Capítulo 5

Comparación entre métodos

En este capítulo se comparan entre sí los distintos métodos y algoritmos descritos y analizados en los capítulos 2, 3 y 4. Más en concreto, se compara el tiempo teórico de todos los algoritmos BSP en las distintas máquinas sobre las que se han obtenido resultados numéricos y se busca el algoritmo óptimo para cada una de las situaciones. Como se ha comprobado en los capítulos mencionados, el tiempo experimental se ajusta mejor al tiempo teórico esperado para tamaños de sistema grandes, por ello se realiza la comparación para tamaños de sistema entre 4096 y 65536 ecuaciones para el *cluster* de PC's y entre 16384 y 524288 ecuaciones para el resto de máquinas. Por comodidad, en la tabla 5.1 se vuelven a mostrar los valores de los parámetros BSP de todas las máquinas para las que se han obtenido resultados numéricos.

En las figuras 5.1–5.22 se muestran los tiempos teóricos, medidos en segundos, de todos los algoritmos BSP en las distintas máquinas. Cada una de las figuras contiene una gráfica, que para los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.2 (TW1) y 3.3 (TW2) representa el tiempo esperado tomando $m = 5$, y una tabla que muestra el algoritmo más rápido considerando diversos valores de m. Conviene recordar que m es el número de ecuaciones superpuestas y viene dado por la expresión 2.14 de la página 64.

s	p	l	g	$n_{\frac{1}{2}}$
45	1	423	2.3	26
	2	3294	9.5	25
	4	5366	12.4	25
	6	8164	12.5	25

(a) *IBM SP2 switch*

s	p	l	g	$n_{\frac{1}{2}}$
45	1	423	2.3	8
	2	20235	709.7	3
	4	54163	1362.6	9
	6	121958	3211.2	9

(b) *IBM SP2 ethernet*

s	p	l	g	$n_{\frac{1}{2}}$
16.4	1	23	0.2	22
	2	2556	6.9	5
	4	5152	7.4	4
	6	7538	6.8	4

(c) *Cluster de PC's*

s	p	l	g	$n_{\frac{1}{2}}$
12	1	68	0.3	94
	2	164	0.7	71
	4	168	0.7	66
	8	175	0.8	59
	16	181	0.9	61
	32	201	1.1	28
	64	148	1	27
	128	301	1.1	20
	256	387	1.2	15

(d) *CRAY T3D*

s	p	l	g	$n_{\frac{1}{2}}$
46.7	1	86	2.12	9
	2	269	0.87	33
	4	357	0.87	40
	8	506	0.81	40
	16	751	1.04	38
	32	1252	1.31	45

(e) *CRAY T3E*

Tabla 5.1: *Valores de parámetros BSP.*

5.1 IBM SP2

5.1.1 *Switch*

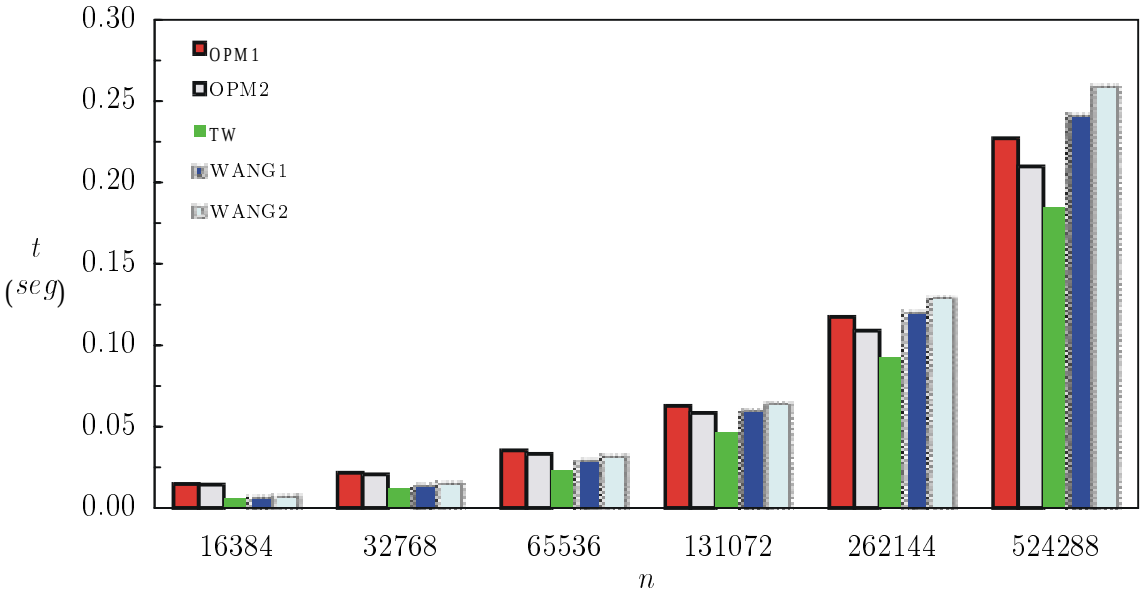
En las figuras 5.1, 5.2 y 5.3, se muestran los tiempos teóricos, medidos en segundos, en un IBM SP2 utilizando *switch*.

Para 2 procesadores el algoritmo 3.1 (TW) es el más rápido, con una diferencia sobre los demás algoritmos que va desde el 9.52% hasta el 41.13% para $n = 524288$ y $m = 5$. Los porcentajes representan la cantidad de tiempo adicional sobre el que necesita el algoritmo más rápido.

Para 4 procesadores el algoritmo más rápido es el 2.3 (OPM2), con muy poca diferencia sobre el algoritmo 3.3 (TW2) (el 0.04% para el tamaño máximo de sistema). Si $m = 100$, el algoritmo más rápido es ahora 3.3 (TW2), con poca diferencia sobre el algoritmo 2.3 (OPM2); si $m = 1000$ esta diferencia es mayor y el algoritmo 4.2 (WANG1) es más rápido para $n \in \{16384, 32768\}$.

Para 6 procesadores se tiene una situación parecida a la existente para 4 procesadores, la diferencia está en que el algoritmo 4.2 (WANG1) es óptimo en más ocasiones (para $m = 100$, $n = 16128$ y $m = 1000$, $n = 64512$).

En cualquiera de los casos, 2, 4, ó 6 procesadores, es más rápido resolver el sistema en secuencial mediante el método de eliminación de Gauss para sistemas tridiagonales. Para $n = 524288$ y $m = 5$, el incremento de tiempo del algoritmo paralelo más rápido sobre el tiempo necesario en secuencial varía desde el 97.91% para 2 procesadores hasta el $250,01\%$ para 6 procesadores. Es evidente por tanto que para el IBM SP2 utilizando *switch*, los algoritmos paralelos que se han analizado no son útiles si el objetivo es ahorrar tiempo en la resolución de los sistemas (ese objetivo se cumple mejor en máquinas con valores de g menores), ahora bien resultan de utilidad si el objetivo es resolver sistemas de tamaño superior al máximo admitido por un sólo procesador. Se pueden resolver, por ejemplo, sistemas de tamaño 3145728 $(6 \cdot 524288)$ sólo en paralelo.

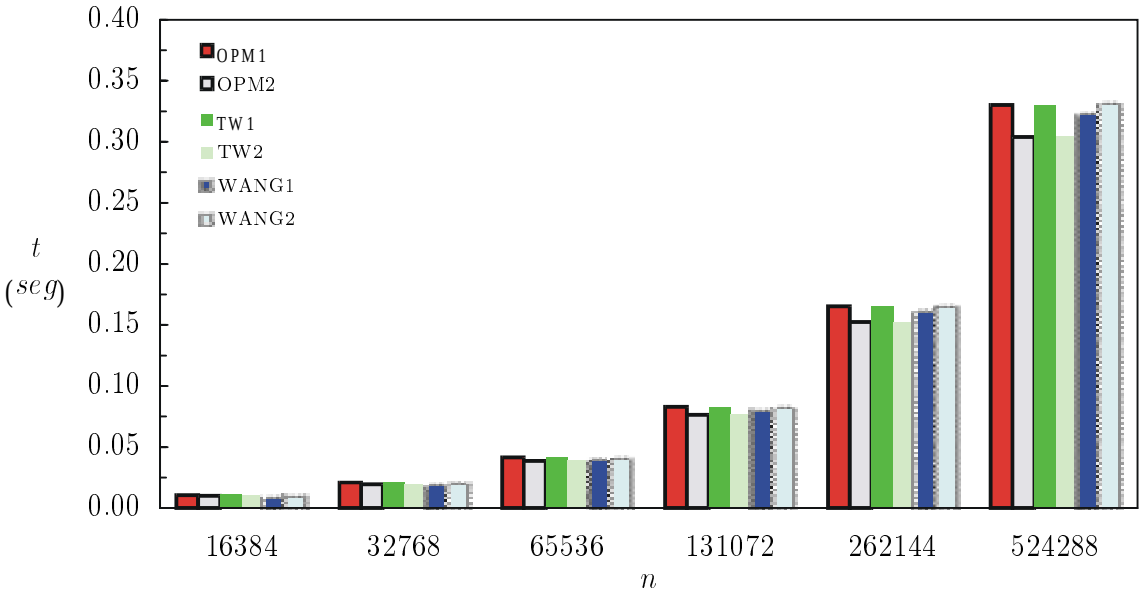


(a) $16384 \leq n \leq 524288$

IBM SP2 2 procesadores *switch*				
	m			
n	5	10	100	1000
16384	TW	TW	TW	TW
32768	TW	TW	TW	TW
65536	TW	TW	TW	TW
131072	TW	TW	TW	TW
262144	TW	TW	TW	TW
524288	TW	TW	TW	TW

(b) *Algoritmo óptimo*

Figura 5.1: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.1 (TW), 4.2 (WANG1) y 4.3 (WANG2) en un IBM SP2 con 2 procesadores interconectados mediante switch, para* $16384 \leq n \leq 524288$.

