Ingeniería algorítmica
Armonización de patrones

Anna Szpoton, Rafał Tulwin

June 15, 2011

1 Introducción

El objetivo del trabajo fue implementar un programa que busque patrones de letras en un texto y devuelva el numero de patrones encontrados junto con el tiempo de procesamiento. El programa debió servir para comparar distintos algoritmos de armonización de patrones asi que implementar y añadir nuevos algoritmos al programa debería ser fácil y rápido. Un objetivo muy importante fue también mostrar los resultados en los grafos presentados en el interfaz gráfico, para que el usuario pueda ver las diferencias entre los algoritmos comparados.

La última parte del trabajo contiene los resultados de experimentos hechos con distintos tipos de algoritmos implementados y ficheros del texto de tamaños más pequeños y más grandes. Generalmente, es una comparación de eficiencia que tiene cada uno de los algoritmos dado varios patrones, tamaños y tipos de texto.

2 El programa

El programa está implementado en C++, usando la biblioteca QT, que permite fácilmente crear un interfaz gráfico. El lenguaje C++, pareció el mejor para la tarea, con sus grandes poderes de controlar la alocación de memoria.

El interfaz gráfico del programa permite al usuario cargar distintos ficheros del texto, introducir un patrón para buscar en el texto, elegir el algoritmo y ver los resultados en dos tipos de grafos. En cada momento, el usuario puede borrar los resultados de la lista y empezar todo de nuevo.

La figura 1 presenta la ventana del programa con el grafo de tipo histograma. Este tipo de grafo presenta los tiempos de procesamiento obtenidos en las diez ultimas pruebas. El usuario puede cambiar el algoritmo, el patrón, o cargar un nuevo fichero en cada nueva prueba. La legenda que está en la parte derecha de la ventana contiene información sobre el algoritmo elegido para cada de las pruebas junto con el tamaño de patrón buscado y el tamaño del fichero del texto. El grafo se cambia con cada nueva prueba. La prueba con el tiempo de procesamiento más largo de todas presentadas siempre corresponde con la barra con máximo posible tamaño.

La figura 2 presenta parte de la ventana con el grafo de otro tipo. Este tipo de grafo, presenta el tiempo de procesamiento dependiente del tamaño del texto y el patrón buscado. Cada uno de los algoritmos usados tiene su propia marca asignada. Además, la legenda enseña cual marca corresponde con cada de los algoritmos.

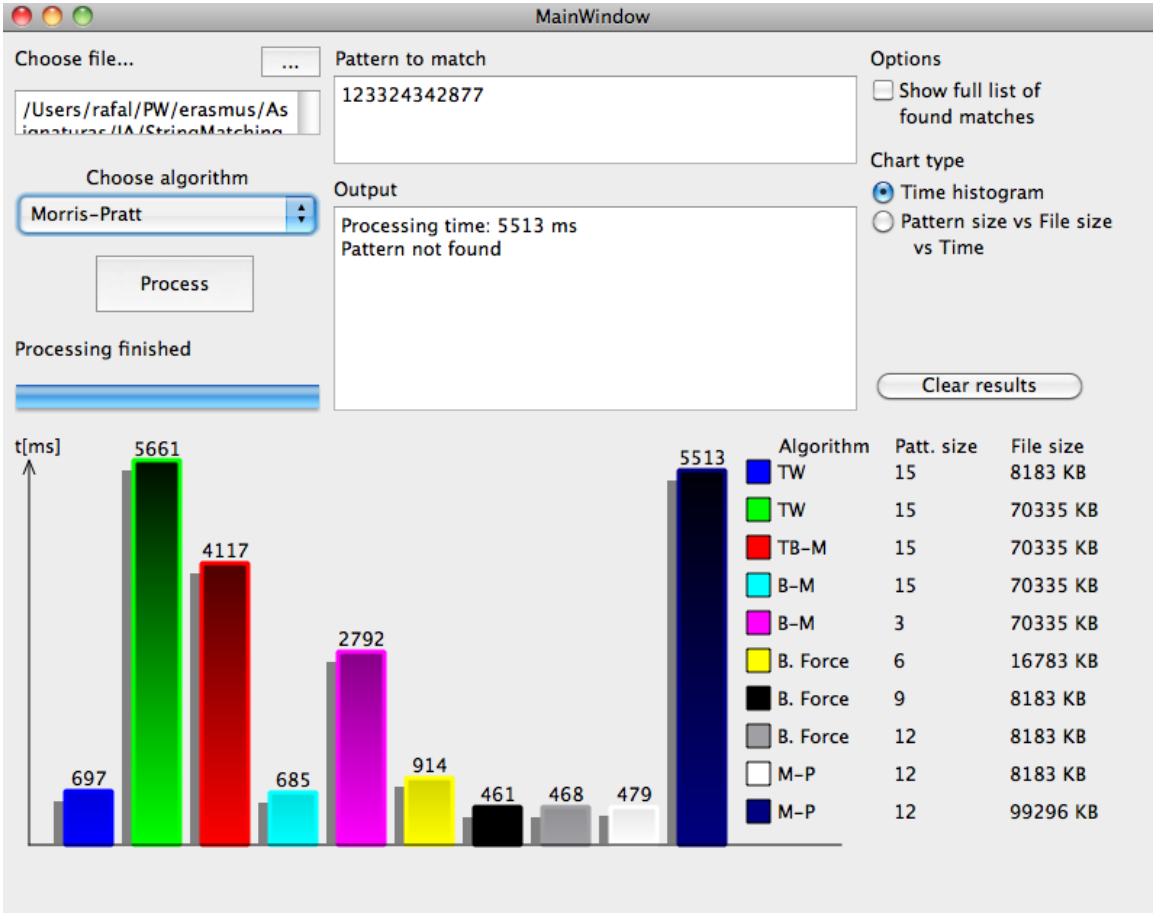

Figure 1: La ventana del programa con grafo del tipo histograma

El tiempo está presentado como el color de la marca correspondiente del algoritmo usado en la prueba. Los colores cambian desde azul (el tiempo mas corto) hasta rojo (el tiempo mas largo). Añadir nuevas pruebas causa el cambio del grafo. La escala de los ejes se ajusta automáticamente para poder presentar cada uno de nuevos valores que aparezcan. Lo mismo pasa con los colores. El más rojo, siempre marca el tiempo más largo de las pruebas y el más azul, el tiempo más corto.

