Teoría avanzada de la computación

Algoritmos aplicados a Optimización Combinatoria
Entrega 1

Grado en Ingeniería Informática - 4.º Curso

Campus de Leganés Grupo 83

Carlos Gallego Jiménez

Carlos Rubio Olivares

Jorge Rodríguez Fraile

Índice

1. Introducción

El objetivo de esta práctica es estudiar la complejidad y optimización de problemas de clase NP-C. Estos problemas están relacionados con el uso de grafos:

- Conteo de los clusters de un grafo: Contar el n.º de subgrafos encontrados en un grafo G
- Problema de k-colorabilidad: Dado un grafo G y un número de colores k, asignar a cada nodo del grafo un color de manera que sus nodos adyacentes tengan un color distinto.
- Problema k-clique: Conteo de los subgrafos completos de k nodos encontrados en un grafo G.
- Problema máx.-clique: Dado un grafo G, determinar el valor máximo de k que permite hallar una solución al problema k-clique.

Para cada uno de estos problemas se hará un estudio tanto analítico como empírico para conocer sus tendencias y comportamiento. Además, será necesario establecer los peores casos que se pueden encontrar en los algoritmos que los resuelven y por qué son estos. Cualquier punto o idea que se establezca durante este trabajo será argumentado y estará apoyado en la bibliografía utilizada.

2. Ejercicio 1

a. Estudio analítico

El estudio analítico de este problema se hará explicando el proceso realizado en la función de propagación de marca del código. Esta función se llama desde la función principal del programa, *contar_clusters*. Esto implica que se deba comprobar si un nodo no ha sido marcado y en dicho caso, llamar a *propagar_marca*.

En una situación extrema (grafo completamente inconexo, por ejemplo), el programa hará tantas llamadas de *propagar_marca* como nodos haya*.* Esto se deriva en ejecutar el bucle que encontramos en *propagar_marca* tantas veces como nodos encontremos en el grafo, aunque en este caso, como no se encuentran conexiones, no se efectúa ningún tipo de recursividad.

Podemos ver, por tanto, dos procesos anidados en los que se recorren todos los nodos, por lo que cada uno tiene complejidad *O(n),* lo que da como resultado una complejidad total de $O(n^2)$, qué es lo que debemos esperar ver en los estudios empíricos que se realicen en secciones posteriores. Este tipo de análisis podemos extrapolarlo a cualquier tipo de grafo donde existiera la recursividad, ya que la complejidad se mantiene constante para diferentes grafos.

A continuación, se presenta la justificación de estos razonamientos mediante diferencias finitas, en la que se cuentan las operaciones elementales hechas. Se ejecutan sucesivas llamadas de *contar_clusters*, desde 1 nodo hasta 10 nodos, en primer lugar, para uno completamente conexo y después con uno totalmente inconexo.

Nodos (n)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Operaciones Elementales	**25**	**55**	**97**	**151**	**217**	**295**	**385**	**487**	**601**	**727**
Dif. 1		30	42	54	66	78	90	102	114	126
Dif. 2			12	12	12	12	12	12	12	12
Dif. 3				0	0	0	0	0	0	0

Tabla 1. Diferencias finitas para el algoritmo base de conteo de clusters para grafos completos desde 1 hasta 10 nodos.

Nodos (n)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Operaciones Elementales	**25**	**57**	**101**	**157**	**225**	**305**	**397**	**501**	**617**	**745**
Dif. 1		32	44	56	68	80	92	104	116	128
Dif. 2			12	12	12	12	12	12	12	12
Dif. 3				0	0	0	0	0	0	0

Tabla 2. Diferencias finitas para el algoritmo base de conteo de clústers para grafos completamente inconexos desde 1 hasta 10 nodos.