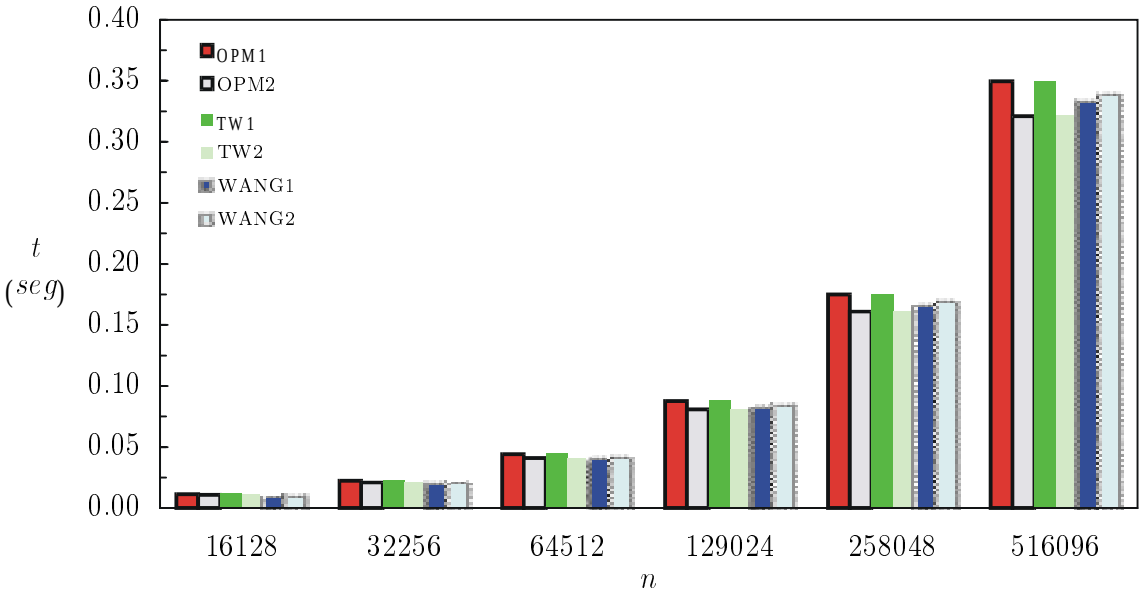


(a) $16384 \leq n \leq 524288$

IBM SP2 4 procesadores				
	m			
n	5	10	100	1000
16384	OPM2	OPM2	TW2	WANG1
32768	OPM2	OPM2	TW2	WANG1
65536	OPM2	OPM2	TW2	TW2
131072	OPM2	OPM2	TW2	TW2
262144	OPM2	OPM2	TW2	TW2
524288	OPM2	OPM2	TW2	TW2

(b) *Algoritmo óptimo*

Figura 5.2: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.2 (TW1), 3.3 (TW2), 4.2 (WANG1) y 4.3 (WANG2) en un IBM SP2 con 4 procesadores interconectados mediante switch, para* $16384 \leq n \leq 524288$.



(a) $16128 \leq n \leq 516096$

IBM SP2 6 procesadores *switch*				
	m			
n	5	10	100	1000
16128	OPM2	OPM2	WANG1	WANG1
32256	OPM2	OPM2	TW2	WANG1
64512	OPM2	OPM2	TW2	WANG1
129024	OPM2	OPM2	TW2	TW2
258048	OPM2	OPM2	TW2	TW2
516096	OPM2	OPM2	TW2	TW2

(b) *Algoritmo óptimo*

Figura 5.3: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.2 (TW1), 3.3 (TW2), 4.2 (WANG1) y 4.3 (WANG2) en un IBM SP2 con 6 procesadores interconectados mediante switch, para* $16128 \leq n \leq 516096$.

5.1.2 *Ethernet*

En las figuras 5.4, 5.5 y 5.6, se muestran los tiempos teóricos, medidos en segundos, en un IBM SP2 utilizando *ethernet*.

Para 2 procesadores, de nuevo, el algoritmo 3.1 (TW) es el más rápido, pero con poca diferencia sobre todos los demás. Para 4 procesadores el algoritmo más rápido para $m \geq 100$ es (casi siempre) el 4.2 (WANG1) y para 6 procesadores lo es el algoritmo 4.2 (WANG1) para $m \geq 10$; véanse las figuras 5.5(b) y 5.6(b). En todos los casos no existe gran diferencia en los tiempos.

Utilizando *ethernet* los tiempos que se obtienen son mucho mayores que utilizando *switch*, por lo que resulta siempre mejor utilizar este último dispositivo de comunicación entre los procesadores.

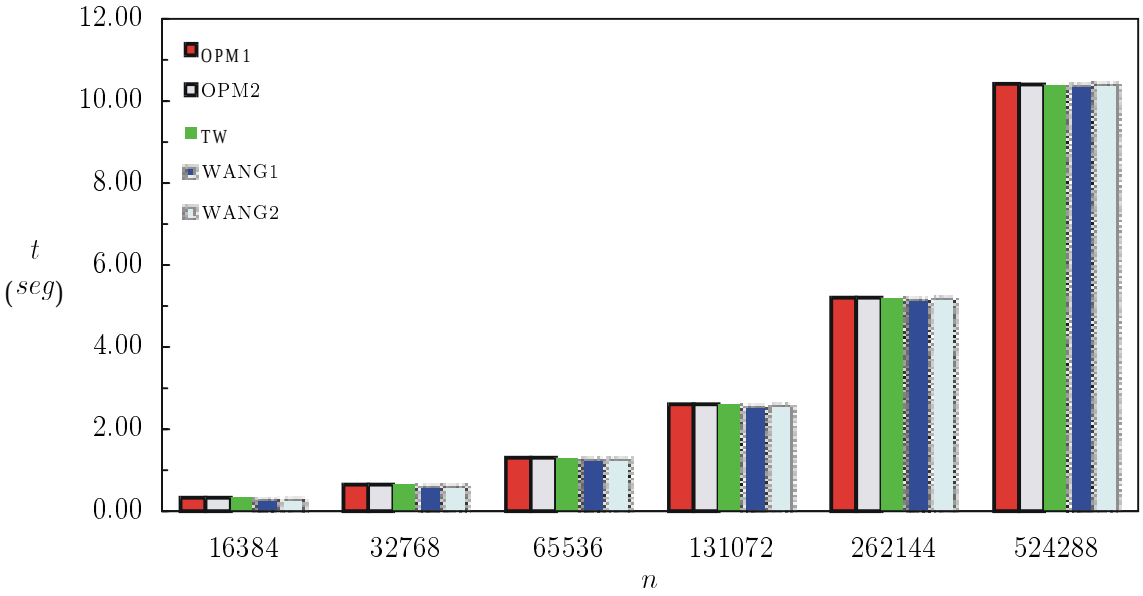
5.2 *Cluster* de PC's

En las figuras 5.7, 5.8 y 5.9, se muestran los tiempos teóricos, medidos en segundos, en un *cluster* de PC's.

Para 2 procesadores el algoritmo más rápido es, otra vez, el 3.1 (TW) (excepto para $n = 4096$ y $m \in \{5, 10, 100\}$), la diferencia sobre los demás algoritmos, para $n = 65536$ y $m = 5$, varía desde el 13.36% hasta el 52.50%.

Para 4 procesadores el algoritmo más rápido, en la mayoría de situaciones, es el 2.2 (OPM1). Para el máximo tamaño de sistema es más rápido el algoritmo 2.3 (OPM2), la diferencia sobre los demás algoritmos varía desde el 2.99% hasta el 16.65%. Para $m = 1000$ y $n \in \{4096, 8192, 16384, 32768\}$ es mejor el algoritmo 3.2 (TW1).

En cualquiera de los casos, 2, 4 ó 6 procesadores, es más rápido resolver el sistema en secuencial mediante el método de eliminación de Gauss para sistemas tridiagonales. Para el máximo tamaño de sistema y $m = 5$, el incremento de tiempo del algoritmo paralelo más rápido sobre el tiempo necesario en secuencial varía desde el 69.49% para 2 procesadores hasta el 205.67% para 6 procesadores. En el *cluster* de PC's ocurre lo mismo que en el IBM SP2, los algoritmos paralelos que se han analizado son más útiles

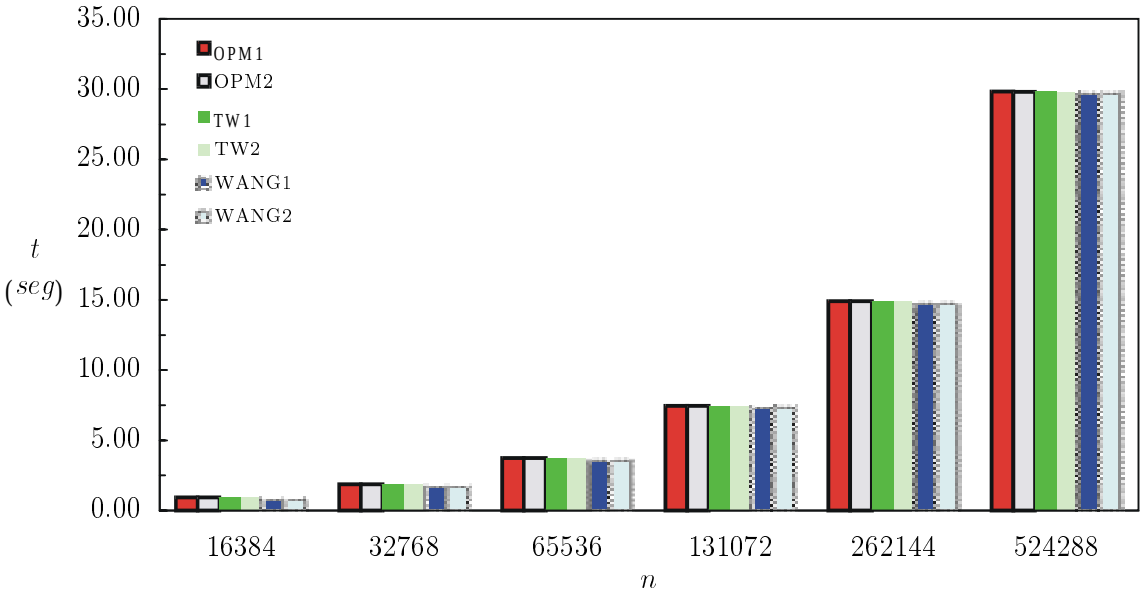


(a) $16384 \leq n \leq 524288$

IBM SP2 2 procesadores *ethernet*				
	m			
n	5	10	100	1000
16384	TW	TW	TW	TW
32768	TW	TW	TW	TW
65536	TW	TW	TW	TW
131072	TW	TW	TW	TW
262144	TW	TW	TW	TW
524288	TW	TW	TW	TW

(b) *Algoritmo óptimo*

Figura 5.4: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.1 (TW), 4.2 (WANG1) y 4.3 (WANG2) en un IBM SP2 con 2 procesadores interconectados mediante ethernet, para* $16384 \leq n \leq 524288$.

