Práctica 2: Algoritmos Divide y Vencerás

Irene Huertas González
Javier Alcántara García
César Muñoz Reinoso

Índice

Objetivo de la practica

El objetivo de esta practica es apreciar la utilidad de la tecnica Divide y Venceras para resolver problemas de forma mas eficiente que otras alternativas mas sencillas o directas.
Para ello se nos ha pedido que desarrollemos dos problemas usando tanto un algoritmo sencillo como otro que implementa Divide y Venceras y ver las diferencias de ejecucion entre ambos.

Problema comun: Traspuesta de una matriz

Enunciado

Dada una matriz de tamaño $n = 2^k$, diseñar el algoritmo que devuelva la traspuesta de dicha matriz, siendo n el numero de elementos.

Como todos sabemos, la traspuesta de una matriz consiste en intercambiar las filas por las columnas

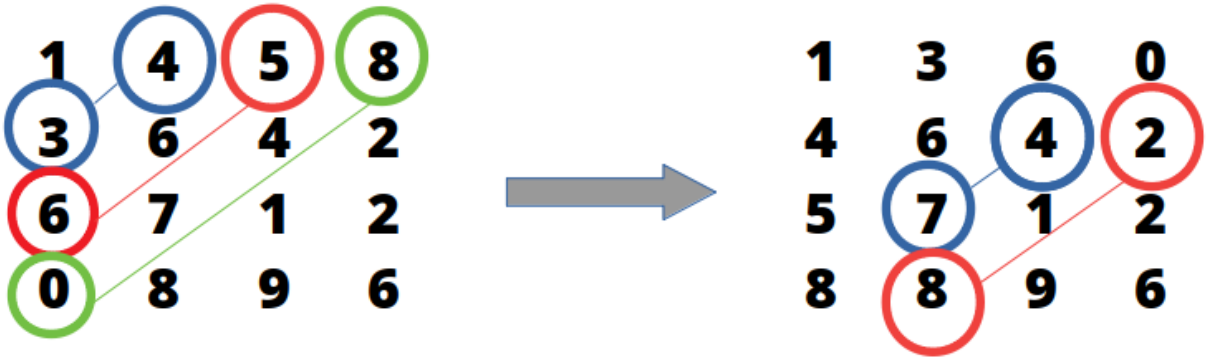
Traspuesta de una matriz. Algoritmo sencillo

Codigo

```
void swap (vector< vector<double> >& m,int i, int j) {
  double aux = m[i][j];
  m[i][j] = m[j][i];
  m[j][i] = aux;
}
void traspuesta (vector< vector<double> >& m, int n) {
  for (int i=0; i<n; i++){
    for (int j=i+1; j<n; j++){
      swap(m,i,j);
    }
  }
}
```

Traspuesta de una matriz. Algoritmo sencillo

¿Como funciona?



Recorre uno a uno los elementos de la diagonal superior y con cada uno va llamando a la funcion swap, que intercambia el valor de la posicion *ij* por el de la posicion *ji*

Traspuesta de una matriz. Algoritmo Divide y Venceras

Codigo

```
void swap (vector< vector<double> >& m,int finiA, int ciniA,
  int finiB, int ciniB, int dimen) {
  for (int i=0; i<dimen; i++){
    for (int j=0; j<dimen; j++) {
      int aux = m[finiA+i][ciniA+j];
      m[finiA+i][ciniA+j] = m[finiB+i][ciniB+j];
      m[finiB+i][ciniB+j] = aux;
}
  }
}
```

Traspuesta de una matriz. Algoritmo Divide y Venceras

Codigo

```
void traspuestaDyV (vector< vector<double> >& m, int fini, i
  if (fini < fult) {
    int fmed = (fini + fult) / 2;
    int cmed = (cini + cult) / 2;
    traspuestaDyV (m, fini, fmed, cini, cmed); //A1
    traspuestaDyV (m, fini, fmed, cmed+1, cult); //A2
    traspuestaDyV (m, fmed+1, fult, cini, cmed); //A3
    traspuestaDyV (m, fmed+1, fult, cmed+1, cult); //A4
    swap (m, fmed+1, cini, fini, cmed+1, fult - fmed);
  }
}
```

Traspuesta de una matriz. Algoritmo Divide y Venceras

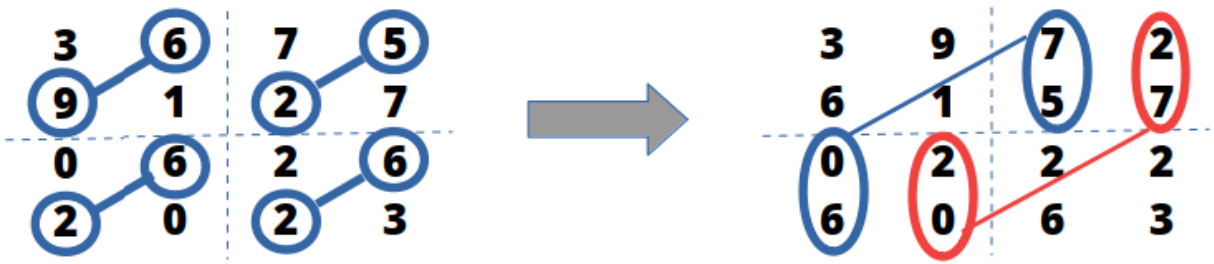
¿Como funciona?

3	6	7	5	3	5	6	2
9	1	2	7	0	9	3	6
0	6	2	6	1	8	7	9
2	0	2	3	7	5	9	2
2	8	9	7	3	6	1	2
9	3	1	9	4	7	8	4
5	0	3	6	1	0	6	3
2	0	6	1	5	5	4	7

En primer lugar subdivide la matriz en cuadrantes de tamaño n/2

Traspuesta de una matriz. Algoritmo Divide y Venceras

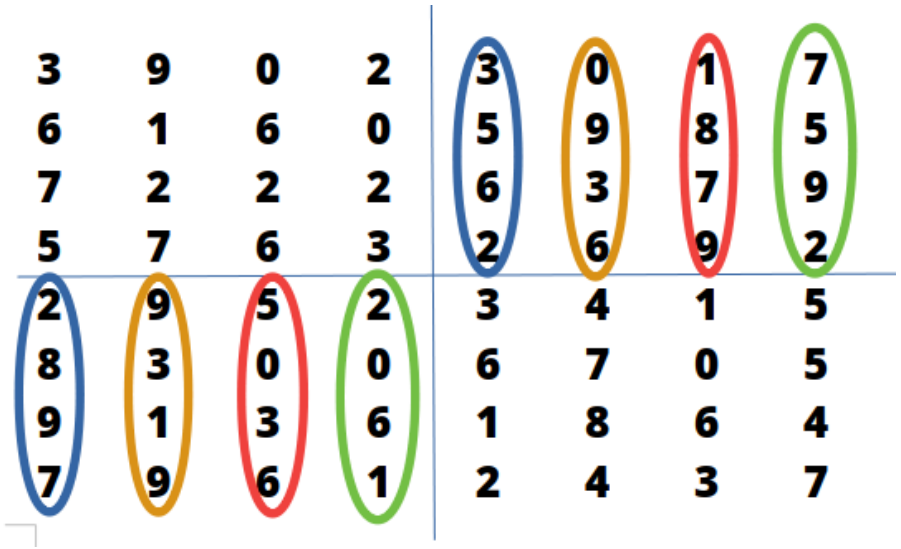
A continuacion empieza por el primer cuadrante (arriba a la izquierda) y lo subdivide hasta tener matrices base (un elemento por cuadrante) y les hace su traspuesta (imagen izquierda)



Una vez tiene la traspuesta de las submatrices, intercambia los cuadrantes que no contienen la diagonal principal: el segundo y el tercero (imagen de la derecha)

Traspuesta de una matriz. Algoritmo Divide y Venceras

Asi se tendria la traspuesta del primer cuadrante de la primera division que se hizo de la matriz original. Se siguen exactamente los mismos pasos con los otros tres cuadrantes y, cuando ya tenemos todos los cuadrantes con su traspuesta, se vuelven a intercambiar los cuadrantes 2 y 3 de la matriz



Traspuesta de una matriz. Analisis teorico

Algoritmo sencillo

Encontramos con dos bucles for anidados tras los cuales se llama a la función swap que son operaciones elementales.

$$\sum_{i=0}^{n}(\sum_{i+1}^{n}(11)) = \sum_{i=0}^{n}(11(n-i-1)) = 11 * (n^2 - n(i+1))$$

