Licenciatura en ciencia de la computación

Combinaciones

Matemática Computacional

Profesor:
Nicolas Thériault

Autor:
Sergio Salinas
Danilo Abellá

Contents

Introducción

En este informe se pretende mostrar las distintas formas y/o estrategias para calcular la combinatoria de n y r valores , para lo cual se utilizaron distintas técnicas y estrategias tanto algo-rítmicamente como en el uso de conocimiento matemático, además claro de formas de programación.

Cabe mencionar que se utilizó la librería gmp como forma de cálculo mas rápido, todo esto en lenguaje C/C++.

1 Algoritmo Implementado

1.1 Estrategía uno

Esta estrategía es la más simple, solo se calcularon los valores n!, (n-r)! y r!, luego se obtuvo la combinatoria multiplicando r! por (n-r)! y diviendo n! por el resultado de la multiplicación anterior.

1.2 Estrategía dos

Para la creación de este algoritmo se baso en el siguiente ejemplo:

$$\binom{7}{4} = \frac{7!}{4! \cdot 3!} = \frac{7 \cdot 6 \cdot 5}{3 \cdot 2}$$

$$\binom{7}{3} = \frac{7!}{3! \cdot 4!} = \frac{7 \cdot 6 \cdot 5}{3 \cdot 2}$$

Gracias a la propiedad de simetría da el mismo resultado pero con los resultados cambiados, cuando $n - r > r$ ocurre que el mayor número del denominador se queda a la derecha en caso contrario el mayor se queda la izquierda.

Además se eso se observa cuando se factoriza factoriales el númerador queda de la forma $n \cdot (n-1) \cdot \ldots \cdot (r+1)!$, en base a todo esto se creo el siguiente algoritmo

- Se comprueba cuál número es más grande, si r ó n-r.
- Se calcula la multiplicación de r + 1 o (n -r) + 1 (el que sea el mayor) hasta n, de esta forma se simula como si hubiera factorizado n a mayor denominador.
- Se calcula el factorial del denominador menor.
- Se calcula la combinatoria diviendo el numerador por el denominador.

1.3 Estrategía tres

Se aplico el mismo algoritmo que en la estrategía uno pero se cambiaron los mpz por mpf para así usar el punto flotante en los calculos.

1.4 Estregía cuatro

En este problema se utilizó la combinación numérica con la estrategia 2.2 mediante la cual sacamos el valor de una combinatoria utilizando el factorial de la división entre la variable

n y r respectivamente , con r menor a n.

Para dicha estrategia se utilizó un algoritmo que primero saca el valor de la primera división (en real) , luego le resta 1 tanto al denominador como al numerador , y multiplica el resultado de la nueva división (en real) con el resultado de la primera... y así sucesivamente hasta que r = 1.

Una vez finalizado se tendría el valor de la combinatoria tras todas las multiplicaciones de cada fracción en cada ciclo recorrido.

1.5 Estrategía cinco

En este caso en nuestra fórmula de Striling primero se realizó una factorización de la fórmula para conseguir una mayor eficiencia y facilidad al trabajar con su algoritmo, evitando así variables de más y conseguir un resultado más rápido.

La factorización relizada fue la siguiente:

$$n! \approx (\sqrt{(2 \cdot \pi)}) \cdot (\frac{n^{(n+(1/2)}}{e^n})$$

$$n! \approx (\sqrt{(2 \cdot \pi)}) \cdot (\frac{n^n \cdot n^{(1/2)}}{e^n})$$

$$n! \approx (\sqrt{(2 \cdot \pi)}) \cdot ((\frac{n}{e})^n \cdot \sqrt{n})$$

2 Formulación experimentos

Para probar los algoritmos se hizo un script en bash que compilara una vez y ejecutara varias veces el ejecutable, variando los metodos de entrada.

Los experimentos fueron 3.

- n fijo para n=100 y r crece hasta n
- n fijo para n=300 y r crece hasta n
- n fijo para n=500 y r crece hasta n
- n fijo para n=10000 y r crece hasta n

3 Información de Hardware y Software

3.1 Notebook - Danilo Abellá

3.1.1 Software

- SO: Xubuntu 16.04.1 LTS
- GMP Library
- Mousepad 0.4.0

3.1.2 Hardware

- AMD Turion(tm) X2 Dual-Core Mobile RM-72 2.10GHz
- Memoria (RAM): 4,00 GB(3,75 GB utilizable)
- Adaptador de pantalla: ATI Raedon HD 3200 Graphics

3.2 Notebook - Sergio Salinas

3.2.1 Software

- SO: ubuntu Gnome 16.04 LTS
- Compilador: gcc version 5.4.0 20160609
- Editor de text: Atom

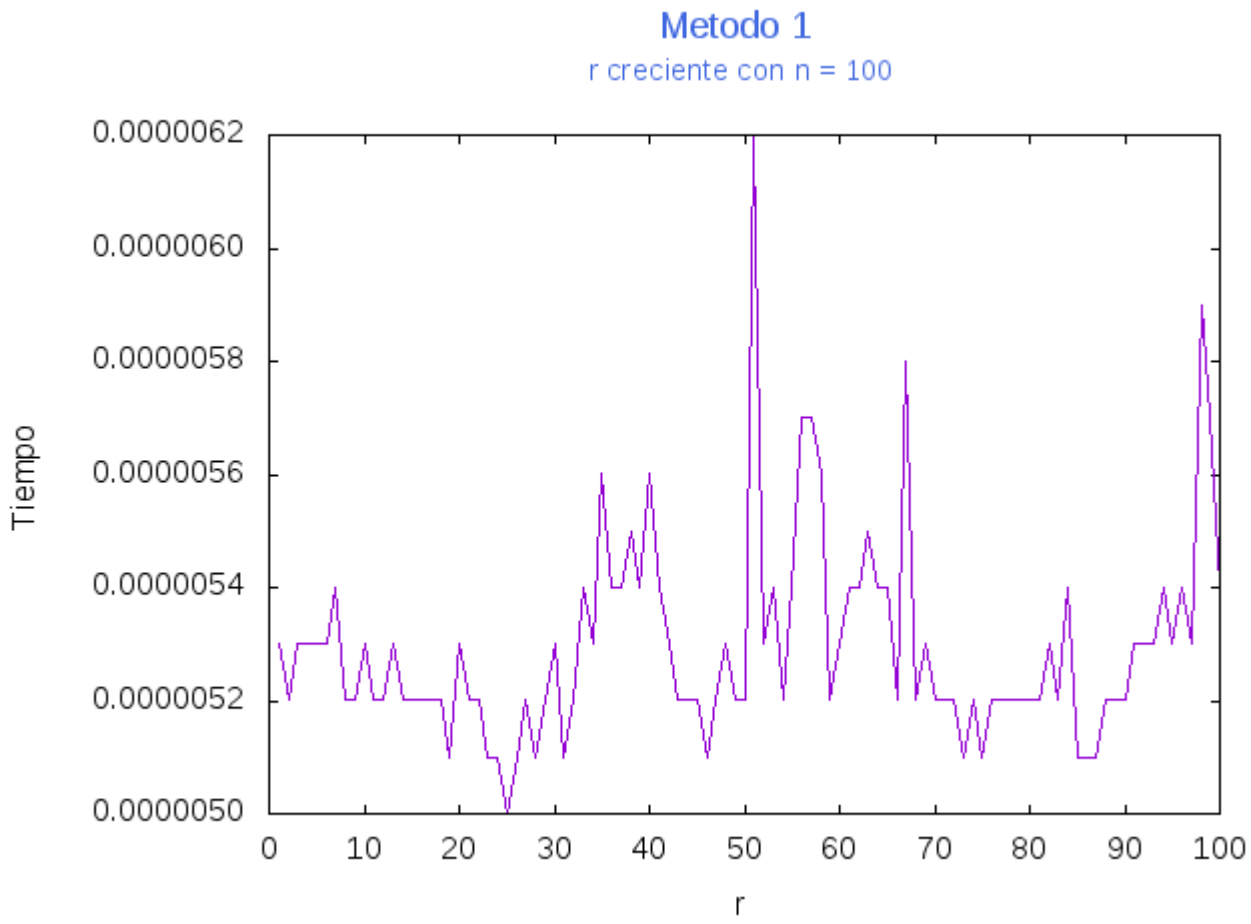
3.2.2 Hardware

- Procesador: Intel Core i7-6500U CPU 2.50GHz x 4
- Video: Intel HD Graphics 520 (Skylake GT2)

