UNIVERSIDAD DE CÓRDOBA
ESCUELA POLITÉCNICA SUPERIOR

METAHEURÍSTICAS

CODIFICACIÓN Y EVALUACIÓN DE SOLUCIONES
Práctica 1

GRUPO A2C2

Álvaro José Camino González
Carlos Carmona Muñoz
Antonio Manuel Gómez Orellana
Carlos Gómez Pino

ÍNDICE

1. Problema del TSP (Viajante de comercio)

1.1 Planteamiento

El problema del ***TSP*** (Viajante de comercio) consiste en encontrar el **recorrido más corto** que une un conjunto de ***N*** nodos, de forma que ningún nodo se visite dos veces y el recorrido termine en el nodo de partida.

En la versión del problema que vamos a analizar, todos los nodos **están conectados entre sí** y el grafo que representa la unión de los nodos es **no dirigido**.

Para representar dicho problema, podemos hacer uso de **permutaciones de los nodos**, representados por un vector, en el que cada elemento contiene un entero con la posición del siguiente nodo a visitar.

4	*3*	*1*	*5*	*2*

Expresado formalmente el objetivo del problema consiste en **minimizar** la siguiente expresión:

$$f(x) = d_{x_n x_1} + \sum_{i=1}^{N-1} d_{x_i x_{i+1}}$$

En la que $d_{xi\ xj}$ representa la **distancia euclídea** entre los nodos i y j.

1.2 Descripción de las pruebas

El objetivo de esta práctica es **codificar soluciones** generadas de forma aleatoria y evaluarlas.

Para el caso del ***TSP*** disponemos de tres instancias del problema:

- *berlin52.tsp*
- *ch150.tsp*
- *d2103.tsp*

Realizaremos pruebas utilizando **tres semillas** diferentes para generar soluciones y para cada instancia realizaremos pruebas con 1.000, 1.500, 3.000, 5.000, 10.000 y 30.000 iteraciones para encontrar la **solución óptima**. La idea de realizar más iteraciones de las estipuladas (1000) es la de analizar si este incremento influye en la obtención de una mejor solución.

El siguiente esquema ilustra mejor la metodología de las pruebas:

➔ Para cada instancia :
  → Para cada semilla :
    - Generar 1.000 soluciones
    - Generar 1.500 soluciones
    - Generar 3.000 soluciones
    - Generar 5.000 soluciones
    - Generar 10.000 soluciones
    - Generar 30.000 soluciones

Para la realización de las pruebas hemos procedido, en primer lugar, a la codificación de las distintas clases y métodos indicados en el enunciado de la práctica en el lenguaje **C++**. Una vez comprobado el correcto funcionamiento del programa hemos pasado a la realización de dichas pruebas.

1.3 Análisis de los resultados

En las siguientes tablas se resumen los resultados de las pruebas, mostrando el valor de aptitud (**fitness**) de la mejor solución encontrada:

	Iteraciones realizadas					
	1.000	1.500	3.000	5.000	10.000	30.000
berlin52.tsp	25.319,29	23.801,90	23.409,58	23.409,58	23.409,58	23.240,98
ch150.tsp	47.766,80	47.766,80	47.170,18	47.170,18	47.170,18	45.685,39
d2103.tsp	3.148.144,70	3.148.144,70	3.148.144,70	3.140.841,12	3.138.980,92	3.120.175,67

Tabla 1: Resultados obtenidos utilizando la semilla por defecto (TSP)

	Iteraciones realizadas					
	1.000	1.500	3.000	5.000	10.000	30.000
berlin52.tsp	23.892,89	23.892,89	23.892,89	23.892,89	23.465,21	21.831,66
ch150.tsp	48.671,46	48.614,99	43.634,36	43.634,36	43.634,36	43.634,36
d2103.tsp	3.141.981,44	3.141.981,44	3.141.981,44	3.141.981,44	3.129.579,28	3.125.675,35

