Práctica 3.

Metaheurísticas Basadas en Trayectorias

Metaheurísticas – Grado de Ingeniería Informática

Universidad de Córdoba

2015 / 2016

Grupo: NPGroup

Miembros:

Daniel Carmona Martínez
Cristian García Jiménez
Antonio Jesús Jiménez Urbano
Guillermo Sugrañez Pérez

índice de contenidos

Índice de Grafos

Índice de Gráficas

Introducción

En esta práctica se ha abordado nuevamente la resolución de los problemas de la mochila y el viajante de comercio (vistos anteriormente en la práctica 1 y 2), pero siguiendo dos nuevas metaheurísticas basadas en trayectorias (Enfriamiento Simulado e Iterated Greedy) con el objetivo de comparar diversas medidas de rendimiento (fitness, tiempo) en función de las iteraciones realizadas. A la hora de realizar optimizaciones para los tiempos de ejecución se introdujo en las distintas estrategias la posibilidad de elegir el número de soluciones a evaluar.

Este documento consta de tres secciones diferentes. Una primera sección donde se detallan los aspectos de la codificación de cada nueva metaheurística, una segunda donde se evalúan las pruebas realizadas, una tercera donde se listan las conclusiones obtenidas a partir de dichas pruebas y una cuarta donde se continua con el estudio de nuestro problema (Máxima Diversidad) en base a los estudios realizados en esta práctica.

Capítulo 1 - Codificación

Enfriamiento Simulado (simulated annealing)

```
std::vector<double> EnfriamientoSimulado()
```

Este algoritmo se basa en el concepto de vecindario (si encuentra una solución vecina que sea mejor la acepta), pero puede evitar el problema de estancamiento en óptimos locales. Esto lo consigue añadiendo una nueva característica, la temperatura, que permite aceptar soluciones peores, permitiendo abandonar el óptimo local y dirigirse en las sucesivas iteraciones siguientes hacia el óptimo global. Esto lo consigue reduciendo progresivamente la temperatura. Cuando la temperatura es alta (principio del problema) será más probable aceptar soluciones peores (la diversificación será mayor) y cuando la temperatura vaya descendiendo, será menos probable aceptar soluciones peores y por tanto la intensificación será mayor.

La fórmula que determina la **probabilidad** de **aceptar** una solución vecina es la siguiente:

$$P(\Delta E)=\begin{cases}1 & \text{if } \Delta E<0\\ \exp(-\Delta E/kT) & \text{otherwise}\end{cases}$$

- **ΔE**: Es la diferencia de fitness entre una solución y su vecina.
- **k**: Constante de Boltzmann (usada en termodinámica, en nuestro caso, es un valor que nosotros definimos, concretamente 1)
- **T**: Temperatura

Para **enfriar** hemos usado la estrategia **exponencial**:

$$T=\frac{-\Delta E}{\log(\alpha)*k}$$

- **α**: Probabilidad (En nuestro caso α = 0.99)

La **temperatura inicial** viene determinada por:

$$T=\alpha\cdot T$$

- **α**: Probabilidad (En nuestro caso α = 0.99)

Para ambos problemas hemos implementado 3 funciones diferentes. Una función para calcular la temperatura inicial, otra para otra para enfriar y otra que finalmente acepte o no la solución:

```
double TemperaturaInicial()

bool Aceptar(Solucion &s1, Solucion &s2, double &T)

double Enfriar(const double &T,const int &it)
```

Para la ***temperatura inicial*** se han generado 10 soluciones aleatorias y 10 soluciones vecinas a éstas. Para cada pareja de soluciones hemos calculado el correspondiente **ΔE** y se ha realizado la media aritmética para determinar el **ΔE** de la fórmula de la temperatura incial.

Problema de la Mochila (KP) con Enfriamiento Simulado.

Para el problema de la mochila se ha utilizado un ***criterio de aceptación*** que consiste en calcular el **ΔE** y si este es **mayor que 0**, se **acepta** la solución puesto que ésta ha mejorado. Si el **incremento** es **negativo**, entonces **aplicamos la fórmula** que determina la probabilidad de aceptarla. Todas las soluciones generadas no sobrepasan la capacidad de la mochila.

Problema del Viajante de Comercio (TSP) con Enfriamiento Simulado.

Para el problema del viajante se ha utilizado el mismo criterio pero considerando que la solución **mejora** cuando **ΔE** sea **menor que 0**. En **caso contrario**, se aplica la **fórmula** al igual que en el caso anterior salvo que al **ΔE** se le **cambia el signo** para minimizar.

Greedy Iterativo (Iterated Greedy)

```
std::vector<double> IteratedGreedy(int num)
```

En primera instancia el algoritmo genera una primera solución usando heurística y emplea una búsqueda local empleando el algoritmo de primera mejora para optimizarla. Seguidamente entra en un ciclo donde se destruye parcialmente la solución, se reconstruye para posteriormente mejorarla con una búsqueda local y finalmente se selecciona una solución siguiendo un criterio determinado. Para la implementación de este método se han creado tres funciones diferentes y cuyas cabeceras son:

```
void greedy(const int &numElementos)

Solucion Destruir(const int &porcentajeDestruccion, std::vector< int > &eliminados)

void reconstruir(Solucion &parcial, const std::vector<int> &eliminados)
```

Problema de la Mochila (KP) con Greedy Iterativo.

Para el problema de la mochila se ha considerado como ***criterio*** para ***generar una primera solución*** con heurística que ésta no sobrepase la capacidad máxima de la mochila y que a su vez genere una serie de candidatos aleatorios ordenados de mejor a peor ratio. Se generan nuevos candidatos mientras uno consiga entrar en la solución. La fórmula para calcular el mejor ratio es la siguiente:

$$\frac{beneficioMaterial}{pesoMaterial}$$

Para ***destruir*** de forma parcial la solución actual se han considerado tan sólo los materiales que ya han entrado. Se genera una lista con las posiciones que contienen un 1 y se desordena de manera aleatoria eliminando una parte de la solución.

Para ***reconstruir*** se ha seguido el mismo criterio que el empleado para generar una primera solución de manera que partiendo de una solución destruida se pueda generar una nueva empleando heurística y complementándola con una búsqueda local de primera mejora.

Como ***criterio de aceptación*** se ha considerado aquella solución que mejora la anterior y que no supera la capacidad máxima. Si alguna supera esta capacidad es penalizada fijando el fitness como la resta de el valor de ésta y el peso de la solución actual.

Problema del Viajante de Comercio (TSP) con Greedy Iterativo.

Para el problema del viajante de comercio se ha considerado como ***criterio*** para ***generar una primera solución*** un algoritmo voraz aleatorio que selecciona 5 candidatos de forma aleatoria de entre todos los posibles y escoge el mejor de éstos. Este proceso se repite hasta agotar el número de nodos.

Para ***destruir*** de forma parcial la solución actual se elimina una porción de los nodos que la componen de forma aleatoria y se guardan para su posterior reinserción.

Para ***reconstruir*** se introducen de forma aleatoria los nodos extraídos anteriormente. Cabe recalcar que este paso no se realiza de manera heurística porque la solución obtenida sería la misma que la anterior puesto que, la posición en que se reubicaría cada nodo al calcular la distancia mínima, seria la que ocupaba anteriormente.

Como ***criterio de aceptación*** se ha considerado aquella solución que mejora la anterior y que minimiza el camino entre los distintos nodos.

Capítulo 2 - Evaluación

Con el objetivo de comprobar qué algoritmo de los implementados consigue obtener soluciones más óptimas y su evolución a lo largo de las distintas evaluaciones se han ido recogiendo los diferentes datos en cada una de éstas.

Como ***criterio de experimentación*** se ha llevado a cabo la ejecución de 50 pruebas independientes para los casos más pequeños y 20 en los más grandes para no alargar el tiempo de estas últimas.

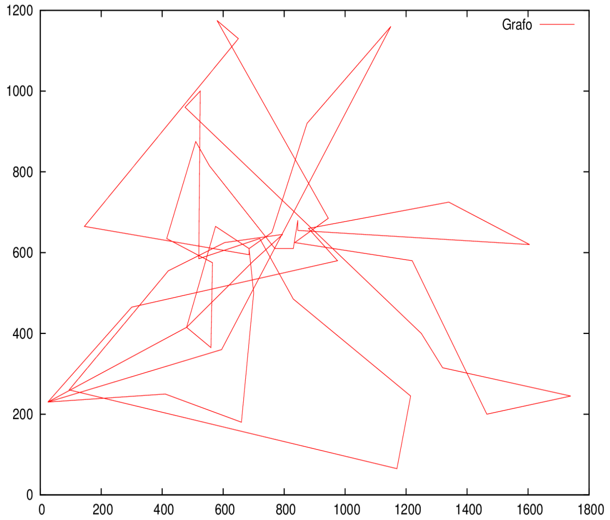
Para cada algoritmo, al observar que aproximadamente tras 1000 evaluaciones no se apreciaban cambios significativos y consultarlo con el profesor, se decidió fijar el número de éstas en dicho valor. Además de esto, no se ha considerado ningún otro criterio de parada.