Se puede apreciar en ambas tablas que las ecuaciones asociadas se corresponderán a una ecuación de orden 2 ($T(n) = xn^2 + yn + z$), dado que en Dif. 3 se vuelven 0 las diferencias. Sus resoluciones son las siguientes:

$$\left(\begin{array}{ccc|c} 1 & 1 & 1 & 25 \\ 4 & 2 & 1 & 55 \\ 9 & 3 & 1 & 97 \end{array}\right);\left(\begin{array}{ccc|c} 1 & 1 & 1 & 25 \\ 3 & 1 & 0 & 30 \\ 8 & 2 & 0 & 72 \end{array}\right);\left(\begin{array}{ccc|c} 1 & 1 & 1 & 25 \\ 3 & 1 & 0 & 30 \\ 2 & 0 & 0 & 12 \end{array}\right);\left(\begin{array}{ccc|c} 0 & 1 & 1 & 19 \\ 0 & 1 & 0 & 12 \\ 1 & 0 & 0 & 6 \end{array}\right);\left(\begin{array}{ccc|c} 0 & 0 & 1 & 7 \\ 0 & 1 & 0 & 12 \\ 1 & 0 & 0 & 6 \end{array}\right);\begin{array}{r} x = 6 \\ y = 11 \\ z = 7 \end{array}$$

$$\left(\begin{array}{ccc|c} 1 & 1 & 1 & 25 \\ 4 & 2 & 1 & 57 \\ 9 & 3 & 1 & 101 \end{array}\right);\left(\begin{array}{ccc|c} 1 & 1 & 1 & 25 \\ 3 & 1 & 0 & 32 \\ 8 & 2 & 0 & 76 \end{array}\right);\left(\begin{array}{ccc|c} 1 & 1 & 1 & 25 \\ 3 & 1 & 0 & 32 \\ 2 & 0 & 0 & 12 \end{array}\right);\left(\begin{array}{ccc|c} 0 & 1 & 1 & 19 \\ 0 & 1 & 0 & 14 \\ 1 & 0 & 0 & 6 \end{array}\right);\left(\begin{array}{ccc|c} 0 & 0 & 1 & 5 \\ 0 & 1 & 0 & 14 \\ 1 & 0 & 0 & 6 \end{array}\right);\begin{array}{r} x = 6 \\ y = 14 \\ z = 5 \end{array}$$

Que dan lugar a las ecuaciones: $T(n) = 6n^2 + 12n + 7$ y $T(n) = 6n^2 + 14n + 5$. Ambas T(n) serán menores o iguales que $c * g(n)$, para $\forall n \geq n_0$ con $g(n) = n^2, n_0 = 1$ y $c = 20$. Como se cumple dicha condición, la complejidad queda demostrada ser $O(g(n)) = O(n^2)$, para ambos casos extremos.

b. Estudio empírico

Durante esta sección se expondrán las gráficas relacionadas con el estudio empírico del problema de conteo de clústers. Los gráficos mostrarán en el eje horizontal el n.º de nodos del grafo, mientras que en el eje vertical se mostrará el tiempo de ejecución asociado.

Para ejecutar estas pruebas, se ha realizado una modificación del código. Se crea una función denominada *prueba_empírico* a la que se le pasa un n.º de nodos. Hecho esto, se genera un grafo con $\alpha = n/2$ aristas, siendo n el número máximo de arcos ($n(n-1)/2$), ya que según la bibliografía [X] en este punto hay un cambio de fase considerable en la ejecución de este algoritmo. Una vez creado el grafo con estas características se llama a la función

contar_clusters y se inicia una variable del tipo *clock_t* para medir el tiempo pertinente a la ejecución.

Los gráficos mostrarán una serie de ejecuciones utilizando este método, yendo desde 0 a 20.000, de 10 en 10 nodos. Junto a esta gráfica se mostrará su conversión en escala logarítmica para poder estudiar su comportamiento de manera más precisa.

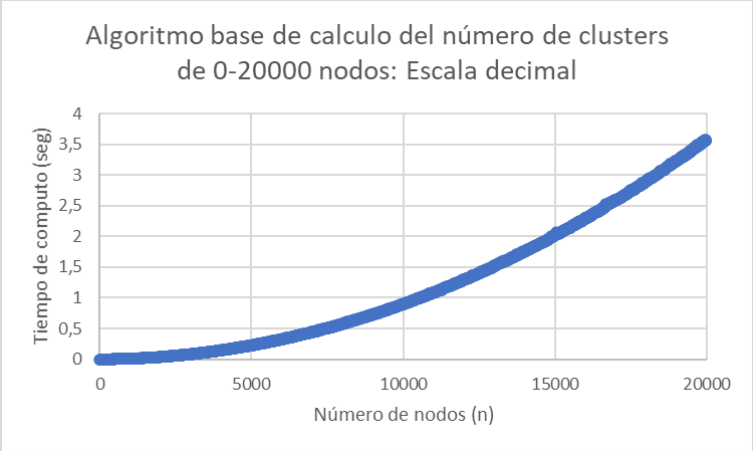

Figura 1. Tiempo de cómputo del algoritmo base en el problema de conteo de clusters en escala decimal.