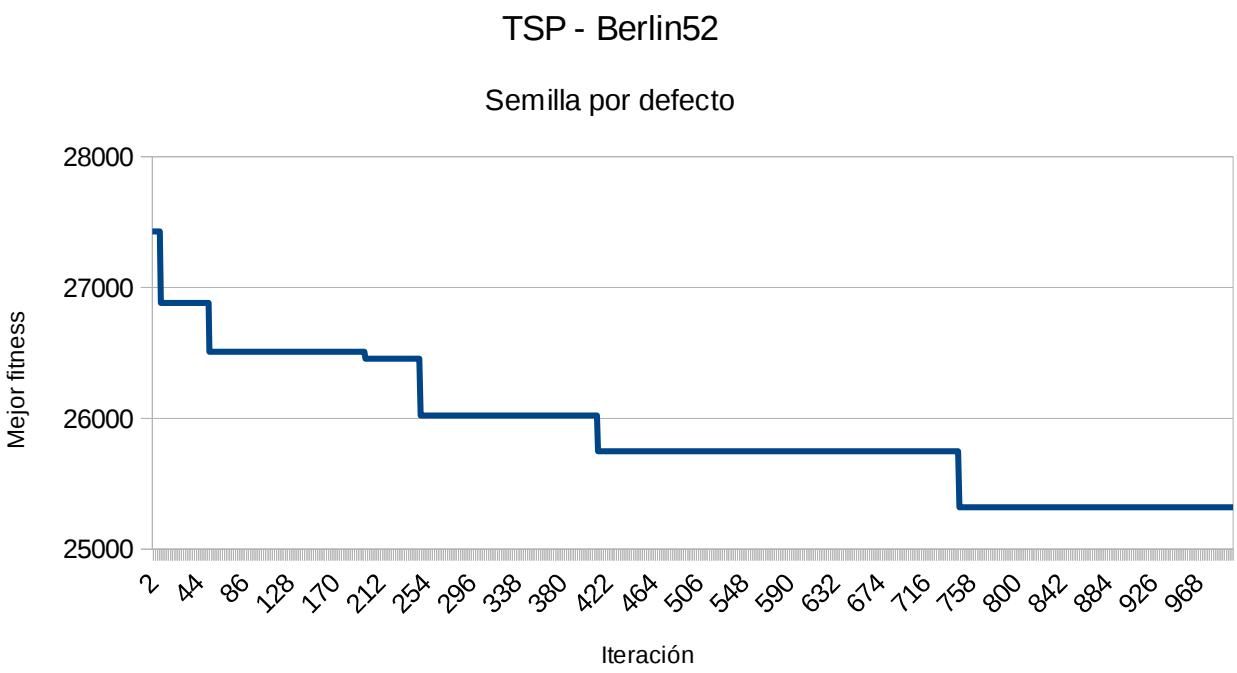
Tabla 2: Resultados obtenidos utilizando la semilla de índice 1 (TSP)

	Iteraciones realizadas					
	1.000	1.500	3.000	5.000	10.000	30.000
berlin52.tsp	24.854,74	24.854,74	23.818,89	23.818,89	23.370,65	23.264,51
ch150.tsp	47.928,26	47.928,26	47.734,43	47.734,43	47.188,66	45.646,46
d2103.tsp	3.155.551,01	3.155.551,01	3.127.495,18	3.127.495,18	3.124.381,94	3.124.381,94