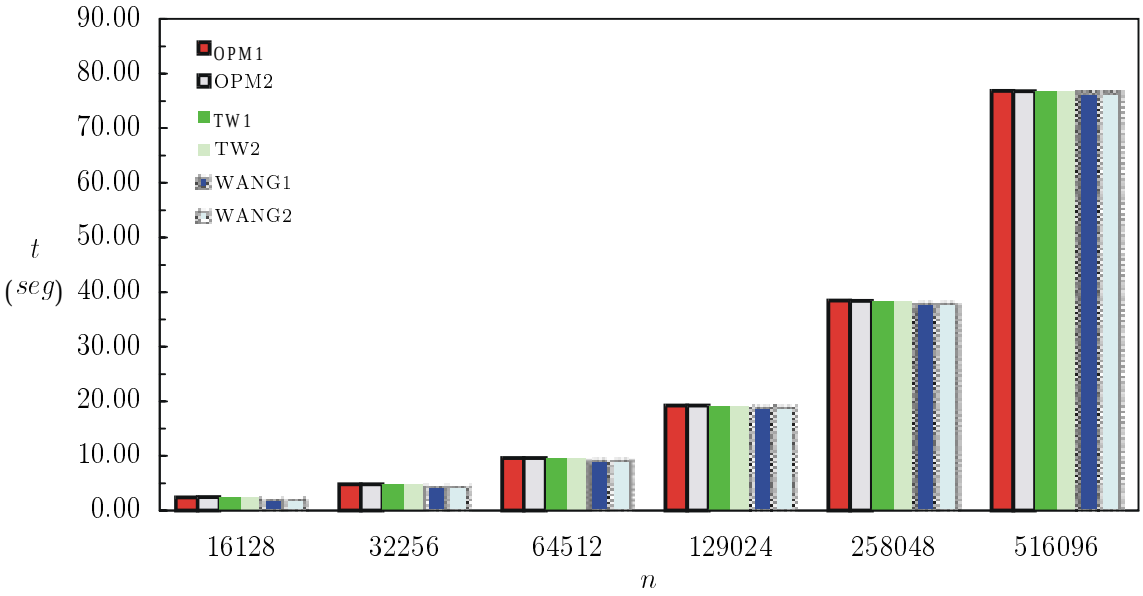


(a) $16384 \leq n \leq 524288$

IBM SP2 4 procesadores *ethernet*				
	m			
n	5	10	100	1000
16384	OPM1	TW1	WANG1	WANG1
32768	OPM1	TW1	WANG1	WANG1
65536	OPM2	WANG1	WANG1	WANG1
131072	OPM2	TW2	WANG1	WANG1
262144	OPM2	TW2	WANG1	WANG1
524288	OPM2	TW2	TW2	WANG1

(b) *Algoritmo óptimo*

Figura 5.5: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.2 (TW1), 3.3 (TW2), 4.2 (WANG1) y 4.3 (WANG2) en un IBM SP2 con 4 procesadores interconectados mediante ethernet, para* $16384 \leq n \leq 524288$.

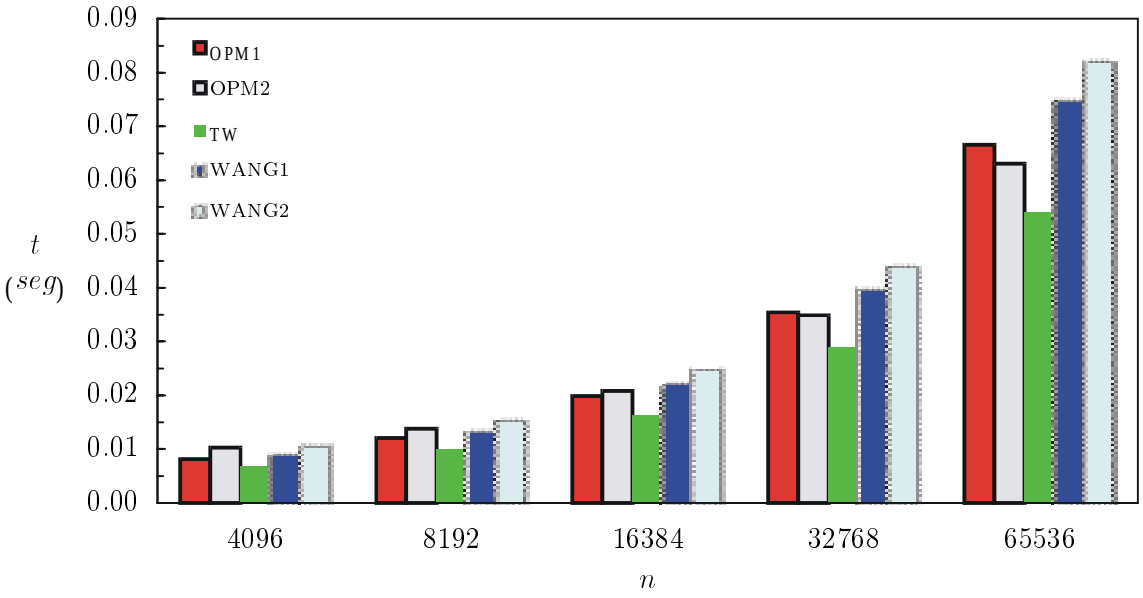


(a) $16128 \leq n \leq 516096$

IBM SP2 6 procesadores *ethernet*				
	m			
n	5	10	100	1000
16128	TW1	WANG1	WANG1	WANG1
32256	TW1	WANG1	WANG1	WANG1
64512	WANG1	WANG1	WANG1	WANG1
129024	WANG1	WANG1	WANG1	WANG1
258048	OPM2	WANG1	WANG1	WANG1
516096	OPM2	TW2	WANG1	WANG1

(b) *Algoritmo óptimo*

Figura 5.6: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.2 (TW1), 3.3 (TW2), 4.2 (WANG1) y 4.3 (WANG2) en un IBM SP2 con* 6 *procesadores interconectados mediante ethernet, para* $16128 \leq n \leq 516096$.

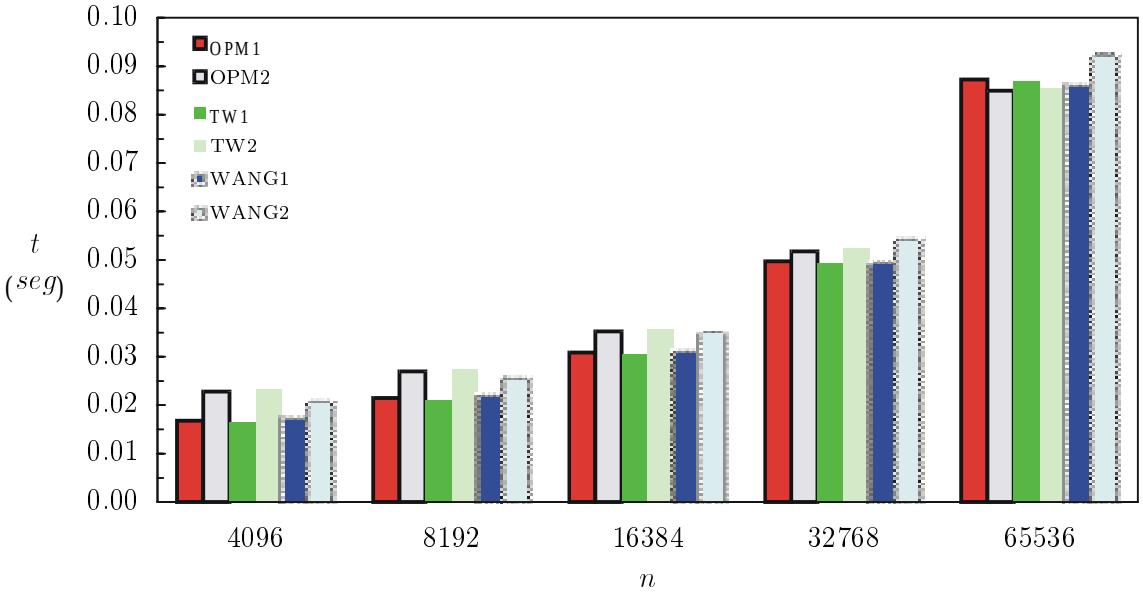


(a) $4096 \leq n \leq 65536$

Cluster de PC's 2 procesadores				
	m			
n	5	10	100	1000
4096	OPM1	OPM1	OPM1	TW
8192	TW	TW	TW	TW
16384	TW	TW	TW	TW
32768	TW	TW	TW	TW
65536	TW	TW	TW	TW

(b) *Algoritmo óptimo*

Figura 5.7: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.1 (TW), 4.2 (WANG1) y 4.3 (WANG2) en un cluster de PC's, para 2 procesadores y* $4096 \leq n \leq 65536$.