Los dos tipos de grafos usan la misma lista de resultados. Gracias a esto, el usuario puede cambiar el tipo de grafo visible cuando quiera. La diferencia entre los dos es que el grafo de tipo histograma presenta solo los diez últimos resultados dando todos detalles de las pruebas presentadas. El grafo de segundo tipo presenta todos los resultados en la lista, pero no da exactamente el valor del tiempo de procesamiento, sino lo presenta en forma de color, relativo a otras pruebas. El problema con este tipo de grafo aparece cuando se hace pruebas con distintos tipos de los algoritmos y exactamente el mismo tamaño de el patrón y el fichero. En este caso las marcas están pintadas en el mismo sitio en el área del grafo y la información no está clara.

Distintos tipos de grafos permiten investigar resultados de varias pruebas en distintas

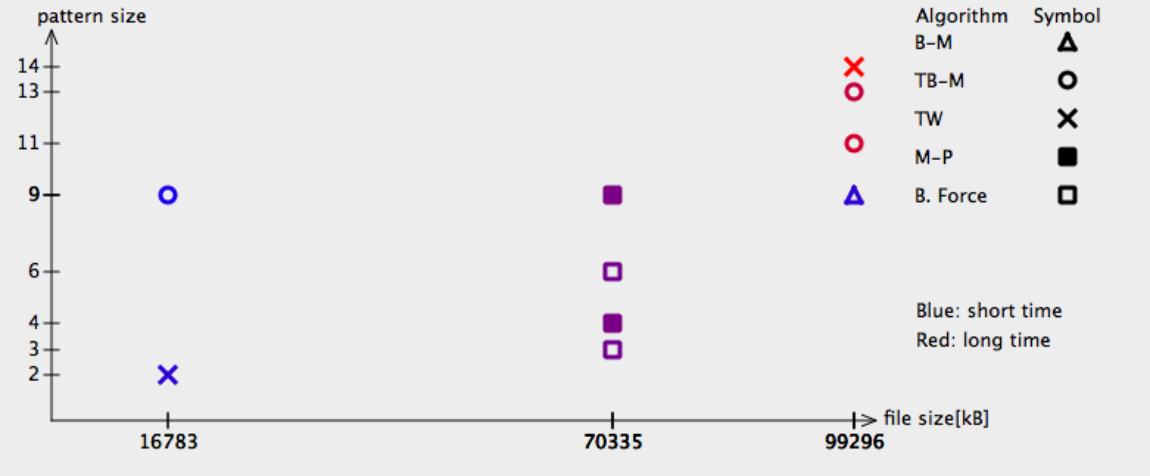

Figure 2: La parte de la ventana del programa con grafo del segundo tipo

maneras. La posibilidad de ver los mismos resultados en dos tipos de grafos permite el usuario tener la información completa. El histograma sirve mejor para comparar los algoritmos usando el mismo fichero del texto y el mismo patrón. El segundo tipo de grafo funciona mejor cuando se compara el mismo algoritmo con distintos tamaños de patrón o fichero.

3 Algoritmos

La implementación corriente, contiene cuatro algoritmos implementados. Se supone, que la fuerza bruta, como el algoritmo mas sencillo, debe ser el peor y mas lento de todos. Se lo trata como un punto de referencia para otros algoritmos.

3.1 Fuerza bruta

El algoritmo de la fuerza bruta hace los n-m bucles, donde n, es el número de las letras en el fichero y m es el número de las letras en el patrón. El algoritmo en cada bucle hace otra bucle que comprueba los caracteres del fichero con el patrón. Si algún carácter no es igual que en el patrón la bucle dentro se termina y la comparación se empieza desde principio con el próximo carácter del fichero.

3.2 Morris Pratt

El algoritmo del Morris Pratt es el otro modo para encontrar el patrón. En primer paso hace preprocesamiento: eso es la preparación de una tabla con valores que indiquen cuando tenemos el mismo carácter en principio y dentro de la cadena de las letras en el patrón. Eso es importante cuando el algoritmo encuentre desigualdad en un carácter. En caso de empezar la búsqueda desde principio podemos seguir desde el lugar donde se encuentra un prefijo igual. En el proceso de búsqueda se comprueba los caracteres del patrón y lugar de búsqueda. Si los caracteres en el patrón y el fichero no son iguales se busca en la tabla de preprocesamiento el próximo carácter en el patrón comprobado.

