Cómputo Evolutivo

Tarea 03 : Análisis de Rendimiento

Celeste Lorenzo Guerrero : 316162027
Antonio Sebastián Dromundo Escobedo : 419004327
Diego Dozal Magnani : 316032708

April 10, 2023

Nuevo esquema de representación para el problema de 0-1 Knapsack

Esquema de representación de soluciones

`Implementación : src/knapsack.py -> clase Solution`

Para esta tarea decidimos cambiar ligeramente el esquema de representación, aún son permutaciones sin embargo nos basamos en la representación descrita en [1], por lo que cada solución S es un objeto tipo `Solution` que contiene dos listas : `carried_items` y `non_carried_items`, en la primera se guardan los *items* que considera la solución mientras que en la segunda lista se guarda el restante de *items* que no están en `carried_items` pero sí en el conjunto total de *items*. Este cambio se implementó debido a que facilita la implementación del nuevo operador de vecindad para esta representación.

Descripción de generador de soluciones aleatorias

`Implementación : src/knapsack.py -> generate_random_sol`

El generador de souluciones aleatorias selecciona *items* de conjunto total con una probabilidad de 0.5, los *items* seleccionados se guardan en `Solution.carried_items` mientras que el resto se guarda en `Solution.non_carried_items`.

Métodos de perturbación

Consideramos para las perturbaciones a `S` una instancia de la clase `Solution`.

- **Perturbación aleatoria**

  `Implementación : src/iterative_local_search.py -> random_perturbation`

  Esté método está basado en el descrito en la tarea 3 de la materia. Consideramos la fuerza de perturbación η y el tamaño del ejemplar para determinar el número de items aa intercambiar de forma aleatoria, la cantidad de items a intercambiar es la $\eta * |$`Solution.carried_items`$|$.

- **Perturbación basada en Frecuencias**

`Implementación : src/iterative_local_search.py -> frecuency_perturbation`

El método de perturbación está basado en [1] 3.5 *Frequency-based local optima escaping phase.*
Consideramos un arreglo `F` de frecuencias, sea i el índice del i-ésimo el elemento de `Solution.carried_items`, entonces `F[`i`]` guarda la frecuencia con la que el elemento `Solution.carried_items[`i`]` ha sido perturbado, la frecuencia es un `int` que incrementa en 1 cada vez que el elemento `Solution.carried_items[`i`]` es perturbado. Sea η la **fuerza de perturbación**, la cantidad de elementos a perturbar para la solución `S` es igual a

$$\texttt{number\_of\_perturbed} = \eta * |\texttt{Solution.carried\_items}|$$

Los elementos a perturbar serán los primeros `number_of_perturbed` elementos con menor frecuencia en `F`.
Finalmente el valor recomendado por [1] para $\eta = .4$.

Detalles sobre implementación de ILS

Por motivos de simplicidad, a cada ejemplar considerado para las pruebas le asignamos un número y un nombre que permite identificarlos fácilmente. A continuación se describen

1 - Ejemplar tamaño 45 : `ejeL14n45.txt`

2 - Ejemplar tamaño 200 : `ejeknapPI_3_200_1000_14.txt`

3 - Ejemplar tamaño 1000 a) : `eje1n1000.txt`

4 - Ejemplar tamaño 1000 b) : `eje2n1000.txt`

5 - Ejemplar tamaño 1000 c) : `n_1000_c_1000000_g_6_f_0.3_eps_0.001_s_100`

Es importante resaltar que tanto para `Hill Climbing` como para `Iterated Local Search` el criterio de paro son número de iteraciones. Por lo que el total de iteraciones para el algoritmo completo es $n * m$ con n el número de iteraciones para `Hill Climbing` y m el número de iteraciones para `Iterated Local Search`. Para todas las ejecuciones, consideramos 1000 iteraciones para `ITS` y 500 iteraciones para `Hill Climbing`.

Asímismo, para esta implementación como en la anterior tarea, la función objetivo busca un valor cercano a 0 ya que buscamos reducir la pérdida y con pérdida nos referimos a **Máximo Beneficio Posible - Beneficio de los items que la solución lleva**. De este modo, en los resultados buscamos que el valor encontrado sea el menor posible.

Análisis de Resultados

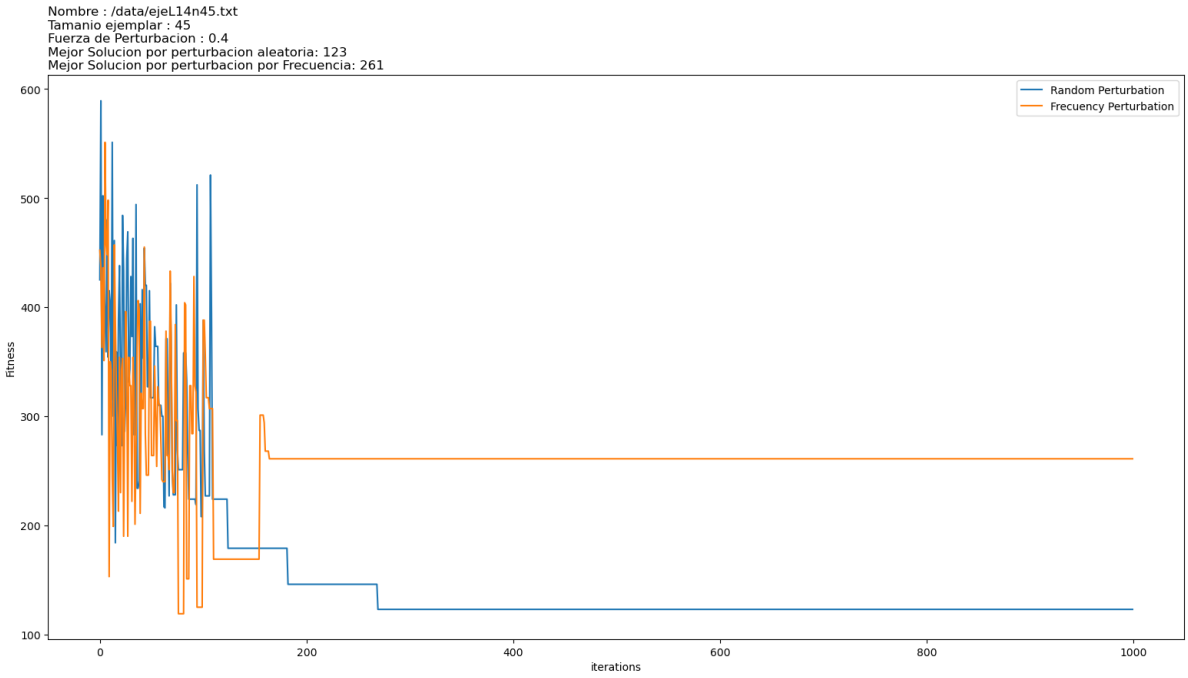
Configuración de parámetros para estrategias de perturbación

