2015

Universidad de Chile, Facultad de Ciencias Físicas y Matemáticas, Departamento de Computación

Cristian Andrade Muñoz

[DISEÑO Y ANÁLISIS DE ALGORITMOS: BÚSQUEDA EN TEXTO]

Análisis cuantitativo del desempeño de algoritmos e Fuerza Bruta, Knuth-Morris-Pratt y Boyer-Moore-Horspool, de acuerdo a su tiempo de ejecución y comparaciones realizadas sobre textos dados.

Resumen Ejecutivo:

Se asignó la construcción del código fuente para la implementación de algoritmos de búsqueda en texto, específicamente búsqueda por Fuerza Bruta, algoritmo de Knuth-Morris-Pratt, y algoritmo Boyer-Moore (Boyer-Moore-Horspool). Para esto, se utilizó el lenguaje Java, con implementación en Clases para los distintos algoritmos, y empaquetamiento en JAR para ejecución headless en un servidor de 64 bits, con procesador de 4 núcleos de 2.93GHz, con 6 Gb de RAM a 1333Mhz.

Para los archivos de búsqueda, se utilizaron archivos generados a partir de código provisto en http://pizzachili.dcc.uchile.cl/texts.html, específicamente el generador de datos binarios, archivos de código ADN real y sintético (en su formato de caracteres c-g-a-t), y archivos de texto real y sintético en alfabeto inglés alfanumérico.

Los resultados del algoritmo utilizado indican que el algoritmo con mejor desempeño *over all* es el algoritmo Boyer-Moore-Horspool, mostrando tiempos de ejecución y comparaciones marcadamente más bajos que sus competidores en las pruebas más complejas. Sin embargo, para los alfabetos más acotados (binario, 2 caracteres), el desempeño en tiempo se iguala, incluso se castiga en cantidad de comparaciones realizadas, debido a la función de cálculo del desplazamiento en BMH. Esto marca su utilidad para grandes textos con alfabetos más amplios.

Introducción:

La búsqueda en texto es una tarea del día a día en los tiempos modernos, donde es necesario aplicarla en situaciones desde un procesador de texto básico, hasta en procesos de análisis de datos de gran envergadura, en laboratorios, observaciones astronómicas, servicios de búsqueda de datos online, entre otras. Es necesario medir, en estos casos, el desempeño de los algoritmos utilizados, buscando con ello el más adecuado en cuanto al tiempo necesario para ejecutarlo, la cantidad de datos revisados, y la dificultad de implementarlos.

Se indicó para esta tarea que se debía realizar la implementación de 3 algoritmos de búsqueda en texto, donde se debía especificar métodos para analizar y cuantificar su desempeño en distintas situaciones. En particular, estos algoritmos fueron:

Algoritmo de Fuerza Bruta:

Este algoritmo es el más básico de todos, consistente en comparar caracter por caracter el patrón con el texto de búsqueda, desplazando en 1 unidad cada vez que el patrón no calce.

Algoritmo Knuth-Morris-Pratt

Muy similar a Fuerza Bruta, realiza las comparaciones de izquierda a derecha, caracter por caracter con la diferencia que, para el desplazamiento del patrón sobre el texto, se utiliza una función de *fracaso* precalculada, la cual busca cuantificar los prefijos y sufijos concordantes que pudiera presentar el patrón, y sobre ellos maximizar el desplazamiento.

Algoritmo Boyer-Moore-Horspool

A diferencia de los anteriores, este algoritmo recorre el patrón de forma inversa, pero también busca maximizar el desplazamiento mediante el calce de la última letra comparada con su equivalente en el patrón, si existe, o desplazar el patrón completamente, en caso contrario.

Hipótesis:

Asumimos *n* el tamaño del texto, *m* el tamaño del patrón, y *c* el tamaño de cada alfabeto (binario, ADN, y texto alfanumérico en inglés, todos fijos). Asumimos que un alfabeto mayor implica mayores chances de desplazar el patrón en el texto por no calce, lo que lo lleva a hacer menos comparaciones. Asumimos también que los algoritmos recorrerán necesariamente la totalidad del texto, debido a que se solicita que se encuentren todas las ocurrencias del patrón en el texto.