Por lo tanto la eficiencia teórica es de orden $\mathbf{O}(n^2)$

Algoritmo DyV

Hay dos operaciones de orden 1 seguidas de 4 de orden O(n/2) (los cuatro cuadrantes que llaman recursivamente a la funcion) y la llamada a swap de orden O(n)

$$T(n) = 4T(n/2) + n \in O(n^2)$$

Luego la eficiencia es tambien $\mathbf{O}(n^2)$

Traspuesta de una matriz. Analisis empirico

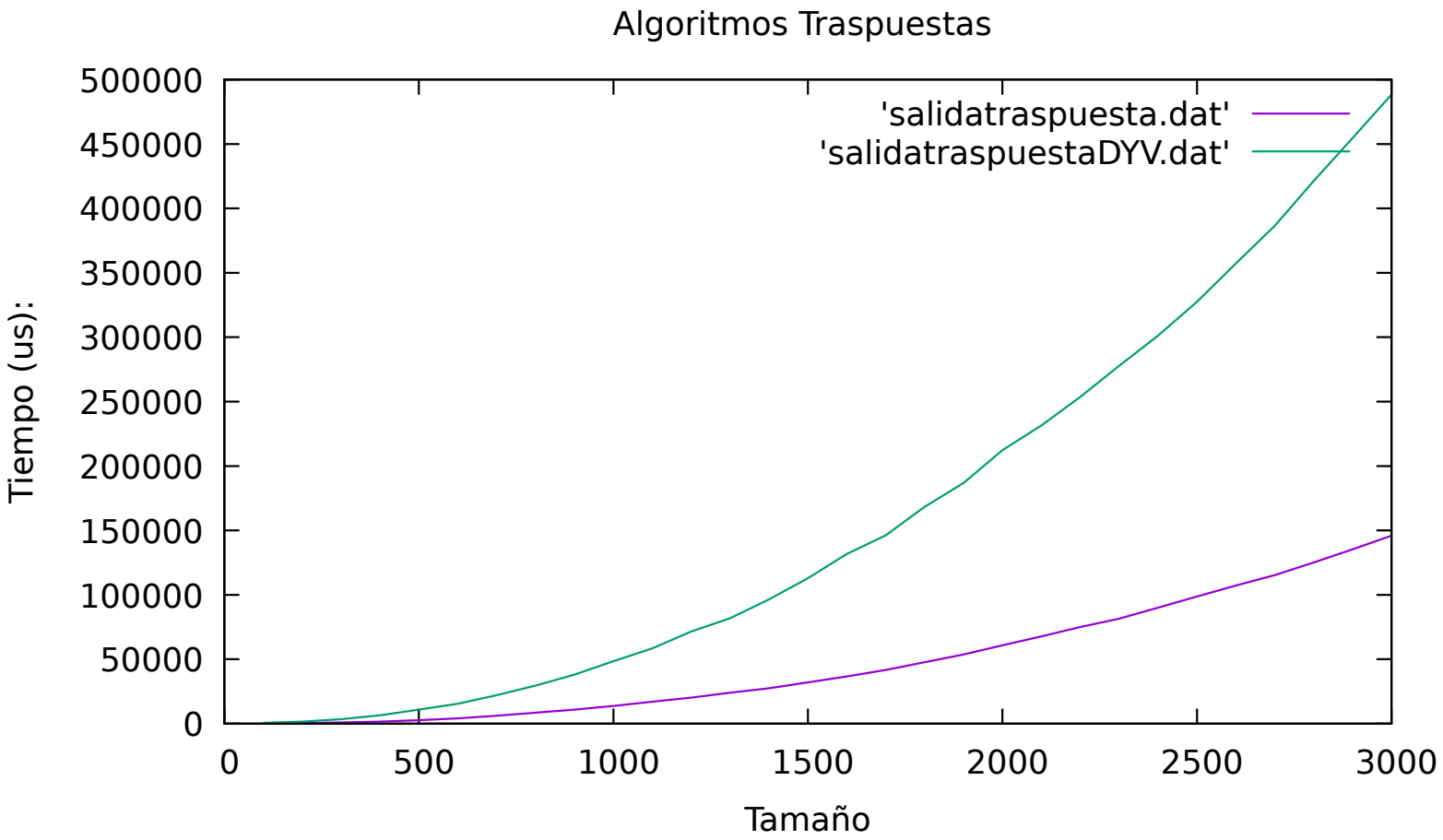