3.3 Boyer Moore

Este algoritmo es conocido como el mas rápido y eficiente en aplicaciones usuales. La implementación sencilla de este algoritmo esta usada con mucha frecuencia en las editores del texto donde se usa mucho la búsqueda y sustitución de las cadenas de caracteres. En la fase de preprocesamiento se busca dos tablas. Primera tabla se llama tabla de mal carácter. Contiene el numero de cambios a la izquierda en la cadena del patrón para recibir el carácter adecuado. La segunda tabla es la tabla de buenos sufijos. Esta tabla contiene el numero de las cambias para mover la cadena del patrón a cada uno de sufijos. Durante la búsqueda el algoritmo comprueba los caracteres del patrón con el fichero y si encuentra el patrón usa la tabla de los sufijos para empezar con el próximo sufijo. En contrario si encuentra alguna desigualdad, empieza la búsqueda usando el máximo de las tablas con mal carácter y el próximo sufijo con los parámetros del numero del carácter.

3.4 Two Way

Two Way es el algoritmo que en la fase de preprocesamiento factóriza el patrón. Se recibe dos cadenas del patrón: la cadena derecha y la cadena izquierda. La mayoría de preprocesamiento es elegir buen modo de factorización. El algoritmo comprueba los caracteres del fichero y la cadena derecha. En caso de igualdad con toda la cadena a la derecha empieza la comprobación con la cadena izquierda desde su último carácter. Si hay alguna desigualdad durante la comprobación de la cadena a la derecha se hace cambio a tantas posiciones cuanto es el número del carácter en la cadena. Cuando en la cadena izquierda se va a la derecha al número de caracteres del máximo sufijo.

4 Experimentación

Los experimentos fueron hechos con dos tipos de textos. El primer tipo de texto contiene una cadena de cifras. El segundo texto es un libro copiado muchas veces para obtener los tamaños queridos de los ficheros. Para ver la diferencia de resultados obtenidos con ficheros del distinto tamaño, cuatro ficheros de cada tipo del texto fueron creados. Tamaños de los ficheros son (aproximadamente): 102 mb, 72 mb, 17 mb, 8 mb. Juntos, son 8 ficheros del texto.

Los patronos fueron preparados como partes de los ficheros. Para observar la diferencia del tamaño del patrón para cada tipo de fichero (fichero con texto y con números) 6 distintos patrones fueron creados: 1, 4, 51, 504, 2232 y 4140 caracteres.

Al principio, para cada tamaño del fichero se realizó pruebas con todos distintos patrones. Entonces juntos, son 48 pruebas. Los resultados están presentados en los grafos de tipo histograma.

Luego, se hizo pruebas con distintos tamaños de los ficheros y de los patrones para cada uno de los algoritmos, para ver como se cambia el tiempo de procesamiento con el cambio del tamaño de patrón y fichero. Los resultados están presentados en el grafo de segundo tipo (un grafo para un algoritmo y un fichero). En total, son 8 grafos (2 ficheros, 4 algoritmos).

Los resultados están incluidos en la última parte de la memoria.

Que se puede observar muy fácil es que con el algoritmo de fuerza bruta para un tamaño de fichero se recibe los mismos resultados del tiempo con todos los patrones usados. Entonces considerar los tiempos de procesamiento no importa el patrón, solo el tamaño del fichero. Es lo que se puede inferir del código del algoritmo. El algoritmo comprueba el patrón n-m veces donde n es el número de los caracteres en fichero y m numero de los caracteres en el patrón.

El algoritmo que parece lo más interesante de su disimilitud, Two Way, finalmente da los peores resultados. Sus búsquedas son las más largas, aún más largas que estas de la fuerza bruta. Probablemente, es la cosa de mal división del patrón de que depende la calidad de la búsqueda.

Morris-Pratt es básicamente el algoritmo de fuerza bruta con el preprocesamiento. En nuestros pruebas tampoco dio la aceleración de búsqueda. Quizás, esto es el razón por lo cual se usa con más frecuencia la versión mejor de este algoritmo - Knuth Morris Pratt, sobre que hay muchos más publicaciones. La estructura del algoritmo deja suponer que algoritmo se puede comportar mejor con los patrones con muchas repeticiones del primer carácter.

Los mejores resultados se obtiene del algoritmo Boyer-Moore. Tiene el preprocesamiento más complicado y usa dos tablas auxiliares que se usa depende de sufijo y carácter. Eso ahorra los pasos de comparación de las cadenas. Boyer-Moore parece el mejor, más rápido y no hay ninguna duda con esto, porque está usado en todos los editores del texto que usan el mecanismo de busca/cambia.

5 Ideas para el futuro

Este parte de la memoria indica las ideas para el posible desarollo del programa en el futuro. La implementación corriente tiene toda la funcionalidad básica, necesaria para proveer la oportunidad de comparar los algoritmos de armonización de patrones en una manera sencilla. Añadir una nueva funcionalidad podría extender la potencia del programa.

Una buena idea sea implementar la posibilidad para guardar los resultados en un fichero de tipo xml, para poder cargarlos y verlos otra vez. También, una funcionalidad que permita hacer muchas pruebas con muchos ficheros del texto automáticamente podría ser muy útil.

El programa deja la oportunidad para implementar nuevos tipos de grafos, para presentar los datos en otras maneras. Los tipos de grafos que faltan son los que puedan enseñar el tiempo de procesamiento en una de los ejes, como función del tamaño del fichero o el patrón.

6 Conclusiones

Mirando los resultados obtenidos se puede concluir, que la Fuerza Bruta no es un algoritmo malo para este tipo de tarea, aunque es el mas sencillo. Para los patrones muy cortos es el algoritmo mejor de todos comprobados. En realidad, solo el algoritmo Boyer-Moore,

da un aumento de eficiencia significante, y por eso está usado en muchas aplicaciones comerciales.

