METAHEURÍSTICAS
CRITICAL NODE PROBLEM

Escuela
Politécnica
Superior

Contenido

Descripción del problema

Dado un gráfico no dirigido G (V, E) y un entero K, el problema del nodo crítico (CNP) consiste en la búsqueda de un subconjunto de K nodos tal que $S \subseteq V$, de modo que el número de pares de nodos aún se encuentran conectados en el subgrafo G [V \ S] es el más pequeño posible.

$$f(S) = |\{i, j \in V \setminus S : i \text{ y } j \text{ están conectados por un camino en } G[V \setminus S]\}|$$

Por último, debemos saber que el CNP es un problema NP-completo, cuya solución es polinomial mediante árboles y grafos estructurados.

Definiremos la calidad de una solución candidata como la métrica betweeness centrality (bc) de un nodo concreto *i*, obtenida mediante la ecuación:

$$bc(i) = \sum_{\substack{j,k \\ i \neq j \neq k}} \frac{b_{jik}}{b_{jk}}$$

Siendo b_{jk} el número de caminos más cortos entre los nodos *j* y *k*, y b_{jik} el número de caminos más cortos entre los nodos *j* y *k* que pasan por el nodo *i*.

Debemos destacar que en caso de que exista un camino más corto entre los dos nodos *j* y *k*, estaríamos midiendo si el nodo *i* se encuentra en dicho camino más corto.

Descripción de la codificación de soluciones utilizada, junto a la función de evaluación de éstas.

Clase CNPSolution

Esta clase es la encargada de almacenar las soluciones estocásticas al problema dado, es un vector de enteros que almacena los N nodos del grafo que se van a evaluar y el fitness asociado a dicha solución.

El proceso de codificación de esta clase ha sido mediante la representación de nuestra solución en un vector de tamaño *m*, siendo *m* el número de nodos a borrar y en el que cada posición del vector almacenará un número entero, que representará el nodo a borrar del grafo.

Se han incluido otras variables que nos permitirán operar con las distintas metaheurísticas tales como:

- **Fitness de la solución**: Será la suma de los Betweenness centrality individuales del grafo, habiendo eliminado los respectivos nodos.
- **FitnessAsigned**: Variable booleana que nos permitirá determinar si una solución ha sido previamente evaluada, lo cual nos permitirá ser de ayuda a la hora de codificar nuestro algoritmo genético obviando así volver a calcular soluciones que ya habían sido calculadas.
    - Esta variable será modificada una vez se llame a la función computeFitness.
- **NodesSolSize**: Será un número entero que almacenará el número de nodos que deberá tener el vector solución, lo que nos permitirá llevar un control respecto al tamaño del vector solución.

Finalmente, se mostrará una solución cualquiera, que podrá tener la siguiente estructura:

- **Vector solución**:

0	**3**	**2**	**7**

- **Fitness**: 27
- **FitnessAsigned**: True
- **NodesSolSize**: 4

Funciones necesarias para el correcto funcionamiento

Función addNodeToSol(int node): Permite introducir al vector solución el nodo pasado como argumento, no será necesaria la comprobación de buscar el nodo en el vector, por lo que es recomendable comprobar su existencia previamente.

Función getNodeFromSol(int index): Permite dado el índice pasado como argumento, devolver el nodo del grafo que se encuentre en el vector solución.

Función checkDuplicity(int node): Función clave en el desarrollo de la práctica, debido que nos será de utilidad para comprobar si el nodo pasado como argumento existe en nuestro vector solución, para agilizar su búsqueda se ha optado por utilizar métodos ya optimizados de la STL.

Función changeNode(int index, int node): Permitirá intercambiar el nodo que se encuentra en la posición "index" del vector solución por un nuevo nodo que será pasado como argumento (node).

Función deleteNode(int node): Permite eliminar el nodo pasado como argumento del vector solución, haciendo uso de las funciones optimizadas de la STL para la búsqueda de elementos en un vector.

Clase CNPEvaluator

Esta clase es la encargada de realizar la evaluación de una solución obtenida mediante las respectivas metaheurísticas empleadas.

Principalmente, se han planteado dos funciones que serán las encargadas de realizar todo el procedimiento de computo.

computeFitness(CNPInstance &instance, CNPSolution &solution): Hemos planteado su funcionamiento tal que:

1. Se realiza una copia del objeto instance para no modificar el original.
2. Se eliminan las aristas de los nodos que se encuentran en el vector solución.
3. Una vez borradas, se procede a realizar el cálculo del betweenness centrality, que será la suma de los BCs individuales de cada nodo.
4. Realizamos una copia del vector solución a la instancia original, que será de ayuda para el Iterated greedy.
5. Retornamos el fitness obtenido.

computeDeltaFitness(CNPInstance &instance, CNPSolution &solution, int index, int node): Su planteamiento ha sido el siguiente:

1. Guardamos el fitness Actual
2. Intercambiamos los nodos para obtener su nuevo fitness.
3. Una vez obtenidos se realiza la diferencia entre fitness y se devuelve, dejando los nodos igual que estaban.

Resumidamente la idea para una solución será aplicar a un grafo de copia las operaciones de destrucción de nodos dados por la solución y calcular la métrica de Betweeness centrality del grafo parcialmente destruido.

Una vez calculado la métrica se revuelve el valor de su BC y este es el fitness de la solución dada.

Metaheurísticas consideradas

Hemos considerado seis Metaheurísticas puras para abordar nuestro problema, así como se ha realizado una modificación en una y una hibridación entre dos de ellas, por lo que en total serían ocho metaheurísticas.

En cada una de ellas se ha debido de tener en cuenta que para el tratamiento y generación de soluciones no se incluyan nodos duplicados a tratar por una evaluación, dicho de otro modo, las soluciones con los N nodos a calcular jamás tendrán una duplicidad dentro de la solución, pues es una restricción del problema dado.

Todas las Metaheurísticas consideradas han sido realizadas considerando que el problema es de minimización, debido que estamos tratando de obtener el mayor grado de debilidad de un grafo.

Escalada Simple y máxima pendiente

En el proceso de adaptación de estas metaheurísticas se ha planteado lo siguiente:

- Estas metaheurísticas consisten sobre todo en buscar soluciones vecinas. Anteriormente, en el problema de la mochila, ocurría que el índice de la mochila se podía repetir en el vector solución. En cambio, en este problema, los valores de cada índice del vector solución representan un nodo del grafo, por lo que no se deben repetir en el vector solución cuando cambiamos un nodo por otro buscando una solución vecina. Por tanto, se ha tenido que controlar este factor cada vez que pasamos a evaluar una solución vecina.
- Los nodos que intercambiaremos con la solución para obtener la solución vecina vienen de un vector, el cual contiene una permutación del número de nodos total del grafo.
- Como se trata de un problema de minimización, tenemos que evaluar las soluciones en función de si el DeltaFitness que producen es menor que 0.
- Cada vez que se acepta una solución vecina, hay que controlar que se haga el cambio de nodo en la operación de asignación.

Enfriamiento Simulado

Para la adaptación de este problema al algoritmo se han tenido en cuenta varias cuestiones:

1. Para una instancia de M nodos y N nodos a evaluar, dependiendo de si el tamaño es muy grande o pequeño, los valores de la inicialización del algoritmo funcionan mejor en torno a unos valores u a otros, por ejemplo, en un problema de pocos nodos a evaluar, el factor de enfriamiento debe de ser un valor lo suficientemente capaz de lograr que el enfriamiento se haga lentamente para así poder pues evitar una convergencia rápida de la Metaheurística y por ello una solución no tan buena.
2. El factor de enfriamiento utilizado no es ninguno en los vistos en el temario, básicamente es un valor entre [0,1] que al multiplicar por la temperatura esta decrece en menor o mayor grado, mientras más cercano a 1 este ese valor más lentamente se enfriara la temperatura.
3. En la generación de las soluciones se tuvo en cuenta la duplicidad de los nodos en la solución a evaluar.

Búsqueda Tabú

Para la adaptación de este problema al algoritmo se han tenido en cuenta las cuestiones:

1. Para una instancia de M nodos y N nodos a evaluar, dependiendo de si el tamaño es muy grande o pequeño, los valores de la inicialización del algoritmo funcionan mejor en torno a unos valores u a otros, por ejemplo, en un problema de pocos nodos a evaluar, el factor de tiempo en la búsqueda Tabú era más corto que en los problemas de más nodos, se decidió usar la métrica de numero Nodos/2.5.
2. En la generación de las soluciones se tuvo en cuenta la duplicidad de los nodos en la solución a evaluar.
3. En la comprobación de los nodos en la búsqueda, se comprobó también que no fuera un nodo duplicado.

Greedy Iterativo

Los aspectos más relevantes a la hora de codificar esta metaheurística respecto a la vista en las prácticas han sido algo extensos, sin embargo, los puntos más importantes son:

1. La función de inicialización se ha visto modificada, recibiendo un nuevo parámetro booleano, que nos indicará si estaremos haciendo uso de un iterativo greedy puro o bien de la versión modificada.
2. La función de ejecución de la heurística se ha visto simplificada ya que inicializamos la mejor solución de manera greedy y no aleatoria como en la mochila, esto nos asegurará una convergencia mucho más temprana.
3. La función de destrucción únicamente se ha visto modificada, de tal forma que recorrerá todos los nodos del grafo eliminando aquellos que se encuentren en el vector solución con una determinada probabilidad Alpha.
4. Se ha eliminado la función rebuild y se ha creado una función de construcción de soluciones iniciales, así como de reconstrucción de una solución parcial.
    a. Esta función se basa en iterar mientras que el vector de la solución dada no alcance el número de nodos máximo permitido.
    b. Se ha hecho uso de una lista de candidatos, representada mediante un vector, el cual almacenará los nodos que han intervenido en el cálculo del BC (no se consideran los nodos eliminados).
    c. Esta lista de candidatos nos será útil debido que obtendremos el mejor elemento de ella, es decir, el que al eliminarlo nos proporcione mayor debilidad para el grafo.
    d. Debido que al obtener siempre el nodo que mayor debilidad aporta al grafo, las soluciones con un iterativo greedy convergen al cabo de pocas iteraciones, de tal forma que se ha planteado introducir un factor de aleatoriedad, haciendo una selección del nodo entre los 5 mejores, dependiendo del tamaño del grafo.

Greedy Iterativo modificado

Al igual que en el greedy Iterativo comentado previamente, su funcionamiento es exactamente el mismo, únicamente variarán los siguientes aspectos.

1. La función initialise recibirá true, indicando que haremos uso de este heurística.
2. La probabilidad de destrucción inicializada deberá ser alta.
3. Su funcionamiento estará basado en destruir inicialmente una gran cantidad de nodos de las soluciones, de tal forma que podremos generar mayor variedad de soluciones abarcando más vecindarios y no estancarnos en posibles óptimos locales.
4. La probabilidad de destrucción irá siendo reducida al paso de las iteraciones, manteniéndose constante al alcanzar 0.25, de esta forma nos aseguramos de que en cada nueva iteración destruya al menos un nodo solución y haciendo que las soluciones converjan en dicha vecindad.

Algoritmo Genético

El funcionamiento de esta metaheurística es similar al planteado en la práctica, donde las principales modificaciones han sido:

1. Añadir una variable booleana indicando si estamos haciendo uso de un genético usando soluciones aleatorias, o bien rellenando la población 90% aleatorias y 10% greedys.
2. La inicialización de la población, como he indicado previamente, dependiendo de la variable booleana, haremos uso de un método u otro.
3. Se ha establecido una población de 60 individuos, por lo que el torneo será realizado de un 10% de la población, tratando de introducir presión selectiva en nuestro algoritmo.
4. El operador de cruce ha variado significativamente debido a las siguientes razones.
    a. Al igual que en la mochila, iteramos sobre los distintos genes del individuo asignando el del padre1 o el padre2, sin embargo, podemos incurrir en introducir nodos del grafo repetidos, por lo que entraríamos a un bucle infinito al estar comprobando la duplicidad y no poder rellenar el vector solución nuevo.
    b. La alternativa planteada ha sido, comprobar el número de inserciones, así como el número de iteraciones, de tal forma que al llegar a un determinado número impuesto y no haber finalizado, el hijo devuelto será la copia del padre con mejor fitness.
    c. En caso de que según la probabilidad dichos padres no se crucen, el hijo será una copia del mejor padre.
5. Respecto al operador de mutación, será similar al propuesto en el problema de la mochila, variando únicamente que el nuevo gen mutado, en nuestro caso un nodo del grafo, no se encuentre ya insertado en el vector solución.

Algoritmo Genético con Iterativo greedy

Al igual que se ha explicado previamente, este método hará uso de un 10% de soluciones generadas de manera greedy, completando con un 90% con soluciones aleatorias.

Esto se ha realizado debido que se buscaba introducir una variación en el algoritmo genético, tratando de buscar obtener mejores resultados al introducir soluciones obtenidas mediante un algoritmo voraz.

Funcionamiento de mi MH

Enfriamiento Simulado

Esta metaheurística funciono bien en el problema del CNP en base a que es una MH que permite aceptar en base a los criterios que hemos visto óptimos peores soluciones para salir de las zonas de óptimos locales, el algoritmo ha demostrado ser eficiente para instancias de nodos elevados como se mostrara en el apartado de las gráficas, generando muchísimas iteraciones en el tiempo estipulado por el algoritmo.

Se llega a la conclusión de que el resultado de la MH fue bastante bueno, ya que en comparación con otros métodos usados la convergencia llega en pocas iteraciones en algunos de los casos expuestos en las gráficas.

Respecto a la calidad de la solución dada vemos que el enfriamiento por ejemplos en la instancia de 50_30 Ilustración 5 llega a una convergencia con apenas 88 puntos de fitness mientras las variantes del greedy están por encima de 120 puntos, el algoritmo genético se muestra predominante en esta comparativa en esa instancia.

Finalmente haciendo referencia al estudio de la densidad de aristas del grafo, obviamente a mayor cantidad de conexiones, mayor será la complejidad computacional, así como el fitness obtenido, sin embargo, se puede observar en las distintas gráficas como la convergencia de los resultados es similar.

Búsqueda Tabú

De esta MH no podemos decir que funcione demasiado bien para el problema del CNP, el encadenamiento de bucles dentro del algoritmo hace que pueda ser de tiempo $O(n^3)$ requiriendo de esta manera muchísimo tiempo para calcular cada una de las iteraciones de la instancia proporcionada, se observa que con las instancias de 50 nodos el algoritmo realiza en un tiempo bastante grande un promedio de 100 iteraciones, aquellas que se ven en este y este punto.

Para las instancias de 75 decaen a un promedio de 50 iteraciones y para las de instancias de 100 se hizo inviable sacar una conclusión por el gran tiempo de cómputo y de escasos resultados dados, en una de las pruebas se constató que se utilizaron 47 minutos para el cálculo del algoritmo sobre la instancia de 100 nodos y densidades de 30% y 60% con una cantidad de nodos a eliminar de 30, los resultados arrojados fueron dados en apenas 5 y 6 iteraciones, algo escaso para poder exponer un ejemplo gráfico y practico con un gran número de elementos en la instancia.

Escalada simple y máxima pendiente

Estas MH funcionan bien siempre que el número de nodos del grafo esté alrededor de 50 (Ilustración 1) (Ilustración 2). Podemos observar que la escalada simple obtiene algo de mejor rendimiento, consiguiendo una solución ligeramente mejor a la máxima pendiente, la cual tiene una rápida convergencia.

Pero una vez superado el umbral de 50 nodos, comenzamos a tener problemas y lo podemos apreciar con 75 nodos (Ilustración 3)(Ilustración 4), ya que tarda demasiado en realizar una iteración. Podríamos obtener mejores resultados si aumentásemos el parámetro MAX_SECONDS_PER_RUN, pero ocurre que el tiempo de cómputo se dispararía aún más.

El motivo de que esto ocurra es similar a el problema que tiene la búsqueda tabú. Resulta que hay que recorrer todos los nodos totales del grafo junto con todos los nodos de la solución. Además, tenemos que añadir el factor de la duplicidad de los nodos, para el cual se utiliza otro método que emplea un bucle.

Al final el tiempo de computo viene a ser de $O(n^3)$ y por este motivo, no es viable para un número de nodos elevado, y al tiempo del algoritmo tenemos que añadir el tiempo del cálculo del BC. De hecho, no ha sido posible incluir las gráficas con la instancia de 100 nodos, debido al exagerado tiempo de computo que estas toman.

Por otro lado, hemos creado instancias con diferente densidad de aristas. En el caso de esta metaheurística, apenas existe diferencia entre una densidad del 30% y una densidad del 60%, lo cual se puede apreciar en las gráficas(Ilustración 1)(Ilustración 2)(Ilustración 3)(Ilustración 4).

Sin embargo, aunque no influye en los resultados, sí que lo hace en el tiempo de computo.

Iterativo Greedy

La aplicación de ésta metaheurística ha resultado bastante satisfactoria, debido que realizando comparaciones respecto a otras heurísticas como son el enfriamiento simulado o la búsqueda tabú, el iterativo greedy consigue una convergencia absoluta tal y como podemos observar en la Ilustración 20, donde consigue dicha convergencia en apenas 100 iteraciones al contrario del enfriamiento simulado, visible en la Ilustración 10 o incluso en la búsqueda Tabú Ilustración 14.

Donde la convergencia para obtener la solución final se da al cabo de las 4000 iteraciones.

Se ha de destacar que el funcionamiento de esta heurística es bueno debido que estaremos haciendo siempre uso de los nodos del grafo que mayor debilidad nos aportan, por lo que al estar tratando siempre de debilitar el grafo lo máximo posible converge muy rápido.

Respecto al fitness obtenido en estos métodos, tanto el iterativo Greedy puro como la modificación presentada, podemos observar como mediante la modificación conseguimos una mejora mínima del fitness en algunas instancias, por lo que creemos no merece realmente la pena para este problema la aplicación de dicha modificación.

Así como también haciendo referencia a los resultados y pudiendo verse en gráficas como Ilustración 5 o Ilustración 11 el fitness obtenido es mucho mejor que el obtenido para nuestro algoritmo greedy.

Lo cual puede deberse en parte que nunca estamos iterando absolutamente todos los nodos del grafo, tratando de eliminar de la solución aquel que intercambiándolo por el nuevo nodo nos mejore el fitness, sino más bien destruimos la solución y volvemos a reconstruirla utilizando el nuevo nodo proporcionado por el cálculo de los BCs.

Finalmente haciendo referencia al estudio de la densidad de aristas del grafo, obviamente a mayor cantidad de conexiones, mayor será la complejidad computacional, así como el fitness obtenido, sin embargo, se puede observar en las distintas gráficas como la convergencia de los resultados es similar a las instancias con densidad del 30%.

Algoritmo Genético

La implementación del algoritmo genético tal y como se ha explicado previamente, creemos que ha sido bastante buena y se ha adaptado bien a nuestro problema.

El principal motivo ha sido que obtiene resultados mejores respecto a las demás heurísticas empleadas, tal y como podemos ver en las Ilustración 44 o Ilustración 50, donde al igual que en el resto de heurísticas la complejidad añadida de la densidad ha provocado que el tiempo requerido para obtener una solución sea bastante mayor, por lo que para instancias demasiado grandes provocará que el tiempo se dispare.

Referenciando a la Ilustración 28 e Ilustración 40, podemos observar las principales diferencias entre el algoritmo genético puro frente al generado con un 10% de soluciones voraces.

En el algoritmo puro, podemos observar como todos los individuos tienen un conjunto de fitness muy similar, por lo que dichos individuos van casi paralelamente a la mejor solución obtenida, sin embargo, cuando tenemos un 10% de individuos greedys, que pueden ser mejores que los aleatorios y realizamos sus cruces, así como mutaciones tenemos una mayor variedad de individuos, que como se aprecia abarcan una mayor región del espacio de búsqueda.

Respecto a la Ilustración 44 o Ilustración 50, podemos observar como a mayor cantidad de nodos (independientemente de la densidad), las soluciones convergen de una forma muchísimo más rápida que en menor número de nodos alcanzando soluciones mejores que las obtenidas en el resto de heurísticas.

El principal motivo por el que convergen a un número de iteraciones cercano a 1000 frente al genético normal, puede ser debido que estamos introduciendo siempre un porcentaje de población generado vorazmente, así como sustituyendo el peor hijo por el mejor padre.

Esto junto la configuración establecida en la selección de padres y probabilidades, conseguimos que la presión selectiva sea mayor, alcanzando una convergencia más temprana.

Comparativa Experimental

A la hora de aplicar nuestras metaheurísticas, hemos generado 8 instancias del problema tal que podemos diferenciarlas en:

- Instancia de 50 nodos con 30% densidad de aristas y 20 nodos a borrar.
- Instancia de 50 nodos con 60% densidad de aristas y 20 nodos a borrar.
- Instancia de 75 nodos con 30% densidad de aristas y 25 nodos a borrar.
- Instancia de 75 nodos con 60% densidad de aristas y 25 nodos a borrar.
- Instancia de 100 nodos con 30% densidad de aristas y 30 nodos a borrar.
- Instancia de 100 nodos con 60% densidad de aristas y 30 nodos a borrar.

Como nota, se ha de destacar que en algunas MH no ha sido posible aplicar la instancia de 100 nodos debido a la alta capacidad de cómputo y tiempo necesaria.

