



UNIVERSIDAD DE GRANADA

Redes y Sistemas Complejos

Práctica 3:
***Estudio Comparativo de Métodos para
Poda y Visualización de Redes***

Javier León Palomares

18 de noviembre de 2017

Índice

1. Análisis de cienciogramas	2
1.1. Poda de redes	2
1.2. Visualización	4
1.2.1. <i>Kamada-Kawai</i>	5
1.2.2. <i>Fruchterman-Reingold</i>	6
1.2.3. <i>Yifan Hu</i>	7
1.2.4. <i>OpenOrd</i>	8
2. Análisis de eficiencia	9
2.1. Análisis de eficiencia híbrido	13
2.1.1. <i>Pathfinder</i> original ($O(n^4)$)	14
2.1.2. <i>Pathfinder</i> binario ($O(n^3 \log n)$)	14
2.1.3. <i>Fast Pathfinder</i> ($O(n^3)$)	15
2.1.4. <i>MST Pathfinder</i> teórico ($O(n^2 \log n)$)	15
2.1.5. <i>MST Pathfinder</i> práctico ($O(n^3)$)	16
2.2. Comparativa final	17

1. Análisis de cienciogramas

1.1. Poda de redes

La primera parte de esta práctica consiste en analizar mediante algunas métricas los efectos que tienen los algoritmos de poda en redes. En este caso, se han podado 10 cienciogramas de distintos países y años, y para ello se ha utilizado el algoritmo *Binary Pathfinder* por su mayor velocidad de ejecución frente al algoritmo *Pathfinder* original. Los cienciogramas considerados son: Argentina 2005, Canadá 2005, Chile 2004, China 2002, Cuba 2004, Mexico 2005, Portugal 2005, España 2002, Reino Unido 2002 y Estados Unidos 2002. Las tablas de resultados para cada uno de los cienciogramas se muestran a continuación:

Argentina 2005 (n = 266)	Número de enlaces	Densidad	Distancia media
Red original	17938	0.5089	1.497
$q = 2$	324	0.0092	4.7
$q = 3$	277	0.0078	5.726
$q = 4$	269	0.0076	6.08
$q = 5$	268	0.0076	6.114
$q = n - 1$	267	0.0076	6.211

Canadá 2005 (n = 288)	Número de enlaces	Densidad	Distancia media
Red original	30574	0.7397	1.2604
$q = 2$	340	0.0082	4.993
$q = 3$	301	0.0072	5.932
$q = 4$	290	0.0070	6.734
$q = 5$	287	0.0069	7.789
$q = n - 1$	287	0.0069	7.789

Chile 2004 (n = 256)	Número de enlaces	Densidad	Distancia media
Red original	14778	0.4528	1.553
$q = 2$	318	0.0097	4.78
$q = 3$	265	0.0081	6.245
$q = 4$	258	0.0079	7.122
$q = 5$	256	0.0078	7.411
$q = n - 1$	256	0.0078	7.411

China 2002 (n = 259)	Número de enlaces	Densidad	Distancia media
Red original	20661	0.6184	1.383
$q = 2$	306	0.0091	4.983
$q = 3$	268	0.0080	5.986
$q = 4$	262	0.0078	6.823
$q = 5$	260	0.0078	7.593
$q = n - 1$	260	0.0078	7.593

Cuba 2004 (n = 237)	Número de enlaces	Densidad	Distancia media
Red original	9518	0.3403	1.694
$q = 2$	285	0.0102	4.407
$q = 3$	255	0.0091	5.271
$q = 4$	249	0.0089	5.607
$q = 5$	247	0.0088	5.975
$q = n - 1$	247	0.0088	5.975

Mexico 2005 (n = 270)	Número de enlaces	Densidad	Distancia media
Red original	20110	0.5537	1.4503
$q = 2$	336	0.0092	4.8887
$q = 3$	284	0.0078	6.0235
$q = 4$	275	0.0076	6.8280
$q = 5$	274	0.0075	7.3910
$q = n - 1$	273	0.0075	7.4173

Portugal 2005 (n = 265)	Número de enlaces	Densidad	Distancia media
Red original	20699	0.5917	1.4101
$q = 2$	318	0.0091	5.0590
$q = 3$	280	0.0080	6.1083
$q = 4$	272	0.0078	6.8656
$q = 5$	270	0.0077	7.3390
$q = n - 1$	268	0.0076	7.7503

España 2002 (n = 264)	Número de enlaces	Densidad	Distancia media
Red original	21807	0.6281	1.3723
$q = 2$	320	0.0092	4.7919
$q = 3$	274	0.0079	5.8289
$q = 4$	265	0.0076	6.5460
$q = 5$	263	0.0076	6.8703
$q = n - 1$	263	0.0076	6.8703

Reino Unido 2002 ($n = 276$)	Número de enlaces	Densidad	Distancia media
Red original	28707	0.7564	1.2444
$q = 2$	326	0.0086	5.1140
$q = 3$	288	0.0076	6.2202
$q = 4$	280	0.0074	6.9354
$q = 5$	279	0.0073	6.9795
$q = n - 1$	276	0.0073	7.7335

Estados Unidos 2002 ($n = 276$)	Número de enlaces	Densidad	Distancia media
Red original	31292	0.8245	1.175
$q = 2$	314	0.0083	5.2260
$q = 3$	287	0.0076	6.2948
$q = 4$	279	0.0073	7.5312
$q = 5$	277	0.0073	7.6031
$q = n - 1$	275	0.0072	8.1678

Lo primero que podemos observar a nivel general es la gran diferencia en número de enlaces entre las redes originales y las resultantes con $q = 2$. Esto se debe a que las redes originales, a la vista de su considerable densidad, presentan una alta redundancia que ya se puede ver muy reducida teniendo en cuenta tan sólo conexiones indirectas entre pares de nodos involucrando un tercero.

Generalmente, con mayores valores de q se consigue rebajar un poco más la densidad de la red, hasta llegar a $q = n - 1$, que permite incluso eliminar un enlace directo si se puede encontrar un camino mejor que pase por todos los demás nodos. Sin embargo, como consecuencia de realizar podas tan fuertes desde el inicio, las diferencias son pequeñas en comparación; en particular, el número final de enlaces parece estabilizarse en torno a $q = 5$, ya que la probabilidad de encontrar caminos mejores involucrando cada vez más nodos decrece muy rápido.

