Conciencia Tecnológica No. 49, Enero-Junio 2015

Análisis de los Objetivos Contrapuestos en la Planificación y Asignación de Tareas en un Sistema de Multicomputadoras

Analysis of the Contrast between the Objectives and the Pareto Front in the Planning and Allocation of Tasks in a Multicomputer System

Dr. ApolinarVelarde Martínez, Dr. Enrique Luna Ramírez
Instituto Tecnológico El LlanoAguascalientes, Departamento de Sistemas y Computación
Km. 18 CarreteraAguascalientes - San Luís Potosí, El Llano, Aguascalientes, México.
Tels. (449)916-12-51 Fax. (449)916-20-94. avelardem@gmail.com

Resumen

Los sistemas de cómputo paralelo con múltiples elementos de procesamiento, tienen la habilidad de ejecutar un conjunto de trabajos paralelos a la vez. Para lograr el paralelismo, estos sistemas utilizan un algoritmo planificador y un algoritmo de asignación de tareas. El planificador debe resolver los problemas de cuántos y cuáles trabajos, que permanecen en la cola de espera se deben ejecutar, de cuántos procesadores deben ser los trabajos seleccionados y cuáles trabajos deben ser ejecutados antes que otros; el algoritmo de asignación, debe determinar en qué procesadores libres de la malla, se deben ejecutar las tareas seleccionadas por el planificador. Los objetivos de ambos algoritmos son maximizar el uso de los procesadores en la malla, maximizar la adyacencia de los procesadores que fueron asignados a un trabajo, minimizar los tiempos de espera de los trabajos y minimizar la espera infinita de trabajos en la cola de espera. Al maximizar y minimizar todos los objetivos a la vez, se producen varias contraposiciones entre estos, provocando una degradación en el desempeño del sistema paralelo. En este trabajo, se presenta un análisis de la forma en que se contraponen los objetivos en la planificación y la asignación de las tareas en los sistemas de cómputo paralelo, específicamente en los sistemas de multicomputadoras, utilizando para ello, un algoritmo de optimización multiobjetivo, mediante el cual se evalúa cada objetivo, para determinar el efecto que cada uno de éstos produce en el desempeño de éste tipo de sistemas. Con los resultados obtenidos, se plantea una escala de prioridades de los objetivos evaluados.

Palabras clave: sistema de multicomputadoras, problema multi-objetivo, optimización multi-objetivo, malla 2D, algoritmo de distribución marginal univariante (UMDA).

Abstract

Parallel computing systems with multiple processing elements have the ability to run a set of different tasks at the same time. To achieve such a parallelism, these systems use typically an algorithm for task planning and another algorithm for task assignment. The planner algorithm must solve the problem of how many and which tasks remaining in a queue must be executed, how many processors should be used to execute the selected tasks, and which tasks must be executed first; meanwhile, the assignment algorithm must determine what free processors in a mesh will be used to execute the selected tasks. The objectives of both algorithms are maximizing the use of all processors and the adjacency of the processors assigned to a same task, as well as minimizing the waiting times of the tasks in a queue. Nonetheless, in the purpose of maximizing and minimizing all the objectives at the same time, several conflicts may occur among them, causing degradation in the performance of a parallel computing system. In this paper, it is presented an analysis of how the objectives in the task planning and assignment may conflict, specifically in multicomputer systems. The analysis is carried out by using a multi-objective optimization algorithm, through which each objective is evaluated to determine its effect in the performance of a parallel computing system. With the results of the evaluated objectives, a scale ofpriorities is proposed.

Keywords: multicomputer system, multiobjective problem, 2D mesh, Univariate Marginal Distribution Algorithm (UMDA).

Introducción

Los sistemas de multicomputadoras con arquitecturas en malla, con topologías de interconexión 2D y 3D, denominadas multicomputadoras en malla 2D o 3D, con fines comerciales y de investigación, han sido dos de las redes más comunes debido a su simplicidad,

escalabilidad, regularidad estructural y facilidad de implementación [1,2, 3] en ambientes de investigación o de la industria.

Varias computadoras paralelas comerciales y experimentales han sido construidas en base a estas dos arquitecturas, tales como la IBM BlueGene/L y la Intel Paragon [2].

Algunos de los sistemas de multicomputadoras comerciales son máquinas MIMD (de sus siglas en Inglés Multiple Instruction Multiple Data), con arquitecturas que permiten particiones de submallas de procesadores y que tienen la ventaja de soportar múltiples procesos, cada uno de los cuales puede ser asignado a una submalla de procesadores independiente para su ejecución. En una malla MIMD que soporta múltiples usuarios, una tarea debe ser asignada a una submalla de procesadores libres del tamaño que el sistema operativo requiere [4]. Las tareas solicitarán diferentes requerimientos computacionales y submallas de procesadores con diferentes tamaños dentro de la malla. Cuando una tarea termina su ejecución, la submalla que ocupaba es liberada para una siguiente asignación, proceso conocido como des asignación. El problema de la asignación de tareas en un sistema de multicomputadoras se puede abordar en dos niveles: a nivel de tarea o a nivel de programa [1, 6]. Para este trabajo de investigación, se utiliza la asignación a nivel de tarea.

El problema principal, para la utilización eficiente de los procesadores en los sistemas multiusuario en mallas dinámicas, es la planificación de los recursos computacionales [4, 5, 6]. La planificación de los recursos de la malla, mediante el particionamiento del hardware involucra dos componentes: un planificador de tareas y un asignador de tareas a la malla. La función del planificador de tareas es escoger la siguiente tarea, o las siguientes tareas de la cola de espera, que serán asignadas a una submalla libre para su ejecución, y la función del asignador de submallas es localizar las submallas libres, que se asignan a las tareas seleccionadas por el planificador [4]; en la figura 1 se muestra un conjunto de 6 tareas en la cola que esperan por ingresar a una malla 2D, de procesadores de tamaño 8 X 8, los procesadores ocupados se muestran en círculos rellenos y los procesadores libres con círculos sin relleno.

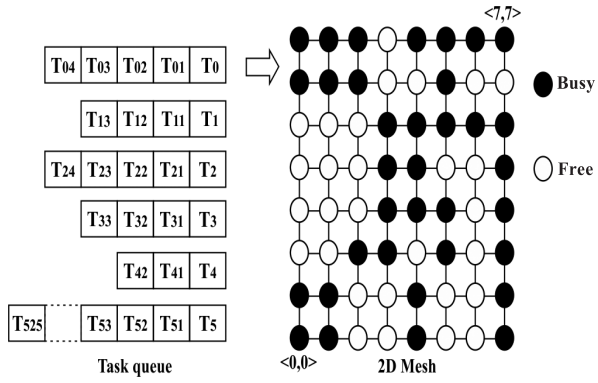

Figura 1. Estructura del sistema para la ejecución de tareas en un sistema de multicomputadoras en malla 2D.