Este punto ya ha sido explicado previamente en las distintas MH empleadas, por lo que no se entrará en más detalles.

Como punto final, hay que destacar que tanto la búsqueda Tabú, enfriamiento simulado y genéticos han obtenido soluciones muy similares, sin embargo, únicamente el genético y el greedy iterativo han sido capaces de converger en todas las instancias abordadas en nuestro problema, tanto para 30% de densidad como 60%.

Escalada Simple y Máxima Pendiente

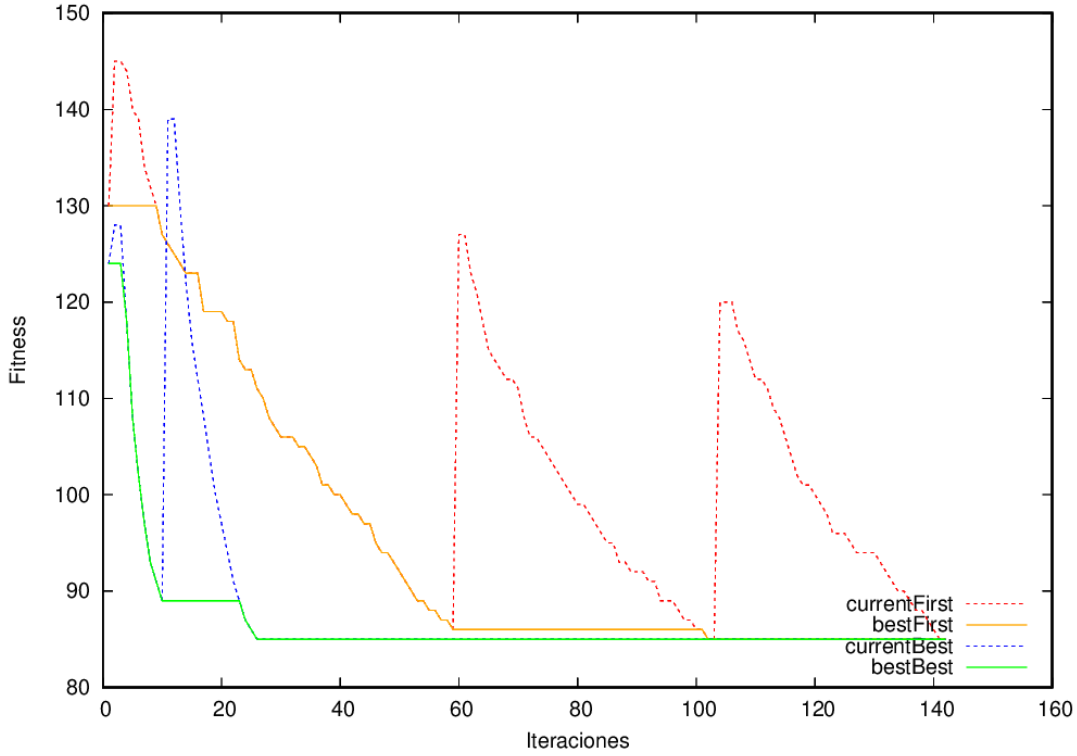

Ilustración 1 Instancia 50 nodos y 30% densidad con 20 nodos de eliminación LS

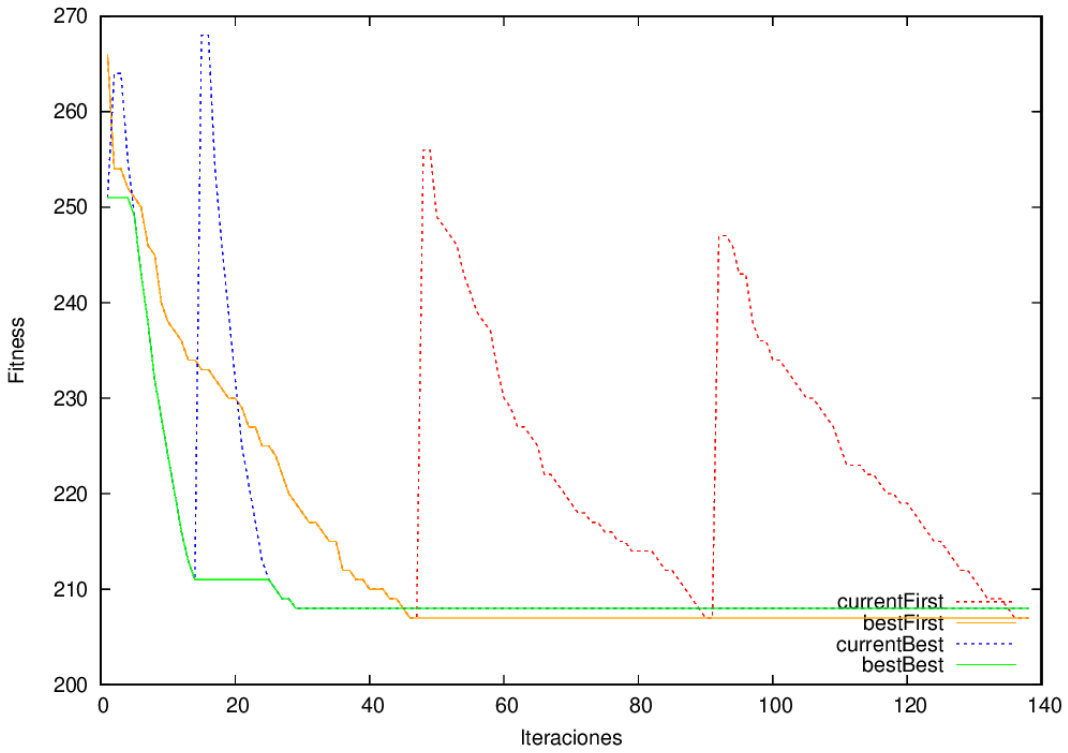

Ilustración 2 Instancia 50 nodos y 60% densidad con 20 nodos de eliminación LS

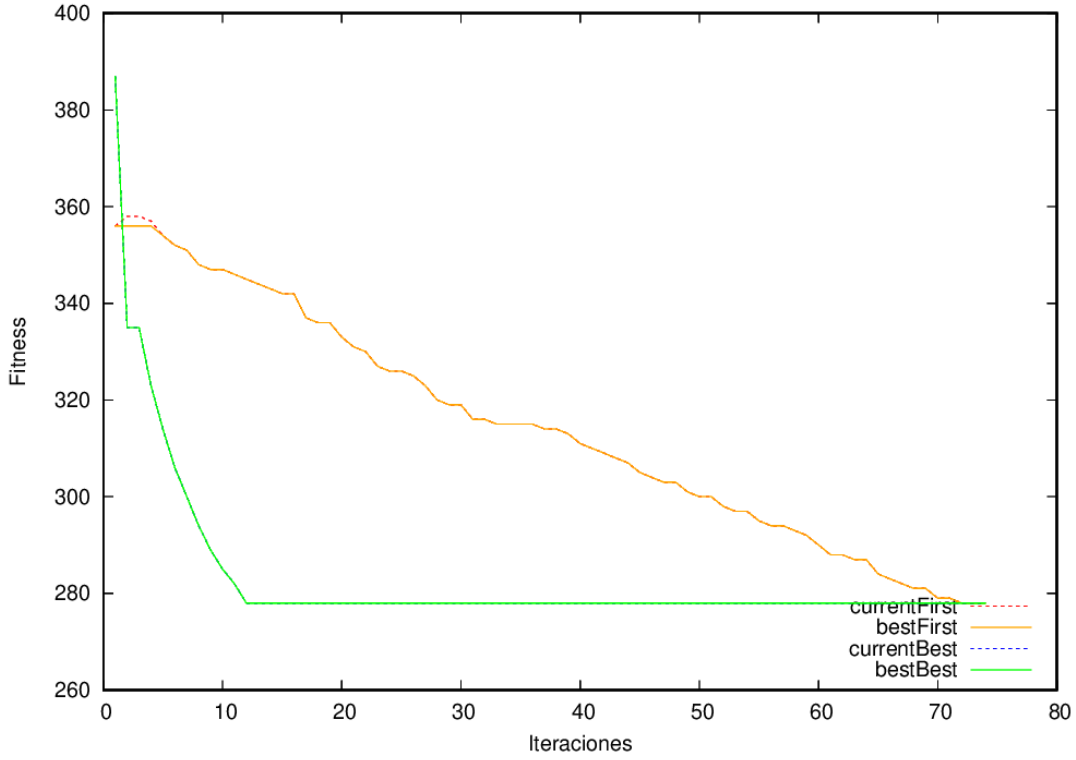

Ilustración 3 Instancia 75 nodos y 30% densidad con 25 nodos de eliminación LS

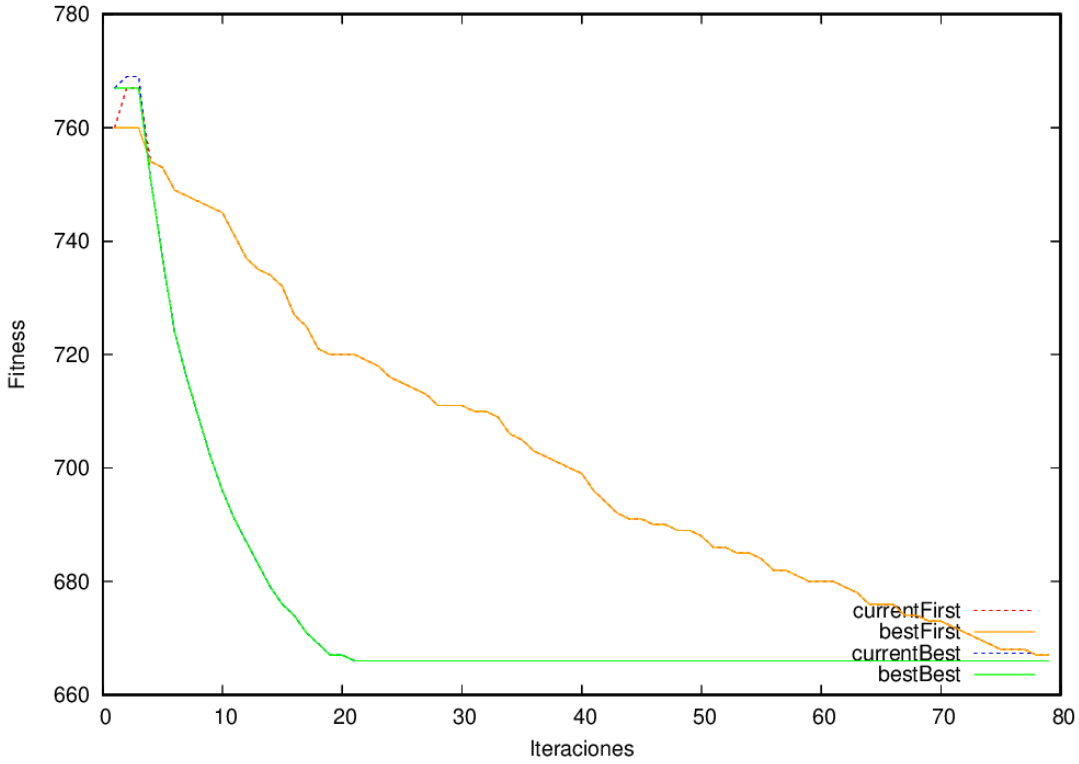

Ilustración 4 Instancia 75 nodos y 60% de densidad con 25 nodos de eliminación LS