Figura: Traspuesta sencilla y DyV de César

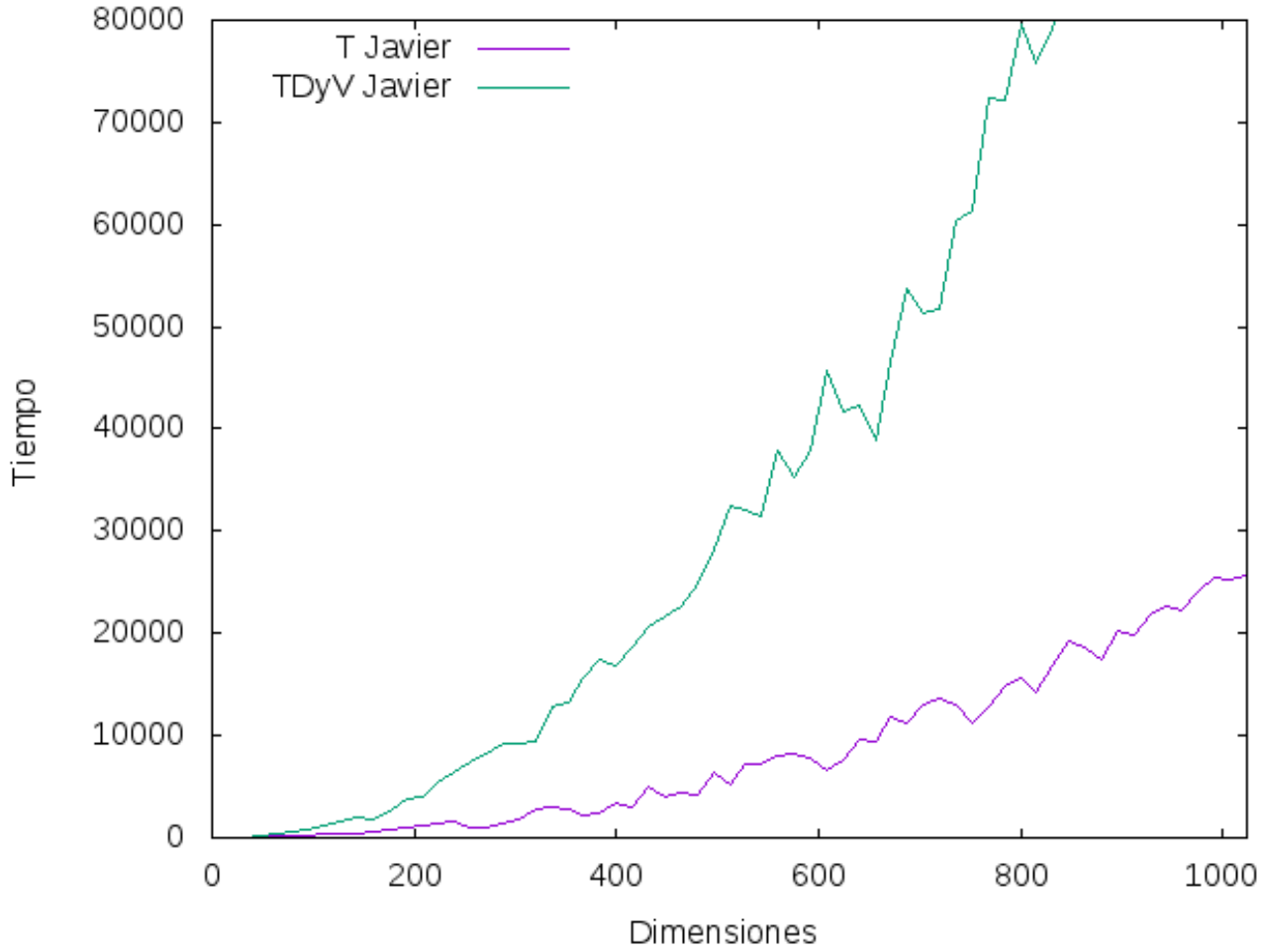


Figura: Traspuesta sencilla y DyV de Javier

Traspuesta de una matriz. Analisis empirico

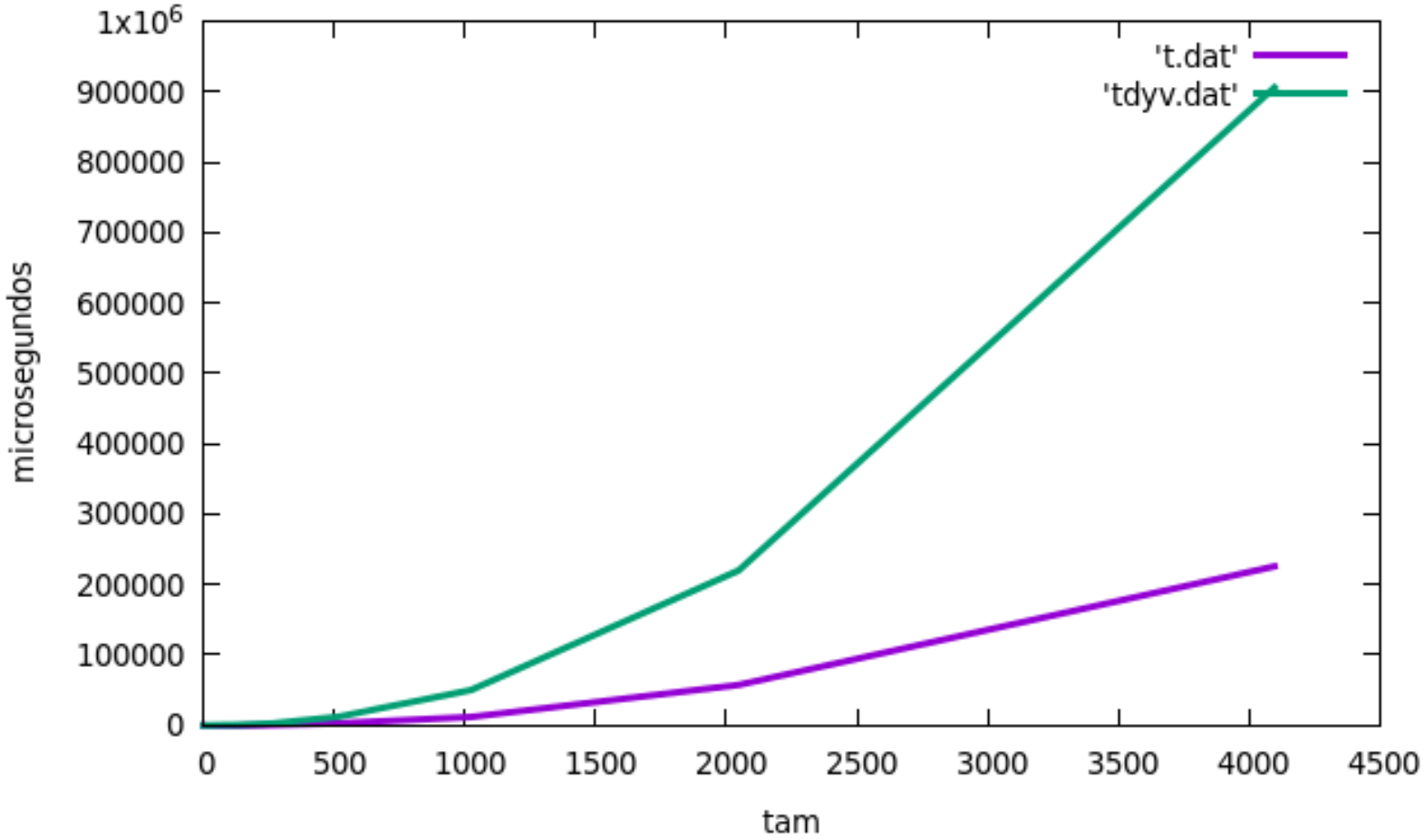
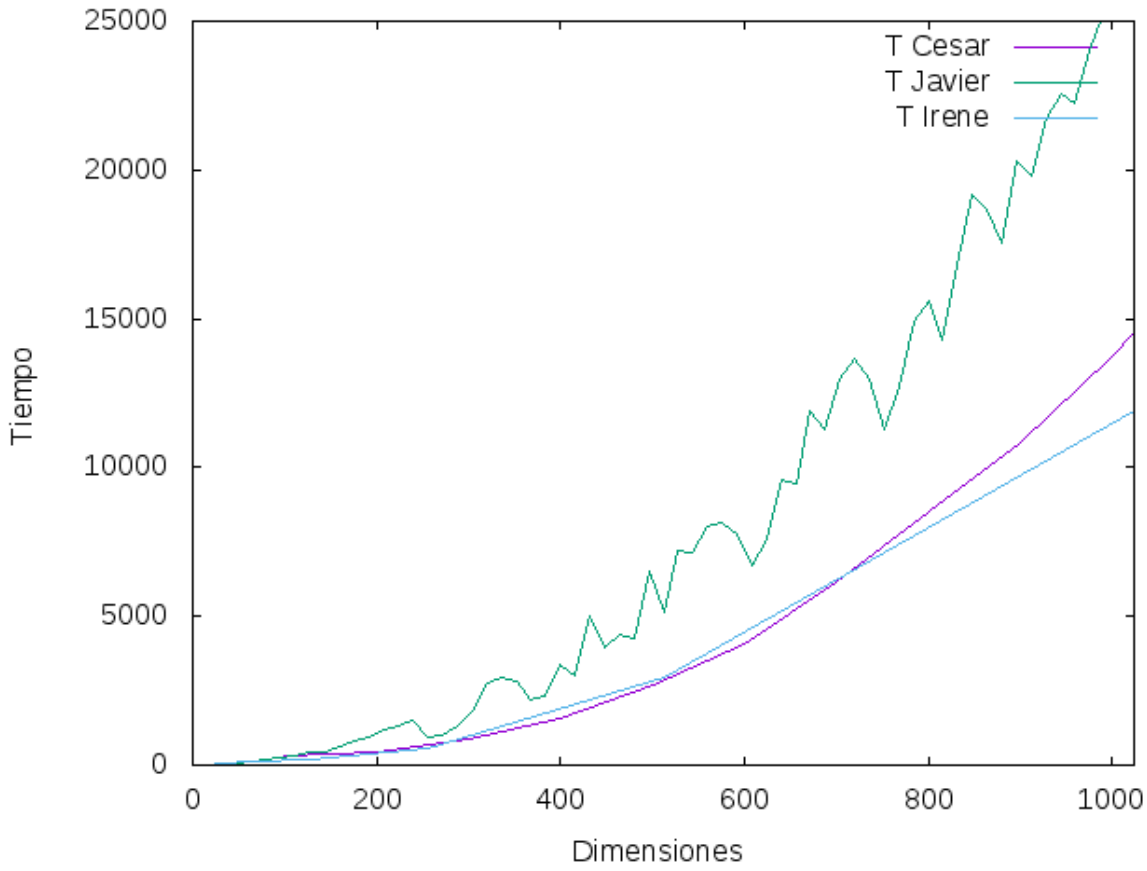


Figura: Traspuesta sencilla y DyV de Irene

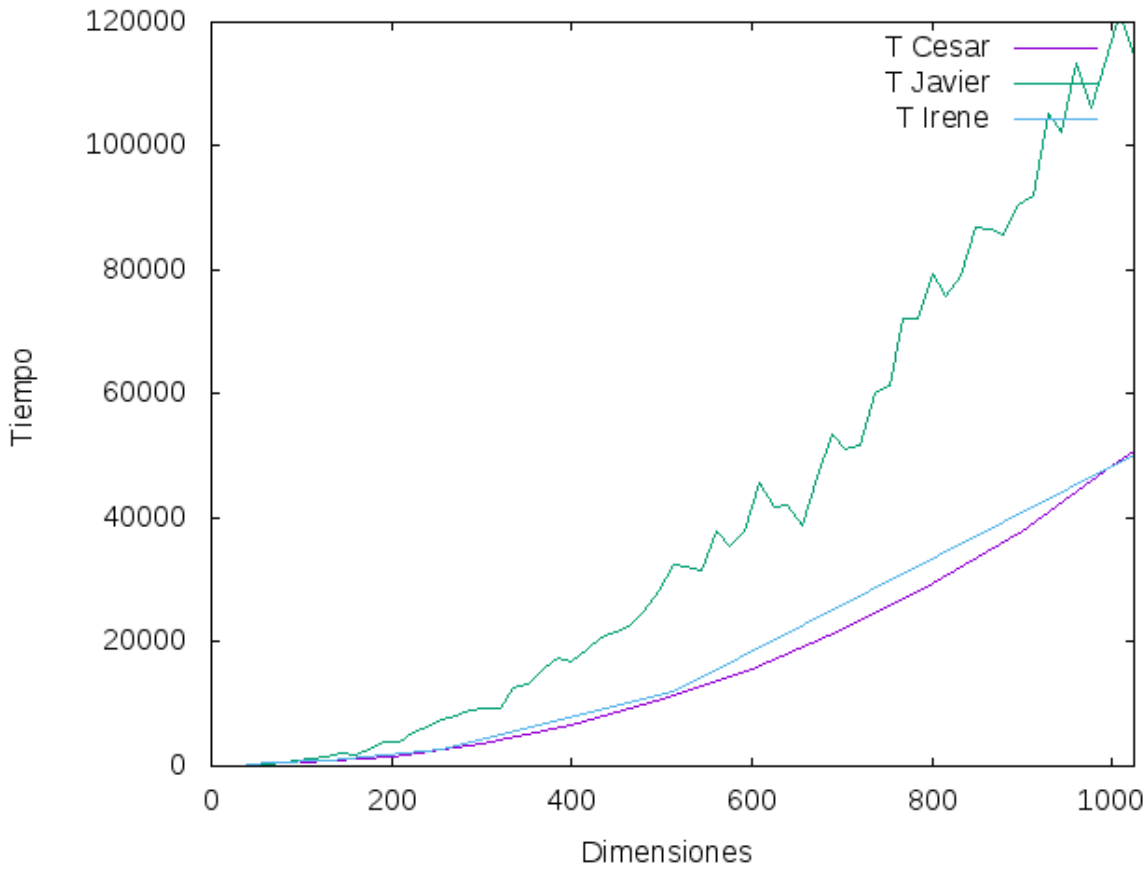
Traspuesta de una matriz. Analisis empirico entre ordenadores