Para la asignación de los trabajos en la malla de procesadores, existen dos métodos: el método de la asignación contigua, en la cual las asignaciones se realizan únicamente en procesadores adyacentes de la malla, y el método de la asignación no contigua, que permite asignar los trabajos en procesadores que no se encuentran adyacentes en la malla.

En la realización de las funciones del planificador y el asignador de tareas, independientemente del tipo de asignación que se realice, se busca maximizar o minimizar uno o varios de los seis criterios u objetivos siguientes [7]: la utilización del sistema, el rendimiento del procesamiento, el tiempo de permanencia promedio, el tiempo de espera, el coeficiente de respuesta y el coeficiente de desempeño. Estos objetivos al ser evaluados con cargas de trabajo en el sistema, generalmente resultan en contraposición, debido a que al mejorar el resultado de uno, se empeora el resultado de otro, provocando que el sistema paralelo sea altamente eficiente en un criterio y tenga resultados de eficiencia muy por debajo de las medias establecidas, en otros [9, 10].

En [11], cinco de estos seis objetivos son evaluados a través de un algoritmo evolutivo de estimación de la distribución (EDA por sus siglas en inglés Evolutionary Distribution Algorithm), específicamente el algoritmo de la distribución marginal univariante (UMDA por sus siglas en inglés Unified Marginal Distribution Algorithm). La evaluación de los objetivos en [11], es con el fin de obtener la asignación que mejor se ajuste a ciertos valores de umbral, que son establecidos como valores óptimos en la planificación y la asignación de las tareas. Pero en el proceso de evaluación individualizado de cada objetivo los resultados se contraponen, produciendo un problema multiobjetivo.

Un ejemplo de lo anterior se presenta cuando, al maximizar la adyacencia entre los procesadores que son asignados a los trabajos, se produce que los tiempos de espera de las tareas y la permanencia de las tareas en la cola se maximicen, produciendo una degradación en los tiempos de respuesta que el sistema paralelo otorga a los usuarios.

En [8], un problema multiobjetivo, se define como aquel que involucra optimizar un número de objetivos simultáneamente. En este tipo de problemas los objetivos están en conflicto uno a otro, esto es, la solución óptima de cada función que se corresponde con cada objetivo (función objetivo) es diferente una de otra. En la solución de tales problemas, con o sin la presencia de restricciones, se da lugar a un conjunto de soluciones óptimas intercambiables, popularmente conocido como soluciones Pareto-óptimas [8]. Debido a la multiplicidad en soluciones, estos problemas fueron propuestos para ser solucionados adecuadamente utilizando algoritmos de optimización evolutiva (EO por sus siglas en inglés, Evolutionary Optimization), los cuales usan un enfoque poblacional en su procedimiento de búsqueda [8]. Los EO usan un enfoque basado en la población en el cual más de una solución participa en una iteración, y evoluciona una nueva población de soluciones en cada iteración [8]. Los problemas de optimización multiobjetivo dan origen a un conjunto de soluciones, las cuales necesitan un procesamiento adicional para llegar a una sola solución principal. Para llevar a cabo la primera tarea, una proposición natural es usar un EO porque el uso de una población en una iteración ayuda a un EO a encontrar simultáneamente múltiples soluciones no dominadas, las cuales representan un intercambio entre objetivos, en una sola corrida de simulación.

Entonces considerando que planificar y asignar trabajos en un sistema paralelo, es un problema multiobjetivo, en este trabajo, se plantean cinco de los objetivos contrapuestos en la planificación y la asignación de tareas en un sistema de multicomputadoras, los cuales se definen a continuación.

Los objetivos que el algoritmo para la planificación de tareas debe cubrir durante su ejecución y propuestos en [11] son:

- ✓ Disminuir el tiempo de permanencia de las tareas en la cola de espera.
- ✓ Disminuir la inanición de tareas, es decir evitar una discriminación en la asignación de las tareas que requieren una gran cantidad de procesadores (tareas grandes), provocada porque las tareas que requieren una poca cantidad de procesadores (tareas pequeñas) estén siendo asignadas continuamente.

Mientras que el algoritmo de asignación de tareas durante su ejecución, es responsable de cubrir los siguientes objetivos propuestos en [11]:

- ✓ Disminuir el número de asignaciones a la malla de procesadores que realiza el algoritmo de asignación de tareas.
- ✓ Maximizar el uso de procesadores de la malla, es decir, disminuir el porcentaje de procesadores que permanecen libres después que el algoritmo de asignación coloca una o más tareas en la malla de procesadores (fragmentación externa) [9].
- ✓ Maximizar la contigüidad entre procesadores (asignar el conjunto de procesadores libres lo más cercanos posible) para minimizar la distancia en la ruta de comunicación y evitar la interferencia entre los mismos [10]; esto para obtener un buen algoritmo paralelo que disminuya el tiempo de comunicación y maximice el tiempo de procesamiento [10].

La selección de los cinco objetivos anteriores obedece a la realización de una revisión del estado del arte, de los trabajos de investigación, relacionados con diferentes técnicas o métodos propuestos para la planificación y la asignación de tareas, que buscan optimizar aisladamente uno o más de los seis criterios u objetivos de la planificación y asignación de tareas.

En este trabajo, abordamos el problema de la planificación y la asignación de trabajos a una malla de procesadores como un problema multiobjetivo, usando para ello el algoritmo evolutivo expuesto en [11] y explicado en la sección: el algoritmo UMDA, (de este trabajo) pero evaluando cada función objetivo para realizar un análisis de los resultados y determinar cuál o cuáles son los objetivos determinantes o con mayor decisión al asignar y planificar las tareas en un sistema de multicomputadoras.

Para la realización del análisis de la contraposición de los objetivos, se utilizó el sistema de multicomputadoras Liebres Inteligentes, un sistema Sistema de Multicomputadoras para la enseñanza de la programación, de los sistemas de cómputo de alto rendimiento en instituciones de educación superior [12]. Los resultados de cada una de las funciones objetivo evaluadas, se obtienen con diferentes cargas de trabajo en la cola de espera del sistema objetivo, y con mallas de procesadores de tamaños de *4X4*, *8X8* y *16X16*.Cuando los valores de las funciones objetivo son similares,seestablece una escala de prioridades para el conjunto de los objetivos propuestos, de tal forma que, se realiza una aplicación sucesiva de dicha escala, hasta llegar a la mejor asignación, como se establece en [11].