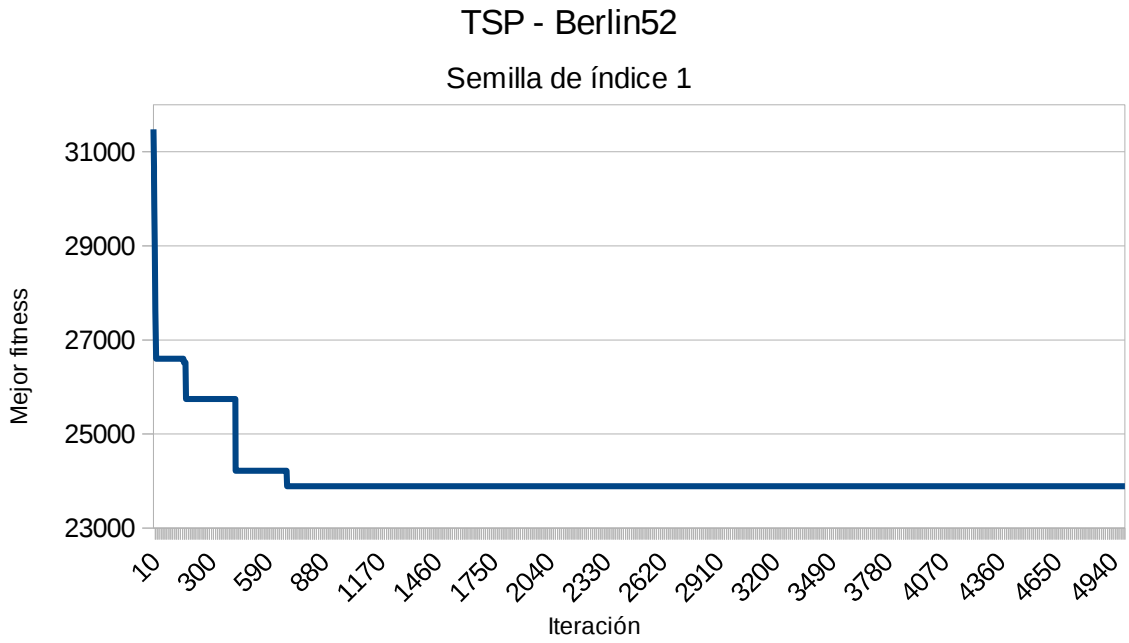
Tabla 3: Resultados obtenidos utilizando la semilla de índice 20 (TSP)

Un primer análisis de los resultados de las tablas parece indicar que cuantos más intentos realizados para obtener solución, más probablemente se encontrará esa **solución óptima**.

A continuación se mostrarán algunas gráficas que ayudarán a ilustrar el comportamiento de la **búsqueda aleatoria** de soluciones:

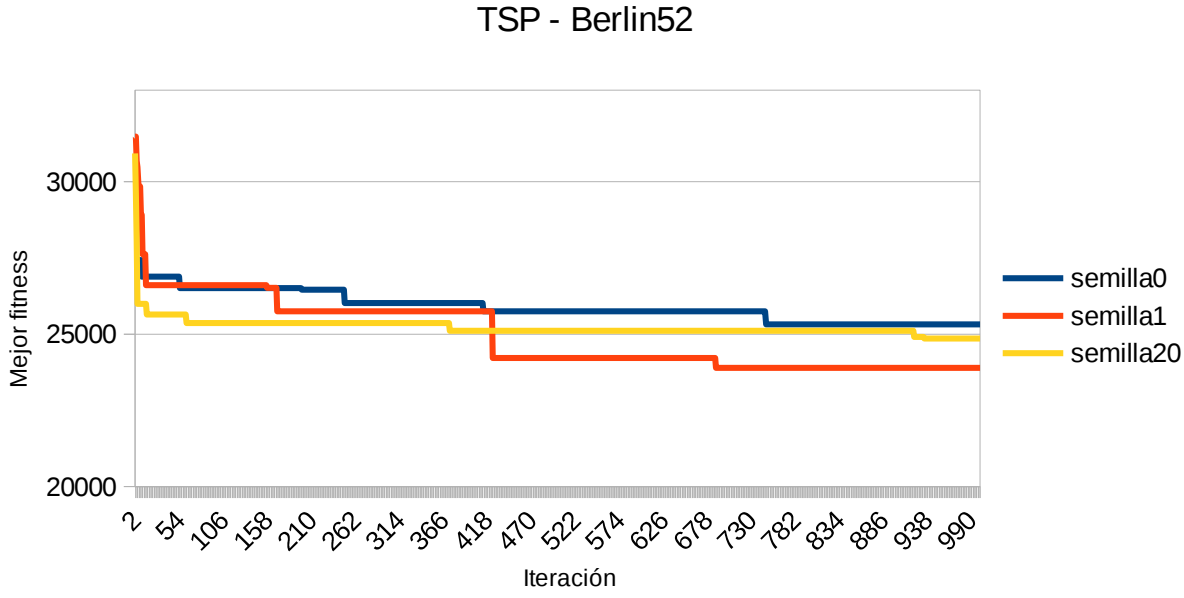


En esta gráfica se aprecia el comportamiento típico de la búsqueda aleatoria. Encontramos una solución con un valor de aptitud (**fitness**) ***f*** y en posteriores iteraciones intentamos encontrar un mejor valor del parámetro. Los escalones que vemos en el gráfico se deben a que durante esos intentos no se encuentran soluciones con **mejor aptitud**.