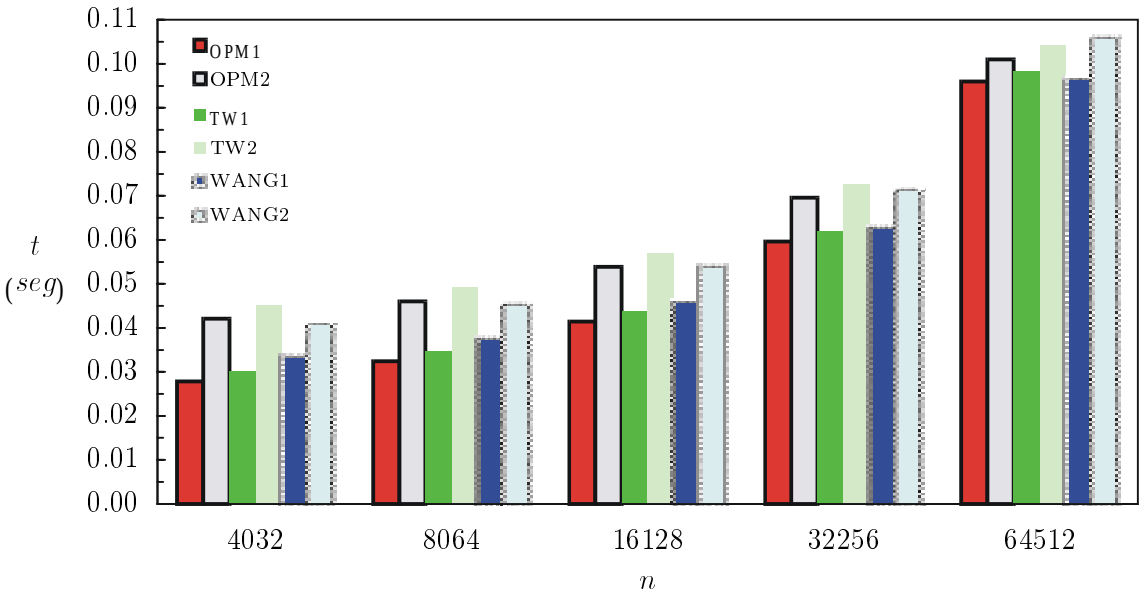


(a) $4096 \leq n \leq 65536$

Cluster de PC's 4 procesadores				
	m			
n	5	10	100	1000
4096	OPM1	OPM1	OPM1	TW1
8192	OPM1	OPM1	OPM1	TW1
16384	OPM1	OPM1	OPM1	TW1
32768	OPM1	OPM1	OPM1	TW1
65536	OPM2	OPM2	OPM2	OPM2

(b) *Algoritmo óptimo*

Figura 5.8: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.2 (TW1), 3.3 (TW2), 4.2 (WANG1) y 4.3 (WANG2) en un cluster de PC's, para 4 procesadores y* $4096 \leq n \leq 65536$.



(a) $4032 \leq n \leq 64512$

Cluster de PC's 6 procesadores				
	m			
n	5	10	100	1000
4032	OPM1	OPM1	OPM1	OPM1
8064	OPM1	OPM1	OPM1	OPM1
16128	OPM1	OPM1	OPM1	OPM1
32256	OPM1	OPM1	OPM1	OPM1
64512	OPM1	OPM1	OPM1	OPM1

(b) *Algoritmo óptimo*

Figura 5.9: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.2 (TW1), 3.3 (TW2), 4.2 (WANG1) y 4.3 (WANG2) en un cluster de PC's, para 6 procesadores y* $4032 \leq n \leq 64512$.

si el objetivo es resolver sistemas de tamaño superior al máximo admitido por un sólo procesador, en esta máquina se pueden resolver, por ejemplo, sistemas de tamaño 387072 $(6 \cdot 64512)$ sólo en paralelo.

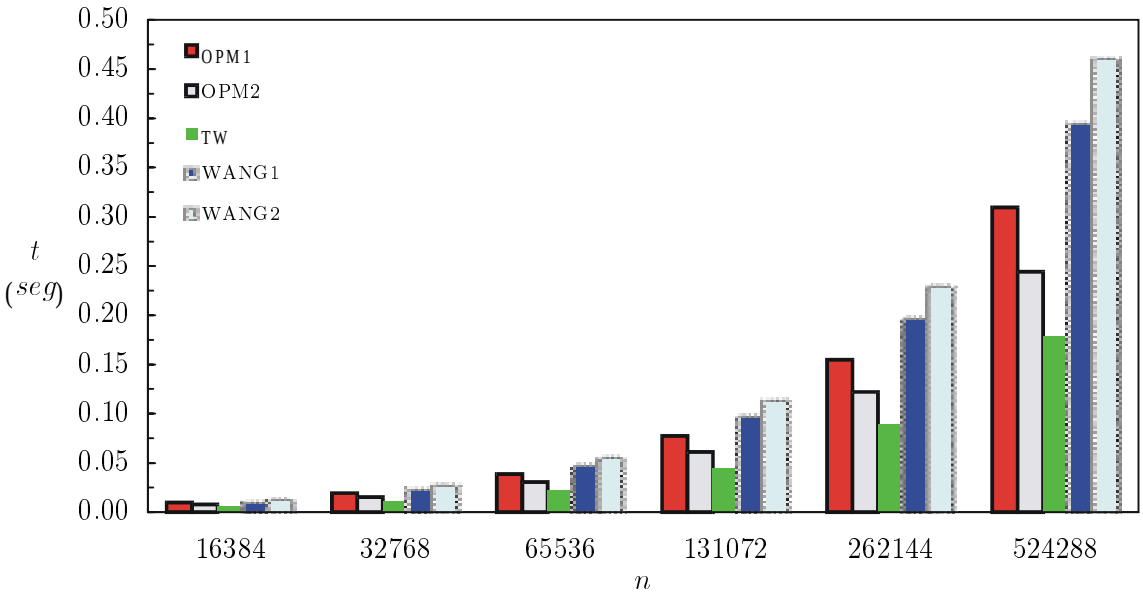
5.3 CRAY T3D

En las figuras 5.10–5.17 se muestran los tiempos teóricos, medidos en segundos, en un CRAY T3D. De nuevo, el algoritmo 3.1 (TW) es el más rápido para 2 procesadores, la diferencia sobre los demás algoritmos varía desde el 36.7% hasta el 159% para $n = 524288$ y $m = 5$. Se recuerda que los porcentajes representan la cantidad de tiempo adicional sobre el que necesita el algoritmo más rápido.

Para 4 procesadores el algoritmo más rápido es el 2.3 (OPM2), con muy poca diferencia sobre el algoritmo 3.3 (TW2). Si el valor de m es 100, es más rápido el algoritmo 3.3 (TW2). No hay una diferencia sustancial en el tiempo invertido por los algoritmos 2.3 (OPM2) y 3.3 (TW2) para valores de m no muy grandes, ahora bien, cuando el valor de m es grande la diferencia es significativa; por ejemplo, para $m = 50000$, $p = 4$ y $n = 524288$ el algoritmo 3.3 (TW2) es el más rápido de todos y tarda 0.2248 segundos mientras que el algoritmo 2.3 (OPM2) tarda 0.2734 segundos, lo que supone un $21,64\%$ de incremento. Para esos mismos valores de m, p y n la resolución del sistema secuencialmente mediante el método de Gauss para sistemas tridiagonales tarda 0.3495 segundos, lo que supone un $55,49\%$ más que con el algoritmo 3.3 (TW2) que es el óptimo.

A medida que va aumentando el número de procesadores, se observa que el algoritmo 3.3 (TW2) es el óptimo para valores de m cada vez más pequeños y que para valores de m grandes el algoritmo 4.2 (WANG1) es el más rápido, a excepción de $p = 255$ donde el óptimo es el algoritmo 4.3 (WANG2) en muchas situaciones.

En esta máquina (véase la tabla 5.1(d)) con valor de g próximo a 1, es más rápido resolver el sistema mediante un algoritmo en paralelo que en secuencial mediante el método de eliminación de Gauss para sistemas tridiagonales. Para 2 procesadores, $n = 524288$ y $m = 5$, el incremento de tiempo necesario en secuencial sobre el algoritmo paralelo más rápido (TW) es del 95.67%, el *speed-up* es muy bueno, $S_2 = 1.96$, al igual que la eficiencia, $E_2 = 97.84\%$. Precisamente este buen comportamiento es el que hace que el

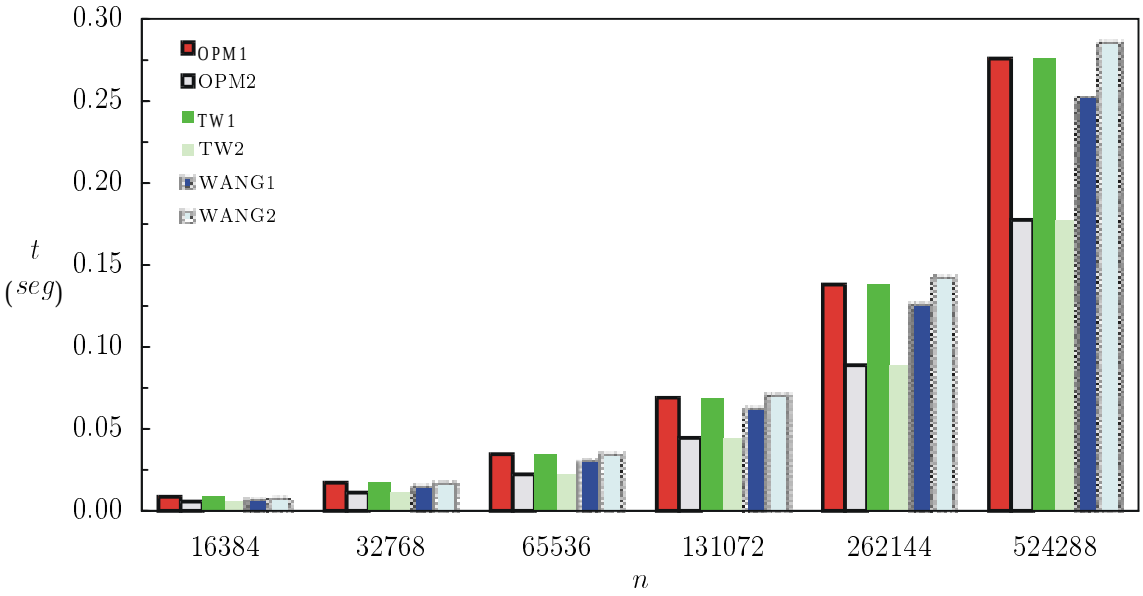


(a) $16384 \leq n \leq 524288$

CRAY T3D 2 procesadores				
	m			
n	5	10	100	1000
16384	TW	TW	TW	TW
32768	TW	TW	TW	TW
65536	TW	TW	TW	TW
131072	TW	TW	TW	TW
262144	TW	TW	TW	TW
524288	TW	TW	TW	TW

(b) *Algoritmo óptimo*

Figura 5.10: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.1 (TW), 4.2 (WANG1) y 4.3 (WANG2) en un CRAY T3D, para 2 procesadores y* $16384 \leq n \leq 524288$.