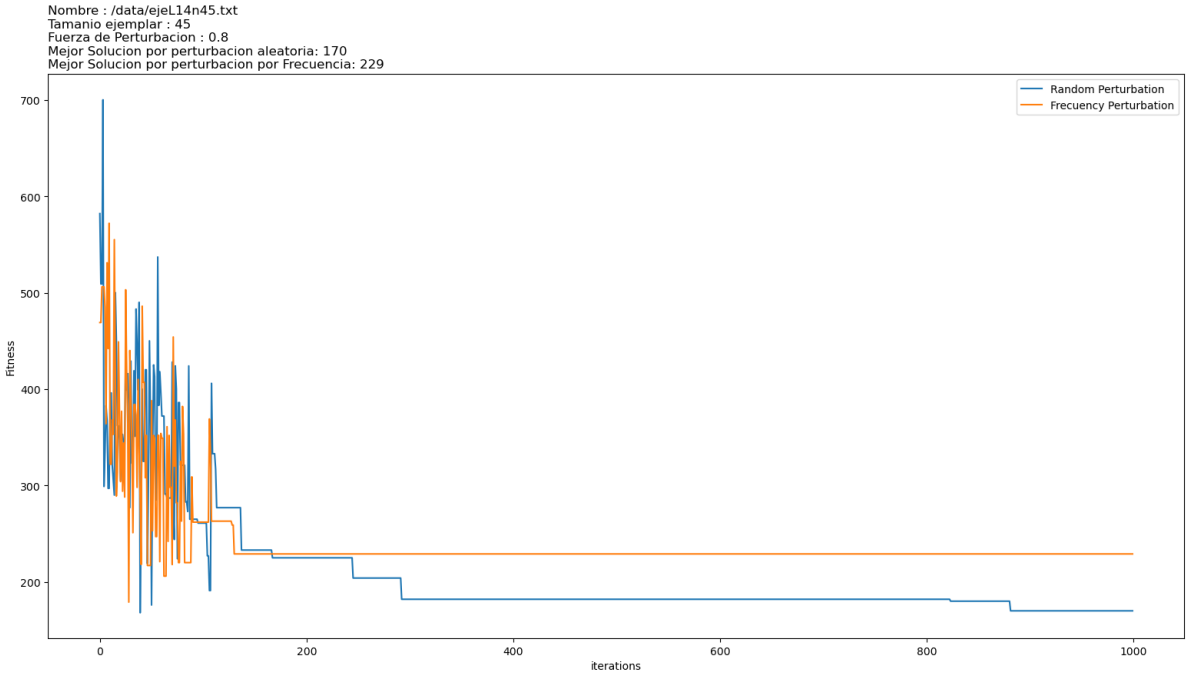
Para la estrategía usando el arreglo de frecuencias el arreglo inicial contiene sólo 0's.

Parámetros utilizados					
estrategia	η : f. perturbación	record	temperatura	iter. hill climbing	iter. ILC
P. Aleatoria	.4	-	20	500	1000
P. Aleatoria	.8	-	20	500	1000
P. de Frecuencias	.4	arr[zeros]	20	500	1000
P. de Frecuencias	.8	arr[zeros]	20	500	1000

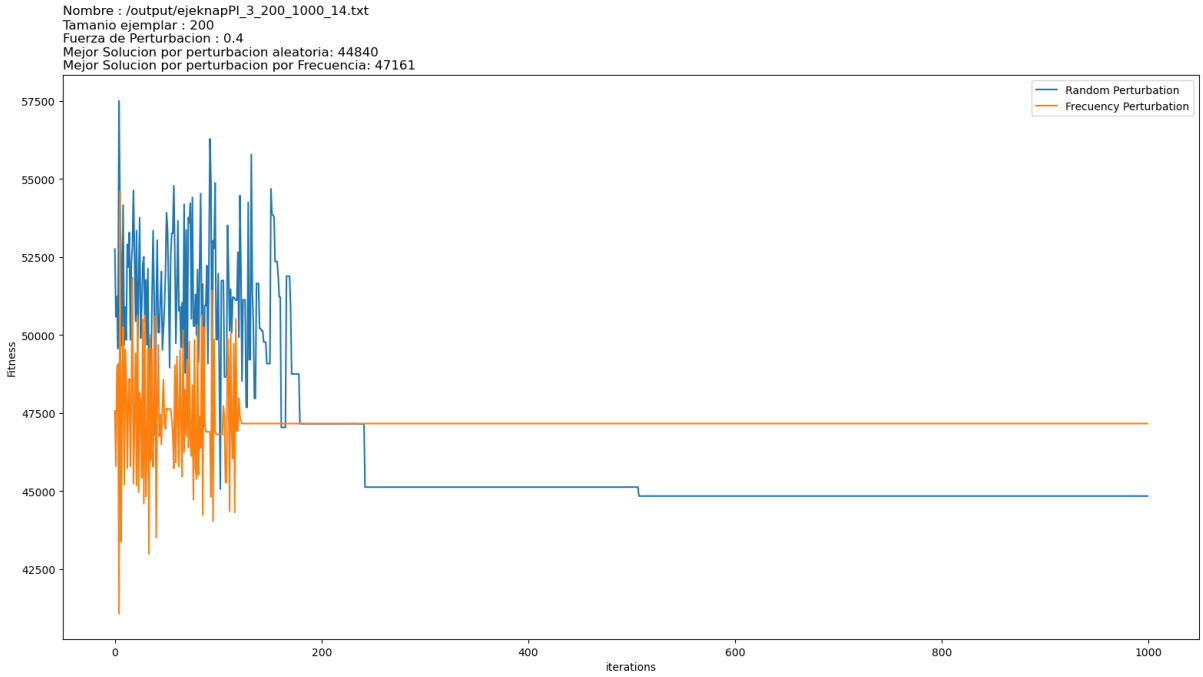
Evolución de las estrategias para ejemplares individuales (sólo una ejecución)