Esta investigación está dividida de la siguiente forma, en la sección de trabajos previos, se plantean las investigaciones desarrolladas en el área de la asignación de procesadores, en la sección de conceptos básicos, se definen los términos relacionados con una arquitectura en malla 2D; en la sección denominada el algoritmo UMDA se plantea el funcionamiento y características de este algoritmo, así como también la forma en que se plantea en [11] y como se adapta a este trabajo de investigación este algoritmo; en la sección: Proceso para realizar el análisis de la contraposición de los objetivos, de la planificación y asignación de las tareas en una malla 2D y contraposición de los objetivos, durante la planificación y asignación de tareas, se explica mediante un conjunto de ejemplos, la forma en que los objetivos propuestos se contraponen durante la planificación y asignación de tareas en un sistema de multicomputadoras; en la sección de experimentaciones, se explican las pruebas realizadas para obtener los análisis de los objetivos contrapuestos, y resolver la contraposición que se produce en los objetivos planteados, durante las fases de planificación y asignación de tareas en la malla. En la última sección se presentan las conclusiones obtenidas de los análisis y experimentaciones realizadas.

Trabajos Previos

Varias han sido las investigaciones realizadas para desarrollar estrategias de asignación de tareas en sistemas de cómputo paralelo, tanto contiguas como no contiguas. En ésta sección se describen, por cuestiones de espacio, únicamente, las técnicas no contiguas de asignación, más significativas que se han desarrollado en diferentes trabajos de investigación, con las cuales ha sido posible extraer las características de los métodos, y definir la escala de prioridades de los diferentes objetivos, propuestos en este trabajo, que se buscan maximizar o minimizar en cada una de éstas características.

La técnica del primer ajuste (FF de sus siglas en inglés First Fit) [12], a través de la búsqueda de submalla libres, y la determinación de la submalla que mejor se ajuste a la solicitud, busca encontrar la máxima adyacencia entre procesadores para disminuir la latencia de comunicación entre tareas. En la técnica de paginación [12], el proceso iterativo de división de la submalla, en particiones de igual tamaño 2_i, donde *i* es un entero positivo que representa el parámetro índice de la paginación, busca asignar una tarea en una página seleccionada, lo que permite que el trabajo se ejecute con una adyacencia total de procesadores evitando interferencia de mensajes por procesadores disjuntos. En MBS [12], un proceso de división de la malla para la obtención de submallas cuadradas no sobrelapadas con longitudes de potencia 2, que de forma recursiva se van disminuyendo hasta lograr adecuarse a la solicitud realizada, provoca que el trabajo se incruste en un conjunto de procesadores 100% adyacentes. En ANCA [14], en el primer intento se busca asignar la tarea a una submalla contigua de procesadores, si ésta falla, se particiona la solicitud en subparticiones de igual tamaño en forma recursiva, hasta lograr asignar las subparticiones en las localidades disponibles de la malla. En la estrategia Random [13], se eligen las tareas que se asignan a la malla dependiendo de un número aleatorio y todos los procesadores libres son considerados en la asignación, con este tipo arbitrario de asignación se busca hace uso de la totalidad de procesadores libres, y eliminar cualquier tipo de fragmentación que se pueda producir, pero se produce una alta interferencia de comunicación entre los trabajos.

Técnicas más recientes tienen connotaciones similares a las propuestas inicialmente, a través del uso de una estrategia inicial se busca asignar los trabajos, pero al no poder realizar la asignación, se activa una segunda estrategia que remplaza a la primera para lograr el objetivo de asignación. Ejemplos de dichas técnicas son la estrategia de Búsqueda Adaptativa y Amigable Múltiple (Adaptive Scan and Multiple Buddy AS & MB) [13], Asignación No ContiguaRápida (QNA por sus siglas en ingles Quick Non-ContiguousAllocation) [16] y la estrategia de asignación propuesta en [15]. En AS & MBS, se busca asignar la tarea en una submalla de igual tamaño a la solicitada, en caso de que no exista, la estrategia MBS se activa para realizar el proceso de división del requerimiento [17].

En la estrategia propuesta en [15], el método FF es utilizado en conjunto con el método BF, de la forma siguiente: si una tarea solicita una submalla de tamaño *4X4* y la solicitud no puede ser otorgada, el tamaño de la solicitud es reducida a un múltiplo de 2, para este caso se solicita una malla de tamaño *2X2*, y así sucesivamente hasta que el requerimiento es de la cantidad mínima de procesadores, para este caso *1X1*. Cuando la primera técnica falla, se activa una segunda técnica que es la BF, a través de ésta técnica, se realiza una búsqueda en las submallas libres pero con el mejor ajuste, es decir, con el número exacto de procesadores que la tarea requiere [18]. Con el hecho de aplicar 2 técnicas alternas dentro del método de asignación, se busca mejorar la condición de contigüidad, mediante el mantenimiento de un buen nivel de cercanía entre procesadores, que ejecutan una misma tarea y, reducir la latencia de comunicación que es causada por la no contigüidad entre procesadores.

Conceptos básicos

El sistema propuesto es un sistema de multicomputadoras conectado en una malla 2D, con una cola de tareas que esperan por su ingreso a la malla, y las asignaciones se establecen como una asignación cuadrática dinámica. Las siguientes definiciones describen formalmente a un sistema de éste tipo.

Definición 1. Una malla *n-dimensional* tiene k_0 x k_1 x... x k_{n-2} x k_{n-1} nodos, donde k_i es el número de nodos a lo largo de la i-ésima dimensión y $k_i \geq 2$. Cada nodo se identifica por *n* coordenadas:

$$\rho_0(a), \rho_1(a),..., \rho_{n-2}(a), \rho_{n-1}(a),$$

Donde:

$$0 \leq \rho_i(a) < k_i \text{ para } 0 \leq i < n.$$

Dos nodos *a* y *b* son vecinos si y solo si $\rho_i(a) = \rho_i(b)$ para todas las dimensiones excepto para una dimensión *j*, donde $\rho_j(b) = \rho_j(a) \pm 1$. Cada nodo en una malla se refiere a un procesador y dos vecinos están conectados por un enlace de comunicación directo.

Definición 2. Una malla *2D*, la cual es referenciada como *M (W, L)* consiste de *W X L* procesadores, donde *W* es el ancho de la malla y *L* es la altura de la malla. Cada procesador se denota por un par de coordenadas *(x, y)*, donde:

$$0 \leq x < W \text{ y } 0 \leq y < L$$

Un procesador está conectado por un enlace de comunicación bidireccional a cada uno de sus vecinos. Para cada malla $2Da=P_{ij}$.

Definición 3. En una malla *2D*, *M(W, L)*, una sub-malla: *S(w, l)* es una malla de dos dimensiones que pertenece a *M(W, L)* con un ancho *w* y una altura *l*, donde $0 < w \leq W$ y $0 < l \leq L$. *S (w, l)* están representadas por las coordenadas *(x, y, x', y')*, donde *(x, y)* es la esquina inferior izquierda de la sub-malla y *(x', y')* es la esquina superior derecha. El nodo de la esquina inferior izquierda es llamado el nodo base de la sub-malla y la esquina superior derecha es el nodo final. En este caso $w=x'-x+1$ y $l=y'-y+1$. El tamaño de *S(w, l)* es: *w xl* procesadores.