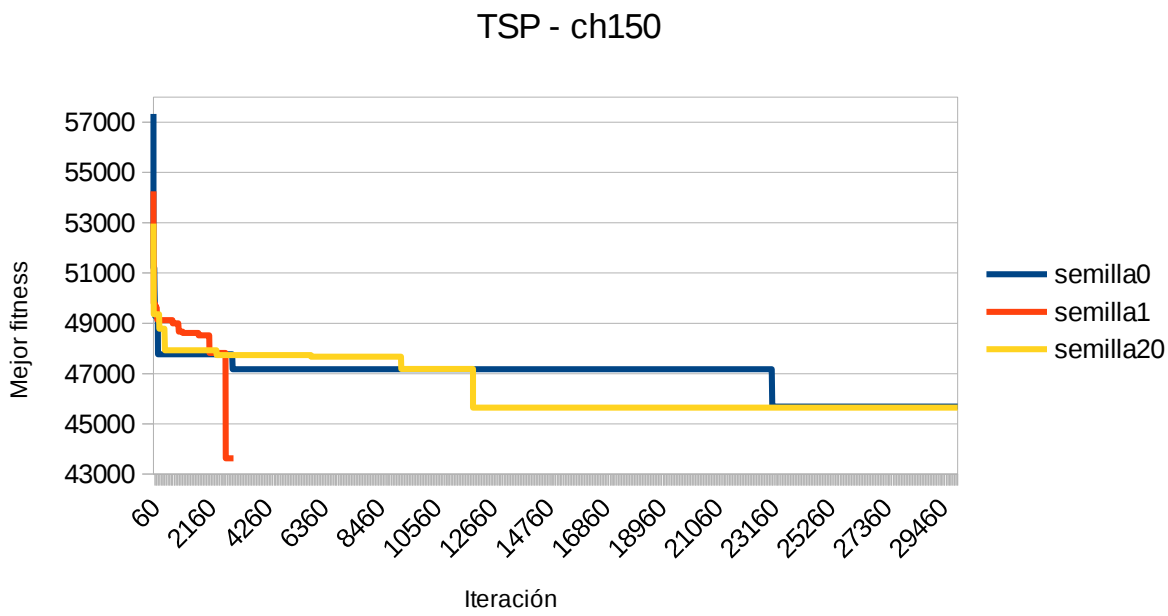
En esta otra vemos una **gran estabilidad** durante la mayor parte del proceso de búsqueda. Esto se debe a que en las primeras iteraciones ya se ha encontrado una solución "buena" difícil de mejorar.

Otro dato relevante acerca de la búsqueda aleatoria se encuentra en el hecho de generar secuencias de soluciones a partir de una semilla. Según ésta, la probabilidad de encontrar antes (en menos intentos) una solución mejor que otra búsqueda que parta de otra semilla es diferente. Veamos algunas comparaciones:



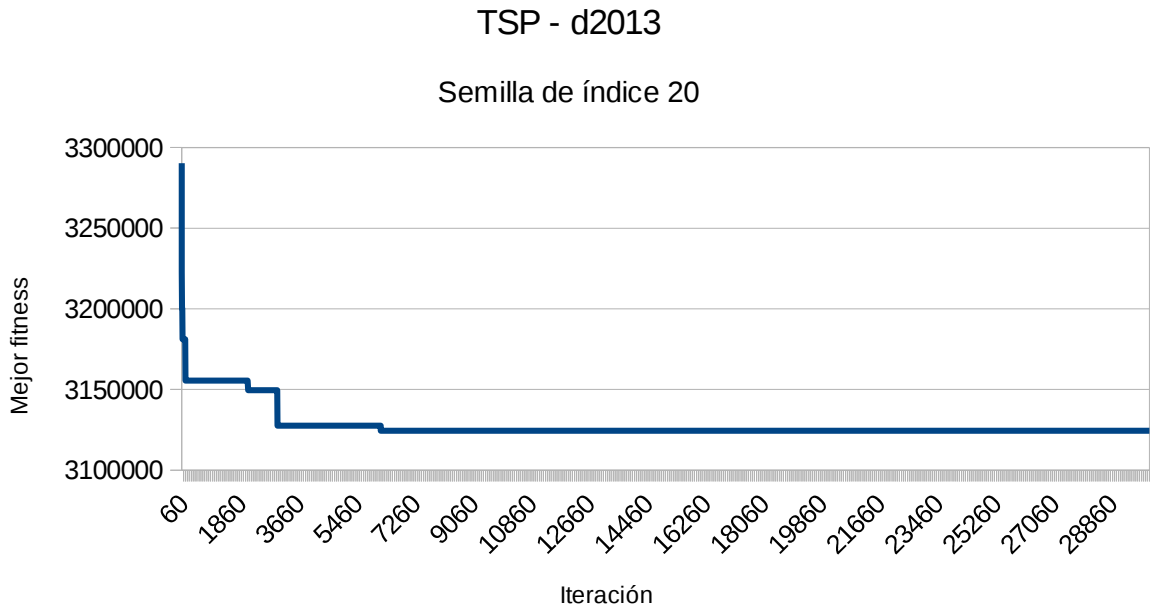
Se pueden observar algunas fluctuaciones al obtener la mejor solución, sobre todo al principio del proceso, y también se observa, como aparecía en las tablas resumen del principio de este apartado, que la mejor solución encontrada es **diferente para cada semilla**.

Esta idea de "suerte" se ve mejor en la gráfica siguiente, ya en la instancia *ch150*:



Comparamos tres ejecuciones con **semillas diferentes** (el número que acompaña a la palabra semilla en la leyenda del gráfico indica el índice de la semilla utilizada, siendo 0 el valor por defecto). La semilla de índice 1 ha encontrado en menos intentos (3000, concretamente) una solución significativamente mejor que las otras dos en 30000 intentos.

Para acabar el análisis, veamos algunos datos sobre la última instancia, *d2103*. Esta es la instancia del TSP con más nodos, por lo que es razonable pensar que la posibilidad de encontrar una buena solución es remota.



En el gráfico observamos que hay un tramo estable muy extenso. Se podría pensar que se debe a que se ha encontrado una solución óptima o muy cercana a la óptima, sin embargo, aunque no conozcamos el ***fitness*** de la mejor solución sabemos que ésta encontrada con la semilla de índice 20 no es la mejor, ya que las otras ejecuciones con semillas diferentes arrojan mejores valores de aptitud. Entonces ¿qué ocurre? La razón de estos tramos estables es que al aumentar el número de caminos posibles (el número de soluciones diferentes) disminuye la probabilidad de mejorar un camino ya encontrado.

Como conclusión general sobre la búsqueda aleatoria podemos decir que la probabilidad de encontrar la **solución óptima** es muy baja en los casos con un número de variables no demasiado alto, y que sólo aumenta con el número de intentos.

2. Problema del KP (Mochila)

2.1 Planteamiento

El problema del ***KP*** (Mochila) consiste en llenar una mochila de materiales de tal forma que sin exceder la capacidad de ésta se maximice la **relación beneficio/peso**.

Las variables del problema son ***N*** materiales con un beneficio ($\mathbf{p_i}$), un peso ($\mathbf{w_i}$) y la capacidad ***K*** de la mochila.

Para representar una solución a dicho problema, podemos hacer uso de un **vector de booleanos (x)**, en el que cada elemento contiene un 1 ó 0, en función en si el material es introducido en la mochila o no, respectivamente.

