Calendarización Multi-Criterio en Grid Computacional de Dos Niveles con Algoritmos Genéticos

[1]Yair Castro García, [1]Andrei Tchernykh, [2]Victor Yaurima
[1]*Centro de Investigación Científica y de Educación Superior de Ensenada, Ensenada, B.C.*
{ycastro, chernykh}@cicese.mx,
[2]*Centro de Estudios Superiores del Estado de Sonora. San Luis R.C., Sonora.*
vyaurima@yahoo.com

Resumen

El presente trabajo se enfoca en el problema de calendarización multi-criterio de trabajos paralelos en un Grid computacional jerárquico con dos niveles. En el primer nivel se realiza una asignación de trabajos a recursos. En el segundo nivel, cada recurso aplica una estrategia de calendarización local. Se propone un algoritmo genético como estrategia de asignación en el primer nivel. Se adopta un método de agregación de criterios. Dicho método toma en cuenta las preferencias de los participantes en el proceso de calendarización. Se presenta un análisis experimental con base en cargas de trabajos reales. Los resultados se comparan con resultados de estrategias de calendarización conocidas en la literatura.

1. Introducción

El paradigma Grid comprende recursos distribuidos geográficamente que pertenecen a diferentes organizaciones, cuentan con sus propias políticas de acceso, costos y restricciones [11]. Este paradigma se enfoca en resolver problemas que solicitan poder de cómputo. Algunos de los beneficios que se logran son: procesar aplicaciones en paralelo reduciendo su tiempo de ejecución, reducir costos de producción, cumplir con los tiempos de entrega de los productos e incrementar ganancias económicas, entre otros. El Grid computacional está compuesto por una variedad de recursos heterogéneos (*clusters* y súper-computadoras). La eficiencia de las estrategias de calendarización es crucial para el desempeño del Grid. La calendarización de trabajos para una sola computadora paralela difiere significativamente de la calendarización para un Grid computacional. El problema se complica cuando las computadoras son de diferentes tamaños y aplican diferentes estrategias de calendarización [13][15][27]. Una solución posible es aplicar un esquema de calendarización de dos niveles [25] (Figura1). En el primer nivel, los trabajos se asignan a computadoras por un meta-calendarizador y posteriormente, cada computadora aplica su propio calendarizador. Una responsabilidad del meta-calendarizador es proveer acceso a los recursos distribuidos por medio de servicios intermedios (middleware) actuando como un mediador entre los usuarios y los recursos, su intención es presentar los recursos del Grid al usuario como un recurso unificado [25]. Desde los inicios del Grid computacional en los 90s [11] se han investigado una amplia variedad de estrategias para calendarización [25] suponiendo diferentes modelos de Grid (centralizados o descentralizados), jerarquías, estrategias de asignación (co-asignación) y variedad de trabajos.

En el presente trabajo se considera el problema de calendarización fuera de línea (cuando todos los trabajos están disponibles para ser calendarizados) multi-criterio que optimiza cuatro criterios simultáneamente. Se propone un algoritmo genético (AG) como estrategia de asignación de los trabajos a recursos. Se aplica un método de agregación de los criterios y una función generadora de pesos que representan la relativa importancia de cada criterio. Se presenta un análisis experimental y los resultados se comparan con estrategias enfocadas en optimizar un solo criterio. Se utiliza una carga de trabajo generada con base en cargas reales de varios centros de cómputo.

El resto del documento se encuentra organizado de la siguiente manera: la Sección 2 contiene el trabajo relacionado y la Sección 3 los antecedentes. La Sección 4 contiene la definición del problema y los criterios que se consideran. La Sección 5 describe el AG. La Sección 6 muestra los parámetros de simulación y los resultados experimentales. Finalmente, la Sección 7 contiene las conclusiones y trabajo futuro.

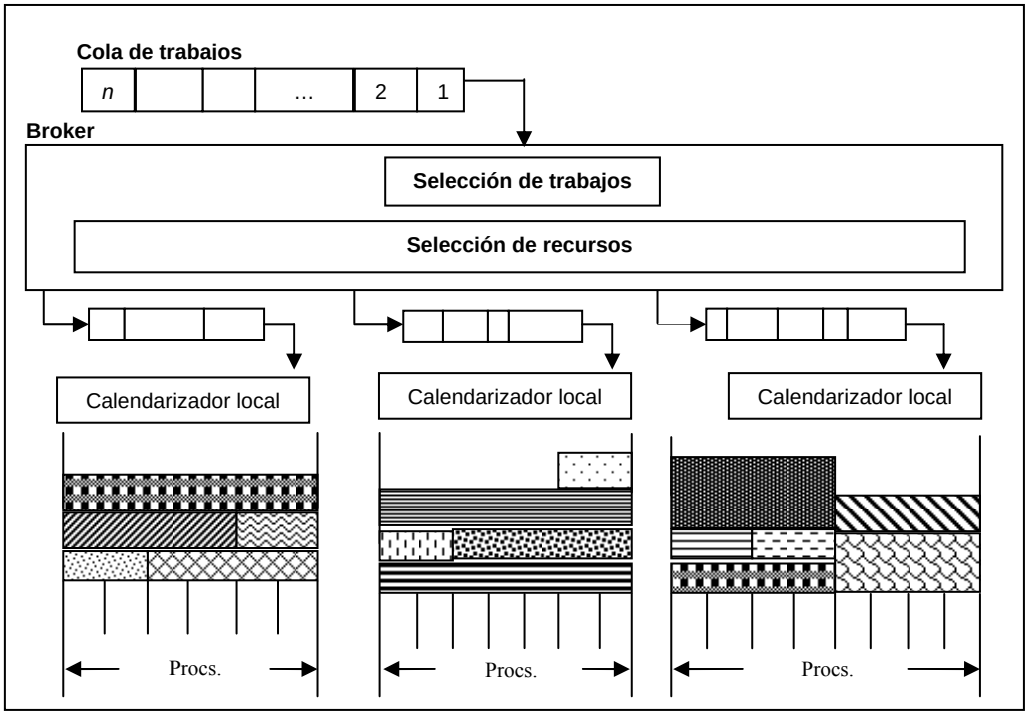

Figura 1. Grid computacional de dos niveles.