Definición 4. En una malla *2D M (W, L)*, una sub-malla disponible *S (w, l)* es una sub-malla que satisface las condiciones: $w \geq \alpha$ y $w \geq \beta$ asumiendo que la asignación de S (α, β) requerida, donde la asignación se refiere a seleccionar un conjunto de procesadores para una tarea de llegada.

Definición 5. Sea ϑ un conjunto de tareas del sistema, tal que $\vartheta = J_1, J_2, ..., J_n$ donde *n* es el número de tareas en el tiempo *t* y ϑ_k un conjunto de sub-tareas de la tarea *k* donde: $\vartheta_k = j_{k1}, j_{k2}, ..., j_{kf(k)}$ y *f(k)* es el número total de sub-tareas de la tarea *j*. Para cada tarea *j* y cada sub-tarea $f(k) \in j$ se tiene un procesador $m_i \in P$ en el que se debe ejecutar cada tarea *j* y cada sub-tarea $j_{kf(k)}$, consumiendo un tiempo $t \in N$ ininterrumpido.

Definición 6. Dadas dos matrices de tamaño *n x n*: una matriz de flujo *F* cuyos *(i,j)*ésimos elementos representan los flujos entre tareas *i* y *j* y un arreglo de distancias *D* cuyos *(i,j)*esimos elementos representan la distancia entre sitios *i* y *j*. Una asignación se representa por el vector *p*, el cual es una permutación de los números 1, 2,…, n. *p(j)* es el lugar donde la tarea *j* es asignada. Así, la asignación cuadrática de tareas puede ser escrita como:

$$\min_p \varepsilon \sum_{i=1}^{n} \sum_{i=1}^{n} f_{ij} dp(\mathrm{i}) p(\mathrm{j}) \qquad (1)$$

Definición 7. Un problema de optimización, es aquel cuya solución implica encontrar en un conjunto de soluciones candidatas alternativas, aquella que mejor satisface unos objetivos. Formalmente, el problema se compone del espacio de soluciones *S* y la función objetivo *f*. Resolver el problema de optimización *(S, f)* consiste en determinar una solución óptima, es decir, una solución factible $x^* \in S$ tal que $f(x^*) \leq f(x)$, para cualquier $x \in S$. Las soluciones alternativas se pueden expresar por la asignación de valores a algún conjunto finito de variables X= {X_i: i=1,2,…, n}. Si por U_i se denota al dominio o universo (conjunto de valores posibles) de cada una de éstas *n* variables, el problema consiste en seleccionar el valor x_i asignado a cada variable X_i del dominio U_i que, sometido a ciertas restricciones, optimiza una función objetivo *f*. El universo de soluciones se identifica con el conjunto:

$$U=\{x=(x_i: i=1,2,...,n) : x_i \in U_i\}$$

Las restricciones del problema, reducen el universo de soluciones a un subconjunto de soluciones $S \subseteq U$, denominado espacio factible.

La evaluación del desempeño de un sistema paralelo se evalúa con los siguientes criterios [1]:

Definición 8. Utilización (utilization). Se define como la fracción del tiempo en la cual el sistema fue utilizado. Y está dado por:

$$U_G = W_G / (C_G * m_G) \qquad (2)$$

Donde:

W_G es la cantidad de trabajo del Sistema

C_G es el tiempo de finalización de ejecución de todas las tareas en el Sistema

m_G es el número total de procesadores en el sistema

Definición 9. Rendimiento del procesamiento (throughput). Es el número de tareas finalizadas por unidad de tiempo en el sistema, y está dado por:

$$n / C_G \qquad (3)$$

Donde:

n es el número total de trabajos en el sistema.

Definición 10. Tiempo de permanencia promedio (mean turnaround time). Es el promedio del tiempo que tardan todas las tareas desde que entran a la cola local hasta que terminan su ejecución. Se calcula como:

$$\frac{1}{n}\sum_{j=1}^{n} t_t^j \tag{4}$$

Donde:

$$t_t^j = c^j - r^j$$

c^j es el tiempo de finalización de la tarea
r^j es el tiempo de entrega de la tarea j

Definición 11. Tiempo de espera (waiting time). Está definido como el tiempo promedio de espera de las tareas antes de iniciar su ejecución. Se calcula como:

$$\frac{1}{n}\sum_{j=1}^{n} t_w^j \tag{5}$$

Donde:

$$t_w^j = t_s^j - r^j$$

t_s^j es el tiempo de inicio de ejecución de la tarea *j*

Definición 11. Coeficiente de respuesta (response ratio). Está definido como el promedio de los coeficientes de respuesta de todas las tareas. Se define como:

$$\frac{1}{n}\sum_{j=1}^{n} (t_w^j + p^j)/p^j \tag{6}$$

Donde:

p^jes el tiempo de ejecución

t_w^j es el tiempo de espera de la tarea *j*

Definición 12. Coeficiente de desempeño (competitive ratio). Es la medida de rendimiento del Sistema definida como:

$$p = c_g / c_{LB} \tag{7}$$

Dónde:
C_G es el tiempo de la finalización
c_{LB} es el tiempo mínimo posible para terminar las tareas, calculado como:

$$\max w_G / m_g, p_G^{\max} \tag{8}$$

Donde:

$p_G^{\max}$ Es el máximo tiempo de ejecución de las*n* tareas.

El algoritmo UMDA

Los algoritmos de estimación de distribuciones (EDA, por sus siglas en inglés Estimation of Distribution Algorithms), son algoritmos evolutivos que usan una colección de soluciones candidatas para realizar trayectorias de búsqueda evitando mínimos locales [19, 20]. Estos algoritmos usan la estimación y simulación de la distribución de probabilidad conjunta como un mecanismo de evolución, en lugar de manipular directamente a los individuos que representan soluciones del problema. Un algoritmo EDA comienza generando aleatoriamente una población de individuos que representan soluciones del problema. Se realizan iterativamente tres tipos de operaciones sobre la población. El primer tipo de operación, consiste en la generación de un subconjunto de los mejores individuos de la población. En segundo lugar, se realiza un proceso de aprendizaje de un modelo de distribución de probabilidad a partir de los individuos seleccionados. En tercer lugar, se generan nuevos individuos simulando el modelo de distribución obtenido. El algoritmo se detiene cuando se alcanza un cierto número de generaciones o cuando el rendimiento de la población deja de mejorar significativamente.
Para estimar en cada generación la distribución de probabilidad conjunta, a partir de los individuos seleccionados, usamos el algoritmo de la distribución marginal univariante (UMDA por sus sigla en inglés, Univariate Marginal DistributionAlgorithm). Así, la distribución de probabilidad conjunta se factoriza como producto de distribuciones univariantes independientes [19, 20], es decir:

$$p_l(x) = p(x \mid D_{l-1}^{Se}) = \prod_{i=1}^{n} p_l(x_i) \tag{12}$$

Cada distribución de probabilidad univariante se estima a partir de las frecuencias marginales:

$$p_l(x_i) = \frac{\sum_{j=1}^{n} \delta_j(X_i = x_i \mid D_{l-1}^{Se})}{N} \tag{13}$$