A continuación, se mostrarán los resultados obtenidos en ambos problemas.

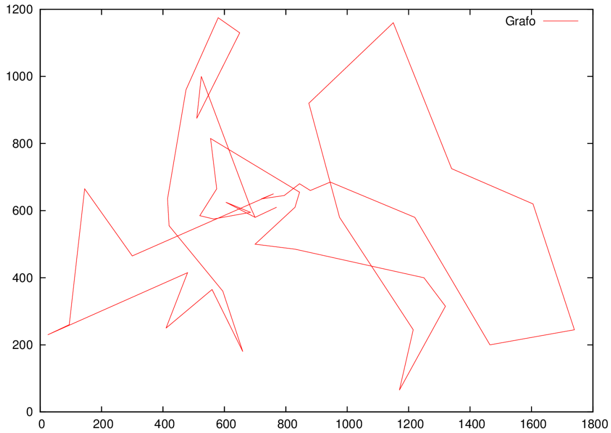
Grafos Viajante de Comercio

Para este problema se ha decidido, además de las gráficas de fitness frente iteraciones, representar la evolución del problema en grafos obtenidos en diferentes instantes de la evaluación.

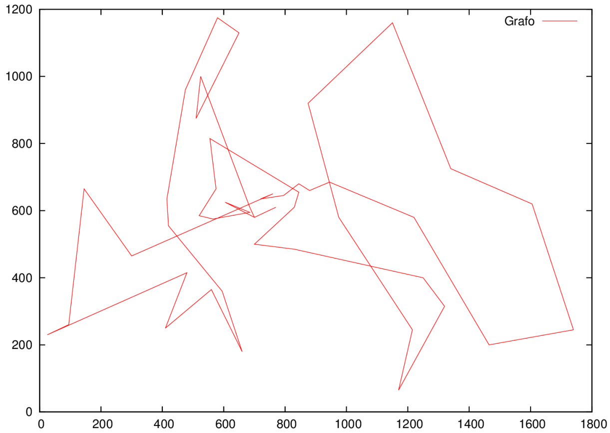
Evolución Greedy Iterativo Berlín_52



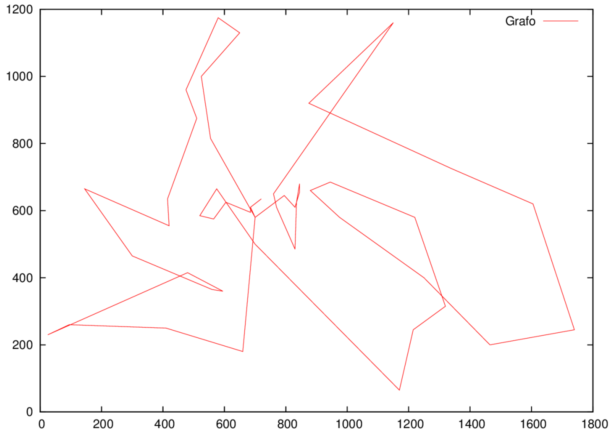
Grafo 1: Berlín_52 Iteración 0. Solución Inicial



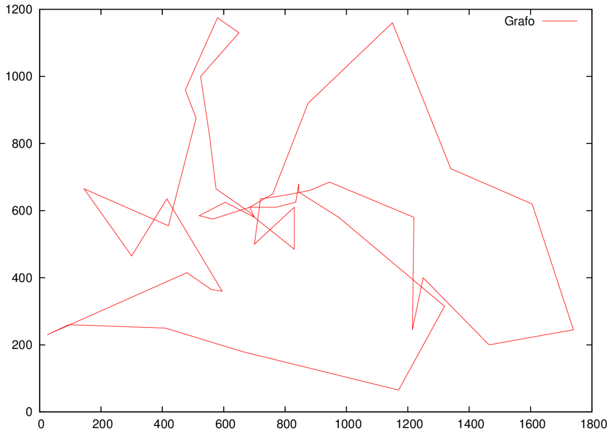
Grafo 2: Berlín_52 Iteración 0. Evaluación 250



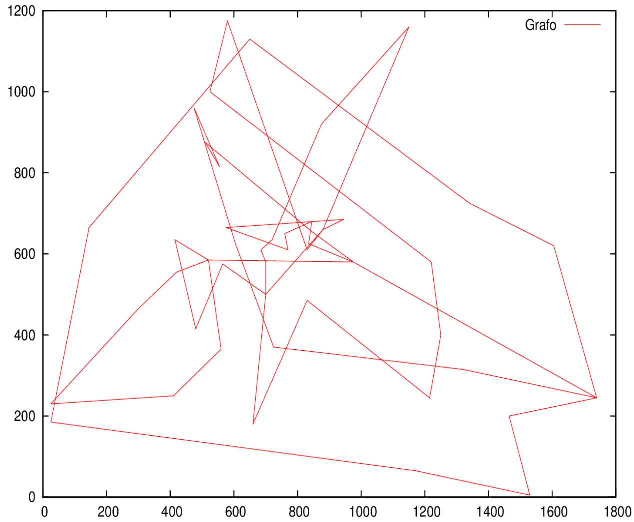
Grafo 3: Berlín_52 Iteración 0. Evaluación 500



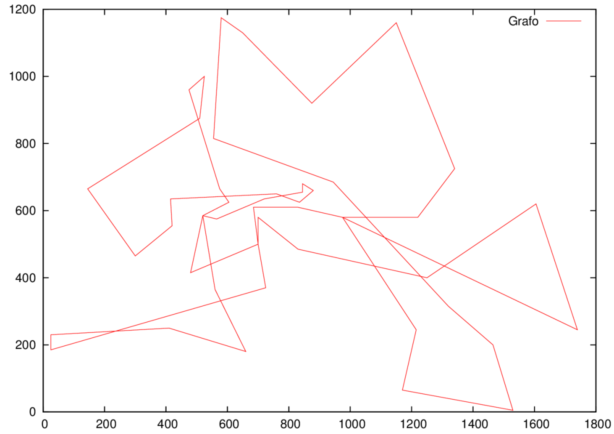
Grafo 4: Berlín_52 Iteración 0. Evaluación 750



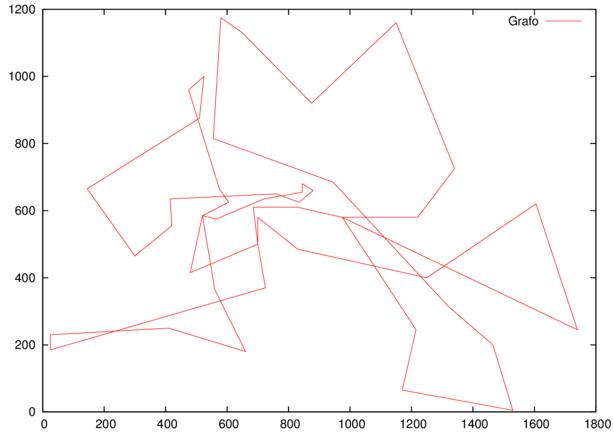
Grafo 5: Berlín_52 Iteración 0. Solución Óptima



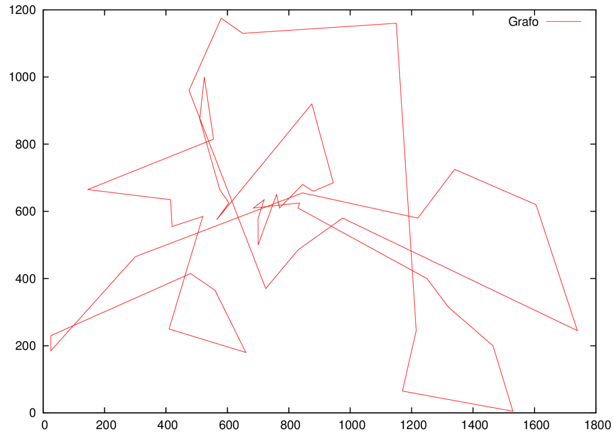
Grafo 6: Berlín_52 Iteración 48. Solución Inicial