2. Trabajos relacionados

Varios trabajos que abordan el problema de calendarización en Grid aplican estrategias definidas por los administradores de sistemas considerando únicamente un criterio (e.g. EGEE Workload Management System [3], NorduGrid bróker [9], eNANOS [25], Gridway [17]). Sin embargo, parte de la comunidad científica está realizando esfuerzos para aplicar métodos que mejoren los resultados cuando se consideran múltiples criterios simultáneamente. En [19] abordan el problema de calendarización en Grid cuando no se conocen los tiempos de procesamiento de los trabajos (calendarización no clarividente). Aplican el método de agregación de los criterios para modelar las preferencias de los participantes (propietarios, administradores de los sistemas y usuarios). En [18], abordan el problema de calendarización con múltiples criterios en un sistema Grid computacional; consideran un Grid jerárquico de dos niveles; toman en cuenta las preferencias de los participantes, suponen que desconocen los tiempos de procesamiento de los trabajos; y estudian el impacto del tamaño de los conjuntos de trabajos (lotes) en la eficiencia de la calendarización. Utilizan cargas sintéticas y un algoritmo del tipo voraz (*greedy*); concluyen que el desempeño de las estrategias depende de los parámetros de la carga de trabajo. En [21] presentan estrategias de asignación de trabajos hacia los diferentes sitios de un Grid y proponen un modelo para calendarización de trabajos considerando múltiples criterios. Aplican un AG para generar calendarios; muestran que las soluciones encontradas minimizan el tiempo de espera promedio y el tiempo máximo de finalización de los trabajos; concluyen que la calendarización considerando varios criterios puede realizarse de manera eficiente aplicando AGs. En [5][6][7] los autores efectúan la calendarización de trabajos sobre los recursos de un *cluster* considerando el tiempo de finalización máximo de los trabajos y el tiempo promedio de finalización. En [23] toman en cuenta las preferencias de los usuarios para una calendarización de trabajos basada en la negociación de recursos usando reservación anticipadamente. Las preferencias de los usuarios se modelan con base a una función de utilidad en la que los usuarios deben especificar los valores y niveles de negociación.

3. Antecedentes

El principal inconveniente de las estrategias de calendarización para Grid es que se enfocan en optimizar un solo objetivo: minimizar el tiempo de finalización de todos los trabajos, minimizar el tiempo promedio de permanencia de los trabajos en el sistema, minimizar el tiempo que en promedio esperan los

trabajos para ejecutarse, maximizar la eficiencia de las computadoras o maximizar el número de trabajos procesados por unidad de tiempo. Sin embargo en los ambientes Grid reales, generalmente los participantes en el proceso de calendarización tienen diferentes objetivos, ello genera conflicto de intereses, por lo que es importante implementar estrategias que mejoren los intereses de todos los participantes simultáneamente. A diferencia de un problema de un solo objetivo en donde se desea encontrar una solución, en un problema multi-objetivo se pretende encontrar un conjunto de soluciones que son causadas por los objetivos en conflicto. Un problema multi-objetivo tiene varios componentes: las variables de decisión son las componentes de un vector $x=[x_1, x_2, ..., x_n]^T$ cuyos valores se deben escoger para la optimización. Las restricciones son parte inherente del problema, pueden ser físicas, de tiempo o de otro tipo. Estas restricciones se deben satisfacer para que la solución sea apropiada. Pueden ser restricciones de igualdad o desigualdad:

$$g_i(x) \geq 0,\ i=1,2,...,z \quad (1)$$

$$h_i(x) = 0,\ i=1,2,...,p \quad (2)$$

donde z y p indican el número de restricciones. Las restricciones fuertes se deben cumplir de manera estricta. Las restricciones flexibles pueden no cumplirse en su totalidad, sin embargo, se deben considerar para mejorar la calendarización porque dirigen el proceso para seleccionar las mejores soluciones [18]. Las Funciones objetivo son las funciones que se deben optimizar (maximizar o minimizar). Se puede ver como un vector de la forma: $f(x)=[f_1(x), f_2(x),..., f_u(x)]^T$. Las variables de decisión y las funciones objetivo definen dos espacios a considerar: el espacio de las variables de decisión x (*decision space*), el cual es de dimensión a. Cada punto en este espacio corresponde a un vector x. El espacio de criterios $f(x)$ (*criteria space*), el cual es de dimensión b. Cada punto es la correspondiente imagen, a por medio de $f(\square)$, de una solución x al problema.

Definición 1. El Punto Ideal o Vector Ideal z^* que está compuesto por los mejores valores objetivo que se pueden obtener, $z_j^* = \max/\min\{f_j(x) \mid x \in A\}$, A es el conjunto de todos los posibles valores de x, $j=1,2,...,m$, donde $m>1$ es el número de funciones objetivo.

Definición 2. Dos funciones objetivo están en conflicto si la distancia Euclidiana desde el punto ideal al conjunto de los mejores valores de las soluciones apropiadas es diferente de cero.

Definición 3. Tres o más funciones objetivo están en conflicto si están en conflicto por pares.

Por razones de simplicidad, normalmente todas las funciones se convierten a la forma de maximización o minimización. Un problema multi-objetivo se resuelve encontrando un conjunto de vectores que se pueden comparar entre sí. Enseguida se presentan las relaciones entre dos vectores (cuyos valores se deben escoger para la optimización).

Para dos vectores cualesquiera $v=[v_1, v_2, ..., v_u]^T$ y $w=[w_1, w_2, ..., w_u]^T$ existen cuatro relaciones entre ellos. Estas relaciones son de utilidad, ya que cada solución de un problema mult-objetivo se puede ver como un vector de u componentes, (número de funciones objetivo). Cada componente del vector es el valor que toma la función objetivo respectiva. Enseguida se describen tales relaciones ejemplificando para el caso de minimización, sin pérdida de generalidad se puede aplicar a maximización:

El vector v *domina* al vector w ($v \prec w$). Para cada componente de w los componentes de v son iguales o mejores ($v_i \leq w_i$ para todo $i=1,2,...u$; y $v_i \neq w_i$ para al menos un i). El vector v *es dominado* por el vector w ($v \succ w$). Para cada componente de v los componentes de w son iguales o mejores ($v_i \geq w_i$ para todo $i=1,2,...u$; y $v_i \neq w_i$ para al menos un i). El vector v *es igual* al vector w ($v = w$). En cada uno de los componentes de v, los componentes de w son los mismos ($v_i = w_i$ para todo $i=1,2,...u$). El vector v *no el comparable* con el vector w ($v \succ\prec w$). En este caso se dice que ambos vectores son igual de buenos o igual de malos ($v_i \leq w_i$ y $v_j \geq w_j$ para algún $i, j=1,2,...u$ y $v \neq w$).

El conjunto de soluciones que se quiere encontrar se denomina frente Pareto:

Definición 4. Para un problema de optimización multi-objetivo, el conjunto óptimo de Pareto (P_{true}) es $p_{true} = \{x \in A \mid \nexists y \in A\ f(y)\ f(x)\}$, A es el conjunto de soluciones.