Enfriamiento Simulado

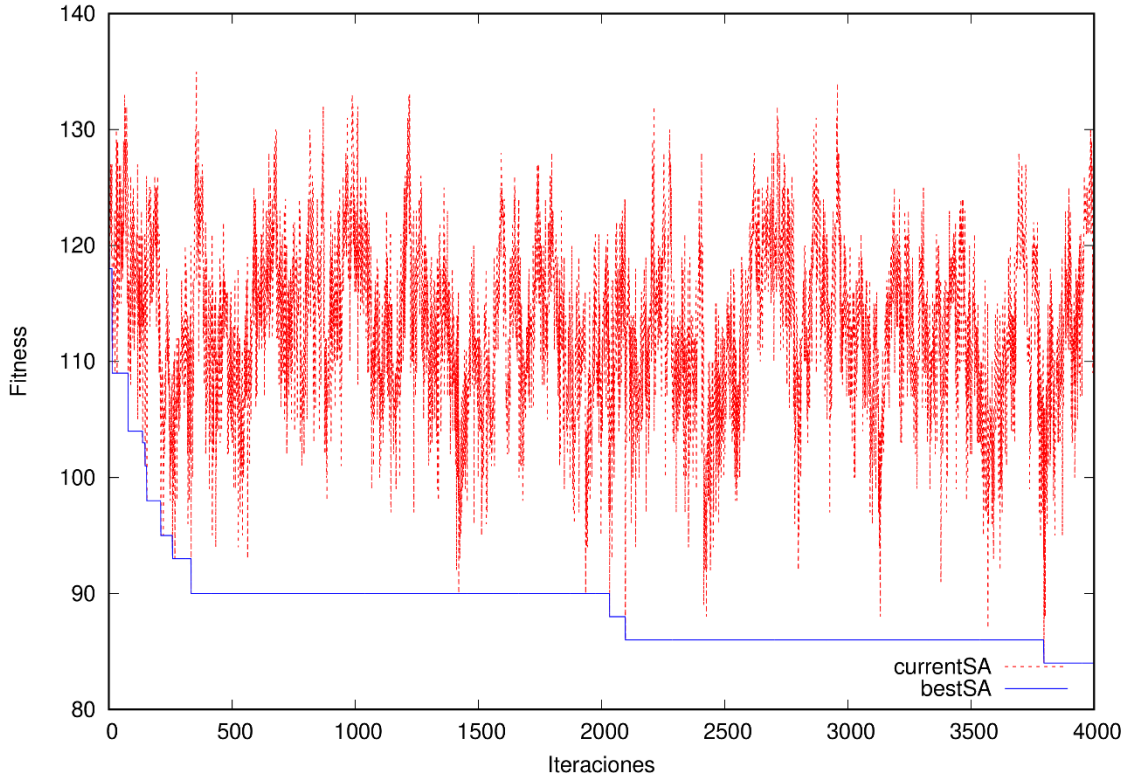

Ilustración 5 Instancia 50 nodos y 30% densidad con 20 nodos de eliminación SA

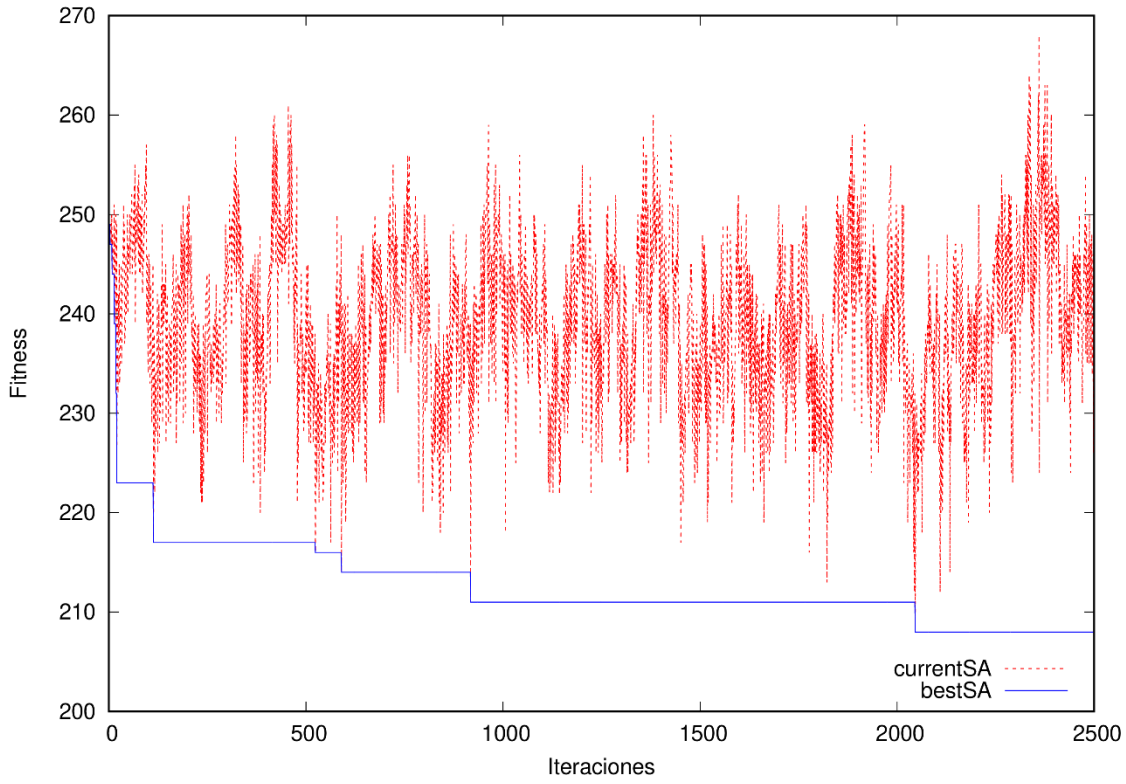

Ilustración 6 Instancia 50 nodos y 60% densidad con 20 nodos de eliminación SA

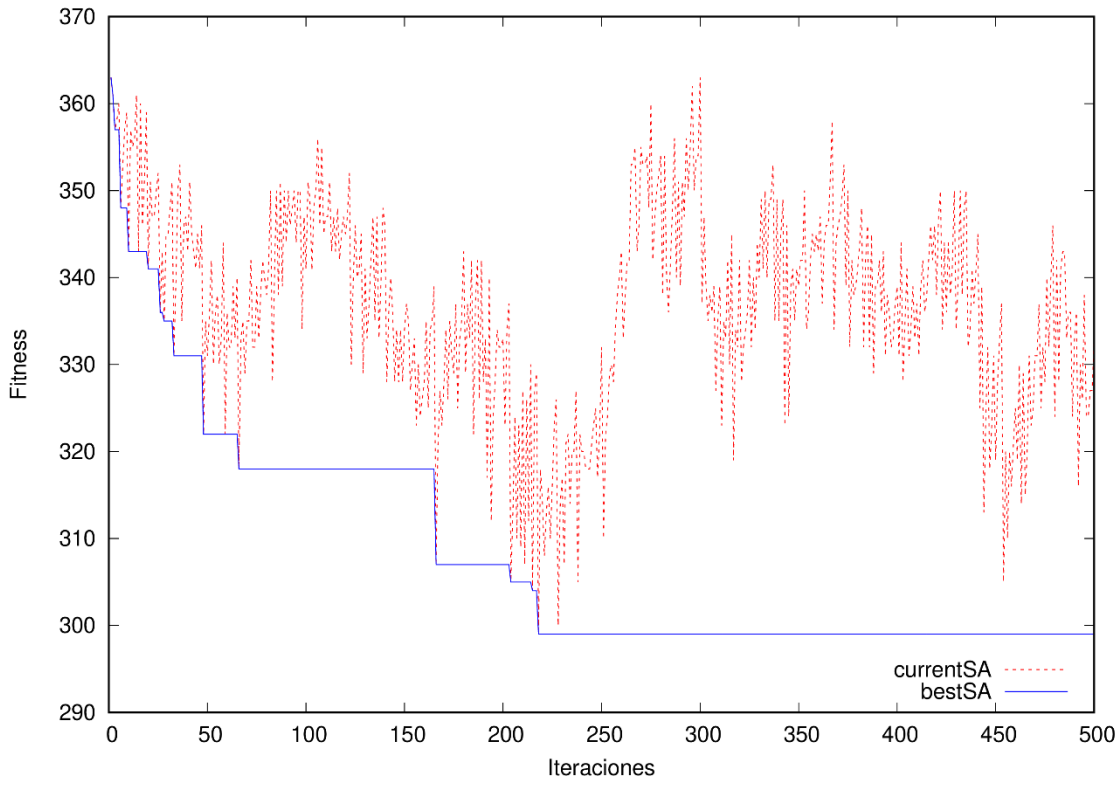

Ilustración 7 Instancia 75 nodos y 30% densidad con 25 nodos de eliminación SA

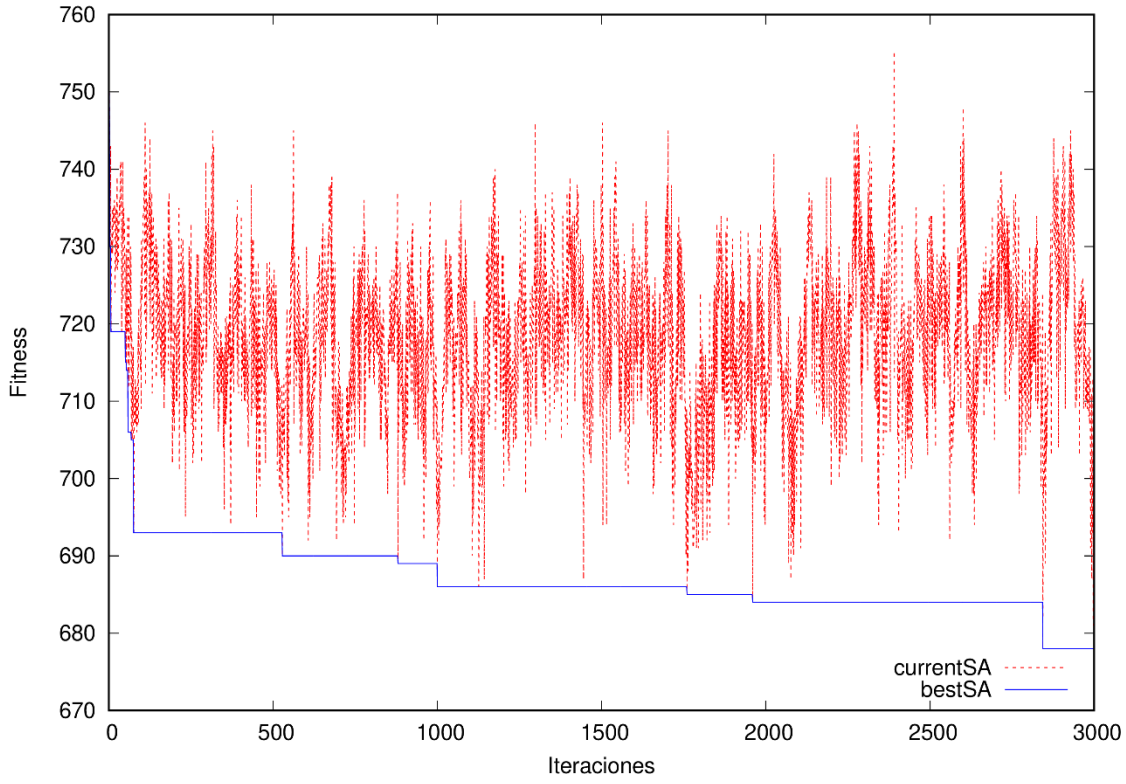

Ilustración 8 Instancia 75 nodos y 60% densidad con 25 nodos de eliminación SA

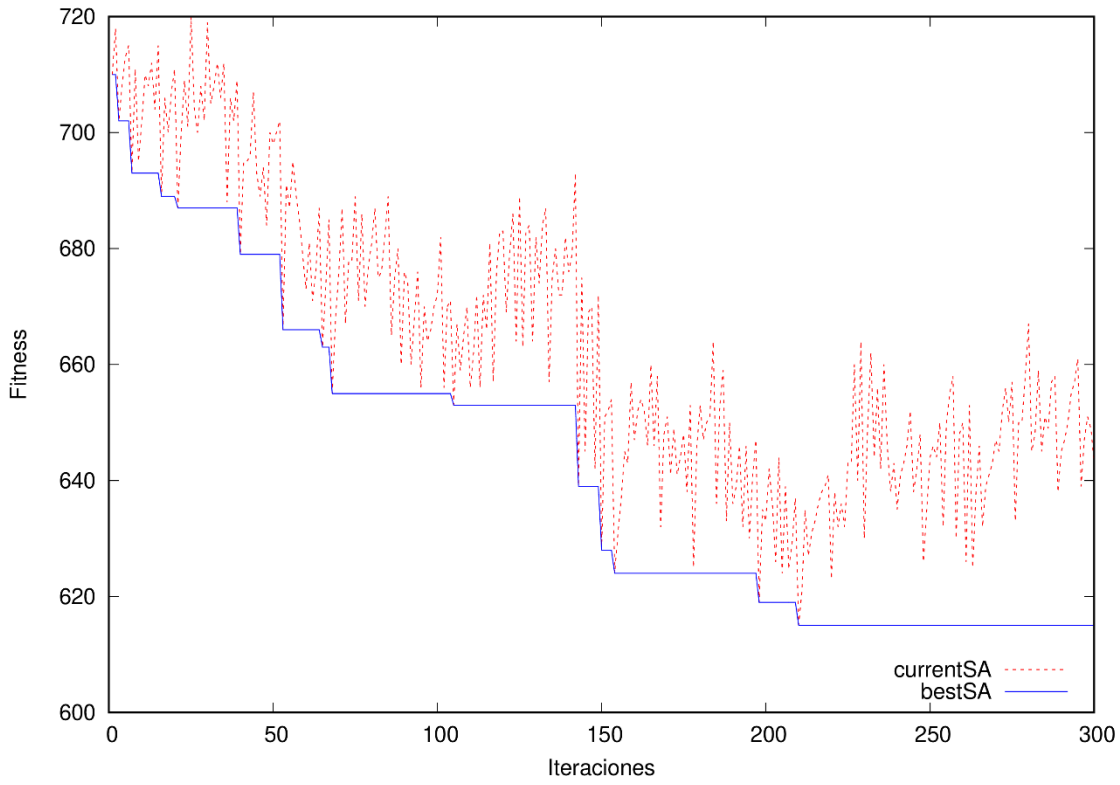

Ilustración 9 Instancia 100 nodos y 30% densidad con 30 nodos de eliminación SA

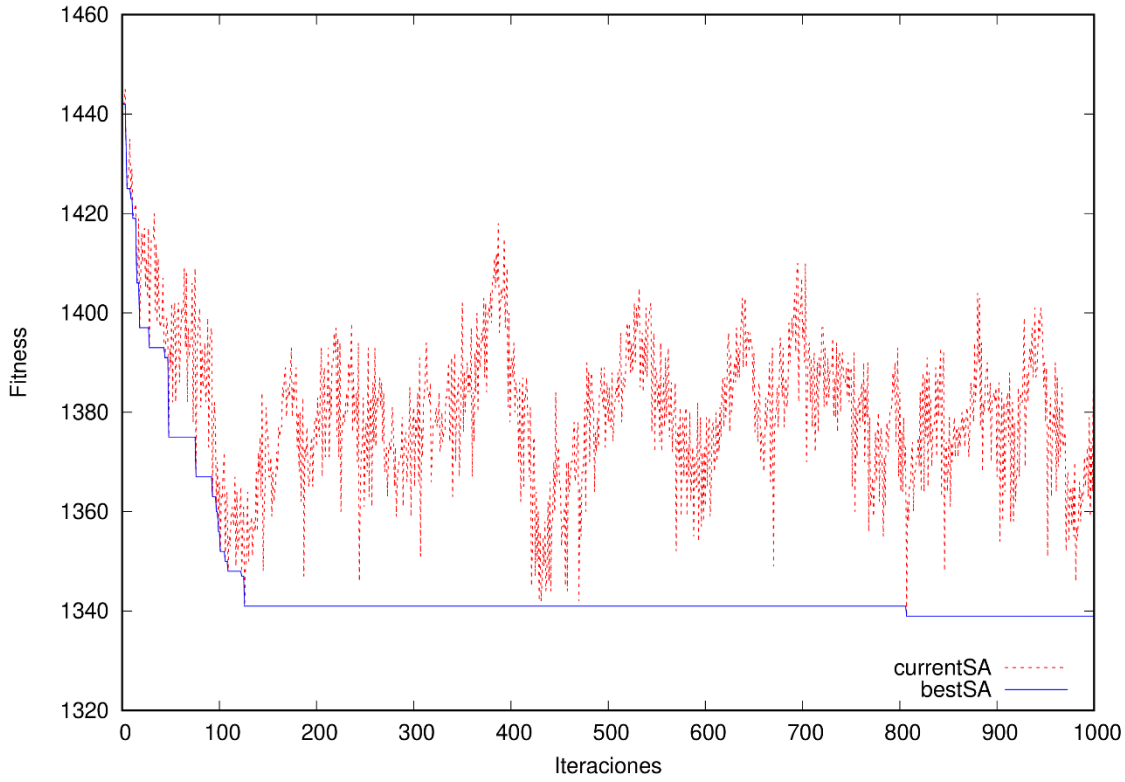

Ilustración 10 Instancia 100 nodos y 60% densidad con 30 nodos de eliminación SA

Búsqueda Tabú

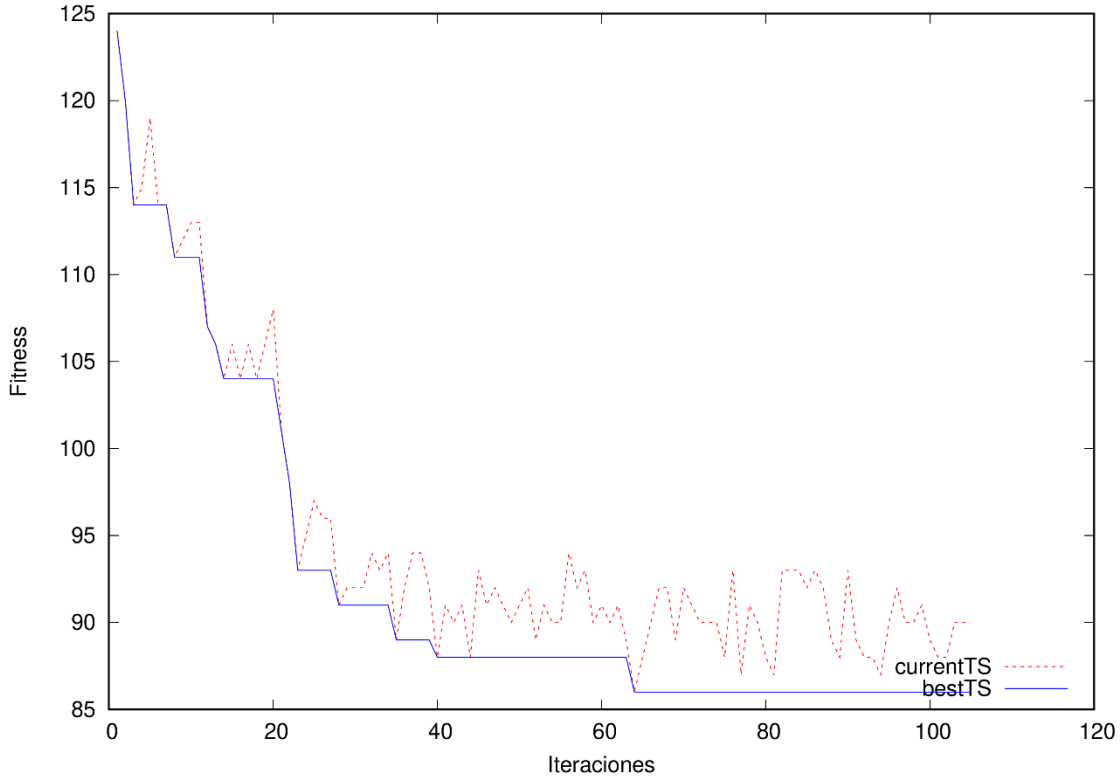

Ilustración 11 Instancia 50 nodos y 30% densidad con 20 nodos de eliminación TS

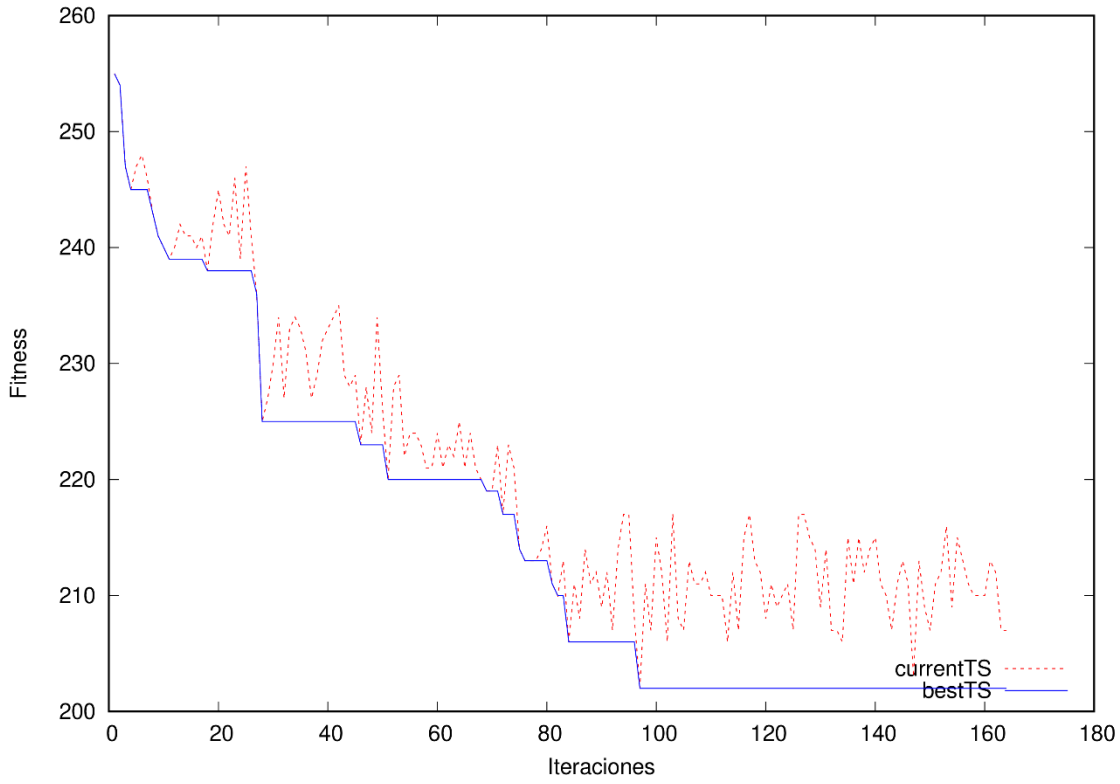

Ilustración 12 Instancia 50 nodos y 60% densidad con 20 nodos de eliminación TS

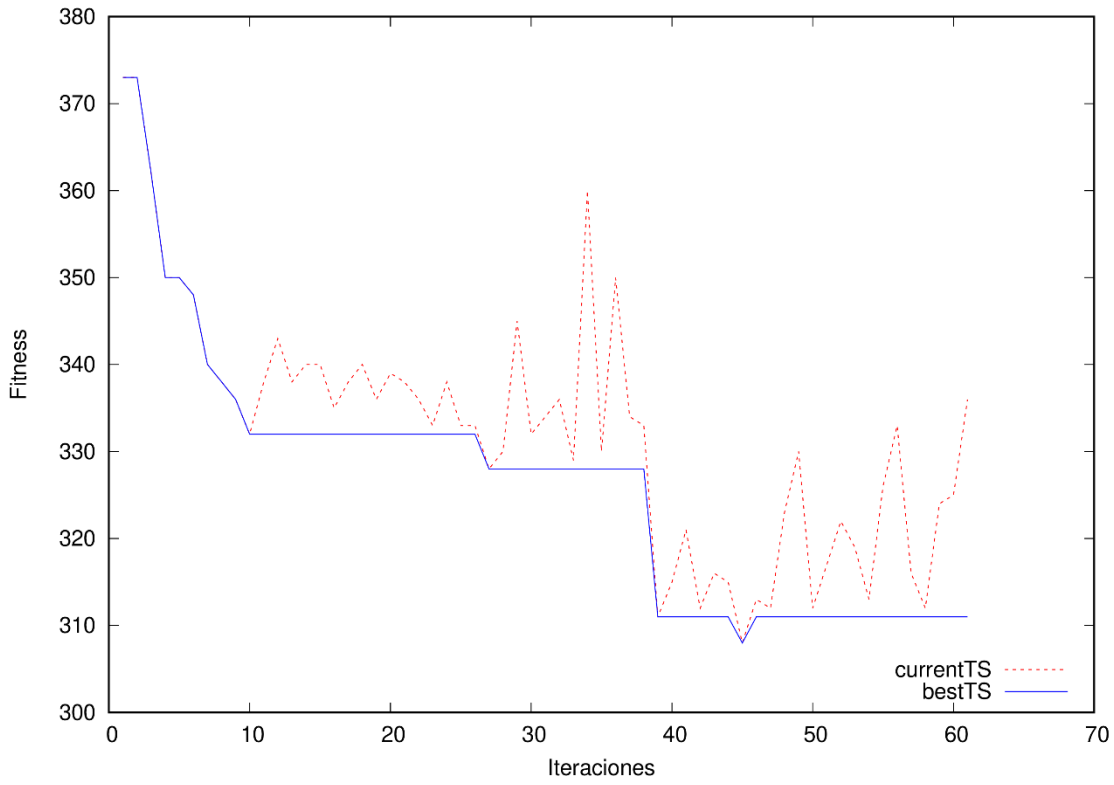

Ilustración 13 Instancia 75 nodos y 30% densidad con 25 nodos de eliminación TS

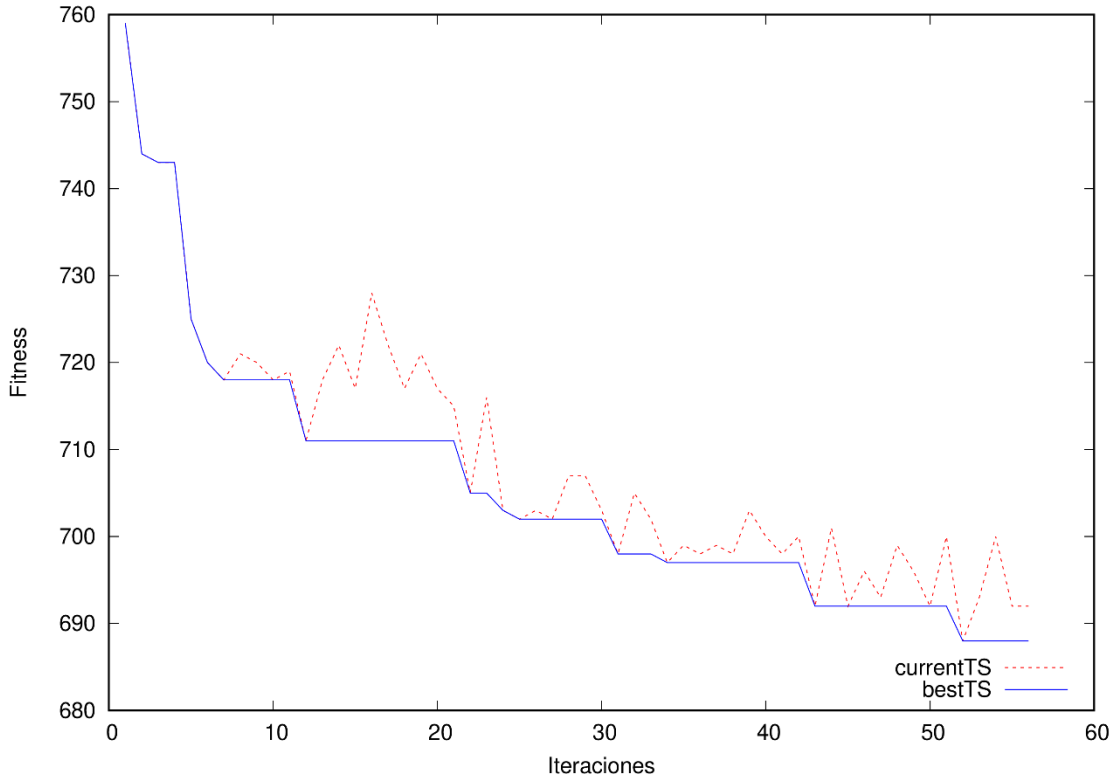