Traspuesta sencilla



Traspuesta de una matriz. Analisis empirico entre ordenadores

Traspuesta DyV



Traspuesta de una matriz. Analisis hibrido

Hemos ajustado ambas con la funcion

$$f(x) = a0x^2 + a1x + a2$$

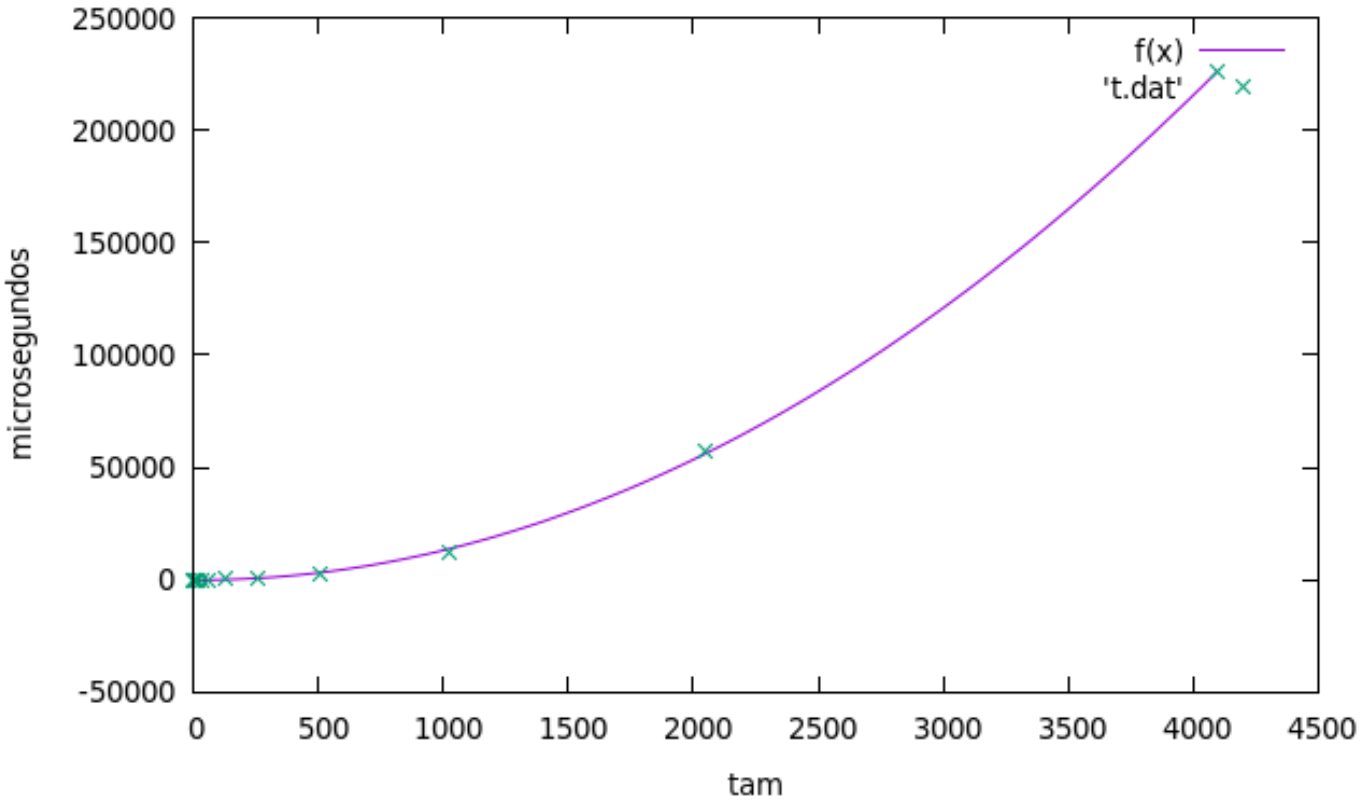


Figura: Hibrida traspuesta sencilla de Irene

Traspuesta de una matriz. Analisis hibrido

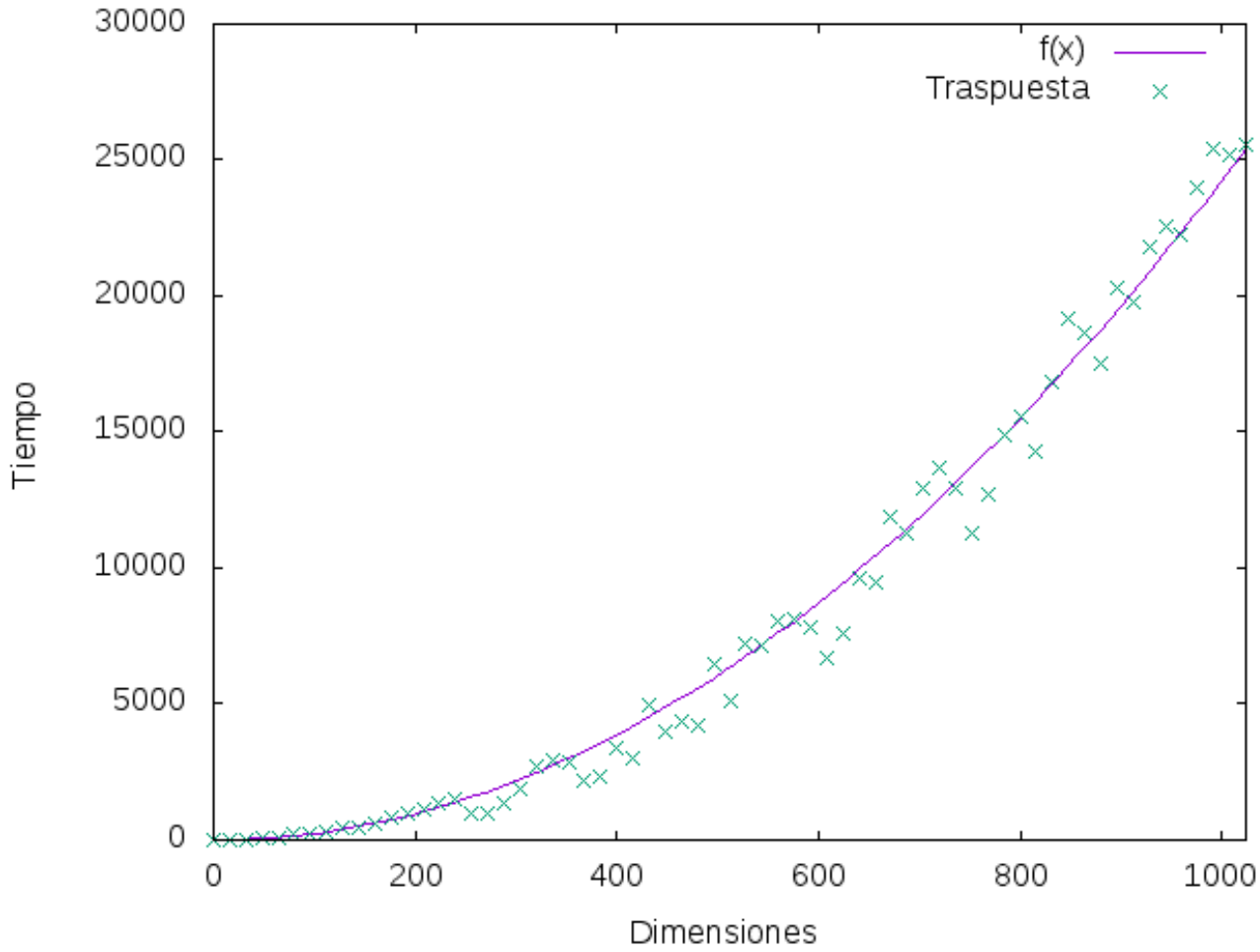


Figura: Hibrida traspuesta sencilla de Javi

Traspuesta de una matriz. Analisis hibrido

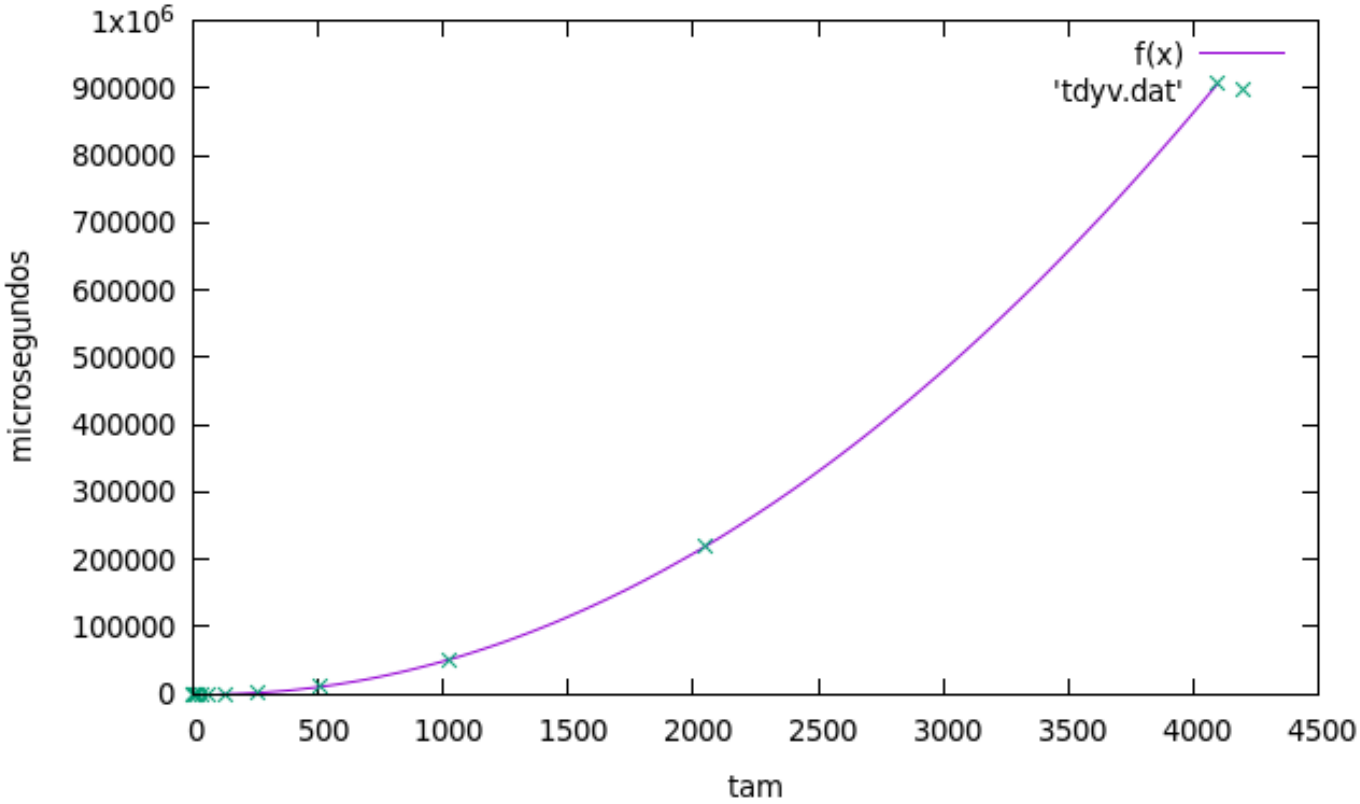


Figura: Hibrida traspuesta DyV de Irene

Traspuesta de una matriz. Analisis hibrido

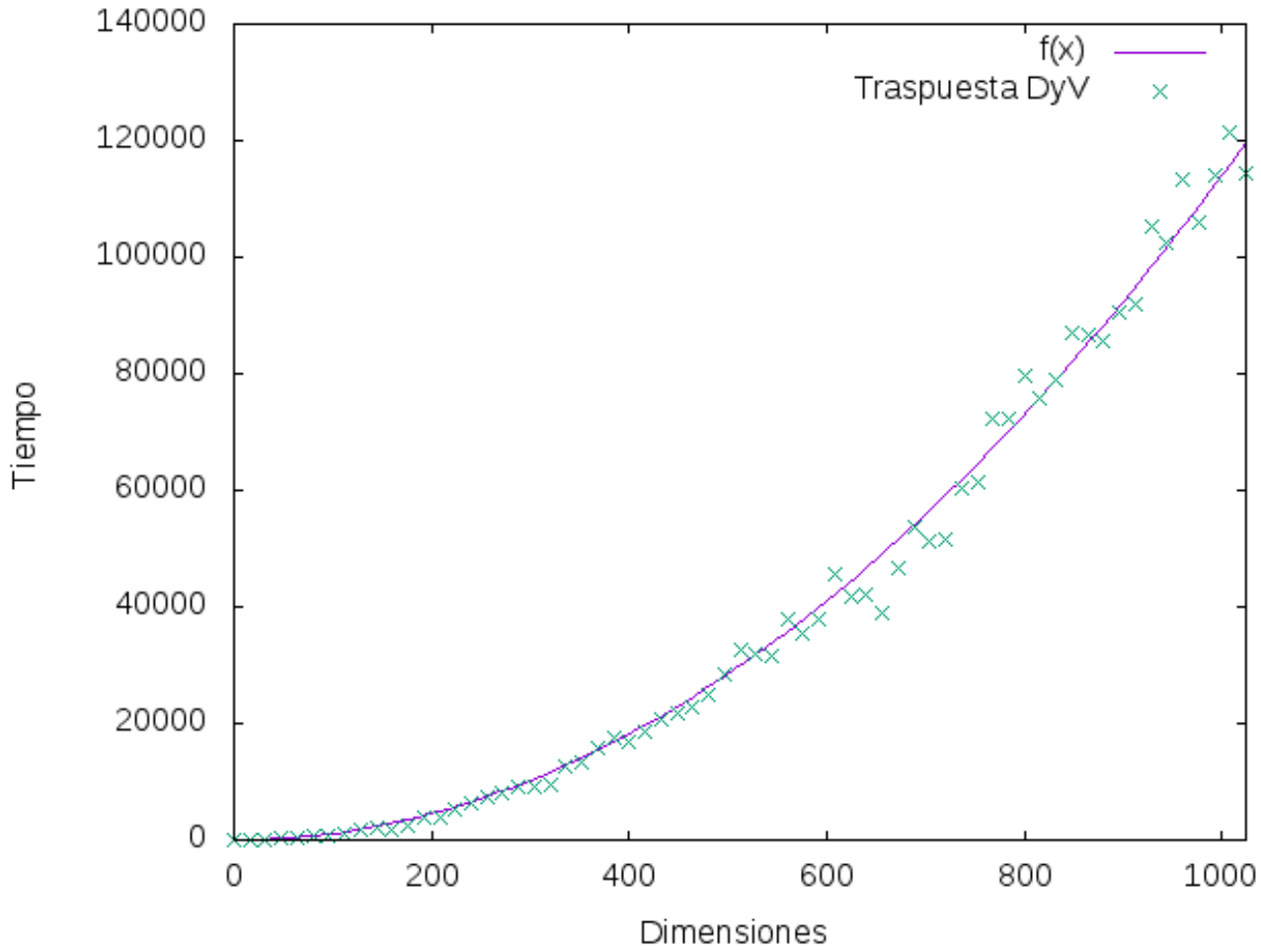


Figura: Hibrida traspuesta DyV de Javi

Problema asignado: Serie unimodal de numeros

Enunciado Sea un vector v de números de tamaño n, todos distintos, de forma que existe un indice p (que no es el primero ni el ultimo) tal que a la izquierda de p los números están ordenados de forma creciente y a la derecha de p están ordenados de forma decreciente. Es decir

$$\forall i, j \geq p,\ i < j \rightarrow v[i] > v[j]$$

De forma que el máximo se encuentra en la posición p.