En Fuerza Bruta, asumimos que la cantidad de comparaciones será siempre del orden de *n x m*, con *n* fijo, y *m* variable según el largo del patrón utilizado.

En Knuth-Morris-Pratt dependemos del tamaño *m* de patrón extraído y el tamaño del lenguaje *c*, pues una mayor cantidad de subcadenas similares dentro del texto disminuirá la efectividad de la función de salto, acercando su rendimiento a Fuerza Bruta

En Boyer-Moore-Horspool dependemos también del tamaño del lenguaje *c*, pues un lenguaje reducido aumentará las posibilidades de que un carácter que no calce en el patrón, aún esté presente más adelante. Con un lenguaje más amplio, la probabilidad de que esto ocurra baja significativamente.

Diseño e Implementación:

Se utilizó el lenguaje Java, bajo el ambiente de desarrollo Eclipse. Se utilizaron metodologías de desarrollo tipo Template para la generación de los algoritmos de búsqueda, y se empaquetó el software en formato JAR para ser ejecutado en una máquina Intel de 64 bits, con 4 núcleos de 2.93Ghz y 6Gb de RAM.

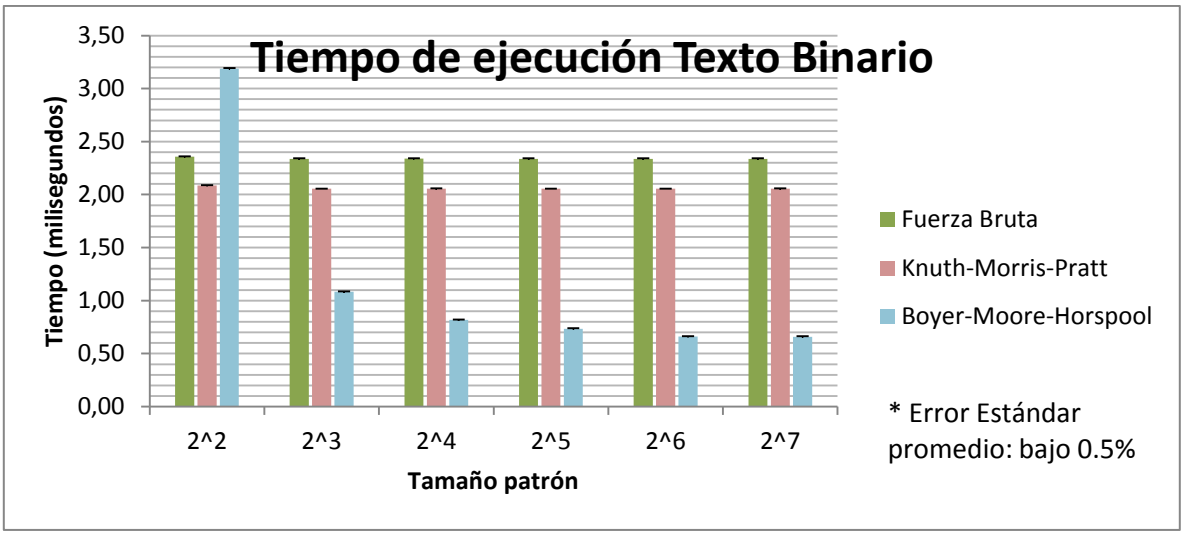
Se realizaron pruebas para tamaños de patrón de texto entre 4 y 128 caracteres, con textos de aproximadamente 1Mb cada uno. Se iteró 50.000 veces cada experimento, con patrones elegidos aleatoriamente de acuerdo a las indicaciones (patrones binarios generados por código, patrones de texto extraídos de la muestra), de forma de reducir el error asociado a las muestras (de tiempo y de comparaciones). Se utilizó la librería Apache Commons Math (archive.apache.org/dist/commons/math/) para el cálculo automático de promedio, varianza y desviación estándar.