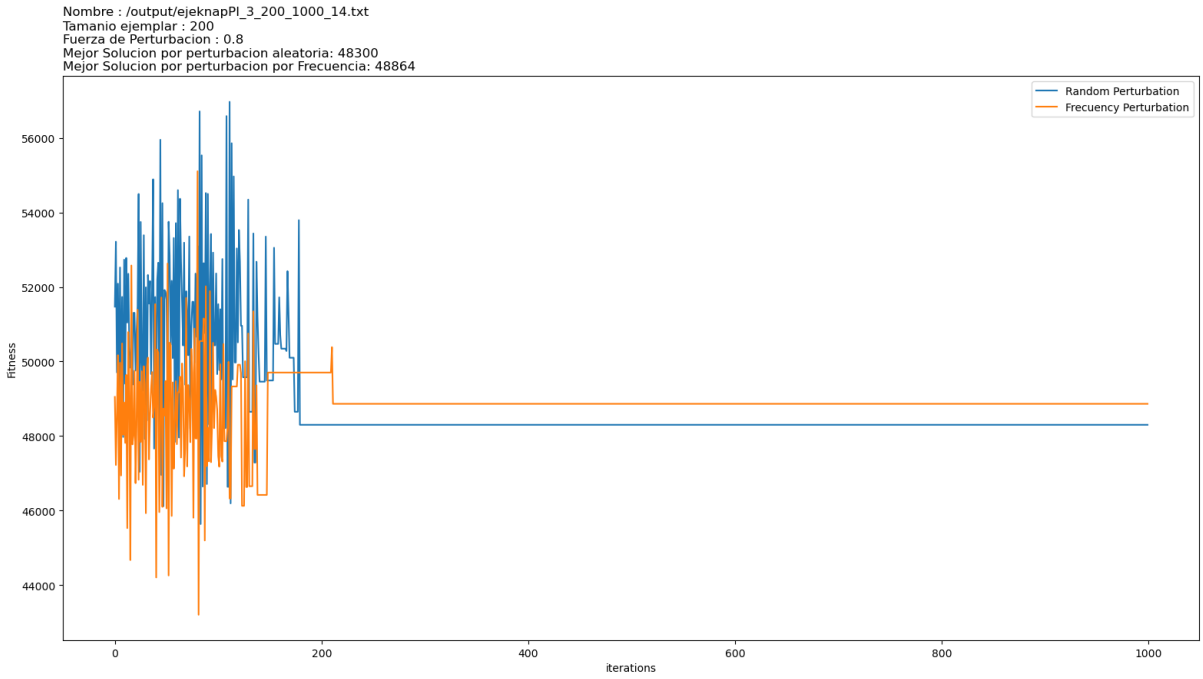
Ejemplar de Tamaño 45




Nombre : /data/ejeL14n45.txt
Tamanio ejemplar : 45
Fuerza de Perturbacion : 0.8
Mejor Solucion por perturbacion aleatoria: 170
Mejor Solucion por perturbacion por Frecuencia: 229
Random Perturbation
Frecuency Perturbation
Fitness
iterations
700
600
500
400
300
200
0
200
400
600
800
1000

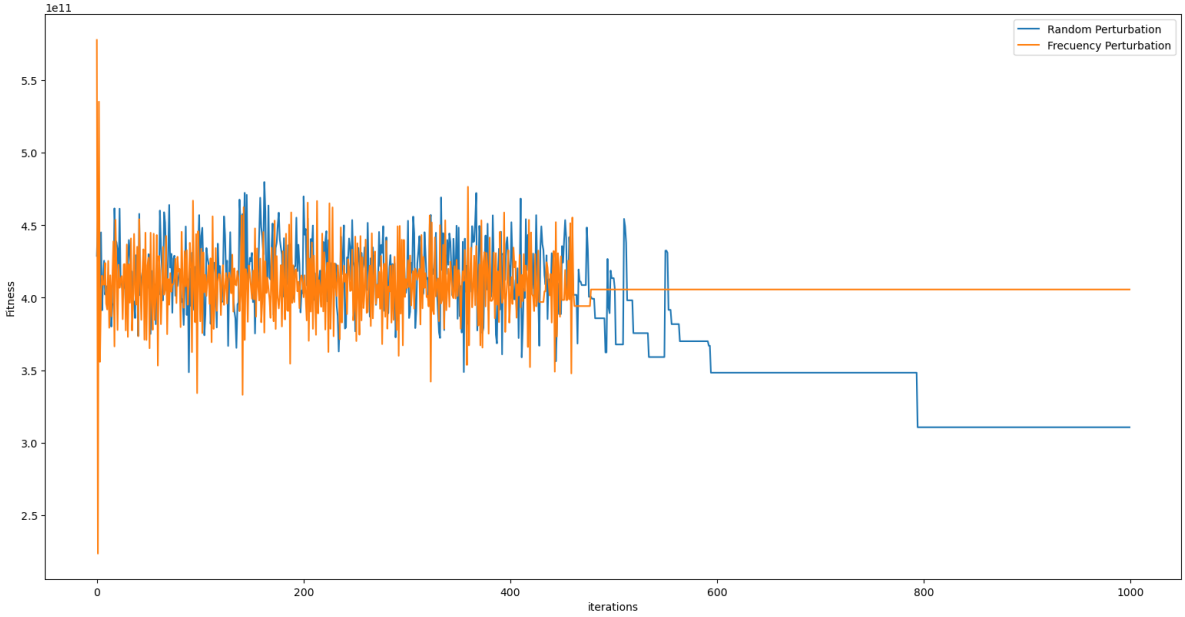
Ejemplar de Tamaño 200



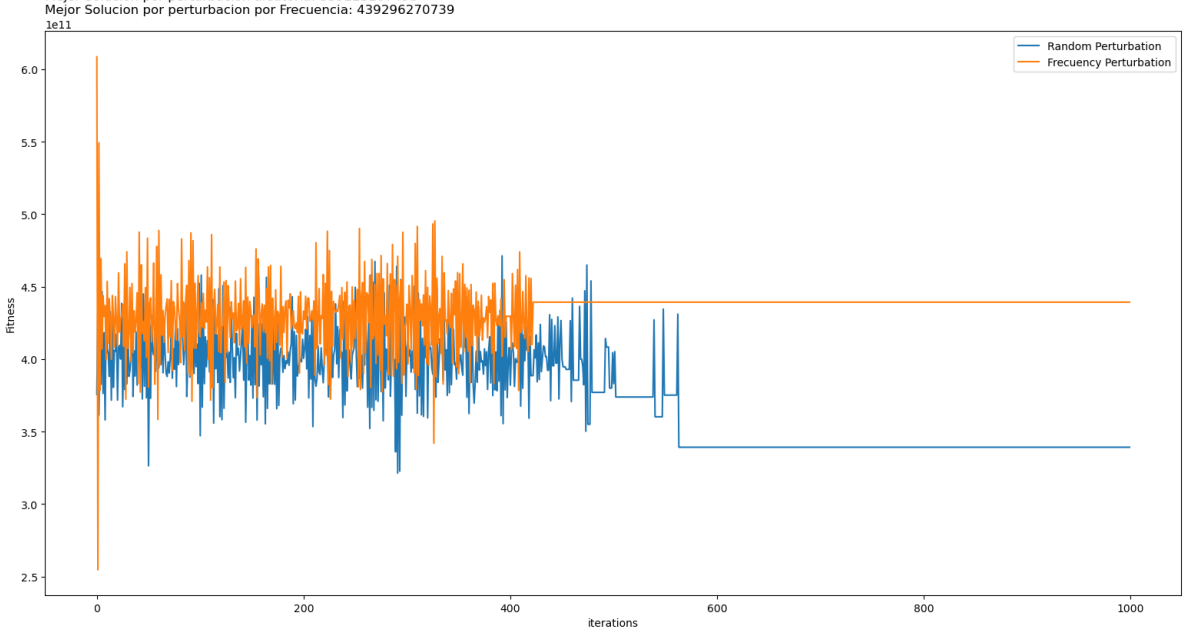


Ejemplar de Tamaño 1000 a)

Nombre : /data/eje1n1000.txt
Tamanio ejemplar : 1000
Fuerza de Perturbacion : 0.4
Mejor Solucion por perturbacion aleatoria: 310620927182
Mejor Solucion por perturbacion por Frecuencia: 405565333716