7 Resultados de experimentación

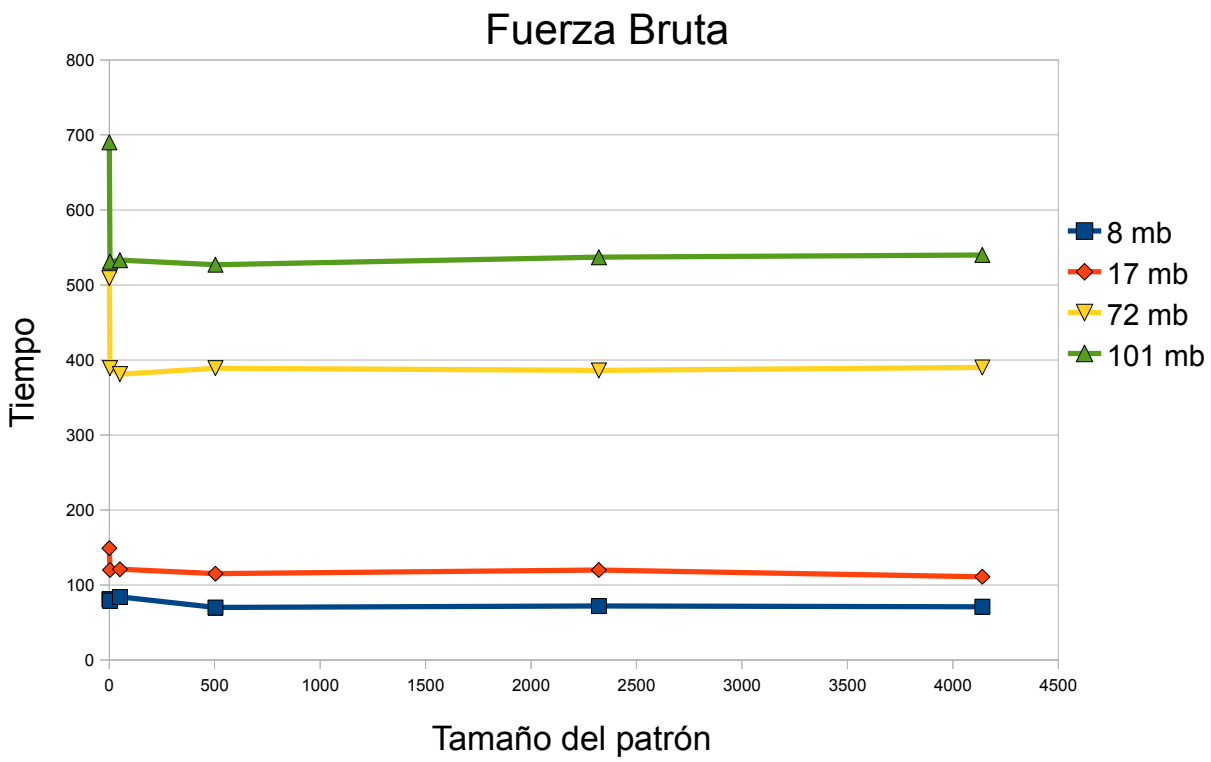

Figure 3: Resultados obtenidos con el algoritmo de Fuerza Bruta

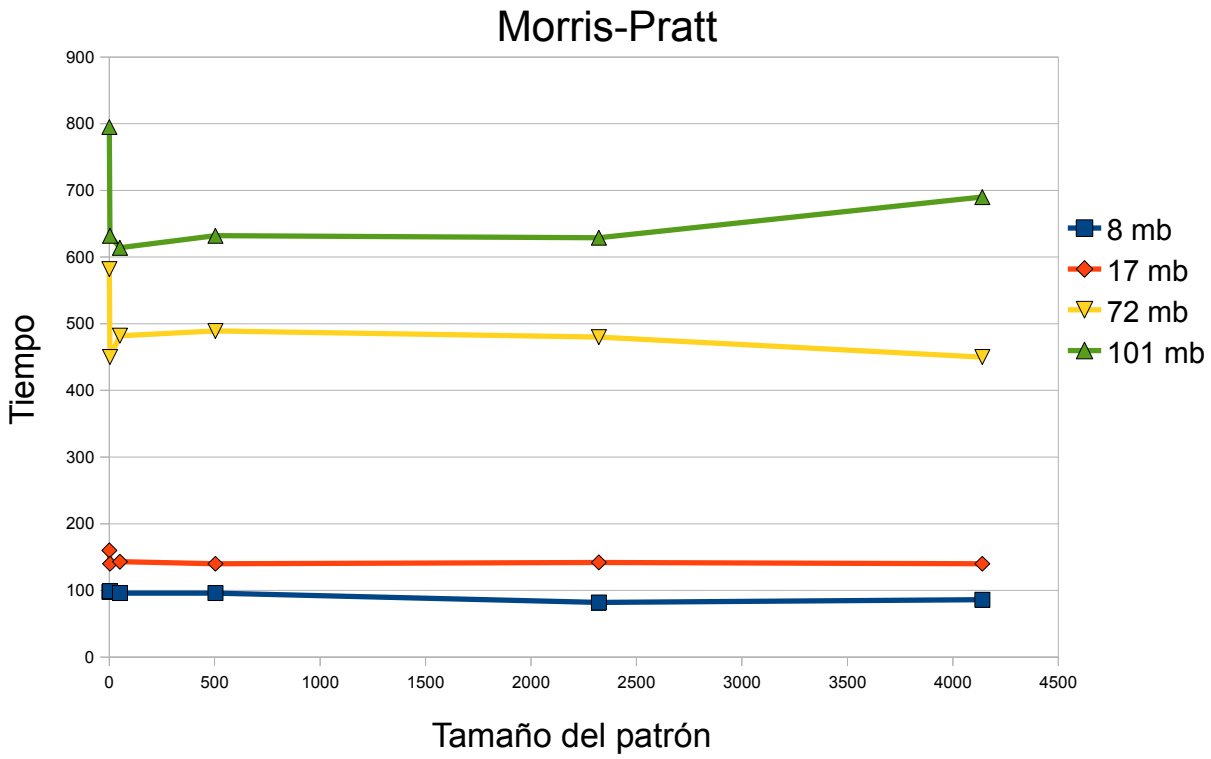