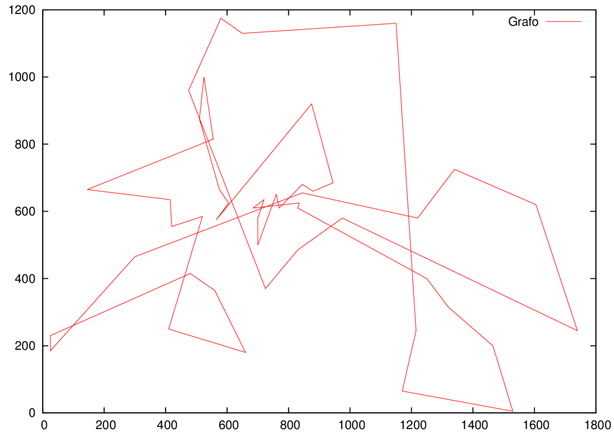
Grafo 7: Berlín_52 Iteración 48. Evaluación 250



Grafo 8: Berlín_52 Iteración 48. Evaluación 500

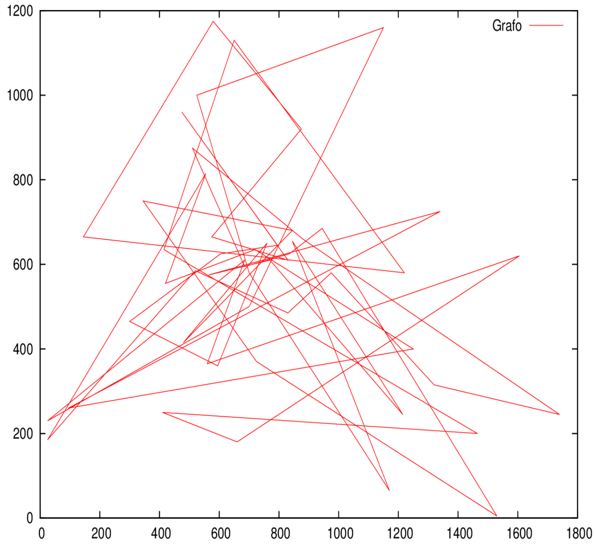


Grafo 9: Berlín_52 Iteración 48. Evaluación 750

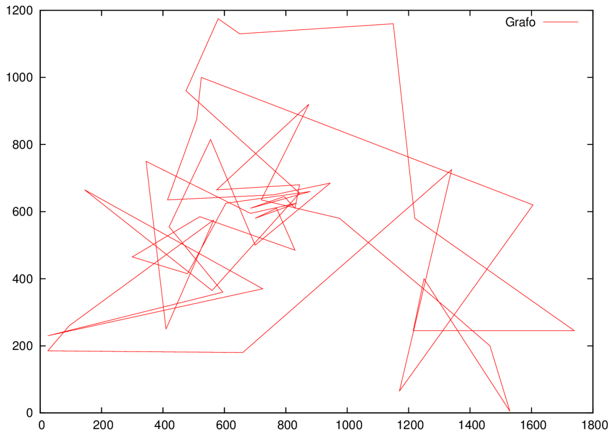


Grafo 10: Berlín_52 Iteración 48. Solución Óptima

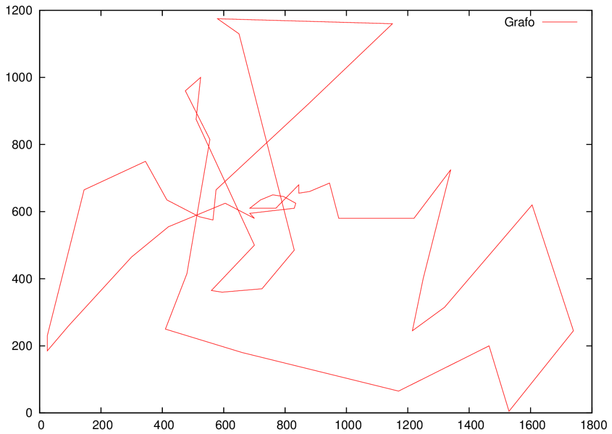
Evolución Enfriamiento Simulado Berlín_52



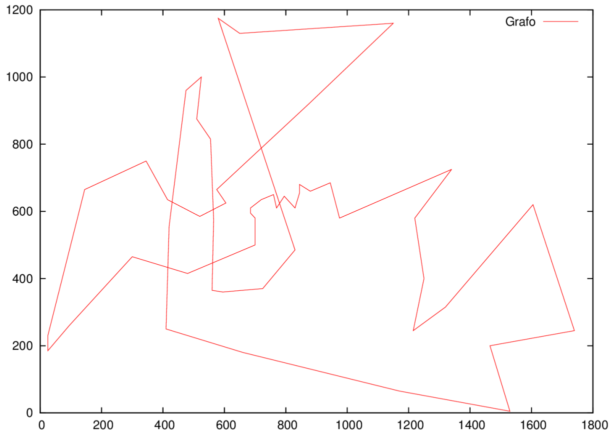
Grafo 11: Berlín_52 Iteración 32. Solución Inicial



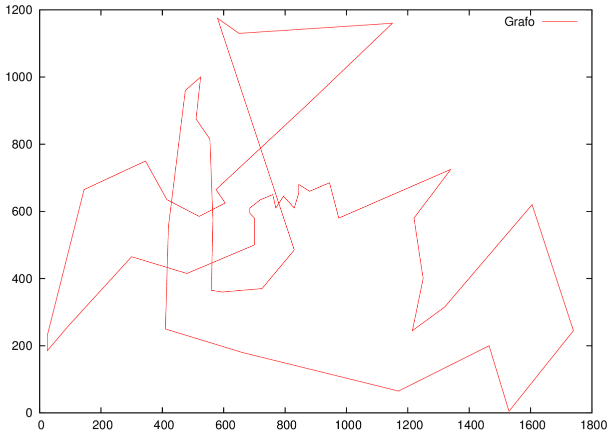
Grafo 12: Berlín_52 Iteración 32. Evaluación 250



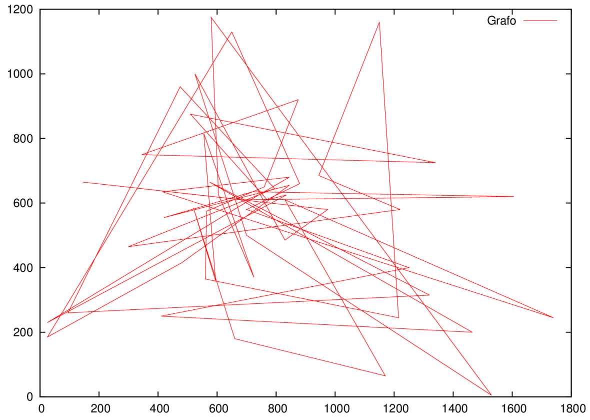
Grafo 13: Berlín_52 Iteración 32. Evaluación 500



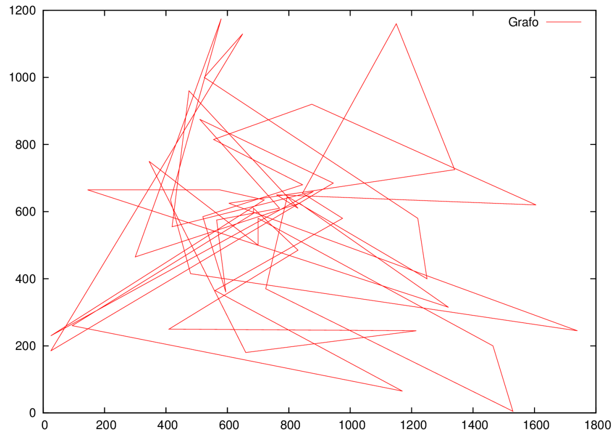
Grafo 14: Berlín_52 Iteración 32. Evaluación 750



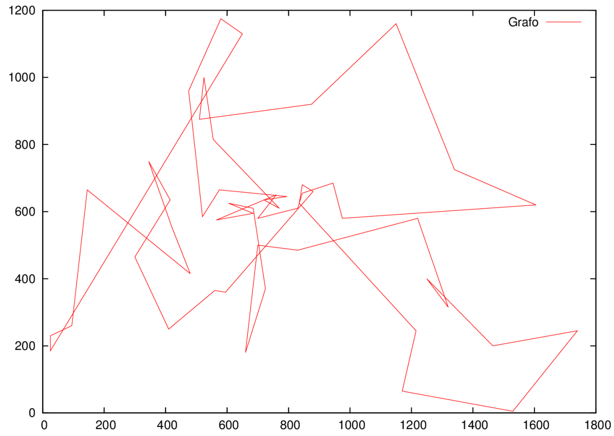
Grafo 15: Berlín_52 Iteración 32. Solución Óptima