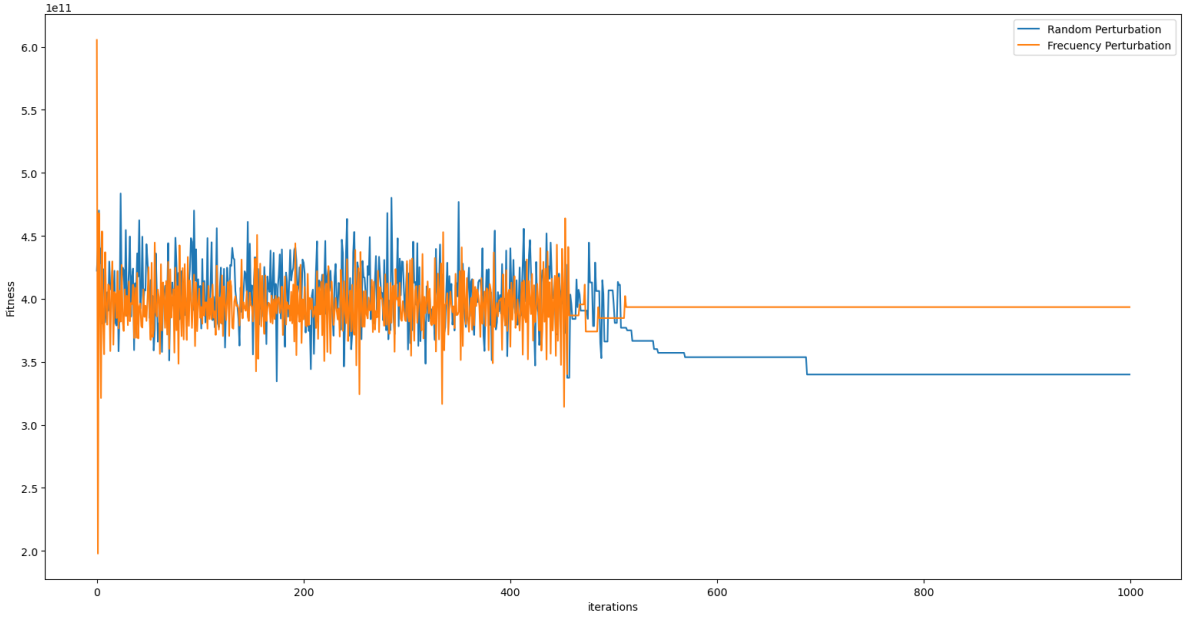


Nombre : /data/eje1n1000.txt
Tamanio ejemplar : 1000
Fuerza de Perturbacion : 0.8
Mejor Solucion por perturbacion aleatoria: 339225207498
Mejor Solucion por perturbacion por Frecuencia: 439296270739

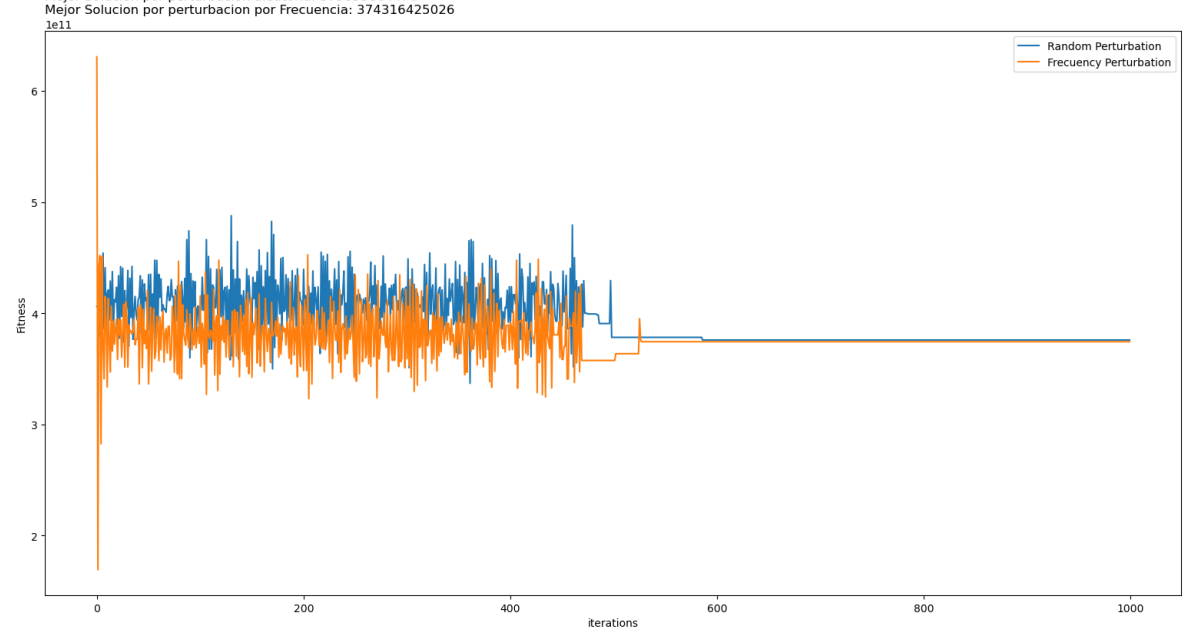


Ejemplar de Tamaño 1000 b)

Nombre : /data/eje2n1000.txt
Tamanio ejemplar : 1000
Fuerza de Perturbacion : 0.4
Mejor Solucion por perturbacion aleatoria: 339921894871
Mejor Solucion por perturbacion por Frecuencia: 393437519522

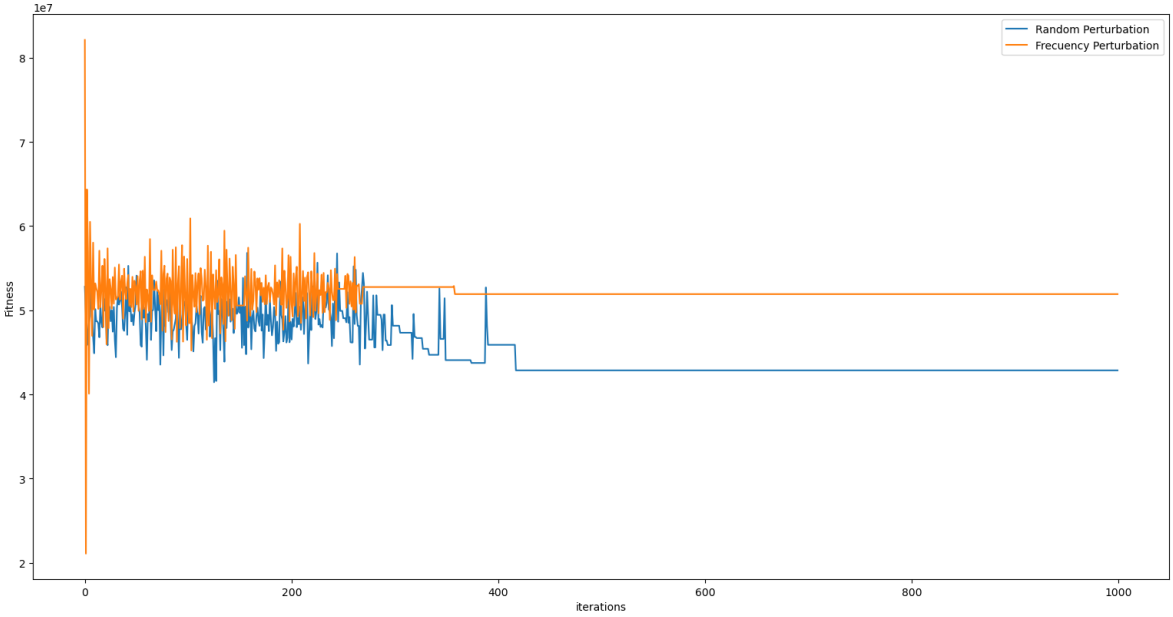


Nombre : /data/eje2n1000.txt
Tamanio ejemplar : 1000
Fuerza de Perturbacion : 0.8
Mejor Solucion por perturbacion aleatoria: 375839864470
Mejor Solucion por perturbacion por Frecuencia: 374316425026