Definición 5. Para un problema de optimización multi-objetivo y un conjunto óptimo Pareto (P_{true}), el *frente Pareto* se define como:

$$PF_{true} = \{u = f(x) = [f_1(x), f_2(x), ..., f_m(x)]^T \mid x \in P_{true}\}$$

Entonces, un problema multi-objetivo se define de la siguiente manera:

Definición 6. *Problema multi-objetivo*. Encontrar los vectores $x = \{x_1, x_2, ..., x_n\}$ que satisfacen las z y p restricciones de desigualdad correspondientemente

$$g_i(x) \geq 0,\ i = 1, 2, ..., z \qquad (3)$$

$$h_i(x) = 0,\ i = 1, 2, ..., p \qquad (4)$$

y optimizan el vector compuesto por u funciones $f(x) = [f_1(x), f_2(x), ..., f_u(x)]^T$. Donde optimizar $f(x)$ significa encontrar el conjunto óptimo de Pareto.

La solución de un problema multi-objetivo es un conjunto de soluciones que tienen el mejor valor posible en todas las funciones objetivo, y que no pueden ser mejores entre ellas. Este conjunto se llama Frente Pareto (soluciones no dominadas). El conjunto óptimo de Pareto se encuentra en el espacio de decisión y el frente Pareto se encuentra en el espacio de criterios.

Existen diversos métodos de resolver un problema multi-objetivo: la intención del enfoque basado en la población (*Population-Based Approaches*) es diversificar la búsqueda por medio de varios grupos de soluciones (poblaciones. El enfoque basado en Pareto (*Pareto-based Approaches*) engloba los algoritmos evolutivos multi-objetivo que cuentan con el concepto de Pareto óptimo. La principal restricción de estos métodos por la manera en que operan es el tiempo que consumen para encontrar soluciones, lo cual afecta a los sistemas Grid ya que la toma de decisiones deben hacerse lo más rápido posible. Las Funciones de agregación lineal (*Linear Aggregating Functions*) encajan en los problemas multi-criterio y en el ambiente Grid porque toma en cuenta los requisitos y preferencias de los participantes en el proceso de calendarización. Típicamente, la función de agregación de criterios tiene la forma de una sumatoria ponderada. Este método es aceptable para encontrar una solución atractiva en problemas multi-criterio si el tiempo disponible para la búsqueda de soluciones es corto. Para alguna función de aptitud y algún conjunto de pesos positivos, el óptimo global que se obtiene, siempre es un elemento del conjunto Pareto óptimo. Este enfoque no requiere cambios para aplicarlo con AGs.

La principal suposición en el presente trabajo es que el meta-calendarizador toma la decisión final (asignación de recursos a los trabajos). Por estas razones aplicamos el método de agregación de los criterios.

En el presente trabajo se aplica el operador promedio ponderado de orden (*Ordered Weighted Averaging*, *OWA*) [18]:

$$OWA(x_1, x_2, ..., x_n) = \sum_{j=1}^{k} w_j s(x)_{\sigma(j)} \qquad (5)$$

donde $x_j,\ x_j j = 1, ..., k$ es un valor asociado a la satisfacción del *j-ésimo* criterio. Se realiza una permutación (σ) que ordena los valores: $s(x)_{\sigma(1)} \leq s(x)_{\sigma(2)} \leq ... s(x)_{\sigma(k)}$. Los pesos (w_j) son no negativos y $\sum_{j=1}^{k} w_j$. Al poner un peso w_1 en 1 y el resto de los pesos en 0, el operador *OWA* ayuda a minimizar. En el análisis multi-criterio, ello indica que se buscan los mejores valores para todos los criterios. Si a todos los pesos se les asigna el mismo valor, el operador *OWA* se comporta como el promedio aritmético. En este caso, los valores altos de algún criterio compensan los valores bajos de los otros criterios. El peso más grande es w_1 y los subsecuentes se van decrementando pero nunca son 0. Significa que se toman en cuenta los casos anteriores, el caso del peor criterio y el caso promedio. Ello representa la posibilidad de evaluar calendarios que toman en cuenta varios criterios. La idea es encontrar un esquema de pesos que proporcionen el mejor valor promedio para todas las funciones de evaluación y el valor más alto para la peor función de evaluación. Para lograrlo, el peso w_1 debe ser relativamente grande, mientras el peso w_k debe ser chico. El resto de los pesos intermedios deben decrementarse de acuerdo a la función 18:

$$(w_j) = \begin{cases} \dfrac{3}{2k}, & j = 1 \\ \dfrac{3k - 2i - 1}{2n(k-1)}, & 1 < j \leq k \end{cases} \qquad (6)$$

Los criterios no se pueden comparar en su forma original debido a que están expresados en diferentes unidades y escalas. Los métodos escalares se utilizan para asegurarse que las funciones realmente reflejen la importancia de cada criterio [18]. En el presente trabajo se utiliza la función escalar $s(x)$ que distribuye los valores de los criterios en el intervalo (0,1):

$$s(x) = \frac{s(x)}{\max_{s(x)}} \qquad (7)$$

4. Definición del problema

El problema de calendarización de trabajos en el Grid se plantea de la siguiente manera: se deben calendarizar n trabajos paralelos $J_1, J_2, ..., J_n$ independientes en m máquinas del Grid computacional $M_1, M_2, ..., M_m$; M_i denota el número de procesadores idénticos en la máquina M_i. Se realiza una indización de las máquinas paralelas en orden ascendente de acuerdo a su tamaño $M_1 \leq M_2 \leq, ..., \leq M_m$. Cada trabajo j_j se describe por una tripleta $(r_j, size_j, p_j)$, donde r_j es el tiempo a partir del cual el trabajo j_j está disponible para ser procesado. Su tamaño (número de procesadores sobre los que se ejecuta) $size_j$ está en el intervalo $1 \leq size_j \leq m_m$ (indica su grado de paralelismo); el tiempo de ejecución se denota por p_j y se procesa exclusivamente sobre $size_j$ procesadores. Se supone que no hay dependencia entre los trabajos y se pueden ejecutar en algún momento, en algún orden y sobre alguna máquina. También se supone una calendarización en modo de espacio compartido, donde para cada trabajo j_j, el área de trabajo se denota como $p_j \cdot size_j$. Todos los trabajos están disponibles al instante cero $(r_j = 0)$ y se procesan en un mismo lote. El tiempo de ejecución no se conoce hasta que el trabajo finaliza su procesamiento. Se supone que el trabajo j_j puede ejecutarse en la máquina M_i si se cumple la restricción $size_j \leq m_i$. Un trabajo paralelo j_j se ejecuta exactamente sobre $size_j$ procesadores sin interrupciones.

