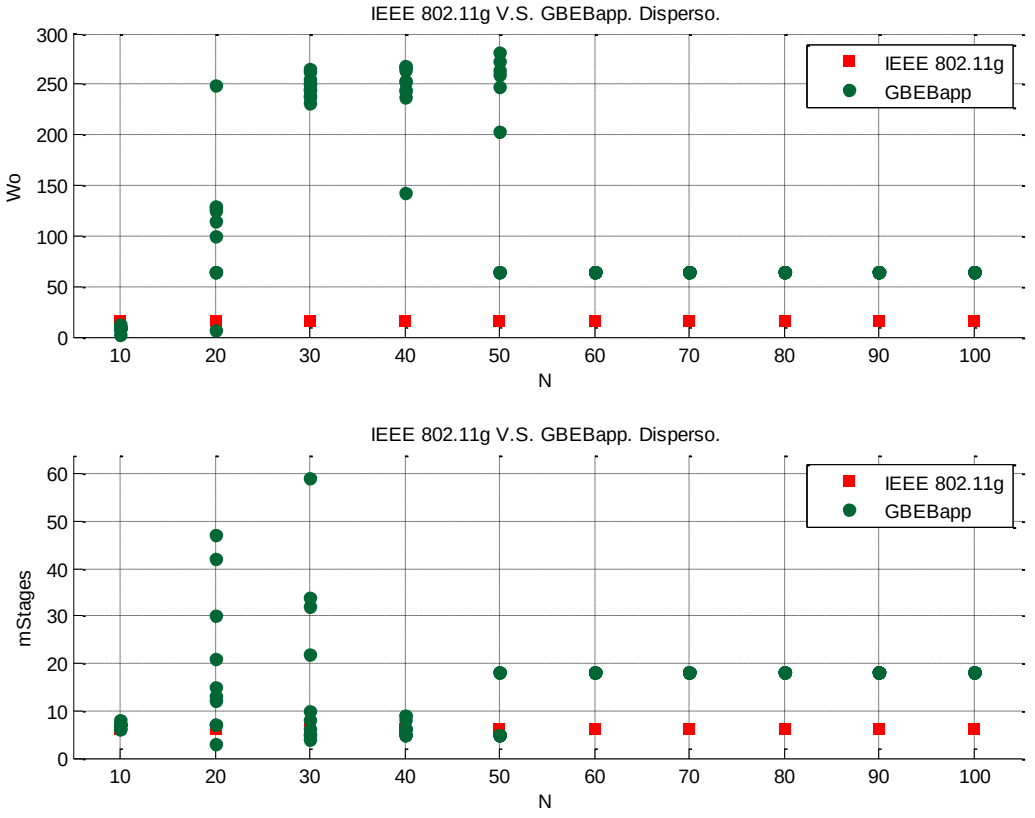


Figura 5.22 – Disperso. Se observan los valores W y mStages sugeridos.

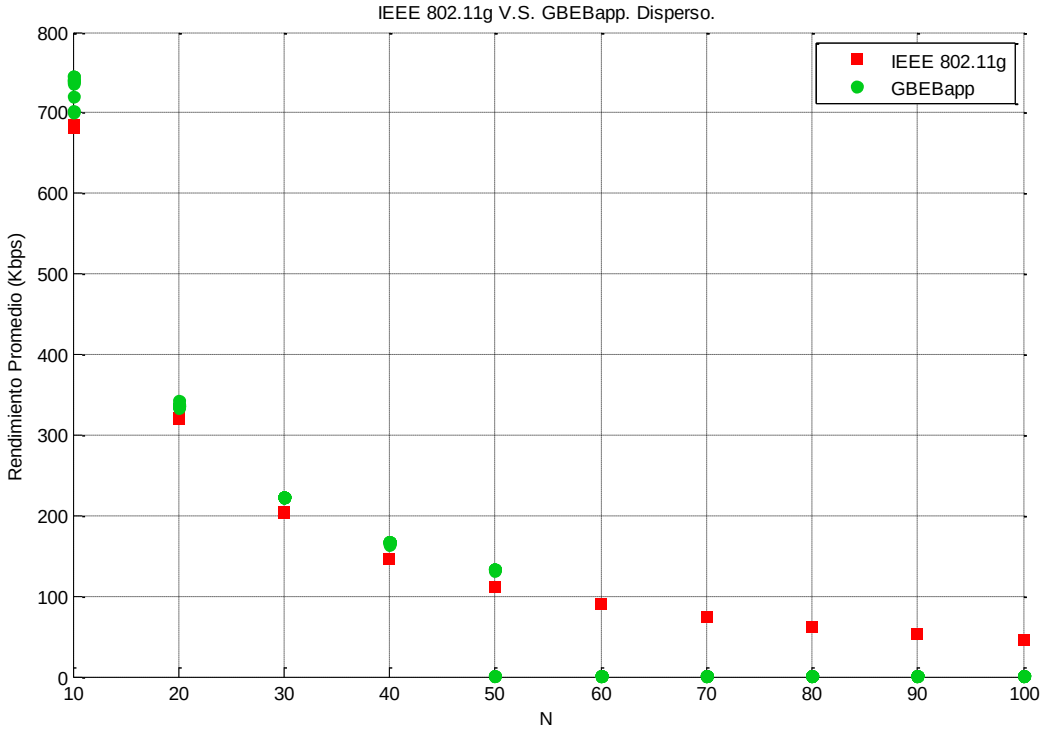


Figura 5.23 – Disperso. N vs Rendimiento (kbps).

Para el último escenario propuesto se llevó la aplicación GBEBapp a los límites, es decir, se estresó su capacidad de procesar, simular y generar valores de Wo y mStages que mejoraran el rendimiento promedio de la red. Dicha configuración se logró con valores de W y mStages iniciales fuera de lo convencional. Datos de entrada 'dispersos'.