Otro efecto apreciable en las tablas obtenidas es el incremento de la distancia media. Debido a que se eliminan muchas conexiones directas, el número de enlaces necesarios para conectar cada par de nodos de la red es mayor. Esto es natural, porque los algoritmos de poda tratan de facilitar el análisis de una red a cambio de una pérdida asumible de información; por ello, la conectividad se mantiene, pero no a su coste original.

1.2. Visualización

Una vez analizados los cambios provocados por las podas utilizando algunas métricas, es el momento de comprobar cómo afectan a la visualización de las redes. Con este objetivo, vamos a representar los cienciogramas de España 2002 y Estados Unidos 2002 utilizando cuatro algoritmos distintos: *Kamada-Kawai*, *Fruchterman-Reingold*, *Yifan-Hu* y *OpenOrd*.

1.2.1. Kamada-Kawai

En las siguientes figuras podemos comprobar cómo la poda permite ahora identificar de un vistazo los principales nodos de los cienciogramas y algunas comunidades.

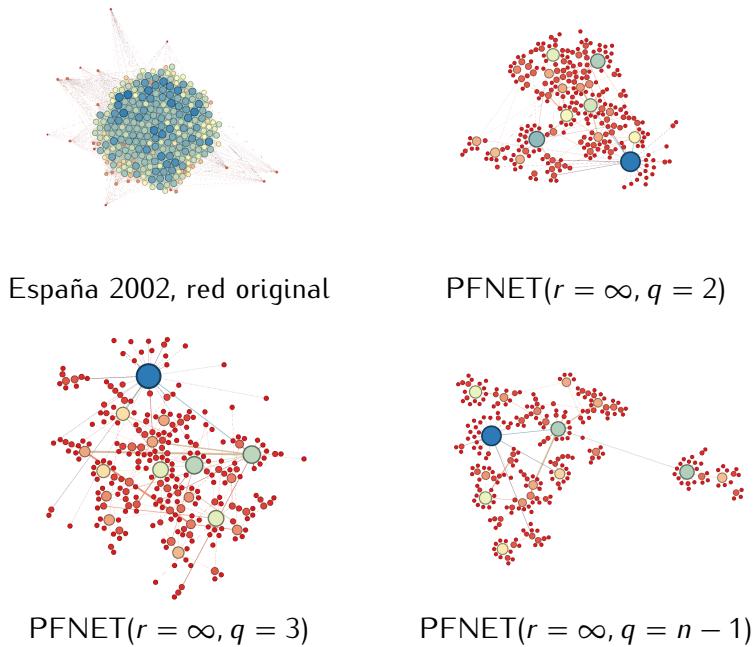


Figura 1: Visualizaciones mediante *Kamada-Kawai* de las PFNETs de España 2002.

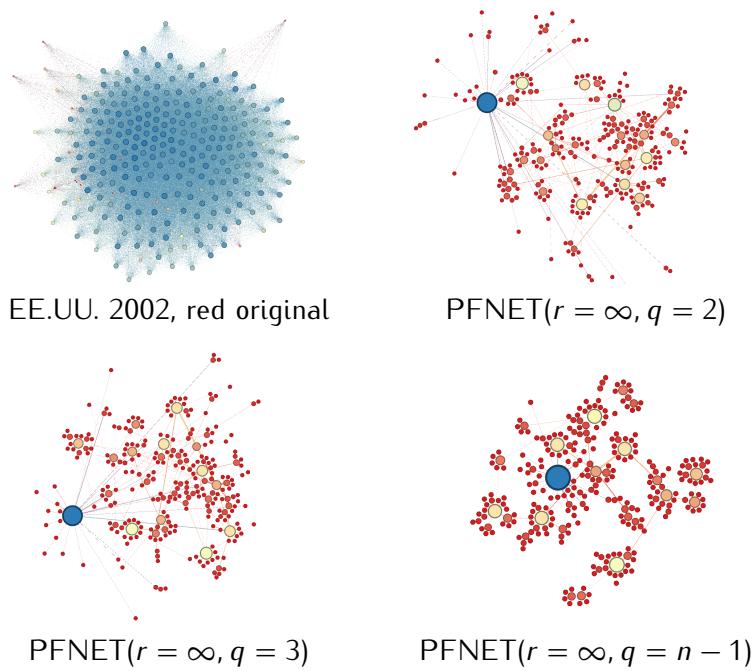


Figura 2: Visualizaciones mediante *Kamada-Kawai* de las PFNETs de Estados Unidos 2002.

1.2.2. *Fruchterman-Reingold*

En este caso, la poda ayuda de nuevo a esclarecer cuáles son los nodos principales, pero el algoritmo de visualización no contribuye a detectar comunidades fácilmente.

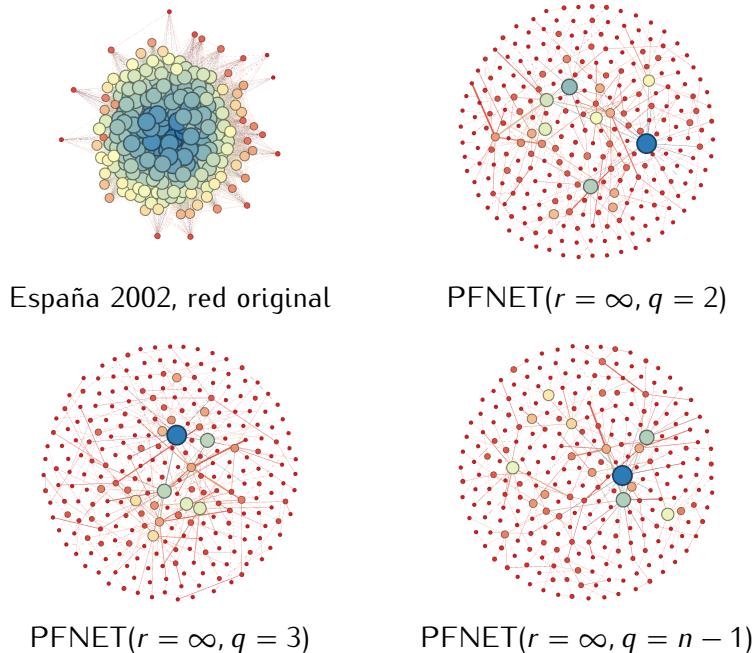


Figura 3: Visualizaciones mediante *Fruchterman-Reingold* de las PFNETs de España 2002.