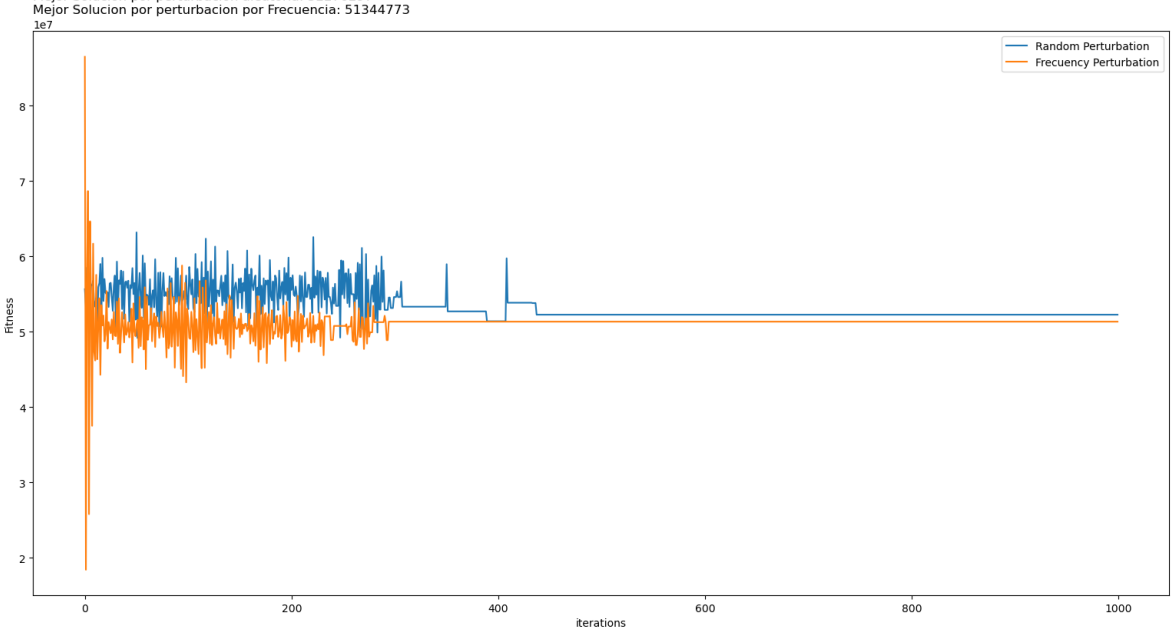


Ejemplar de Tamaño 1000 c)

Nombre : /data/n_1000_c_1000000_g_6_f_0.3_eps_0.001_s_100.txt
Tamanio ejemplar : 1000
Fuerza de Perturbacion : 0.4
Mejor Solucion por perturbacion aleatoria: 42857351
Mejor Solucion por perturbacion por Frecuencia: 51920831



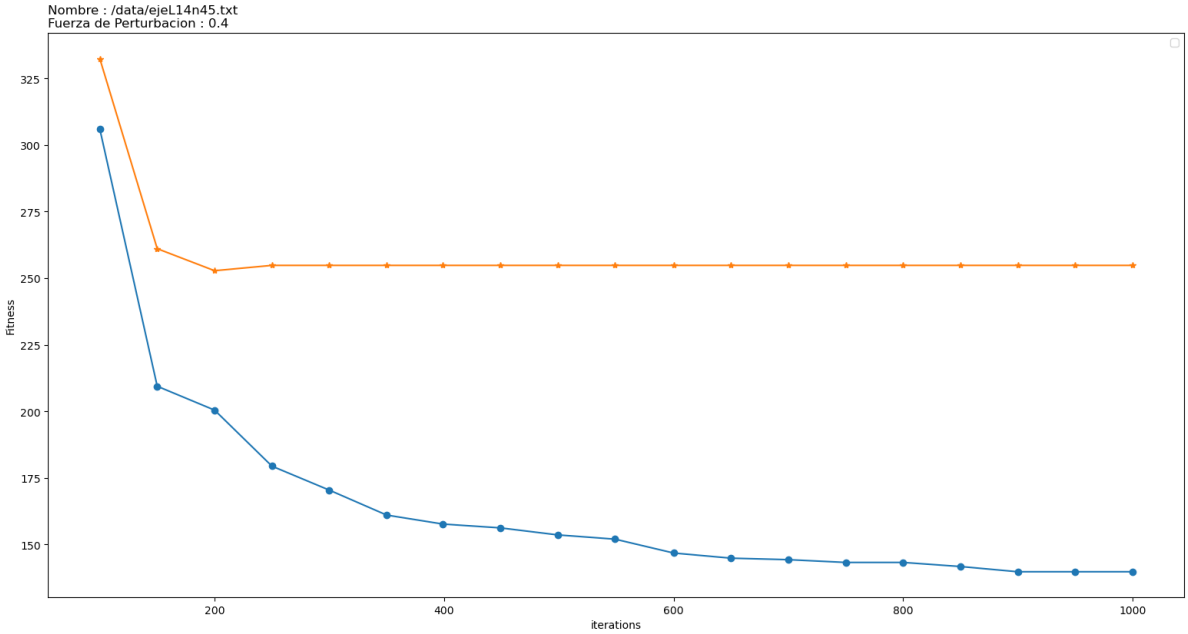
Nombre : /data/n_1000_c_1000000_g_6_f_0.3_eps_0.001_s_100.txt
Tamanio ejemplar : 1000
Fuerza de Perturbacion : 0.8
Mejor Solucion por perturbacion aleatoria: 52270297
Mejor Solucion por perturbacion por Frecuencia: 51344773



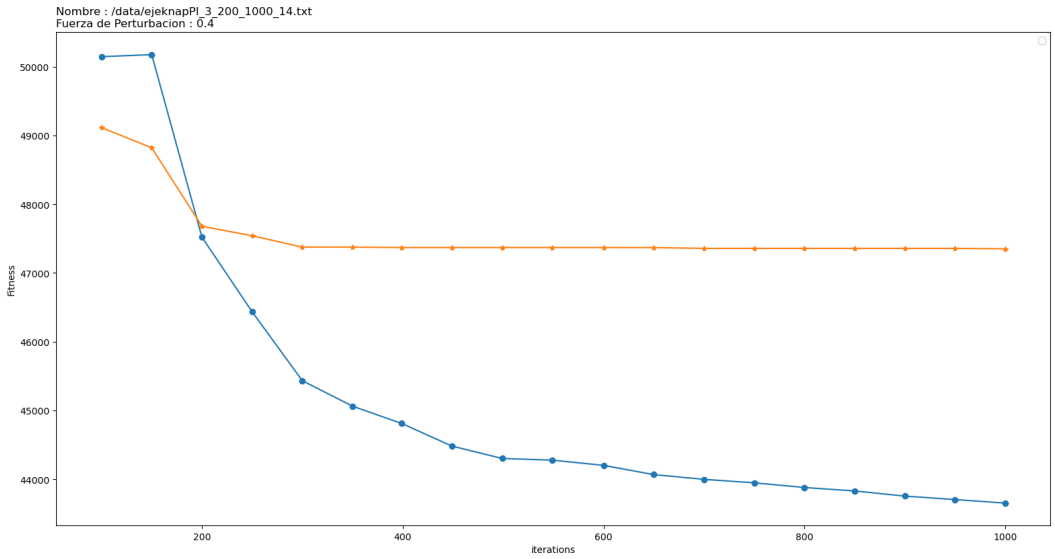
Observaciones
Como podemos notar en las gráficas de las ejecuciones individuales, en casi todos los casos (con excepción en la ejecución del ejemplar de **tamaño 1000 b)**) la perturbación aleatoria (color azul) logró encontrar soluciones con menor valor de función objetivo, mientras que la perturbación por frecuencia en la mayoría de los casos convergió mucho antes que la perturbación aleatoria. Es importante notar que en los ejemplares de tamaño 1000, la perturbación por frecuencia tuvo mayor capacidad de exploración en las primeras iteraciones, sin embargo terminó convergiendo de ingual forma.