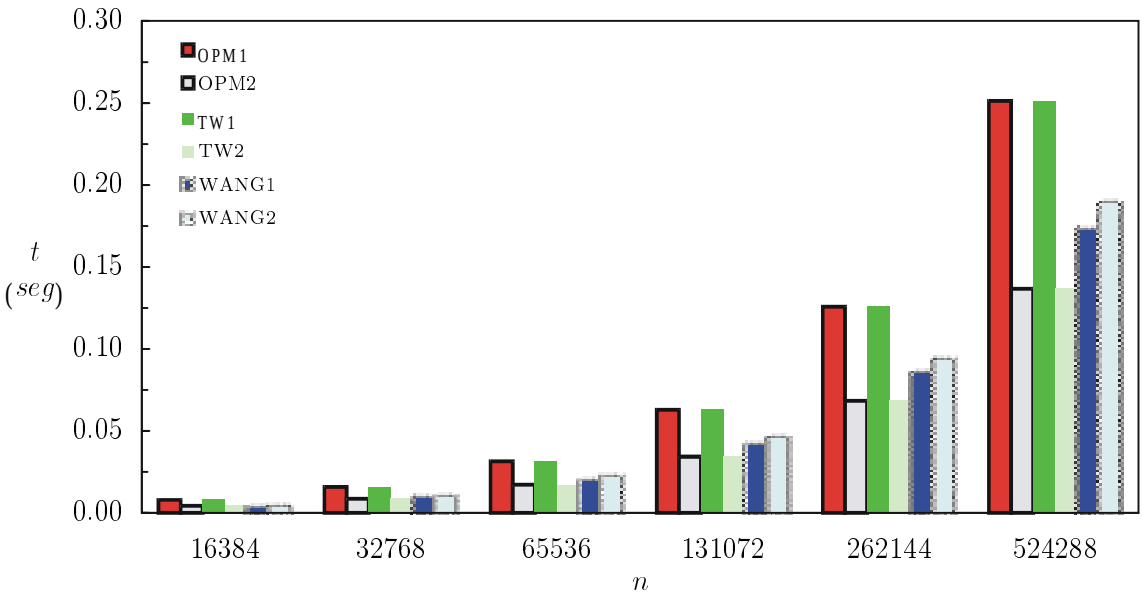


(a) $16384 \leq n \leq 524288$

CRAY T3D 4 procesadores				
	m			
n	5	10	100	1000
16384	OPM2	OPM2	TW2	TW2
32768	OPM2	OPM2	TW2	TW2
65536	OPM2	OPM2	TW2	TW2
131072	OPM2	OPM2	TW2	TW2
262144	OPM2	OPM2	TW2	TW2
524288	OPM2	OPM2	TW2	TW2

(b) *Algoritmo óptimo*

Figura 5.11: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.2 (TW1), 3.3 (TW2), 4.2 (WANG1) y 4.3 (WANG2) en un CRAY T3D, para 4 procesadores y* $16384 \leq n \leq 524288$.

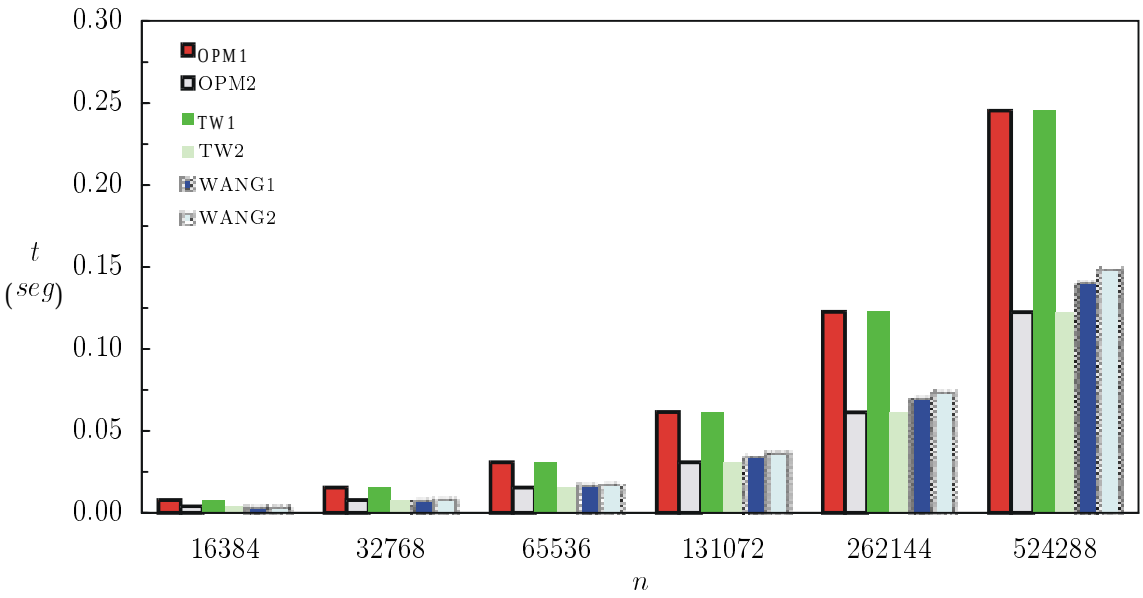


(a) $16384 \leq n \leq 524288$

CRAY T3D 8 procesadores				
	m			
n	5	10	100	1000
16384	OPM2	OPM2	TW2	WANG1
32768	OPM2	OPM2	TW2	TW2
65536	OPM2	OPM2	TW2	TW2
131072	OPM2	OPM2	TW2	TW2
262144	OPM2	OPM2	TW2	TW2
524288	OPM2	OPM2	TW2	TW2

(b) *Algoritmo óptimo*

Figura 5.12: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.2 (TW1), 3.3 (TW2), 4.2 (WANG1) y 4.3 (WANG2) en un CRAY T3D, para 8 procesadores y* $16384 \leq n \leq 524288$.

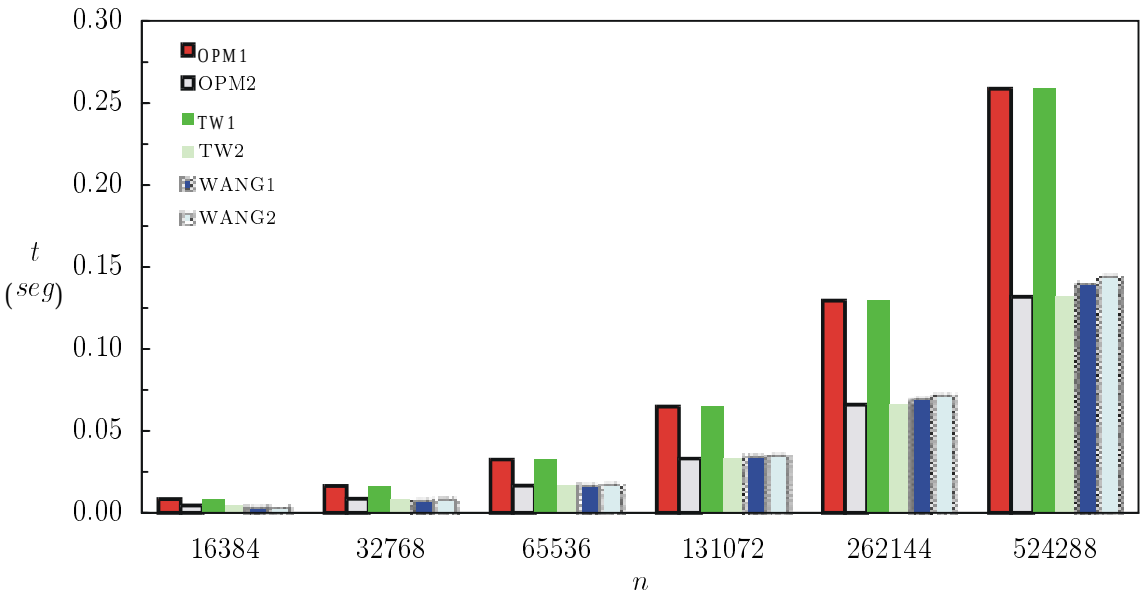


(a) $16384 \leq n \leq 524288$

CRAY T3D 16 procesadores				
	m			
n	5	10	100	1000
16384	OPM2	TW2	TW2	WANG1
32768	OPM2	TW2	TW2	WANG1
65536	OPM2	TW2	TW2	WANG1
131072	OPM2	TW2	TW2	TW2
262144	OPM2	TW2	TW2	TW2
524288	OPM2	TW2	TW2	TW2

(b) *Algoritmo óptimo*

Figura 5.13: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.2 (TW1), 3.3 (TW2), 4.2 (WANG1) y 4.3 (WANG2) en un CRAY T3D, para* 16 *procesadores y* $16384 \leq n \leq 524288$.

