Programación Dinámica Paralela, Aplicación para el Despacho Óptimo de Unidades Generadoras en una Central Hidroeléctrica

Eustaquio A. Martínez[1], Sebastián Arce[1,2]

[1]Facultad Politécnica - Universidad Nacional del Este, [2]Itaipú Binacional

Paraguay

RESUMEN

La técnica de Programación Dinámica (PD), basado en el principio de optimalidad de Bellman, ha sido ampliamente utilizada en varios campos. Aplicaciones de algoritmos secuenciales para la solución de los más diversos problemas han sido estudiados, sin embargo en todos los casos la dimensión del problema ha tenido un rol muy importante, pues el tamaño del problema tratado exige gran capacidad de procesamiento. Esto ha propiciado el estudio de técnicas de paralelización de la Programación Dinámica, para los más diversos ambientes computacionales paralelos.

En la ingeniería eléctrica, el despacho óptimo de unidades generadoras en centrales hidroeléctricas ha motivado varios estudios, aplicando también PD. El despacho de máquinas en una central hidroeléctrica, así como en centrales de térmicas, requiere una programación adecuada de manera a minimizar los costos como consecuencia de la demanda, debido a los costos asociados a las pérdidas de arranques y paradas de las unidades generadoras. Este problema generalmente se resuelve discretizando los datos del sistema y aplicando PD, sin embargo la programación de la operación puede requerir varios horizontes: largo, medio y corto. Es obvio que cuanto más largo es el plazo, y la cantidad de unidades generadoras de la central, mayor es la dimensión del problema. La idea de la paralelización del método de PD, en este trabajo, se basa en el modelo Maestro – Esclavo, implementado en un clúster de estaciones de trabajo conectadas en red (NOWS). Análisis preliminares sugieren que la codificación menos complicada es la que paraleliza el método mencionado en el espacio de estados en cada etapa de la programación, es decir, en cada etapa del proceso se asignan porciones del espacio de estados a procesadores esclavos, de manera que simultáneamente busquen el costo mínimo para llegar a ese estado, considerando la programación "para adelante" o *forward* de la PD.

Como problema ejemplo se ha utilizado el despacho máquinas para los horizontes de programación de un día, una semana un mes y seis meses, de la usina hidroeléctrica de Itaipú, utilizando 20 unidades generadoras de 700MW. Como medida de desempeño de la propuesta, en el contexto paralelo, se utilizó el *Speedup* (S_p). Los resultados obtenidos en este estudio de caso, muestran una tendencia creciente en el speedup y una tendencia decreciente de los tiempos, principalmente en los casos con horizontes más extensos. Estos resultados alientan el estudio más profundo de la paralelización aplicada en la PD, como método de solución de problemas relacionados al planeamiento de la operación de sistemas hidrotérmicos, que en la medida que aumentan el número de componentes, el horizonte de planeamiento y el grado de discretización, tanto de las variables de estado y etapas, puede resultar imposible su tratamiento con PD convencional.

PALABRAS CLAVES

Programación dinámical paralela, Despacho óptimo de unidade generadoras, Cluster de computadoras personales

1. INTRODUCCIÓN

La técnica de Programación Dinámica (PD), basado en el principio de optimalidad de Bellman [3], ha sido ampliamente utilizada en varios campos, tales como Teoría de Control, Investigación de Operaciones, Ciencia de la Computación, Biología, etc. [4]. También se ha verificado su amplia aplicación en la Ingeniería, en particular en la Ingeniería Eléctrica [1, 5], en algunos casos combinados con otras técnicas [5]. Aplicaciones de algoritmos secuenciales para la solución de los más diversos problemas han sido estudiadas, sin embargo en todos los casos la dimensión del problema ha tenido un rol muy importante, dado que un problema de tamaño considerable, exige gran capacidad de procesamiento, lo que ha propiciado el estudio de técnicas de paralelización de la Programación Dinámica, para los más diversos ambientes computacionales paralelos [4, 9, 13, 14].