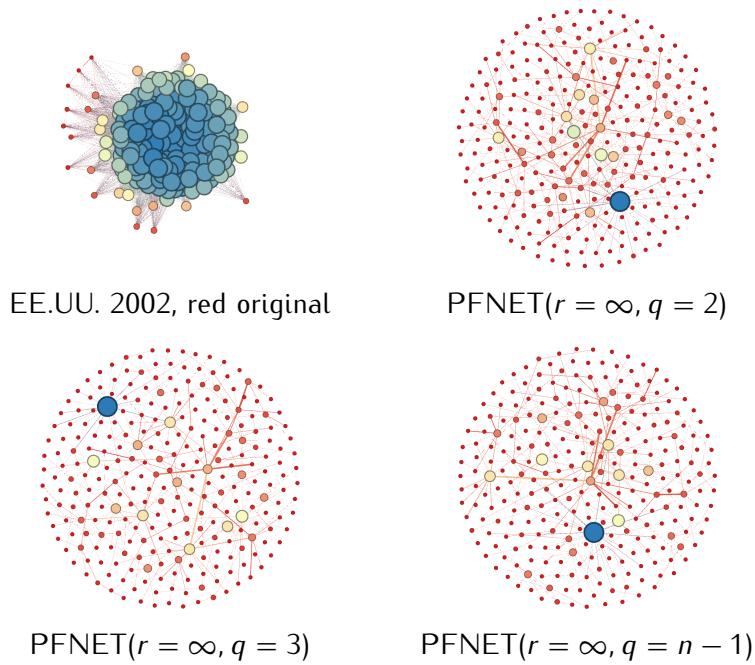


Figura 4: Visualizaciones mediante *Fruchterman-Reingold* de las PFNETs de Estados Unidos 2002.

1.2.3. *Yifan Hu*

Adicionalmente, consideramos este algoritmo de visualización. Vemos que, una vez realizada la poda, también es capaz de hacer fácil la detección de nodos importantes y comunidades.

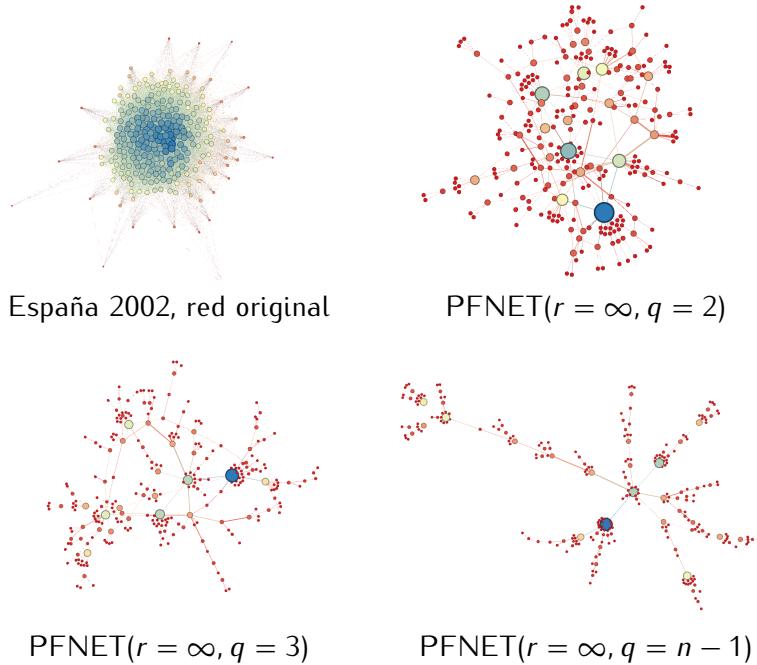


Figura 5: Visualizaciones mediante *Yifan-Hu* de las PFNETs de España 2002.

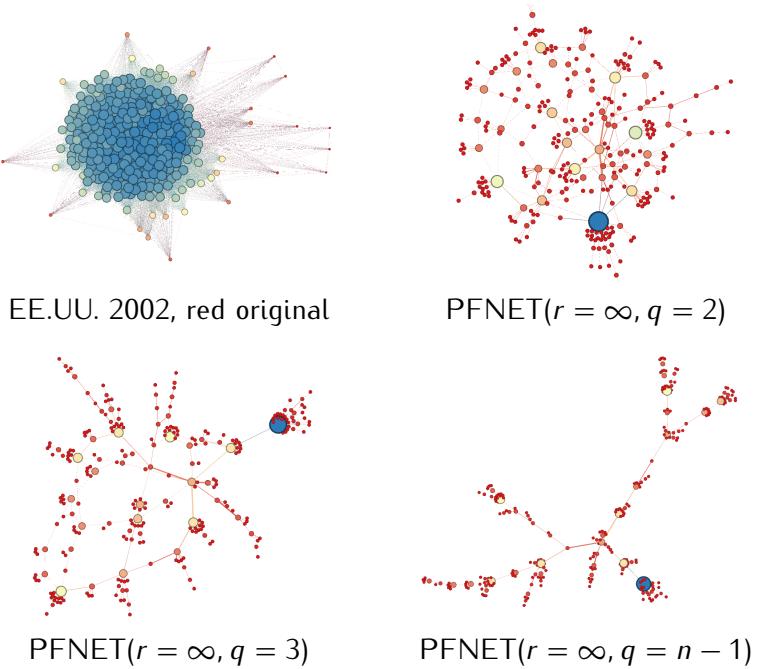


Figura 6: Visualizaciones mediante *Yifan-Hu* de las PFNETs de Estados Unidos 2002.

1.2.4. *OpenOrd*

Para finalizar, el algoritmo *OpenOrd* se basa en el enfriamiento simulado para dividir en fases la distribución de la red. Adicionalmente y según su documentación, es uno de los pocos algoritmos de fuerzas escalable a más de un millón de nodos.