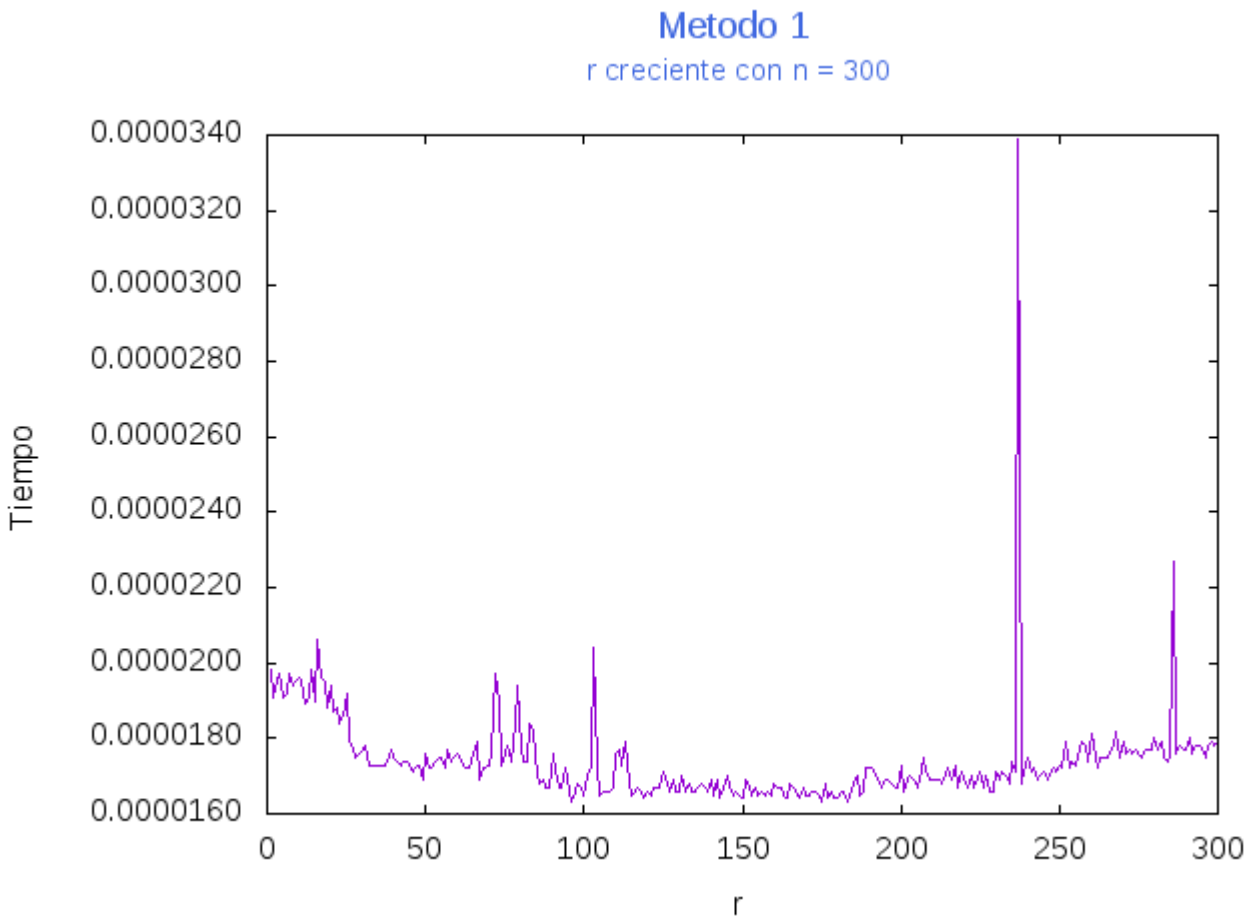
4 Curvas de desempeño de resultados

4.1 Estrategía uno

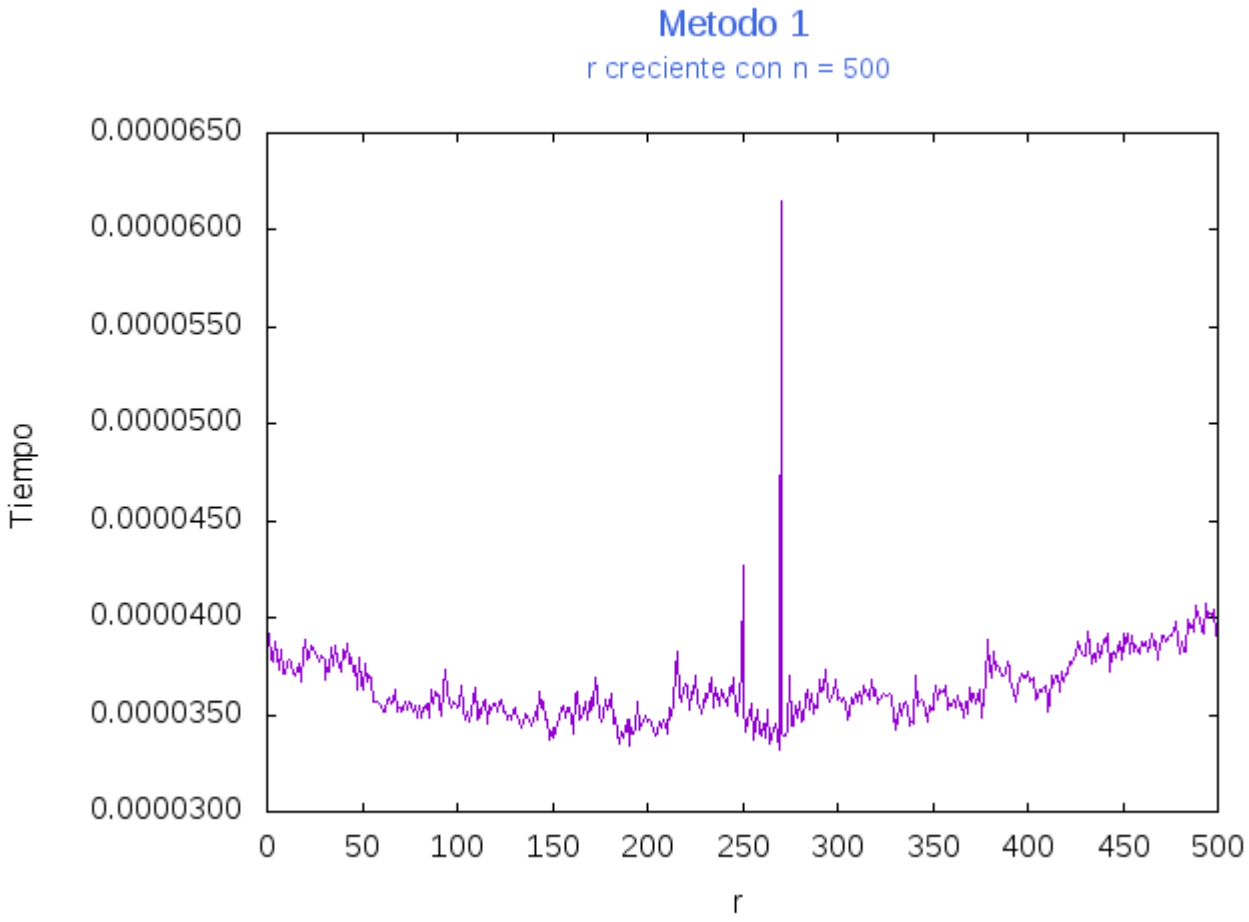
4.1.1 Para n fijo y r crece hasta n=100



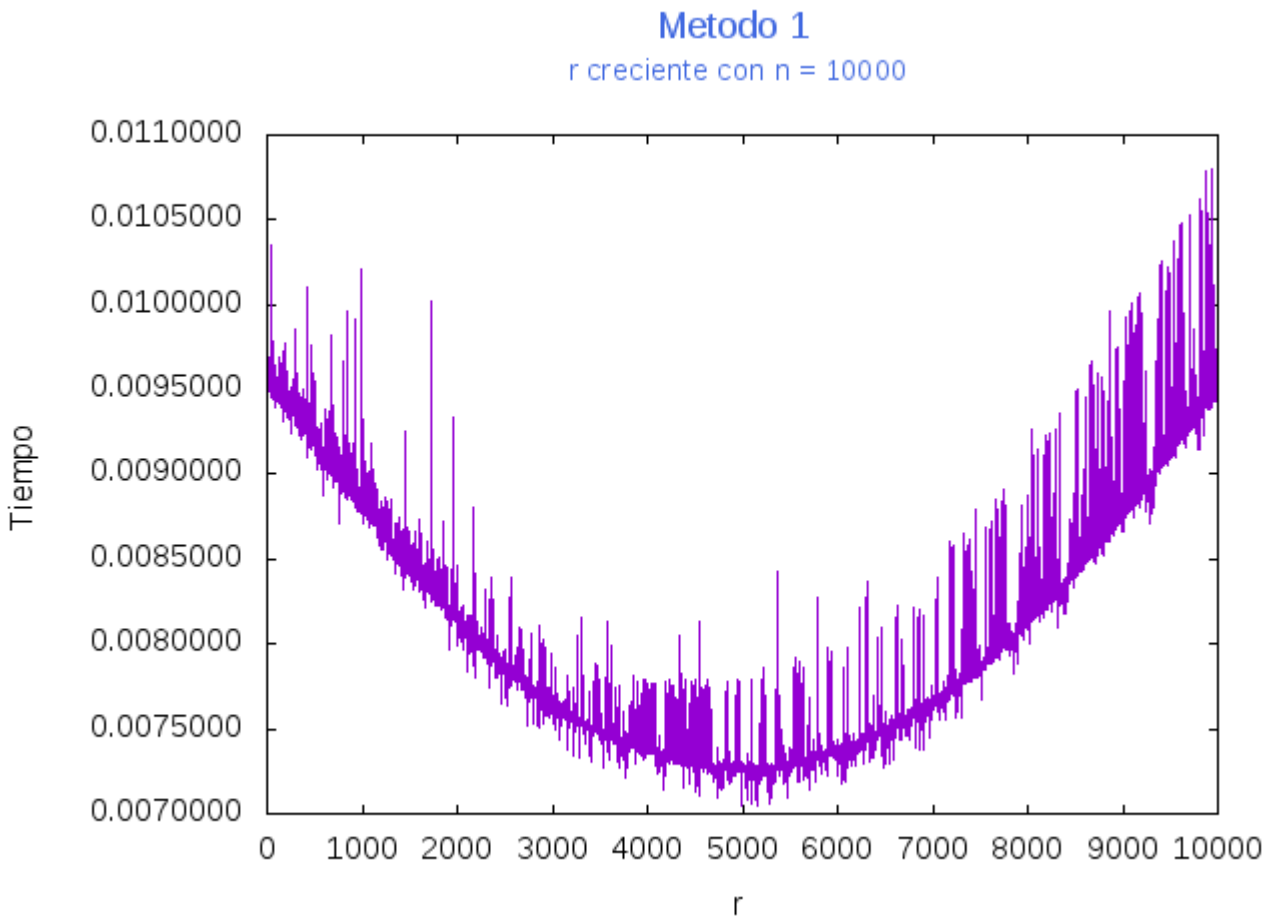
4.1.2 Para n fijo y r crece hasta n=300



4.1.3 Para n fijo y r crece hasta n=500

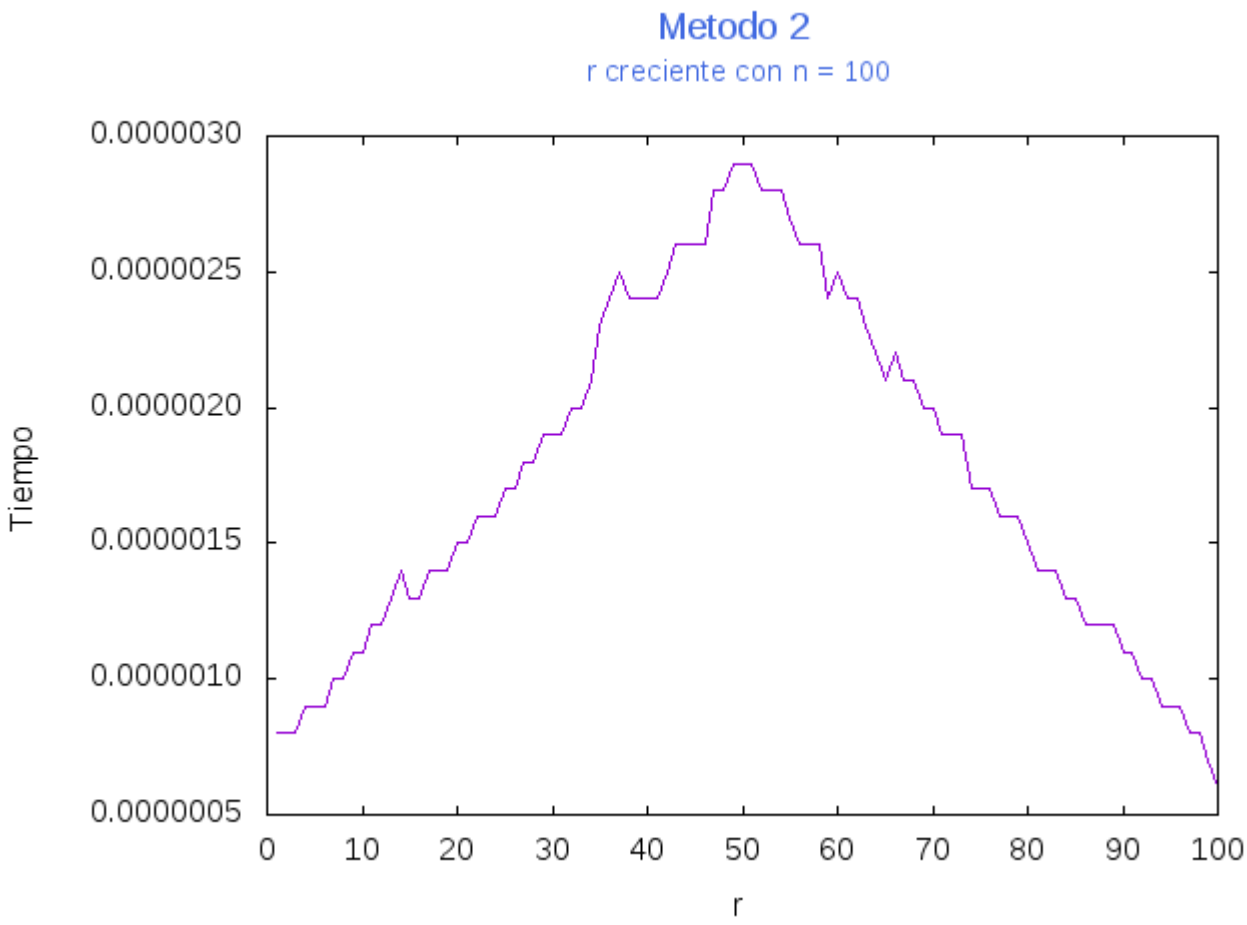


4.1.4 Para n fijo y r crece hasta n=10000