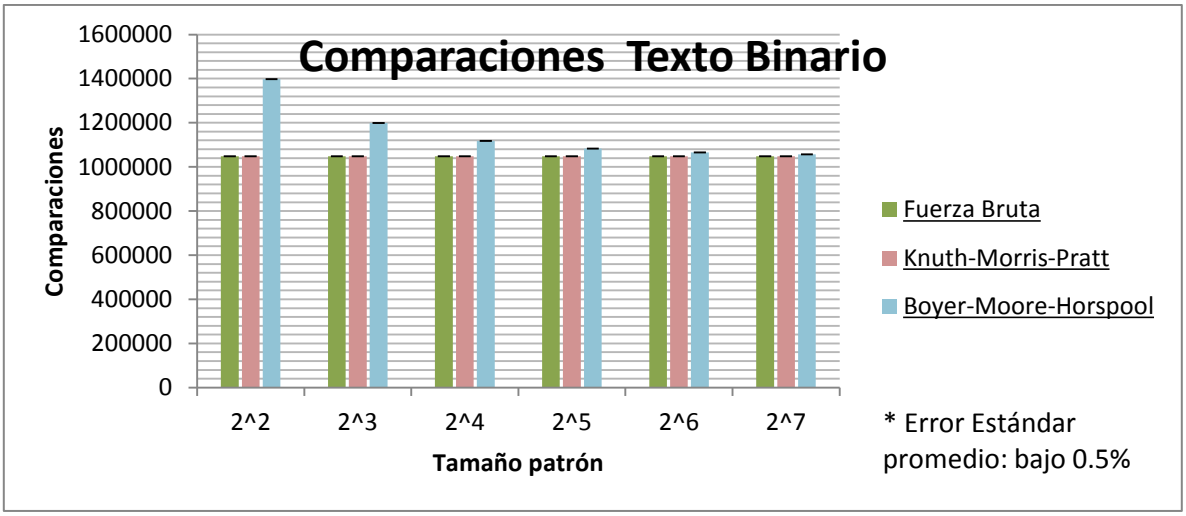
Los resultados se volcaron en un archivo de texto, el cual fue volcado manualmente en los gráficos mostrados a continuación. Se incluye error estándar en todos los gráficos.

Análisis

De acuerdo a los datos entregados por el código ejecutado, se tienen los siguientes resultados:

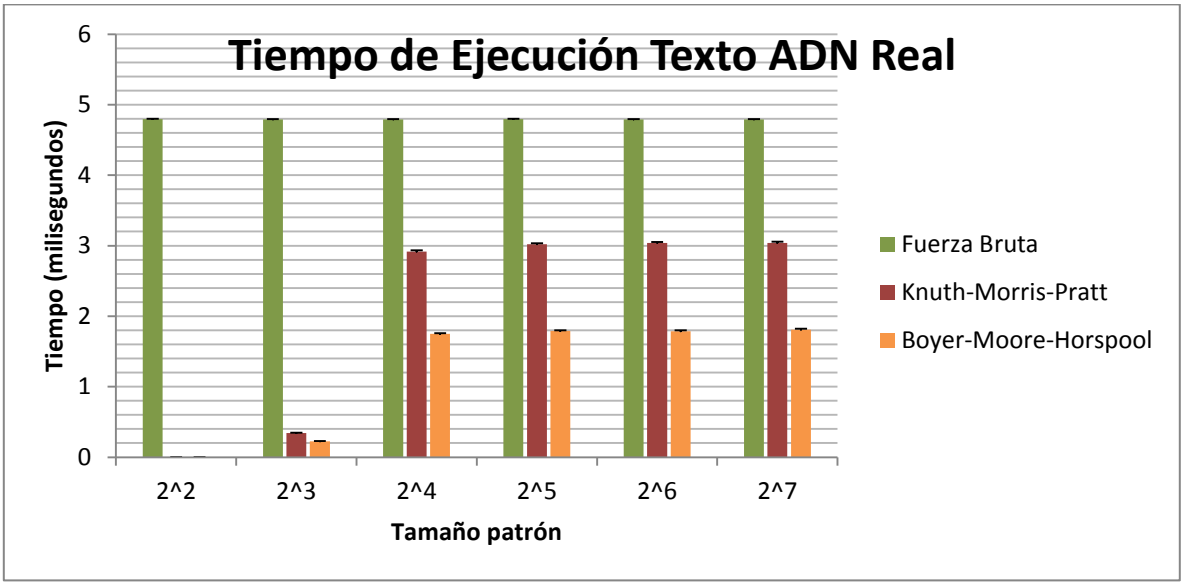
Texto Binario:

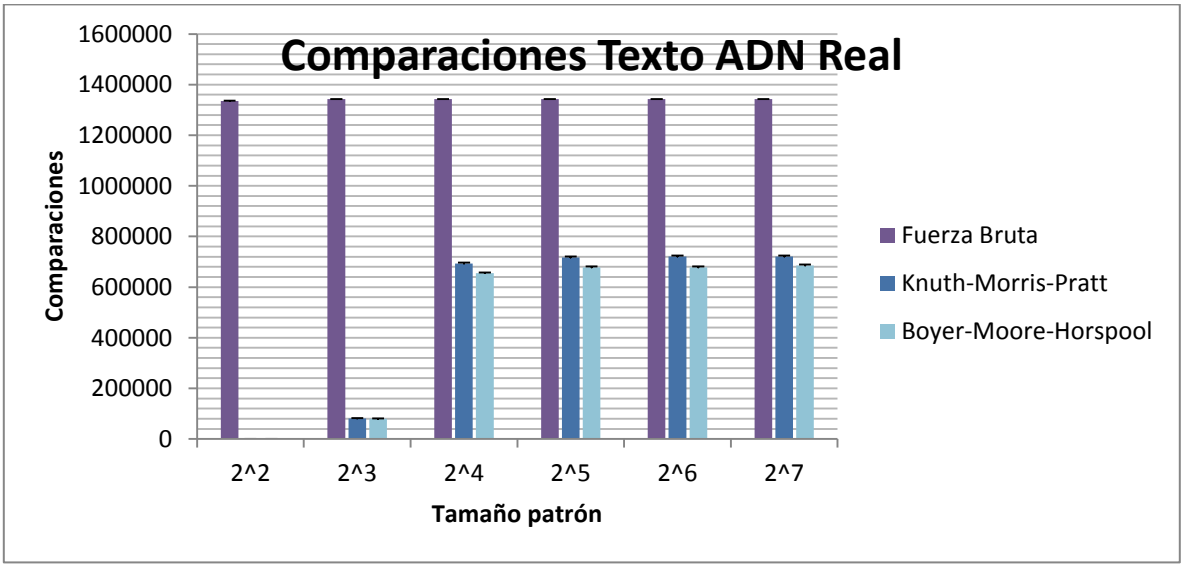




De acuerdo a estos resultados (tabla en Anexo), el desempeño entre Fuerza Bruta y Knuth-Morris-Pratt es muy similar en cuanto a comparaciones, y levemente mejor para el segundo en cuanto a tiempo de ejecución. Sin embargo, para ambas medidas, el algoritmo Boyer-Moore-Horspool tiene peor desempeño para patrones cortos, pero mejora levemente en comparaciones, y drásticamente en tiempo de ejecución, en cuanto aumenta el tamaño del patrón.