Donde:

$$\delta_j(X_i = x_i \mid D_{l-1}^{Se}) = \begin{cases} 0 \text{ si en el } j\text{-esimo } D_{l-1}^{Se}, X_i = x_i \\ 1 \text{ en otro caso} \end{cases}$$

El seudocódigo para el UMDA, propuesto en [19, 20] es el siguiente:

UMDA

$D_0 \leftarrow$ Generar M individuos (la población inicial) al azar
Repetir para l=1,2,.. hasta el criterio de parada
$D_{l-1}^{Se} \leftarrow$ *Seleccionar N≤M Individuos de* D_{l-1} *acorde al método de selección*

$$p_l(x) = p(x \mid D_{l-1}^{Se}) = \prod_{i=1}^{n} p_l(x_i) = \prod_{i=1}^{n} \frac{\sum_{j=1}^{n} \delta_j(X_i = x_i \mid D_{l-1}^{Se}}{N} \leftarrow$$

Estimar la distribución de probabilidad conjunta

D_l *Muestrear M individuos (la nueva población) de* $p_l(x)$

En este trabajo, la aplicación del algoritmo evolutivo UMDA, se realiza de la siguiente forma:

1) Se extrae de manera dinámica un conjunto de tareas de la cola de espera que caben en las submallas libres, este conjunto de tareas representa una posible asignación (individuo); este proceso se repite hasta alcanzar un número *n* de individuos (definidos por el usuario), que constituyen una población.
2) Por cada individuo de la población, son evaluados los cinco objetivos, para determinar el subconjunto de las asignaciones (subconjunto de los mejores individuos) que muestren los resultados más cercanos a la maximización o minimización establecida por cada función objetivo.
3) El proceso de aprendizaje del modelo de distribución de probabilidad se produce a partir de los individuos seleccionados, que representan las mejores asignaciones a la malla de procesadores.
4) Una generación de nuevos individuos se produce mediante la simulación del modelo de distribución, obtenido en el paso anterior.
5) Un mecanismo de paro del algoritmo se activa al alcanzar la minimización o maximización, de las funciones objetivo.

Durante el proceso de evaluación de cada objetivo para cada uno de los individuos, se presentan las contraposiciones en los resultados, debido que al mejorarse el resultado de una función objetivo se empeora el resultado de otra. La siguiente sección, explica mediante ejemplos la forma en los objetivos se contraponen.

Proceso para realizar el análisis de la contraposición de los objetivos de la planificación y asignación de las tareas en una malla 2D

Para realizar el análisis de la contraposición de los 5 objetivos, extraídos de los trabajos de investigación y previamente expuestos en las secciones anteriores, en primer lugar, se consideraron los resultados obtenidos en [10, 11] de las corridas del algoritmo UMDA, que evalúa de forma separada cada objetivo. En segundo lugar, se llevaron a cabo corridas adicionales del mismo algoritmo para determinar la contraposición de cada uno de los objetivos con el resto de los mismos.

El planteamiento formal de las contraposiciones encontradas, se detalla en la siguiente sección. Una vez finalizadas las experimentaciones, y en base a los resultados se propone una escala de prioridades, la cual se utiliza en este trabajo de investigación como determinante para encontrar la mejor asignación de las tareas, a la malla de procesadores.

Contraposición de los objetivos durante la planificación y asignación de tareas

En ésta sección, se explica mediante un conjunto de ejemplos las contraposiciones encontradas entre los objetivos, que se persiguen cumplir para la óptima utilización de los procesadores de la malla. También se realiza un planteamiento formal de las mismas.

Los objetivos 1 y 2, que buscan minimizar el número de asignaciones a la malla de procesadores, para minimizar el tiempo que las tareas permanecen en la cola de espera, se contrapone con los objetivos de la minimización del uso de los procesadores en la malla y la minimización de la inanición de las tareas. Para ejemplificar la forma en que se contraponen estos cuatro objetivos, considere que en un tiempo *t*, el asignador informa de 29 procesadores libres (como lo muestra la figura1), con este dato el planificador determina que, el conjunto de las 5 tareas T_0, T_1, T_2, T_3 y T_4 son candidatas para ocupar 21 procesadores en la malla, o asignar la tarea T_5 que requiere 26 procesadores y la tarea T_4 que solicita 3. Asignar el conjunto de 5 tareas libera el mismo número de posiciones en la cola para el ingreso de nuevas tareas, disminuyendo así el número de accesos a la cola para buscar tareas, permite que un mayor número de usuarios y trabajos sean atendidos y, se disminuye el tiempo de espera de las tareas en la cabeza de la cola; pero de manera contrapuesta, se genera una fragmentación externa de 8 procesadores y se incrementa la inanición de tareas en un ciclo, al no ser atendida una tarea que requiere un gran número de procesadores.

Ahora, si se asigna las tareas T_4 y T_5 no se produce inanición ni fragmentación externa, pero un número menor de tareas pueden ser aceptadas en la cola, por lo que se incrementa el número de asignaciones a la malla y por tanto, el tiempo que las tareas deben esperar para ingresar a la malla de procesadores.

El objetivo 3, que busca maximizar el uso de los procesadores en la malla, se contraponen con el objetivo 5, que maximiza la adyacencia de los procesadores ocupados en la malla 2D. Para ejemplificar la contraposición entre estos objetivos, se considera el ejemplo anterior. Al realizar la búsqueda del costo más bajo de comunicación, el conjunto de las 5 tareas seleccionadas: T_0, T_1, T_2, T_3 y T_4, se asignan en procesadores contiguos, de la siguiente forma: la tarea T_0 se asigna en la submalla <4,0><5,2> sin considerar el procesador en la posición <4,2>, la tarea T_1 se asigna en la submalla <2,0><3,1>, la tarea T_2 se asigna en la submalla <0,5><2,6> sin considerar el procesador en la posición <2,5>, la tarea T_3 se asigna en la submalla <0,2><1,3>, y la tarea T_4 se asigna en

la submalla<6,3><7,4> sin considerar el procesador en la posición<7,4>. Si el método propuesto detecta el incremento de inanición en el sistema, la tarea T_5 será asignada a la malla junto con la tarea T_1 o la tarea T_3 que se selecciona para ocupar la totalidad de los procesadores, después de una búsqueda en la cola y evitar la fragmentación externa; de ésta forma los 29 procesadores libres de la malla permanecerán ocupados. Si se contraponen los objetivos 3 y 5 podemos deducir que al asignar la tarea T_1 o T_3 y la tarea T_5 se maximiza el uso de procesadores en la malla, pero se minimiza la adyacencia entre los procesadores, y en contraposición, si se asigna el conjunto de las 5 tareas, se maximiza la adyacencia entre procesadores, pero se produce una fragmentación externa de 8 procesadores.