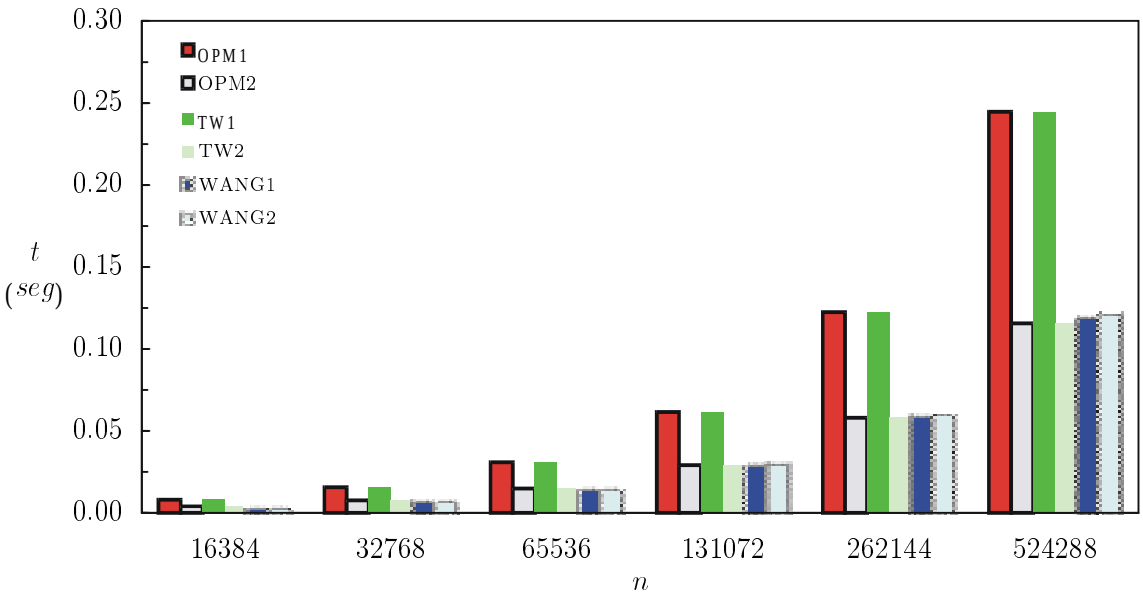


(a) $16384 \leq n \leq 524288$

CRAY T3D 32 procesadores				
	m			
n	5	10	100	1000
16384	TW2	TW2	WANG1	WANG1
32768	TW2	TW2	WANG1	WANG1
65536	TW2	TW2	TW2	WANG1
131072	TW2	TW2	TW2	WANG1
262144	TW2	TW2	TW2	WANG1
524288	TW2	TW2	TW2	TW2

(b) *Algoritmo óptimo*

Figura 5.14: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.2 (TW1), 3.3 (TW2), 4.2 (WANG1) y 4.3 (WANG2) en un CRAY T3D, para* 32 *procesadores y* $16384 \leq n \leq 524288$.

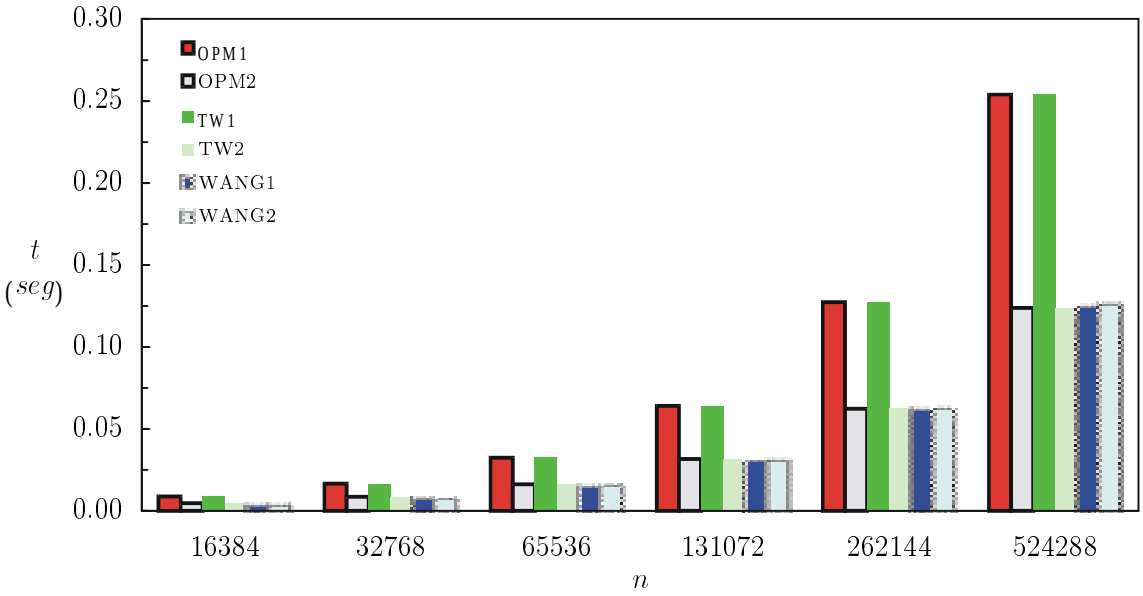


(a) $16384 \leq n \leq 524288$

CRAY T3D 64 procesadores				
	m			
n	5	10	100	1000
16384	TW2	TW2	WANG1	WANG1
32768	TW2	TW2	WANG1	WANG1
65536	TW2	TW2	WANG1	WANG1
131072	TW2	TW2	TW2	WANG1
262144	TW2	TW2	TW2	WANG1
524288	TW2	TW2	TW2	WANG1

(b) *Algoritmo óptimo*

Figura 5.15: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.2 (TW1), 3.3 (TW2), 4.2 (WANG1) y 4.3 (WANG2) en un CRAY T3D, para 64 procesadores y* $16384 \leq n \leq 524288$.

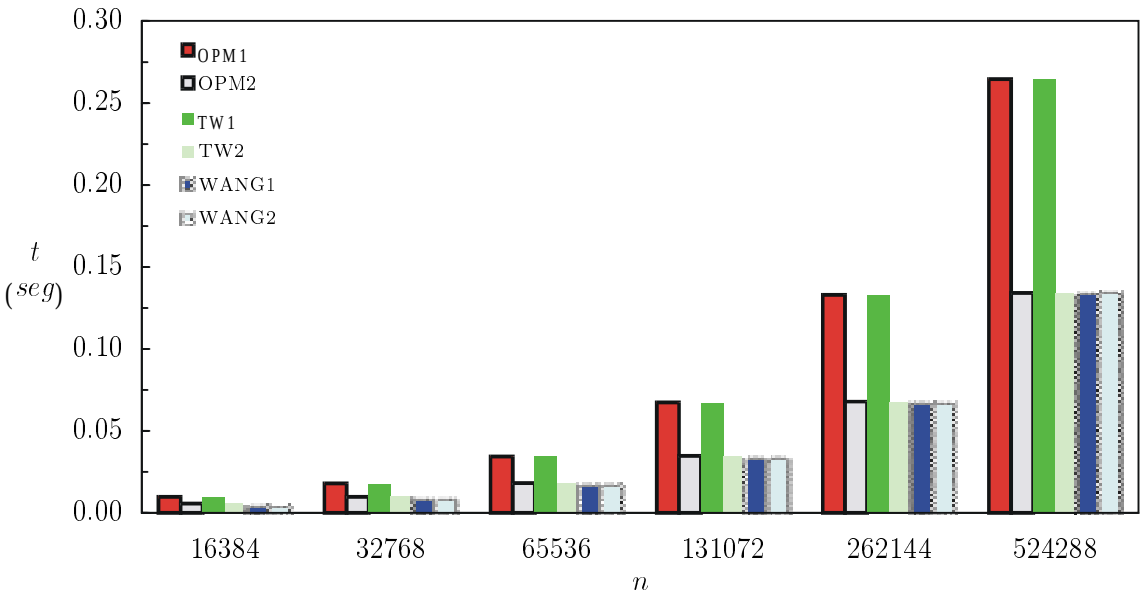


(a) $16384 \leq n \leq 524288$

CRAY T3D 128 procesadores				
	m			
n	5	10	100	1000
16384	WANG2	WANG2	WANG2	WANG2
32768	TW2	WANG1	WANG1	WANG1
65536	TW2	TW2	WANG1	WANG1
131072	TW2	TW2	WANG1	WANG1
262144	TW2	TW2	WANG1	WANG1
524288	TW2	TW2	TW2	WANG1

(b) *Algoritmo óptimo*

Figura 5.16: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.2 (TW1), 3.3 (TW2), 4.2 (WANG1) y 4.3 (WANG2) en un CRAY T3D, para* 128 *procesadores y* $16384 \leq n \leq 524288$.



(a) $16384 \leq n \leq 524288$

CRAY T3D 256 procesadores				
	m			
n	5	10	100	1000
16384	WANG2	WANG2	WANG2	WANG2
32768	WANG2	WANG2	WANG2	WANG2
65536	TW2	WANG2	WANG2	WANG2
131072	TW2	WANG1	WANG1	WANG1
262144	TW2	TW2	WANG1	WANG1
524288	TW2	TW2	WANG1	WANG1

(b) *Algoritmo óptimo*

Figura 5.17: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.2 (TW1), 3.3 (TW2), 4.2 (WANG1) y 4.3 (WANG2) en un CRAY T3D, para* 128 *procesadores y* $16384 \leq n \leq 524288$.

algoritmo 3.3 (TW2) sea el más rápido en muchas situaciones.

En la tabla 5.2 se muestra, para diversos valores de m, el número óptimo de procesadores, si el objetivo es conseguir la mayor rapidez posible en la resolución del sistema tridiagonal. En las columnas etiquetadas con Proc se muestra el número de procesadores óptimo, en las columnas etiquetadas con Alg se muestra el algoritmo con el que se consigue el mejor tiempo y en las columnas etiquetadas con Gauss se muestra el incremento porcentual de tiempo que se produce al resolver el sistema en secuencial por el método de eliminación de Gauss para sistemas tridiagonales sobre el tiempo utilizado en paralelo por el algoritmo más rápido; un 100% de incremento significa que en secuencial se tarda el doble que en paralelo.