Los criterios que se consideran son cuatro: minimizar el tiempo de finalización de todos los trabajos procesados ($C_{\max}$, *makespan* o longitud del calendario):

$$f_1 = C_{\max} = \max(C_i), i = 1, 2, 3, ..., M_m \qquad (8)$$

donde C_i es el tiempo de finalización máximo en la máquina M_i. Minimizar el tiempo promedio de permanencia de los trabajos en el sistema (*Mean turnaround* o *Mean flow time*):

$$f_2 = TA = \frac{1}{n}\sum_{j=1}^{n}(c_j - r_j) \qquad (9)$$

donde c_j es el instante de finalización del trabajo. Minimizar el cociente de respuesta promedio (*Mean response ratio* o *Mean Slowdown*):

$$f_2 = MRR = \frac{1}{n}\sum_{j=1}^{n}\left(\frac{c_j - r_j}{p_j}\right) \qquad (10)$$

Finalmente, minimizar el tiempo que en promedio esperan los trabajos para ejecutarse (*Mean waiting time*):

$$f_4 = MWT = \frac{1}{n}\sum_{j=1}^{n}(c_j - r_j - p_j) \qquad (11)$$

La notación corta de tres campos para el problema de calendarización en Grid se describe como: $GP_m \mid r_j = 0, size_j, p_j \mid OWA$. OWA es el operador de agregación de los criterios ($OWA = w_1 C_{\max} + w_2 TA + w_3 MRR + w_4 MWT$), w_i es la combinación lineal de pesos. La notación MPS (*Multi-Parallel-Scheduling*) hace referencia a los dos niveles de calendarización en Grid: $MPS = MPS_{Alloc} + PS$ [14, 28]. En el primer nivel (MPS_{Alloc}) se asigna una máquina disponible para cada trabajo utilizando alguna estrategia de selección de recursos. En el segundo nivel se aplica un algoritmo PS (*Parallel-Scheduling*) a cada máquina con el conjunto de trabajos que le fueron asignados a partir del primer nivel. El problema de calendarización en el segundo nivel se denota con $P_m \mid r_j = 0, size_j, p_j \mid C_{\max}$ [26].

En el presente trabajo, se utiliza el factor de competitividad (*Competitive factor*) para analizar los algoritmos. Sea $C^*_{\max}$ y $C_{\max}(A)$ las longitudes de los calendarios óptimo y el determinado por el algoritmo A, respectivamente. El factor de competitividad del algoritmo A se define como $\rho_A = \max\left(\frac{C_{\max}(A)}{C^*_{\max}}\right)$ para todas las instancias del problema. El $C^*_{\max}$ es el máximo de dos posibilidades: el máximo tiempo de procesamiento de un trabajo o el cociente de la suma de las áreas de todos los trabajos por el número de procesadores.

5. Algoritmo genético para calendarización de trabajos

Los AGs propuestos se basan en el proceso evolutivo de los organismos [16]. En el presente trabajo, la función de los AGs es asignar recursos a trabajos, similar a la que realizan las estrategias denominadas: menor tiempo de finalización (*Min Completion Time, Min_CT*), menor tiempo de inicio (*Min Start Time, Min_ST*), menor tiempo de permanencia en el sistema (*Min Turnaround, Min_TA*) y menor tiempo de espera (*Min Waiting Time, Min_WT*). La ventaja de los AGs en relación a dichas estrategias, es que tienen la capacidad de simular el intercambio de trabajos entre máquinas, en cambio, una vez que las otras estrategias asignan recursos a los trabajos ya no los pueden reasignar.

Población inicial. Para generar la población inicial para cada trabajo se identifica su tamaño; se selecciona una máquina aleatoriamente del intervalo de máquinas que cumplen con la solicitud. Para buscar una máquina, se supone que el conjunto de ellas está en un arreglo ordenado de menor a mayor tamaño, se realiza una búsqueda secuencial hasta encontrar la primera máquina que cumpla con la solicitud. Este procedimiento se realiza hasta terminar de asignar los trabajos. Se repite el proceso hasta completar el tamaño de la población total T_p.

Solución. Una definición apropiada de un problema permite codificar las soluciones en vectores denominados individuos. Utilizar una representación adecuada permite simplificar el proceso de búsqueda de un AG. Por lo tanto, se requieren representaciones que tengan la capacidad de generar todas las soluciones posibles para un problema. El uso adecuado de la representación permite ahorrar trabajo innecesario y concentrarse en la aplicación del AG en lugar de enfocar esfuerzo en el diseño de operadores genéticos (OG) especiales. En calendarización de trabajos, la aptitud de un individuo se relaciona directamente con el calendario [24], entonces, el objetivo es encontrar el calendario más corto posible. Cada solución se codifica en un vector bidimensional (Figura 2). Los renglones representan las m máquinas que conforman al Grid. Las columnas indican los n trabajos. En los índices de la matriz se muestran los identificadores de los trabajos a procesar (respetando el orden) sobre la máquina en que se encuentran.

Asignación de aptitud. Como función de aptitud se aplica el operador OWA (Sección 3) ya que asigna un valor escalar a cada individuo, el cual refleja su aptitud. La mayor aptitud indica que se encuentra el calendario más corto.

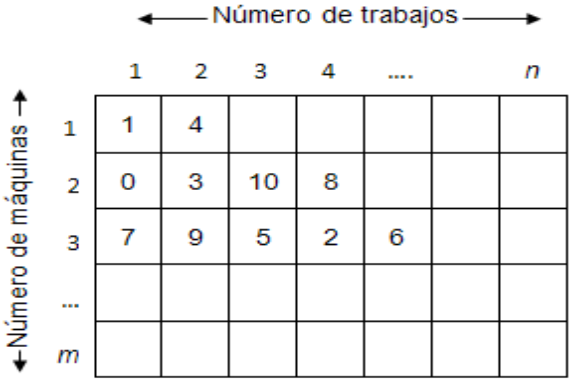

Figura 2. Solución codificada en un vector de dos dimensiones.

5.1. Operadores genéticos

Enseguida se describen los operadores genéticos:

Operador de cruzamiento. El operador de cruzamiento se aplica bajo cierta probabilidad (P_c).OBX (*Order Based Crossover*) (Figura 3) conserva el orden de los bloques que se van formando durante las iteraciones del AG [12]. Se genera una máscara binaria al azar para identificar a los elementos (columnas) del Padre 1 que se heredan al Hijo 1. Después, se identifican los elementos que aun no se encuentran en el Hijo 1 y se copian a este último respetando el orden. Finalmente, el Hijo 1 cuenta con elementos heredados de ambos padres y sin elementos repetidos. Antes de iniciar la operación de cruzamiento se revuelve la población de padres con la intención de que cada padre tenga la misma oportunidad de ser recombinado con otro padre, después, se toman dos padres contiguos para generar dos descendientes. Cada nuevo descendiente se genera a partir de una máscara distinta, ello con la intención de abarcar un mayor espacio en la búsqueda de soluciones

Operadores de mutación. El operador de mutación (Figura 4) genera cambios pequeños en un hijo, se aplica bajo cierta probabilidad (P_m) y se considera como secundario [4]. Su principal objetivo es evitar la convergencia prematura a un óptimo local. En [12] se explican los operadores *Insert*, *Swap* y *Switch* para vectores unidimensionales.