El objetivo 1 que minimiza las asignaciones a la malla de procesadores, se contrapone con el objetivo 5, que maximiza la adyacencia entre procesadores. La contraposición entre estos dos objetivos, aparece cuando se pretende asignar un gran número de tareas en la malla de procesadores, y los procesadores a los que se asignan las tareas no se encuentran lo suficientemente juntos o contiguos, para evitar producir costos de comunicación muy altos.

Si consideramos asignar el conjunto de las 5 tareas, T_0, T_1, T_2, T_3 y T_4, se minimiza el número de asignaciones realizadas a la malla, pero si las posiciones de los procesadores libres en la malla, no se encuentran adyacentes, se producirá que las tareas se asignen de una forma muy disjunta en la malla, provocando que la adyacencia de procesadores sea mínima y los costos de comunicación entre tareas muy altos.

El objetivo 2 que busca minimizar el tiempo de espera en la cola de espera de las tareas, se contrapone con el objetivo de maximiza la adyacencia entre procesadores.

Utilizando el mismo ejemplo de la sección anterior y considerando el objetivo 2 que establece minimizar el tiempo de espera de las tareas en la cola, al asignar el mayor número de tareas en la malla permite minimizar a un conjunto de tareas su tiempo de espera, es decir, en la asignación que ocurre en un tiempo *t*, un menor número de tareas esperarán en la cola. Para este tipo de casos si dos objetivos son afectados al mismo tiempo por un tercer objetivo, y logramos mejorarlos, el algoritmo decidirá por ésta opción al realizar la planificación y la asignación, aun considerando el costo que representa la fragmentación externa generada y el incremento en la inanición de tareas.

El objetivo 4, que busca minimizar la inanición de tareas, se contrapone con el objetivo 5, que maximiza la adyacencia entre procesadores. Decidirse por asignar un mayor número de tareas para minimizar los tiempos de espera y maximizar el uso de procesadores, parece una opción viable, pero si consideramos un tercer objetivo en conflicto que establece disminuir la inanición de tareas en el sistema, podemos encontrar entonces que estamos a favor de 2 objetivos (2 y 3), y sacrificamos también 2 (los objetivos 4 y 5).

En base a los ejemplos explicados, hemos encontrado que mantener un control estricto de tareas que quepan en la malla, para cumplir los objetivos propuestos, produce búsquedas exhaustivas en la cola, y cálculos para ubicar en la mejor posición las tareas en la malla [10]. Más que buscar las mejores posiciones delas tareas en la malla, se debe realizar un análisis de los objetivos, que se buscan optimizar, porque al tratar de localizar submallas del tamaño que las tareas requieren con el único objetivo de estar contiguos, sin considerar una evaluación de otros objetivos, puede conducir a resultados pobres en los tiempos de respuesta y en el desempeño del sistema.

En base a las explicaciones anteriores, en este trabajo se plantea una escala de prioridades de los objetivos propuestos, a ser considerados durante la planificación y la asignación de tareas en los sistemas multicomputadoras. Cabe puntualizar que ésta escala de prioridades, se considera en el algoritmo propuesto en este trabajo, y con el cual se obtuvieron los resultados que se explican en la sección de resultados obtenidos.

Escala de prioridades de los objetivos propuestos

En este trabajo se propone una estratificación de los objetivos, en base a los resultados obtenidos con el algoritmo UMDA y las observaciones realizadas en las corridas de la experimentación lo anterior se realiza con el fin de determinar las mejores asignaciones, en caso de que se hayan encontrado valores muy semejantes o iguales en el rankeo de los objetivos.

La estratificación propuesta es la siguiente:

1. El objetivo 5, que establece maximizar la adyacencia entre procesadores, se considera el objetivo de mayor importancia en la asignación, por cinco situaciones que se presentan durante las asignaciones de las tareas a la malla de procesadores, en los experimentos realizados: el primer factor a considerar es el tiempo de comunicación que las tareas consumen durante su ejecución, debido a que los procesadores no adyacentes generan costos de comunicación muy altos, y más aún cuando se deben ejecutar tareas que requieren, grandes cantidades de procesadores dentro de la malla. Aunque la búsqueda de submallas libres es un proceso tedioso y consume tiempo del procesador, es una tarea que debe ser extensiva en todo momento dentro.

2. El objetivo 3, maximizar el uso de procesadores para disminuir la fragmentación externa, se considera en segundo lugar, por su importancia en la asignación de los procesadores en la malla. Su importancia radica en que funge como soporte para el objetivo 1, las experimentaciones realizadas nos permiten observar que, para maximizar el uso de procesadores las asignaciones se deben realizar en el mayor número de procesadores que se encuentren adyacentes dentro de la malla de procesadores. Para cumplir este objetivo, el algoritmo de que realiza la búsqueda de submallas libres debe ser lo suficientemente robusto.
3. El objetivo 4, minimizar la inanición de tareas, se ha ubicado en tercer grado de importancia, debido a que al cumplir con los dos objetivos anteriores nos permite un manejo seguro de las tareas que requieren grandes cantidades de procesadores en la malla, evitando la inanición de tareas. Si el algoritmo de búsqueda de submallas libres, proporciona conjuntos de procesadores libres adyacentes, es posible evitar la inanición de tareas al ubicarlas con mayor facilidad dentro de la malla.
4. El objetivo 1, minimizar el número de asignaciones a la malla de procesadores, es decir minimizar el número de planificaciones que el algoritmo debe realizar con las tareas que permanecen en la malla, se considera como el objetivo en cuarto lugar de importancia para la evaluación.
5. El objetivo 2, es considerado con una importancia en nivel 5, que establece disminuir el tiempo de espera de las tareas en la cola.
6. El objetivo 1, se considera que incluye al objetivo 2, porque el lograr minimizar el número de asignaciones a la malla de procesadores, permite disminuir el tiempo que las tareas esperan en la cola.