0	*1*	*0*	*1*	*1*

El objetivo del problema consiste en **maximizar** la siguiente expresión:

$$f(x) = \sum_{i=1}^{n} p_i x_i$$

Teniendo que satisfacer al mismo tiempo la siguiente expresión:

$$f(x) = \sum_{i=1}^{n} w_i x_i \leq K$$

2.2 Descripción de las pruebas

*En el caso del **KP** disponemos de tres ficheros con instancias del problema:*

- *knapPI_1_200_10000.csv*
- *knapPI_1_1000_1000000.csv*
- *knapPI_12_500_1000.csv*

Realizaremos pruebas utilizando **dos semillas** diferentes para generar soluciones y para cada fichero tomaremos dos instancias suficientemente diferenciadas para intentar observar comportamientos diferentes. Al igual que en el TSP, buscamos el posible impacto a la hora de encontrar una mejor solución realizando más iteraciones.

El siguiente esquema ilustra la metodología de las pruebas:

➔ Para cada fichero :
  → Para cada instancia :
    ➢ Para cada semilla :
      - Generar 1.000 soluciones
      - Generar 1.500 soluciones
      - Generar 30.000 soluciones

Como ya sucedió en el ***TSP***, el problema se ha codificado con las clases y métodos especificados en el enunciado con el lenguaje **C++**.

2.3 Análisis de los resultados

En las siguientes tablas se muestran los resultados desvelados por las pruebas anteriormente descritas:

	Iteraciones realizadas		
	1.000	1.500	30.000
KnapPI_1_200_10000 (instancia 19)	-353.846	-353.846	-301.571
KnapPI_1_200_10000 (instancia 31)	-377.847	-364.446	-345.519
KnapPI_12_500_1000 (instancia 3)	-111.307	-108.889	-101.258
KnapPI_12_500_1000 (instancia 31)	-42.068	-42.068	-39.137
KnapPI_1_10000_1000000 (instancia 67)	2.577.702.704	2.593.342.278	2.599.234.078
KnapPI_1_10000_1000000 (instancia 90)	2.563.446.857	2.598.713.750	2.601.917.475

Tabla 4: Resultados obtenidos utilizando la semilla por defecto (KP)

	Iteraciones realizadas		
	1.000	1.500	30.000
KnapPI_1_200_10000 (instancia 19)	-360.863	-349.937	-327.671
KnapPI_1_200_10000 (instancia 31)	-360.895	-354.849	-320.132
KnapPI_12_500_1000 (instancia 3)	-111.821	-104.775	-102.933
KnapPI_12_500_1000 (instancia 31)	-39.893	-39.893	-39.893
KnapPI_1_10000_1000000 (instancia 67)	2.585.578.676	2.595.446.301	2.602.786.115
KnapPI_1_10000_1000000 (instancia 90)	2.576.676.503	2.590.091.303	2.590.400.470

Tabla 5: Resultados obtenidos utilizando la semilla de índice 20 (KP)