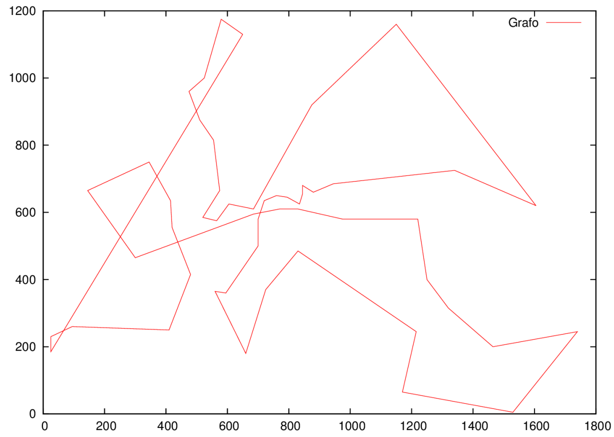
Grafo 16: Berlín_52 Iteración 48. Solución Inicial



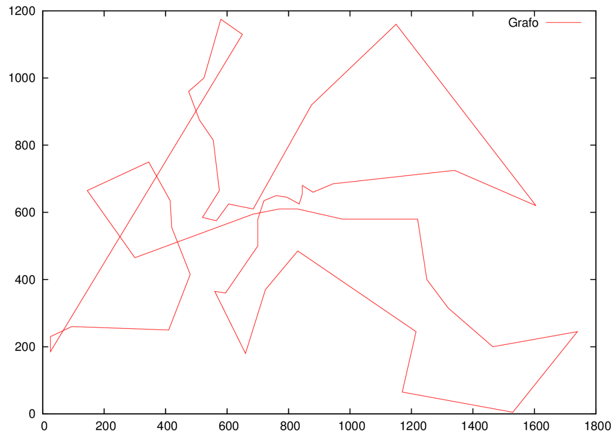
Grafo 17: Berlín_52 Iteración 48. Evaluación 250