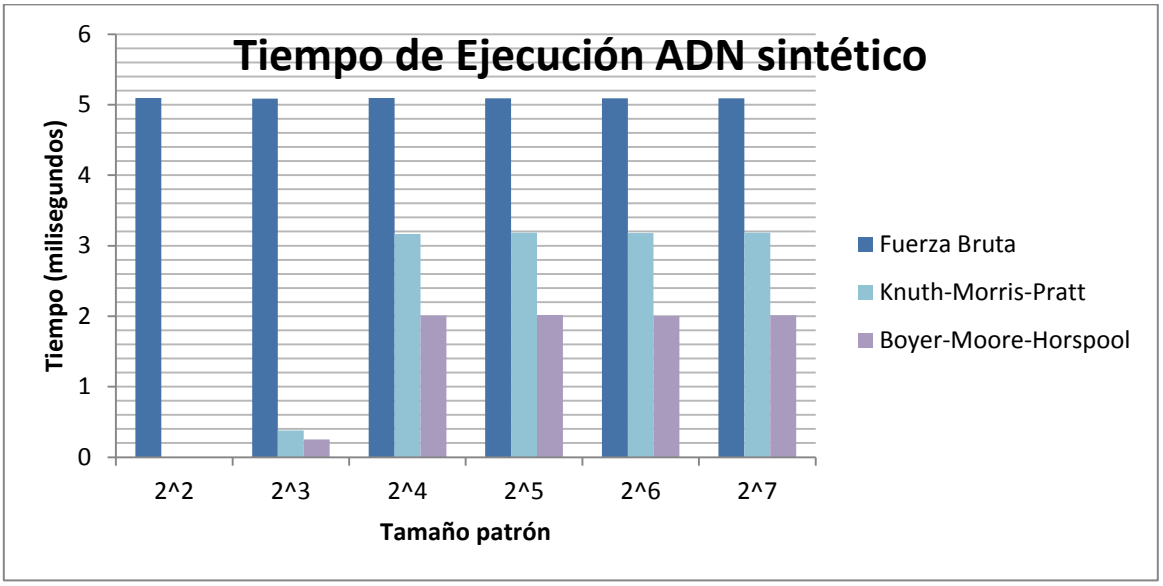
ADN real:

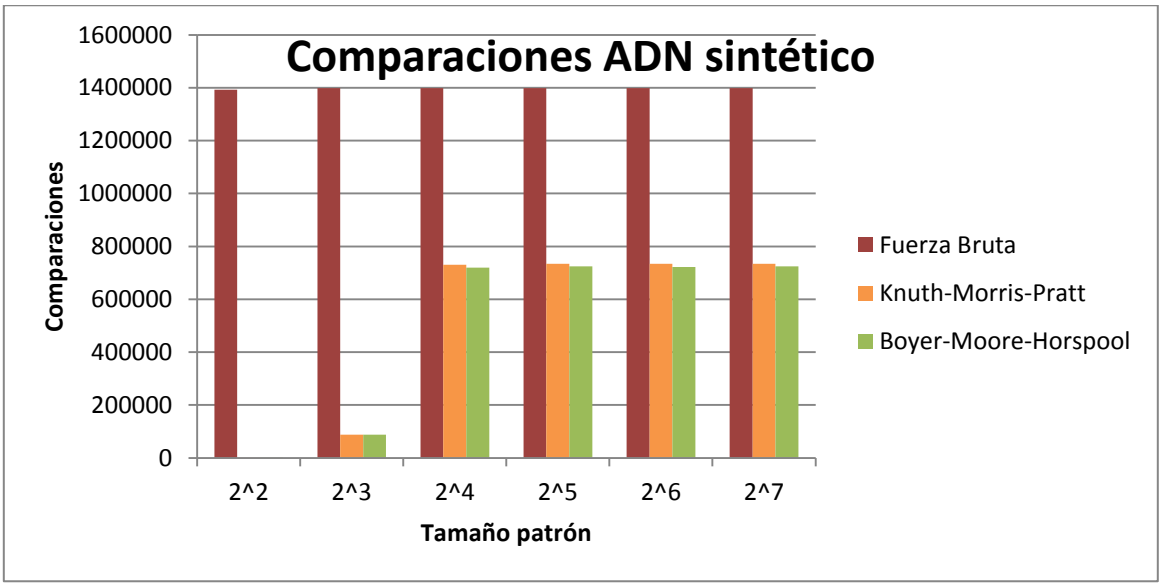




Con respecto al ADN real, se da cuenta que el algoritmo de Fuerza Bruta es relativamente constante en su accionar (tanto en comparaciones como en tiempo de ejecución), dando cuenta de su O(n*m). Con respecto a Knuth-Morris-Pratt y Boyer-Moore-Horspool se da cuenta de rendimientos parecidos, gracias al crecimiento del tamaño del alfabeto (de 2 a 4 caracteres). Sin embargo, el segundo ocupa cerca de 3/5 del tiempo del primero (y menos de la mitad de Fuerza Bruta), aunque toma cerca de la misma cantidad de comparaciones. Esto supondría que calcular bajo necesidad la función de salto en BMH es más eficiente que tenerla pre-calculada como KMP.