Como se puede apreciar en la Figuras 5.22 los resultados fueron interesantes al exponer las capacidades de procesamiento de la aplicación. En un inicio, los valores sugeridos de Wo parecen comportarse similar al caso de la configuración 'mStages fijo' pero después de N = 50 (cincuenta terminales contendientes en el canal inalámbrico), los valores de Wo se quedan fijos en 64 que es el valor de los datos de entrada iniciales. Mismo caso sucede para los valores de mStages arrojados. En otras palabras, el programa parece 'romperse' o llegar a su límite cuando nos acercamos a 50 terminales.

Lo anterior se corrobora con la Figura 5.23. El rendimiento promedio es mejorado para el caso de N < 50 terminales pero de inmediato se corrompe el funcionamiento óptimo del programa genético y la GBEBapp arroja resultados nulos cuando superamos este umbral (puntos en cero para N >= 50).

Esta última prueba nos ayuda a delimitar las capacidades de la aplicación GBEBapp y concluimos que para su correcto desempeño no basta con escoger un criterio 'Estricto' sino que los datos de entrada deben ser lo más reales posible para no caer en un escenario 'disperso', donde las iteraciones del algoritmo genético no convergen en valores que maximicen el rendimiento promedio.

5.3 Cuadro resumen.

Finalmente se presenta un cuadro resumen con la información clave recabada en las pruebas de los Capítulos 4 y 5, donde se incluyen las Tablas 4.1 y 5.1. La intención no es otra más que la de evidenciar las capacidades de mejora que presenta el algoritmo y la aplicación en Java GBEBapp desarrollados a lo largo de este documento. Para la gran mayoría de los escenarios aprovisionados se mostraron resultados por encima de lo que obtenemos con el estándar IEEE 802.11g por sí solo y en lo que respecta al aprovechamiento del canal en redes inalámbricas con N terminales contendientes bajo el esquema RTS-CTS.

Para los campos nombrados 'promedio' se hizo la media matemática para el número de corridas totales en el experimento mencionado. Es el caso, por ejemplo, del # de colisiones observadas para los experimentos con N terminales. En los últimos experimentos se hizo la segregación de N que va desde 10 hasta 100. Para el caso particular del experimento con salto en décadas (Figura 5.13) con N = 10 hasta 200, se muestran sólo los datos para la mitad de saltos con el fin de no hacer más grande el cuadro resumen.

Experimento	Figura	Wo	m_o	N (terminales)	M (población)	Criterio de Terminación	Tiempo procesamiento	# Colisiones promedio	Rendimiento IEEE 802.11g (bps)	Rendimiento GBEBapp Promedio (bps)	Mejora sobre IEEE 802.11g
Tabla 4.1 - #1	4.5	16	6	2	3	# gens = 500	Bajo	2437.5	3769638.5	3881219.8	2.96%
Tabla 4.1 - #2	4.6	16	6	2	15	# gens = 500	Medio	3280.6	3769490.1	3915746.4	3.88%
Tabla 4.1 - #3	4.7	16	6	2	25	# gens = 500	Alto	652.2	3769528.0	3923324.8	4.08%
Tabla 4.1 - #4	4.8	16	6	2	15	# gens = 20	Bajo	1265.7	3769603.4	3911340.5	3.76%
Tabla 4.1 - #5	4.9	16	6	2	15	# gens = 200	Bajo	5660.1	3769588.1	3916979.0	3.91%
Tabla 4.1 - #6	4.10	16	6	2	15	# gens = 1000	Alto	772.2	3769624.9	3919656.0	3.98%
Tabla 4.1 - #7	4.11	16	6	2	15	más fuerte = 0.01%	Alto	531.6	3769929.2	3914694.5	3.84%
Tabla 5.1 - #1	5.2	16	6	4	25	# gens = 500	Medio	2226.6	1827969.7	1917723.3	4.91%
Tabla 5.1 - #2	5.3	16	6	8	25	# gens = 500	Medio	8270.9	866977.3	931307.5	7.42%
Tabla 5.1 - #3	5.4	16	6	16	25	# gens = 500	Medio	6443.5	408005.1	444848.1	9.03%
Tabla 5.1 - #4	5.5	16	6	32	25	# gens = 500	Medio	1285.2	189005.0	208567.4	10.35%
Tabla 5.1 - #5	5.6	16	6	64	25	# gens = 500	Medio	4707.2	82999.8	102845.8	23.91%
Tabla 5.1 - #6	5.7	16	6 fijo	4	25	# gens = 500	Medio	2223.0	1828010.7	1915024.4	4.76%
Tabla 5.1 - #7	5.8	16	6 fijo	8	25	# gens = 500	Medio	5216.2	866975.7	905296.1	4.42%
Tabla 5.1 - #8	5.9	16	6 fijo	16	25	# gens = 500	Medio	4546.8	407995.5	419705.2	2.87%
Tabla 5.1 - #9	5.10	16	6 fijo	32	25	# gens = 500	Medio	2844.3	189005.8	208360.6	10.24%
Tabla 5.1 - #10	5.11	16	6 fijo	64	25	# gens = 500	Medio	2471.3	82997.5	100925.5	21.60%