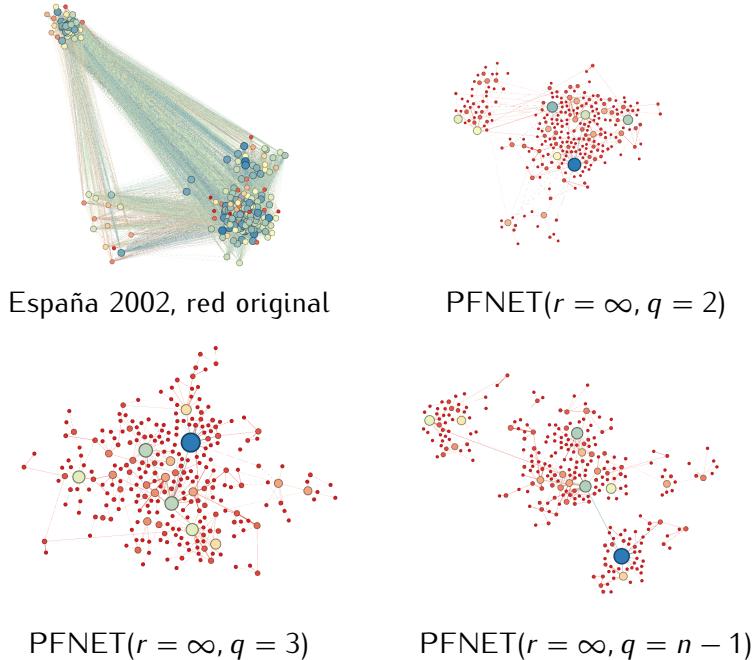


Figura 7: Visualizaciones mediante *OpenOrd* de las PFNETs de España 2002.

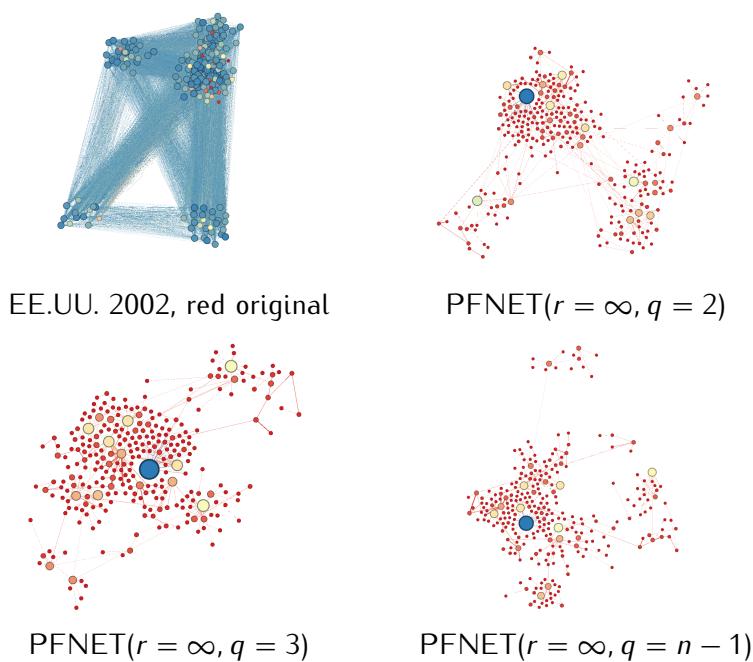


Figura 8: Visualizaciones mediante *OpenOrd* de las PFNETs de Estados Unidos 2002.

Como hemos observado en las figuras anteriores, llevar a cabo una poda de la red es un primer paso importante para conseguir una visualización aceptable al ojo humano. Una vez hecho esto, es necesario considerar qué algoritmo de visualización se adapta mejor a nuestro objetivo.

Por una parte, el algoritmo *Fruchterman-Reingold* dibuja la red en forma circular, lo cual puede complicar la tarea de identificar comunidades (e incluso nodos relevantes, si no asignamos algún tipo de distinción por grado u otra medida de importancia). Asimismo, el algoritmo *OpenOrd* es útil en su función pero no mucho, ya que está especialmente indicado para redes de gran tamaño.

Por otra parte, los algoritmos *Kamada-Kawai* y *Yifan-Hu* son en general superiores a la hora de separar los distintos grupos de nodos fuertemente relacionados entre sí, como ha quedado patente en las imágenes anteriores. Además de esto, el algoritmo *Yifan-Hu* refleja particularmente bien la estructura de árbol generador minimal que resulta de aplicar una poda con $q = n - 1$.

2. Análisis de eficiencia

La segunda parte de esta práctica consiste en, ya estudiados los efectos de los algoritmos de poda sobre las redes, obtener una visión aproximada de la eficiencia de los mismos. Con este fin, se han ejecutado todos ellos sobre 5 redes aleatorias distintas no dirigidas, de densidad 0.1 y valores de los enlaces en el intervalo [1,0, 10,0]. A continuación podemos ver los resultados promedio:

Tamaño	Media E	Media D	Pathfinder Original	Pathfinder Binario	Fast Pathfinder	MST Pathfinder (Teórico)	MST Pathfinder (Práctico)
500	12413	0.0995	98.5062	4.6066	0.2987	0.0022	0.0019
1000	49954	0.1000	1792.5008	96.7949	2.3471	0.0081	0.0081
2000	199926	0.1000	>1800	1097.7609	18.8061	0.0411	0.0406
5000	1249204	0.0999	>1800	>1800	284.5615	0.3350	0.3356
10000	4999258	0.0999	>1800	>1800	>1800	1.5655	1.5662
15000	11246362	0.0999	>1800	>1800	>1800	3.9314	3.8992
20000	19998724	0.0999	>1800	>1800	>1800	7.6756	7.5956

En primer lugar, comprobamos cómo los tiempos de ejecución del algoritmo **Pathfinder original** se disparan para tamaños de red relativamente pequeños; esto es debido a que su orden de complejidad teórico es $O(n^4)$; además, su alto consumo de memoria principal ralentiza aún más su ejecución.

A continuación, tenemos el algoritmo **Pathfinder binario**, que rebaja la complejidad a $O(n^3 \log n)$; como queda patente, es una mejora importante pero no suficiente, ya que sigue sobrepasando el límite establecido por nosotros de 1800 segundos de ejecución.

El siguiente algoritmo es el **Fast Pathfinder**, cuya complejidad es de $O(n^3)$. Viendo los tiempos que consigue, está clara la mejora, aunque una vez más termina colapsando al aumentar el tamaño de la entrada.

Finalmente, los tres algoritmos anteriores estaban basados en programación dinámica; una vez constatada su ineficiencia y para concluir esta comparativa, es el momento de ver qué ocurre si se emplea un enfoque *Greedy*, como hacen las dos últimas variantes. Estas

variantes, **teórica** y **práctica**, son esencialmente el mismo algoritmo, pero cambiando la estructura de datos subyacente para conseguir órdenes de eficiencia de $O(n^2 \log n)$ y $O(n^3)$ respectivamente. La variante **práctica** se llama así porque termina antes en muchos casos aun teniendo una complejidad mayor que la **teórica**; en la tabla se puede observar que esto ocurre para varios de los tamaños probados, si bien la diferencia no parece ser muy significativa. No obstante, si realizamos una comparación con los algoritmos basados en programación dinámica, la ganancia en velocidad es lo bastante relevante como para hacer factible su ejecución para tamaños de red mucho mayores, al menos en términos de tiempo.