Evolución promedio de los ejemplares

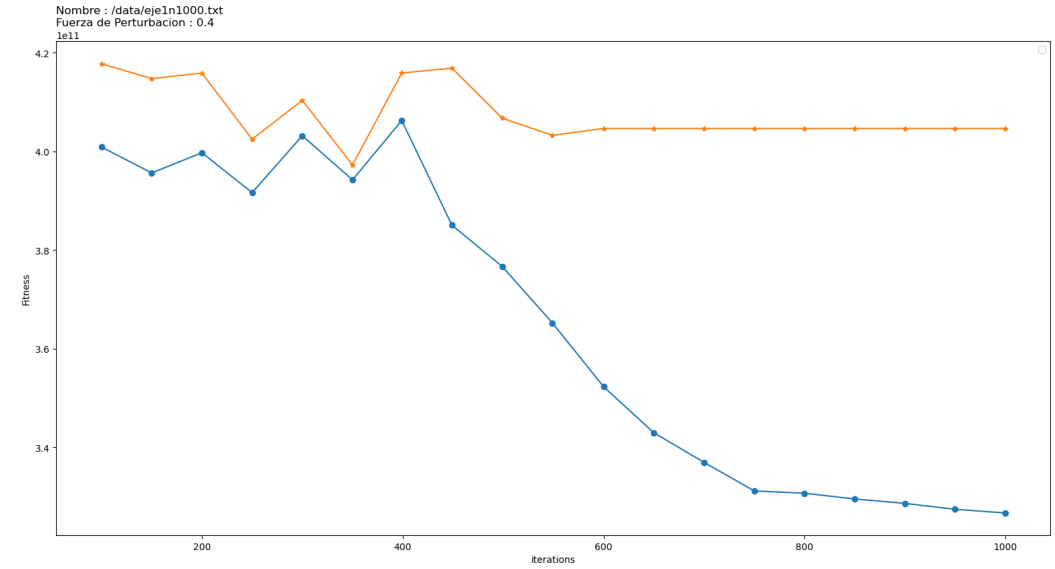
Cada gráfica muestra la evolución del promedio de 10 repeticiones para cada ejemplar. Cada punto está ubicado cada .05*`iteraciones` totales, es decir dividimos las iteraciones totales entre 20 y en cada uno de esos 20 puntos obtuvimos el promedio de las 10 ejecuciones.
`Nota : Se usaron 10 repeticiones ya que con los ejemplares de tamaño 1000 la ejeución con 20 o 30 repeticiones tardaba demasiado, adicionalmente sólo se realizaron ejecuciones con` **fuerza de perturbación** η **=.4** .
Ejemplar de Tamaño 45



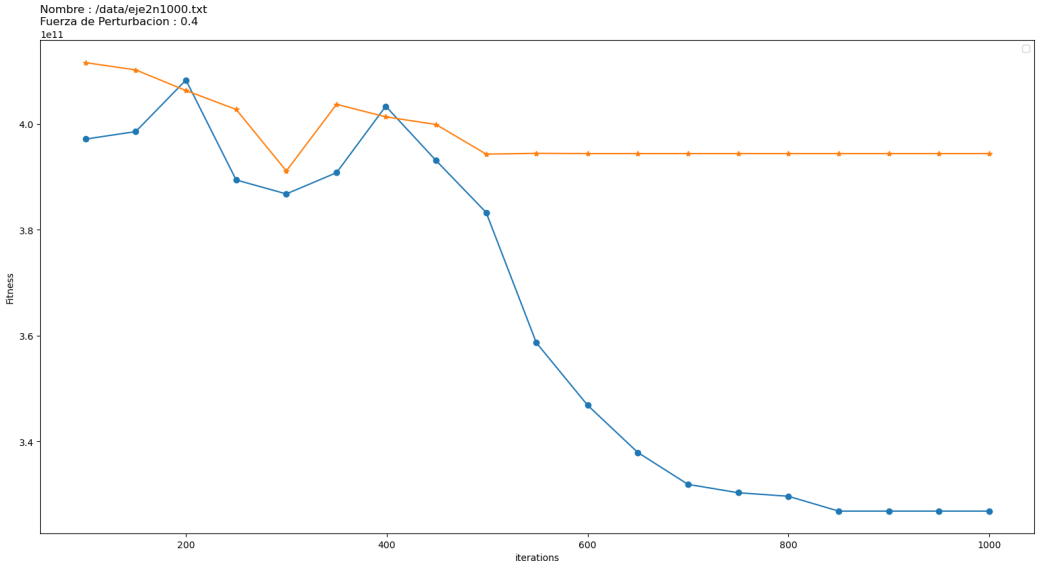
Ejemplar de Tamaño 200



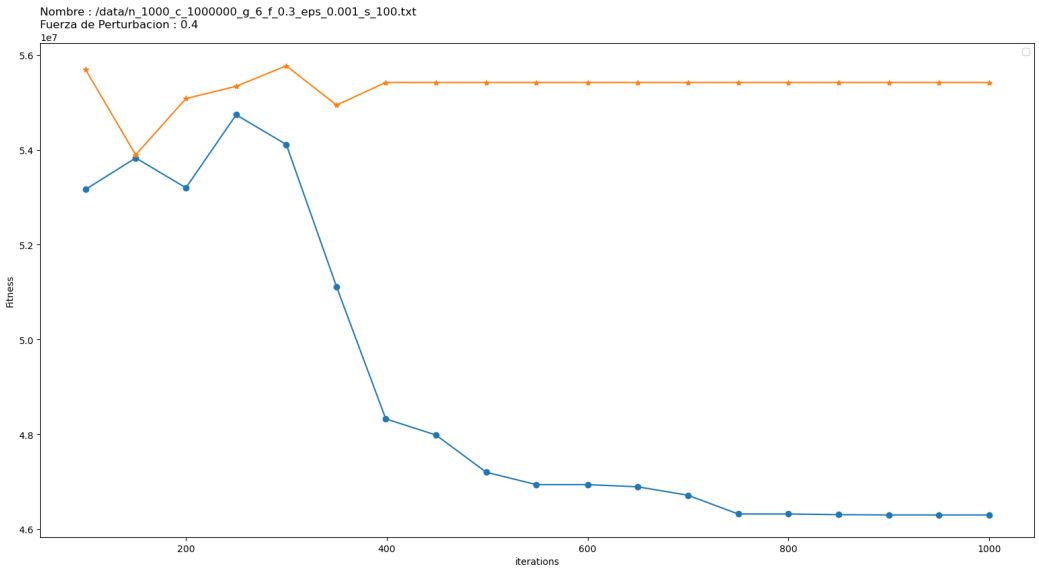
Ejemplar de Tamaño 1000 a)