Experimento	Figura	Wo	m_o	N (terminales)	M (población)	Criterio de Terminación	Tiempo procesamiento	# Colisiones promedio	Rendimiento IEEE 802.11g (bps)	Rendimiento GBEBapp Promedio (bps)	Mejora sobre IEEE 802.11g
Satos de potencia 2	5.12	16	6	2	15	# gens = 500	Bajo	573.1	3764535.3	3917881.0	4.07%
Satos de potencia 2	5.12	16	6	4	15	# gens = 500	Bajo	3321.9	1828451.1	1916579.4	4.82%
Satos de potencia 2	5.12	16	6	8	15	# gens = 500	Bajo	3802.0	869052.3	926669.6	6.63%
Satos de potencia 2	5.12	16	6	16	15	# gens = 500	Medio	10131.0	409177.9	440517.5	7.66%
Satos de potencia 2	5.12	16	6	32	15	# gens = 500	Medio	2469.1	189110.1	208216.2	10.10%
Satos de potencia 2	5.12	16	6	64	15	# gens = 500	Medio	8236.9	82965.9	101543.0	22.39%
Satos de potencia 2	5.12	16	6	128	15	# gens = 500	Alto	6560.8	31294	47389.5	**51.43%**
Salto en décadas	5.13	16	6	20	15	# gens = 500	Bajo	3332.2	319819.9	338913.5	5.97%
Salto en décadas	5.13	16	6	40	15	# gens = 500	Medio	2804.6	146408.3	166023.3	13.40%
Salto en décadas	5.13	16	6	80	15	# gens = 500	Medio	2400.4	89989.7	109718.0	21.92%
Salto en décadas	5.13	16	6	100	15	# gens = 500	Alto	11186.7	62109.9	80882.4	30.22%
Salto en décadas	5.13	16	6	120	15	# gens = 500	Alto	9945.4	45684.9	63933.6	39.94%
Salto en décadas	5.13	16	6	140	15	# gens = 500	Alto	11073	34682.9	50357.8	**45.19%**
Salto en décadas	5.13	16	6	150	15	# gens = 500	Alto	6844.1	26968.0	42862.4	**58.94%**
Salto en décadas	5.13	16	6	160	15	# gens = 500	Alto	16487.9	23925.0	39000.0	**63.01%**
Salto en décadas	5.13	16	6	180	15	# gens = 500	Alto	30009.2	21252.4	31810.3	**49.68%**
Salto en décadas	5.13	16	6	200	15	# gens = 500	Alto	15476.5	16873.7	17997.0	6.66%

Experimento	Figura	Wo	m_o	N (terminales)	M (población)	Criterio de Terminación	Tiempo procesamiento	# Colisiones promedio	Rendimiento IEEE 802.11g (bps)	Rendimiento GBEBapp Promedio (bps)	Mejora sobre IEEE 802.11g
Débil	5.17	16	6	10	3	# gens = 50	Bajo	7266.3	682149.1	699362.9	2.52%
Débil	5.17	16	6	20	3	# gens = 500	Bajo	6658.3	319921.1	332110.6	3.81%
Débil	5.17	16	6	30	3	# gens = 500	Bajo	6922.4	203482.2	218295.4	7.28%
Débil	5.17	16	6	40	3	# gens = 500	Bajo	15933.1	146515.6	160280.8	9.40%
Débil	5.17	16	6	50	3	# gens = 500	Bajo	9711.7	112402.9	122483.0	8.97%
Débil	5.17	16	6	60	3	# gens = 500	Bajo	14989.8	90033.1	99353.2	10.35%
Débil	5.17	16	6	70	3	# gens = 500	Bajo	16876.5	74057.6	83420.6	12.64%
Débil	5.17	16	6	80	3	# gens = 500	Bajo	22924.0	62221.8	69298.0	11.37%
Débil	5.17	16	6	90	3	# gens = 500	Bajo	7967.0	52958.5	59782.5	12.89%
Débil	5.17	16	6	100	3	# gens = 500	Bajo	15759.1	45655.1	51668.6	13.17%
Estricto	5.19	16	6	10	25	más fuerte = 0.001%	Medio	31267.0	681736.7	741743.7	8.80%
Estricto	5.19	16	6	20	25	más fuerte = 0.001%	Medio	5216.0	319569.7	342244.0	7.10%
Estricto	5.19	16	6	30	25	más fuerte = 0.001%	Medio	681.7	203562.7	221799.7	8.96%
Estricto	5.19	16	6	40	25	más fuerte = 0.001%	Alto	644.3	146391.0	165866.0	13.30%
Estricto	5.19	16	6	50	25	más fuerte = 0.001%	Alto	3757.7	112433.3	132025.0	17.43%
Estricto	5.19	16	6	60	25	más fuerte = 0.001%	Alto	4937.7	89946.0	108996.7	21.18%
Estricto	5.19	16	6	70	25	más fuerte = 0.001%	Alto	4422.7	74019.3	93736.0	26.64%
Estricto	5.19	16	6	80	25	más fuerte = 0.001%	Alto	3427.0	62138.0	80543.0	29.62%