Debido a las posibilidades múltiples que se presentan por la doble dimensión, se pueden aplicar los operadores tomando en cuenta una sola cola (renglón) o dos colas (dos renglones). A continuación se describen dichos operadores: *1Q-Insert*: Seleccionar un renglón *q*l al azar y aplicar el operador *Insert* sobre *q*l (Figura 4-a). *2Q-Insert*: seleccionar dos renglones *q*l y

*q*2 al azar, seleccionar una posición *s*1 de *q*1. Insertar el elemento de *s*1 en una posición *s*2 seleccionada al azar en *q*2 (Figura 4-b). *1Q-Swap*: seleccionar un renglón *q*1 al azar y aplicar el operador *Swap* sobre *q*1 (Figura 4-c). *2Q-Swap*: seleccionar dos renglones *q*1 y *q*2 al azar, seleccionar una posición *s*1 de *q*1 y *s*1 de *q*2 e intercambiar los elementos de *q*1 y *s*2 (Figura 4-d). *1Q-Switch*: seleccionar un renglón *q*1 al azar y aplicar el operador *Switch* sobre *q*1 (Figura 4-e). *2Q-Switch*: seleccionar dos renglones *q*1 y *q*2 al azar, seleccionar una posición *s*1 (misma posición para ambos), intercambiar los elementos de *q*1 y *q*2 que se encuentran en la posición *s*1 (Figura 4-f).

Los parámetros de los OG descritos son:, $T_p = 100$, $P_c = 0.9$ y $P_m = 0.01$, el criterio de paro se lleva a cabo cuando se cumplen 10 veces sin cambio en el resultado de la función objetivo y el reemplazo de la población es generacional.

Operador de selección. Se implementó el operador denominado torneo determinístico [8] que se lleva a cabo bajo los siguientes pasos: revolver los individuos de la población, escoger un número r de individuos (comúnmente 2), se seleccionan los que tienen mayor aptitud, el ganador del torneo es el individuo con mayor aptitud.

6. Análisis experimental

6.1. Carga de trabajo

Los experimentos se basan en una carga de trabajo generada a partir de los registros de los siguientes centros [10]: Cornell Theory Center, High Performance Computing Center North, Swedish Royal Institute of Technology, Los Alamos National Lab, y Una proviene del *Grid Workload Archive* (*GWA*) [2]: The Advanced School for Computing and Imaging. La carga está normalizada con base en la zona horaria mínima encontrada en los registros (GMT-7).

Respecto al tamaño de los trabajos sometidos, la mayoría son menores a 16, siendo los de tamaño 1, 2 y 4 los que predominan (Figura 5). Los tamaños siguen el patrón con base en la potencia de dos. El tipo común de trabajos que predomina es aquel que cuenta con un tiempo de procesamiento de hasta 20 horas, aunque los trabajos pequeños tienden a acumularse en los tiempos de menor ejecución (Figura 6). Los trabajos con potencia de dos son los que cuentan con mayor tiempo de ejecución y destacan los secuenciales.

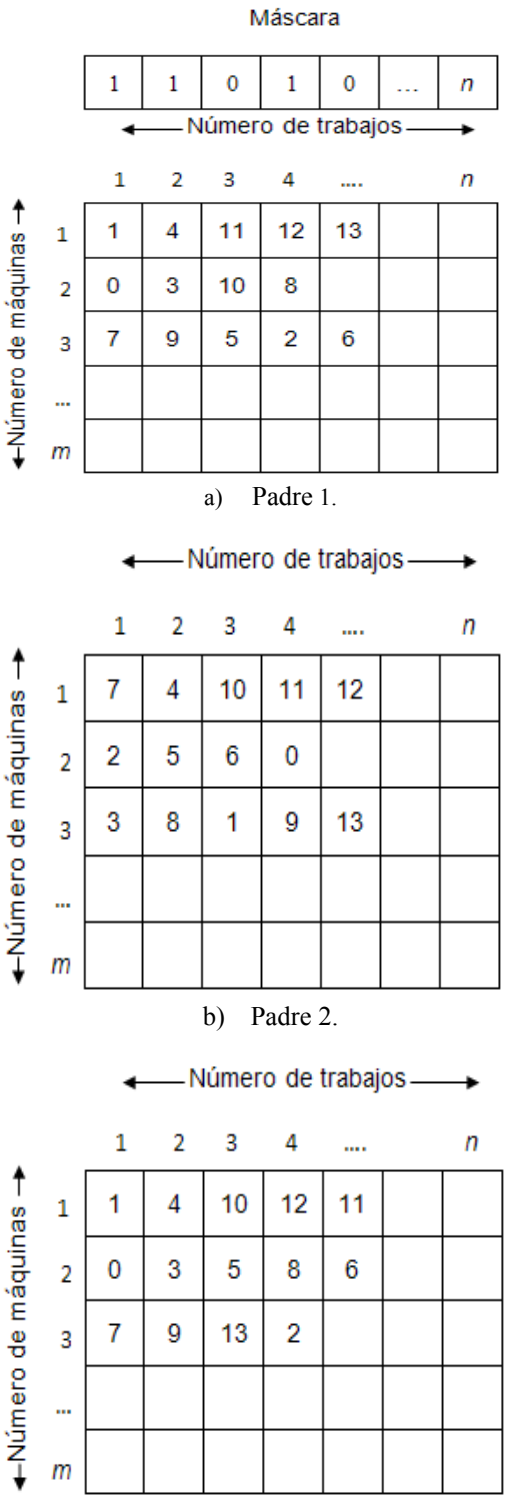

Figura 3. Operador de cruzamiento basado en el orden (OBX).