Grafo 18: Berlín_52 Iteración 48. Evaluación 500

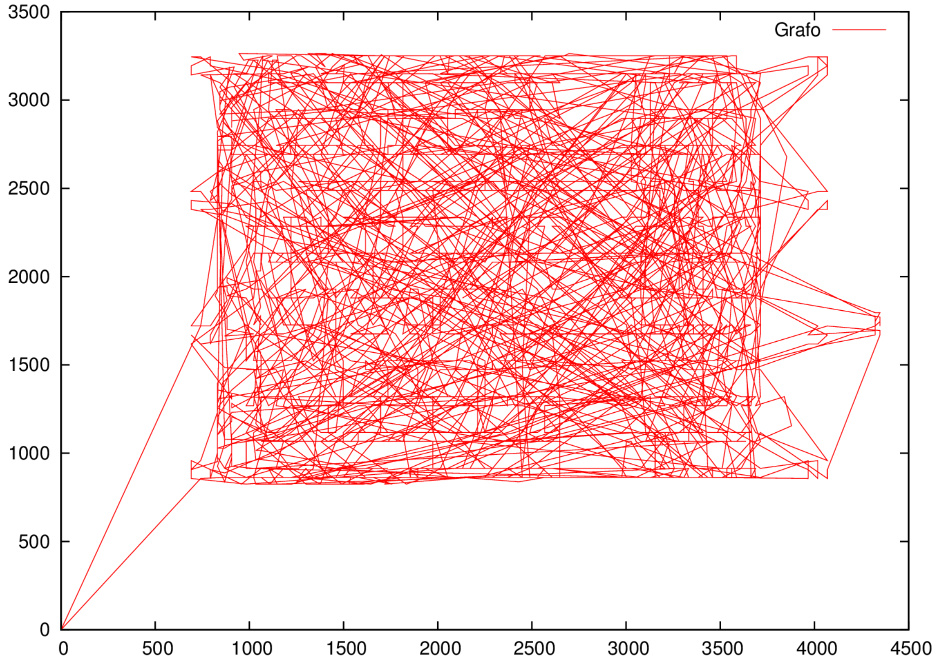


Grafo 19: Berlín_52 Iteración 48. Evaluación 750

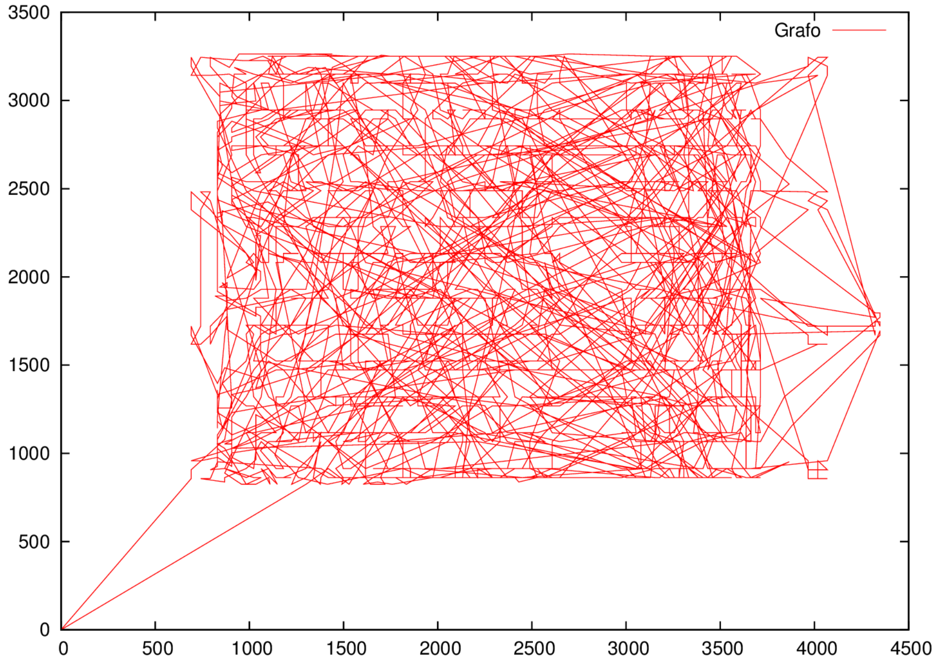


Grafo 20: Berlín_52 Iteración 48. Solución Óptima

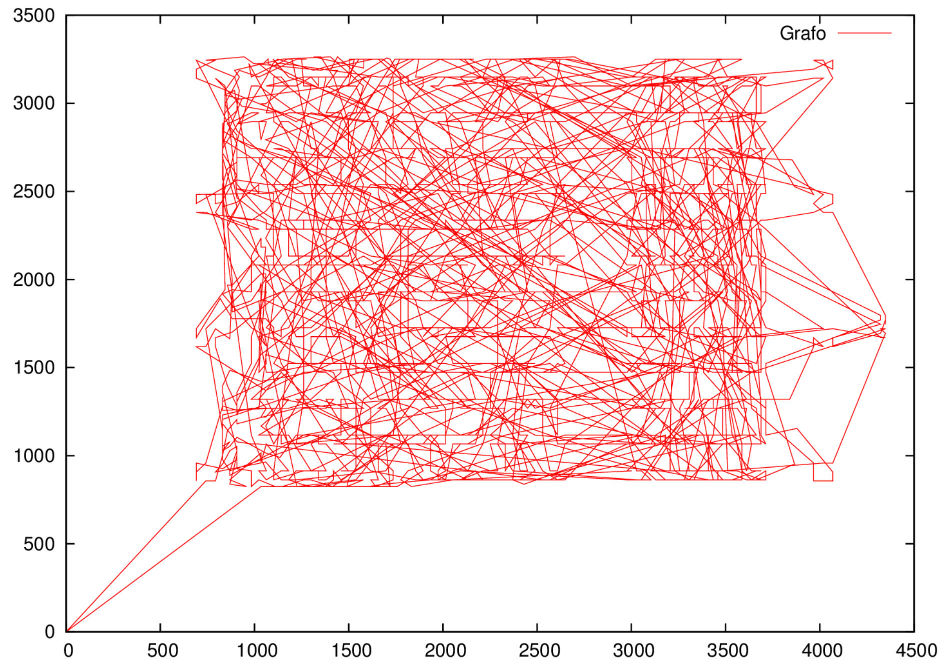
Evolución Greedy Iterativo d_2103



Grafo 21: d_2103 Iteración 0. Solución Inicial

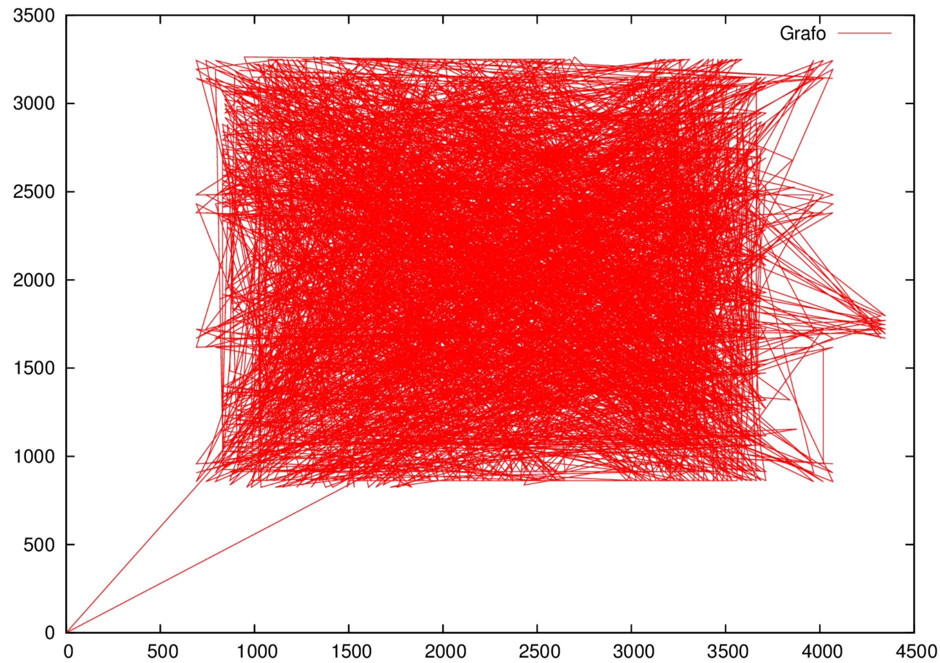


Grafo 22: d_2103 Iteración 0.Evaluación 500

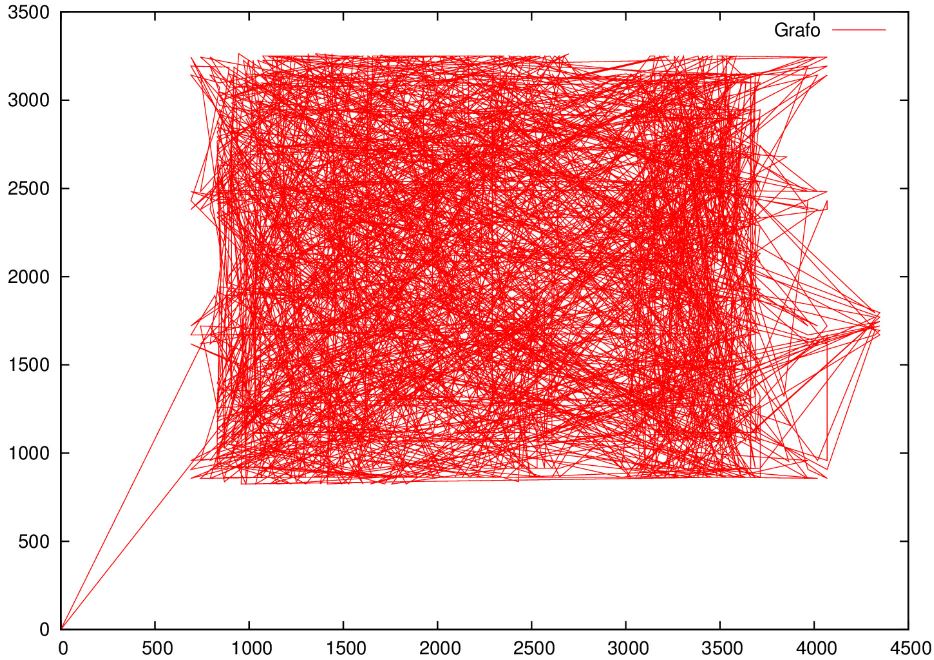


Grafo 23: d_2103 Iteración 0. Solución Óptima

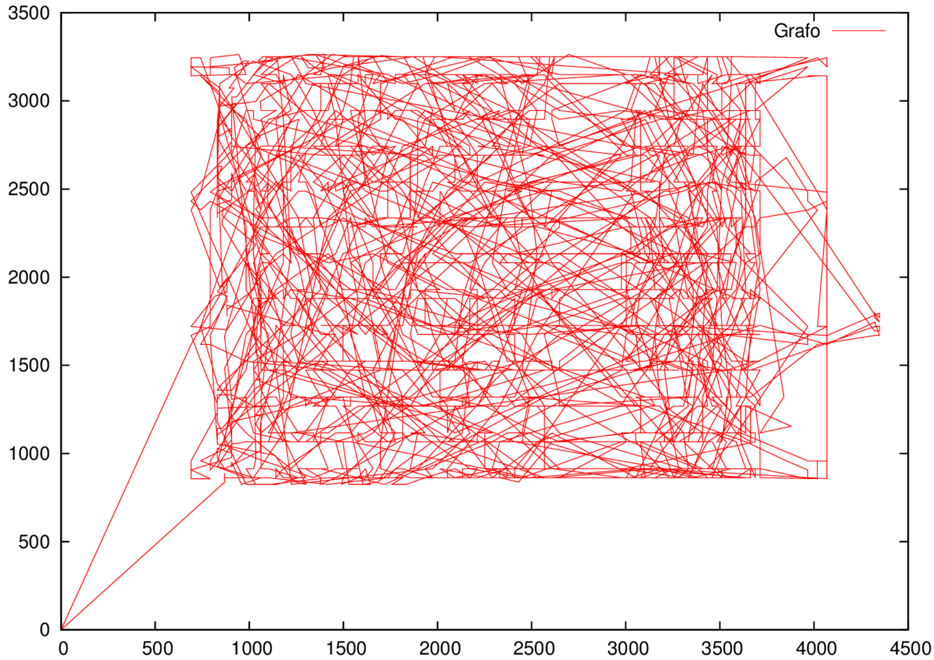
Evolución Enfriamiento Simulado D_2103



Grafo 24: d_2103 Iteración 0. Solución Inicial



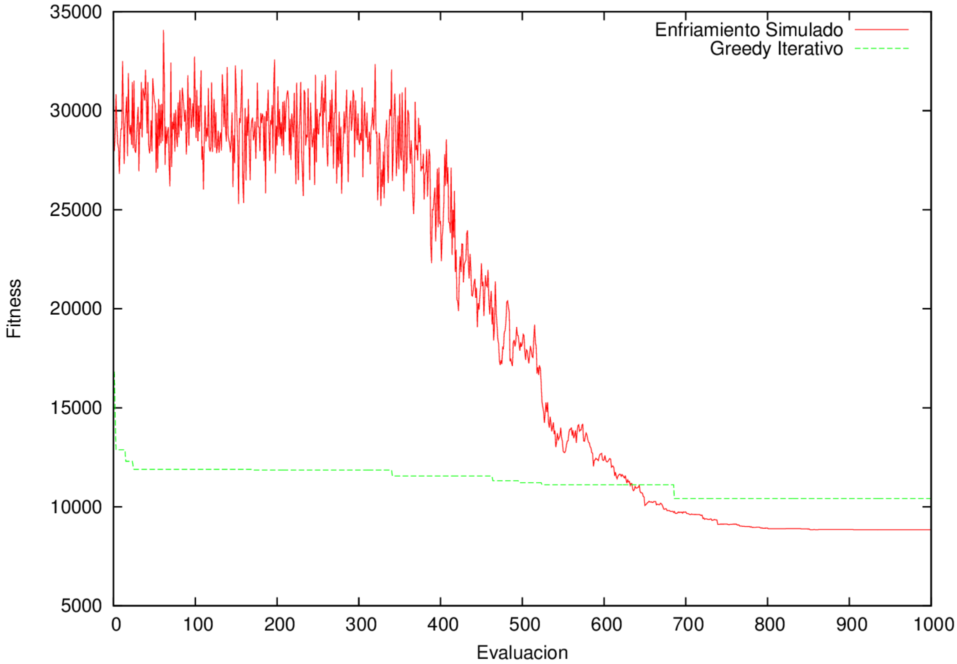
Grafo 25: d_2103 Iteración 0. Evaluación 500