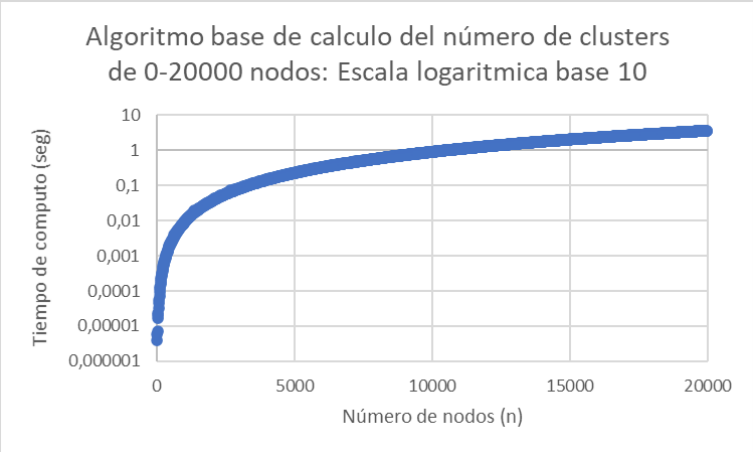

Figura 2. Tiempo de cómputo del algoritmo base en el problema de conteo de clusters en escala logarítmica.

Podemos ver una clara tendencia polinómica (aproximadamente de n^2) en la figura 1. Esta idea puede ser argumentada debido a que, al pasar los resultados a escala logarítmica, en la figura 2, se sigue una tendencia logarítmica. Todo este planteamiento empírico es compatible con lo que se ha planteado en el estudio analítico del algoritmo.

c. Circunstancias para el peor caso

Para estudiar los peores casos que se pueden dar en este problema, se opta por un método parecido al de la sección anterior, mediante casos empíricos. De esta manera, se vuelve a modificar el comportamiento del código.

En este caso, se crea un grafo con el número máximo de aristas posibles, es decir, cada nodo tendrá una arista al resto de los nodos creando así un grafo completo. Una vez hecho esto se llama a la función *contar_clusters* para calcular su tiempo de ejecución. Hecho esto, se eliminará una arista y se volverá a repetir el proceso explicado anteriormente. La función principalmente parte de un grafo completo hasta uno completamente inconexo y mide el tiempo de ejecución de cada iteración.

En las gráficas mostradas en esta sección aparecerán en el eje horizontal el número de arcos o aristas y en el eje y el tiempo de cómputo. Se espera que las gráficas apoyen lo descrito en la bibliografía [X], y que el cambio de fase se encuentre cerca de los valores de $\alpha = n/2$ con n como el n.º máximo de aristas.

Cabe recalcar que se realizan pruebas sobre grafos con diferentes tamaños de nodos para tener un conjunto de datos de mayor calidad.

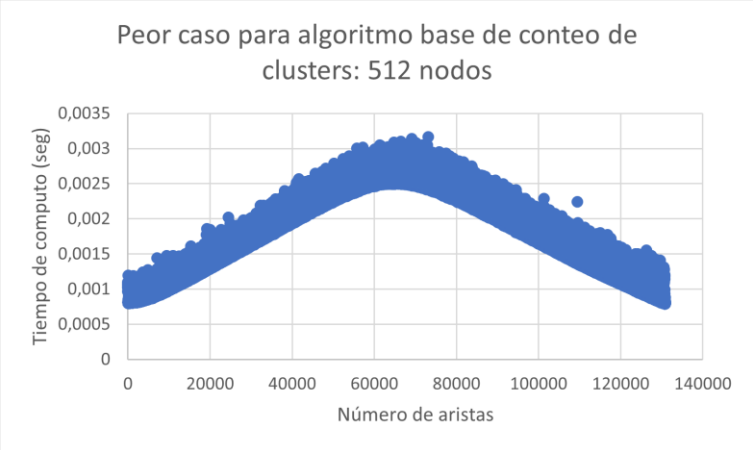

Figura 3. Estudio empírico del peor caso en el problema de conteo de clusters para 512 nodos.