El despacho óptimo de unidades generadoras en centrales hidroeléctricas ha motivado varios estudios, aplicando también Programación Dinámica. En particular, el estudio del despacho óptimo de máquinas (unidades generadoras) ha despertado interés, dado su relevancia en el contexto económico [1, 5, 6]. El despacho de máquinas en una central hidroeléctrica, así como en centrales térmicas, requiere una programación adecuada de manera a minimizar los costos como consecuencia de la demanda, dado que existen costos importantes asociados a las pérdidas, arranques y paradas de las unidades generadoras [1]. El problema de despacho, en general se resuelve discretizando los datos del sistema y aplicando Programación Dinámica, sin embargo la programación de la operación puede requerir varios horizontes: Largo, medio, corto o diario, que pueden corresponder a varios años, un año, una semana, un día, etc. [1, 2]. Es obvio que cuanto más largo es el plazo, y la cantidad de unidades generadoras de la central, mayor es la dimensión del problema, hecho que motiva el estudio de la paralelización del método de PD para el despacho óptimo.

Este artículo está organizado de la siguiente manera: en la sección 2, se presentan los fundamentos de la Programación Dinámica, en la sección 3 se expone el problema de despacho óptimo de unidades generadoras de una central hidroeléctrica, incluyendo su formulación matemática y el algoritmo secuencial utilizado en su solución. En la sección 4 se plantea el enfoque paralelo para la Programación Dinámica en el contexto del despacho óptimo de unidades generadoras, dejando para la sección 5 los resultados experimentales de la aplicación de la propuesta en un problema ejemplo y finalmente en la sección 6, se exponen las conclusiones del trabajo.

2. PROGRAMACIÓN DINÁMICA

2.1. Fundamentos

La Programación Dinámica fue creada para el tratamiento matemático de situaciones en que el problema se reduce a procesos de decisión multietapas. Físicamente podría considerarse como un sistema que en cierto instante, su estado es determinado por un conjunto de valores denominados variables de estado [3]. En ciertos momentos predeterminados o determinados por el mismo proceso, es necesario que se tome una decisión que afectará el estado del sistema. La referida decisión equivale a una transformación de las variables de estado. El resultado del proceso de decisión es para guiar las decisiones futuras con el propósito de maximizar o minimizar alguna función de los parámetros, describiendo el estado final. La decisión es denominada política y esa política presentará mayores ventajas en función a algún criterio preasignado, denominado política óptima. En resumen la idea básica de la Programación Dinámica es, en función a la política óptima, determinar la decisión requerida a cada estado del sistema, basada en el principio

de optimalidad [3], que establece que una óptima política tiene la propiedad de que cualquiera sean el estado y la decisión iniciales, las decisiones remanentes deben constituir una óptima política, respecto al estado resultante de la primera decisión.

2.2. Formulación Matemática

La formulación aquí presentada, sigue los planteamientos esgrimidos en [4]. Un problema de optimización puede ser planteado como: encontrar la secuencia de decisiones $(u(1), ..., u(n))$, que puede describirse como política y su correspondiente secuencia de estados $(x(1), ...x(n))$, denominada generalmente trayectoria, que minimiza la función costo

$$J = \sum_{k=1}^{n} L(x(k)), u(k), k), \tag{1}$$

donde

$$x(k+1) = g(x(k), u(k), k) \tag{2}$$

y sujeto al conjunto de restricciones de los estados y las variables de decisión, formuladas como

$$x \in X(k) \subset \Re^n, u \in U(x(k), k) \subset \Re^m$$

En este caso, x es la variable de estado, X es el conjunto de estados posibles, u es la variable de decisión (política), Ues el conjunto de decisiones admisibles, k es la etapa, y J es el costo de la función objetivo; L representa el costo de una etapa en particular.

Entonces, la función costo mínimo en la etapa k, en el final del problema de decisión, se define como

$$I(x,k) = \min_{u(k),u(k+1),...,u(n)} \left\{ \sum_{j=k}^{n} L(x(j), u(j), j) \right\} \tag{3}$$

Según el principio de optimalidad de Bellman [3], todas las porciones de una trayectoria óptima, son trayectoria óptimas en si mismo, es decir

$$\begin{aligned} I(x,k) &= \min_{u} \{ L(x(k), u(k), k) + I(g(k), u(k), k), k+1) \} \\ &= \min_{u} \{ L(x(k), u(k), k) + I(x(k+1), k+1) \} \end{aligned} \tag{4}$$