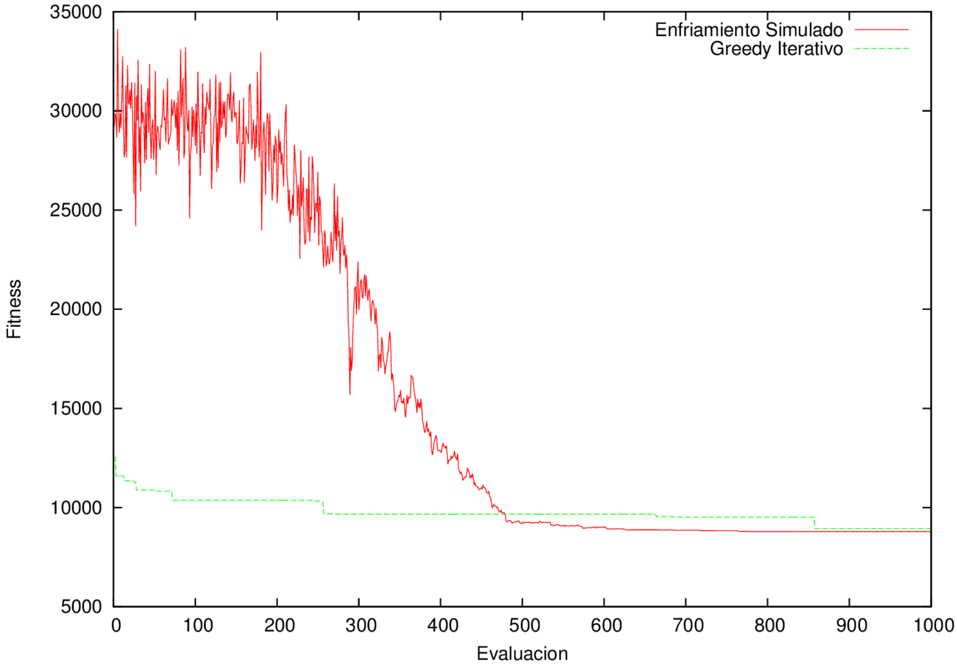
Grafo 26: d_2103 Iteración 0. Solución Óptima

Gráficas TSP

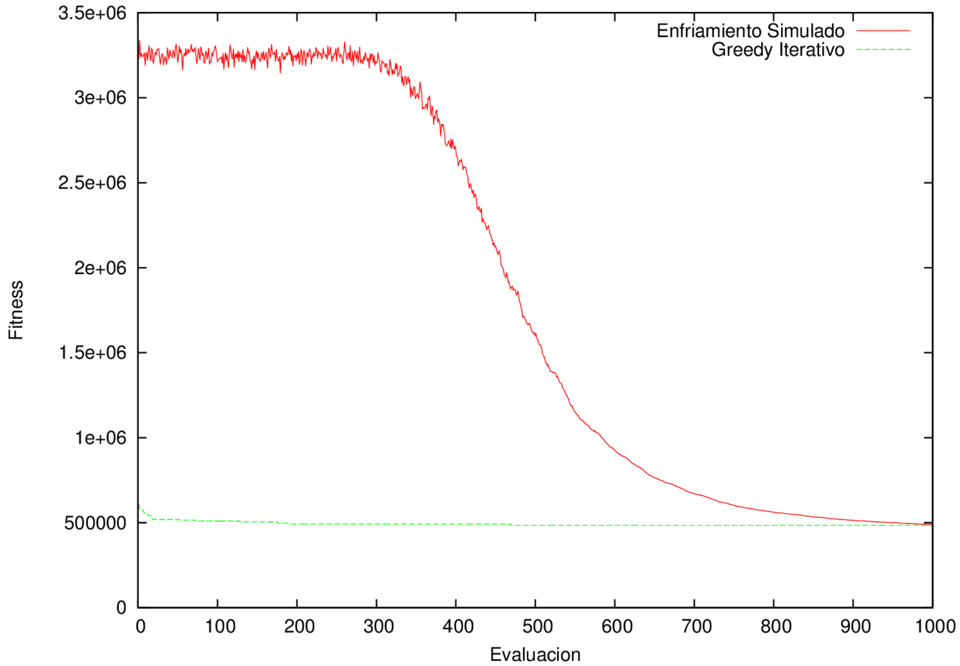
De las 50 gráficas generadas se han seleccionado las correspondientes a 2 iteraciones (las más interesantes observadas) de las totales para cada instancia. Además, se han seleccionado 2 instancias de las 3 disponibles (berlín_52 y d_2103) por considerarse más relevantes. A continuación se muestran:



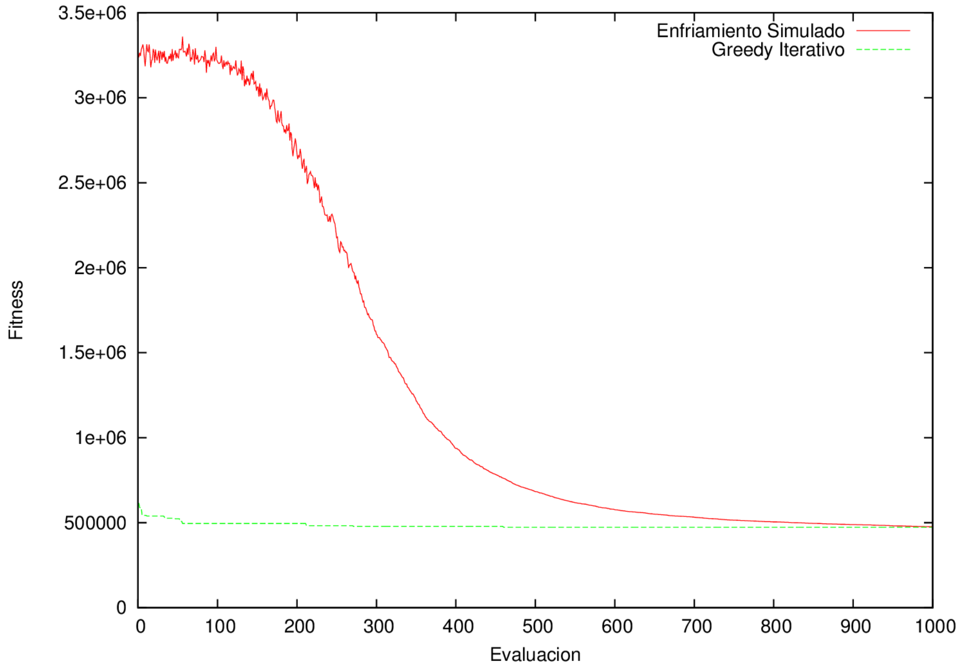
Gráfica 1: Berlín_52. Iteración 5



Gráfica 2: Berlín_52. Iteración 21



Gráfica 3: d_2103. Iteración 3



Gráfica 4: d_2103. Iteración 1

Conclusiones Gráficas TSP

Tabla de costes Enfriamiento Simulado

A continuación se van a mostrar los resultados de fitness óptimos obtenidos en los distintos experimentos para las 3 instancias del problema para poder comprobar la media de éstos y compararlos.

- Berlín_52

9143,08	9712,8	9161,92	8517	9068,35	9480,25	8552,85	8836,82	9447,93	8848,01
9347,52	9011,36	8762,23	8308,61	9143,04	8713,17	9673	8645,97	9567,42	8222,24
9145,53	9575,32	8951,98	8891,16	8754,11	9399,25	9943,08	9417,19	8737,56	9333,05
8669,69	8503,86	8787,75	8806,97	9366,1	9064,91	9224,89	9115,56	8985,16	9613,39
8688,9	9219,05	8764,32	8931,9	8223,69	9392,16	8434,09	8774,93	8203,38	9107,49
Media:									**9003,7998**

- Ch150

10750,3	10094,6	10503,7	9407,09	10689,5	10385,6	11506,4	10539,8	10156,7	10940,8
10061,4	10431,4	11820,7	11096,1	11027,3	11781,7	10414,5	11156,2	10608,8	10080,9
10449,1	9866,42	10516	10144	10431,3	11221,9	10345,4	10301,9	10833	10260,7
11596,1	11102,1	9878,2	10575,7	10874	10651,4	11702,8	11320,6	12024,3	10829,5
9788,18	9302,43	10742,5	11144,5	10221,6	10278,4	10377,5	10329,1	10640,3	11001,5
Media:									**10644,0784**

- D2103

509510	462345
476289	485018
500153	472356
489575	488116
476801	487056
Media:	**484721,9**

Tabla de costes Greedy Iterativo

- Berlín_52

9403,8	10601,9	10344,2	9454,54	9941,73	8815,22	9410,58	9059,98	8883,49	10411,5
9311,3	9448,63	9355,37	9919,03	9697,06	9303,84	9618,95	9540,29	10325,5	10480,4
9718,71	9763,8	9563,11	10127	9973,97	9510,92	9317,89	9551,08	10920,7	9388,56
9669,31	9135,49	8941,12	9759,01	9766,87	10201,6	9449,84	9907,84	9783,47	10100,8
9766,85	10257,6	8948,98	9899,04	9383,87	10342,6	10664,1	9964,42	10270	9729,15
Media:									**9742,1002**