El vector se rellena con el generador que nos facilitaron para la practica

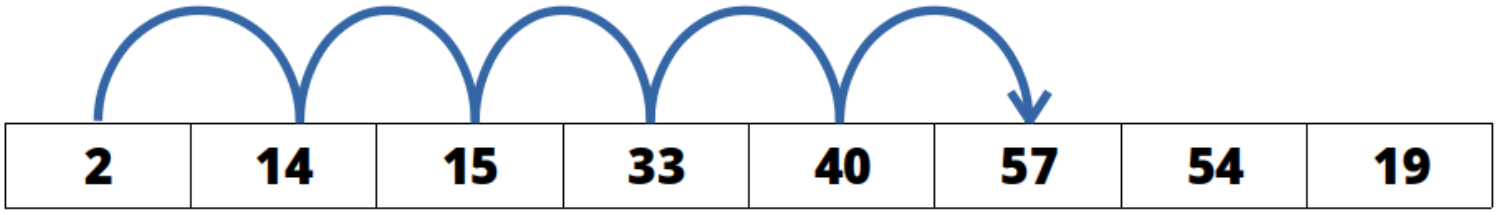
Serie unimodal. Algoritmo sencillo

Codigo

```
int unimodal(int  *T, int n){
  for(int i= 0; i < n; i ++){
    if(T[i]> T[i+1] && T[i]> T[i-1]){
    return i;
    }
  }
}
```

Serie unimodal. Algoritmo sencillo

¿Como funciona?



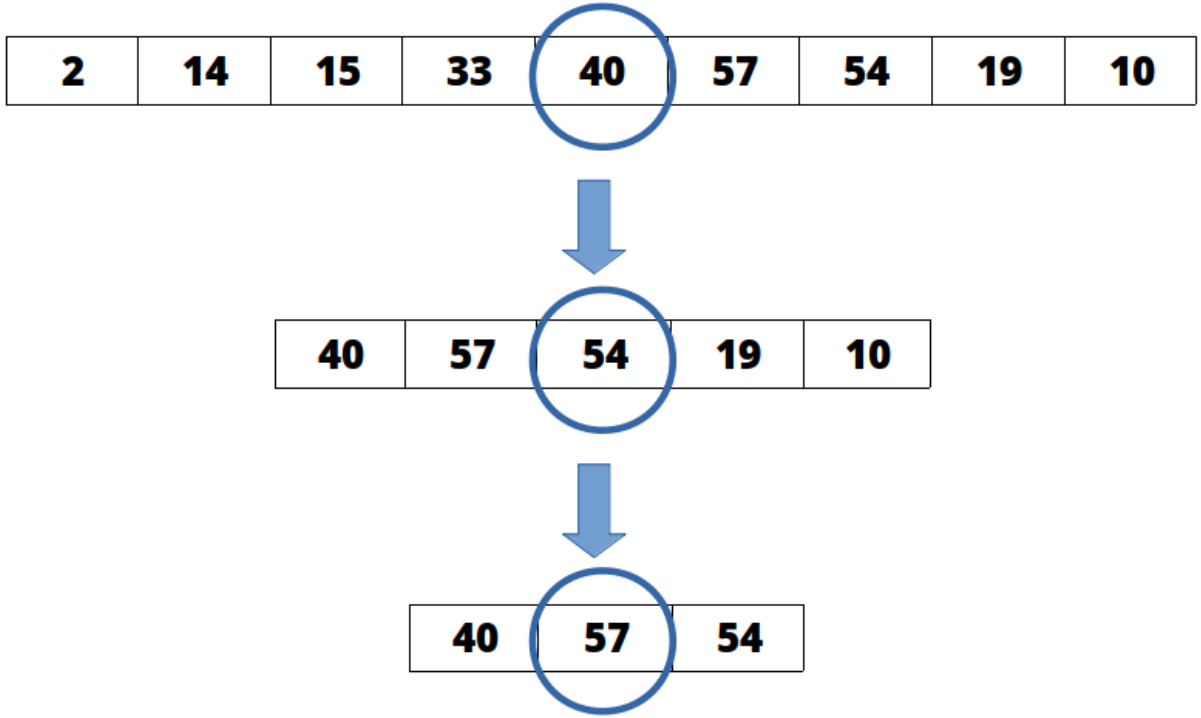
Recorre uno a uno los elementos del vector y va comprobando si se cumple la condicion de ser mayor que el anterior y a su vez mayor que el posterior. Cuando lo encuentra para y devuelve esa posicion.

Serie unimodal. Algoritmo Divide y Venceras

Codigo

```
int unimodal(int  *v, int ini, int fin){
  int med = (fin + ini)/2;
  if(v[med] > v[med-1] && v[med] > v[med + 1])
    return med;
  else if(v[med] > v[med-1])
    unimodal(v, med, fin);
  else
    unimodal(v, ini, med);
}
```

Serie unimodal. Algoritmo Divide y Venceras



Divide el problema en dos recursivamente. Va mirando si el numero que esta en mitad es menor que el de la posición siguiente o anterior para buscar por la derecha o izquierda respectivamente. Acaba cuando el elemento de en medio sea el mayor.