Estricto	5.19	16	6	90	25	más fuerte = 0.001%	Alto	3426.7	52909.7	71130.7	34.44%
Estricto	5.19	16	6	100	25	más fuerte = 0.001%	Alto	997.7	45501.0	63634.7	39.85%
mStages fijo	5.21	16	6 fijo	10	7	# gens = 500	Medio	6197.6	682089.2	700536.2	2.70%
mStages fijo	5.21	16	6 fijo	20	7	# gens = 500	Medio	3307.7	320253.5	334615.6	4.48%
mStages fijo	5.21	16	6 fijo	30	7	# gens = 500	Medio	3172.0	203457.9	222405.9	9.31%
mStages fijo	5.21	16	6 fijo	40	7	# gens = 500	Medio	2436.0	146311.9	166161.7	13.57%
mStages fijo	5.21	16	6 fijo	50	7	# gens = 500	Medio	710.0	112351.3	131706.0	17.23%
mStages fijo	5.21	16	6 fijo	60	7	# gens = 500	Alto	1580.9	90004.2	108221.6	20.24%
mStages fijo	5.21	16	6 fijo	70	7	# gens = 500	Alto	1904.5	74031.6	91392.7	23.45%
mStages fijo	5.21	16	6 fijo	80	7	# gens = 500	Alto	3402.8	62198.8	77643.0	24.83%
mStages fijo	5.21	16	6 fijo	90	7	# gens = 500	Alto	4006.1	52999.8	67609.8	27.57%
mStages fijo	5.21	16	6 fijo	100	7	# gens = 500	Alto	3404.4	45707.2	58089.8	27.09%
Disperso	5.23	64	18	10	15	más fuerte = 0.01%	Bajo	3610.0	681690.1	731142.2	7.25%
Disperso	5.23	64	18	20	15	más fuerte = 0.01%	Bajo	2259.1	319933.0	336618.4	5.22%
Disperso	5.23	64	18	30	15	más fuerte = 0.01%	Bajo	2567.7	203582.3	222564.4	9.32%
Disperso	5.23	64	18	40	15	más fuerte = 0.01%	Medio	1991.7	146316.7	166400.6	13.73%
Disperso	5.23	64	18	50	15	más fuerte = 0.01%	Medio	677.3	112427.1	79559.3	-29.23%
Disperso	5.23	64	18	60	15	más fuerte = 0.01%	Medio	880.7	89882.8	1.0	-100.00%
Disperso	5.23	64	18	70	15	más fuerte = 0.01%	Medio	942.6	74100.8	1.0	-100.00%
Disperso	5.23	64	18	80	15	más fuerte = 0.01%	Alto	999.7	62139.3	1.0	-100.00%
Disperso	5.23	64	18	90	15	más fuerte = 0.01%	Alto	1062.0	52938.6	1.0	-100.00%
Disperso	5.23	64	18	100	15	más fuerte = 0.01%	Alto	1081.9	45641.2	1.0	-100.00%

Tabla 5.2 – Cuadro Resumen.

En términos de lo simulado a lo largo de diversos experimentos podemos observar la mejora lograda en el rendimiento promedio con los valores sugeridos por la aplicación GBEBapp para el diseño de la ventana de contención con un escenario RTS-CTS en redes inalámbricas sobre lo simulado con el estándar IEEE 802.11g por sí solo, la mejora lograda va en porcentajes que van desde el 2.5% hasta el 63%.

Para finalizar el capítulo, sería prudente mencionar algunas observaciones generales de la información recopilada en el Cuadro Resumen. Éstas son:

- La aplicación GBEBapp ha demostrado lograr una mejora de eficiencia para el aprovechamiento del canal inalámbrico por sobre el estándar IEEE 802.11g. Dicha mejora se logra cuando se utilizan los valores de Wo (CWmin) y mStages (CWmax) sugeridos por una búsqueda inteligente (algoritmo genético).
- Durante el desarrollo de los experimentos se hizo un hallazgo sobresaliente acerca del rendimiento de la aplicación GBEBapp; se demostró que la mejora para la utilización del canal no se ve degradada conforme aumentamos el número de terminales contendientes; por el contrario, fue con un alto número de terminales (N entre 140 y 180) cuando el programa genético logró mejoras por arriba del 40%.
- A pesar de lo señalado en el punto anterior, cabe mencionar las limitaciones encontradas también. Como se corroboró, la GBEBapp no es capaz de arrojar resultados fiables para escenarios con 200 terminales o más. Otra limitante del mismo estilo es con una configuración ‘dispersa’. Si no se brindan valores lo más cercanos a la realidad como datos de entrada, los valores sugeridos no poseerán sentido alguno (casos en el cuadro resumen con mejoras del -100%).
- Finalmente cabe resaltar los altos tiempos de procesamiento que puede consumir la aplicación GBEBapp cuando configuramos un criterio de terminación estricto y/o cuando se simula un alto número de terminales contendientes.

VI - CONCLUSIONES.

En el desarrollo del presente trabajo se presentó una solución innovadora para mitigar el problema del bajo rendimiento obtenido en redes inalámbricas con el estándar IEEE 802.11g. Se hizo énfasis en las desventajas que presentaba el mecanismo BEB con valores estáticos para la ventana de contención. Nuestro objetivo inicial fue tratado como un problema de optimización, y por lo tanto se recurrió a un modelo de mejora basado en algoritmos genéticos. Se desarrolló un programa Java (GBEBapp) con capacidades para mejorar el rendimiento promedio en una red inalámbrica simulada con N terminales bajo el mecanismo BEB con modelo CSMA/CA al sugerir mejores valores de configuración de la ventana de contención.

La versatilidad de los algoritmos genéticos para 'mutar' constantemente un 'campo de búsqueda' con múltiples opciones, permitió desarrollar una aplicación inteligente a partir del pseudocódigo propuesto. La aplicación programada se comportó de forma eficiente y presentó pocas deficiencias que se explicarán más adelante.

Se sometió al programa a pruebas de resistencia y fiabilidad de los datos procesados de salida. Las conclusiones generales de dichas pruebas nos llevaron a determinar que era mejor utilizar un criterio de terminación complejo (alto número de generaciones o un cambio mínimo porcentual en el individuo más fuerte) con una población extensa (M); sin embargo, para corridas complejas, se necesitó un tiempo elevado de procesamiento (de hasta 2 horas).

La aplicación responde correctamente a cualquier configuración de entrada, pero sus resultados tienden a ser mejores con una configuración 'realista'. La aplicación realiza sus tareas de optimización arrojando, en un 95% de los casos, valores para la ventana de contención que logran un rendimiento promedio por encima de lo obtenido con el estándar IEEE 802.11g; demostrando que el algoritmo genético se ha implementado satisfactoriamente en la aplicación GBEBapp. Los valores de salida de la aplicación varían ligeramente en cada corrida debido a la naturaleza aleatoria de las 'mutaciones' en los individuos de cada generación.