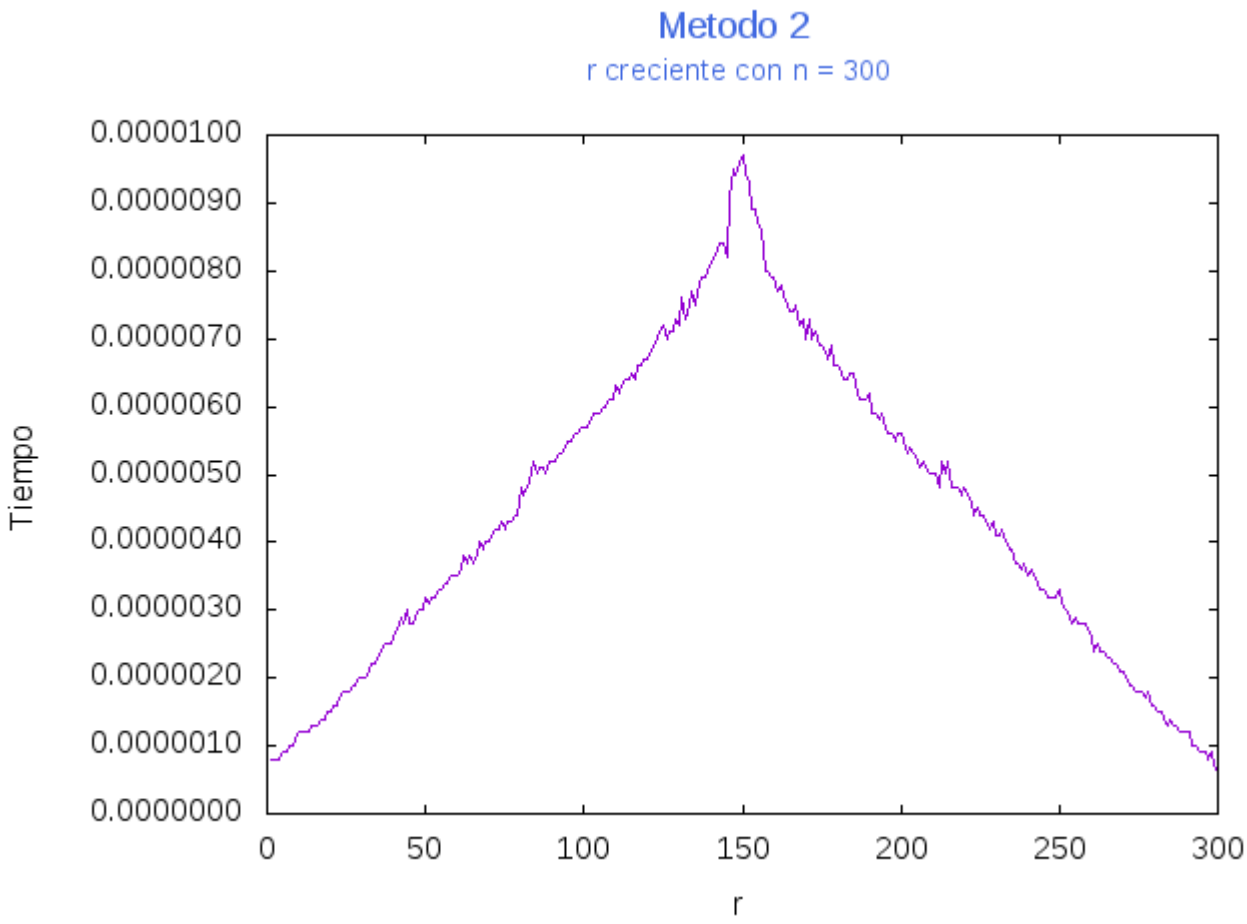


4.2 Estrategia dos

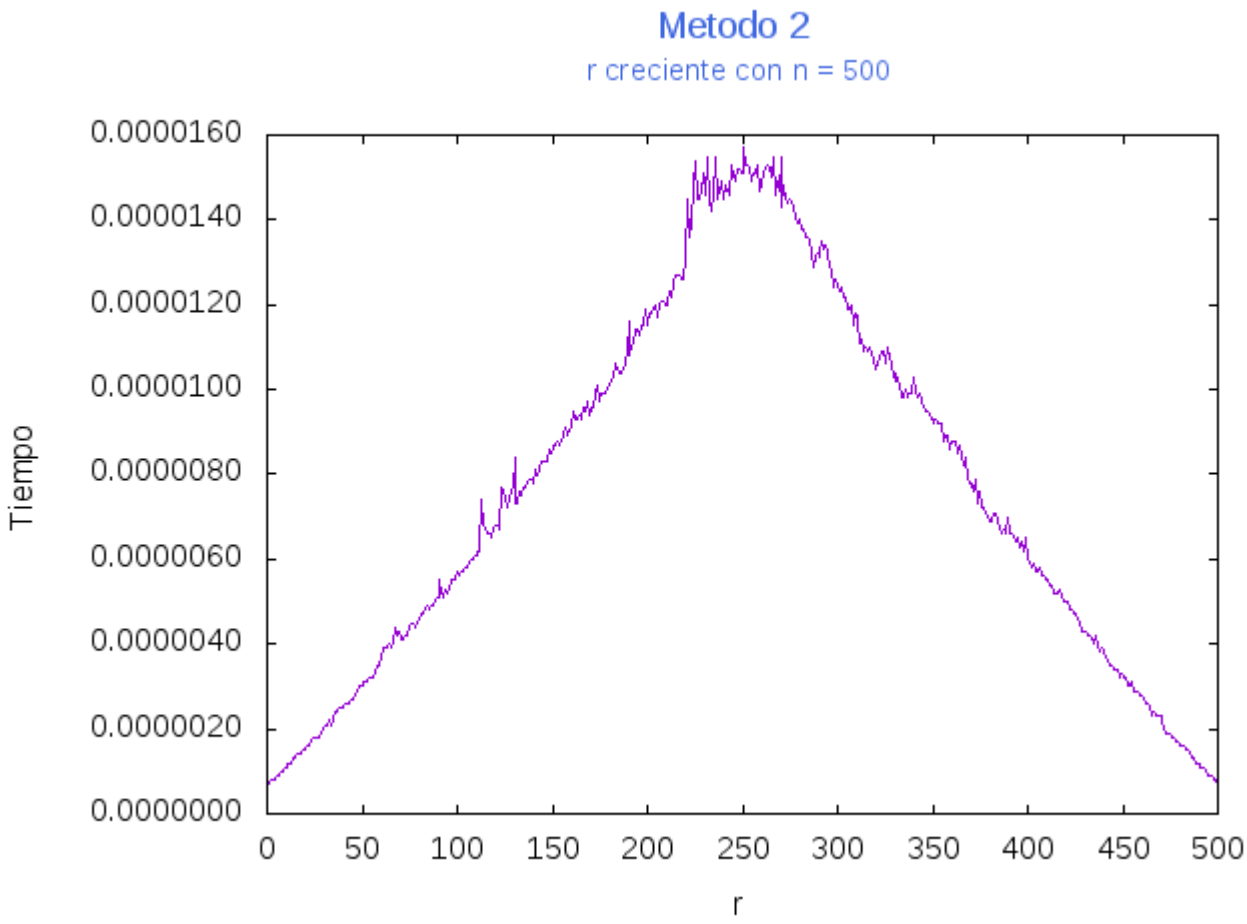
4.2.1 Para n fijo y r crece hasta n=100



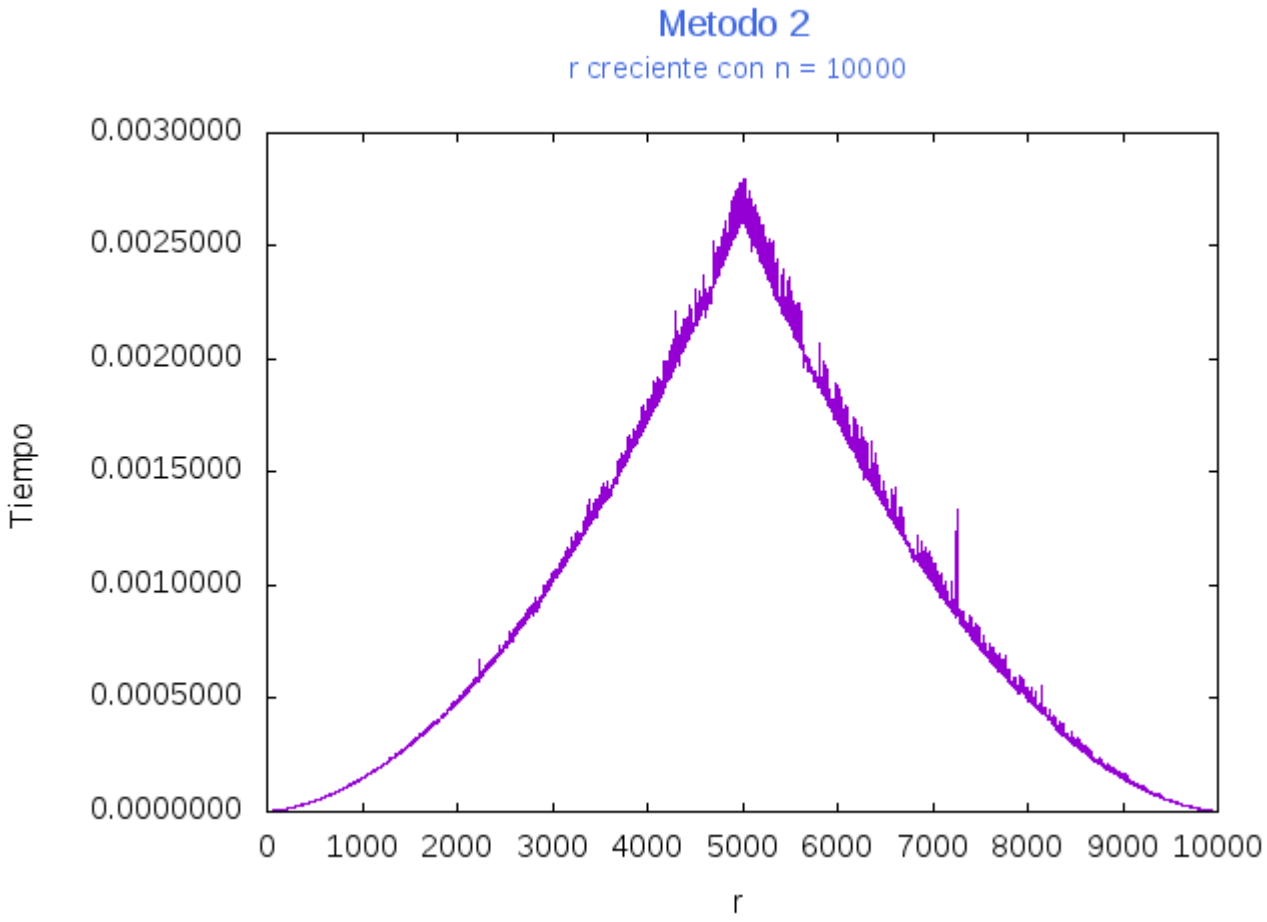
4.2.2 Para n fijo y r crece hasta n=300



4.2.3 Para n fijo y r crece hasta n=500

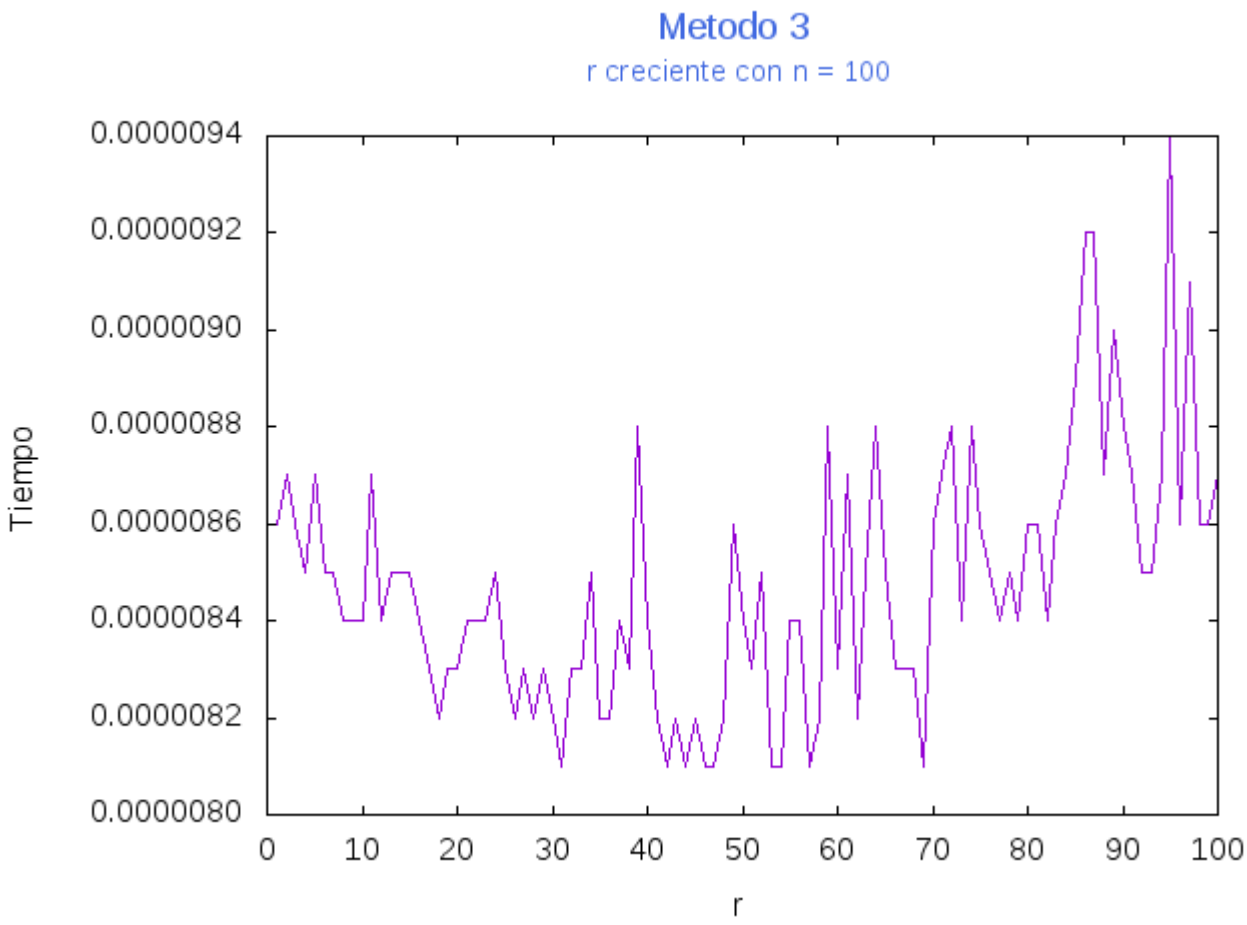


4.2.4 Para n fijo y r crece hasta n=10000