5.4 CRAY T3E

En las figuras 5.18–5.22 se muestran los tiempos teóricos, medidos en segundos, en un CRAY T3E.

En esta máquina, el algoritmo 3.1 (TW) también es el más rápido para 2 procesadores, la diferencia sobre los demás algoritmos varía desde el 29.49% hasta el 127.74%, para $n = 524288$ y $m = 5$.

Para 4, 8, 16 y 32 procesadores el algoritmo más rápido es el 2.3 (OPM2), con muy poca diferencia sobre el algoritmo 3.3 (TW2); por ejemplo, para $n = 524288$ y $m = 5$ la máxima diferencia, que se alcanza para 32 procesadores, es del 0.05%. En cambio, si $m = 100$ el algoritmo 3.3 (TW2) es el más rápido, con muy poca diferencia sobre el algoritmo 2.3 (OPM2). La diferencia es sustancial para valores de m mayores; por ejemplo, para $p = 8$, $n = 524288$ y $m = 10000$ el algoritmo 3.3 (TW2) sigue siendo el más rápido, la diferencia con el algoritmo 2.3 (OPM2) es de un $10,18\%$ y con los demás varía desde el $14,97\%$ hasta el $85,07\%$.

Al igual que en el CRAY T3D, en esta máquina el valor de g está próximo a 1 (véase la tabla 5.1(e)) y es más rápido resolver el sistema mediante un algoritmo en paralelo que en secuencial mediante el método de eliminación de Gauss para sistemas tridiagonales. Para 2 procesadores, $n = 524288$ y $m = 5$, el incremento de tiempo necesario en secuencial sobre el algoritmo paralelo más rápido (TW) es del 57.2%, el *speed-up* también es bueno,

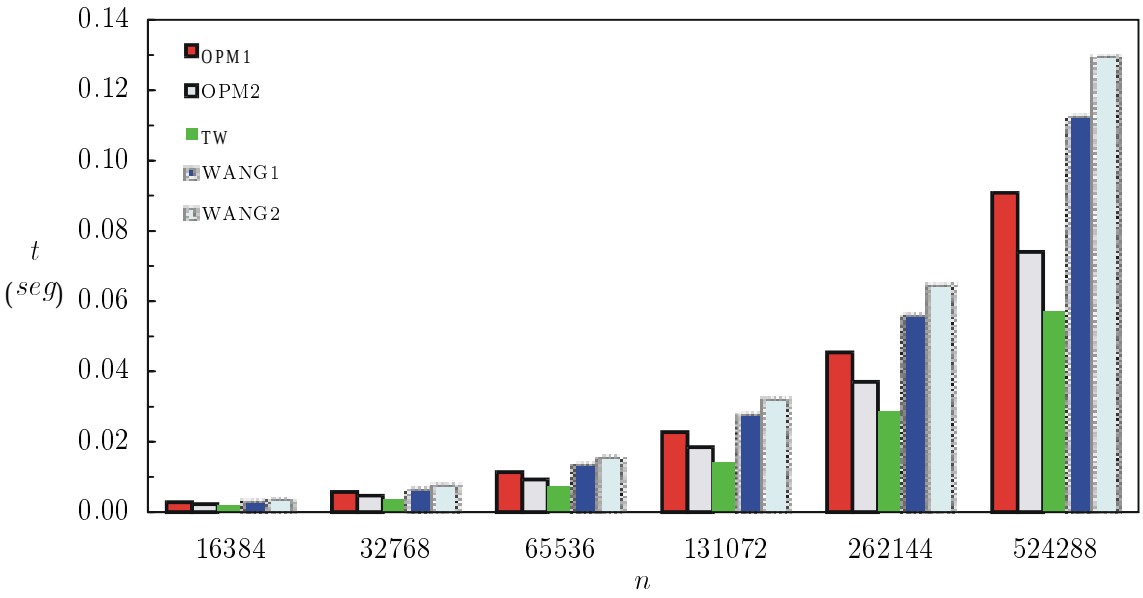
CRAY T3D						
	m					
	5			10		
n	Proc.	Alg.	Gauss	Proc.	Alg.	Gauss
16384	$16p$	OPM2	167.61%	$16p$	TW2	166.45%
32768	$64p$	TW2	184.66%	$64p$	TW2	182.52%
65536	$64p$	TW2	193.98%	$64p$	TW2	192.86%
131072	$64p$	TW2	198.84%	$64p$	TW2	198.27%
262144	$64p$	TW2	201.33%	$64p$	TW2	201.04%
524288	$64p$	TW2	202.59%	$64p$	TW2	202.44%

(a) $m \in \{5, 10\}$

CRAY T3D						
	m					
	100			1000		
n	Proc.	Alg.	Gauss	Proc.	Alg.	Gauss
16384	$64p$	WANG1	159.83%	$64p$	WANG1	159.83%
32768	$64p$	WANG1	174.89%	$64p$	WANG1	174.89%
65536	$64p$	WANG1	183.07%	$64p$	WANG1	183.07%
131072	$64p$	TW2	188.35%	$64p$	WANG1	187.34%
262144	$64p$	TW2	195.94%	$64p$	WANG1	189.52%
524288	$64p$	TW2	199.85%	$64p$	WANG1	190.63%

(b) $m \in \{100, 1000\}$

Tabla 5.2: *Número óptimo de procesadores en un CRAY T3D, para* $16384 \leq n \leq 524288$.

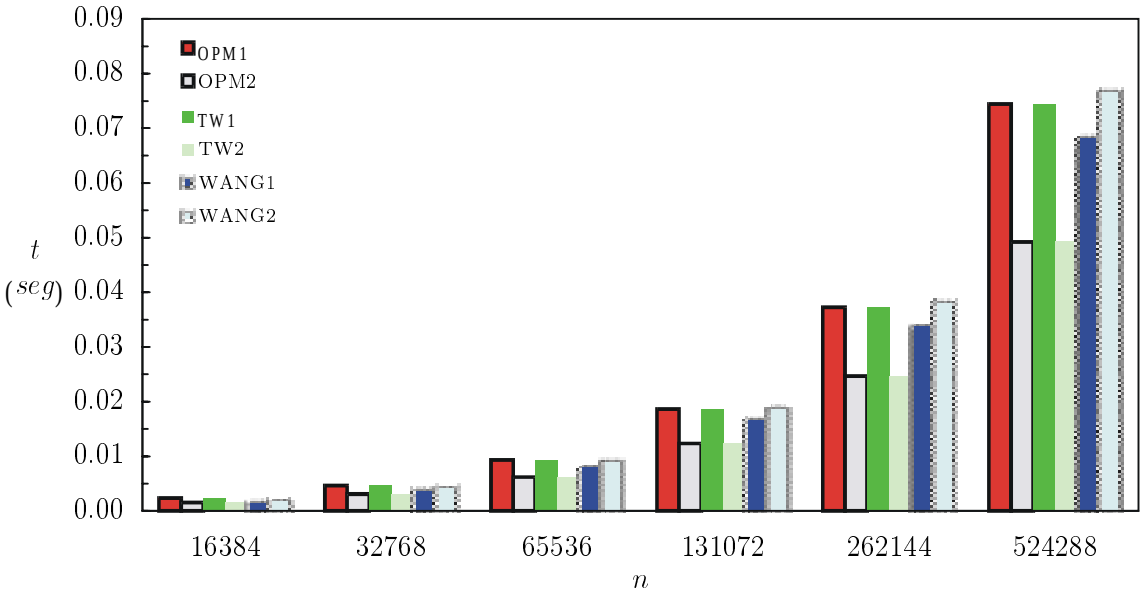


(a) $16384 \leq n \leq 524288$

CRAY T3E 2 procesadores				
	m			
n	5	10	100	1000
16384	TW	TW	TW	TW
32768	TW	TW	TW	TW
65536	TW	TW	TW	TW
131072	TW	TW	TW	TW
262144	TW	TW	TW	TW
524288	TW	TW	TW	TW

(b) *Algoritmo óptimo*

Figura 5.18: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.1 (TW), 4.2 (WANG1) y 4.3 (WANG2) en un CRAY T3E, para 2 procesadores y* $16384 \leq n \leq 524288$.

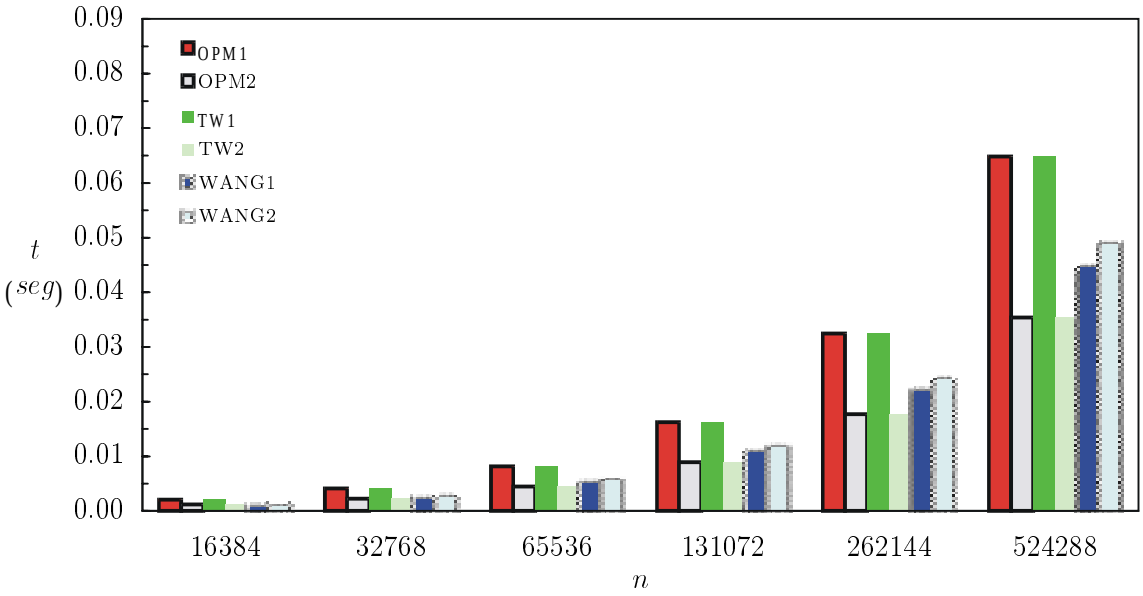


(a) $16384 \leq n \leq 524288$

CRAY T3E 4 procesadores				
	m			
n	5	10	100	1000
16384	OPM2	OPM2	TW2	TW2
32768	OPM2	OPM2	TW2	TW2
65536	OPM2	OPM2	TW2	TW2
131072	OPM2	OPM2	TW2	TW2
262144	OPM2	OPM2	TW2	TW2
524288	OPM2	OPM2	TW2	TW2

(b) *Algoritmo óptimo*

Figura 5.19: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.2 (TW1), 3.3 (TW2), 4.2 (WANG1) y 4.3 (WANG2) en un CRAY T3E, para* 4 *procesadores y* $16384 \leq n \leq 524288$.