Serie unimodal. Analisis teorico

Algoritmo sencillo

Es un bucle for que recorre el vector elemento a elemento en busca del pico (elemento maximo)

$$\sum_{0}^{n}(max(a,0)) \quad a = \ numero\ de\ operaciones\ elementales\ del\ 'if'$$

El orden de eficiencia será **O(n)**

Algoritmo DyV

Aparecen llamadas recursivas.

Tiempo de ejecución: $T(n) = T(n/2) + a$

Tras calcular la ecuacion cartesiana nos queda

$$T(n) = c1\,logn + c2$$

El orden de complejidad es **O(logn)**

Serie unimodal. Analisis empirico

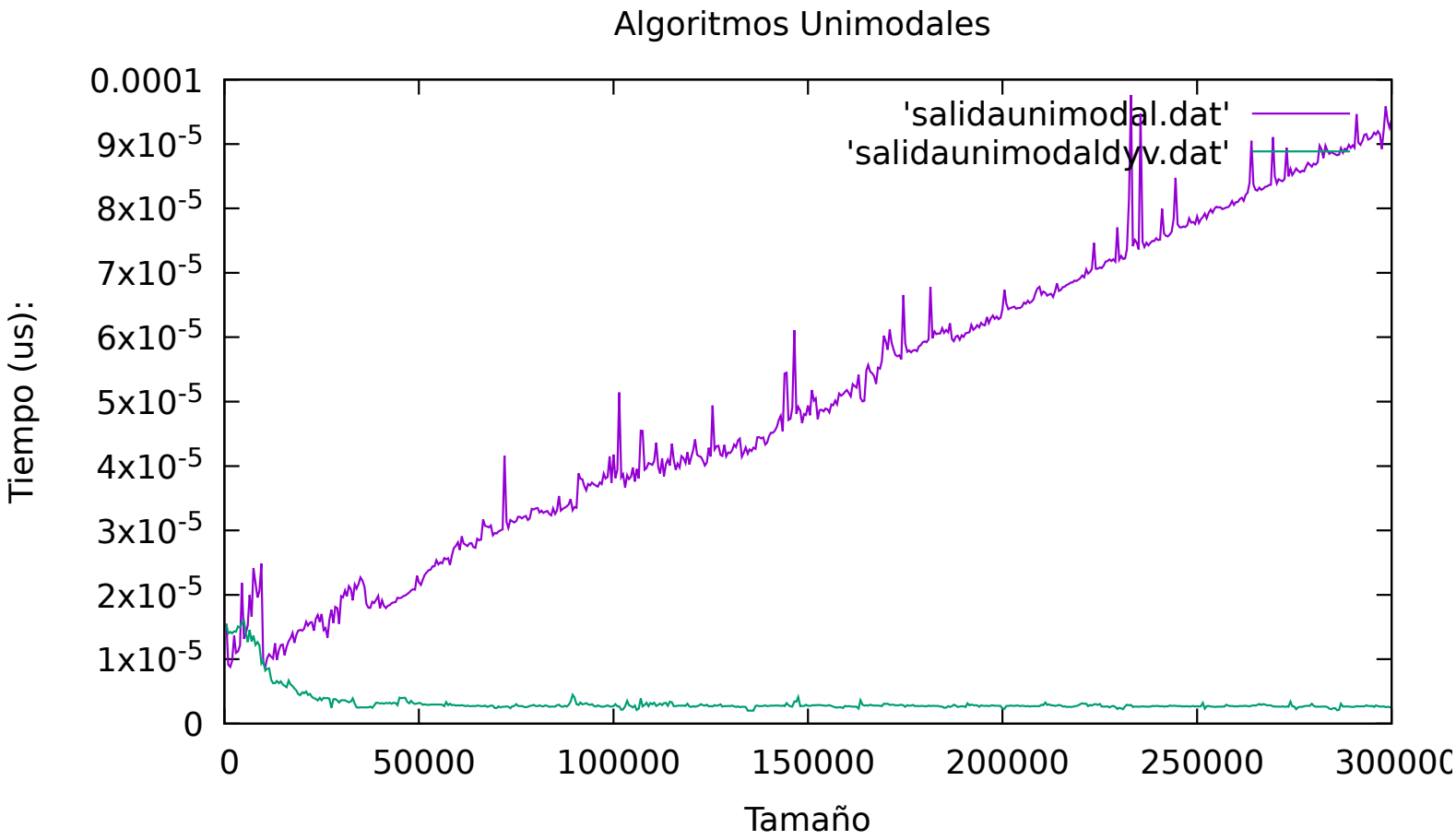
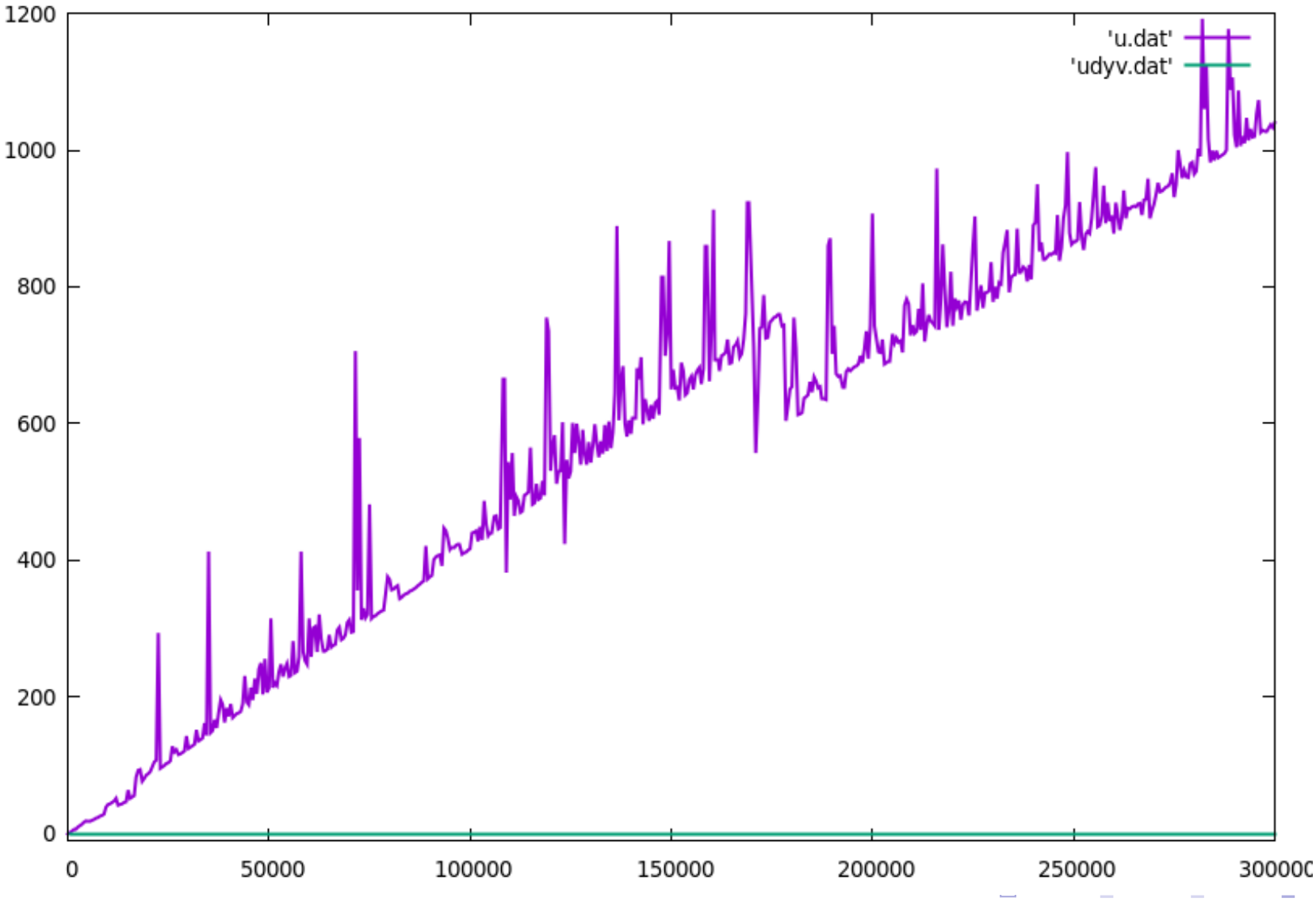