Ilustración 14 Instancia 75 nodos y 60% densidad con 25 nodos de eliminación TS

Greedy Iterativo

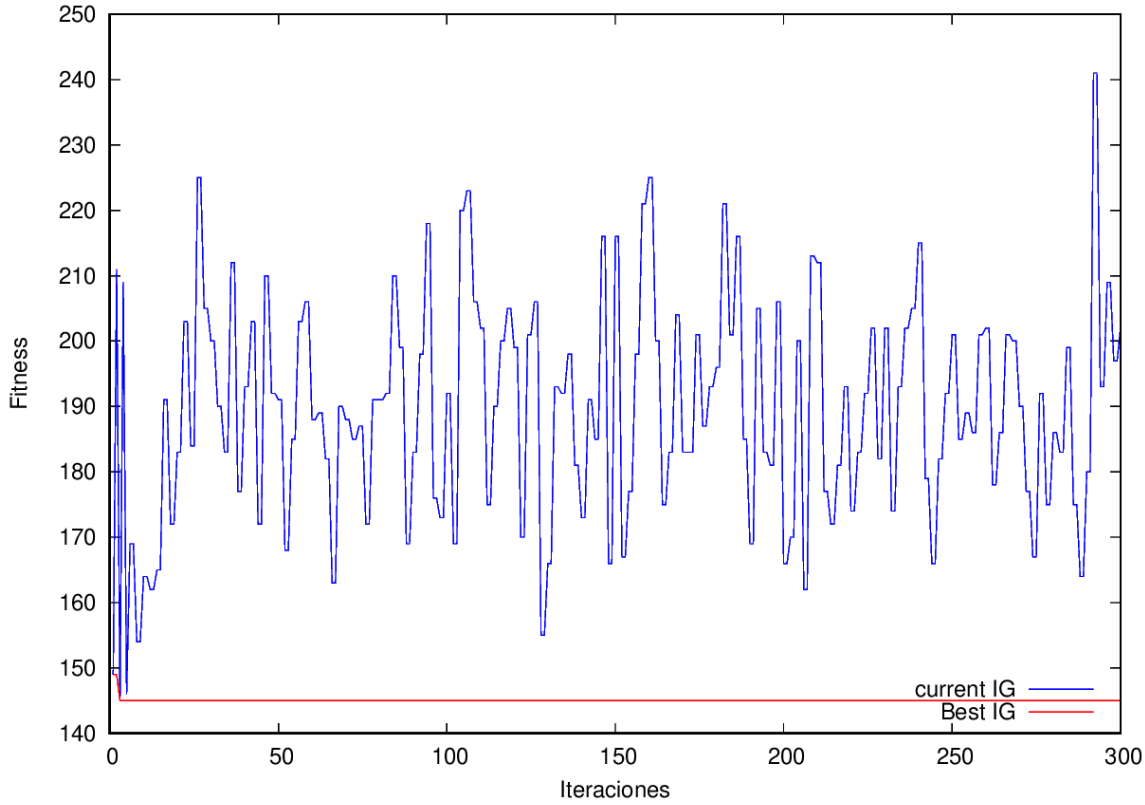

Ilustración 15: Instancia 50 nodos y 30% densidad con 20 nodos de eliminación IG

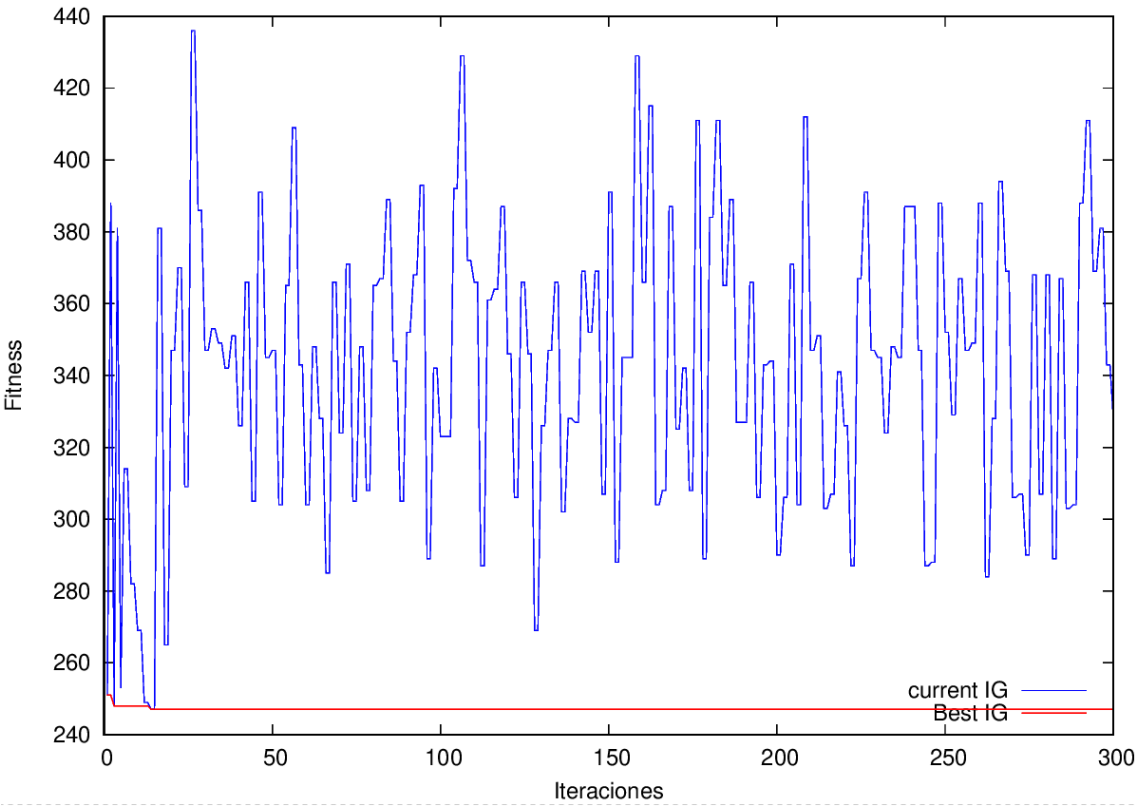

Ilustración 16: Instancia 50 nodos y 60% densidad con 20 nodos de eliminación IG

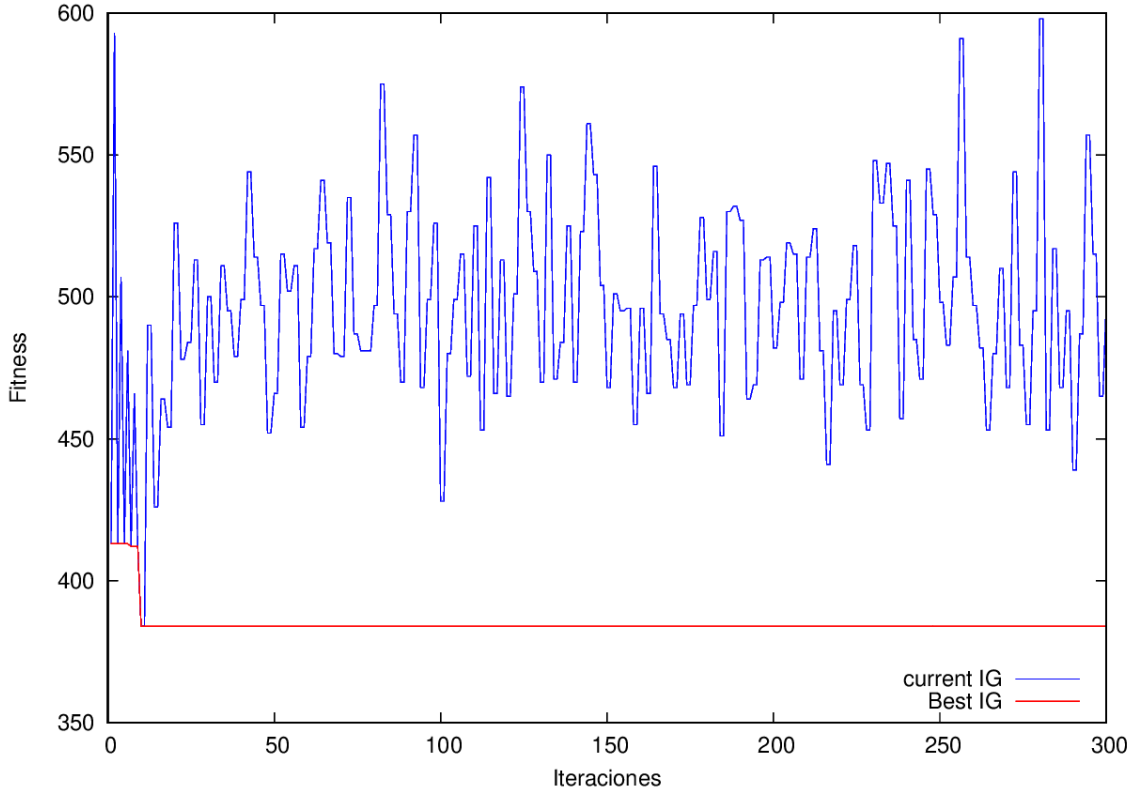

Ilustración 17: Instancia 75 nodos y 30% densidad con 25 nodos de eliminación IG

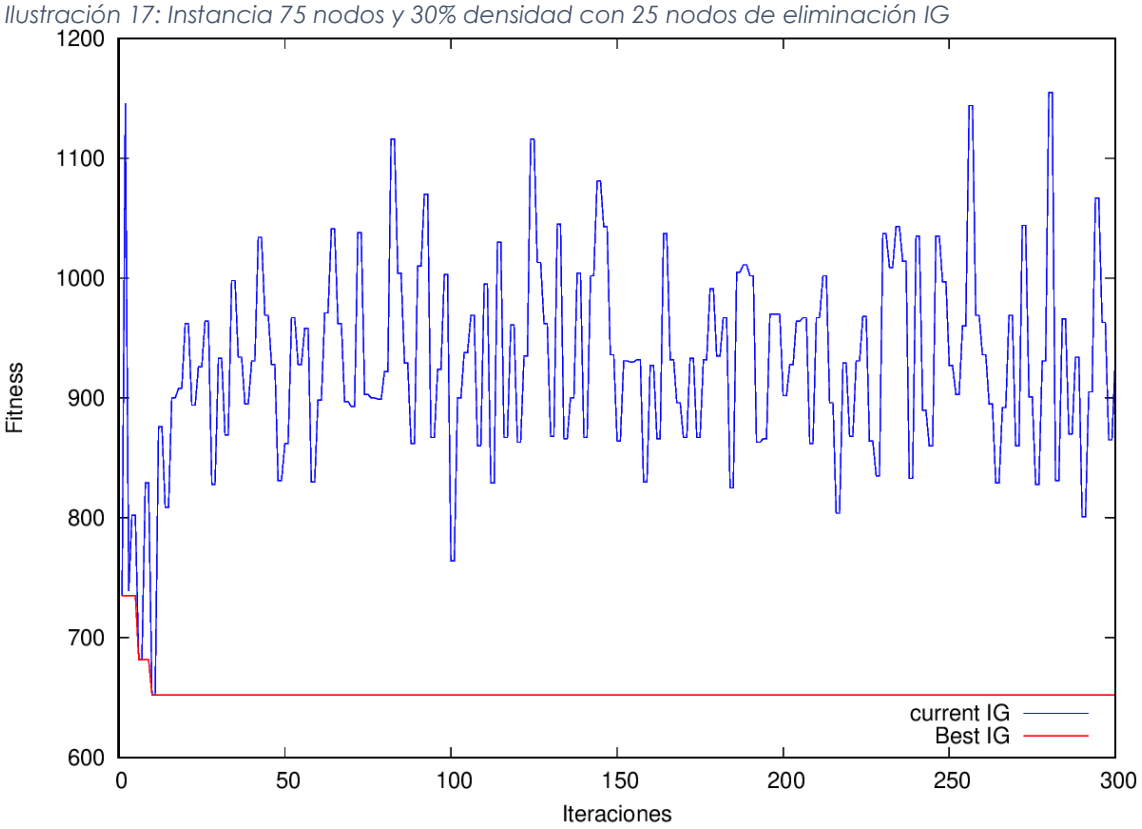

Ilustración 18: Instancia 75 nodos y 60% densidad con 25 nodos de eliminación IG

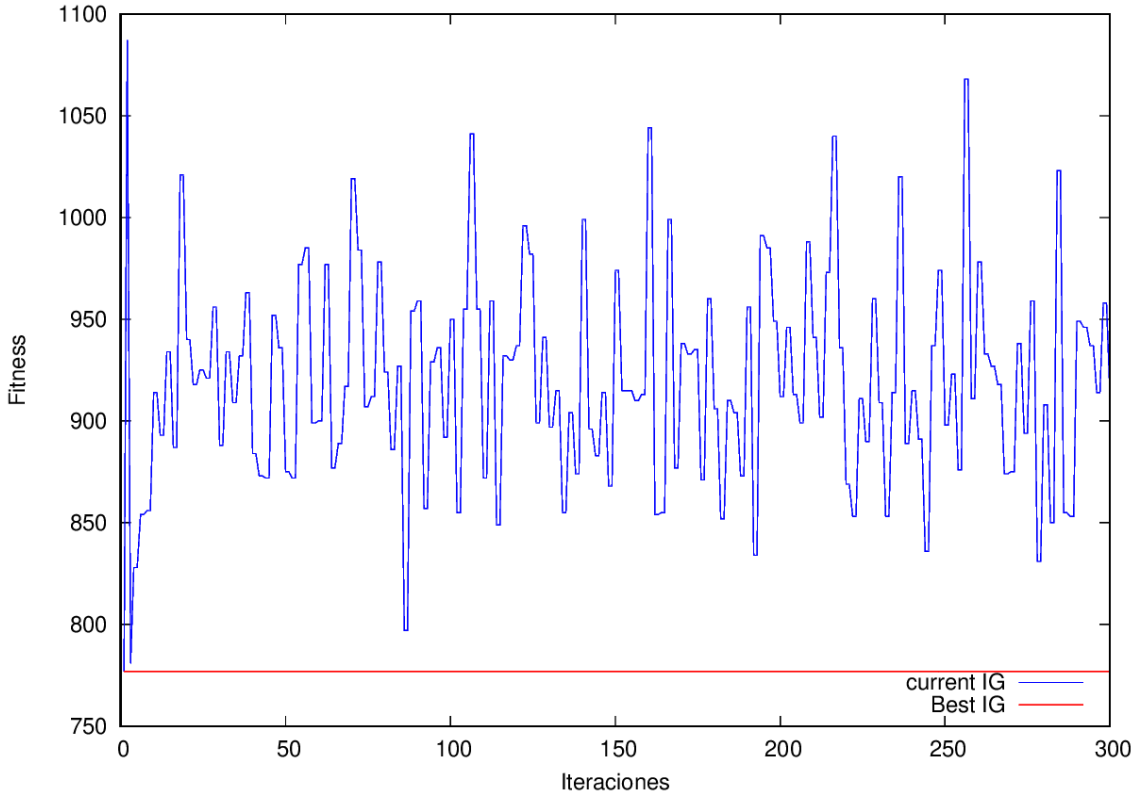

Ilustración 19: Instancia 100 nodos y 30% densidad con 30 nodos de eliminación IG

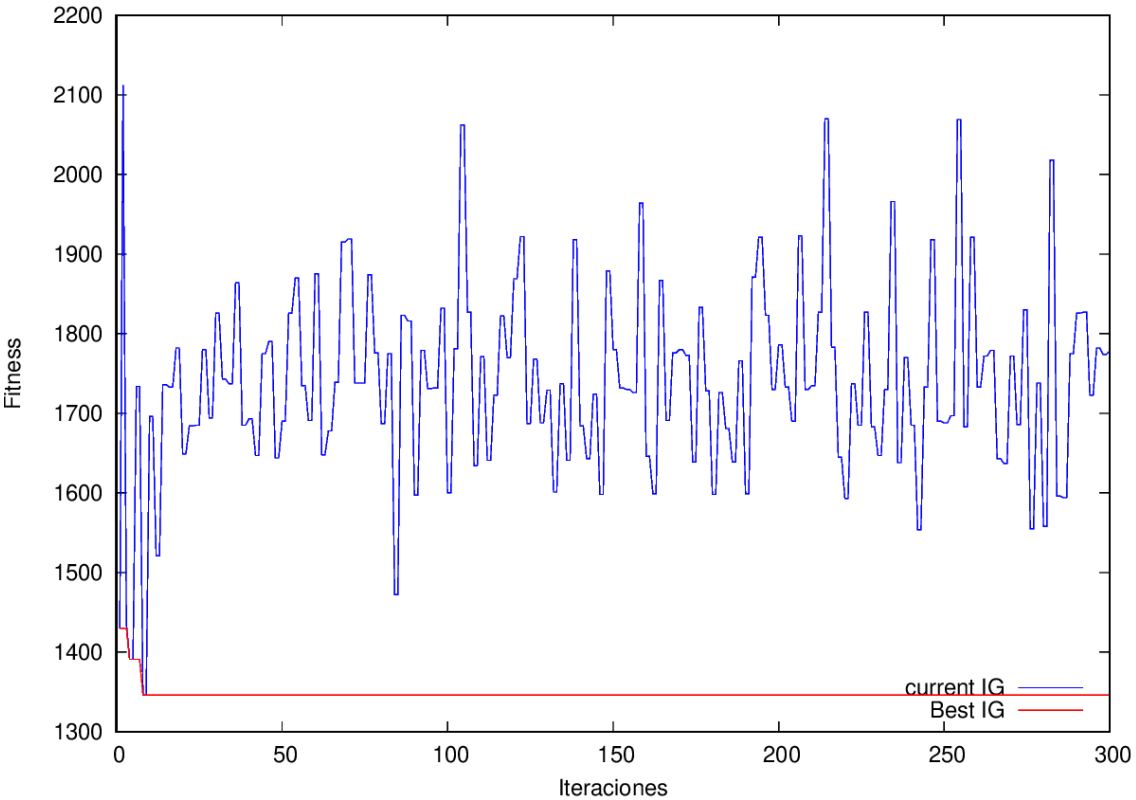

Ilustración 20: Instancia 100 nodos y 60% densidad con 30 nodos de eliminación IG

Greedy Iterativo con probabilidad condicionada

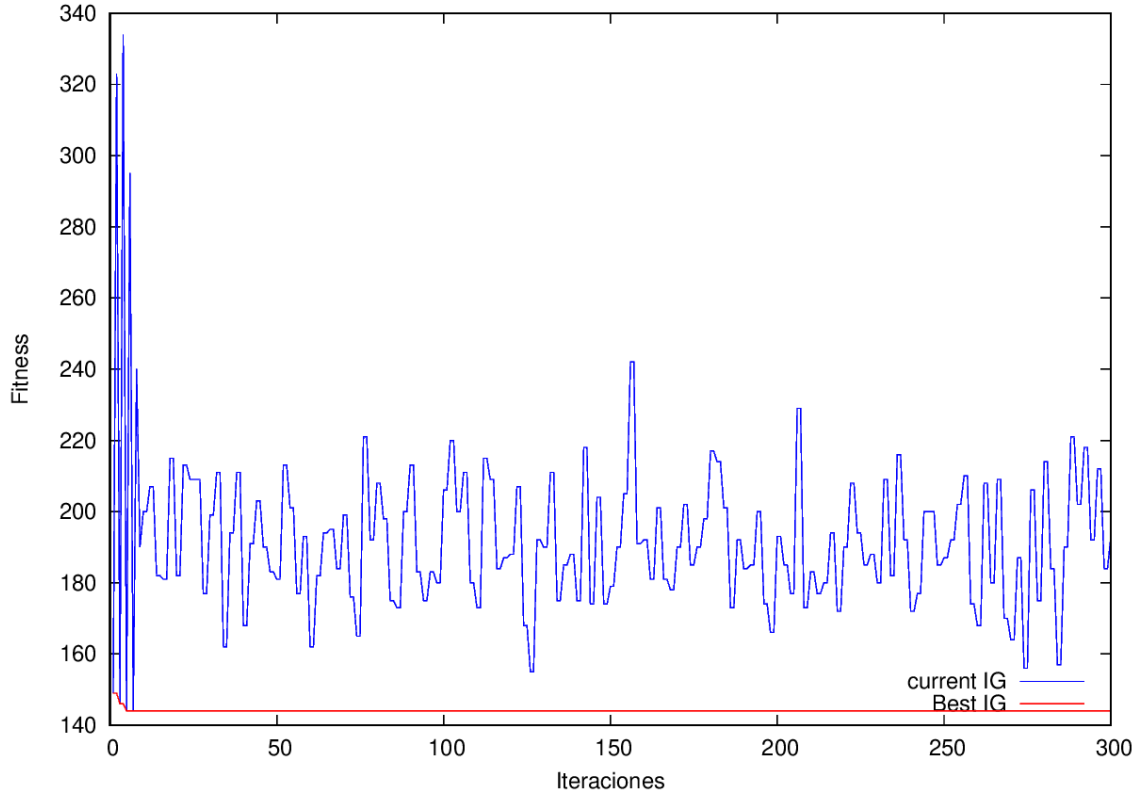

Ilustración 21: Instancia 50 nodos y 30% densidad con 20 nodos de eliminación IGM

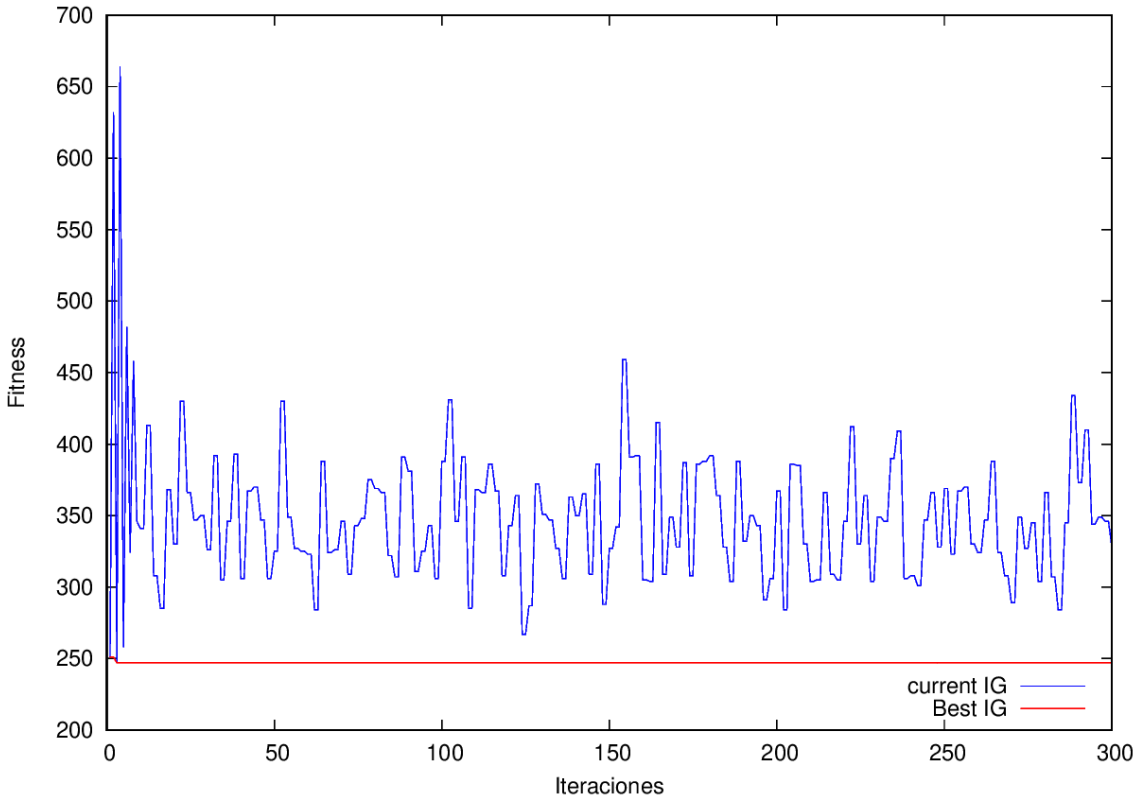

Ilustración 22: Instancia 50 nodos y 60% densidad con 20 nodos de eliminación IGM