Para comprobar la eficacia de las podas en redes más grandes que las de la primera parte de la práctica, vamos a tratar de obtener una visualización lo más estética posible para una red podada de 2000 nodos, otra de 5000 y otra de 10000 (todas con $r = \infty$ y $q = n - 1$).

Es importante destacar que, aunque consigamos una distribución de los nodos que permite al menos distinguirlos unos de otros e identificar enlaces entre ellos, no existe más patrón de interacción que el aleatorio. También podremos observar la propiedad de las redes libres de escala, es decir, que la cantidad de nodos con un determinado grado disminuirá según aumente el valor de éste siguiendo una ley de potencias; en las imágenes que veremos a continuación, esto se traduce en muchos nodos pequeños de color rojo y muy pocos grandes de color azul, existiendo otros de colores y tamaños intermedios:

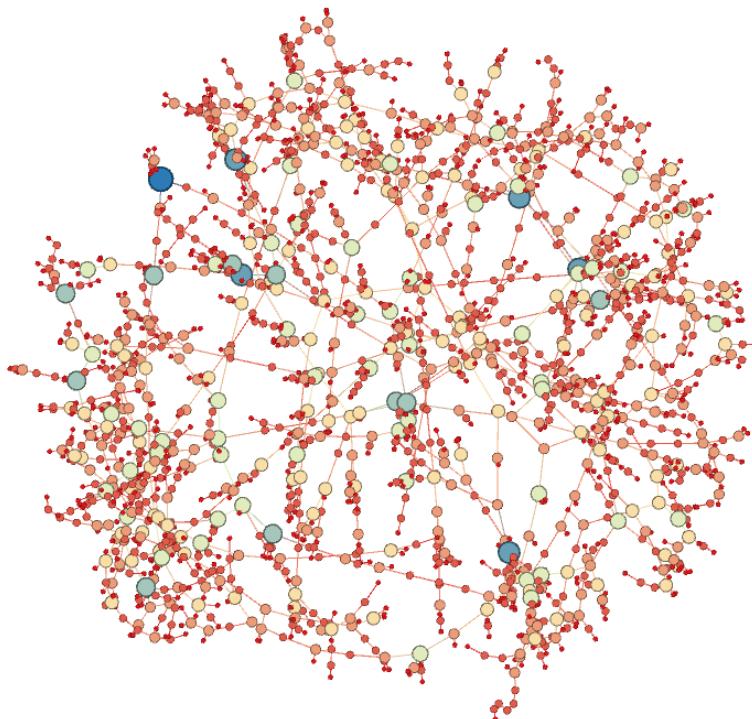


Figura 9: Red aleatoria de 2000 nodos dibujada con el algoritmo *Yifan-Hu*.

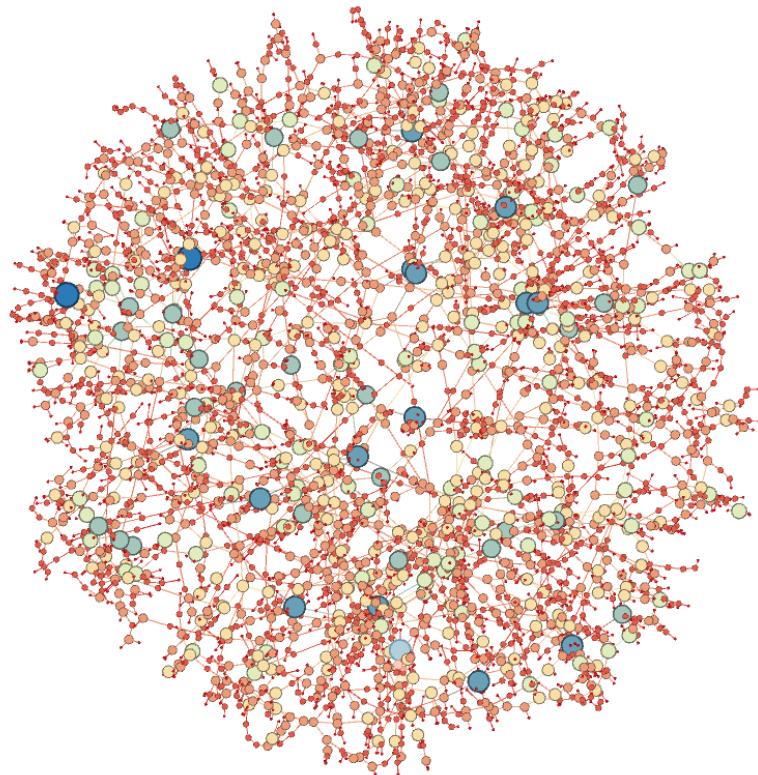


Figura 10: Red aleatoria de 5000 nodos dibujada con el algoritmo *Yifan-Hu*.

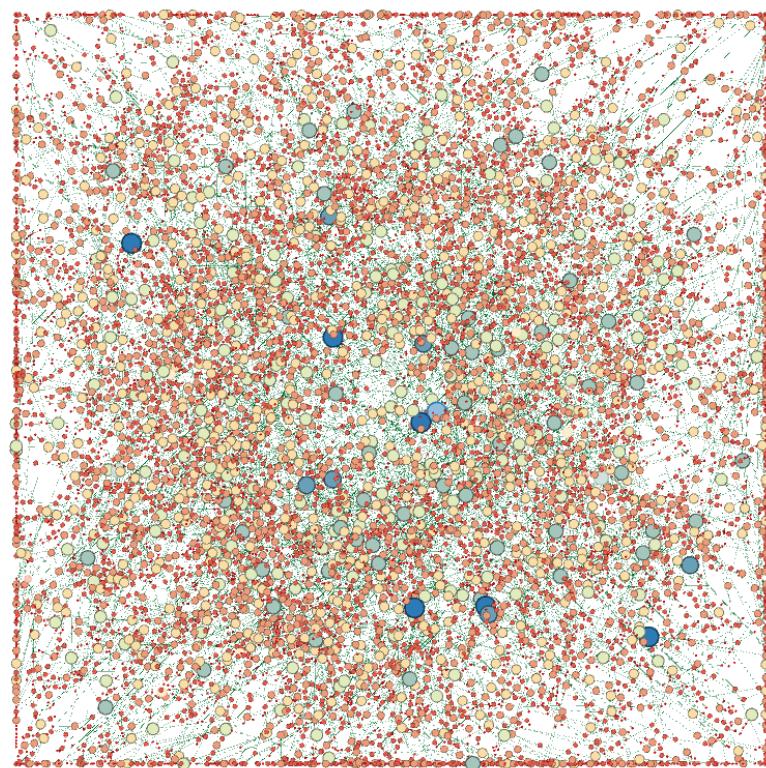


Figura 11: Red aleatoria de 10000 nodos dibujada con el algoritmo *Yifan-Hu*.