con $I(x,n) = \min_{u(n)} \{ L(x(n), u(n), n) \}$ para la última etapa

3. DESPACHO ÓPTIMO DE GENERADORES EN UNA CENTRAL HIDROELÉCTRICA

3.1. El Problema de Despacho Óptimo

La optimización del número de unidades generadoras en operación en una central hidroeléctrica, para una programación de horizonte diario, puede ser formulada a través de modelo dinámico de optimización discreta [1, 2] considerando el equilibrio entre la generación eficiente, el arranque y parada de unidades generadoras, por una función objetivo bicomponente

$$\min \sum_{t=1}^{T} \{ c_{ap}.|\ \Delta n_t| + c_p.p_n(d_t) \} \tag{5}$$

Sujeto a:

$$n_t = n_{t-1} + \Delta n_t \tag{6}$$

$$\underline{n_t}(d_t) \le n_t \le \overline{n_t}(d_t) \quad (7)$$

$$n_t \in \mathrm{N} \quad \mathrm{t} = 1, ..., \mathrm{N} \quad (8)$$

donde
- n_t es el número de unidades generadoras en operación en un instante t;
- $p_n(d_t)$ pérdida de potencia en función a n unidades en operación en el instante t;
- d_t generación programada en el instante t;
- c_{ap} costo de arranque y parada de una unidad ;
- c_p costo de pérdida de potencia por unidad generadora;
- T número de intervalo (24, para un día);
- N Conjunto de números naturales.

Observando las expresiones (5) al (8) se puede establecer la relación existente entre éstas y las planteadas en la sección 2, en las expresiones (1) al (4), considerando que:

$$x(k) = n_t;\ u(k) = n_t(d_t);\ x(k+1) = n_{t+1};\ k = t;\ n = T$$

$$\sum_{k=1}^{n} L(x(k)), u(k), k) = \sum_{t=1}^{T} \{c_{ap}.|\ \Delta n_t| + c_p.p_n(d_t)\}$$

El problema planteado por (5)-(8), puede ser resuelto de manera eficiente utilizando programación dinámica [2], de acuerdo al siguiente esquema:

Cada etapa corresponde a un instante de tiempo, discretizado en intervalos de 1 hora, la variable de estados es el número de unidades generadoras en operación en cada etapa y la variable de control es el número de arranques y paradas entre el número mínimo y máximo de unidades generadoras capaces de atender la generación programada para cada etapa.

3.2. Algoritmo Secuencial para Resolver el Problema

La solución secuencial del problema planteado se sustenta en las siguientes expresiones:

$$f_1(n_1) = \{c_{ap}.|\ n_1 - n_0| + c_p.p_{n_1}(d_1)\}, \quad \underline{n_t}(d_t) \le n_1 \le \overline{n_t}(d_t) \quad (9)$$

$$f_{t+1} = \min\{\lfloor\ c_{ap}.|\ n_t - n_{t-1}| + c_p.p_{n_t}(d_t)\rfloor + f_t(n_t)\} \quad t = 1, ..., T \quad \underline{n_t}(d_t) \le n_t \le \overline{n_t}(d_t) \quad (10)$$

n_0 es el número de unidades generadoras en producción en el instante $t = 0$, en el inicio de la programación y $f_t(n_t)$ es el costo mínimo acumulado en las etapas $1, 2, ..., t$, para un estado n_t.

Las expresiones (9) y (10) modelan la resolución secuencial cuyo algoritmo se puede apreciar en la figura 1.

Calcular los valores iniciales del problema
Calcular $f_1(n_1) = c_{ap}.|\ \Delta n_t| + c_p.p_{n_1}(d_1)$
Para $t = 2$ **hasta** T
$f_t(n_t) = \min\{c_{ap}.|\ \Delta n_t| + c_p.p_n(d_t)\} + f_{t-1}(n_{t-1})$
$n_t = n_{t-1} + \Delta n_t$
Fin Para
Imprimir resultados

Figura 1: Algoritmo secuencial para resolver (5)-(8).

4. ENFOQUE PARALELO DEL PROBLEMA DE DESPACHO ÓPTIMO

4.1. Porqué Paralelizar

Los varios horizontes de planificación y discretizaciones de los parámetros utilizados en el proceso de programación, obviamente definen el tamaño del problema, cuanto más amplio es el horizonte de la planificación, mayor es la cantidad de datos a tratar y la dimensión en el contexto del modelo de Programación Dinámica; por lo tanto, el estudio de las posibilidades de paralelización del método de despacho óptimo de carga, propuesto en [1], parece interesante.