Figure 4: Resultados obtenidos con el algoritmo de Morris-Pratt

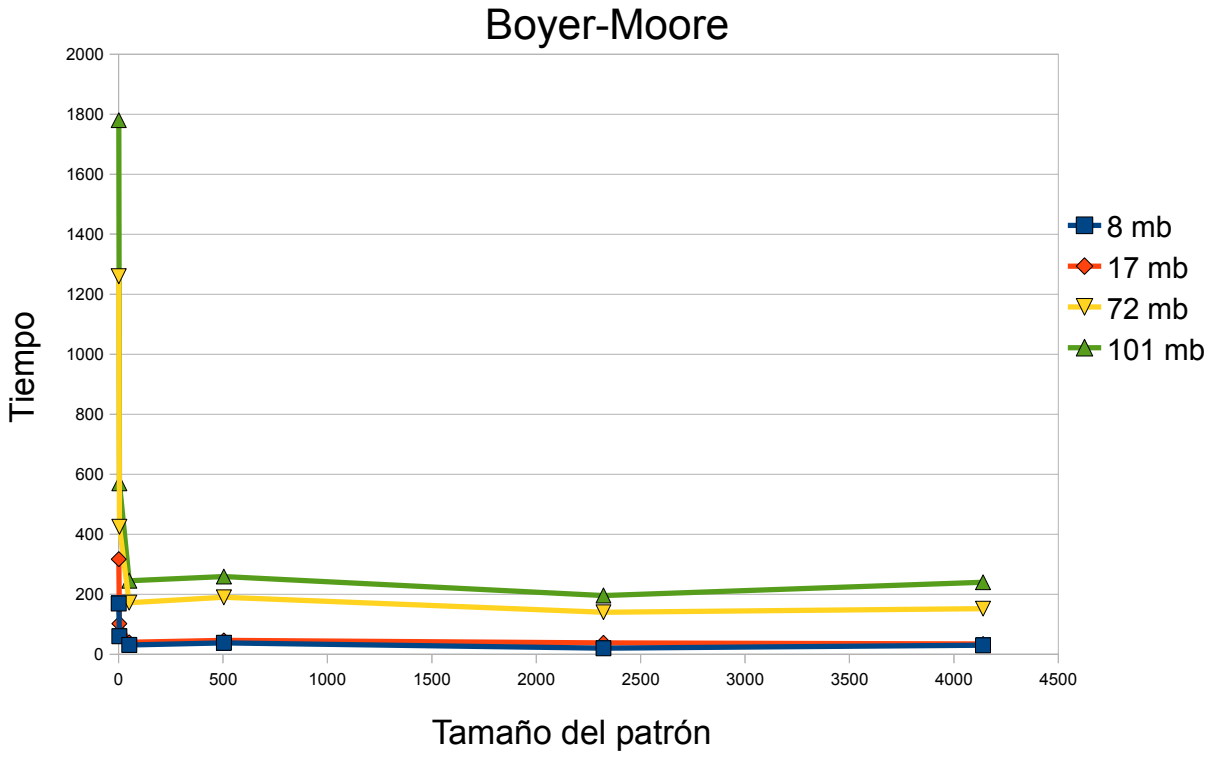

Figure 5: Resultados obtenidos con el algoritmo de Boyer-Moore