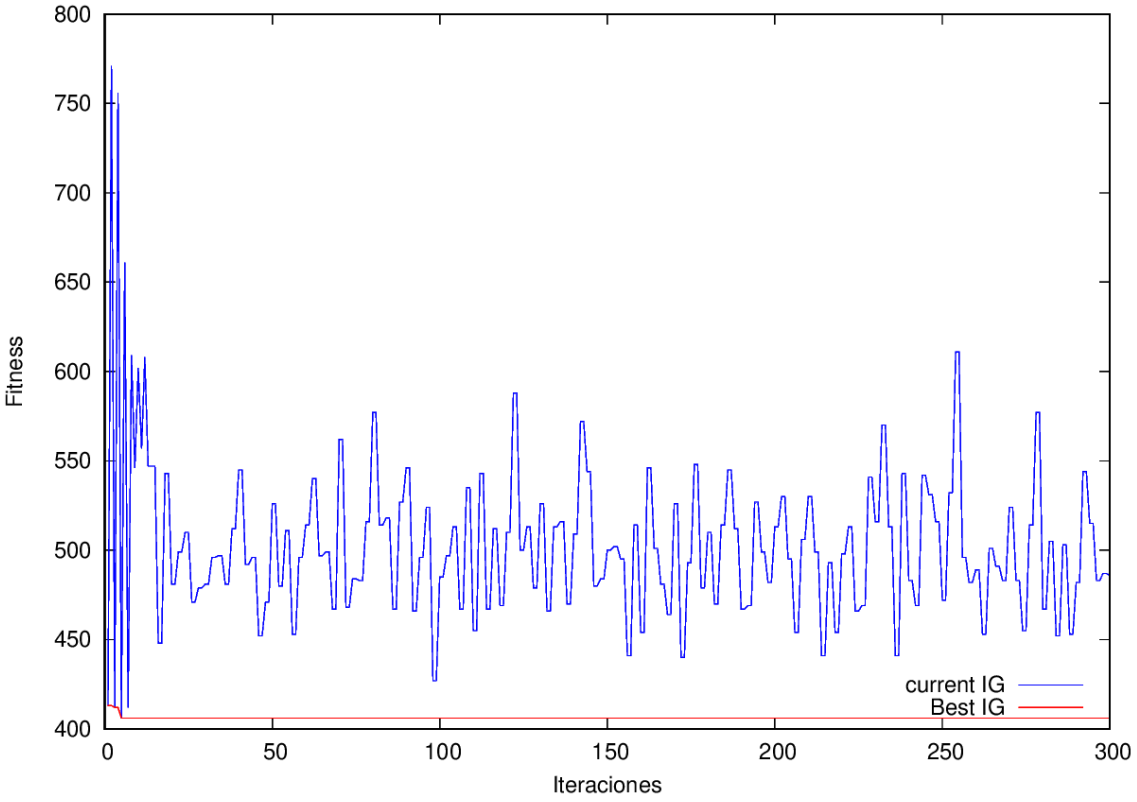

Ilustración 23: Instancia 75 nodos y 30% densidad con 25 nodos de eliminación IGM

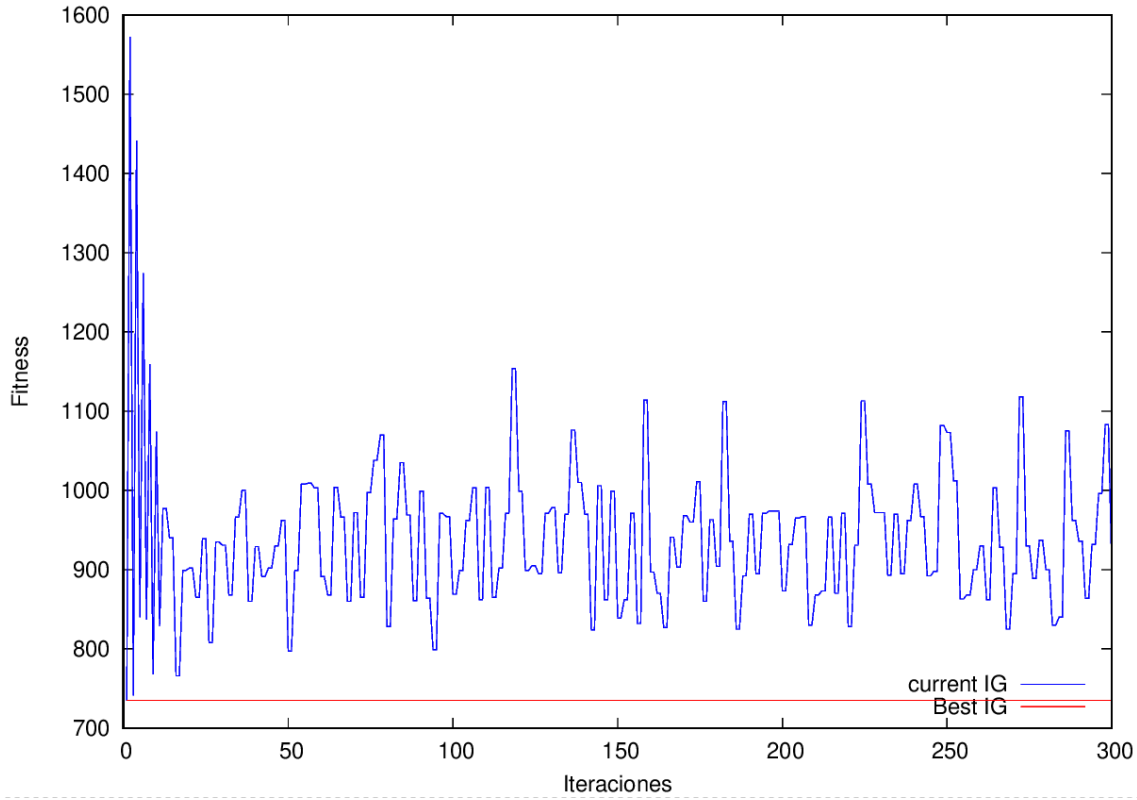

Ilustración 24: Instancia 75 nodos y 60% densidad con 25 nodos de eliminación IGM

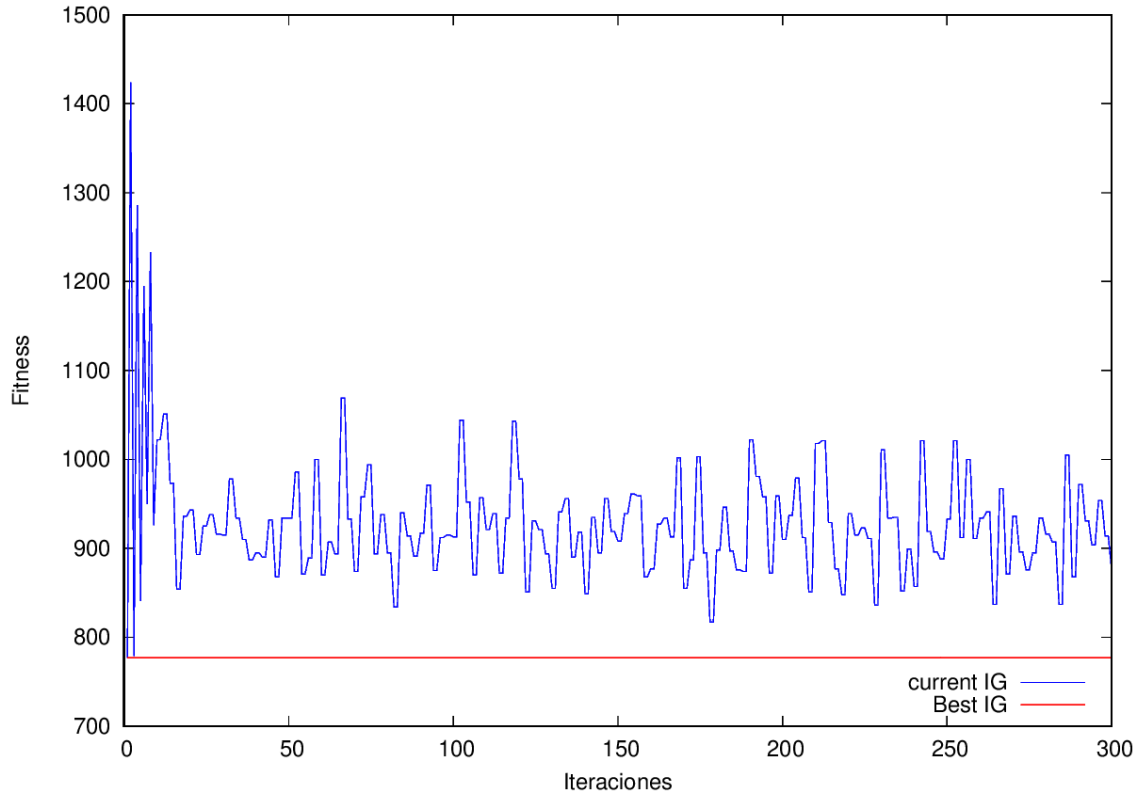

Ilustración 25: Instancia 100 nodos y 30% densidad con 30 nodos de eliminación IGM

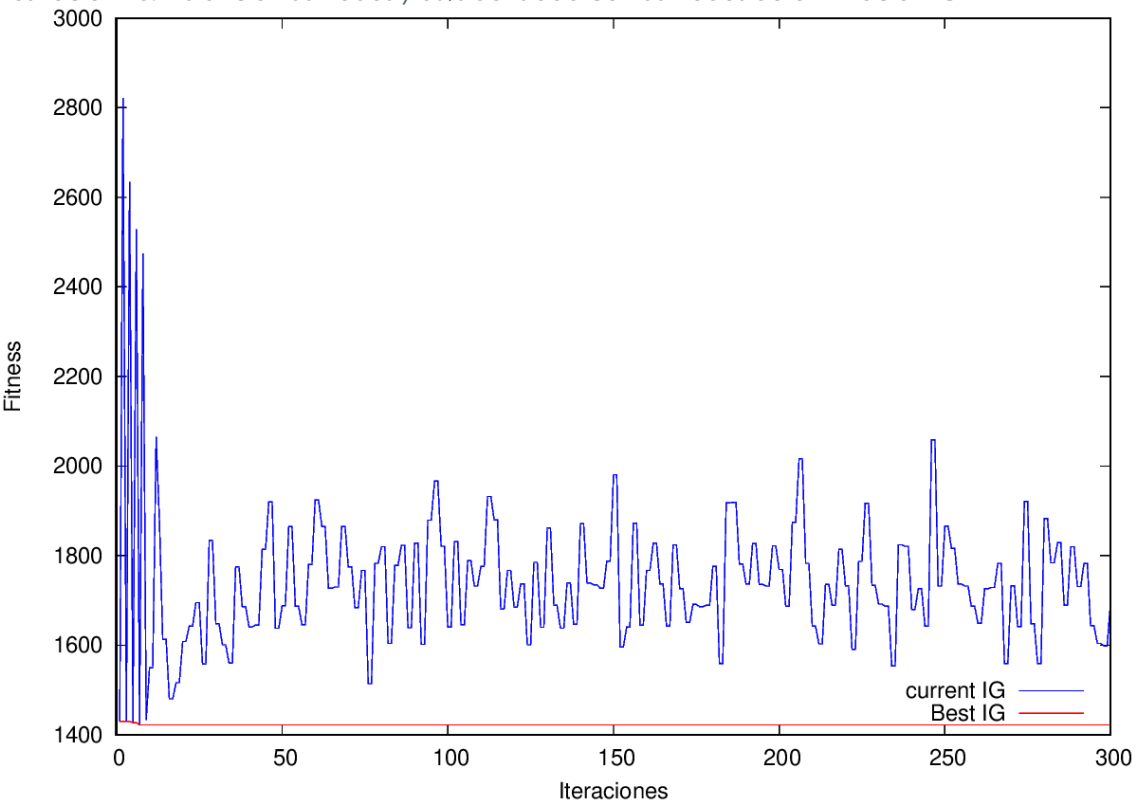

Ilustración 26: Instancia 100 nodos y 60% densidad con 30 nodos de eliminación IGM

Algoritmo genético

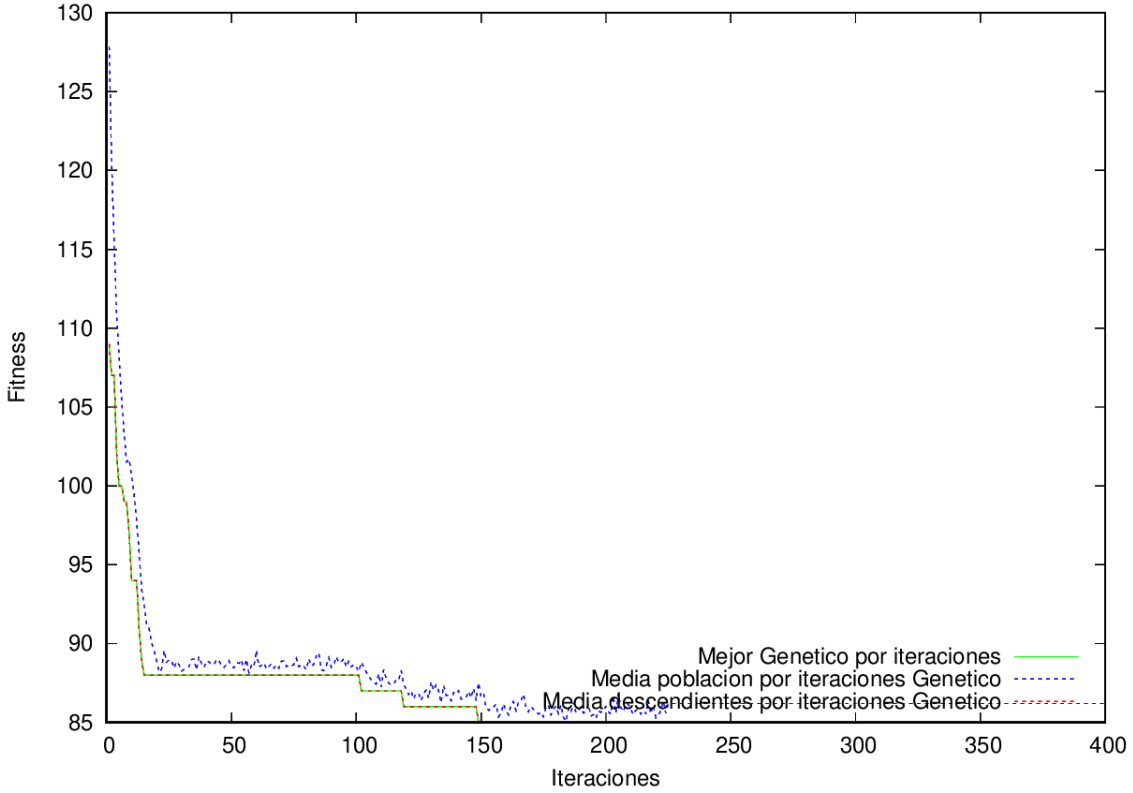

Ilustración 27: Medias Genético 50 nodos y 30% densidad con eliminación de 20 nodos

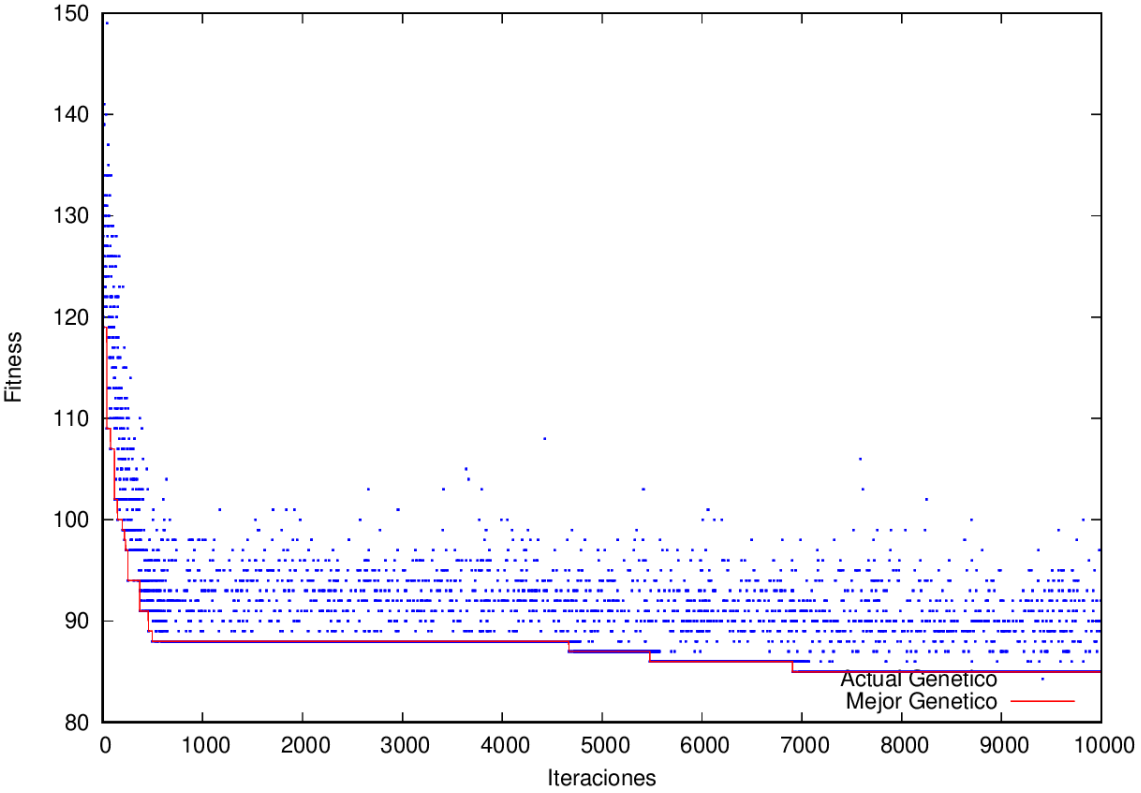

Ilustración 28: GA 50 nodos y 30% densidad con eliminación de 20 nodos

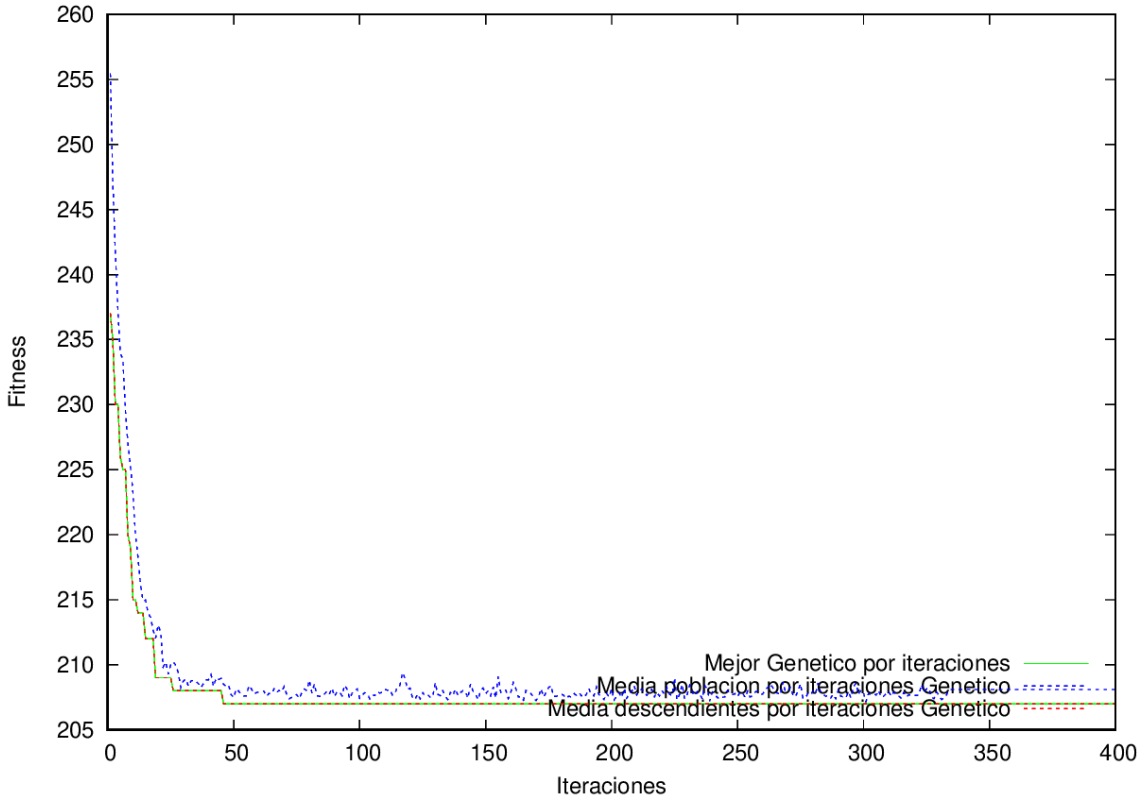

Ilustración 29: Medias Genético 50 nodos y 60% densidad con eliminación de 20 nodos