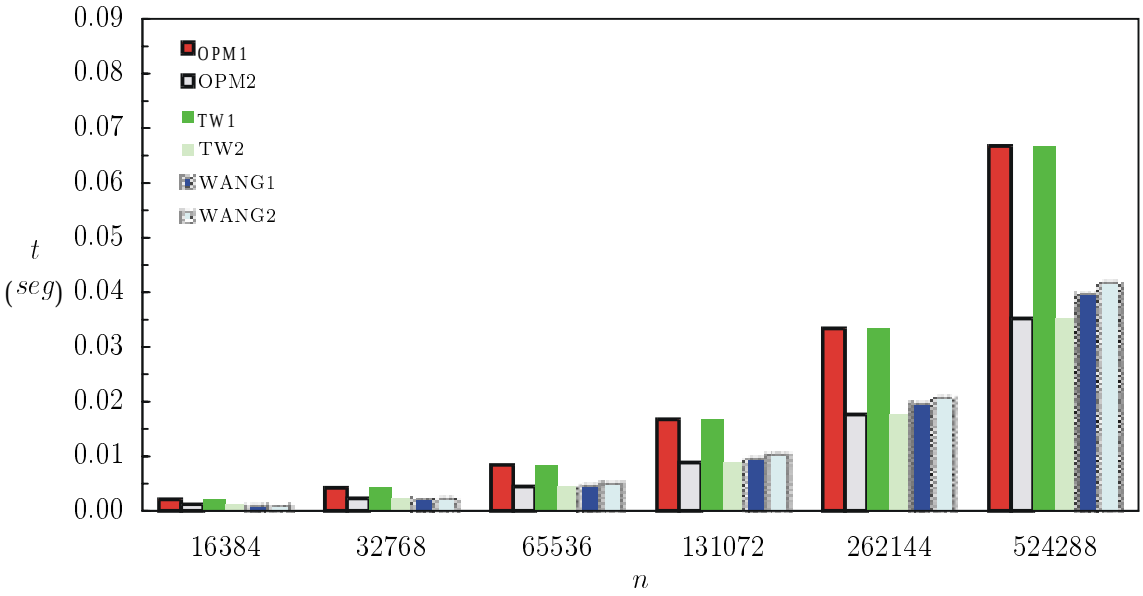


(a) $16384 \leq n \leq 524288$

CRAY T3E 8 procesadores				
	m			
n	5	10	100	1000
16384	OPM2	OPM2	TW2	WANG1
32768	OPM2	OPM2	TW2	TW2
65536	OPM2	OPM2	TW2	TW2
131072	OPM2	OPM2	TW2	TW2
262144	OPM2	OPM2	TW2	TW2
524288	OPM2	OPM2	TW2	TW2

(b) *Algoritmo óptimo*

Figura 5.20: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.2 (TW1), 3.3 (TW2), 4.2 (WANG1) y 4.3 (WANG2) en un CRAY T3E, para* 8 *procesadores y* $16384 \leq n \leq 524288$.

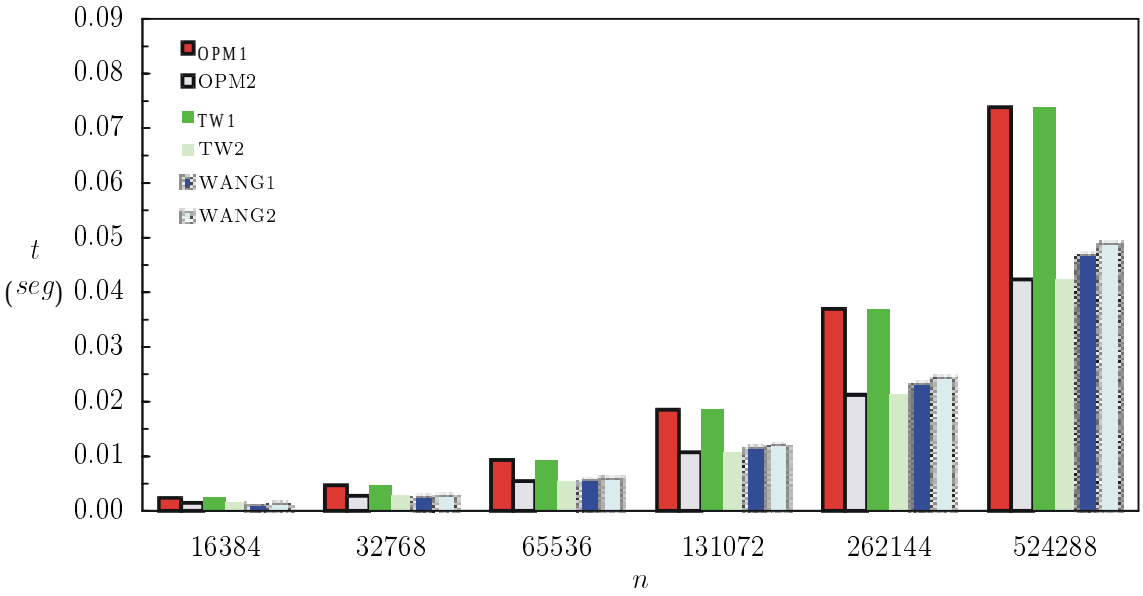


(a) $16384 \leq n \leq 524288$

CRAY T3E 16 procesadores				
	m			
n	5	10	100	1000
16384	OPM2	OPM2	TW2	WANG1
32768	OPM2	OPM2	TW2	WANG1
65536	OPM2	OPM2	TW2	WANG1
131072	OPM2	OPM2	TW2	TW2
262144	OPM2	OPM2	TW2	TW2
524288	OPM2	OPM2	TW2	TW2

(b) *Algoritmo óptimo*

Figura 5.21: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.2 (TW1), 3.3 (TW2), 4.2 (WANG1) y 4.3 (WANG2) en un CRAY T3E, para* 16 *procesadores y* $16384 \leq n \leq 524288$.



(a) $16384 \leq n \leq 524288$

CRAY T3E 32 procesadores				
	m			
n	5	10	100	1000
16384	OPM2	OPM2	TW2	WANG1
32768	OPM2	OPM2	TW2	WANG1
65536	OPM2	OPM2	TW2	WANG1
131072	OPM2	OPM2	TW2	TW2
262144	OPM2	OPM2	TW2	TW2
524288	OPM2	OPM2	TW2	TW2

(b) *Algoritmo óptimo*

Figura 5.22: *Comparación entre los algoritmos 2.2 (OPM1), 2.3 (OPM2), 3.2 (TW1), 3.3 (TW2), 4.2 (WANG1) y 4.3 (WANG2) en un CRAY T3E, para* 32 *procesadores y* $16384 \leq n \leq 524288$.

$S_2 = 1.57$, al igual que la eficiencia, $E_2 = 78.6\%$. Este buen comportamiento es el que hace que el algoritmo 3.3 (TW2) sea el más rápido en muchas situaciones.

En la tabla 5.3 se muestra, para diversos valores de m, el número óptimo de procesadores, si el objetivo es conseguir la mayor rapidez posible en la resolución del sistema tridiagonal. Al igual que en la tabla 5.2, en las columnas etiquetadas con Proc se muestra el número de procesadores óptimo, en las columnas etiquetadas con Alg se muestra el algoritmo con el que se consigue el mejor tiempo y en las columnas etiquetadas con Gauss se muestra el incremento porcentual de tiempo que se produce al resolver el sistema en secuencial por el método de eliminación de Gauss para sistemas tridiagonales sobre el tiempo utilizado en paralelo por el algoritmo más rápido; un 100% de incremento significa que en secuencial se tarda el doble que en paralelo.

CRAY T3E						
	m					
	5			10		
n	Proc.	Alg.	Gauss	Proc.	Alg.	Gauss
16384	$8p$	OPM2	141.17%	$8p$	OPM2	140.34%
32768	$8p$	OPM2	147.52%	$8p$	OPM2	147.09%
65536	$8p$	OPM2	150.83%	$8p$	OPM2	150.61%
131072	$16p$	OPM2	152.98%	$16p$	OPM2	152.75%
262144	$16p$	OPM2	154.48%	$16p$	OPM2	154.36%
524288	$16p$	OPM2	155.24%	$16p$	OPM2	155.18%

(a) $m \in \{5, 10\}$

CRAY T3E						
	m					
	100			1000		
n	Proc.	Alg.	Gauss	Proc.	Alg.	Gauss
16384	$8p$	TW2	131.80%	$16p$	WANG1	108.02%
32768	$8p$	TW2	142.52%	$16p$	WANG1	115.97%
65536	$8p$	TW2	148.24%	$8p$	TW2	130.21%
131072	$8p$	TW2	151.19%	$8p$	TW2	141.63%
262144	$16p$	TW2	153.19%	$8p$	TW2	147.77%
524288	$16p$	TW2	154.59%	$8p$	TW2	150.96%

(b) $m \in \{100, 1000\}$

Tabla 5.3: *Número óptimo de procesadores en un CRAY T3E, para* $16384 \leq n \leq 524288$.