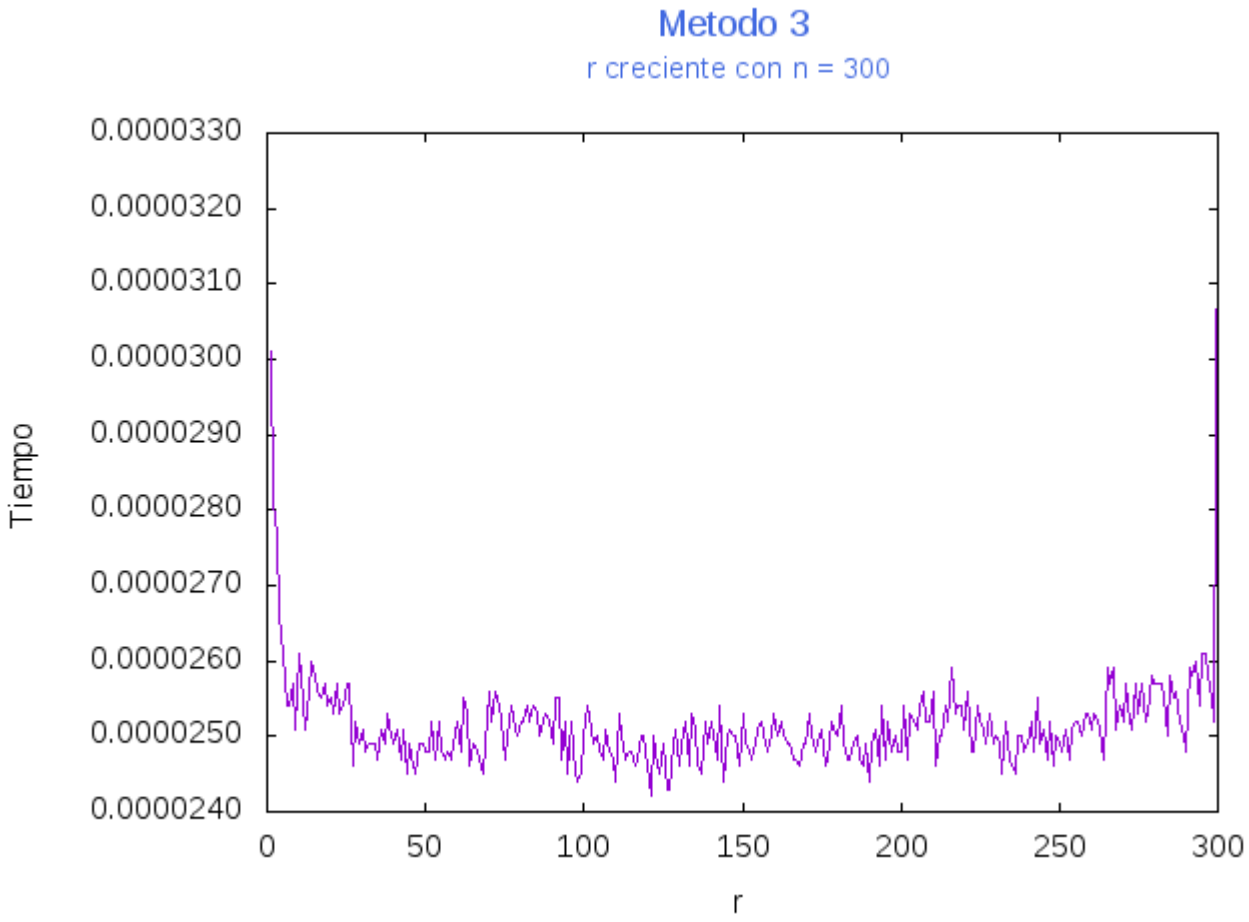


4.3 Estrategia tres

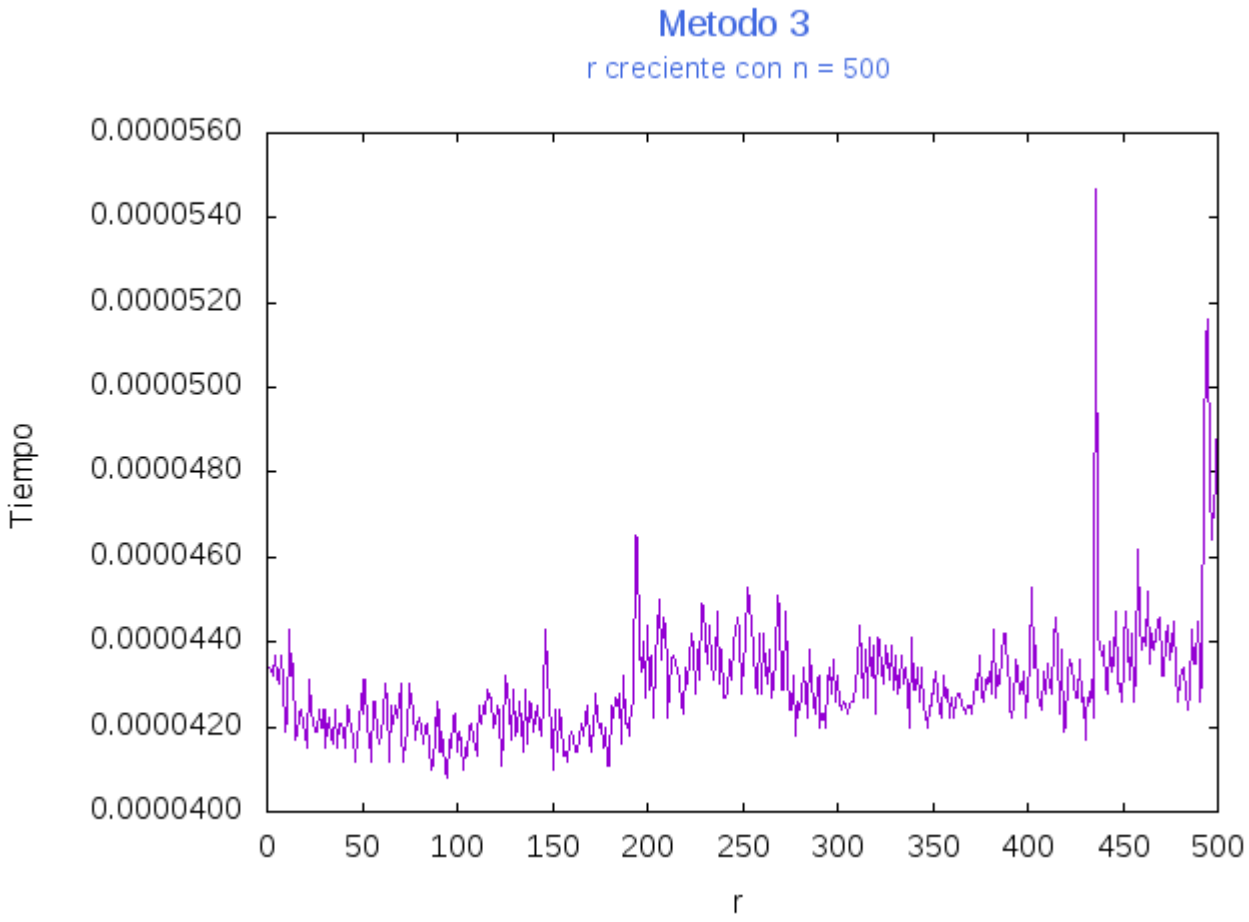
4.3.1 Para n fijo y r crece hasta n=100



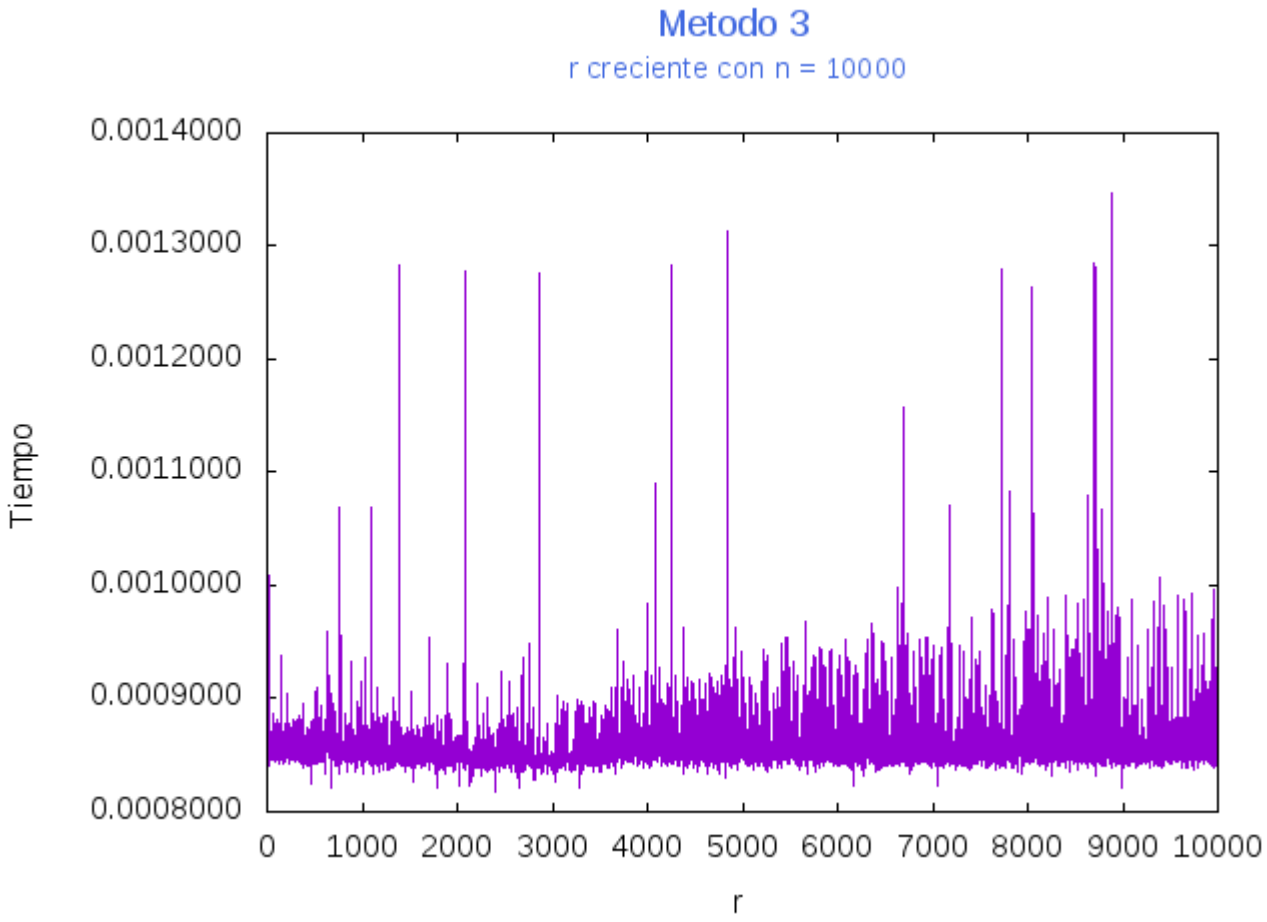
4.3.2 Para n fijo y r crece hasta n=300



4.3.3 Para n fijo y r crece hasta n=500

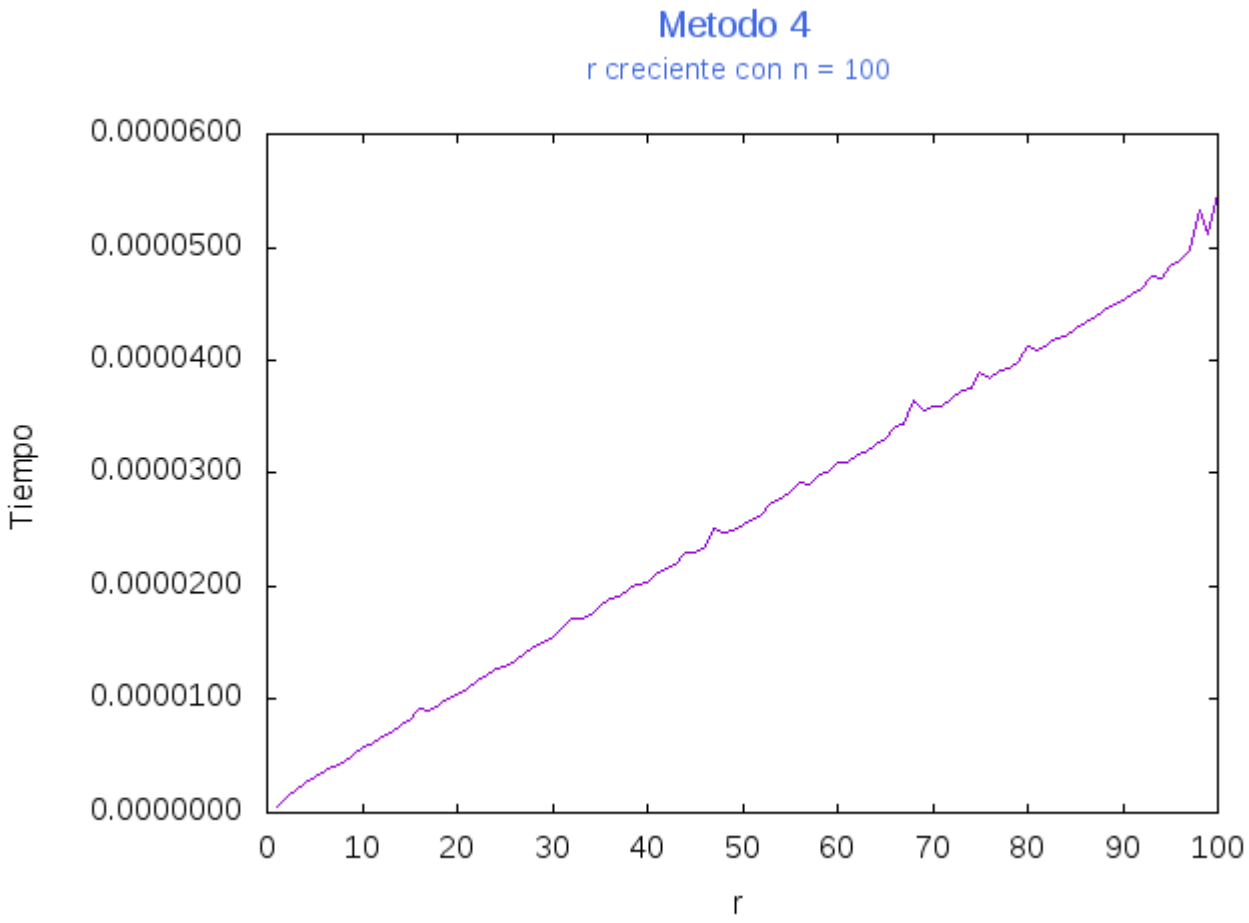


4.3.4 Para n fijo y r crece hasta n=10000