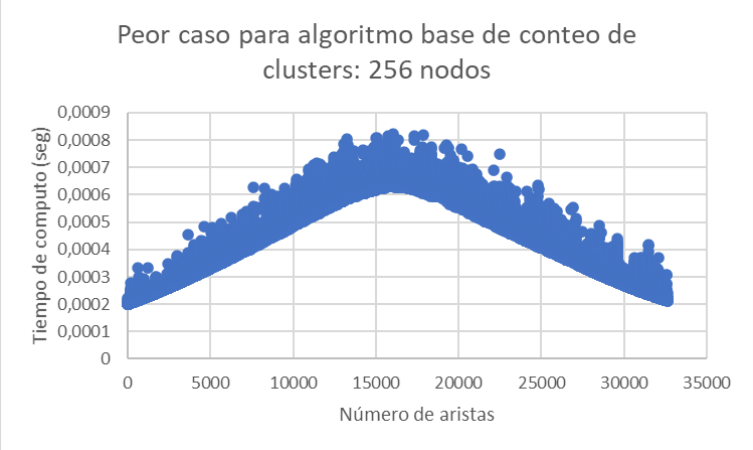

Figura 4. Estudio empírico del peor caso en el problema de conteo de clusters para 256 nodos.

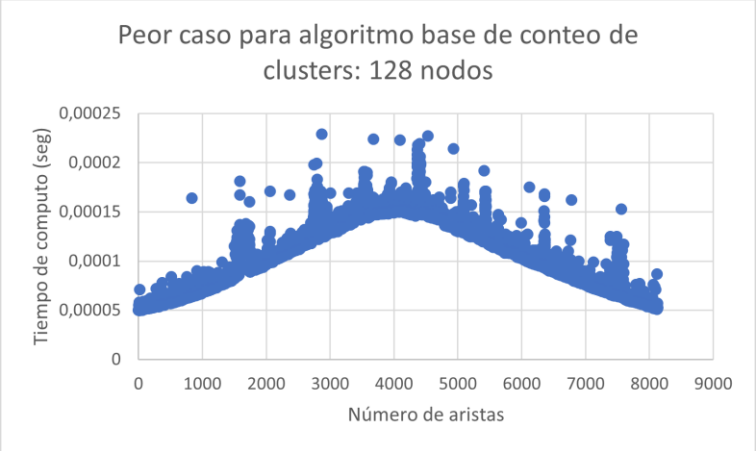

Figura 5. Estudio empírico del peor caso en el problema de conteo de clusters para 128 nodos.

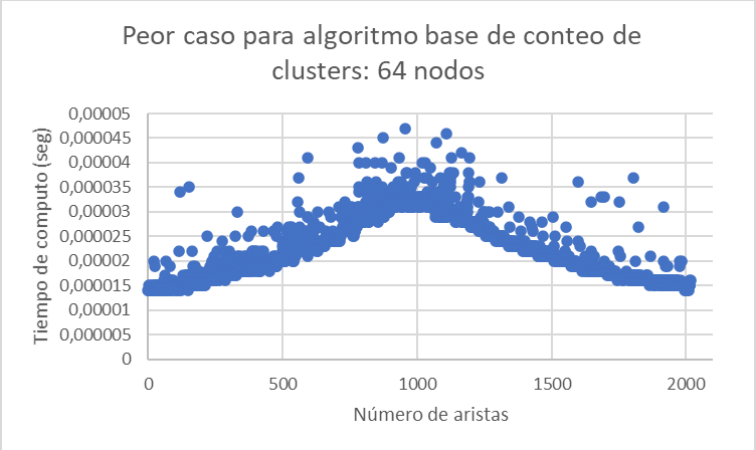

Figura 6. Estudio empírico del peor caso en el problema de conteo de clusters para 64 nodos.