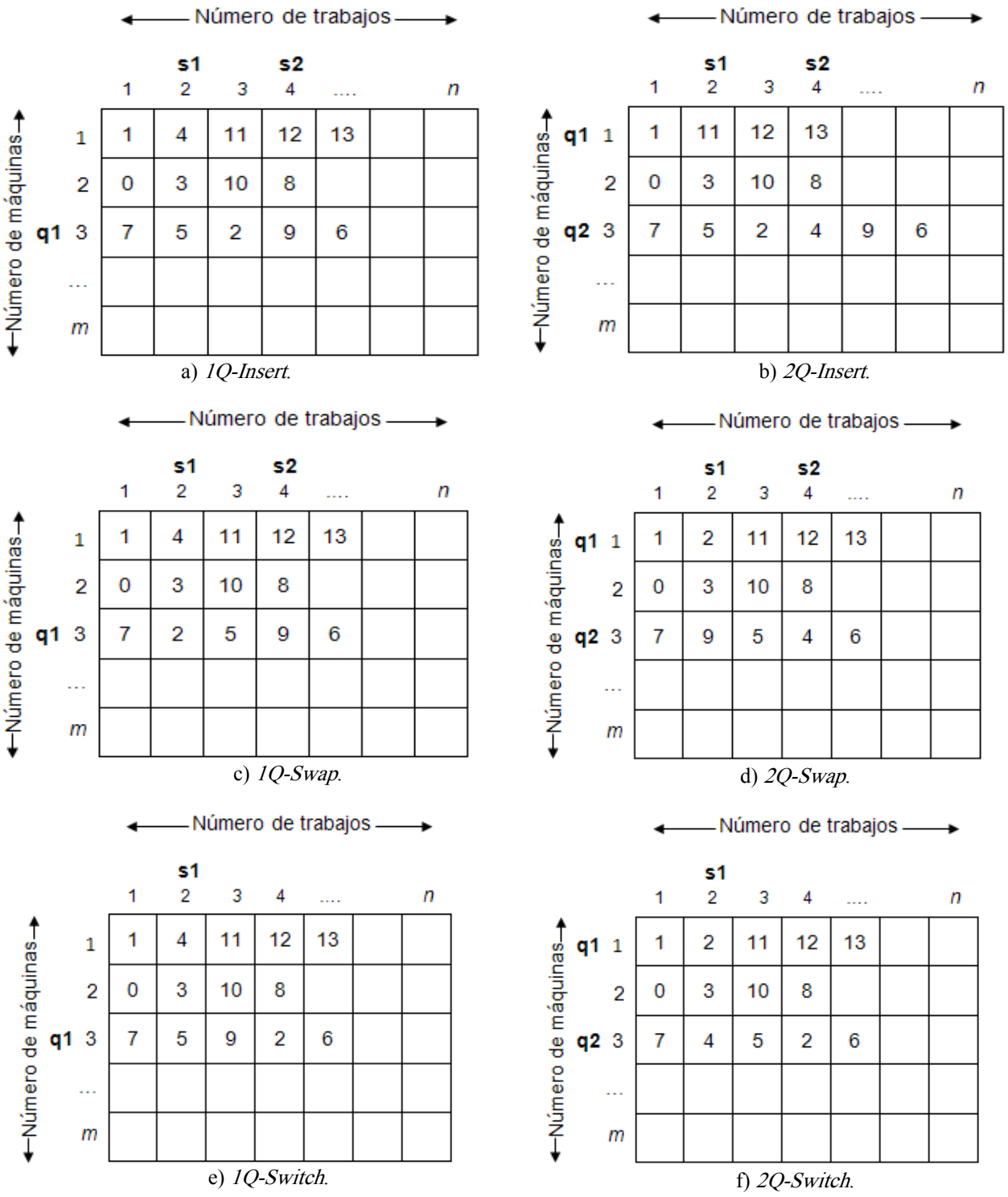

Figura 4. Extensión de los operadores de mutación *Insert*, *Swap* y *Switch* a la representación bidimensional.

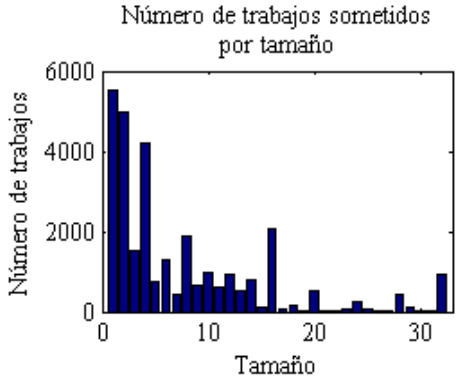

Figura 5. Número de trabajos sometidos por tamaño.

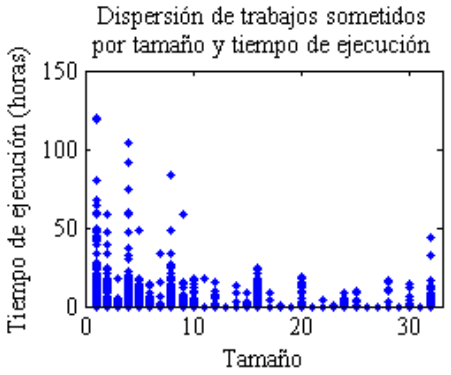

Figura 6. Dispersión de trabajos sometidos por tamaño y tiempo de ejecución.

6.2. Parámetros para la simulación

Se realizaron 30 experimentos, cada uno con lotes de 1,000 trabajos. Los 30,000 trabajos representan aproximadamente una semana de trabajos propuestos al Grid. Como se desconoce los tiempos de ejecución de los trabajos, se toma el tiempo solicitado por el usuario. Configuramos un Grid con sitios de tamaño [4, 4, 4, 4, 8, 8, 8, 16, 16, 32 y 32]. La Tabla I muestra los algoritmos considerados y los criterios en que se enfoca cada uno. Para cada simulación se utiliza la misma carga de trabajo, y el mismo lote de trabajos. Los AGs que consideran un solo objetivo y los que consideran múltiples criterios parten de la misma población inicial. Para evaluar los resultados se realizan simulaciones con otras estrategias (*Min_CT, Min_ST, Min_TA y Min_WT*). Se utiliza un modelo de Grid jerárquico con dos niveles, por lo tanto se requieren estrategias de asignación en el primer nivel. Para la calendarización local en el segundo nivel se aplica *Backfilling-EASY-FirstFit.* Es una mejora de *FCFS* (primero trabajo en llegar, primero en atenderse). Se realizan experimentos con el operador *Q-Insert* en los AGs que consideran un solo objetivo porque en experimentos preliminares (no ilustrados) el resto de los operadores mostraron desempeño menor en promedio al operador *Q-Insert.* Es importante mencionar que la diferencia de desempeño entre el operador *Q-Insert* y el resto de los operadores no fue de gran magnitud, sin embargo, los resultados basados en experimentos preliminares coinciden con la conclusión de [1], en donde se afirma que la combinación del operador de cruzamiento OBX y el operador de mutación *Insert* resulta ser la mejor. A pesar de ello, en los experimentos finales para el caso multi-criterio se toman en cuenta los operadores *Q-Insert*, *Q-Swap* y *Q-Switch* con el fin de notar la diferencia en los resultados bajo la representación de dos dimensiones.

Tabla I. Nombre de los once algoritmos compuestos por las estrategias que se aplican en cada nivel del Grid y criterios que mejoran. En el caso de los AGs, los operadores por los que están compuestos.

Nombre de los algoritmos	Significado				
	Estrategia de asignación. Nivel 1 del Grid	Estrategia de Calendarización. Nivel 2 del Grid	Operador de cruzamiento	Operador de mutación	Criterios considerados
A1	Min_CT	BEFF			Cmax
A2	Min_ST				Slowdown
A3	Min_TA				Tournaround
A4	Min_WT				Wait time
B1	AG		OBX	Q-Insert	Cmax
B2					Slowdown
B3					Tournaround
B4					Wait time
C1	AG-MOAG			Q-Insert	*OWA*
C2				Q-Swap	
C3				Q-Switch	

6.3. Resultados

Todos los algoritmos con enfoque multi-criterio (Figura 7) tienen un comportamiento similar y no existe un marcado dominio por alguno, lo mismo sucede con la desviación estándar (*DS*) correspondientemente.