4.2. Algoritmo Paralelo

La idea de la paralelización se basa en el modelo Maestro – Esclavo. Básicamente lo que se plantea es paralelizar la programación dinámica, del modelo aplicado en el despacho óptimo de máquinas. Análisis preliminares sugieren que la codificación menos complicada es la que paraleliza el método mencionado en el espacio de estados, en cada etapa de la programación dinámica, es decir, en cada etapa del proceso, se asignan porciones del espacio de estados a procesadores esclavos, de manera que simultáneamente busquen el costo mínimo para llegar a ese estado, considerando la programación "para adelante" o *forward* de la programación dinámica [3]. Los algoritmos utilizados en este trabajo para el procesador maestro y los procesadores esclavos se presentan en las figuras 2 y 3 respectivamente, los mismos están escritos con las directivas principales de comunicación de la biblioteca de paso de mensajes Open MPI [8] para el Octave [11].

Proceso MAESTRO:
Calcular los valores iniciales del problema
Iniciar los procesos esclavos (MPI_Init())
Enviar datos fijos a los procesos esclavos (MPI_Bcast())
Calcular $f_1(n_1) = c_{ap}.|\ \Delta n_t| + c_p.p_{n_1}(d_1)$
Enviar $f_1(n_1) = c_{ap}.|\ \Delta n_t| + c_p.p_{n_1}(d_1)$ **a todos los esclavos**(MPI_Bcast())
Para $t = 2$ **hasta** T
    Enviar $f_{t-1}(n_{t-1})$ **a todos los esclavos** (MPI_Bcast())
    Para $r = 1$ **hasta** m (número de esclavos)
        Enviar $(n_t)^r$ **al esclavo** r (MPI_Send())
        Recibir $(f_t(n_t))^r$ **del esclavo** r (MPI_Recv())
    Fin Para (r)
    Calcular $f_t(n_t) = \min\{f_t(n_t))^1, ..., f_t(n_t)^m\}$
    Calcular $n_{t+1} = n_t + \Delta n_{t+1}$
Fin Para (t)
Imprimir Resultados
Fin (MPI_Finalize)

Figura 2: Algoritmo del proceso Maestro

En el algoritmo de la figura 2, $(n_t)^r$ corresponde al número de máquinas en operación asignados al proceso esclavo r y $(f_t(n_t))^r$ corresponde al costo mínimo (local) en la etapa t en el proceso esclavo r

Proceso ESCLAVO:
Recibir los datos fijos del proceso MAESTRO (MPI_Bcast())
Recibir $f_{t-1}(n_{t-1})$ **del MAESTRO** (MPI_Bcast())
Recibir $(n_t)^r$ **del MAESTRO** (MPI_Recv())
Calcular $(f_t(n_t))^r = \min\{c_{ap}.|\ (n_t)^r - n_{t-1}| + c_p.p_n(d_t)\} + f_{t-1}(n_{t-1})$
Enviar $(f_t(n_t))^r$ **al MAESTRO** (MPI_Send())

Figura 3: Algoritmo del proceso Esclavo

Como se puede observar el conjunto de los algoritmos (figuras 2 y 3), resuelve el problema por etapas, es decir en cada etapa el proceso maestro recibe los mínimos locales de los esclavos y calcula en mínimo global de la función objetivo, balanceando la carga, pues espera que todos los esclavos terminen su tarea antes de enviar los datos para la siguiente etapa, de acuerdo a lo planteado en [4]. Ambos algoritmos fueron implementados en una red de computadoras. Los detalles de la implementación así como detalles de la plataforma computacional se presentan en la siguiente subsección.

4.3. Ambiente Computacional de la Implementación

Los algoritmos de las figuras 1 y 2 fueron codificados utilizando el lenguaje de programación del Octave [11], y las directivas de la biblioteca MPITB (MPITool Box) [10], basado en Open MPI [8], en un cluster de estaciones de trabajo (NOWS), construido con el Pelican HPC [12], versión 2.1.1. Las estaciones de trabajo son 11 computadoras idénticas con procesadores Pentium 4, Doble Núcleo de 2,6 GHz, 1GB de memoria RAM y disco SATA2 de 160 GB, interconectadas a través de un Switch de 100Mbps y aislada de cualquier otra red. Una de las máquinas actuaba como Maestro y las demás como Esclavos.