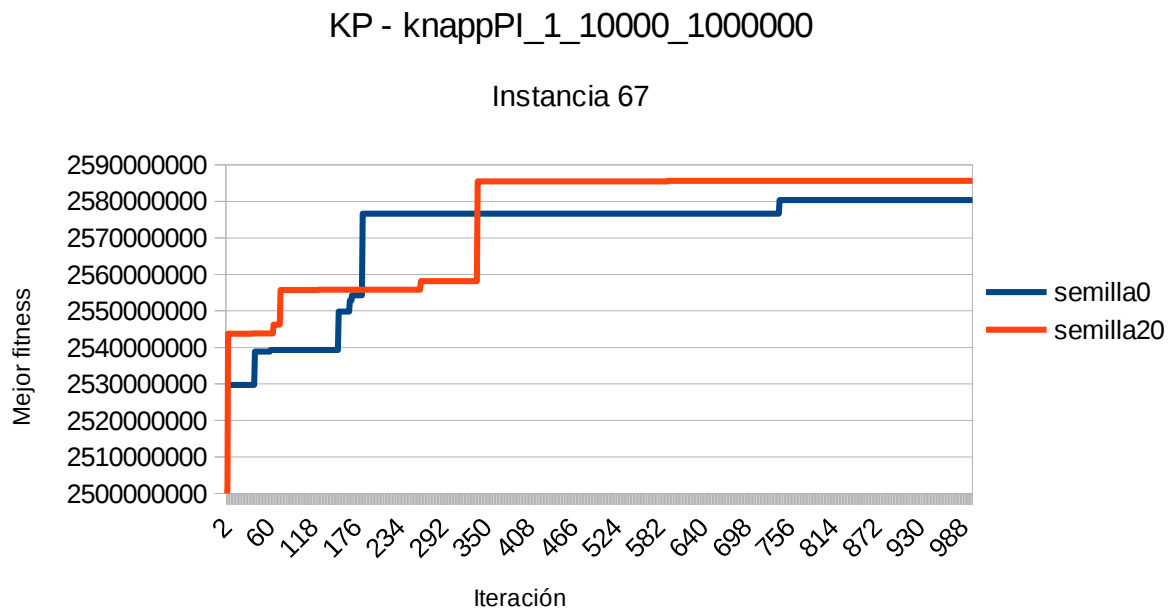
Algo que salta a la vista al analizar las tablas son los valores de **aptitud negativos** en muchas de las pruebas recogidas. Esto se debe a una **penalización**

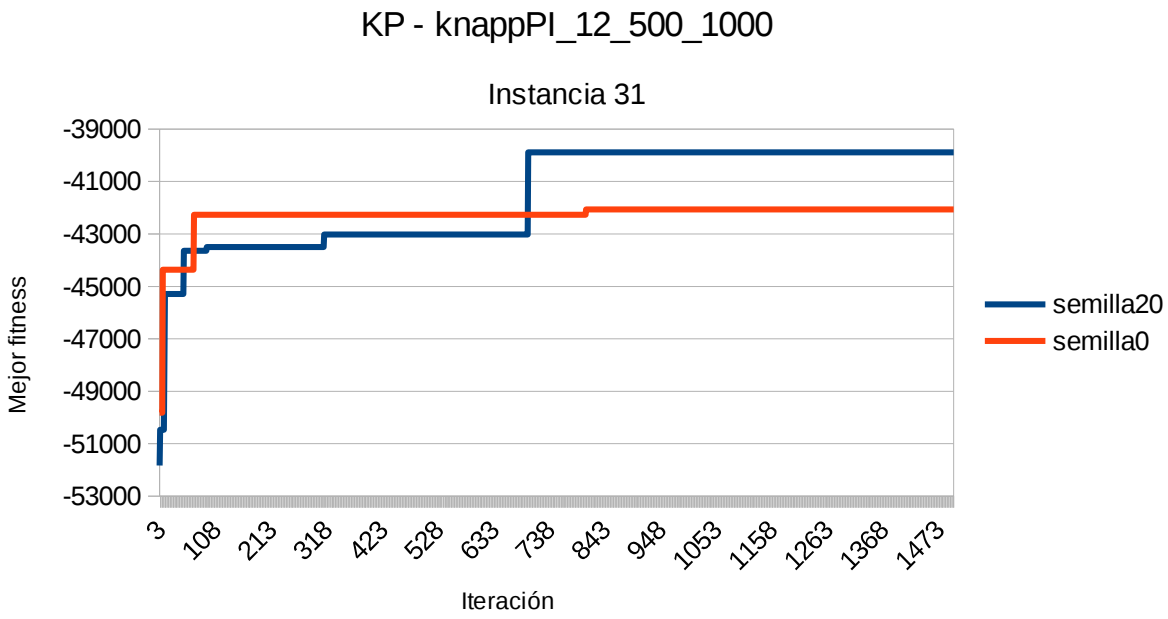
aplicada a las soluciones que hacen uso de un mayor volumen que la capacidad de la mochila (anteriormente referida como **K**). Para estos casos el valor de ***fitness*** es el de la expresión siguiente:

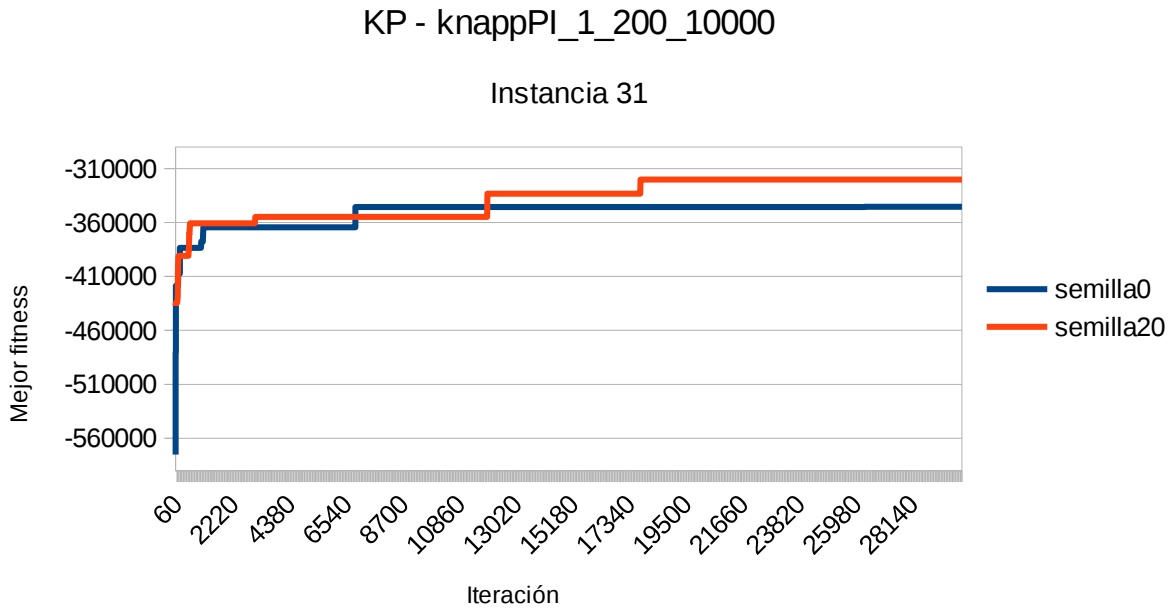
$$f(x)=\sum_{i=1}^{n} p_i x_i - \sum_{i=1}^{n} p_i$$

Es decir, se le resta el **beneficio máximo** (en el que todos los materiales caben en la mochila). El objetivo de esta corrección es poder discriminar entre soluciones no válidas.

Ahora se mostrarán algunas gráficas comparativas entre semillas de soluciones aleatorias para una instancia del problema:







Podemos sacar conclusiones similares a las expuestas anteriormente en el ***TSP***: diferentes semillas conducen a diferentes resultados y la probabilidad de mejorar (o en este caso encontrar) una solución es bastante baja. En general, la búsqueda aleatoria no es un buen método para resolver **problemas complejos**.

3. Problema del CNP (Nodo crítico)

3.1. Codificación y evaluación

El enunciado del problema reza así:

Dado un grafo no dirigido ***G[V, E]*** hay que encontrar un subconjunto ***S*** de ***N*** nodos cuya supresión **interrumpa** el flujo de comunicación del grafo completo.

Una posible representación de una solución para este problema es un **vector de booleanos** en el que cada posición del vector representa un nodo del grafo inicial y el valor asignado indicará si pertenece al subconjunto ***S*** (valor **False**) o no (valor **True**).

0	*1*	*0*	*1*	*1*

El número de nodos pertenecientes a ***S*** no puede superar el valor ***N***.

El objetivo del problema consiste en **minimizar** el número de conexiones, o lo que es lo mismo, **maximizar** el número de desconexiones o componentes conexas:

$$f(x) = \sum_{i,j \in V : i < j} y_{ij}$$

Teniendo que satisfacer al mismo tiempo la siguiente expresión:

$$f(x) = \sum_{i \in V} x_i \leq N$$

Siendo y_{ij} un valor que representa si existe un **camino** entre los nodos ***i*** y ***j***.