Como último apunte, anteriormente se ha comentado que el algoritmo *OpenOrd* era más apropiado para redes de gran tamaño en número de nodos. Si representamos las mismas redes mediante dicho algoritmo, queda lo siguiente:

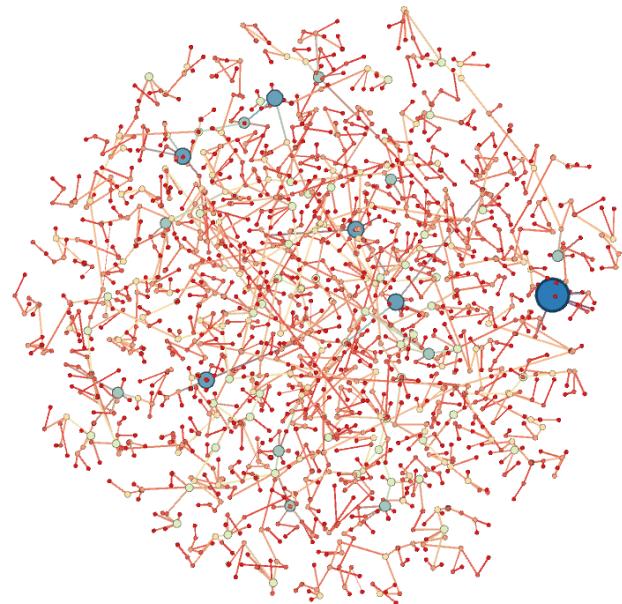


Figura 12: Red aleatoria de 2000 nodos dibujada con el algoritmo *OpenOrd*.

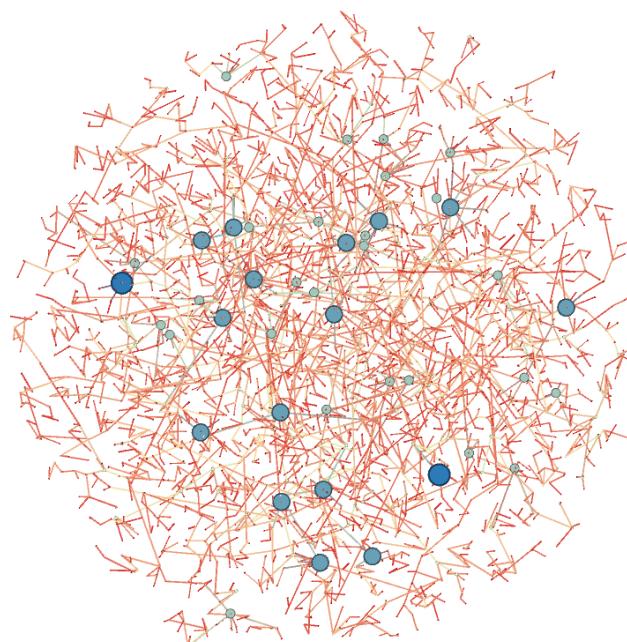


Figura 13: Red aleatoria de 5000 nodos dibujada con el algoritmo *OpenOrd*.

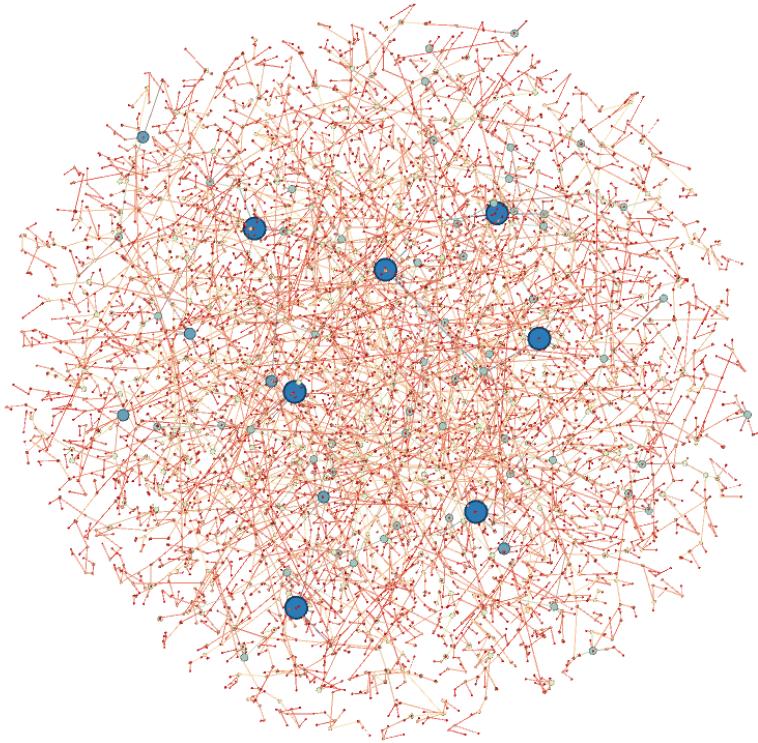


Figura 14: Red aleatoria de 10000 nodos dibujada con el algoritmo *OpenOrd*.

El resultado parece ser que este algoritmo consigue una visualización algo más estética que el *Yifan-Hu*, pese al hecho de que son redes aleatorias.

2.1. Análisis de eficiencia híbrido

Para cerrar la sección de análisis de eficiencia de las distintas variantes del algoritmo *Pathfinder*, vamos a comprobar que la eficiencia teórica concuerda con los datos obtenidos. Con este objetivo en mente, usando *GNUPlot* ajustaremos las funciones correspondientes a los distintos conjuntos de datos y representaremos el proceso con gráficas para poder verificar que todo es correcto. Finalmente, compararemos todos los algoritmos en una misma gráfica para así tener una idea de lo que significan sus diferencias en orden de complejidad.