Podemos ver qué tal y como se había comentado, hay un aumento considerable del tiempo de cómputo en valores cercanos a $\alpha = n/2$. Esto se puede ver de manera mucho más clara en el grafo de 256 y 512 nodos, ya que valores cercanos al epicentro de la gráfica tienden a tener un tiempo de ejecución mayor. Las pruebas empíricas realizadas en esta sección apoyan, por tanto, a lo planteado en la bibliografía [X].

d. Mejora de la eficiencia

Para empezar, se estudió la posibilidad de, en lugar de hacer una búsqueda en profundidad, emplear el algoritmo de Warshall que se menciona en el enunciado.

Se implementó y probó esta alternativa a pesar de tener una clara complejidad de $O(n^3)$, siendo n el número de nodos del grafo. Este algoritmo lo que permitía era de una manera rápida saber desde un mismo nodo todos los nodos a los que se podría conectar e identificar más directamente los clusters, pero el elevado coste hizo que se desechara esta alternativa.

A continuación, se muestra una comparativa de Warshall y el algoritmo dado para los 1700 primeros valores de nodos, con las aristas del peor caso. La primera representa el tiempo obtenido y en la segunda se aplica el logaritmo base 10 para ver claramente que es de un orden mayor de complejidad polinómica.

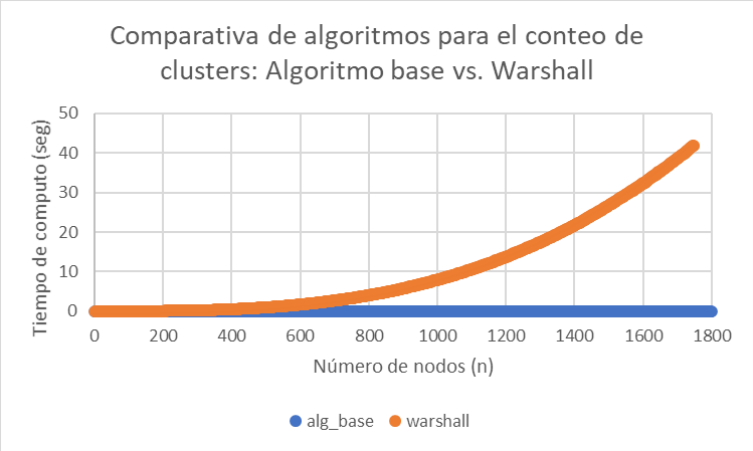

Figura 7. Estudio de mejora para el algoritmo de conteo de clusters base mediante el Algoritmo de Warshall.

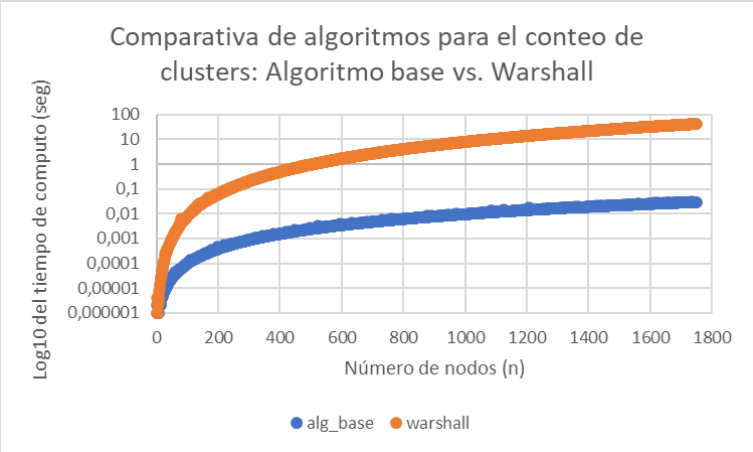

Figura 8. Estudio de mejora para el algoritmo de conteo de clusters base mediante el Algoritmo de Warshall empleando escala logarítmica.

Como se puede ver, los resultados son claramente peores, por lo que se pasa a probar la búsqueda en amplitud, visto que el algoritmo base que lo hace en profundidad da buenos resultados, es otra buena aproximación al problema.