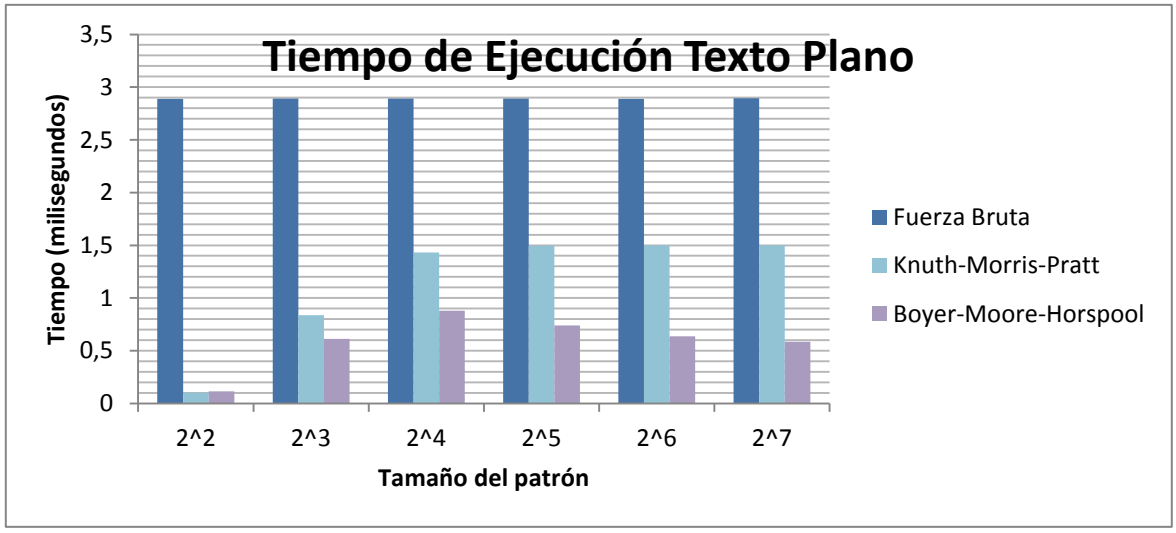
ADN sintético:

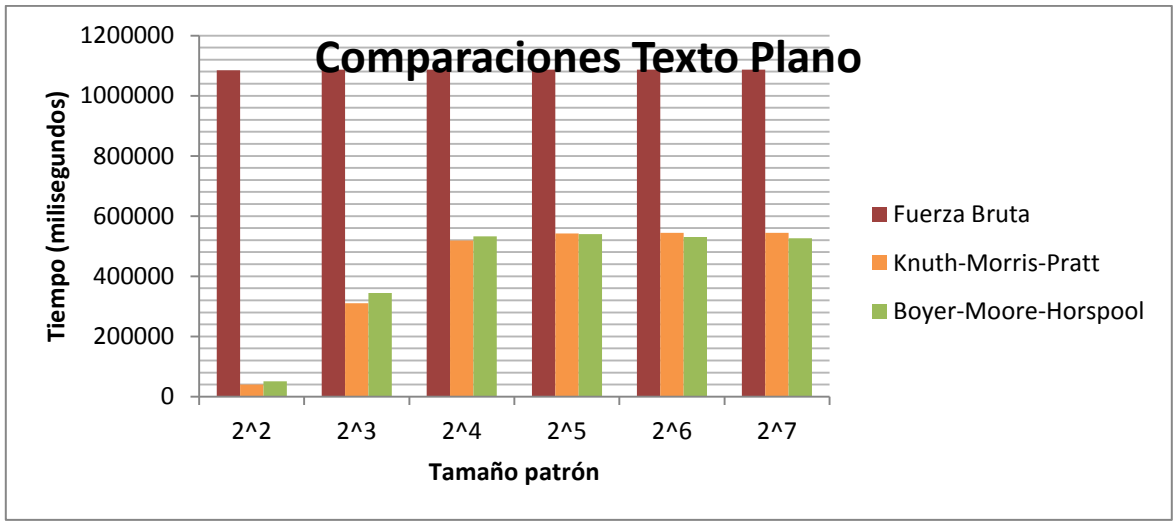




Con respecto al ADN sintético, el comportamiento es muy similar al de ADN real. Se nota un aumento en el tiempo empleado y las comparaciones realizadas por los algoritmos, dado que este archivo tiene menos limitaciones en su configuración (presencia azarosa de los caracteres, sin las limitaciones químicas del ADN real). Aún así, se aprecia un acercamiento en los rendimientos de Knuth-Morris-Pratt y Boyer-Moore-Horspool, bajo las mismas razones expuestas anteriormente.

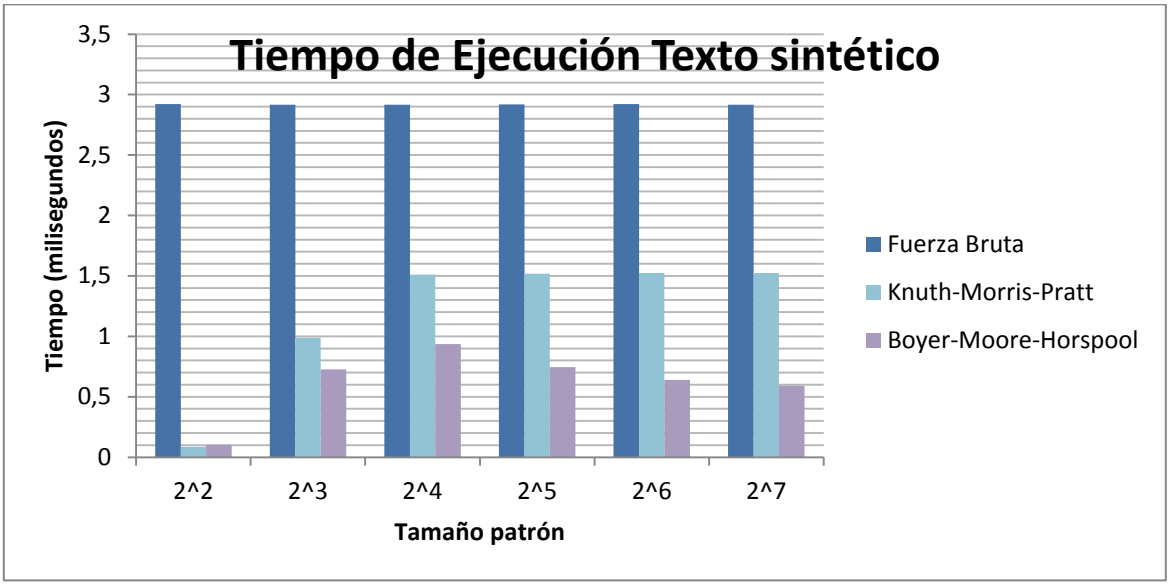
Texto Plano en alfabeto inglés:

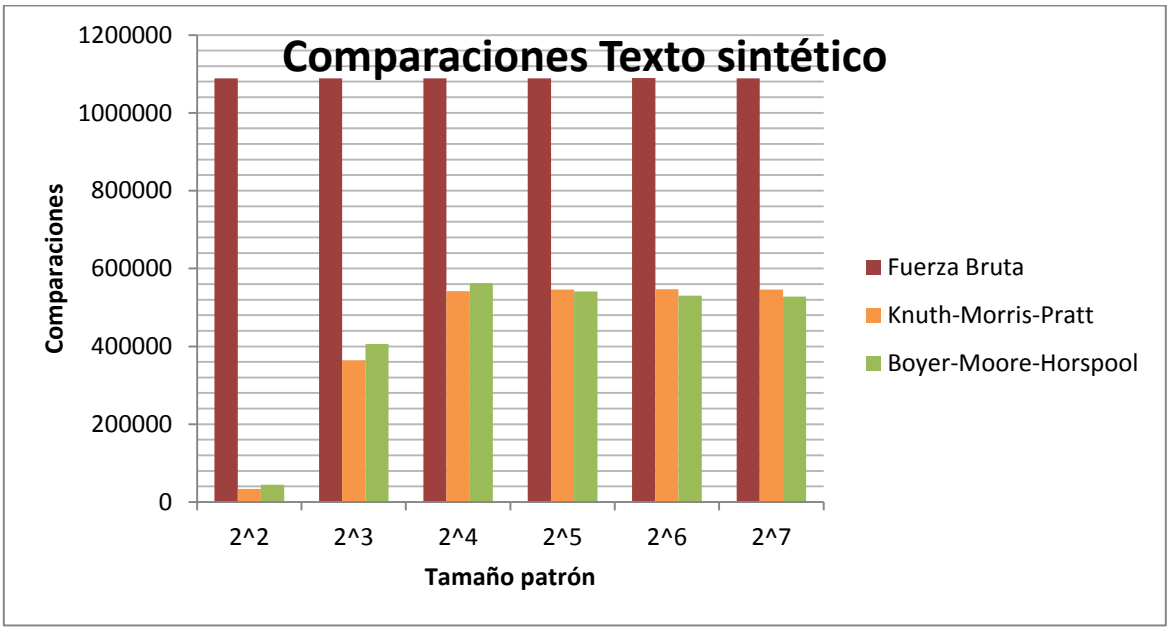




Con respecto al desempeño en texto plano, las iteraciones en Fuerza Bruta sufren una caída importante en tiempo de ejecución y comparaciones, lo cual puede deberse a las características del texto (eventualmente más corto). Con respecto a Knuth-Morris-Pratt se nota un desempeño muy acorde a lo visto en las pruebas anteriores, con tiempo y comparaciones ascendentes de acuerdo al tamaño del patrón. Sin embargo, Boyer-Moore-Horspool muestra un desempeño cóncavo en el tiempo de ejecución, donde se denotan menos movimientos del patrón de acuerdo a su extracción en el texto: se puede suponer que, con tamaños medios (16 caracteres), los calces parciales del patrón son más comunes que en extracciones de texto más cortos o más largos, debido al tamaño del alfabeto.

Texto Plano sintético en alfabeto inglés:





De forma similar al texto plano real, los resultados se asemejan mucho entre ambos experimentos, con un leve aumento en comparaciones y tiempo de ejecución en los 3 algoritmos. Se mantiene la tendencia sobre Fuerza Bruta, y hacia los patrones más largos se acentúa la diferencia entre KMP y BMH.

Conclusiones:

Se detecta fehacientemente los problemas de desempeño que presenta un algoritmo de búsqueda por Fuerza Bruta, castigado fuertemente de acuerdo al tamaño del texto a revisar.

Para Knuth-Morris-Pratt se mantiene un rendimento relativamente constante en la búsqueda sobre el texto binario, pero fuertemente marcada una curva ascendente en alfabetos más amplios, de acuerdo al tamaño del patrón utilizado.

Para Boyer-Moore-Horspool se nota un muy mal rendimiento sobre el alfabeto binario, pero con mejoras en cuanto aumenta el tamaño del patrón. Esta dependencia sobre el patrón y el alfabeto se demuestra en los otros 4 experimentos, notando que mejora consistentemente su rendimiento frente a sus competidores en cuanto el tamaño del patrón es más amplio, y su alfabeto es más rico. Incluso, en texto plano, se nota que mejora incluso su propio rendimiento en cuanto aumenta el tamaño del patrón, denotando la importancia de su función de desplazamiento, ahorrando tiempo y comparaciones.