- Ch150

16064	14093	14205	14325	15143	14798	14186	14710	14674	14508
14579	14864	14415	14056	15066	14207	14479	13986	14897	14398
14598	15011	15553	14537	14917	15215	14565	15099	14687	14265
15014	14775	15288	14688	14095	14748	14527	14642	14939	14302
14036	15500	14654	14728	14764	14520	13949	14630	14380	14377
Media:									**14653,566**

- D2103

478389	470707
473314	487581
472213	478678
484298	482011
480222	479485
Media:	**478689,8**

<u>Diferencia de medias en valor absoluto (Enfriamiento – Greedy)</u>

- Berlín52:
  - 9003,7998 - 9742,1002 = 738,3004
- Ch150:
  - 10644,0784 - 14653,566 = 4009,4876
- D2103:
  - 484721,9 – 478689,8 = 6032,1

Como se puede observar en las gráficas, los algoritmos evolucionan de la forma esperada:

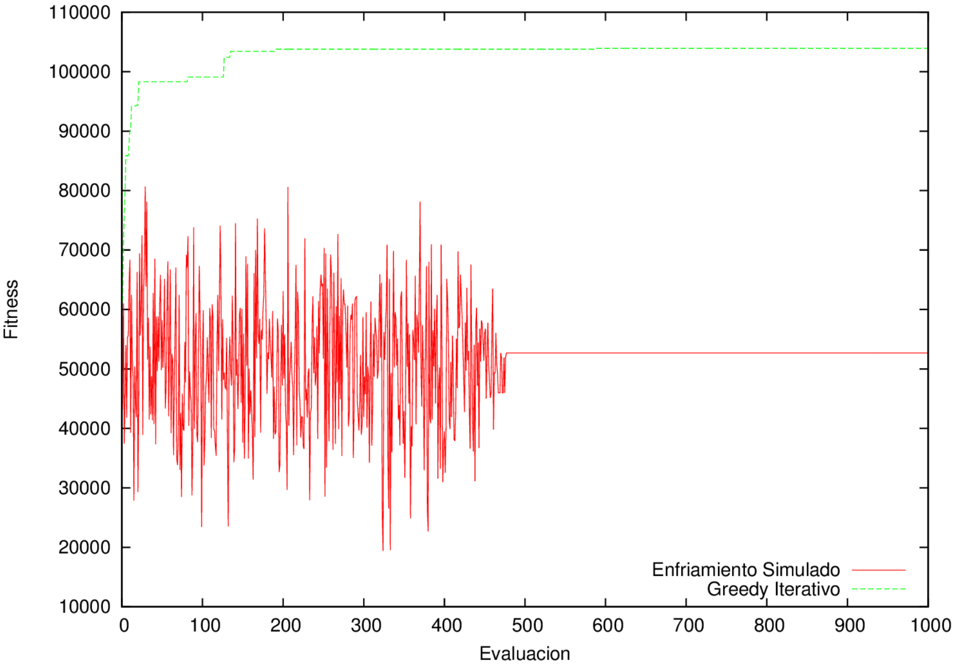
- En ***enfriamiento simulado***, debido a la alta temperatura en primer lugar se aprecia una diversificación porque acepta con mayor probabilidad soluciones peores. A medida que desciende la temperatura se intensifica la obtención de soluciones.
- En ***greedy iterativo***, se aprecia que la solución inicial se genera con heurística, concretamente con un greedy pseudoaleatorio, de modo que es mucho más óptima que la generada por enfriamiento. La linealidad de las soluciones obtenidas es debida a que en cada una de las distintas evaluaciones se aplica una búsqueda local para optimizar la generada con la destrucción y reconstrucción de la mejor solución anterior.

Sin embargo, al ir modificando los diferentes parámetros de ambos algoritmos se ha podido apreciar lo siguiente:

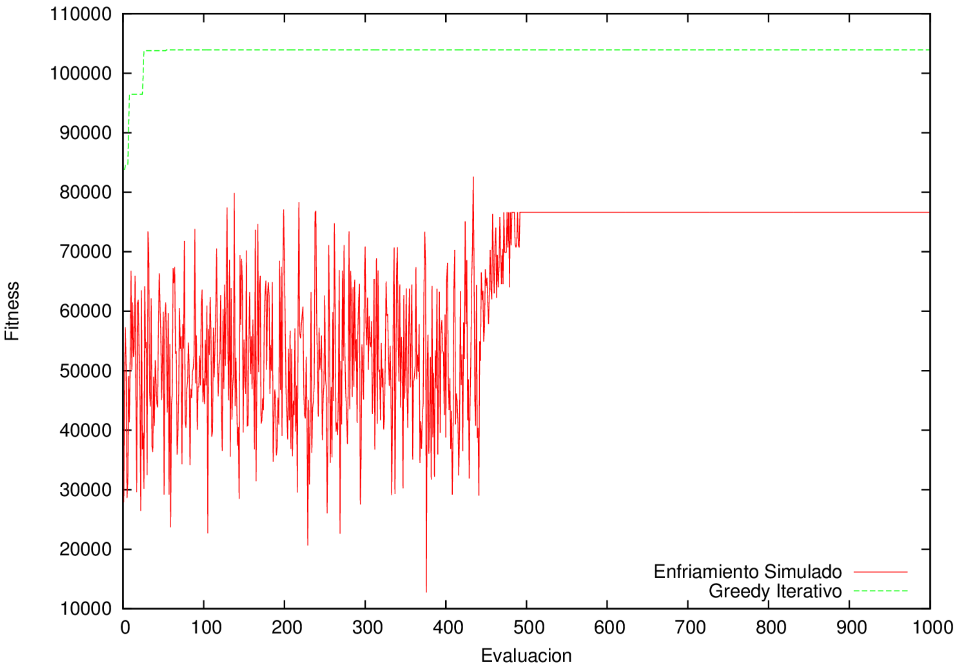
- En ***enfriamiento simulado***:
  - ➢ Al emplear la ***fórmula de Boltzmann*** para enfriar, comprobamos que la temperatura descendía con tanta rapidez que apenas se observaba la diversificación en las gráficas y por eso se ha decidido omitir dichos resultados.
  - ➢ En un principio, se usó como ***condición de parada*** la temperatura, pero era demasiado restrictivo y se decidió dejar como criterio de parada el número de evaluaciones para que diversificase mucho más.
  - ➢ Al ***disminuir la velocidad*** ($\alpha = 0.9999$) con la que desciende la temperatura el algoritmo busca soluciones más despacio (diversificando más) pero necesitaba un mayor número de evaluaciones para encontrar mejores soluciones. Finalmente, conseguía llegar a la misma solución que al enfriar un poco más rápido pero empleando un mayor número de evaluaciones.
- En ***greedy iterativo***:
  - ➢ Se ha podido comprobar que, al modificar el factor de destrucción de la solución, cuanto menos se destruya el vector de puntos mejores soluciones se obtienen en las distintas evaluaciones y viceversa. Esto es debido a que, al considerar el caso más simple, todo los nodos están conectados entre si y esto hace que al destruir mucho dicha solución se obtenga una demasiado alejada de la actual.

Gráficas KP

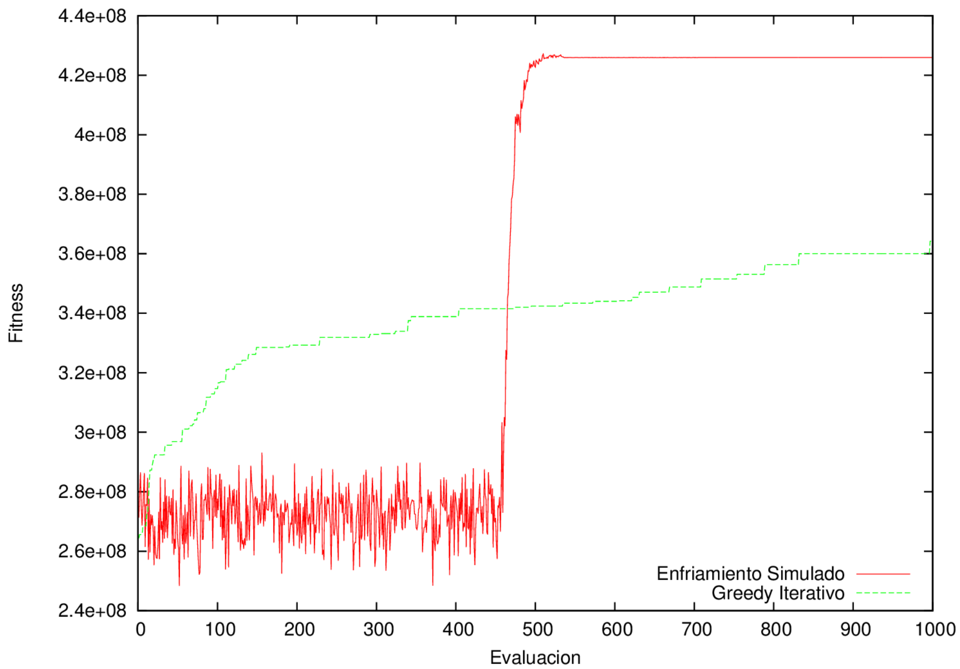
Análogamente a la sección anterior se mostrarán las dos gráficas más significativas correspondientes al problema de la mochila:



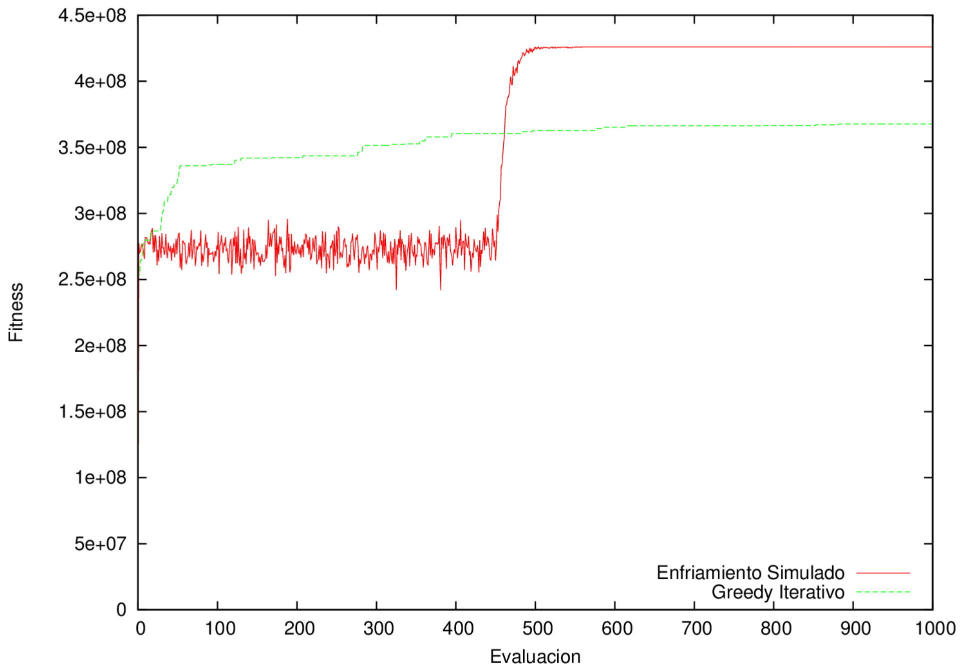
Gráfica 5: Knap_200. Iteración 10



Gráfica 6: Knap_200. Iteración 20



Gráfica 7: Knap_10000. Iteración 12



Gráfica 8: Knap_10000. Iteración 20

Conclusiones Gráficas KP.

Tabla de costes Enfriamiento Simulado

A continuación se van a mostrar los resultados de fitness óptimos obtenidos en los distintos experimentos para las 3 instancias del problema para poder comprobar la media de éstos y compararlos.

- KnapPI_1_200_10000

83876	76639	77989	83842	76639	76646	76639	76639	83842	76639
76639	76639	76639	75806	76639	75806	76639	76605	75772	83842
76639	76605	76639	75806	75806	76639	76605	77148	75806	75806
75806	75806	76639	52697	75806	75806	75806	75806	76818	89200
89713	76639	82351	76639	76639	78241	75806	76639	76639	76639
Media:									**77201**

- KnapPI_12_500_1000

2231	2231	2155	2231	2231	2231	2231	2231	2117	2231
2117	2231	2041	2155	2117	2231	2155	2231	2079	1958
2034	2117	2231	2072	2155	2231	2155	2155	2079	2231
2155	2117	2231	1965	2231	2117	2231	2155	2155	2003
2231	1996	2072	2155	2155	2041	1996	2231	2117	2231
Media:									**2149,6**

- KnapPI_1_10000_1000000

426000000	415000000	428000000	430000000
428000000	432000000	426000000	430000000
440000000	429000000	425000000	425000000
428000000	431000000	429000000	428000000
427000000	430000000	430000000	418000000
Media:			**427781150**

Tabla de costes Greedy Iterativo

- KnapPI_1_200_10000

106025	106025	105221	103913	105353	103913	103913	105353	104948	103913
103913	103913	103913	103913	103913	103913	105296	103913	103786	103913
106025	103913	105221	99210	103913	105221	103913	103627	103913	103913
103101	103913	105353	103913	103913	103913	103913	103913	105296	105890
106025	105296	105221	103913	103913	103913	103913	105221	103913	103913
Media:									**104323,82**

- KnapPI_12_500_1000

2231	2231	2231	2231	2231	2231	2231	2231	2231	2231
2231	2193	2231	2231	2231	2231	2231	2231	2231	2231
2231	2231	2231	2193	2231	2231	2231	2231	2231	2231
2231	2231	2231	2231	2231	2231	2231	2231	2193	2231
2231	2231	2231	2231	2231	2231	2231	2231	2231	2231
Media:									**2228,72**

- KnapPI_1_10000_1000000

368000000	370000000	365000000	359000000
367000000	361000000	364000000	360000000
359000000	361000000	352000000	356000000
361000000	360000000	359000000	362000000
365000000	372000000	359000000	358000000
Media:			**361951300**

Diferencia de medias en valor absoluto (Enfriamiento – Greedy)

- KnapPI_1_200_10000
  - 77201 - 104323,82 = 27122,82
- KnapPI_12_500_1000
  - 2149,6 - 2228,72 = 79,12
- KnapPI_1_10000_1000000
  - 427781150 – 361951300 = 65829850

Tras obtener estos resultados se han sacado las siguientes conclusiones:

- En ***enfriamiento simulado***:
  - Hemos observado que en la ***iteración 10***, ha habido un momento que al enfriar, la temperatura es demasiado baja y el algoritmo no ha sido capaz de escapar de un óptimo local. Esto podría corregirse comprobando en la evaluación de las soluciones que no se ha mejorado la actual en un número de iteraciones y si sucede esto, aumentar de nuevo la temperatura para diversificar.
  - También se puede observar, gracias a las gráficas, que cuanto ***mayor sea la capacidad de la mochila*** y el ***número de elementos que puedan entrar*** en la solución más fácil es alcanzar la solución más óptima.
- En ***greedy iterativo***:
  - Al igual que en el caso del TSP al generar la ***solución inicial con heurística esta es mucho más óptima*** que la generada por enfriamiento simulado sin embargo en este caso la heurística a la hora de realizar la reconstrucción determina más la obtención de buenos resultados que el número de elementos que se destruyen en la solución más óptima hasta el momento. A la hora de destruir en este problema cuantos más elementos se destruyan más nos estaremos alejando de la solución más óptima y por tanto cuanto menos destruyamos esa solución más posible será llegar a una mejor.
  - En este caso ***la primera solución*** obtenida puede hacer que alcanzar otra más óptima sea casi imposible, ya que si la heurística es demasiado buena puede generar la mejor como inicial.

Capítulo 3 - Conclusiones Generales

La necesidad de una heurística en estos algoritmos es un problema ya que en ocasiones esta puede ser excesiva y perjudicar más de lo que aporta, aunque la verdadera complejidad que se ha encontrado en esta práctica es ajustar los parámetros de cada algoritmo para que se adapten correctamente al problema en cuestión y a sus diferentes instancias, esto hace que a pesar de que los diferentes algoritmos funcionen bastante bien se haga muy laborioso conseguir su correcto funcionamiento y se requiera de mucha experimentación para llegar a los resultados óptimos, haciendo estos ajustes críticos para este tipo de algoritmos.

Capítulo 4 - Problema de la diversidad máxima.

Con vista al problema que tenemos que resolver de cara a la práctica 5 se ha decido implementar en ésta una matriz que guarda los costes de las distancias para cada par de nodos y así ahorrar el gran tiempo computacional que supone el calcularlas cada vez que se ejecuta de nuevo el algoritmo.

También, debido a la similitud con respecto a la representación en una gráfica de nuestro problema con el del viajante de comercio se han incluido grafos para visionar la distancia entre nodos.

Con respecto a los demás elementos de la implementación, se ha decido esperar a realizar en la siguiente práctica un algoritmo evolutivo puesto que puede ayudarnos en una mejor medida en la resolución del problema seleccionado.