Se ha implementado BFS (Breadth-first search) mediante una cola de nodos, en la que se van introduciendo sucesivamente los nodos con los que está conectado el nodo y sucesivamente se van sacando nodos de esta cola para ir visitándolos y marcándolo como conectados a dicho clúster. Una vez hemos visitado todos los nodos que hemos podido incluir en la cola, se pasa a revisar si queda algún nodo más por visitar. Si es así, este pertenecerá a otro clúster y se repite el proceso de ir metiendo nodos y visitándolos. Cuando se han marcado todos los nodos como visitados, nuestro contador de nodos nos devolverá cuántas veces hemos tenido que elegir un nuevo nodo para hacer esta expansión en amplitud.

Al igual que pasaba con la búsqueda en profundidad, la búsqueda en amplitud tiene una complejidad de $O(n^2)$, con n siendo el número de nodos del grafo.

Se muestra a continuación unas gráficas de comparación entre el algoritmo base y BFS para una serie de pruebas, tiempo de cómputo para sucesivos números de nodos para el peor

caso, van desde 0 hasta 20000 nodos con saltos de 10 en 10. Primero tiempo real y después, el logaritmo base 10.

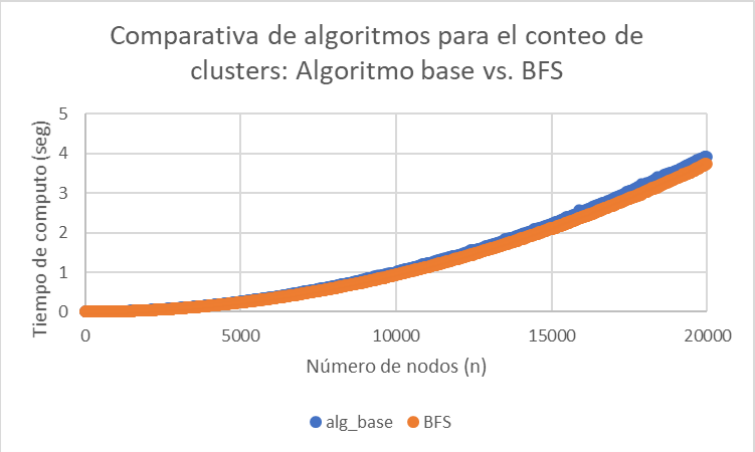

Figura 9. Estudio de mejora para el algoritmo de conteo de clusters base mediante búsqueda en profundidad.

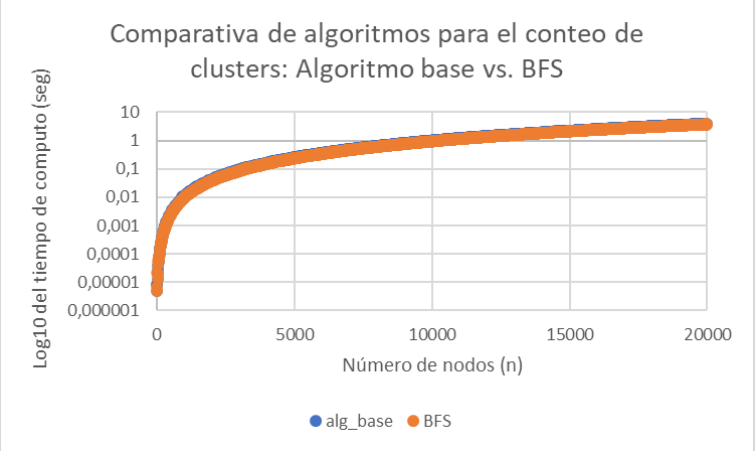

Figura 10. Estudio de mejora para el algoritmo de conteo de clusters base mediante búsqueda en profundidad empleando escala logarítmica.

Como podemos ver en la gráfica, los resultados son prácticamente idénticos, la mejora que supone la búsqueda en amplitud es de un 6,27 % de media sobre el algoritmo base. Para ver mejor esta diferencia en la siguiente gráfica se muestra cuántos segundos de diferencia hay del algoritmo base respecto a BFS.