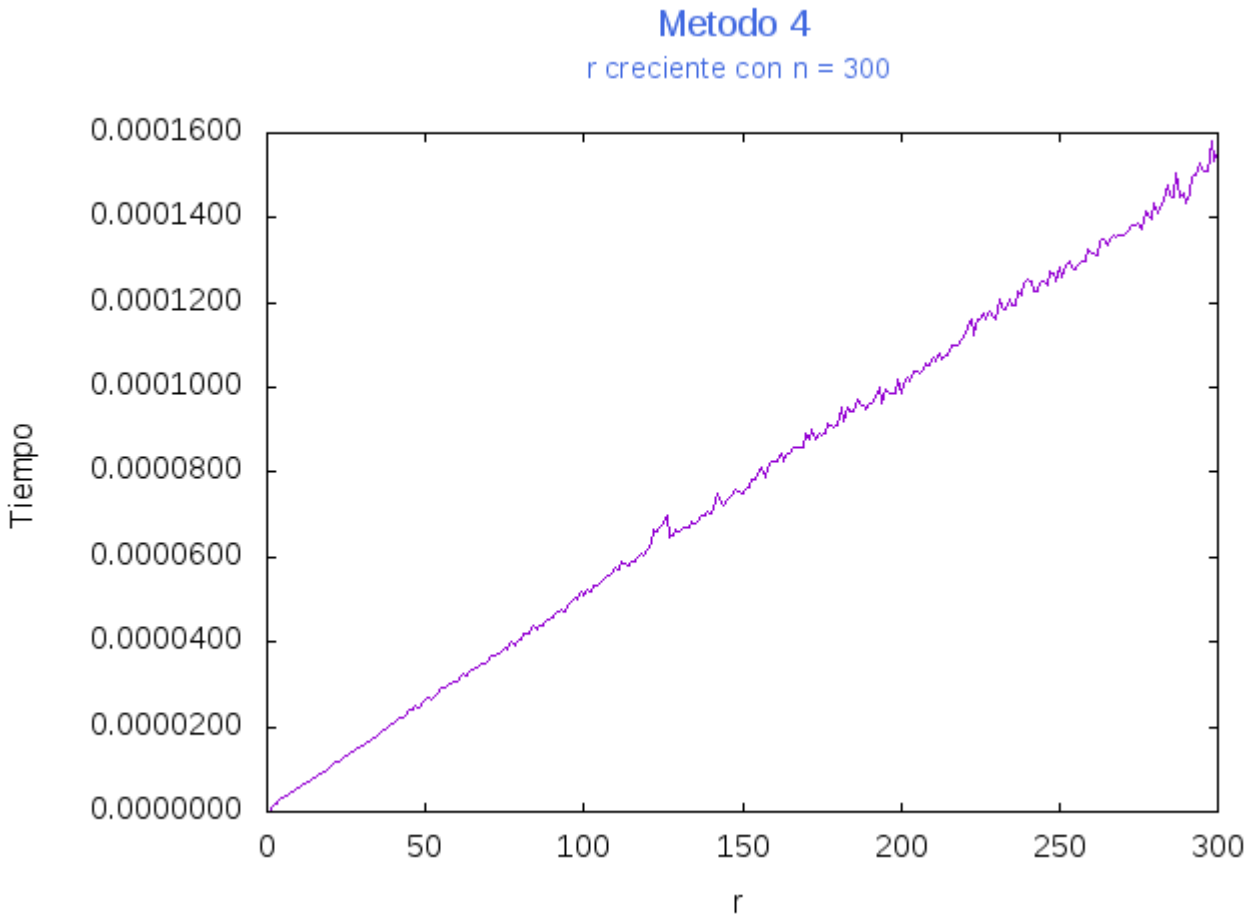


4.4 Estrategia cuatro

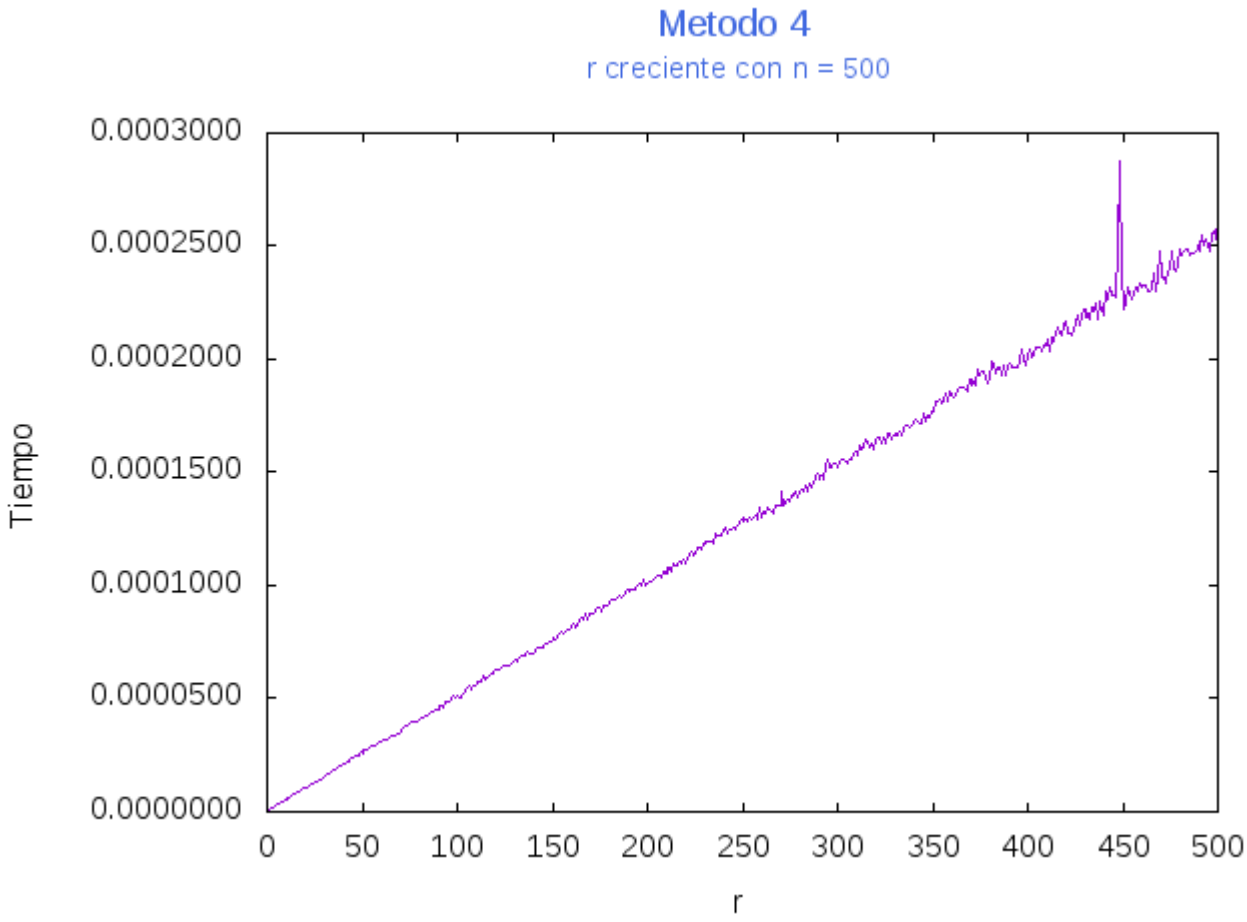
4.4.1 Para n fijo y r crece hasta n=100



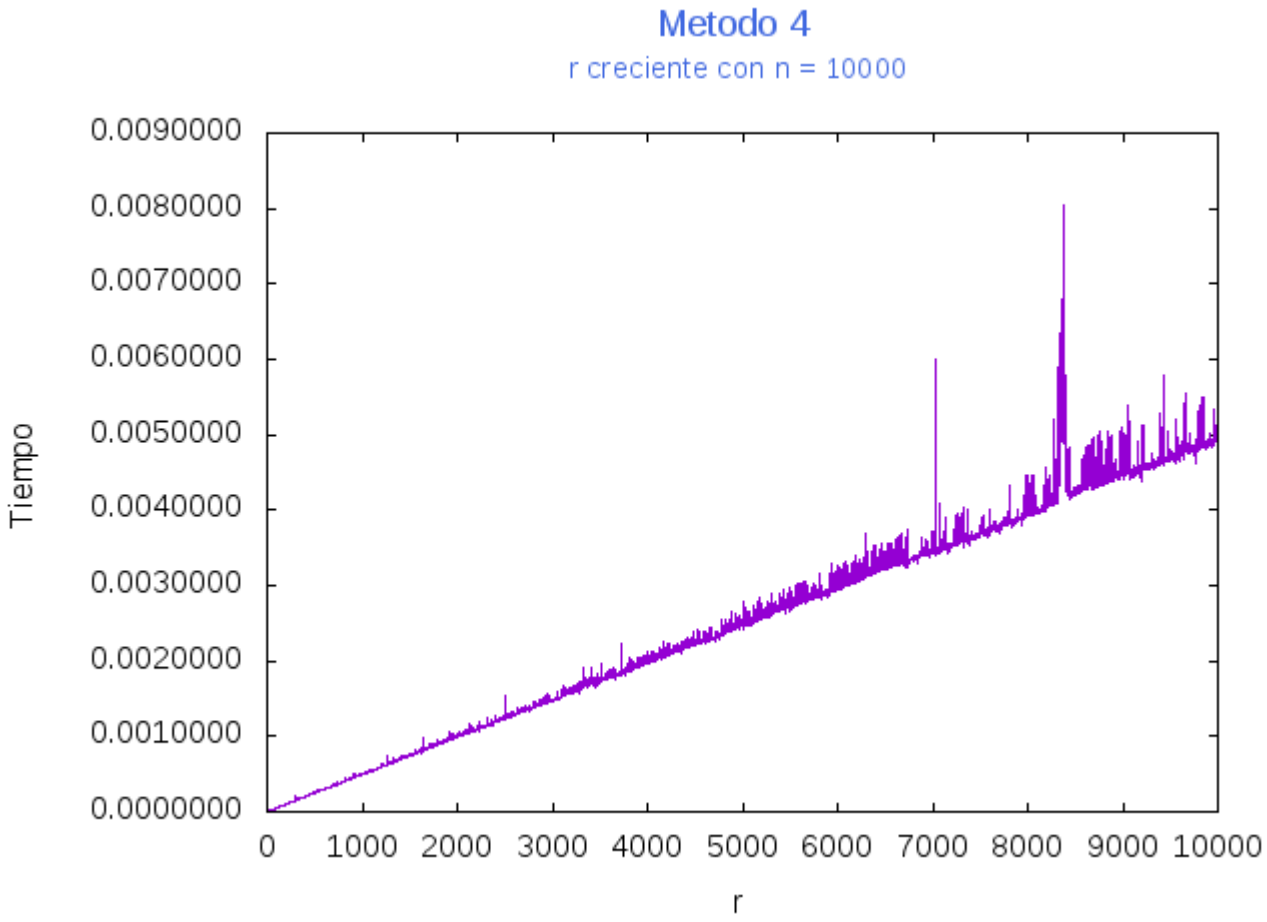
4.4.2 Para n fijo y r crece hasta n=300



4.4.3 Para n fijo y r crece hasta n=500

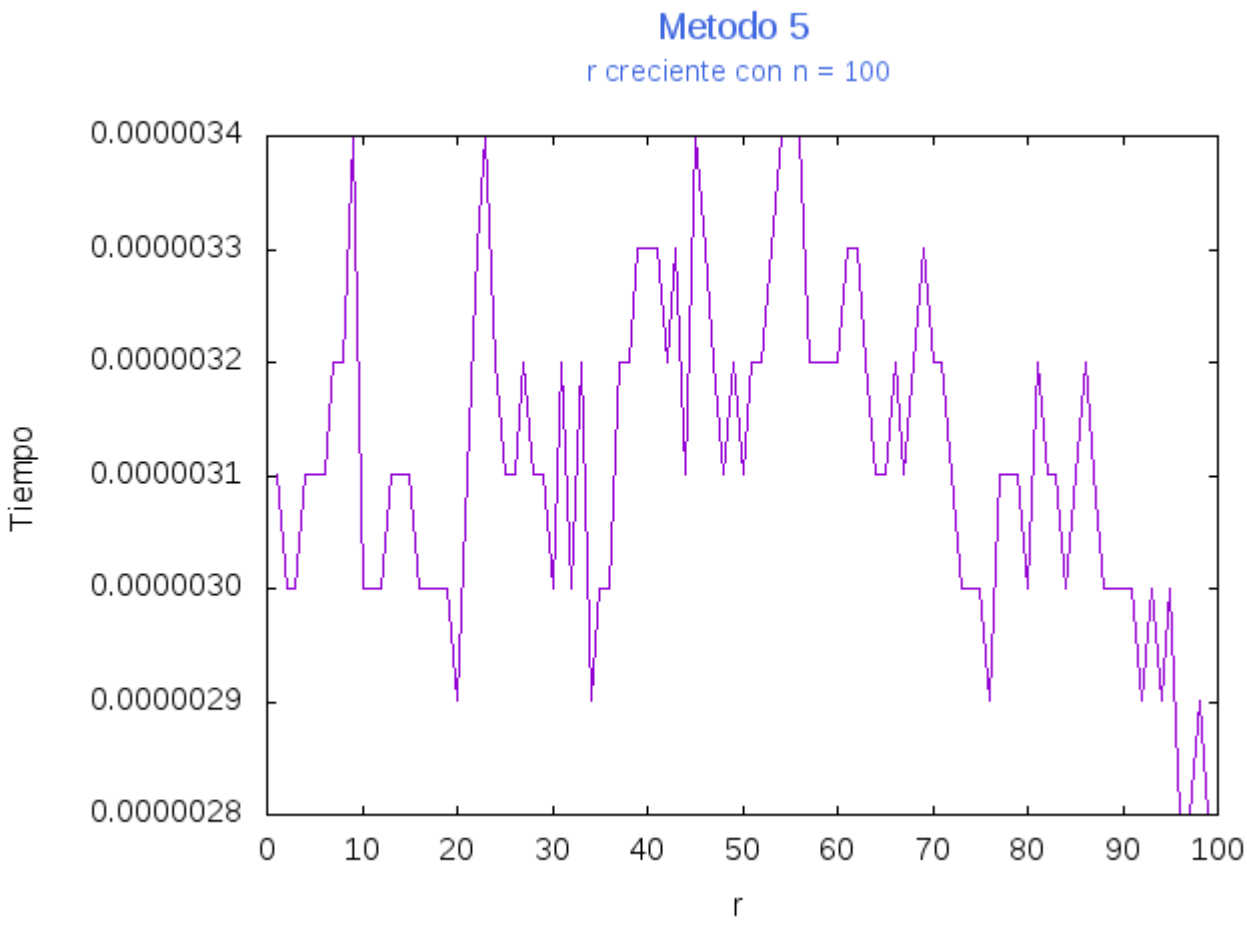


4.4.4 Para n fijo y r crece hasta n=10000

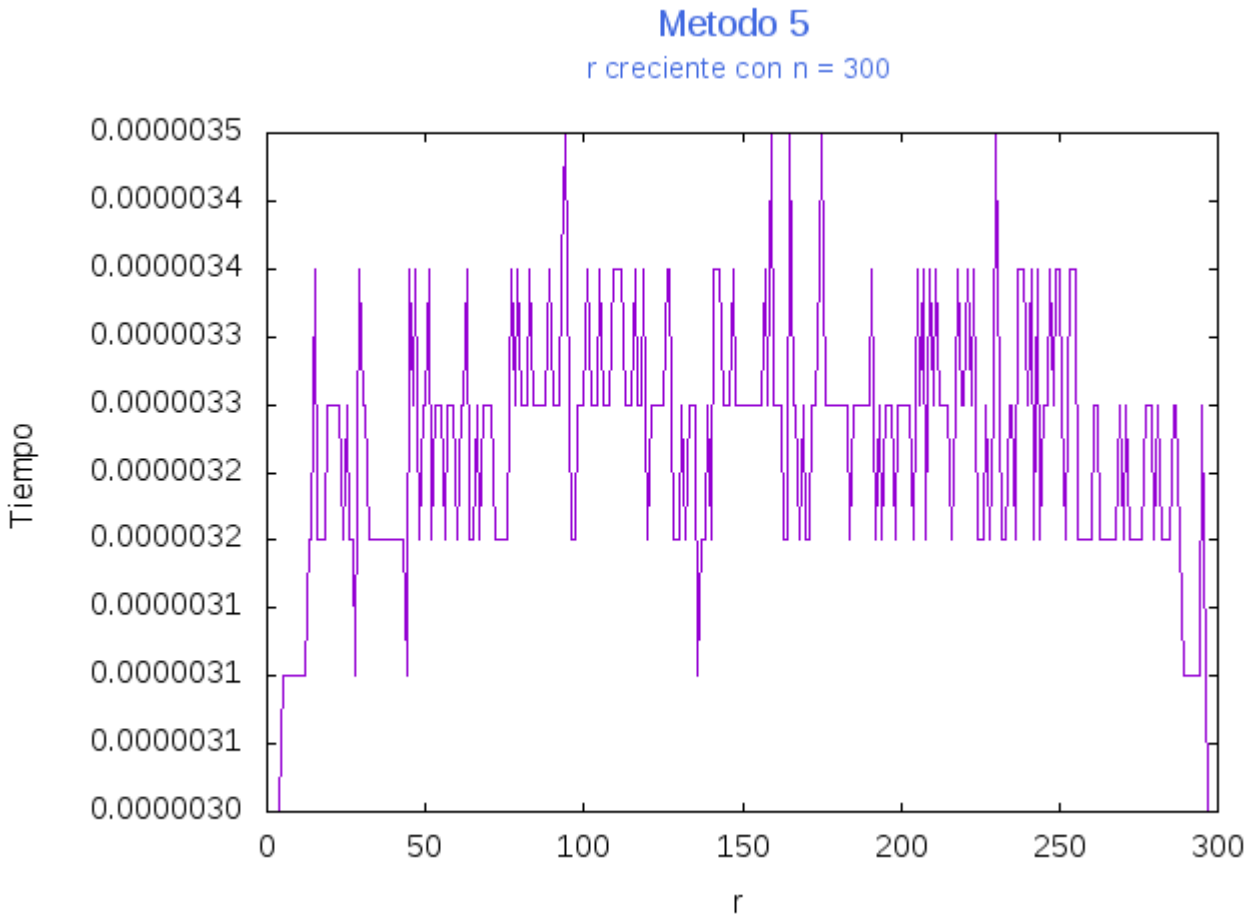