Como se observa en el Cuadro Resumen, se obtuvieron mejoras de hasta el 60% con una media de alrededor del 40% para configuraciones con hasta 200 terminales. La eficiencia de la aplicación GBEBapp logró una mejora considerable en redes inalámbricas con N estaciones contendientes (pudiendo ser *'hotspots'* o entornos públicos con cientos de usuarios entrando y saliendo del área de servicio de un *'access point'*).

El estándar IEEE 802.11g demostró ser óptimo para escenarios con pocas terminales compitiendo por acceso al medio (alrededor de 10 terminales). Ésta situación se hizo notar bajo las pruebas con hasta 200 estaciones contendientes. Los valores para la ventana de contención del estándar IEEE 802.11g están lejos de ser eficientes y presentan altas pérdidas (tiempos largos de espera y múltiples colisiones sucesivas). En respuesta, la aplicación GBEBapp mitigó esta situación al sugerir valores de Wo mayores conforme crecía N. Al final, la tendencia fue ahorrarnos tiempo con una ventana inicial CW más amplia y que provocara menos colisiones.

La aplicación GBEBapp no cambió el mecanismo BEB, los saltos siguen siendo binarios y la selección de los valores de *'backoff'* para la ventana de contención de una terminal deseando transmitir es aleatoria. La esencia del algoritmo presentado es entonces, la obtención de valores iniciales CWmin y CWmax tomando en cuenta el número de terminales. La idea es ayudar al estándar IEEE 802.11g a tomar decisiones más inteligentes en escenarios de contención y, dejar atrás la configuración estática que se ha manejado desde sus inicios.

El producto terminado de la investigación se plasmó con el diseño del algoritmo genético implementado en la aplicación Java GBEBapp y demostró ser una herramienta muy útil para cualquier administrador de red que desee optimizar su rendimiento con escenarios de alta congestión por múltiples terminales contendientes.

Finalmente se hace mención de dos hallazgos encontrados durante el desarrollo del presente trabajo, las limitantes de la aplicación y el trabajo futuro sugerido para el lector.

Las limitantes observadas van en función de las capacidades de la aplicación para arrojar resultados fiables cuando los datos de entrada son dispersos o poco realistas. En el último experimento planteado en el Capítulo V, se hizo evidente que la aplicación no era capaz de procesar soluciones acertadas con 50 o más terminales (configuración 'dispersa').

Otra limitante importante de la GBEBapp es el tiempo de procesamiento que le toma realizar varias corridas para su posterior análisis. Como se observa en el Cuadro Resumen, en la mayoría de los experimentos el tiempo real de espera para que el algoritmo procese todas las generaciones fue alto. Conforme más estrictos se piensan los parámetros de entrada, incluyendo el criterio de terminación, y conforme es mayor el número de terminales contendientes, mayor es el tiempo de procesamiento requerido.

Con base en las limitantes y a oportunidades de mejora detectadas que ayudarían a mejorar el alcance de la aplicación, se proponen los siguientes puntos como trabajo futuro al lector.

Primeramente se puede trabajar en la programación de la aplicación en Java para depurar las clases y/o funciones principales que rigen al algoritmo genético para disminuir los tiempos de procesamiento; siempre respetando los propósitos originales de la búsqueda inteligente.

Para mejorar el alcance de la aplicación convendría agregar algunas características como lo son la definición manual del tamaño del paquete a transmitir, el número de transmisiones a simular y el *'threshold'* para el paquete MSDU. Esto sumado a la posibilidad de simular otros estándares relacionados de la misma familia como los son 802.11b y 802.11n.

VII - REFERENCIAS.

[1]. - David Tse and Pramod Viswanath, 'Fundamentals of wireless communication', Cambridge University Press, 2005.

[2]. - Leonard H. Grokop, 'Interference Management in Wireless Networks: Physical Layer Communication Strategies, MAC Layer Interactions, and High Layer Messaging Structures', ProQuest Dissertations & Theses, University of California, Berkeley, 2008.

[3]. - Alain Sibille et al., 'Special Issue of European Wireless 2007: Advanced Physical Layer Schemes for Robust Wireless Communications', European Transactions on Telecommunications, 19:729-731, Wiley InterScience, 2008.

[4]. - Renny E. Badra and Babak Daneshrad, 'Asymmetric Physical Layer Design for High-Speed Wireless Digital Communications', IEEE Journal on Selected Areas in Communications, Vol. 17, No. 10, October 1999.

[5]. - IEEE Std. 802.3-2005, 'Media Access Control (MAC) service specification', Draft 0.1, March 2005.

[6]. - IEEE Std. 802.11, 'Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications', ANSI/IEEE, 1999.

[7]. - Dimitris Vassis et al., 'The IEEE 802.11g Standard for High Data Rate WLANs', IEEE NEtwork, May/June, 2005.

[8]. - P. Nicopolitidis et al., 'Wireless Networks', Chapter 9, Jhon Wiley & Sons, Ltd., 2003.

[9]. - IEEE Std. 802.11g, 'Further Higher-Speed Physical Layer Extension in the 2.4 GHz Band', 2003.

[10]. - Valle Islas L., 'Tesis profesional para obtener el título de Ingeniero en Electrónica y Comunicaciones', Capítulo 1: 'WLAN Red Inalámbrica de Área Local', Universidad de las Américas Puebla, 2005.

[11]. - Goldsmith Andrea, 'Wireless Communications', pp. 464-466, Stanford University, Cambridge Press, 2005.

[12]. - Vijay K. Garg, 'Wireless Communications and Networking', Chapter 6, Morgan Kaufmann Publishers, Elsevier, 2007.

[13]. - Andreas F. Molisch, 'Wireless Communications', Chapter 24, IEEE Press, 2005.

[14]. - Mischa Schwartz, 'Mobile Wireless Communications', Chapter 12, Cambridge University Press, 2005.

[15]. - Vijay K. Garg, 'Wireless Communications and Networking', Chapter 21, Morgan Kaufmann Publishers, Elsevier, 2007.