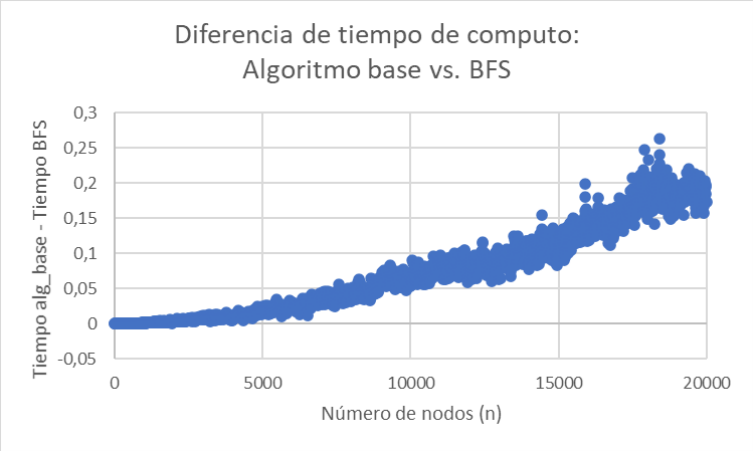

Figura 11. Mejora de tiempos de búsqueda en profundidad respecto al algoritmo base de conteo de clusters.

Esta similitud ya se podía intuir desde un principio conociendo la complejidad de ambos algoritmos, aunque se refuerza con la gráfica logarítmica al tomar la misma forma. La implementación de este último algoritmo es algo mejor (≈6 %), pero nos hace ver que la implementación del algoritmo base proporciona resultados acordes a otros de su misma complejidad.

De todos los probados, los más destacables son el algoritmo base y BFS, aunque el de Warshall se mueve en el rango de las complejidades esperadas, entre cuadrado y cúbico.

3. Ejercicio 2. K-Colorabilidad k=3

a. Nuevo método

b. Tamaño grafo para tiempo razonable

El número de nodos que debe tener un grafo para poder encontrar todas las soluciones de su colorabilidad para k=3 en un tiempo razonable es 20. En la gráfica se mostrará en el eje x el número de nodos de un grafo y en el eje y el tiempo de cómputo en segundos. Se añade también una línea que delimita la una hora de ejecución, para entender la magnitud de los cómputos que se muestran.

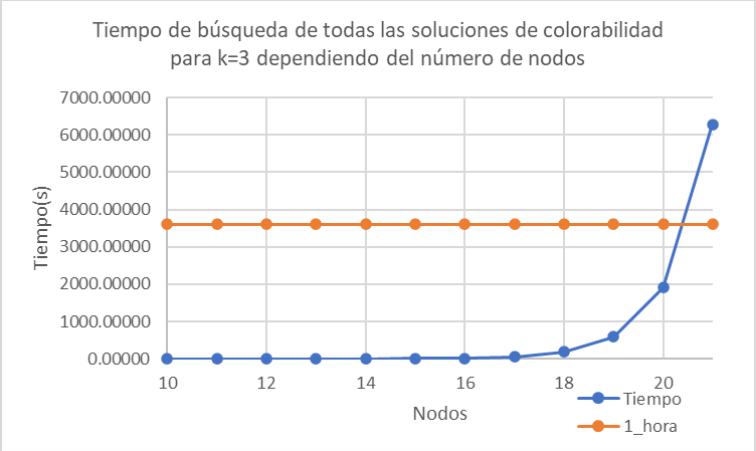

Figura 12. Tiempo de búsqueda de todas las soluciones colorabilidad con k=3, de 10 a 21 nodos.

Se puede observar cómo hay un aumento considerable de tiempo de cómputo para grafos con nodos mayores a 20 nodos. De esta manera, el dominio aceptable para este problema de k-colorabilidad debería encontrarse entre los 1 y 20 nodos.

c. Estudio analítico

d. Estudio empírico

e. …

4. Ejercicio 3. K-Colorabilidad k=2

a. Nuevo método

b. Tamaño grafo para tiempo razonable

c. Estudio analítico

d. Estudio empírico

5. Ejercicio 4. Reducción K-Colorabilidad al de Clique
   a. Nuevo método

   b. Estudio analítico

   c. Estudio empírico

6. Ejercicio 5. Estudio teórico
   a. Clase K-Coloreado y K-Clique

   b. Implicaciones

7. Conclusión