5. RESULTADOS EXPERIMENTALES

Como problema ejemplo se ha utilizado el despacho máquinas para los horizontes de programación de un día, una semana, dos semanas, un mes y seis meses, de la usina hidroeléctrica de Itaipú. La discretización para todos los caso fue de una hora y utilizando las 20 unidades generadoras de 700MW de la usina. Como medida de desempeño de la propuesta en el contexto paralelo se utilizó el speedup (Sp), definido como:

$$Sp = \frac{tp}{ts} \tag{11}$$

donde ts es el tiempo en segundos de la resolución en un solo procesador y tp tiempo en segundos de la resolución con p procesadores.

Los resultados obtenidos de Sp y tiempo de ejecución para cada caso, se pueden apreciar en las figuras 4 y 5 para los horizontes mencionados.

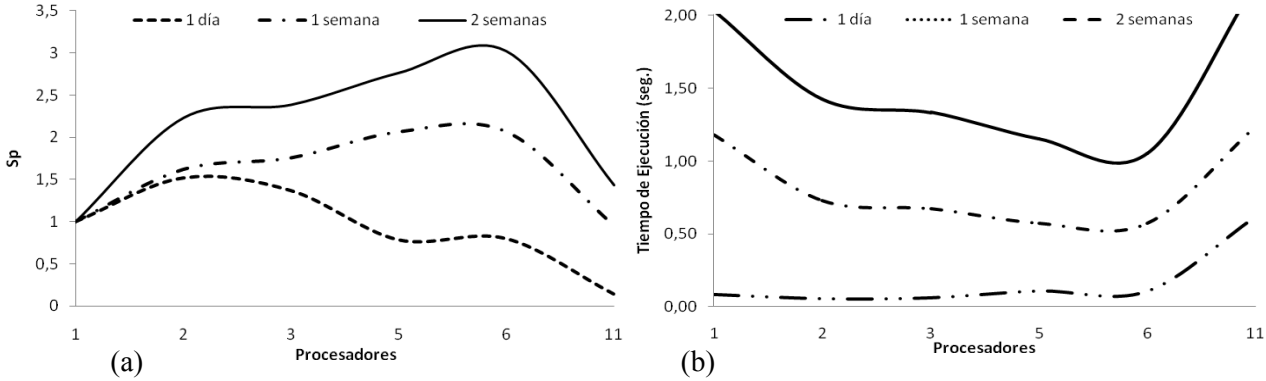

Figura 4: (a) *Speedup*, (b) Tiempo de Ejecución

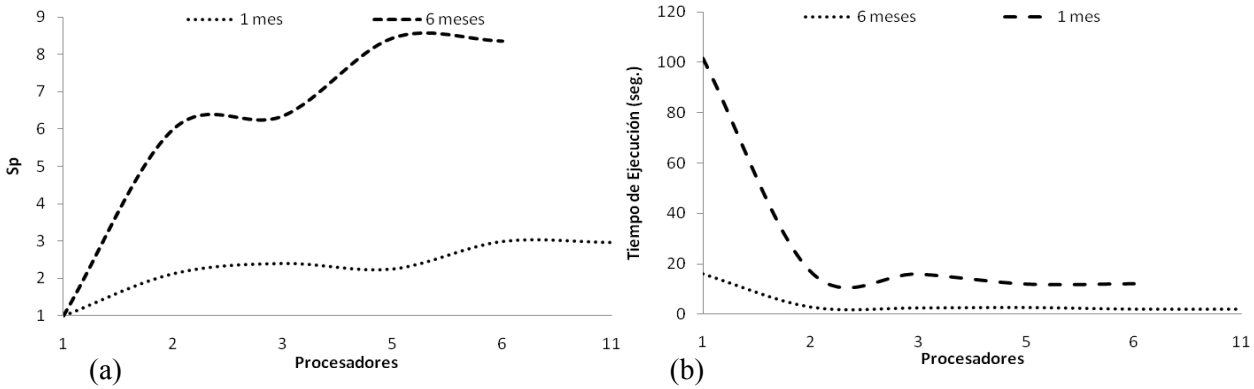

Figura 5: (a) *Speedup*, (b) Tiempo de Ejecución

En la Figura 4 (a) se puede apreciar que el *Speedup* logrado en la resolución paralela del problema de despacho óptimo de máquinas, minimizando pérdidas, arranques y paradas, es quasi lineal, cuando son empleados dos procesadores para los horizontes de un día, una semana y dos semanas, sin embargo con la incorporación de más procesadores el *Speedup* tiende a degradarse, hecho que también se puede observar en el gráfico (b) de la figura 4, pues los tiempos de ejecución crecen abruptamente a partir de la incorporación del sexto procesador en la solución. Esto se debe a que a este nivel de paralelismo la cantidad de mensajes en la red, degradan su eficiencia.