[16]. - Dimitris Vassis et al, 'The IEEE 802.11g Standard for High Data Rate WLANs', IEEE Network, May/June, 2005.

[17]. - Bo Li, Roberto Battiti and Yong Fang, 'Achieving Optimal Performance by Using the IEEE 802.11 MAC Protocol with Service Differentiation Enhancements', IEEE Transactions on Vehicular Technology, Vol. 56, No. 3, May 2007.

[18]. - Giuseppe Bianchi, 'IEEE 802.11 - Saturation Throughput Analysis', IEEE Communications Letters, Vol. 2, No. 12, December 1998.

[19]. - Giuseppe Bianchi, 'Performance Analysis of the IEEE 802.11 Distributed Coordination Function', IEEE Journal on Selected Areas in Communications, Vol. 18, No. 3, March 2000.

[20]. - Krzysztof Szczypiorski and Józef Lubacz, 'Performance analysis of IEEE 802.11 DCF networks', Journal of Zhejiang University Science A, May 2008.

[21]. - Krzysztof Szczypiorski and Józef Lubacz, 'Saturation throughput analysis of IEEE 802.11g (ERP-OFDM) networks', Springer Science+Business Media, LLC 2008.

[22]. - Hongqiang Z., Younggoo K. and Yuguang F., 'Performance analysis of IEEE 802.11 MAC protocols in wireless LANs', Wireless Communications and Mobile Computing, 4:917-931, Wiley InterScience, 2004.

[23]. - Jun He and Hung Keng Pung, 'Performance modeling and evaluation of IEEE 802.11 distributed coordination function in multihop wireless networks', Computer Communications 29 (2006) 1300-1308, ScienceDirect, Singapore 2005.

[24]. - Andrzej Duda, 'Understanding the Performance of 802.11 Networks', Grenoble Institute of Technology, LIG Laboratory, Invited Paper, IEEE 2008.

[25]. - P. Nicopolitidis et al., 'Wireless Networks', Chapter 9, Jhon Wiley & Sons, Ltd., 2003.

[26]. - IEEE Std 802.11g 'Extended Rate PHY/MAC Specifications', 2005.

[27]. - Vijay K. Garg, 'Wireless Communications and Networking', Chapter 21, pp. 741, Morgan Kaufmann Publishers, Elsevier, 2007.

[28]. - Sivanandam, S. N and Deepa, S. N, 'Introduction to Genetic Algorithms', Chapter 2, 4th edition, Springer, 2007.

[29]. - Jhon H. Holland, 'Genetic Algorithms. Computer programs that "evolve" in ways that resemble natural selection can solve complex problems even their creators do not fully understand', Scientific American, July 1992.

[30]. - Darwin C., 'On the Origin of Species by Means of Natural Selection, or the Preservation of Favored Races in the Struggle for Life', United Kingdom, 1859.

[31]. - Jhon R. Koza et al., "Genetic Programming III Darwinian Invention and Problem Solving", 3rd edition, 1999.

[32]. - Sivanandam, S. N and Deepa, S. N, 'Introduction to Genetic Algorithms', Chapter 3, 4th edition, Springer, 2007.

[33]. - Shoaib Tariq, 'MAC Algorithms in Wireless Networks: Applications, Issues and Comparisons', Master's Thesis at Umea University Department of Computing Science, Sweden, pp. 5, 2007.

[34]. - Hadi Minooei, Hassan Nojumi, 'Performance evaluation of a new backoff method for IEEE 802.11', Computer Communications 30 (2007) 3698-3704, ScienceDirect, Iran 2007.

[35]. - Raffaele Bruno, Marco Conti, Enrico Gregori, 'A simple protocol for the dynamic tuning of the backoff mechanism in IEEE 802.11 networks', Computer Networks 37 (2001) 33-444, Elsevier Science, Italy 2001.

[36]. - Maali Albalt, Qassim Nasir, 'Adaptive Backoff Algorithm for IEEE 802.11 MAC Protocol', Scientific Research, Int. J. Communications, Network and System Sciences, 4, pp. 249-323, July 2009.

[37]. - Riggi Alberto, Gomez Javier, 'RegionDCF: a Self-Adapting CSMA/Round-Robin Media Access Protocol for WLAN', 36th annual IEEE Conference on Local Computer Networks, pp. 211-214, 2011.

[38]. - Kyle Jamieson et al., 'Cross-Layer Wireless Bit Rate Adaptation', Chapter 5.1, MIT Open Access Articles, August, 2009.

[39]. - Sivanandam, S. N and Deepa, S. N, 'Introduction to Genetic Algorithms', Chapter 3, 4th edition, Springer, 2007.

[40]. - Melanie Mitchell, 'An Introduction to Genetic Algorithms', MIT Press, Chapter 5, Feb 1998.

[41]. - Dana Vrajitoru, 'Large Population or Many Generations for Genetic Algorithms? Implications in Information Retrieval', EPFL, Department of Mathematics, Switzerland, 2000.

[42]. - G. Rudolph, 'Convergence Analysis of Canonical Genetic Algorithms', IEEE Transaction on Neural Networks, vol. 5, no. 1, January 1994.

[43]. - Mohammed-Reza Akhavan, 'Study the performance Limits of IEEE 802.11 Wireless LANs', Master's Thesis at Lulea University of Technology, pp. 18-30, 2006.

[44]. - LAN MAN Standards Committee of the IEEE Computer Society, 'Information technology Telecommunications and information exchange between systems Local and metropolitan area networks Specific requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications', 1999.

[45]. - Goldberg, David Edward, 'Genetic Algorithms in Search, Optimization and Machine Learning', Chapter 2, Addison-Wesley Pub. Co., 1989.

VIII - GLOSARIO E INVENTARIO DE FIGURAS Y TABLAS.

GLOSARIO.