4.5 Estretegia cinco

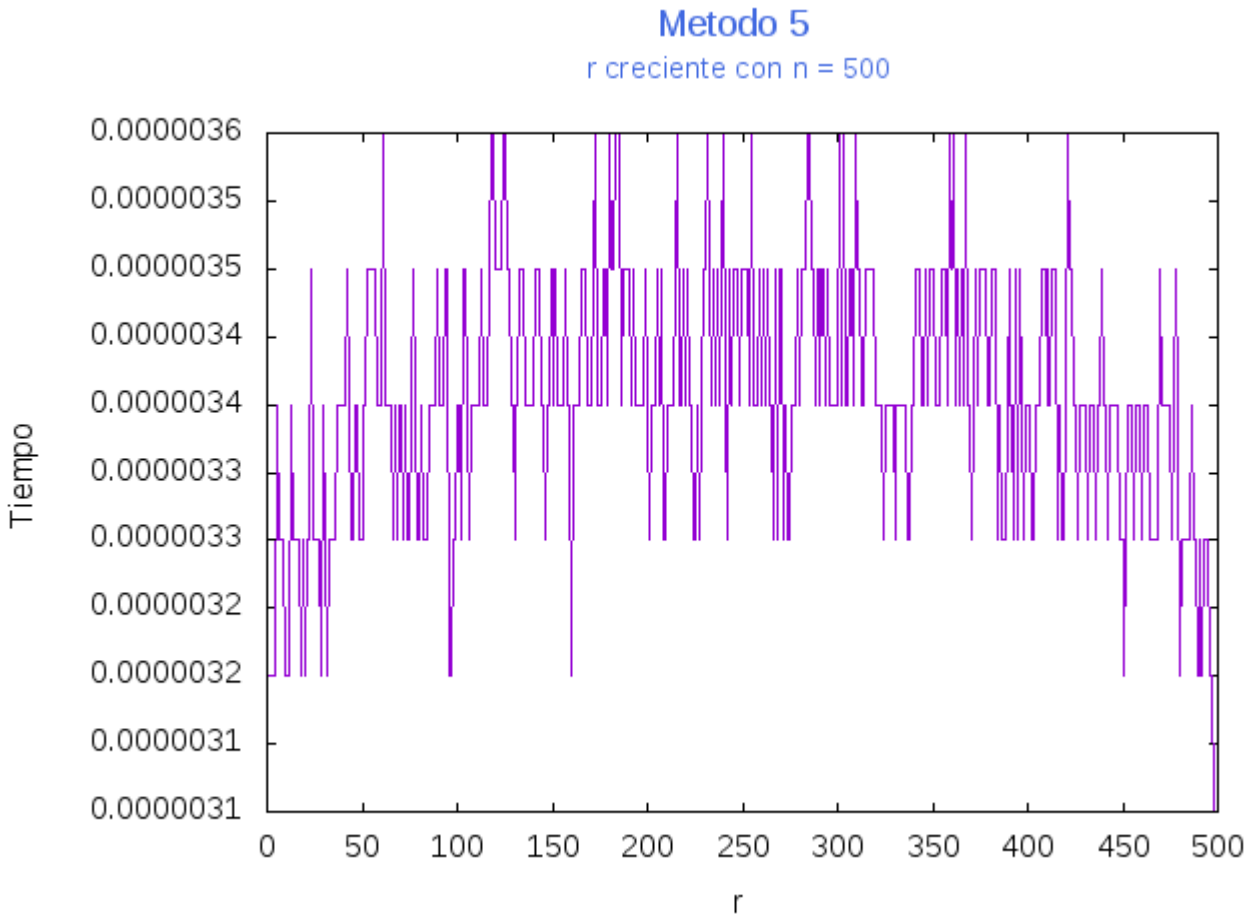
4.5.1 Para n fijo y r crece hasta n=100



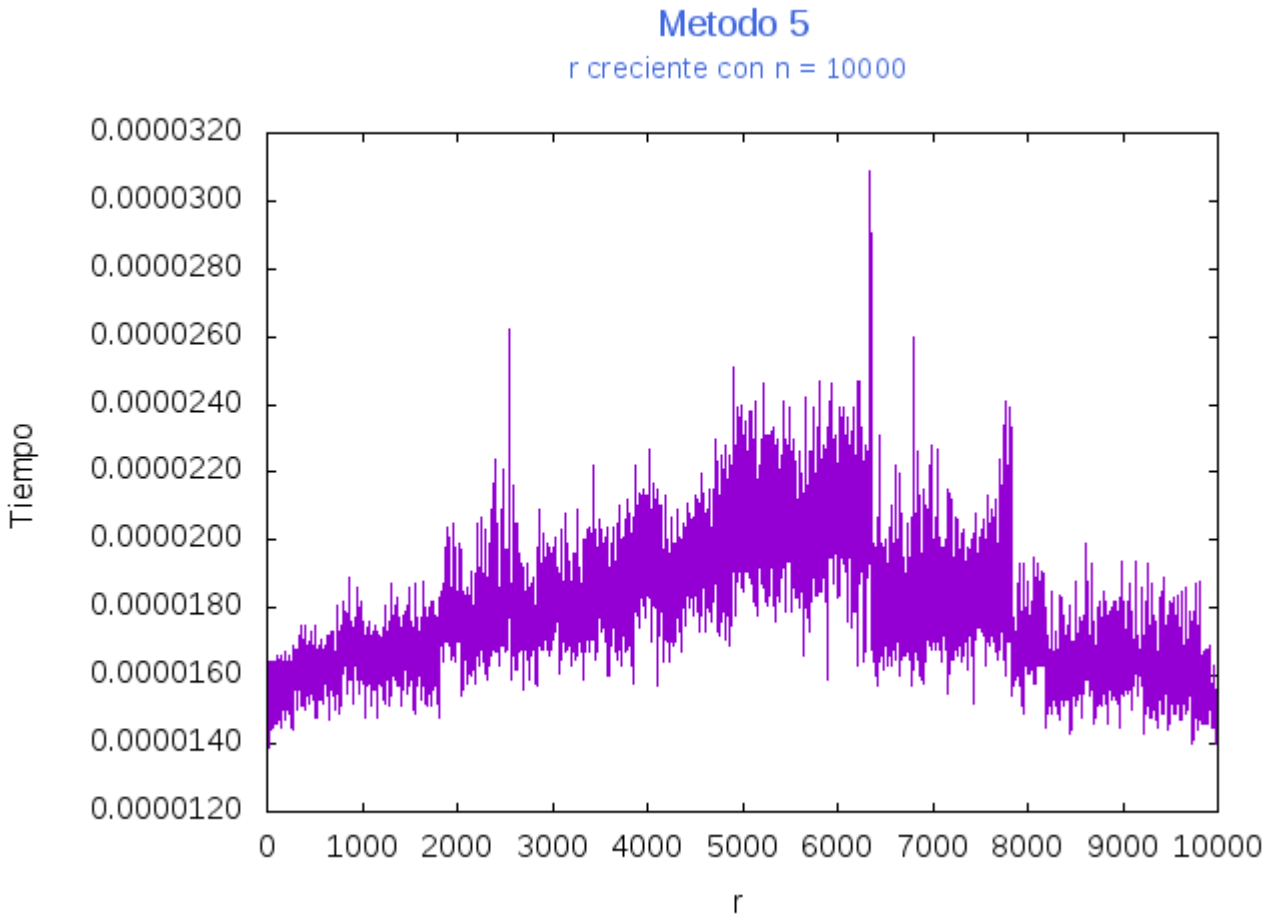
4.5.2 Para n fijo y r crece hasta n=300



4.5.3 Para n fijo y r crece hasta n=500



4.5.4 Para n fijo y r crece hasta n=10000



5 Conclusiones

Gracias a los graficos se puede llegar a las sigueitnes conclusiones:

- La estrategia uno es eficiente cuando se trata de combinatorias en donde el n y r sean cercanos, pero aun así, fue la estrategia que tuva más costes de tiempo.
- La estrategia dos es eficiente cuando se trata de combinatorias en donde el n y r sean lejanos entre si.
- La estrategia tres tiene un coste superior a la estrategia dos pero el tiempo es parejo independiente de los valores de n y r.
- La estrategia cuatro es eficiente para valores de r pequeños pero muy costosa para valores de r cercano al n, llegando a superar a la estrategia uno en lo que es el coste de tiempo.
- La estrategia cinco es la más eficiente en general, aunque tiene más coste en cuando n y r son cercanos entre sí pero aun sigue es más eficiente que el resto.