Ejemplar de Tamaño 1000 b)



Ejemplar de Tamaño 1000 c)



Observaciones

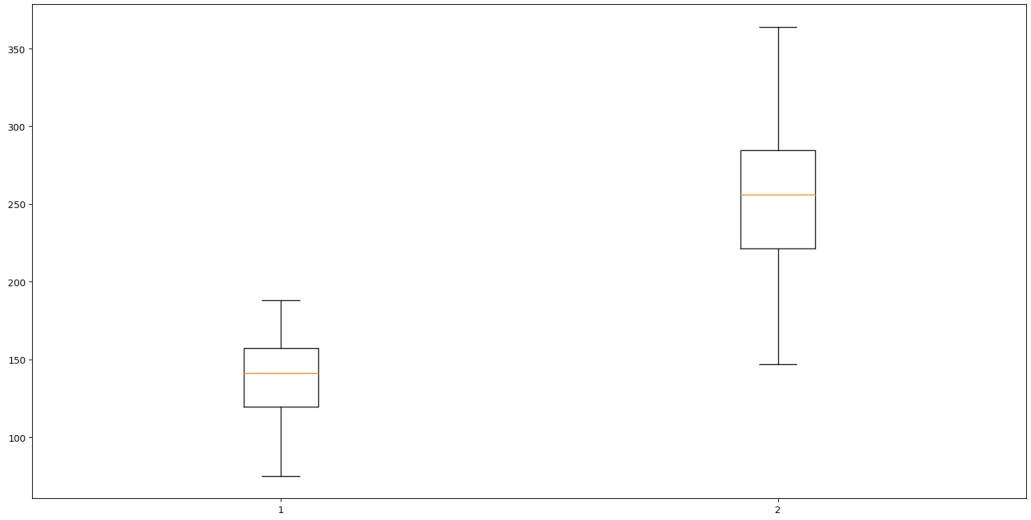
En las gráficas de evolución promedio es mucho más claro que la perturbación por frecuencia llega a converger en las primeras ejecuciones mientras que la perturbación aleatoria llega a muy buenos resultados en todas las ejecu-

ciones. Es importante resaltar que la perturbación por frecuencias converge a un m'ínimo local en las primeras ejecuciones, tratándose de la evolución promedio podemos darnos una idea que el algoritmo por frecuencia no tiene buen rendimiento.

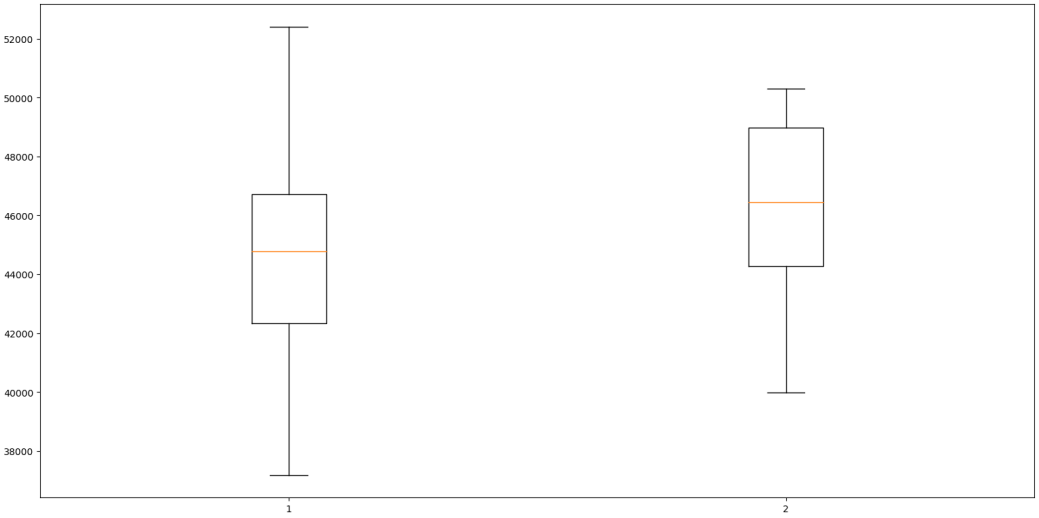
Gráficas Boxplot

Por practicidad en la lectura de las gráficas, para mostrar los resultados del algoritmo aleatorio (**Random Perturbation**) le asignamos el número 1 y para el algoritmo basado en frecuencias (**Frecuency Perturbations**) le asginamos el número 2.