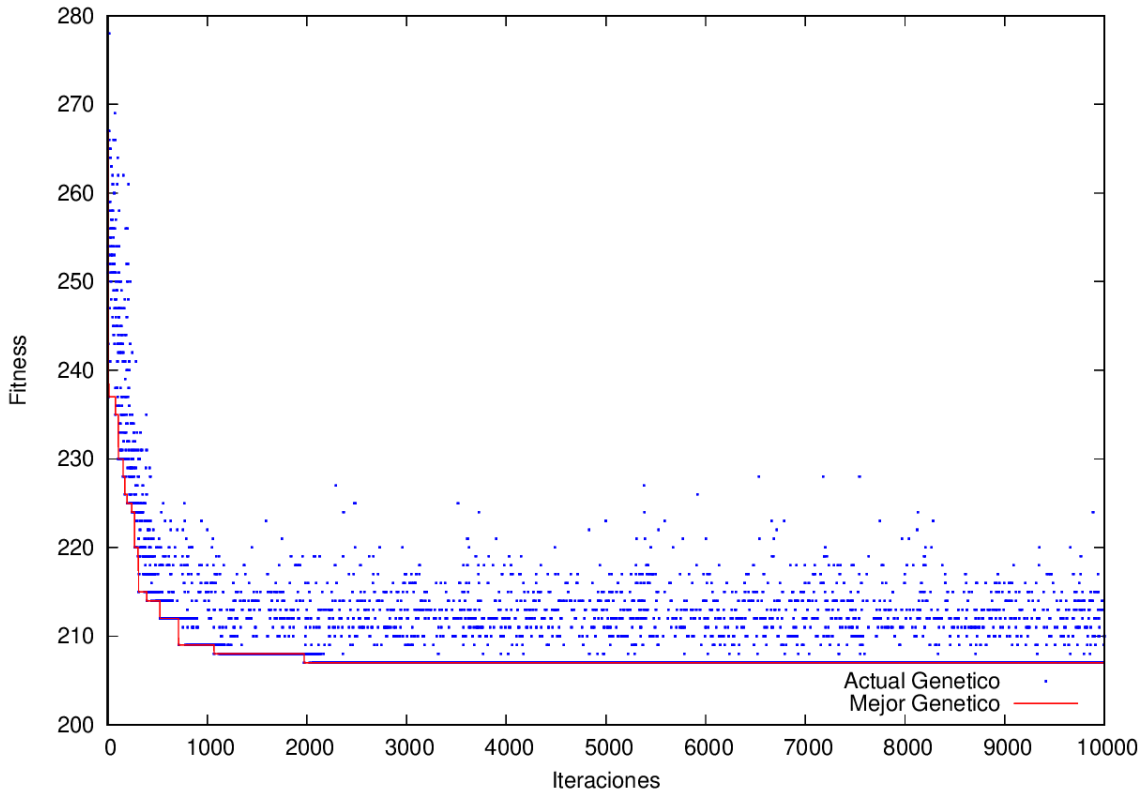

Ilustración 30: GA 50 nodos y 60% densidad con eliminación de 20 nodos

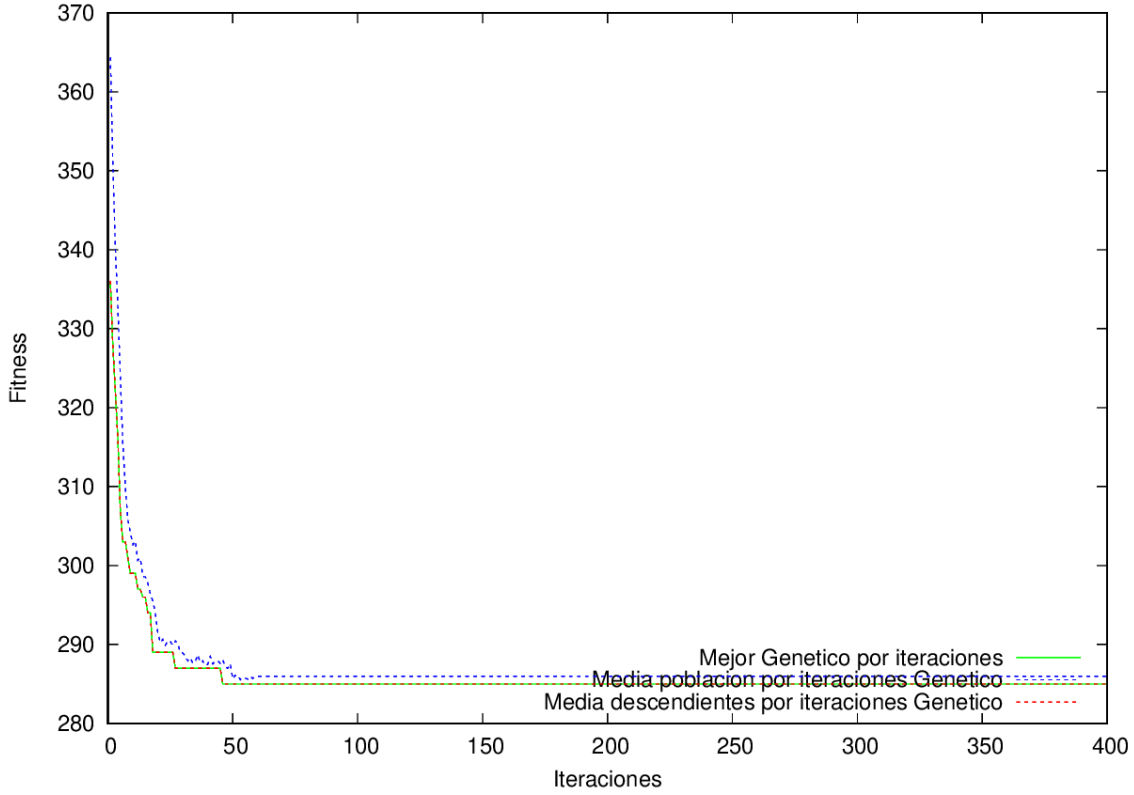

Ilustración 31: Medias Genético 75 nodos y 30% densidad con eliminación de 25 nodos

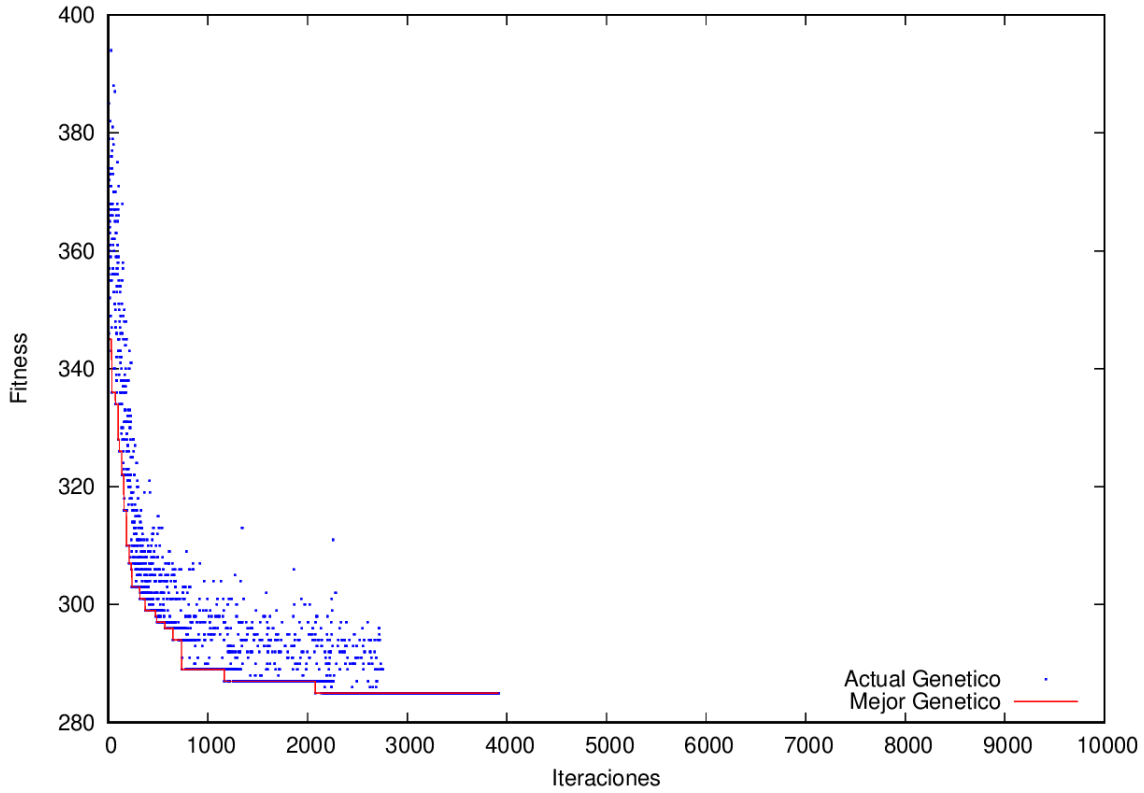

Ilustración 32: GA 75 nodos y 30% densidad con eliminación de 25 nodos

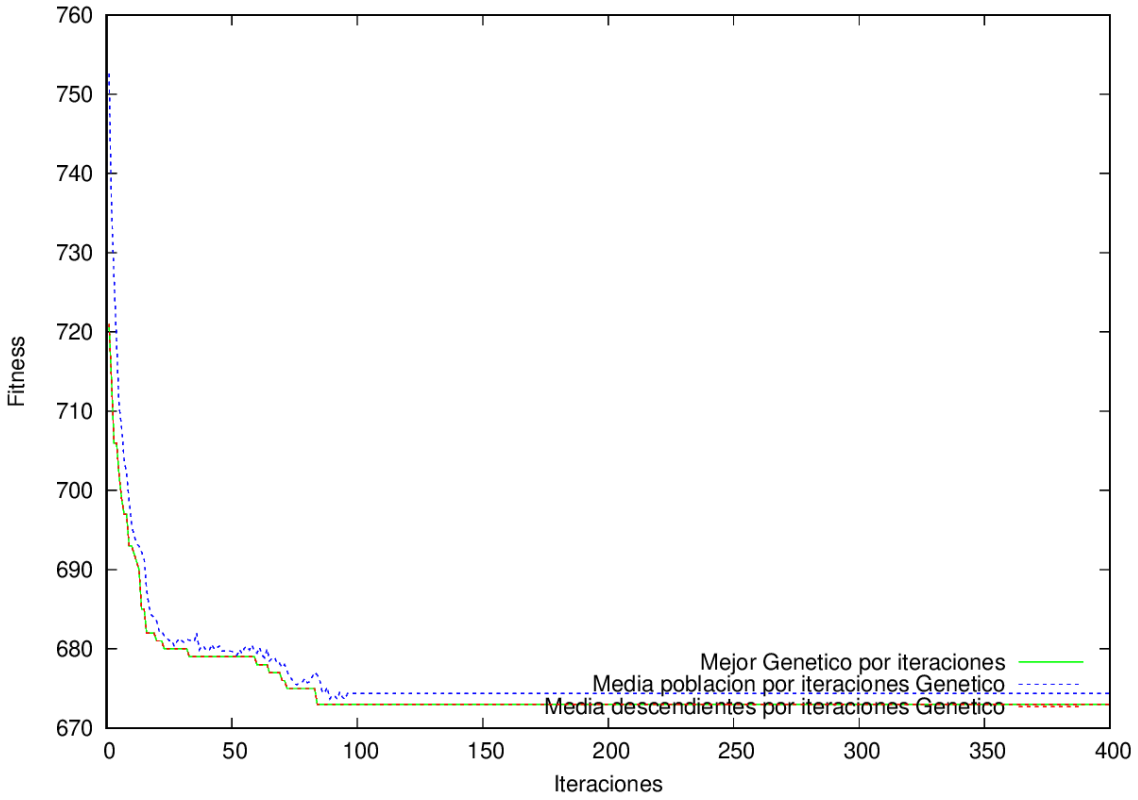

Ilustración 33: Medias Genético 75 nodos y 60% densidad con eliminación de 25 nodos

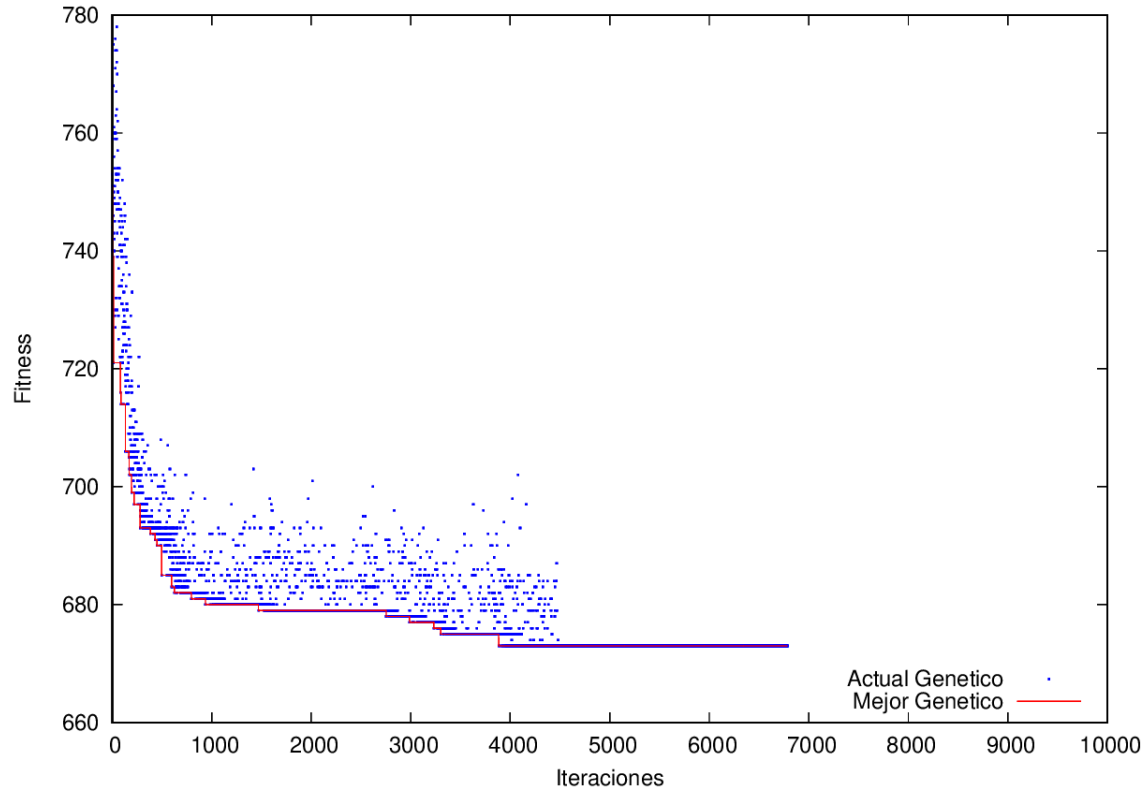

Ilustración 34: GA 75 nodos y 60% densidad con eliminación de 25 nodos

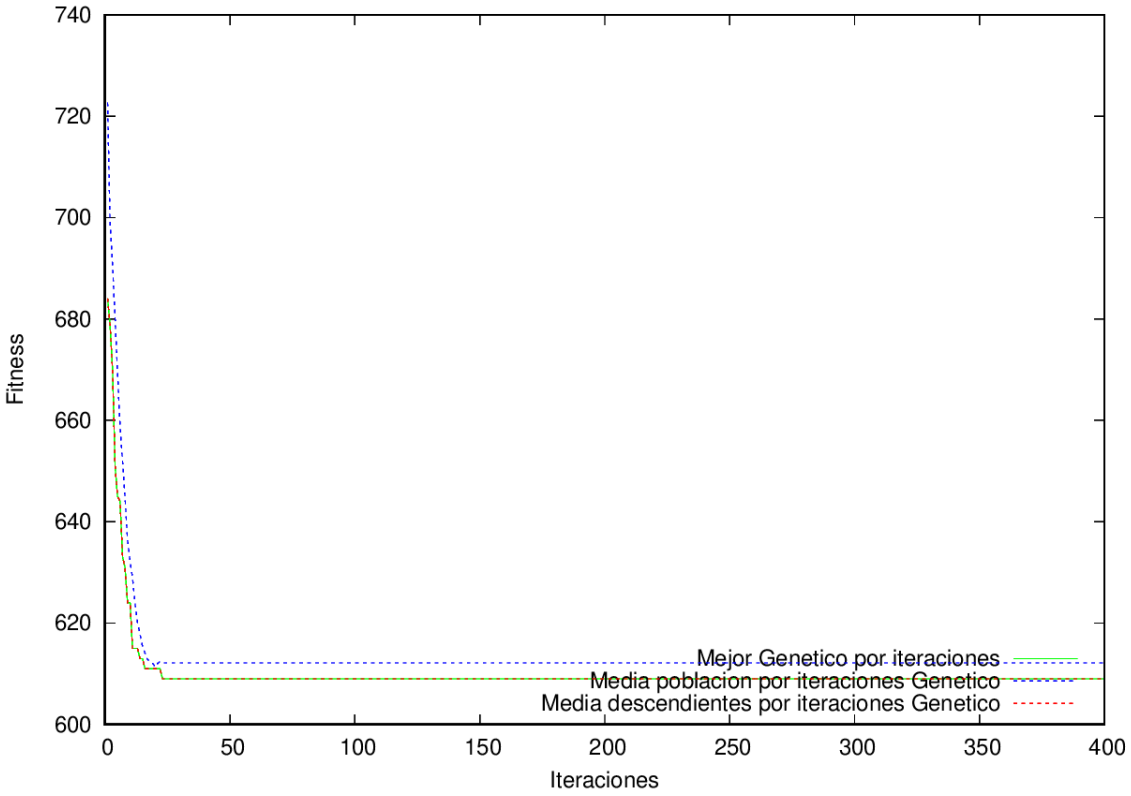

Ilustración 35: Medias Genético 100 nodos y 30% densidad con eliminación de 30 nodos

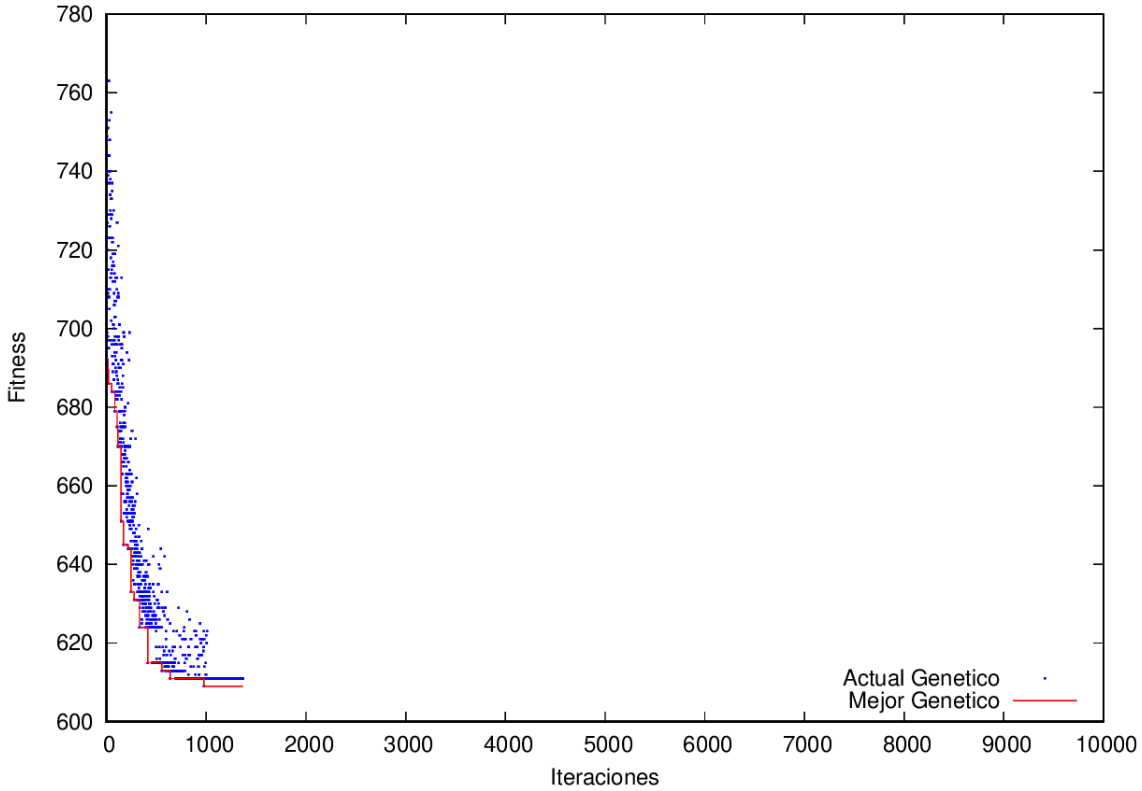

Ilustración 36: GA 100 nodos y 30% densidad con eliminación de 30 nodos

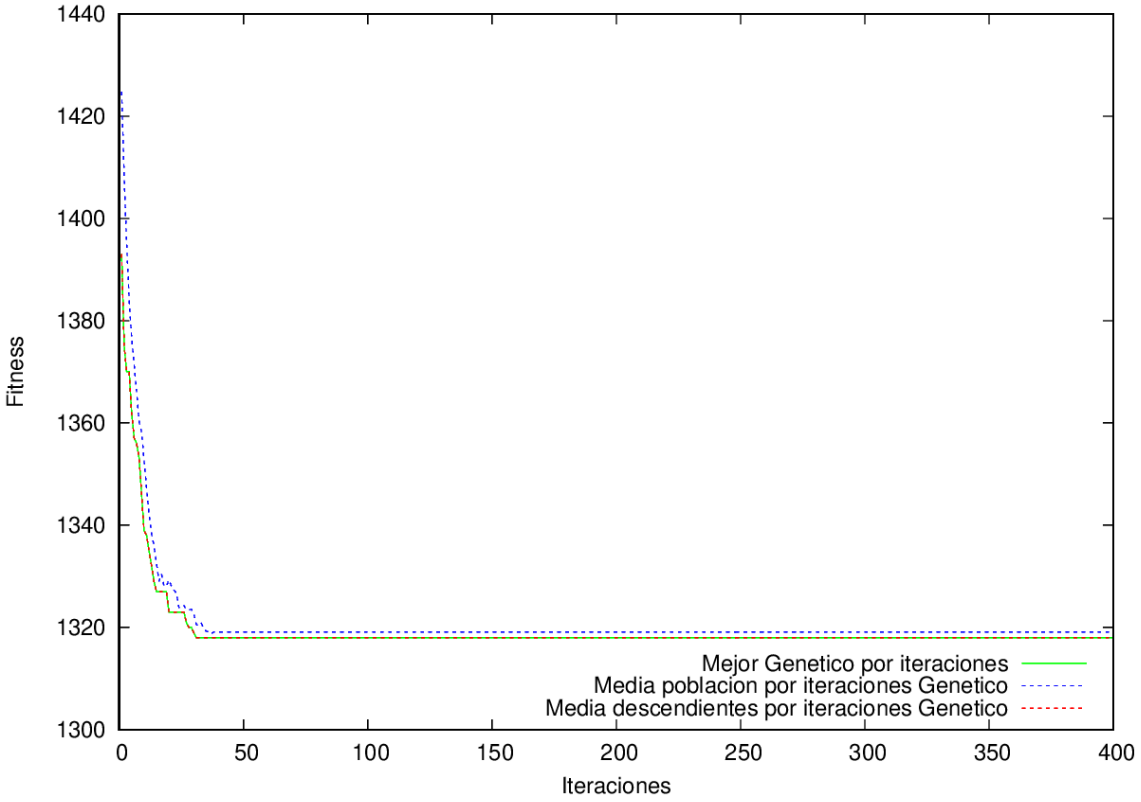

Ilustración 37: Medias Genético 100 nodos y 60% densidad con eliminación de 30 nodos