2.1.1. *Pathfinder* original ($O(n^4)$)

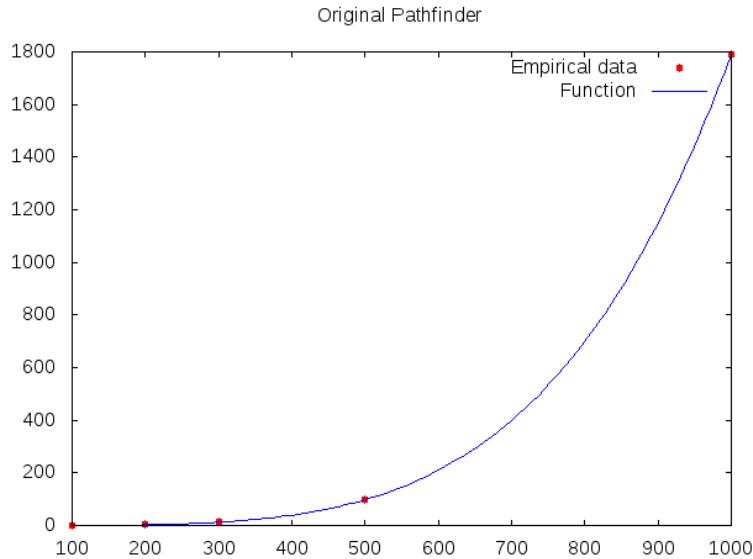


Figura 15: Ejecución del algoritmo *Pathfinder* original y su función ajustada.

La función correspondiente es:

$$f(x) = 2,38964 \cdot 10^{-9} \cdot x^4 - 8,60103 \cdot 10^{-7} \cdot x^3 + 0,000304599 \cdot x^2 - 0,0442996 \cdot x + 2,16017$$

2.1.2. *Pathfinder* binario ($O(n^3 \log n)$)

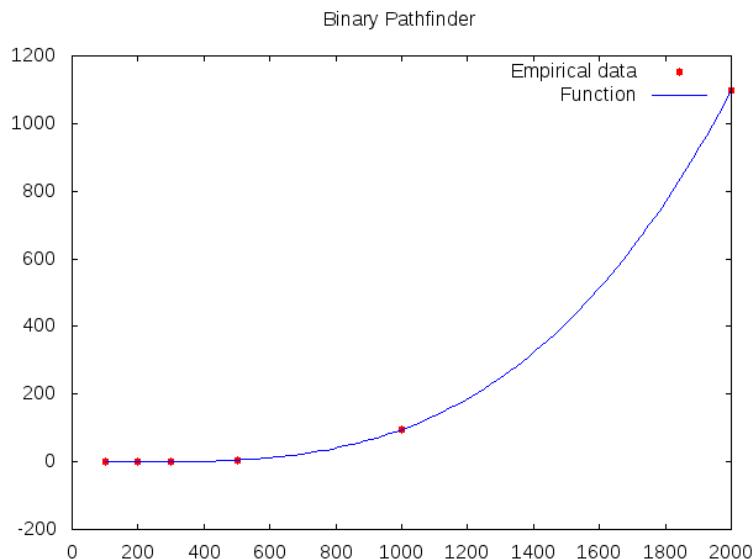


Figura 16: Ejecución del algoritmo *Pathfinder* binario y su función ajustada.

La función correspondiente es:

$$f(x) = 2,84515 \cdot 10^{-8} \cdot x^3 \cdot \log(x) - 0,000133786 \cdot x^2 + 0,0356842 \cdot x - 2,43701$$

2.1.3. *Fast Pathfinder* ($O(n^3)$)

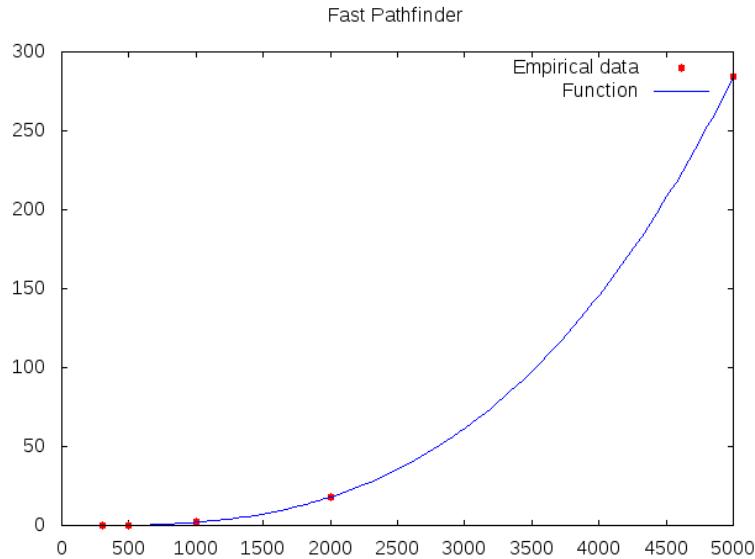


Figura 17: Ejecución del algoritmo *Fast Pathfinder* y su función ajustada.

La función correspondiente es:

$$f(x) = 2,19893 \cdot 10^{-9} \cdot x^3 + 5,58985 \cdot 10^{-7} \cdot x^2 - 0,00100767 \cdot x + 0,197532$$

2.1.4. *MST Pathfinder* teórico ($O(n^2 \log n)$)

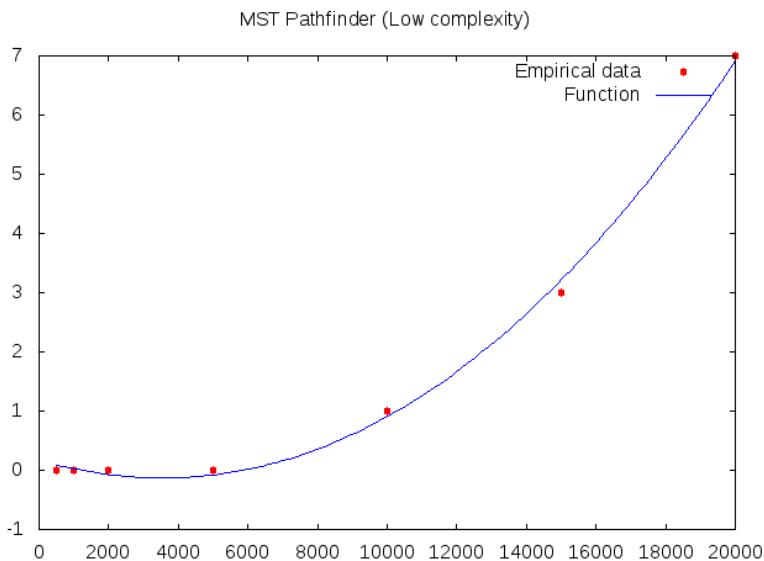


Figura 18: Ejecución del algoritmo *MST Pathfinder* (versión teórica) y su función ajustada.