Resultados

Las experimentaciones se realizaron con diferentes cargas de trabajo en el sistema Liebres Inteligentes [12] y con diferentes tamaños en la cola de espera. Se consideraron cargas de tamaño 256, 512, 1024 y 2048 tareas en el sistema. Los largos de la cola de espera se realizaron en 10, 20, 30 y 50 tareas con sus respectivas subtareas. El númerode subtareas para cada tarea es de 255 como máximo, considerando que el tamaño de la malla de procesadores es <16X16>. En la figura 2, se muestran los resultados obtenidos para cada función objetivo.

Por cuestiones de espacio, se muestran únicamente las cargas en el sistema y los valores obtenidos para las funciones objetivo, de hasta 800 tareas.

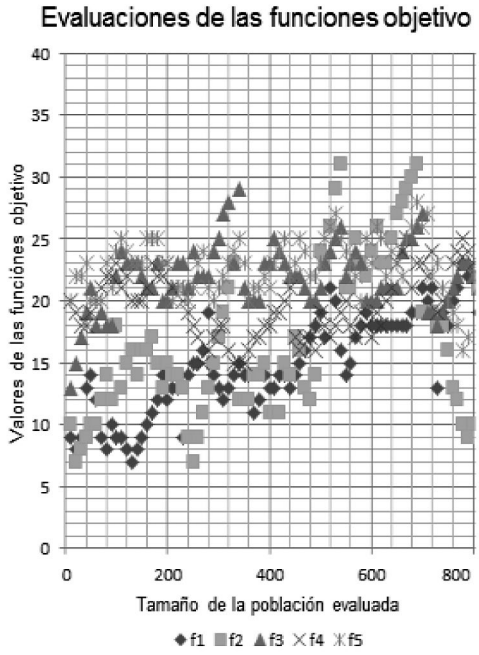

Figura 2. Experimentación 1, la gráfica muestra los valores obtenidos para las funciones objetivo.

En la coordenada X, se muestran las diferentes cargas del sistema y en la coordenada Y, los valores de las funciones objetivo obtenidas. La representación de las funciones, que se muestra en la parte baja de la gráfica, identifican el símbolo que se utiliza para cada función, el ordenamiento de los funciones está en forma ascendente, no en el grado de importancia de cada una de ellas.

En los párrafos siguientes se explican los resultados obtenidos para cada función objetivo y el impacto que tiene en la planificación o la asignación de las tareas.

Los valores para la función 5, maximizar la adyacencia entre procesadores que busca asignar las tareas con el mayor grado de contigüidad, muestra que con valores menores a 100 tareas en las cargas de trabajo, no tiene valores aceptables, pero conforme la cantidad de tareas a ejecutarse se incrementa, los valores de la función mejoran significativamente, es de observarse en esa parte que, el número de subtareas por tarea que se procesaron, excede la media, es decir excede las128 subtareas por tarea.

La función 4, minimizar la inanición de tareas, tiene una tendencia alta, debido a que las tareas que se procesan contienen un alto número de subtareas, provocando que las tareas con menor número de subtareas sean atendidas más rápidamente por el sistema. Conforme se desalojan tareas grandes, la inanición tiende a estabilizarse en valores aceptables

provocando, una mayor fluidez en el procesamiento de las tareas.

La función 3, maximizar el uso de procesadores, considerada una de las funciones de mayor importancia para el sistema, tiene una tendencia a agrupar sus valores, con los resultados obtenidos en la función 5, pero conforme el número de tareas a procesarse se incrementa, existe una dispersión en un punto medio cuyas tendencias muestran que al obtener mayor adyacencia, el uso de procesadores disminuye, sobre todo cuando el sistema inicia a procesar tareas con un gran número de subtareas.

La función 2, disminuir el tiempo que una tarea espera para ser atendida en la cola, muestra una tendencia muy clara, cuando se procesan tareas con pocos requerimientos de recursos, los tiempos de espera son muy cortos, debido a la planificación que realiza el algoritmo. En caso contrario, cuando las tareas que se procesan contienen un gran número de requerimientos, los tiempos de espera son altos, que sin duda alguna, al incrementar el número de recursos ésta tendencia se mejora con facilidad. Lo anterior se muestra en la gráfica cuando se procesan las cargas entre 200 y 400 tareas, los valores de las funciones se disparan con mucha facilidad.

Finalmente, la función 1, que busca minimizar el número de asignaciones que el algoritmo realiza, muestra tendencias muy pobres al arranque del sistema, pero conforme la ejecución avanza, se mejoran significativamente sus valores. Su tendencia con los valores de la función 2, la hace ser una función dependiente que toma una curva hacia los valores que adquiere la función 2. Conforme la función 2 presenta mejores valores que representan la disminución del tiempoque una tarea espera por ser atendida en la cola, se minimiza significativamente el número de asignaciones que el algoritmo asignador realiza. La función 2 en conjunto con la función 1, tienen un alto grado de importancia en los resultados que el sistema arroja, por lo que es de suma importancia considerarlas como prioritarias.

La figura 3, muestra otra ejemplificación de ejecución del sistema, con un mayor número de tareas en el sistema, las tendencias de los valores son similaresa la gráfica anterior. Con este ejemplo, se busca que las funciones tomen valores correspondientes a una mayor carga en el sistema. El análisis que se realiza en ésta experimentación, nos permite observar que la funcionalidad del algoritmo de planificación, es viable en cargas pesadas del sistema.

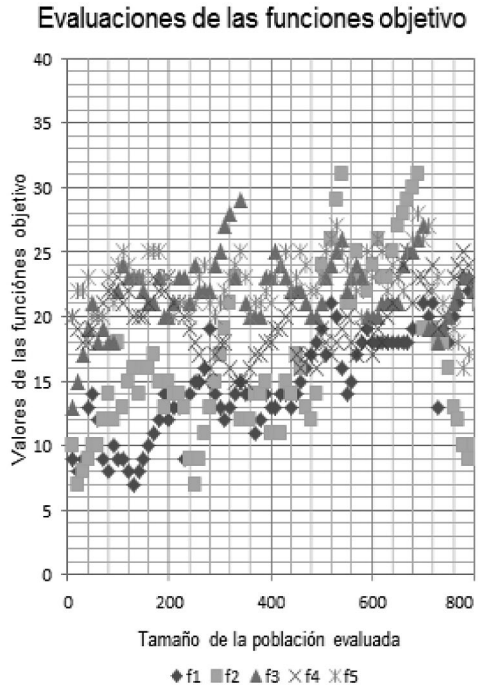

Figura 3. Experimentación 2, la gráfica muestra los valores obtenidos para las funciones objetivo.