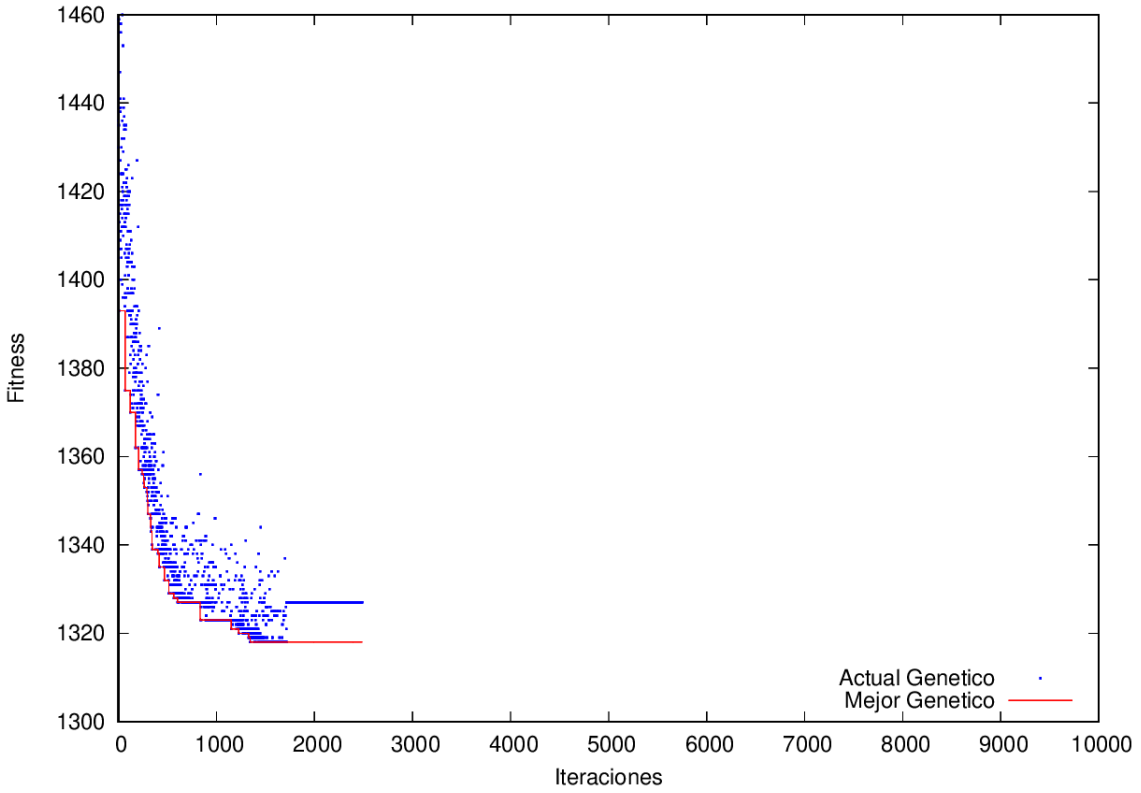

Ilustración 38: GA 100 nodos y 60% densidad con eliminación de 30 nodos

Algoritmo genético con Iterativo Greedy

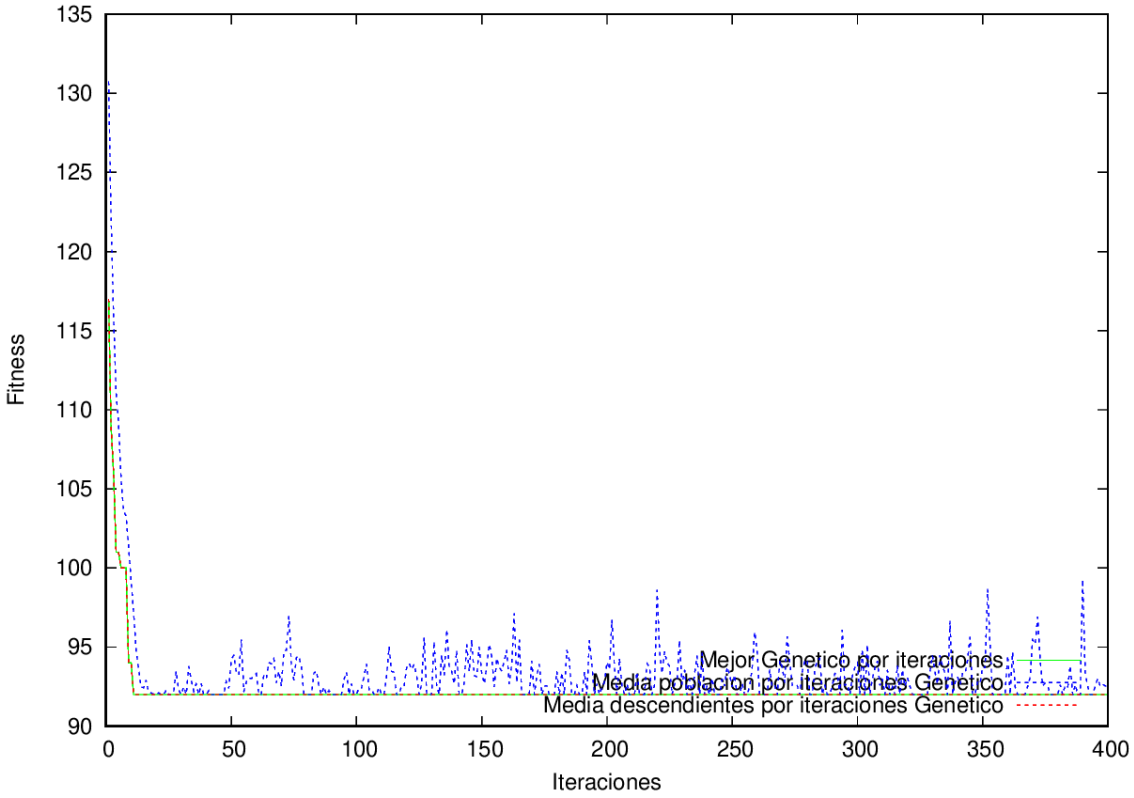

Ilustración 39: Medias Genético IG 50 nodos y 30% densidad con eliminación de 20 nodos

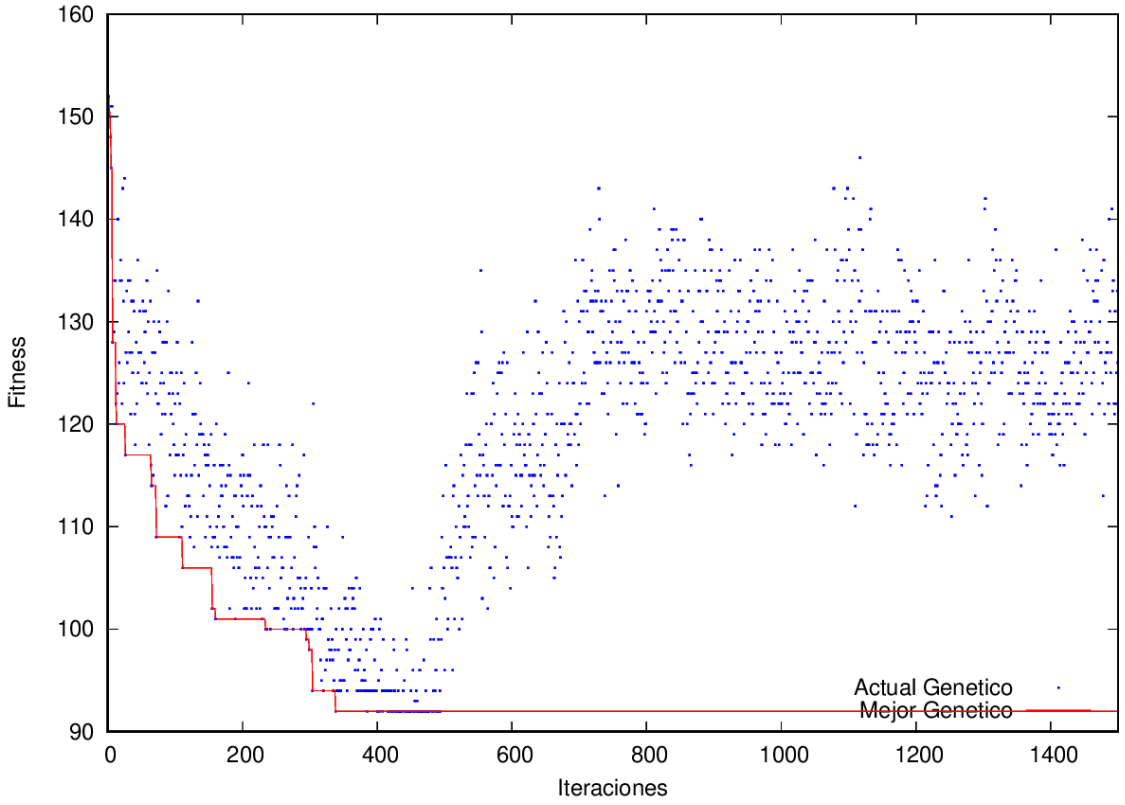

Ilustración 40: GA IG 50 nodos y 30% densidad con eliminación de 20 nodos

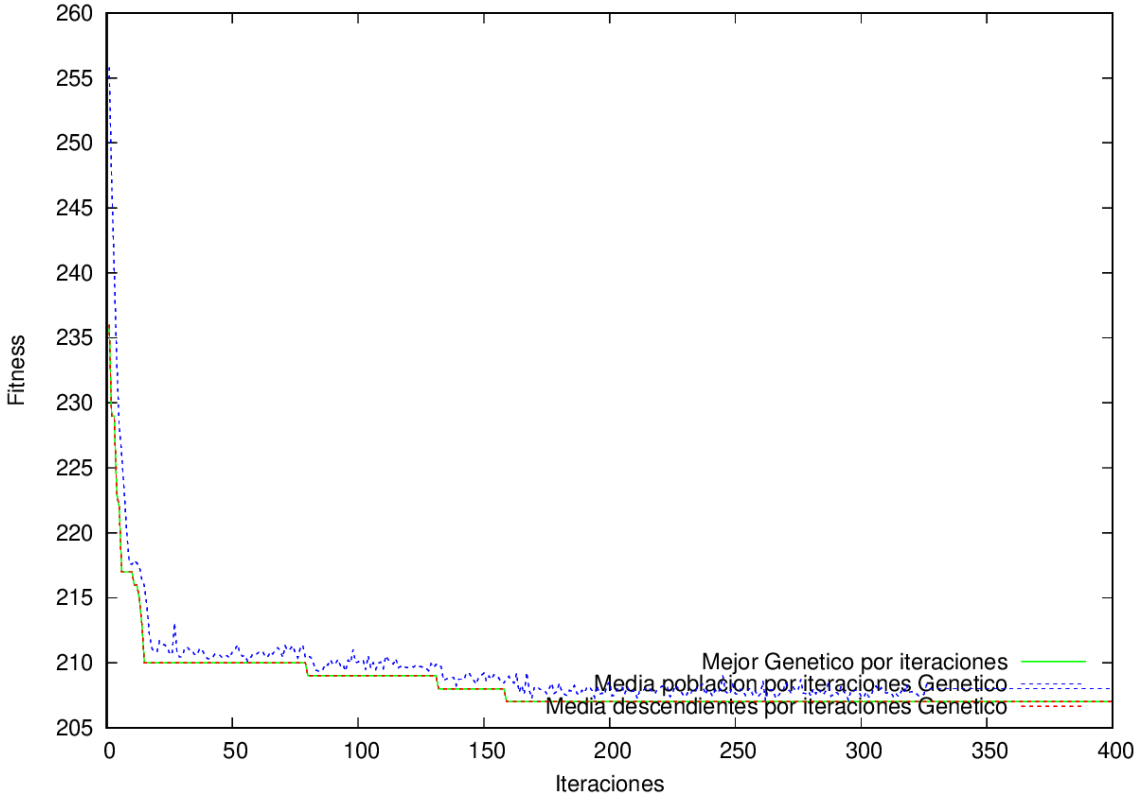

Ilustración 41: Medias Genético IG 50 nodos y 60% densidad con eliminación de 20 nodos

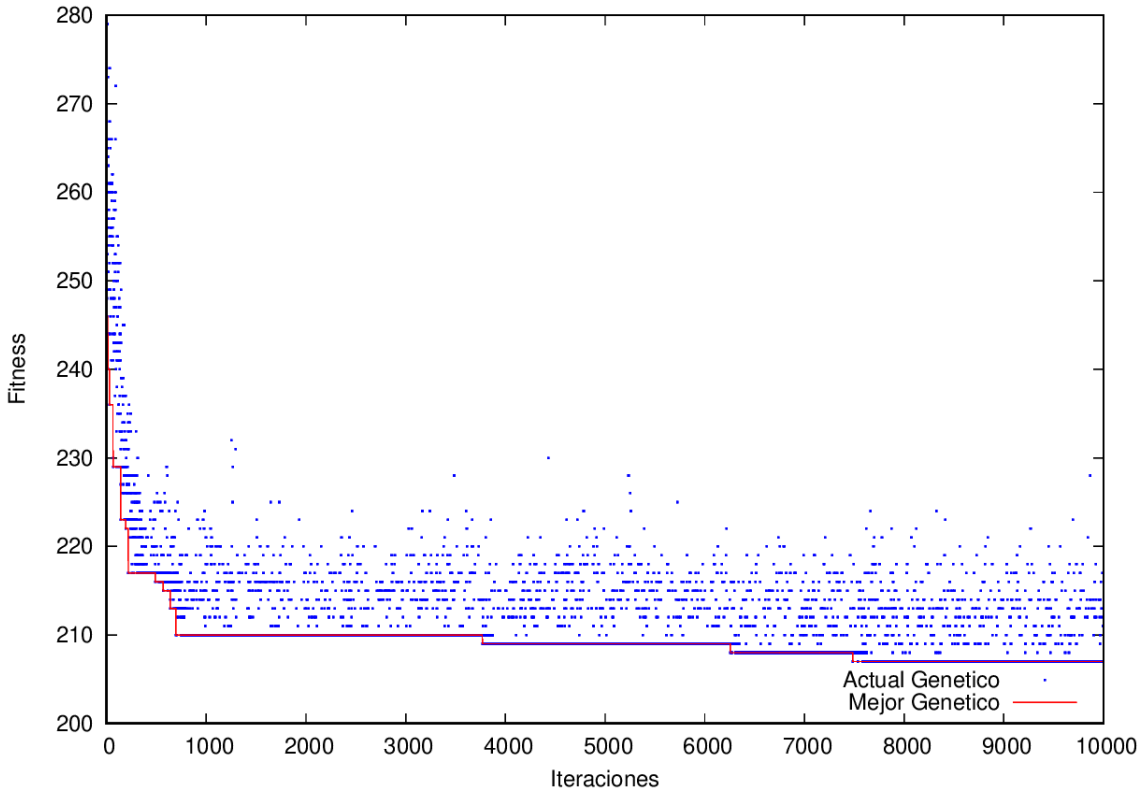

Ilustración 42: GA IG 50 nodos y 60% densidad con eliminación de 20 nodos

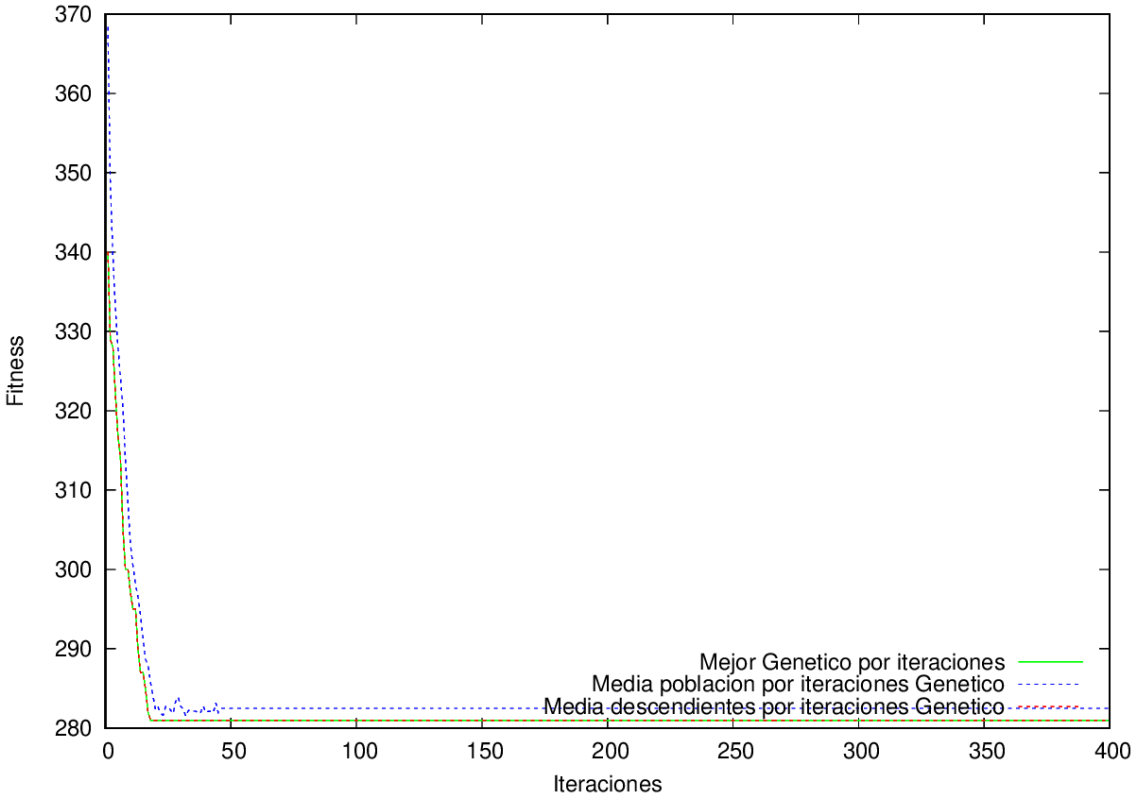

Ilustración 43: Medias Genético IG 75 nodos y 30% densidad con eliminación de 25 nodos