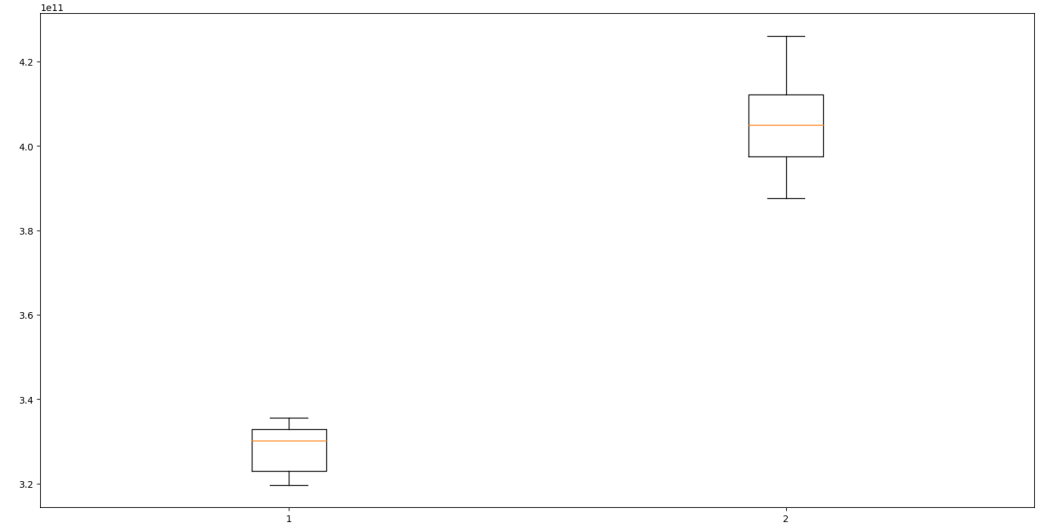
Ejemplar de Tamaño 45



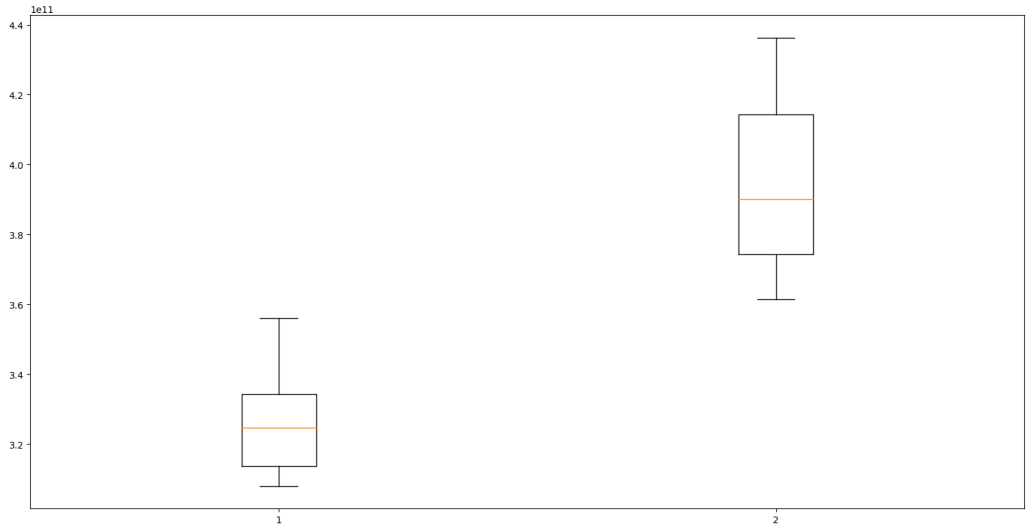
Ejemplar de Tamaño 200



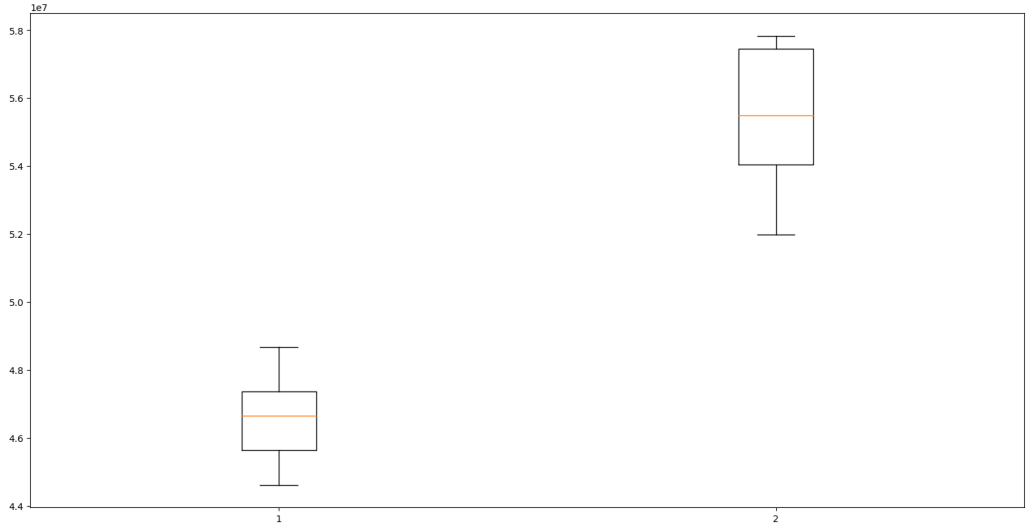
Ejemplar de Tamaño 1000 a)



Ejemplar de Tamaño 1000 b)



Ejemplar de Tamaño 1000 c)



Observaciones : La mayoría de las gráficas boxplot muestran datos acorde con la evolución promedio y con los datos estadísticos (revisados en la siguiente sección) ; el algortimo 1 : **Random Perturbation** logra encontrar un mínimo muy cercano a 0 e incluso el valor promedio encontrado es bajo, mientras que los valores encontrados por la

perturbación basada en frecuencias son peores y el valor promedio no llega a ser muy bueno. Un caso interesante es el **ejemplar de tamaño 200**, en donde a pesar de que el algoritmo 1 muestra encontrar un mejor mínimo, los valores promedio de ambos algoritmos no están muy distantes entre sí.
Podemos conjeturar que los datos muestran que el algoritmo 2 efectivamente converge rápidamente a mínimos locales.

Datos estadísticos

`R: Random Peturbation, F : Frecuency Perturbation`

Datos estadísticos				
Ejemplar-Estrategia	**Bes(min)**	**Worst(max)**	**Mean**	**Std**
`0-R`	75	188	138	25
`0-F`	147	364	254	43
`1-R`	37175	54615	45342	5153
`1-F`	39982	48977	47248	4954
`2-R`	3.0408e+11	3.3562e+11	3.2666e+11	1.47e+09
`2-F`	3.8759e+11	4.2606e+11	4.0466e+11	1.1561e+10
`3-R`	3.0798e+11	3.5595e+11	3.2625e+11	1.5480e+09
`3-F`	3.6138e+11	4.3630e+11	3.9437e+11	2.5607e+10
`4-R`	4.2765e+07	4.8673e+07	4.6298e+07	1.6881e+06
`4-F`	5.1976e+11	5.7827e+07	5.5428e+07	2.1208e+06

Observaciones :
En general, la estrategia que encuentra el menor valor es la perturbación aleatoria, si notamos la diferencia entre los mejores encontrados entre ambas estrategias, es notorio que la perturbación por frecuencias empeora respecto a la aleatoria, en los primeros ejemplares incluso es el doble de peor. Con los ejemplares que tienen 1000 elementos podemos notar que, aunque la perturbación por frecuencia empeora, la diferencia no es tan evidente como en los primeros 2 ejemplares. En base a los datos estadísticos, podemos inferir que la mejor estrategia continúa siendo la aleatoria.

Resultados pruebas de hipótesis

Para cada estrategia se utilizó el mismo parámetro de fuerza de perturbación $\eta = 0.4$ y se ejecutaron 20 repeticiones de las pruebas de hipótesis.

Ejemplar	**Random Perturbation**	**Frecuency Perturbation**
0	↑: 100.0/ ↓: 0.0	↑: 0.0/ ↓: 100.0
1	↑: 0.0/ ↓: 100.0	↑: 100.0/ ↓: 0.0
2	↑: 100.0/ ↓: 0.0	↑: 0.0/ ↓: 100.0
3	↑: 100.0/ ↓: 0.0	↑: 0.0/ ↓: 100.0
4	↑: 100.0/ ↓: 0.0	↑: 0.0/ ↓: 100.0

Conclusiones y comentarios finales
Como se puede observar, la diferencia entre la veces que el algoritmo aleatorio fue mejor que el algoritmo de frecuencias es evidente. En la mayoría de los ejemplares, el algoritmo aleatorio fue mejor el 100 % de las veces sin embargo, para el ejemplar 1 ocurrió justo lo contrario, la razón de este resultado aún no queda clara pero podemos suponer que la muestra generada en las 10 repeticiones (mencionada en la sección de Evolución Promedio) se encuentra sesgada de alguna forma.
Finalmente, considerando cada sección del Análisis de Resultados, podemos afirmar que la Perturbación Aleatoria tiene mejores resultados para encontrar el mínimo global para el algoritmo de Búsqueda Local Iterada. Adicionalmente recalcamos que la Perturbación basada en Frecuencias converge rápidamente a mínimos locales, sin embargo en las gráficas de ejecuciones individuales y evolución promedio podemos ver que en ejecuciones previas a converger, el algoritmo encuentra buenos óptimos por momentos incluso mejores que la perturbación aleatoria pero desafortunadamente termina convergiendo,la razón de esta convergencia aún no queda clara para el equipo.

References

[1] Jin-Kao Hao Zequm Wei. Iterated two-phase local search for the set-union knapsack problem. *ELSEVIER*, 145:13, 2019.