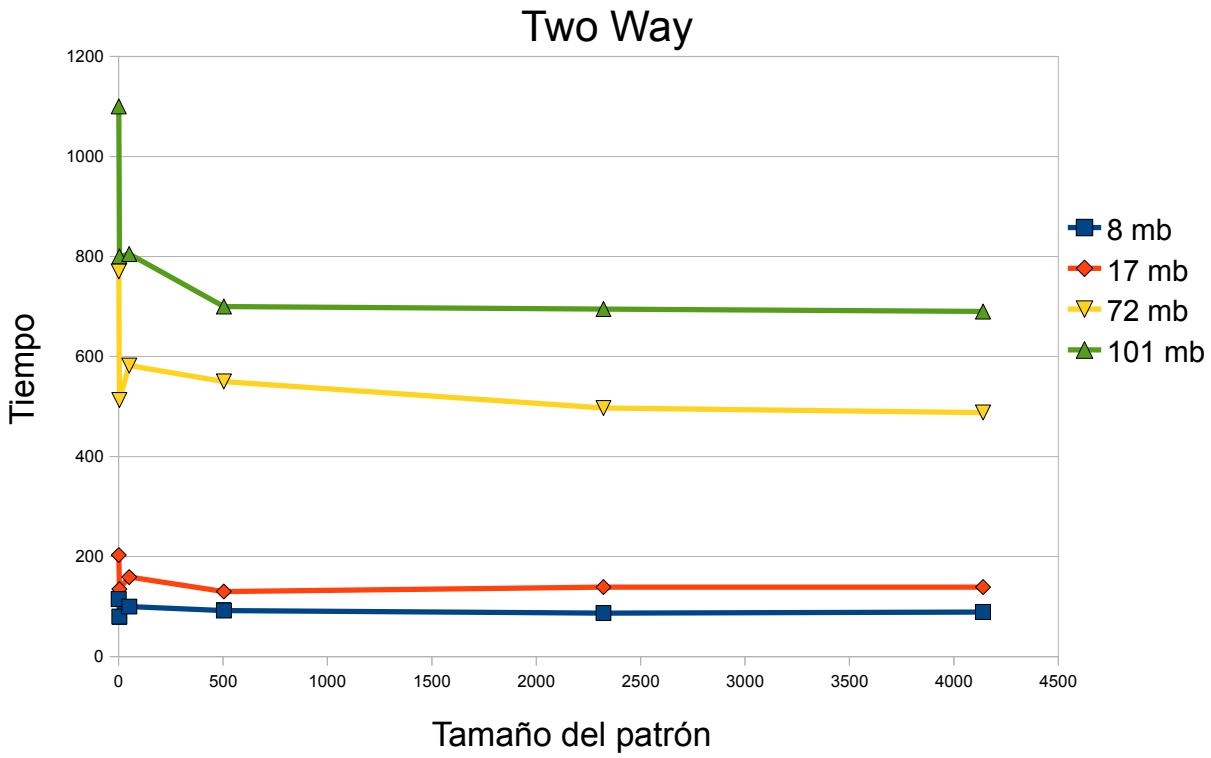

Figure 6: Resultados obtenidos con el algoritmo Two Way

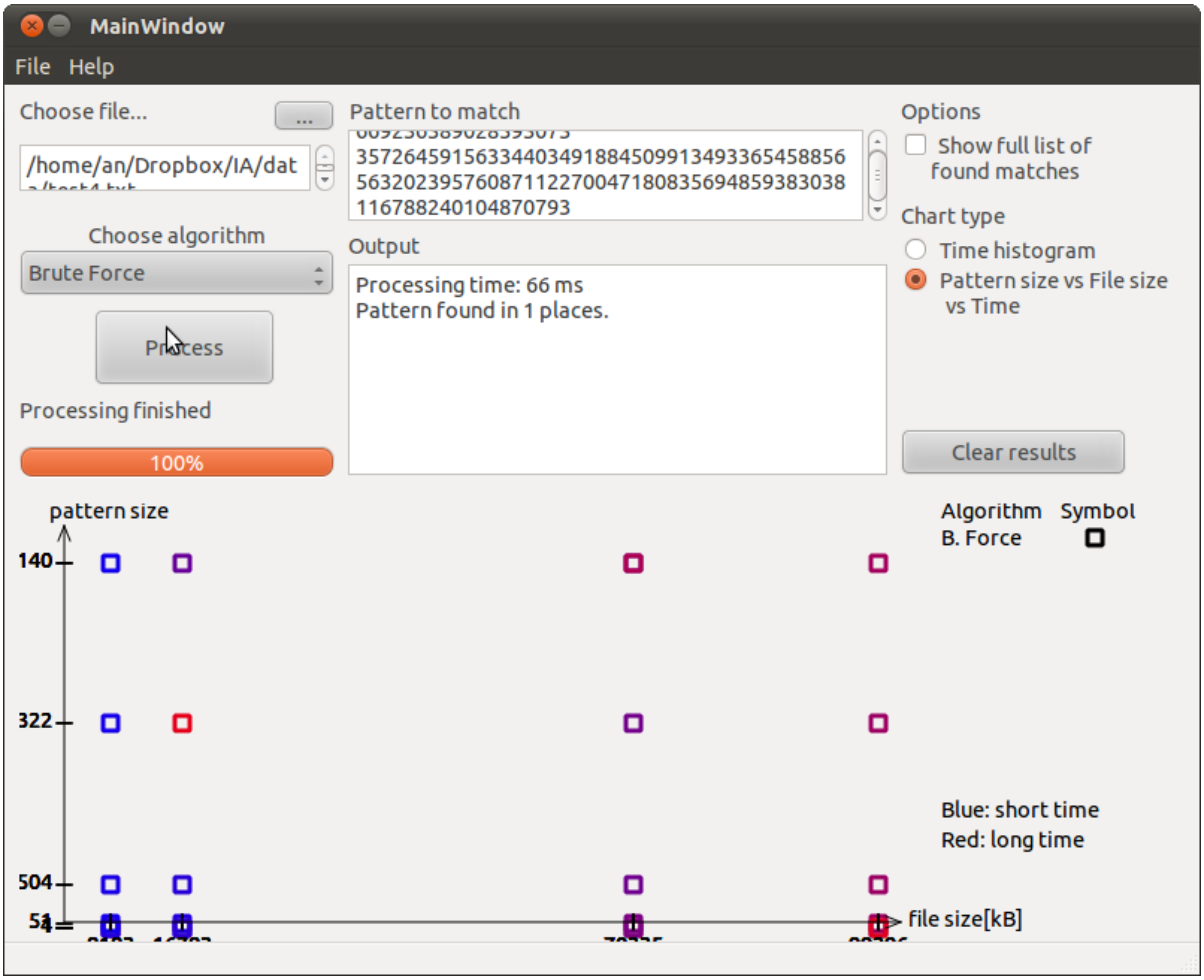

Figure 7: Resultados obtenidos