Por otro lado en la figura 5 (a) se puede apreciar que el *Speedup* presenta una tendencia creciente para los horizontes de programación de un mes y seis meses respectivamente, inclusive con un importante crecimiento para el caso específico del horizonte de programación de seis meses, siendo éste caso el mejor resultado obtenido, lo que implica que para el referido horizonte, considerando las veinte máquinas de la usina de Itaipú, se obtiene una importante ganancia en términos de tiempo de procesamiento, que de hecho que también se puede corroborar en el gráfico (b) de la figura 4.

6. CONCLUSIONES

En términos generales, el problema de despacho óptimo de máquinas de una central hidroeléctrica, que generalmente se resuelve a través de programación dinámica, puede ser resuelto en un ambiente de programación paralela como se plantea en este trabajo. Sin embargo las características particulares del problema tratado hace que cuando se paraleliza el método de Programación Dinámica aplicado al mismo,

exige la comunicación de una gran cantidad de datos a través de la red, lo que hace que la ganancia en términos de tiempo se resienta cuando el tamaño del problema es relativamente pequeño y la cantidad de procesadores involucrados crece. En contrapartida, la tendencia creciente del *Speedup* cuando se trata problemas con horizontes más extensos, así como la tendencia decreciente de los tiempos de resolución de estos casos, alientan el estudio más profundo de la paralelización del método de despacho óptimo de máquinas para una central hidroeléctrica, considerando la inclusión de nuevas variables que hacen a modelos más precisos y complejos del empleado en este estudio.

7. REFERENCIAS BIBLIOGRÁFICAS

[1] Arce A., Ohishi T. and Soares S., "Optimal Dsipatch of generating Units of the Itaipú Hydroelectric Plant", IEEE Trans. on Power Systems, Vol 17, No. 1, Febrero de 2002

[2] Arce A., *Um modelo de Otimizacão do Despacho de Máquinas em Usinas Hidrelétricas,* Universidade Estadual de Campinas, noviembre de 1999.

[3] Bellman R., "The Theory of Dinamic Programing", Bull. Amer. Math. Soc. Volume 60, Number 6 (1954), 503-515.

[4] Dormido S. et al, "Parallel Dynaming Programming on Cluster of Workstations", IEEE Trans. on Parallel and Distributed System, Vol. 16,No. 9, setiembre de 2005

[5] Feng L. , "A Parametric Iteration Method for Sthochastic Dynaming Programming for Optimal Dispatch of Hydroelectric Plants", IEE 2nd International Conference in Power System Control, Operation and Management, Hong Kong, Diciembre de 1993

[6] Fung C., Chow S., Wong K., "A low Cost Parallel Computibg Platform for Power Engineering Applications", Proceedings of the 5th. International Conference on Advances in Power System Control and Management, APSCOM 20, Hong Kong, octubre de 2000.

[7] Itaipú Binacional [en línea] <http://www.itaipu.gov.py > [junio de 2010]

[8] A High Performance Message Passing Library [en línea] < http://www.open-mpi.org/ > [Junio de 2010]

[9] Moreno L., Acosta L., Sánchez J., "Designo f algorithms for Spatial-time Reduction Complexity of Dynaming Programming", IEE Proceedings-D, Vol 139, No.2, marzo de 1992

[10] Morris L., MATLAB Laboratory for MPI toolbox (MPITB) [en línea] http://www.math.hkbu.edu.hk/math2160/matlab4/MPITB.pdf [julio de 2010]

[11] Octave [en línea] http://www. www.gnu.org/software/octave/ [junio de 2010]

[12] PelicanHPC GNU Linux [en linea] http://pareto.uab.es/mcreel/PelicanHPC/ [mayo de 2010]

[13] Rodríguez C. et al., "Paradigms for Parallel Dynamic Programming", Proceedings of Euromicro -22, 1996

[14] Shou-Hsuan S., Hongfei L, Venkatraman V., "Parallel Dynaming Programming" ,IEE Transactions on parallel and Distributed Systems, Vol 5, No. 3, marzo de 1994