ACK - *Acknowledgment (control packet)*

AP - *Access Point*

BEB - *Binary Exponential Backoff*

BER - *Bit Error Rate*

CCK - *Complementary Code Keying*

CSMA/CA - *Carrier-sense multiple access / Collision Avoidance*

CTS - *Clear To Send (control packet)*

CW - *Contention Window*

DCF - *Distributed Coordination Function*

DIFS - *Distributed Inter-Frame Space*

DSSS - *Direct-Sequence Spread Spectrum*

FTP - *File Transfer Protocol*

IEEE - *Institute of Electrical and Electronics Engineers*

MAC - *Medium Access Control*

MSDU - *MAC Service Data Unit*

NAV - *Network Allocation Vector*

OFDM - *Orthogonal Frequency Division Multiplexing*

PBCC - *Packet Binary Convolutional Coding*

PHY - *Phyisical Layer*

PLCP - *Physical Layer Convergence Protocol Preamble*

RTS - *Request To Send (control packet)*

SIFS - *Short Inter-Frame Space*

TCP - *Transmission Control Protocol*

WLAN - *Wireless Local Area Network*

OG - *Operación Genética*

Crossover - *Recombinación*

Mutation - *Mutación*

Reproduction – *Reproducción*

INVENTARIO DE FIGURAS.

INVENTARIO DE TABLAS.

IX - ANEXO I.

En las siguientes páginas se incluye el borrador preparado en forma de Artículo de Investigación de alto nivel para ser considerado a inscripción en un futuro.

GENETIC ALGORITHM TO IMPROVE THE BINARY EXPONENTIAL BACKOFF CONFIGURATION IN A SIMULATED IEEE 802.11g WIRELESS NETWORK

Gómez, J. PhD., Krishnamachari, B. PhD., Marin, Carlos.

Abstract.

In the current work we developed a new strategy to find a better binary exponential backoff configuration for an IEEE 802.11g network. We used genetic programming to implement a way to try-and-error search values for CWmin and CWmax to improve a simulated wireless network with N stations always willing to transmit a packet. This is deployed within a Java application which collects INPUT DATA from the user regarding initial conditions of the wireless network and the genetic algorithm conditions and will throw the OUTPUT DATA which consists of a new BEB configuration, optimized to reach a network performance improvement.

- Introduction -

For the approach of smart wireless network protocols there has been made a lot of research [1-4] with evolutionary, dynamic and adaptive algorithms which would implement a different way of reaching the best network performance. However, in order to enhance the average throughput of an IEEE 802.11 wireless network; we have to implement an intelligent algorithm that will lead us to an optimization solution. Genetic algorithms have proven to be a good approach to find the very best solution to a problem where the search space is way too large, or whenever we do not exactly know the best way to solve it [5].

For an IEEE 802.11 wireless network, some parameters are critical in the way we deal with the access control for the channel or medium (MAC). At this point, the protocol for the network needs to be efficiently enough to manage all the stations willing to transmit a packet and avoid collisions (where the transmitted packet is lost).

IEEE 802.11 standard implements a procedure for which the stations wait a uniform randomized amount of time before they attempt to transmit their packets, this is a way to lower the probabilities for a collision to happen. This procedure is called Binary Exponential Backoff.

The binary exponential backoff (BEB) procedure consists in randomly select a number (n) between 0 and CWmin - 1 and then make the stations to wait for an 'n' SLOTTIME interval, this is known as the 'contention window'. Thus, when the willing-to-transmit station senses the channel idle for a DIFS period (interframe spaces are better described at the IEEE 802.11 standard [6]) they begin a countdown of 'n' times SLITTIME (where SLOTTIME duration varies within the protocol, 802.11b/g/n) as long as the channel stills idle. And when this contention window reaches zero, they will now transmit their packet (which can be DATA itself or a control packet RTS). With this randomized amount of wait time of the BEB, the stations have lower probabilities to transmit at the same time and therefore to collide their packets.

Plus, the BEB procedure implements a way to 'punish' problematic nodes (stations that collide consecutively) this procedure is a binary climbing of the contention window. Each time a station collides, the contention window will be duplicated, thus, if a transmitted packet is lost and retransmission is requested, the stations will randomly select a new value between 0 and 2*CWmin – 1 for its contention window. If more consecutively collisions occur, the contention windows will continue to duplicate its range until a CWmax. The number of binary steps the CWmin takes to reach the CWmax value is known as 'stages' (m).

So, as we know, the network performance is greatly dependent of the channel well utilization. The average throughput will decrease when the channel is not being used for longer amount of time. And therefore, the average network throughput will increase if fewer collisions happen and if we lower the amount of wait time for the stations to successfully transmit.

- Justification -

As we explained before and as some research has proven [7], the average throughput for IEEE 802.11 networks will decrease significantly when the conditions of the network changes (high number of stations or with greater congestion for data transmissions). The IEEE 802.11

based standards don't have a way to deal with these changes and the values for the BEB are static (depending of which protocol we use, 802.11b/g/n).

So, the proposal of this paper is a smart way to implement genetic algorithms to find the best configuration for the BEB procedure, depending on the characteristics of the network. This will be implemented through a Java application. The Java application called 'GBEBapp' will run a genetic algorithm which will try-and-error search for different values of CWmin (W) and stages for CWmax (mStages) in order to get a better average throughput. The application itself has a function to simulate an 'N' stations always willing to transmit wireless IEEE 802.11g network with a noise-free channel. The application will collect INPUT DATA from the user regarding to the initial values for the network (Wo, mStages and N stations) and the genetic algorithm conditions (population size M and termination criteria).

The GBEBapp will then begin the search for the best BEB configuration (W and mStages values) by simulating the N stations network and 'mutating' the results until the termination criteria is accomplished. The OUTPUT DATA consists of these values for the BEB and the average throughput reached improvement.