Figura: Unimodal sencilla y DyV de César

Traspuesta de una matriz. Analisis empirico



Traspuesta de una matriz. Analisis empirico

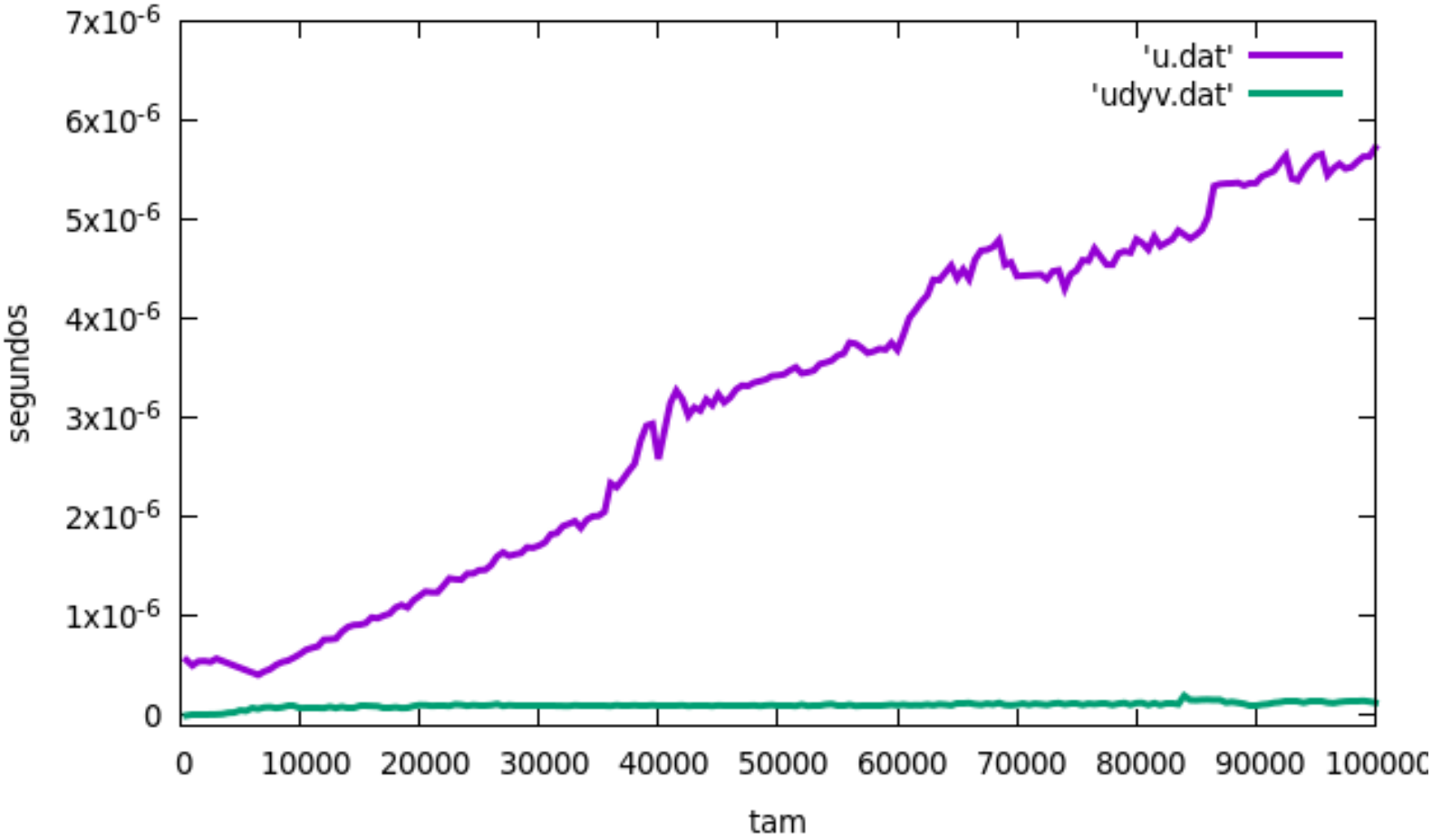


Figura: Unimodal sencilla y DyV de Irene

Serie unimodal. Analisis hibrido

La función con la que ajustamos el algoritmo sencillo es

$$f(x) = a0x + a1$$

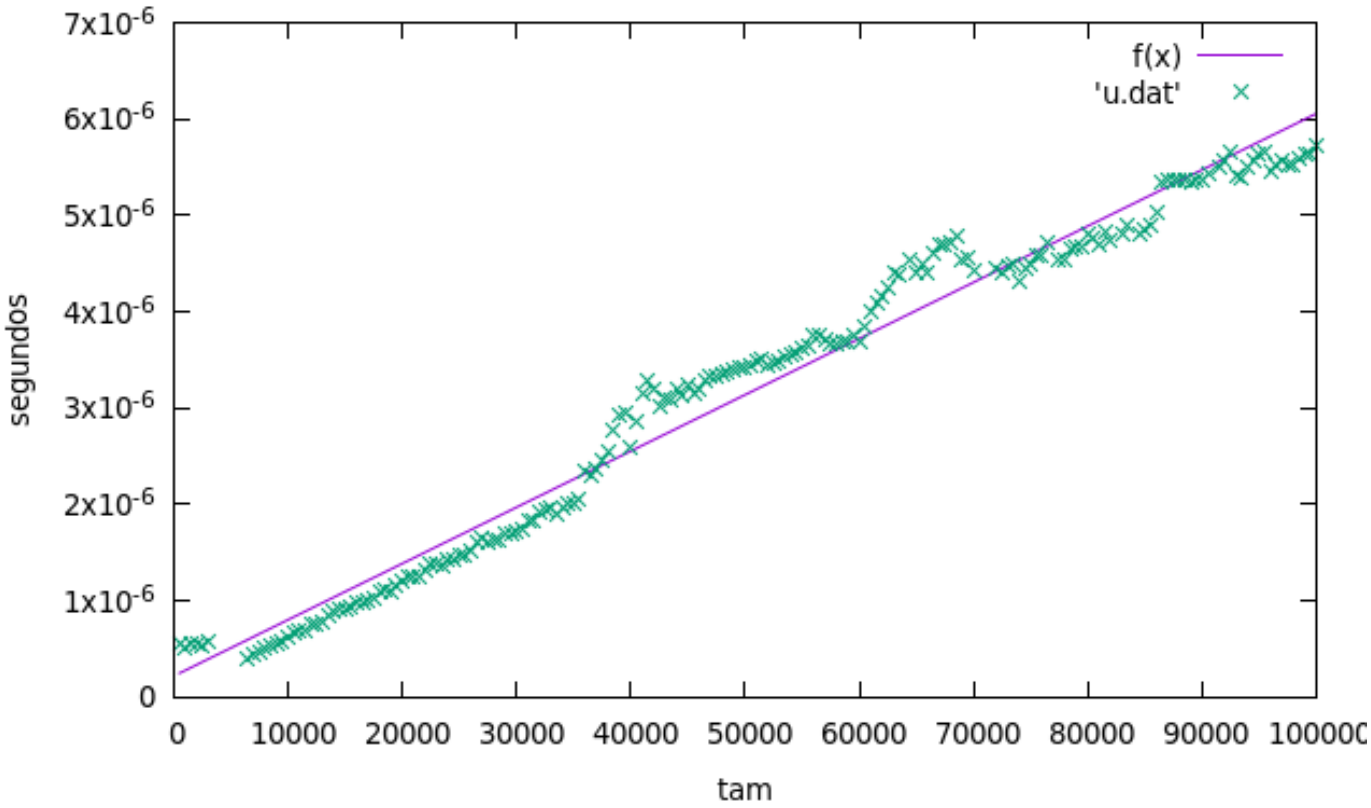


Figura: Hibrida unimodal sencilla de Irene

Serie unimodal. Analisis hibrido

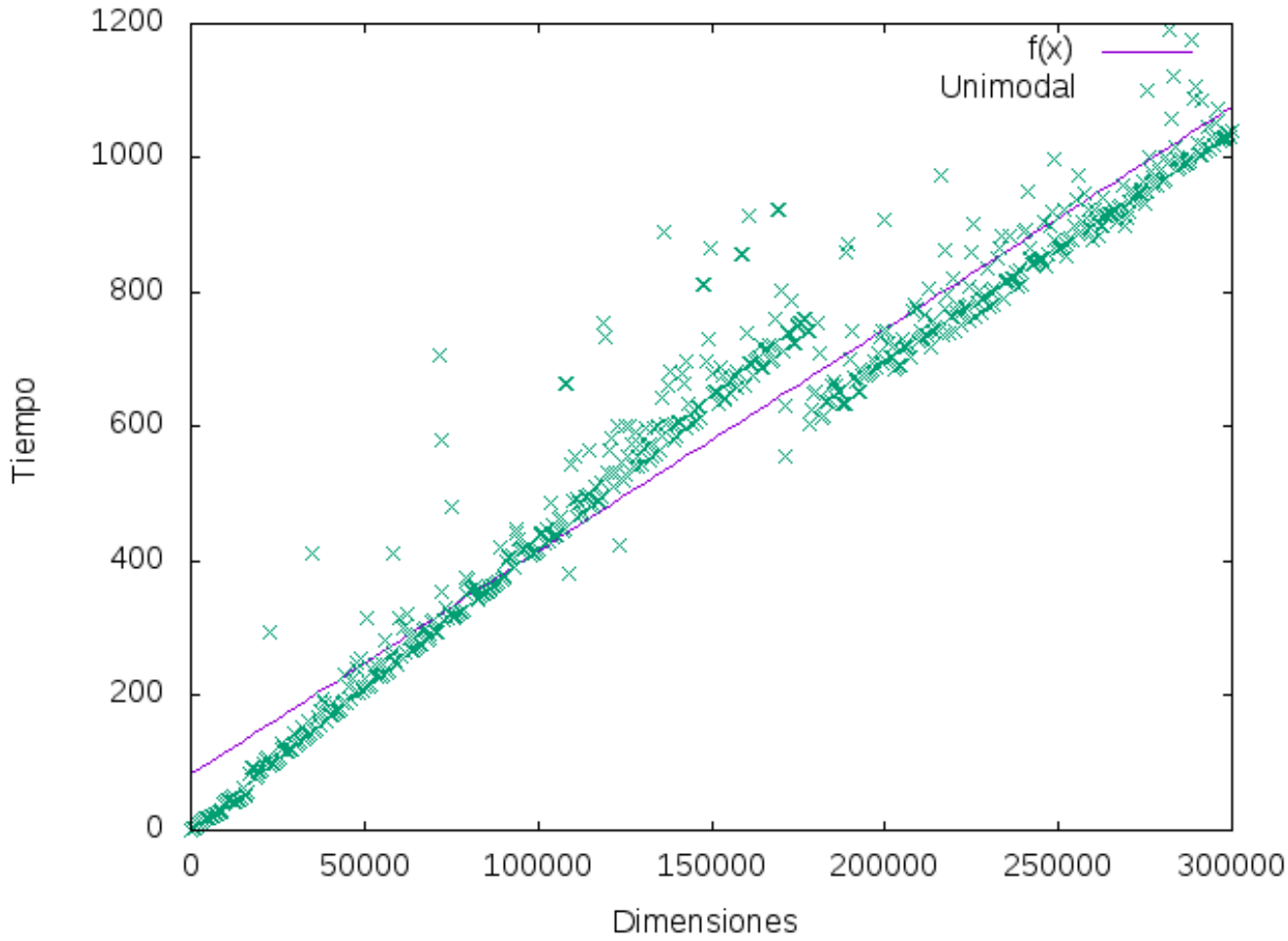


Figura: Hibrida unimodal sencilla de Javi

Serie unimodal. Analisis hibrido

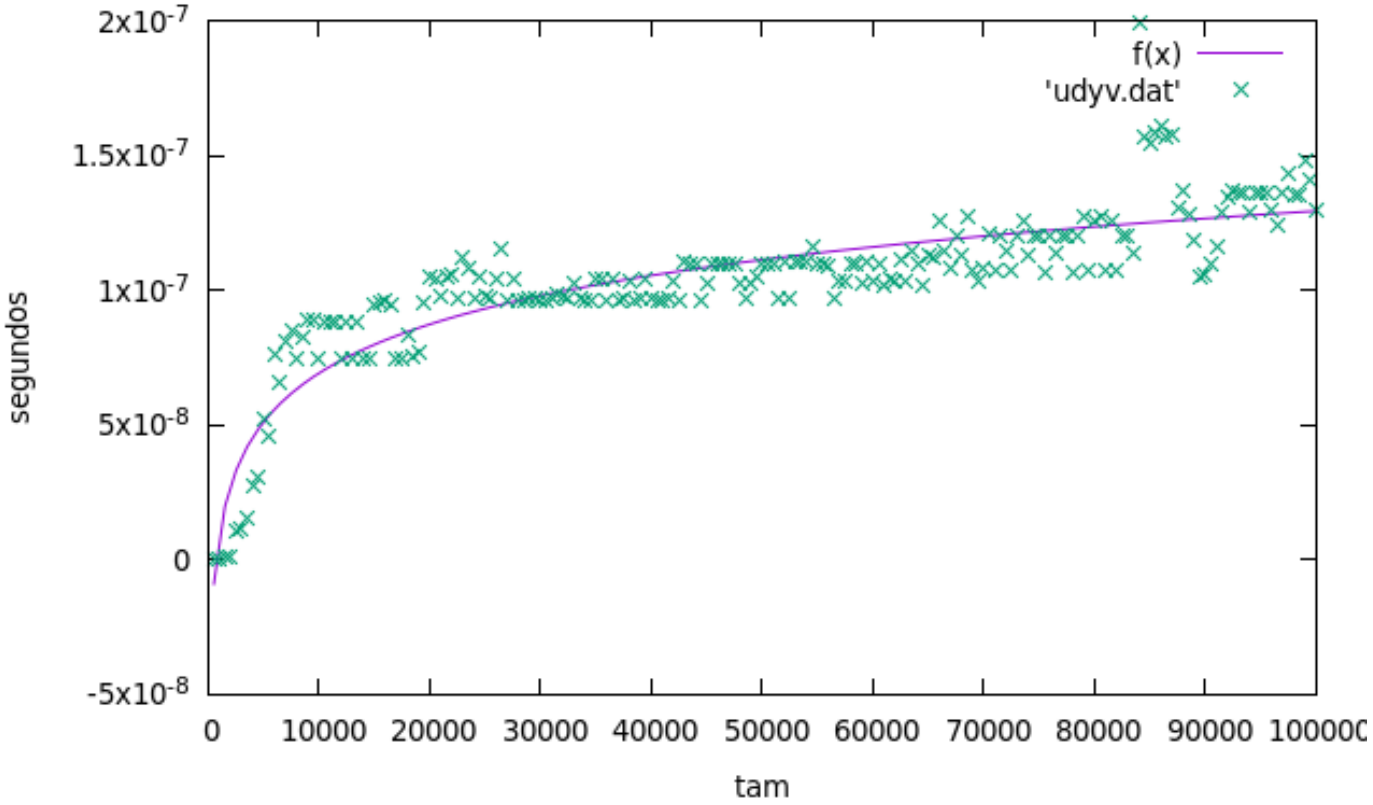


Figura: Hibrida Unimodal DyV de Irene

Serie unimodal. Analisis hibrido