Conclusiones

Los sistemas de Multicomputadoras son una opción viable para el procesamiento paralelo, debido a su crecimiento en cuanto a potencia computacional y almacenamiento distribuido. Los problemas inherentes que conlleva su arquitectura son la planificación y la asignación de tareas. Para la asignación de tareas a la malla de procesadores, se han propuesto un gran número de técnicas basadas en diferentes estrategias a través de: figuras geométricas que se desplazan a lo largo y ancho de la malla para localizar las submallas libres, ajustes de los tamaños de las solicitudes de submallas libres y, técnicas basadas en asignaciones aleatorias. La mayoría de dichas técnicas hacen uso de políticas de planificación, basadas en el primero en llegar es el primero en ser atendido (FIFO de sus siglas en ingles First Input, First Output), es decir, no utilizan una planificación previa en la cola de espera, además de que únicamente se busca resolver un solo problema: el de la asignación adyacente o contigua, para lograr disminuir el paso de mensajes entre tareas y subtareas.

El método presentado en este documento, retoma el problema de la planificación y la asignación de tareas en un sistema de multicomputadoras, como un problema multiobjetivo, llevando a cabo un análisis de la forma en que cada objetivo impacta en el desempeño del sistema, utilizando para ello un algoritmo evolutivo.

El objetivo de este análisis es, mostrar los valores que las funciones presentan cuando los objetivos se contraponen, tanto en la planificación como en la asignación de los procesadores de la malla.

Con los resultados obtenidos, utilizando el sistema de multicomputadoras Liebres Inteligentes, es posible deducir que para evaluar una técnica de asignación de procesadores en una malla, es necesario que dicha técnica evalúe al menos 5 objetivos distintos, porque de ésta manera se puede determinar que, no solamente soluciona un problema, sino que busca un conjunto de valores que permitan equilibrar una solución. Un planteamiento que busca solucionar un solo objetivo, no es viable, por ejemplo, solucionar la adyacencia de la tareas y permitir que el tiempo de respuesta del sistema no sea considerado en el planteamiento de la solución.

Finalmente, es de suma importancia mencionar el trabajo que debe realizar el algoritmo de búsqueda de submallas libres, dentro de la malla de procesadores, debido a que es quien soporta, vigila y cumple los objetivos más importantes dentro de la estratificación planteada, en este trabajo de investigación.

Referencias

[1] Grama A., Gupta A., Karypis G., Kumar V. (2003), *Introduction to Parallel Computing*. Second Edition. Addison Wesley. ISBN: 0-201-64865-2.

[2] Bani S., Ababneh I., and Ould M. A. (2009), Performance Comparison of the Non-Contiguous Allocation Strategies in 2D Mesh Connected Multicomputers. *International Conference On Communication, Computer And Power (ICCCP'09) MUSCAT*, February 15-18, 2009.

[3] Ababneh I., Bani-Mohammad S. , (2011), A new window-based job scheduling scheme for 2D mesh Multicomputers. *Simulation Modeling Practice and Theory* 19, Pp. 482 493.

[4] Ahmad S. E., (2011), Processor Allocation with Reduced Internal and External Fragmentation in 2D Mesh-based Multicomputers. *Journal of Applied Science* 11 (6) 943-952, ISSN 1812-5654 2011 Asian Network for Scientific Information.

[5] Das D. and Pradhan D. K. Job Scheduling in Mesh Multicomputers. *IEEE Transactions On Parallel And Distributed Systems*, Vol. 9, No. 1, JANUARY

[6] Foster I. (1995), *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison Wesley.

[7] Heiss H. U., (1994), Dynamic Partitioning of Large Multicomputer Systems. *Proc. Int. Conf. on Massively Parallel Computing Systems* (IEEE MPCS94), Ischia, May 2-6.

[8] Deb K. (2001), *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley

[9] Velarde A., Ponce de León E., Díaz E., and Padilla A. (2010), Planning and Allocation of processors in 2D meshes. *Doctoral Consortium*. Mexican Internacional Conference on Artificial IntelligenceMICAI 2010, Pachuca Hidalgo, México.

[10] Velarde A., Ponce de León E., Díaz E., Padilla, (2011), A. Dynamic quadratic Assignment to Model Task Assignment Problem to Processors in 2D Mesh. *Advances in Soft Computing Algorithms*, Editors: I. Batyrshin, G. Sidorov. Research in Computing Science Vol. 54, pp. 199-218, Puebla, México. RCS.

[11] Velarde A., Ponce de León E., Diaz E. Planning and Allocation Tasks in a Multicomputer System as a Multi-objective Problem. *Advances in Intelligent Systems and Computing 227*. EVOLVE 2013. International Conference held at Leiden University, July 10-13, 2013. Leiden, The Netherlands. Springer.

[12] Velarde A., (2015). Liebres InTELigentes: Sistema de Multicomputadoras para la enseñanza de la programación de los sistemas de cómputo de alto rendimiento en Instituciones de Educación Superior. Articulo aceptado en: *Congreso Internacional de Investigación en Ciencias y Sustentabilidad de Academia Journals*. Tuxpan, Ver. México. Marzo de 2015.

[13] Lo V., Windisch K., Liu W., and Nitzberg B., (1997), Non-contiguous processor allocation algorithms for mesh-connected multicomputers, *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 7, pp. 712-726.

[14] Chang C.Y. and Mohapatra P. (1998), Performance improvement of allocation schemes for mesh-connected computers, *Journal of Parallel and Distributed Computing,* vol. 52, no. 1, pp. 40-68...

[15] Bani A. S., (2013), Submesh Allocation in 2DMesh multicomputers: Partitioning at the Longest Dimension of Request. *The International Arab Journal of Information*

Technology, Vol. 10, No.3, May 2013. Pp. 245 252.

[16] Procsimity V4.3 *User's Manual*, University Oregon, 1997

[17] Zolfaghari R. (2013), Efficient Algorithm for Processor Allocation in Mesh Multicomputers Network with Limitations and Assumptions. *IJCEM International Journal Of Computational Engineering & Management*, Vol. 16 Issue 4, July 2013. ISSN (Online) 2230-7893. Pp. 513.

[18] Suzaki K., Tanuma H., Hirano S., Ichisugi Y., Connelly C., and Tsukamoto M., (1996), Multitasking Method on Parallel Computers which Combines a Contiguous and Non-contiguous Processor Partitioning Algorithm. *Proceedings of the 3rd International Workshop on Applied Parallel* Computing, Industrial Computation and Optimization, Lecture Notes in Computer Science, Springer, London, pp. 641- 650.

[19] P. Larrañaga, J. A. Lozano y H. Mühlenbein, (2003), Algoritmos de estimación de distribuciones en problemas de optimización combinatoria. Inteligencia Artificial. *Revista Iberoamericana de Inteligencia Artificia*l.

[20] J. A. Lozano y P. Larrañaga. Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Kluwer Academic.

Artículo recibido: 18 de marzo de 2014

Aceptado para publicación: 25 de marzo de 2015