It is important to notice that this GBEBapp will not modify the way the BEB procedure works. Instead, the GBEBapp is a tool for network administrators who would like to modify their wireless network MAC parameters in order to get a better performance according to each individual network characteristics. The program will deliver a more efficient BEB values for it.

- Development -

The GBEBapp implements a genetic algorithm to find the fittest individual of N generations. The individual is the BEB configuration (W and mStages values). In order to 'improve the species', the program would perform genetic operations (analogous to the Darwinian Natural Selection process). These operations: Crossover, Reproduction, Mutation and CrossoverMutation, would change each generation of individuals, conserving the fittest in every generation (elitism) and for the last generation, the very fittest solution is going to be displayed as the OUTPUT DATA.

The genetic operations, as well as each individual of the current generation; would have a probability of being chosen. This probability is fixed for the genetic operations (obeying to the Natural Selection process) and is variable for the individuals.

Each individual would have a probability to being selected for a genetic operation, according to its fitness. So, an individual (W and mStages values) which reaches a high average throughput (in the built-in simulation method of N stations wireless network) has a higher probability of being chosen. By doing this, we are ensuring the survival of the fittest.

All of the genetic program process would run behind a friendly GUI (Figure 1). The user only needs to provide the INPUT DATA and click the SOLVE button.
After several RUNS of the GBEBpp and under tests for multiple scenarios (different number of stations and different termination criteria parameters) we can assert that the genetic program works as planned and efficiently deploys an average network throughput improvement.

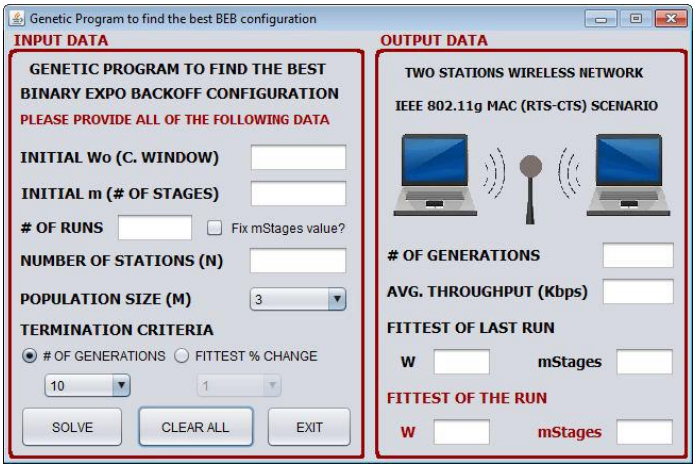


Figure 1 – GBEBapp GUI.

- Results -

The GBEBapp reached an improvement to the average throughput for an IEEE 802.11g network in the two different modalities, with an mStages fixed value or without it. As we can see at the Figure 2, the improvement occurred in all the cases for a sweep of 10 to 60 stations simulations.

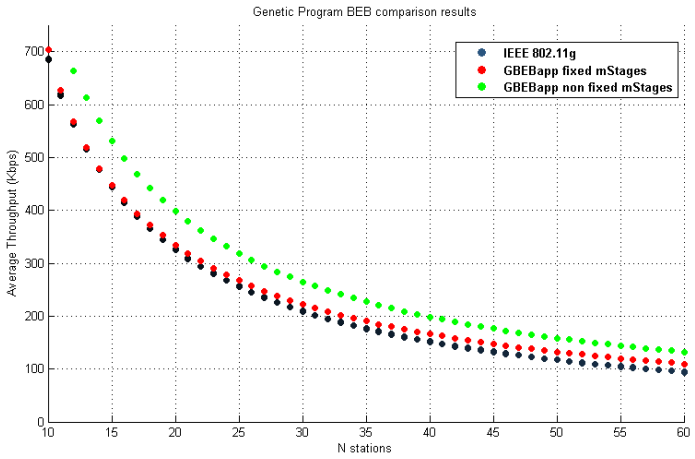


Figure 2 – Results for the GBEBapp.

- Conclusions -

We have proven that there is, in fact, a way to implement genetic algorithms to the search for a more efficient configuration to the IEEE 802.11 standard.

For nowadays wireless networks evolution, a static implementation for the medium access control mechanism is not the best idea. The GBEBapp proved that values for the CWmin and CWmax of the BEB process should change whenever the network conditions change as well. Doing so, we could improve the network performance between 10 and 30% above the average throughput achieved with the stand-alone IEEE 802.11 standard.

- Acknowledgements -

I want to thank people who made this research project possible. Dr. Dorador and Dr. Valero who were responsible for the UNAM-USC Summer Research Internship. Dr. Javier Gómez and Dr. Krishnamachari who took me under their advice to develop this topic.

- References –

[1] Maali Albalt and Qassim Nasir, 'Adaptive Backoff Algorithm for IEEE 802.11 MAC Protocol', Int. J. Communications, 2009.
[2] Raffaele Bruno, Marco Conti, Enrico Gregori, 'A simple protocol for the dynamic tuning of the backoff mechanism in IEEE 802.11 networks', Computer Networks Journal, No. 37, 2001.
[3] Hadi Minooei, Hassan Nojumi, 'Performance evaluation of a new backoff method for IEEE 802.11', Science Direct, 2007.
[4] Tae-Man Han et al., 'A Hybrid MAC Protocol Based on Implicit Token and CSMA/CA for Wireless Network', Korea, 2007.
[5] Jhon R. Koza et all, "Genetic Programming III Darwinian Invention and Problem Solving", 3rd edition, 1999.
[6] LAN MAN Standards Committee of the IEEE Computer Society, "Information technology Telecommunications and information exchange between systems Local and metropolitan area networks Specific requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", 1999.
[7] Mohammed-Reza Akhavan, "Study the performance Limits of IEEE 802.11 Wireless LANs", 2006.