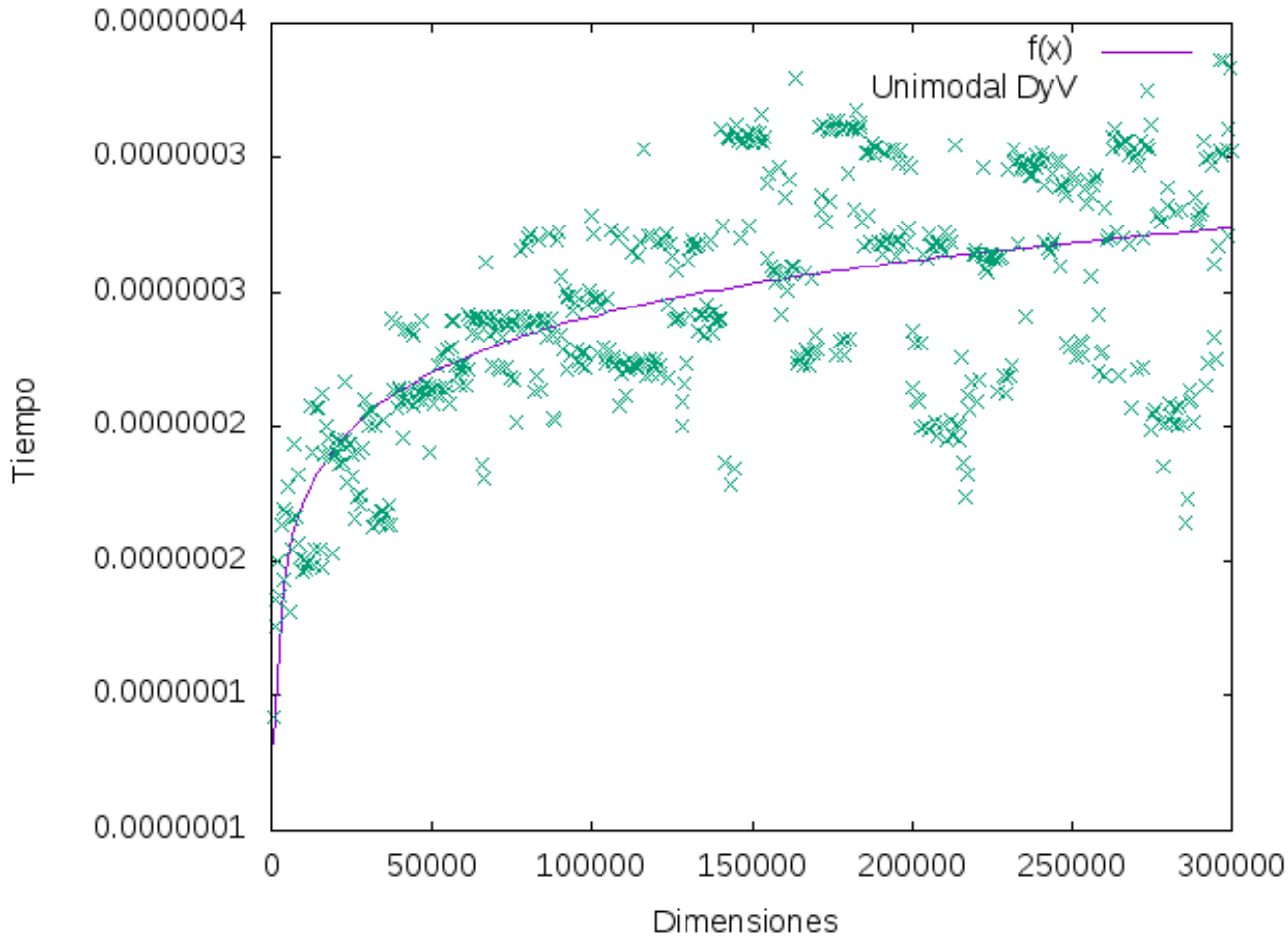


Figura: Hibrida unimodal DyV de Javi

Conclusión

Tras realizar esta práctica hemos llegado a las siguientes conclusiones:

1. En el algoritmo traspuesta si la matriz es de pocos elementos, no importa usar uno u otro código. Sin embargo si nos vamos a matrices con tamaños grandes, es mas eficiente el uso del algoritmo sencilli
2. En la serie unimodal sin embargo, hemos visto el umbral empirico y sea el vector del tamaño que sea siempre va a ser mas eficiente emplear el algoritmo con la tecnica divide y venceras
3. Por lo tanto el uso de la tecnica divide y venceras sera mejor o peor segun el algoritmo en el que se implemente.