En la Tabla II se muestra un resumen de los resultados generados por todos los algoritmos. Los resultados de los Algoritmos A1, A2, A3 y A4 se toman como 100% y los resultados de los Algoritmos B1, B2, B3, B4 CI, C2 y C3 se toman como mejora en relación a los primeros.

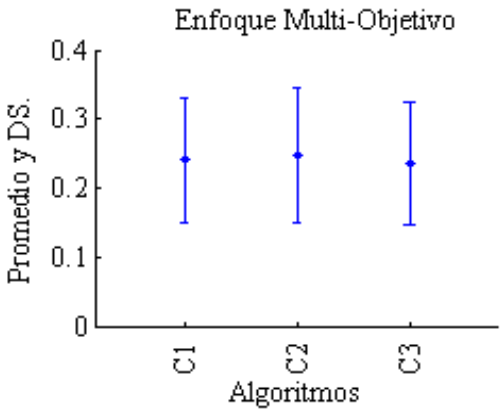

Figura 7. Desempeño de los algoritmos multi-criterio.

Bajo las mismas condiciones todos los algoritmos que consideran criterios múltiples y que usan el método de agregación de criterios (C1, C2 y C3), muestran desempeño similar. Se observa que el mejor desempeño siempre lo obtienen los algoritmos que consideran un solo objetivo (B1, B2, B3 y B4) con respecto a las estrategias de calendarización (A1, A2, A3 y A4). Los algoritmos que aplican el método de agregación de criterios (C1, C2 y C3) también superan el desempeño de las estrategias de calendarización en el criterio en común.

Nótese que minimizar la longitud del calendario resulta "fácil" (Figura 8) para el Algoritmo B1 ya que realiza alrededor de 40 iteraciones en promedio con una desviación estándar relativamente pequeña (comparado con los otros algoritmos). El resto de los algoritmos realizan entre 100 y 150 iteraciones en promedio para minimizar el objetivo; recuérdese que el objetivo de los Algoritmos C1, C2 y C3 implica cuatro criterios que se minimizan simultáneamente.

La Figura 9 muestra los tiempos promedio (en minutos) en que se llevan a cabo las iteraciones de los algoritmos evolutivos. El Algoritmo B1 realiza las iteraciones en un promedio de cinco minutos y en un máximo de diez (solamente optimiza la longitud del calendario). El Algoritmo B2 realiza mayor número de iteraciones y consume mayor tiempo (optimiza el cociente de respuesta promedio), 22 minutos en promedio con una desviación estándar que se extiende por encima de los 33 minutos. Nuevamente, se observa que no hay una diferencia significativa entre los tiempos promedios que necesitan los algoritmos evolutivos que consideran un solo criterio y los que consideran criterios múltiples aunque estos últimos realizan mayor cantidad de operaciones.

Tabla II. Porcentaje (%) de los algoritmos evolutivos con respecto a los algoritmos para calendarizacion.

Métrica	Algoritmo de referencia 100%	Porcentaje de cada criterio con algoritmos evolutivos						
		B1	B2	B3	B4	C1	C2	C3
Longitud del calendario	A1	**18.16**	-29.36	-8.74	-1.10	11.38	12.78	12.02
Factor de competitividad	1.38	1.11	1.74	1.44	1.38	1.23	1.20	1.20
Cociente de respuesta promedio	A2	-7.70	**96.15**	89.68	87.81	93.81	91.86	93.78
Tiempo promedio de permanencia	A3	1.45	51.84	**62.41**	63.27	61.34	59.98	61.60
Tiempo promedio de espera	A4	4.26	58.94	70.40	**71.34**	69.25	67.78	69.53

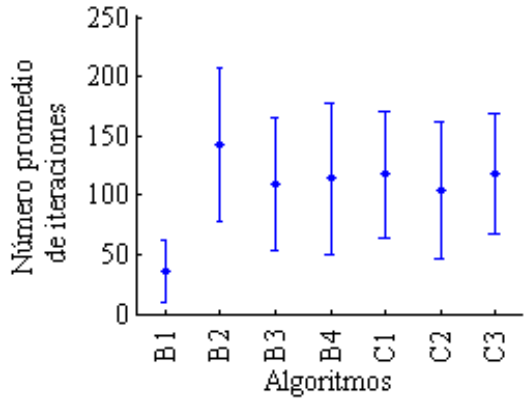

Figura 8. Promedio de iteraciones efectuadas por los algoritmos evolutivos

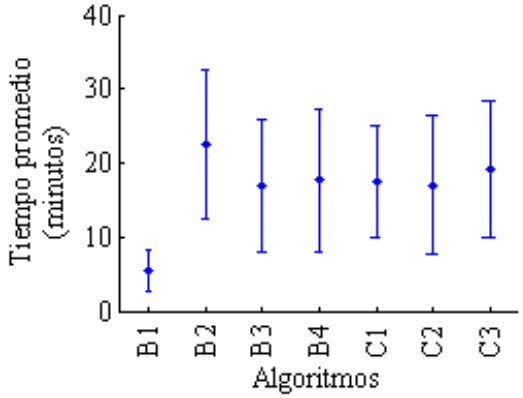

Figura 9. Tiempo promedio consumido por los algoritmos evolutivos durante la simulación.

7. Conclusiones y trabajo futuro

Los AGs superan el desempeño de las estrategias conocidas de calendarización con carga de trabajo real. Los AGs que consideran varios objetivos a la vez y que aplican el método de agregación de los criterios requieren en promedio el mismo tiempo que los AGs que consideran un solo objetivo. Similarmente, los AGs con múltiples criterios que aplican el método de agregación de los criterios efectúan en promedio el mismo número de iteraciones que los AGs que contemplan un solo objetivo. La función usada para generar pesos evita generar pesos al azar, que a la vez reduce esfuerzo innecesario, además de contribuir a la generación de soluciones con calidad. Si se considera una función de utilidad que pondera a todo los criterios relativamente con los mismos pesos, entonces el método de agregación de los criterios obtiene los mejores resultados en promedio. Finalmente, se concluye que es apropiado aplicar AGs al problema de calendarización planteado usando el método de agregación de criterios en Grid computacional de dos niveles. El presente trabajo provee las bases para estudios futuros como: encontrar los operadores y parámetros genéticos que aportan mayor calidad en los resultados. Extender el problema al caso con trabajos en línea.