con el algoritmo de Fuerza Bruta

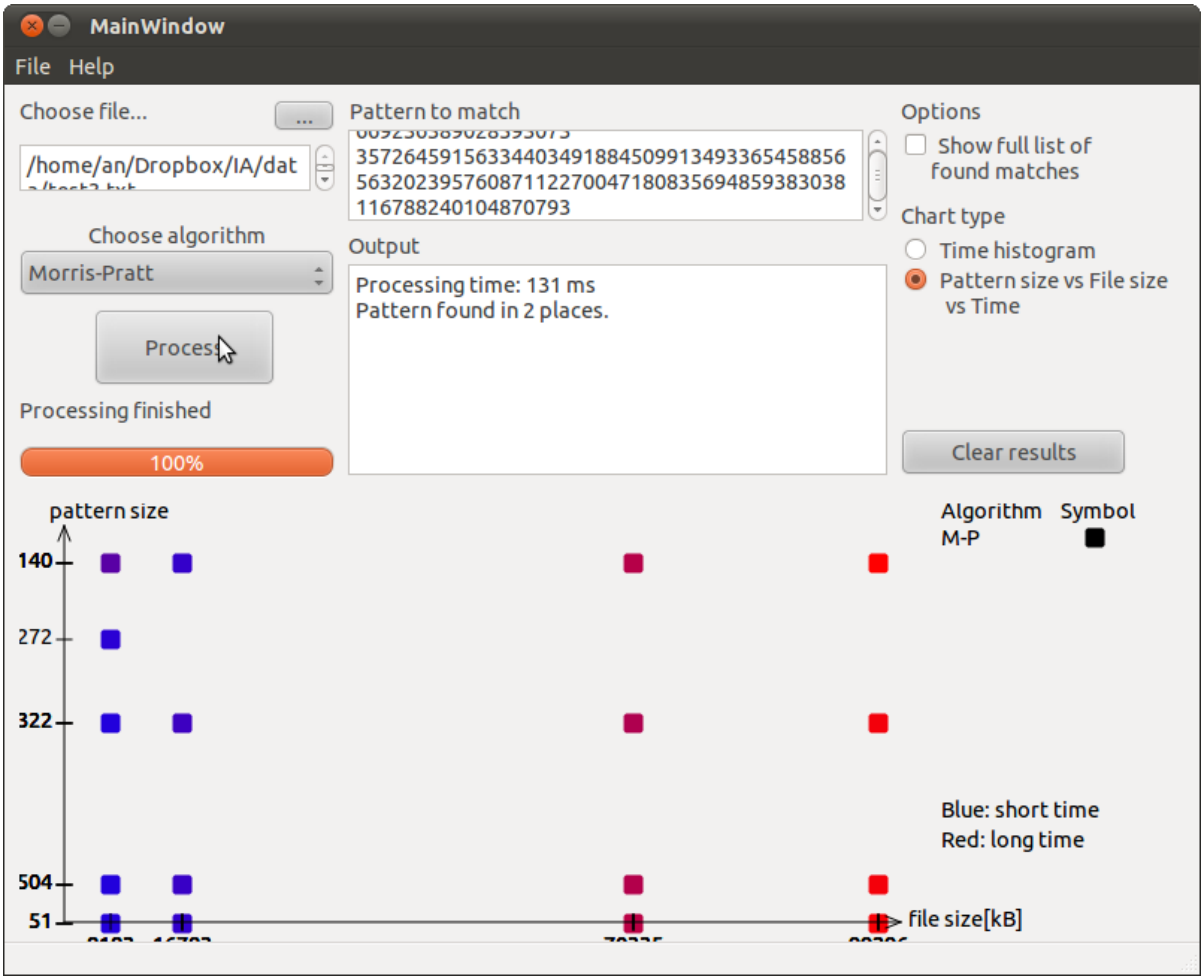

Figure 8: Resultados obtenidos con el algoritmo de Morris-Pratt

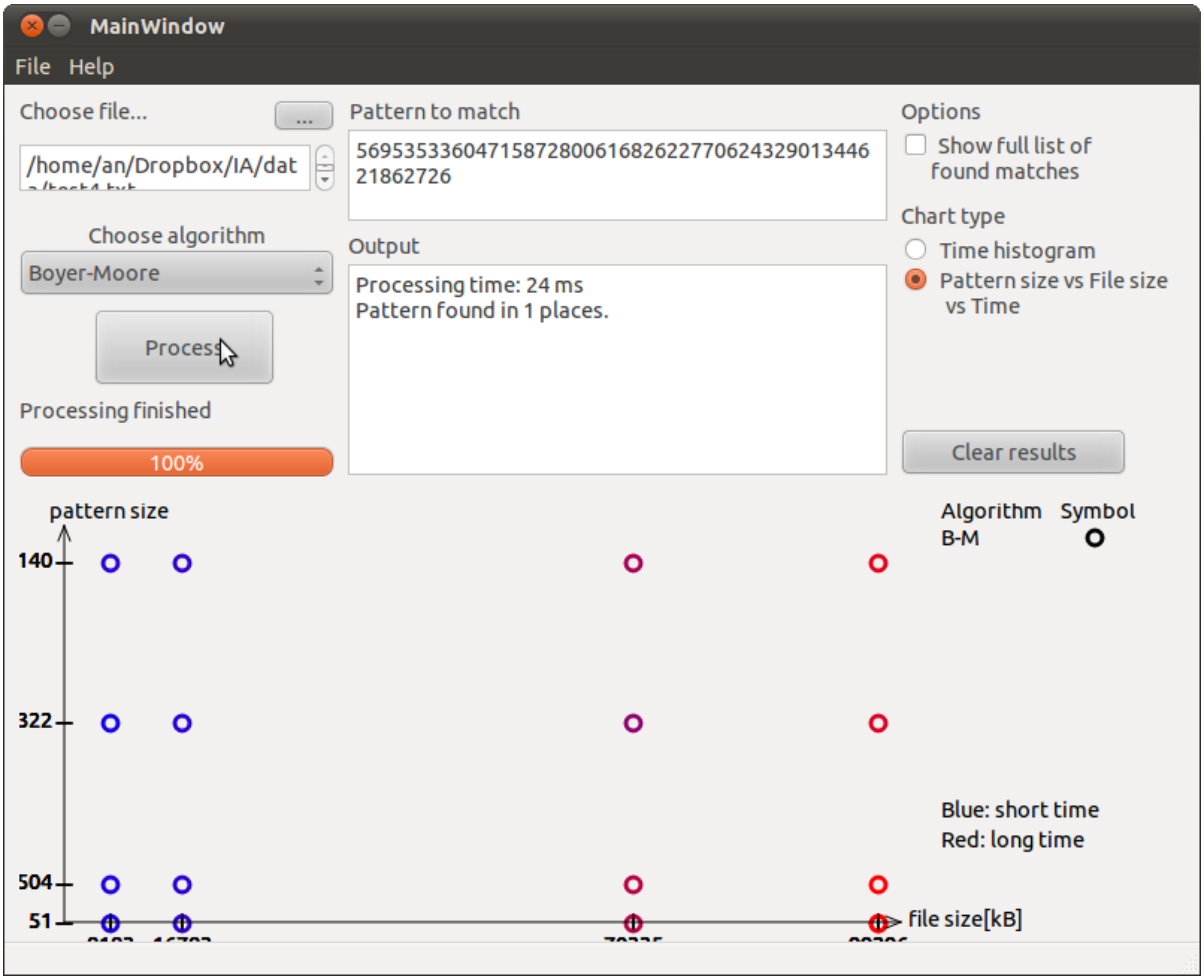