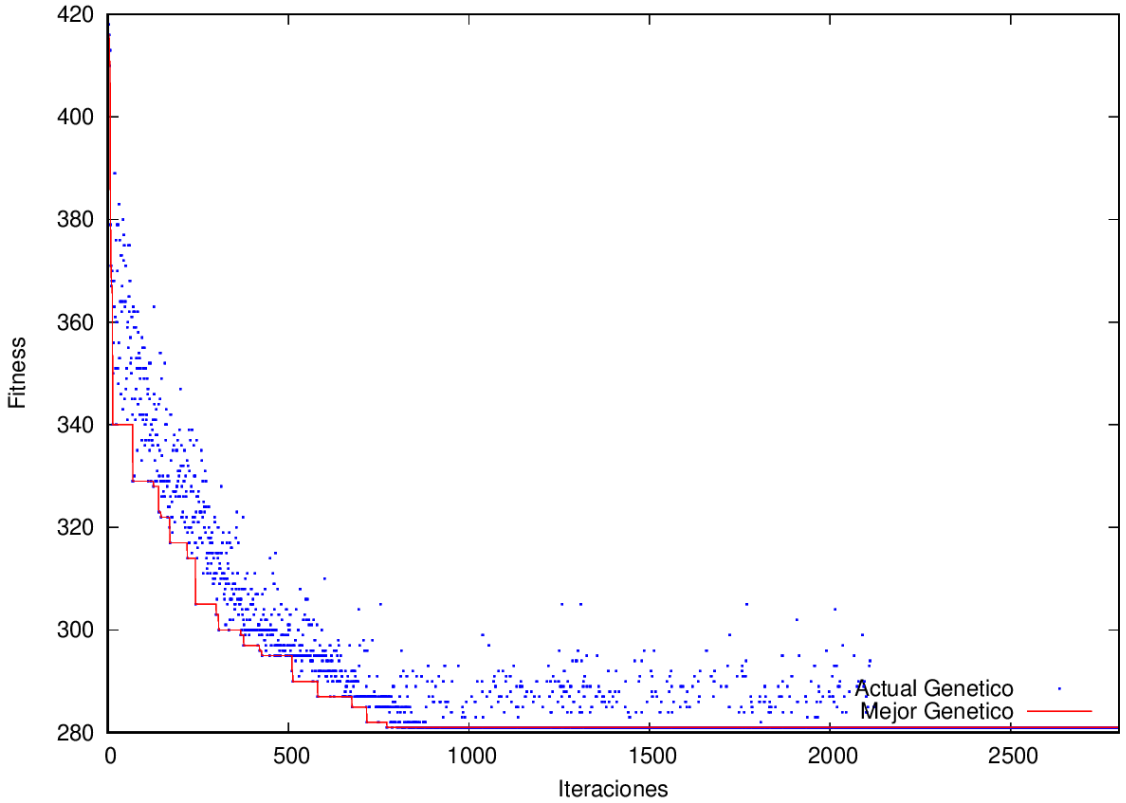

Ilustración 44: GA IG 75 nodos y 30% densidad con eliminación de 25 nodos

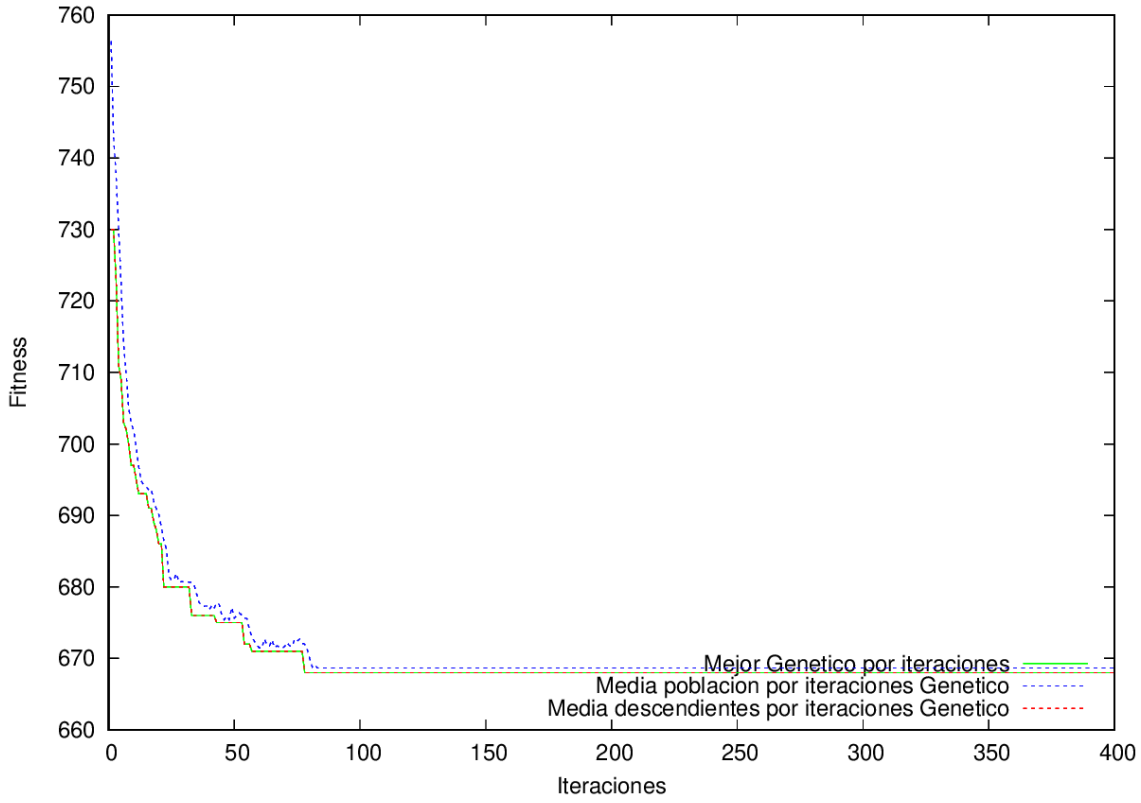

Ilustración 45: Medias Genético IG 75 nodos y 60% densidad con eliminación de 25 nodos

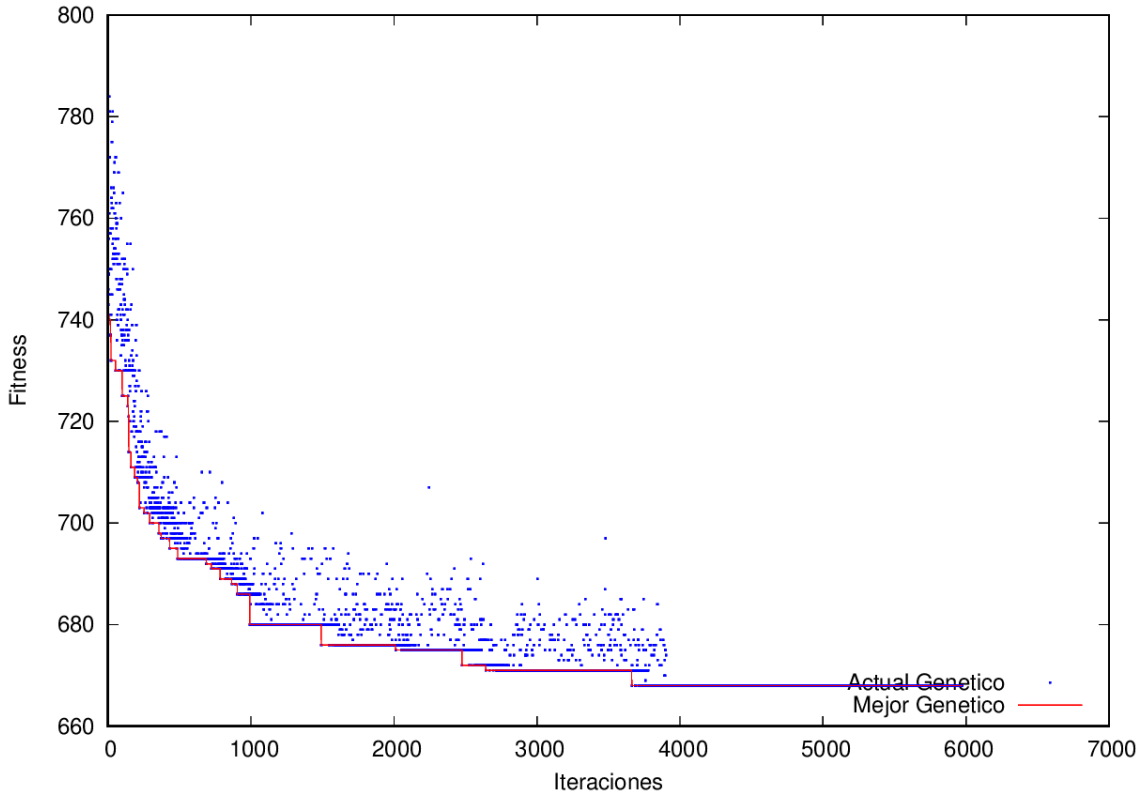

Ilustración 46: GA IG 75 nodos y 60% densidad con eliminación de 25 nodos

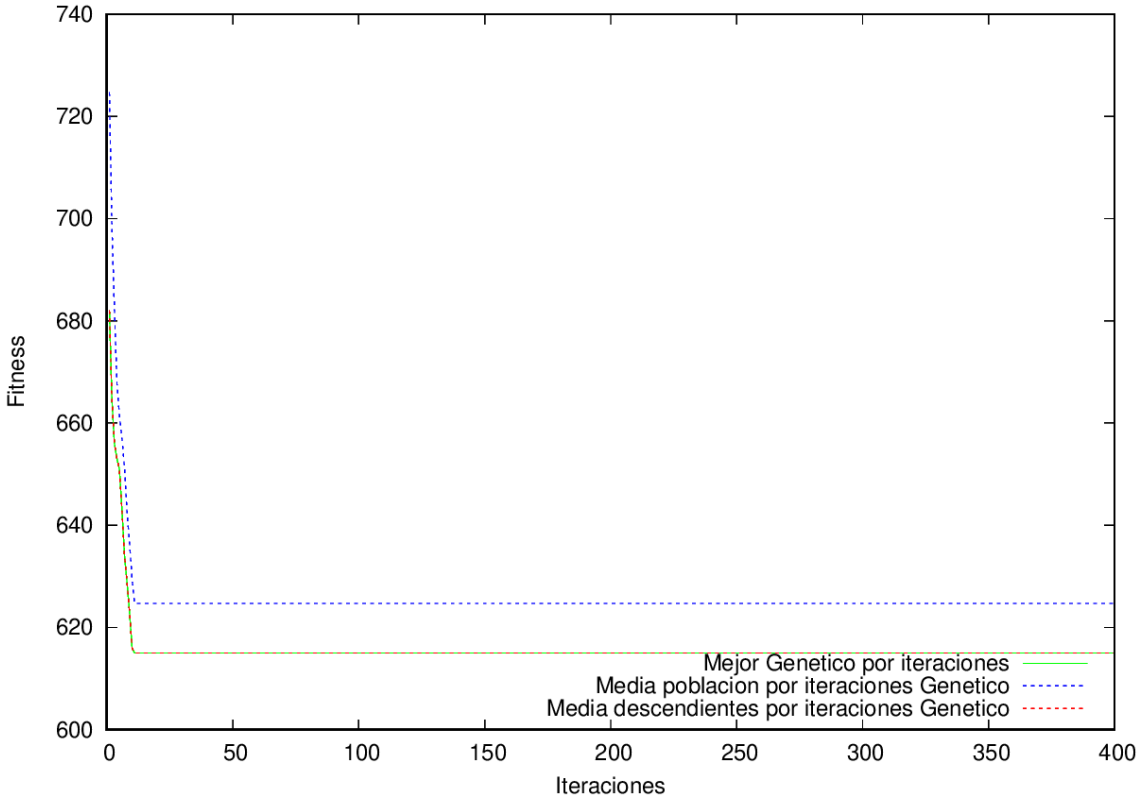

Ilustración 47: Medias Genético IG 100 nodos y 30% densidad con eliminación de 30 nodos

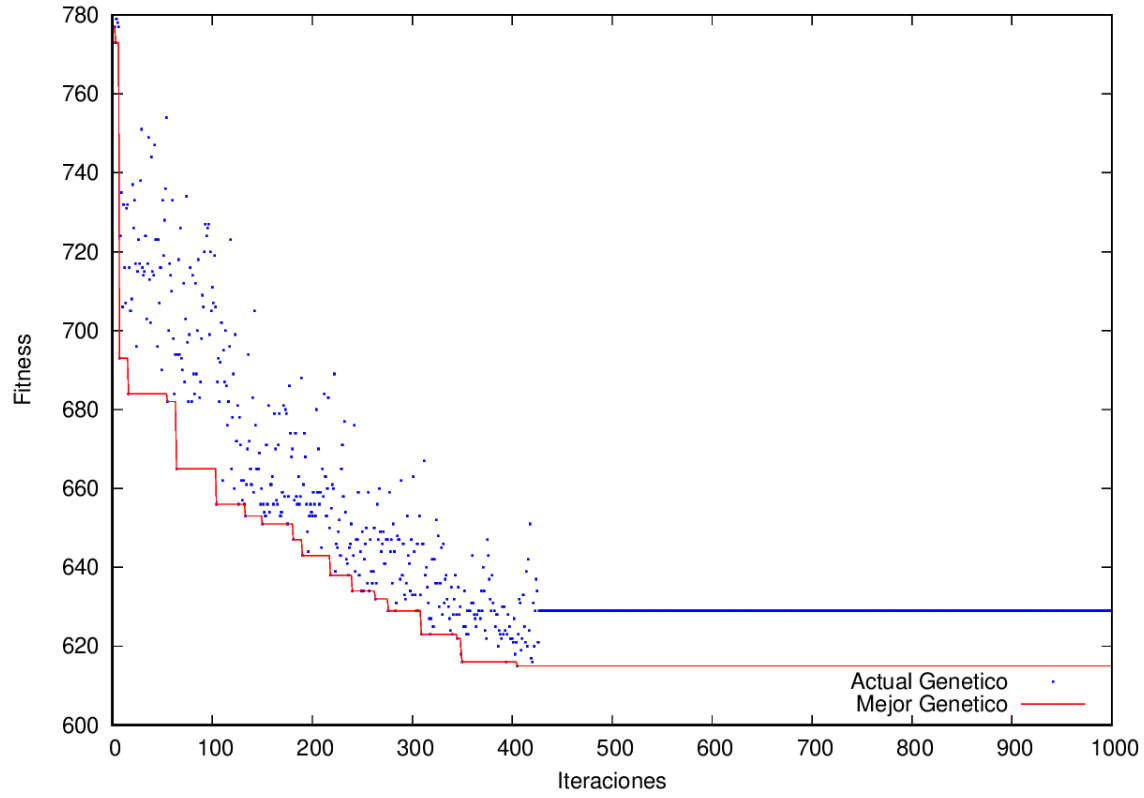

Ilustración 48: GA IG 100 nodos y 30% densidad con eliminación de 30 nodos

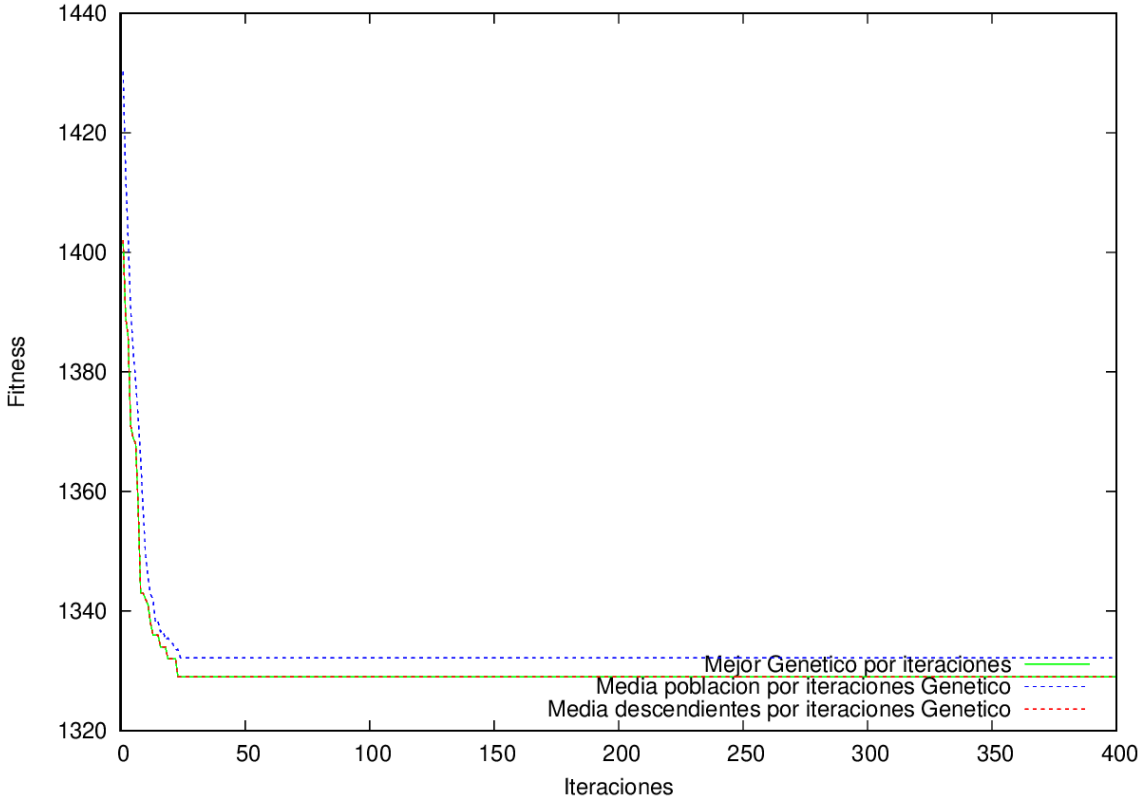

Ilustración 49: Medias Genético IG 100 nodos y 60% densidad con eliminación de 30 nodos

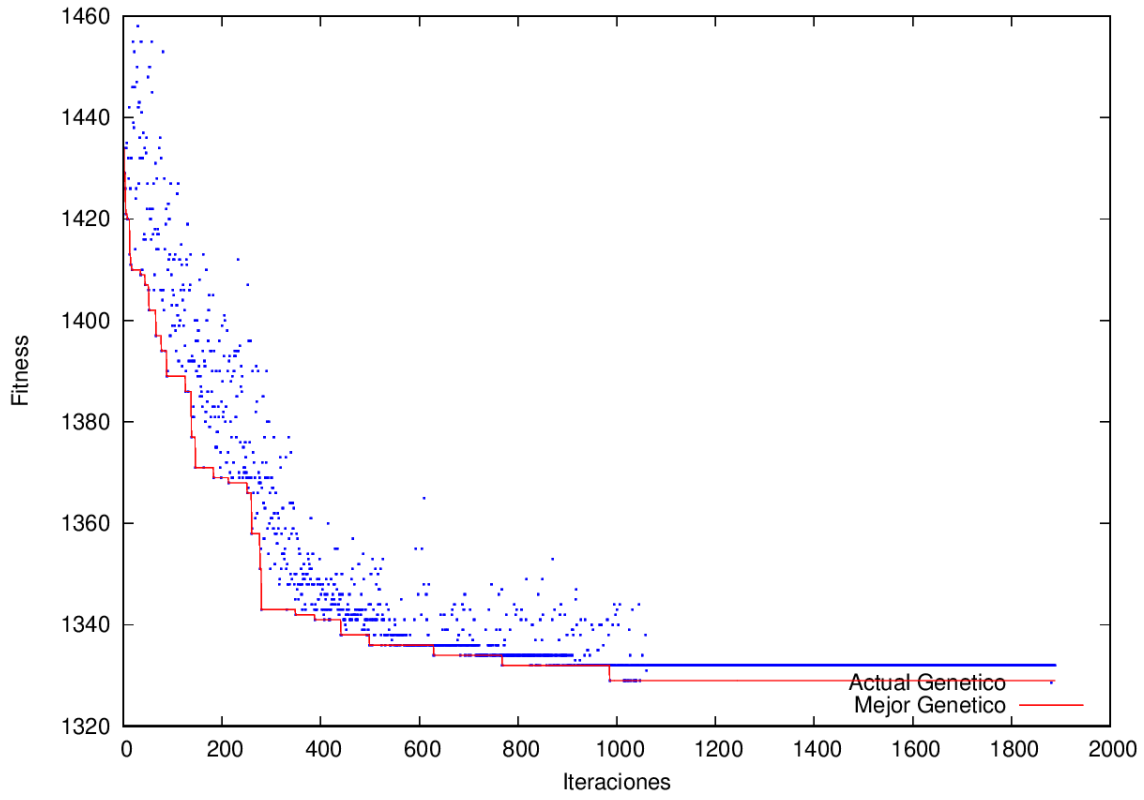

Ilustración 50: GA IG 100 nodos y 60% densidad con eliminación de 30 nodos

Informe del grado de implicación de los miembros del grupo

El desarrollo de esta práctica ha sido dividido, de tal forma que todos realizásemos alguna de las distintas metaheurísticas vistas en prácticas, por tanto, la valoración del grado de implicación igual pueda venir determinada por el grado de complejidad y codificación de estas.

De tal forma, se procederá a indicar el trabajo realizado por cada integrante a la hora de codificación y evaluación de las heurísticas.

- Las clases generales han sido realizadas en su mayor parte por los tres integrantes.
- Las heurísticas escalada simple y máxima pendiente han sido realizadas por Carlos de la Barrera Pérez.
- Las heurísticas búsqueda tabú y enfriamiento simulado fueron realizadas por Alberto Luque Rivas.
- Por último, las heurísticas de iterativo greedy, algoritmo genético y sus variantes han sido realizadas por José Luis Gordillo Relaño.

Tabla de ilustraciones