10. Referencias

[1] Aceves, R. 2003. Estudio de Operadores Genéticos para un Problema de Calendarización Multi-Objetivo. En: Tesis de maestria, CICESE, Ensenada. B. C. Mex. 1-88 p.

[2] The Grid Workload Format. http://gwa.ewi.tudelft.nl/TheGridWorkloadFormat_v001.pdf.

[3] Avellino, G., Barale, S., Beco, S., Cantalupo, B., Colling, D., Giacomini, F., Gianelle, A., Guarise, A., Krenek, A., Kouril, D., Maraschini, A., Matyska, L., Mezzadri, M., Monforte, S., Mulac, M., Pacini, F., Pappalardo, M., Peluso, R., Pospisil, J., Prelz, F., Ronchieri, E., Ruda, M., Salconi, L. Salvet, Z., Sgaravatto, M., Sitera, J., Terracina, A., Vocu, M., Werbrouck, A. 2003. *The EU DataGrid Workload Management System*: towards the second major release. CHEP 2003, La Jolla, California.

[4] Coello, C. A. 2001. *A Short Tutorial on Evolutionary Multiobjective Optimization*. Lecture Notes in Computer Science. 1993(2001), 21-40 p.

[5] Dutot, P.-F., Eyraud, L., Mounie, G., Trystram, D. 2004. *Bi-criteria algorithm for scheduling jobs on cluster platforms*. In Symposium on Parallel Algorithm and Architectures, ″U132, Barcelona, 125 p.

[6] Dutot, P.-F., Eyraud, L., Mounié, G., Trystram, D. 2005. *Scheduling on large scale distributed platforms: from models to implementations*. In: Internat. Journal of Foundations of Computer Science, 16(2), 217-237 p.

[7] Dutot, P.-F., Trystram, D. 2005. *A best-compromise bicriteria scheduling algorithm for parallel tasks*. In: Proceedings of WEA'05 4th International Workshop on Efficient and Experimental Algorithms, Santorini Island, Greece, Poster.

[8] Eiben, A. E. y J. E. Smith. 2003. *Introduction to Evolutionary Computing*. Springer. First Edition. Verlag. 300 pp.

[9] Elmroth, E., Tordsson, J. 2005. *An Interoperable Standards-based Grid Resource Broker and Job Submission Service*, e-Science 2005. First IEEE Conference on e-Science and Grid Computing, IEEE Computer Society Press, USA, 212- 220 p.

[10] *Parallel Workloads Archive*. http://www.cs.huji.ac.il/labs/parallel/workload/.

[11] Foster, I. y C. Kesselman. 1999. *The Grid: Blueprint for a future computing infrastructure.* Morgan Kaufmann. San Fransisco. 748 pp.

[12] Gen, M. y R. Cheng. 1997. *Genetic algorithms & engineering design.* John Wiley & Sons. NY. 342 pp.

[13] Gehring, J. y A. Streit, A. 2000. *Robust Resource Management for Metacomputers,*" In Proc. HPDC '00, pages 105-111.

[14] Graham, R. L., E. L. Lawler, J. K. Lenstra y A. H. G. Rinnooy Kan. 1979. *Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey.* Annals of Discrete Mathematics. 5(1979), 287-326 p.

[15] Hamscher, V., U. Schwiegelshohn, A. Streit y R. Yahyapour. 2000. *Evaluation of Job- Scheduling Strategies for Grid Computing*. R. Buyya and M. Baker (Eds.) In Proc. Grid 2000, LNCS 1971, pp. 191-202

[16] Holland, J. H. 1975. *Adaptation in natural and artificial systems.* University of Michigan Press. Ann Arbor, MI.

[17] Huedo, E., Montero, R.S., Llorente I.M. 2005. *Coordinated Use of Globus Pre-WS and WS Resource Management Services with GridWay*, 2nd Workshop on Grid Computing and its Application to Data Analysis on the Move Federated Conferences, LNCS, vol. 3762, 234-243 pp.

[18] Kurowski, K., J. Nabrzyski, A. Oleksiak y J. Węglarz. 2008. *A multicriteria approach to two-level hierarchy scheduling in grids.* Journal of Scheduling. 11(5), 371 - 379 p.

[19] Kurowski, J., J. Nabrzyski, A. Oleksiak y J. Weglarz. 2006. *Scheduling jobs on the grid--multicriteria approach.* In Computational methods in science and technology. 12(2), 123-138 p.

[20] Lifka, D. A. 1998. *An Extensible Job Scheduling System For Massively Parallel Processor Architectures.*. Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science in the Graduate College of the Illinois Institute of Technology Chicago.

[21] Lorpunmanee, S., M. Noor, A. Hanan y S. Srinoy. 2006. *Genetic algorithm in Grid scheduling with multiple objectives*. Proceedings of the 5th WSEAS int. Conf. on Artificial Intelligence. knowledge Engineering and Data Bases. Madrid, Spain. 429-435 p.

[22] Rodero, I., Corbalt'an, J., Badia, R.M., Labarta, J. 2005. *eNANOS Grid Resource Broker*. In Peter M. A. Sloot et al., editors, Advances in Grid Computing - EGC 2005.

[23] Siddiqui, M., Villazon, A., Fahringer, T. 2005. *Grid Capacity Planning with Negotiation-based Advance Reservation for Optimized QoS*, ACM/IEEE Super Computing (SC06) Tampa, Florida, USA. 11-17 p.

[24] Sinnen, O. 2007. *Task scheduling for parallel systems.* Wiley Series. New Jersey. 296 pp.

[25] Tchernykh, A., J. Ramírez, A. Avetisyan, N. Kuzjurin, D. Grushin y S. Zhuk. 2006. *Two Level Job-Scheduling Strategies for a Computational Grid.* In Parallel Processing and Applied Mathematics. The Second Grid Resource Management Workshop (GRMW'2005) in conjunction with the Sixth International Conference on Parallel Processing and Applied Mathematics - PPAM 2005. Springer-Verlag. Poznan, Poland.

[26] Tchernykh, A., U. Schwiegelshohn, R. Yahyapour y N. Kuzjurin. 2008. *Online hierarchical job scheduling on Grids*. En: Marco Vanneschi Thierry Priol (eds.). From Grids To Service and Pervasive Computing. Springer. Primera edición. Berlin. 77-91 p.

[27] Vadhiyar, S. S. y Dongarra, J. J. 2002. *A Metascheduler for the Grid*. Proc. of 11-th IEEE Symposium on High Performance Distributed Computing (HPDC 2002).

[28] A. Tchernykh, U. Schwiegelsohn, R. Yahyapour, N. Kuzjurin "Online Hierarchical Job Scheduling on Grids with Admissible Allocation", . Journal of Scheduling, Volume 13 , Issue 5 , pp. 545—552. Springer-Verlag, Netherlands, 2010. DOI: 10.1007/s10951-010-0169-x. ISSN: 1094-6136 (Print)