La función correspondiente es:

$$f(x) = 1,64944 \cdot 10^{-9} \cdot x^2 \cdot \log(x) + 5,58897 \cdot 10^{-7} \cdot x - 0,00100767$$

2.1.5. *MST Pathfinder* práctico ($O(n^3)$)

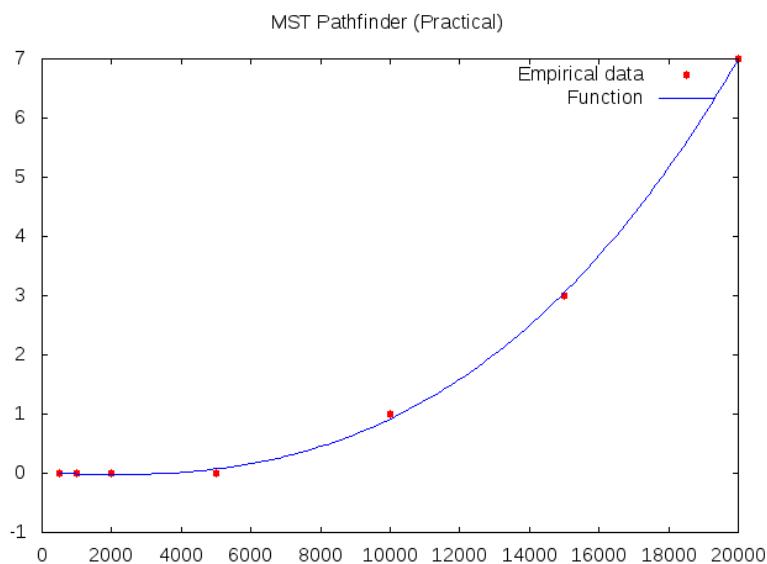


Figura 19: Ejecución del algoritmo *MST Pathfinder* (versión práctica) y su función ajustada.

La función correspondiente es:

$$f(x) = 6,76096 \cdot 10^{-13} \cdot x^3 + 5,63221 \cdot 10^{-9} \cdot x^2 - 3,45822 \cdot 10^{-5} \cdot x + 0,0190147$$

2.2. Comparativa final

Como hemos podido observar en los apartados dedicados al análisis híbrido de cada variante del algoritmo *Pathfinder*, aunque las gráficas parezcan iguales a simple vista, cada curva es distinta a las demás y se ajusta bien a sus correspondientes datos, alcanzando valores de tiempo de ejecución muy distintos de una variante a otra (salvo el caso de los algoritmos *MST*). Veamos ahora la magnitud del cambio que suponen distintos órdenes de eficiencia:

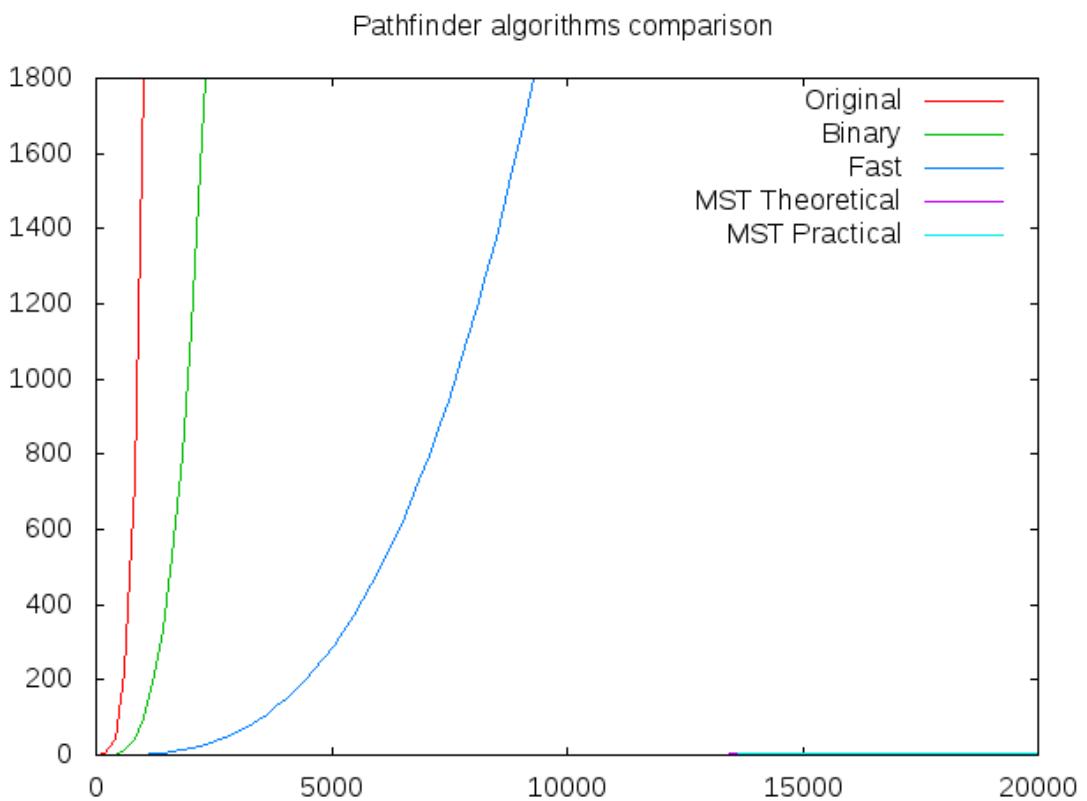


Figura 20: Comparativa de las curvas descritas por los distintos algoritmos.

Tal y como era de esperar, existe una distancia considerable entre los diferentes órdenes de complejidad. En especial, podemos ver cómo difieren los algoritmos basados en programación dinámica de los basados en *Greedy*: tanto que las curvas de estos últimos son prácticamente paralelas al eje horizontal.