Figure 9: Resultados obtenidos con el algoritmo de Boyer-Moore

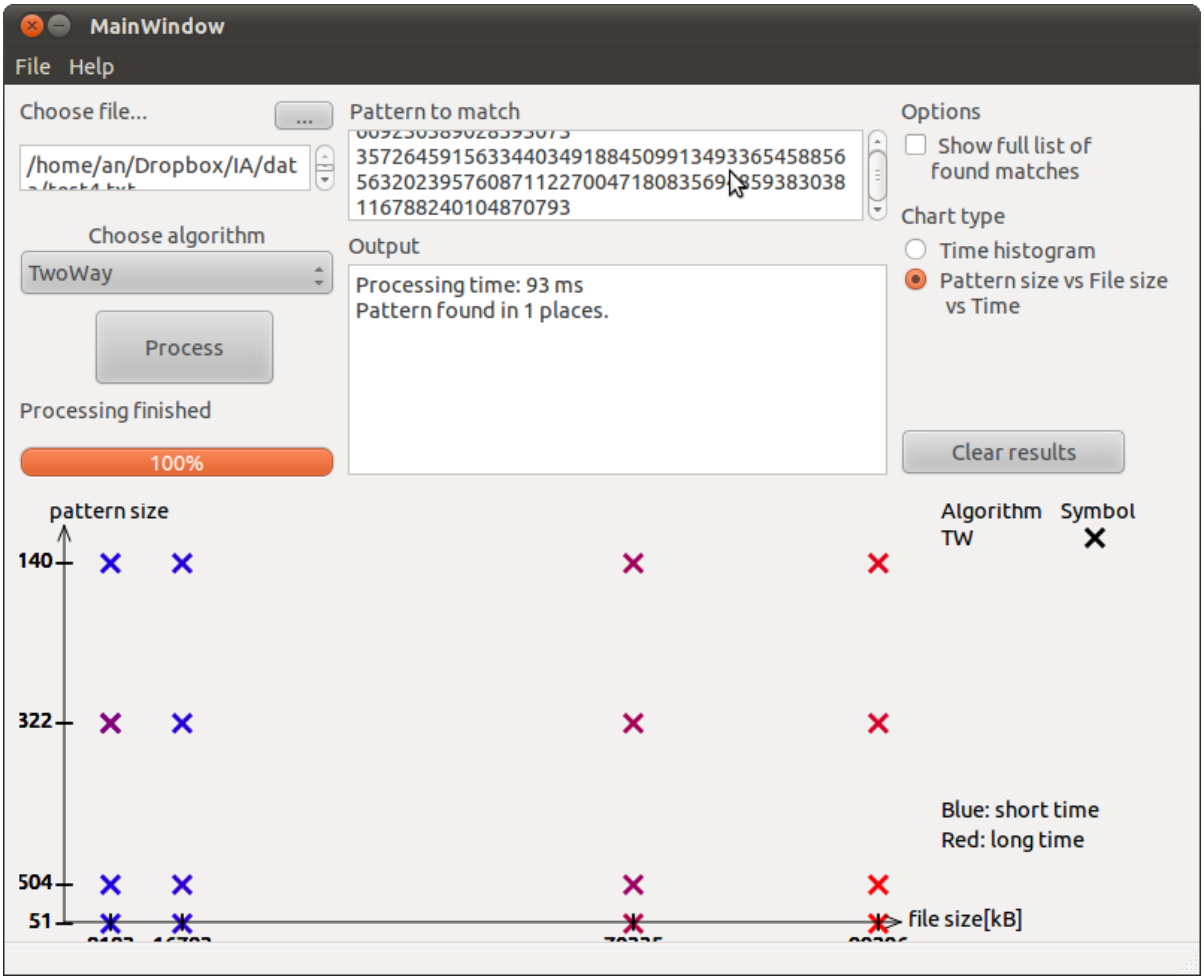

Figure 10: Resultados obtenidos con el algoritmo Two Way