Ingeniare. Revista Chilena de Ingeniería
ISSN: 0718-3291
facing@uta.cl
Universidad de Tarapacá
Chile

Velarde Martínez, Apolinar; Luna Ramírez, Enrique; Soria Cruz, Jorge; Ponce de León Sentí, Eunice

Evaluación de objetivos al planificar y asignar trabajos paralelos en un sistema de multicomputadoras

Ingeniare. Revista Chilena de Ingeniería, vol. 24, núm. 2, abril, 2016, pp. 290-304
Universidad de Tarapacá
Arica, Chile

Evaluación de objetivos al planificar y asignar trabajos paralelos en un sistema de multicomputadoras

Objective evaluation to plan and allocate parallel tasks in multicomputer system

Apolinar Velarde Martínez[1] Enrique Luna Ramírez[1] Jorge Soria Cruz[1]
Eunice Ponce de León Sentí[2]

Recibido 25 de junio de 2014, aceptado 3 de noviembre de 2015
Received: June 25, 2014 Accepted: November 3, 2015

RESUMEN

En este trabajo se propone un método que realiza una evaluación de cada uno de los objetivos que se contraponen en la planificación y la asignación de los trabajos a ser procesados en un sistema de multicomputadoras en malla 2D, como los tiempos de espera, de procesamiento, de transferencia e inanición de tareas. El método propuesto utiliza un algoritmo de selección dinámica de tareas de la cola de espera para determinar cuáles tareas son susceptibles de ser asignadas en las submallas libres de procesadores del sistema objetivo, una vez que el algoritmo de asignación detecta los procesadores libres en la malla. Para seleccionar los procesadores en donde las tareas serán asignadas y evaluar los objetivos contrapuestos, el método utiliza el algoritmo de la estimación de la distribución, para determinar cuáles son los procesadores que ofrecen las mejores soluciones de asignación en cuanto a los objetivos evaluados, así como también la incidencia que cada uno de ellos tiene en el desempeño del sistema. Los resultados obtenidos en las experimentaciones con el método propuesto muestran una mejoría en los tiempos de procesamiento y de espera de las tareas, en comparación con los resultados que producen las técnicas de asignaciones de tareas: lineal y las curvas de Hilbert.

Palabras clave: Sistema de multicomputadoras, problema multiobjetivo, malla 2D, algoritmo de distribución marginal univariante, algoritmo de estimación de la distribución.

ABSTRACT

In this paper, it is proposed a method to carry out an evaluation of the objectives that may conflict during the task planning and assignment task in a two-mesh multicomputer system, which includes the evaluation of the waiting, processing and transfer times of tasks. In the first instance, the proposed method uses two algorithms: an assignment algorithm for detecting the free processors in a mesh and a dynamic algorithm for selecting the tasks in a waiting queue that is susceptible to be assigned to the free objective system processors. To select the processors a set of tasks will be assigned, this method uses additionally a distribution estimation algorithm for determining which processors offer the best assignment solution respect to the impact of the evaluated objectives in the system performance. The results obtained with the proposed method showed an improvement in the waiting task and processing times compared to the results produced by task assignment techniques such as the lineal technique and the Hilbert curves.

Keywords: Multicomputer system, multiobjective problem, 2D mesh, univariate marginal distribution algorithm (UMDA), estimation distribution algorithm (EDA).

[1] Departamento de Sistemas y Computación. Instituto Tecnológico el Llano Aguascalientes. Carretera a San Luis Potosí Kilómetro 18. El Llano, Aguascalientes. C.P. 20330. Aguascalientes, México.
E-mail: apolive@yahoo.com; elunaram@hotmail.com; jorsor777@yahoo.com.mx
[2] Departamento de Ciencias de la Computación. Universidad Autónoma de Aguascalientes. Av. Universidad 940 Ciudad Universitaria. Aguascalientes, México. E-mail: eponce@correo.uaa.mx

INTRODUCCIÓN

De las diferentes arquitecturas paralelas desarrolladas, los sistemas de multicomputadoras con arquitecturas en malla, con topologías de interconexión 2D y 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 [1-3].

A diferencia de la programación para sistemas secuenciales, los sistemas paralelos deben dividir los problemas (tarea) en subproblemas (subtareas) para permitir la distribución de la carga computacional entre los procesadores libres (recursos computacionales disponibles) [3]. No es fácil descomponer todos los problemas grandes en subproblemas, debido a la dependencia de los datos entre los subproblemas. Debido a esta dependencia los procesadores deben estar comunicados entre sí cuando ejecutan una tarea en forma paralela. Es importante resaltar que el tiempo consumido por la comunicación entre dos procesadores es muy alto comparado con el tiempo de procesamiento. Debido a esto, el esquema de comunicación debe ser planeado cuidadosamente para obtener un algoritmo paralelo eficiente [2, 4].

Una vez realizada la división de un trabajo paralelo, la tarea con sus subtareas son puestas en la cola de espera, donde un algoritmo de planificación determina el orden de ejecución de las mismas, después un algoritmo de asignación de tareas es el responsable de colocar dichas tareas en la malla de procesadores, donde se colocan hasta que terminan su ejecución [1, 5-7].

En este trabajo se plantea un metaalgoritmo que permite conjuntar los algoritmos de planificación y de asignación de tareas, y se plantean y evalúan un conjunto de cinco objetivos descritos a continuación, que aglutinan los problemas dispersos que se quieren solucionar en los trabajos propuestos con anterioridad, como [2-6, 8]. Por cuestiones de espacio no se menciona la totalidad de trabajos investigados en este campo.

Los objetivos que el algoritmo para la planificación de tareas debe cubrir durante su ejecución y propuestos en [11-12] 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 [11-12]:

- 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].

Este trabajo está organizado de la siguiente manera, en la sección: la planificación y asignación de tareas en un sistema de multicomputadoras se describe la forma en que se realiza la planificación de las tareas en la cola de espera de los sistemas paralelos, y el proceso de la asignación de tareas a la malla de procesadores. El estado del arte de las técnicas más importantes, que realizan la planificación y la asignación de tareas, se describen en la sección de los trabajos previos. Un conjunto de definiciones, que permiten el mejor entendimiento del sistema expuesto en este trabajo, se plantean en la sección de definiciones previas de los sistemas de multicomputadoras. La descripción del funcionamiento del sistema expuesto se lleva a cabo en la sección del método propuesto. Para describir los resultados de las experimentaciones realizadas se creó la sección de resultados obtenidos, y finalmente se presentan las conclusiones de este trabajo en una sección dedicada para ello.

PLANIFICACIÓN Y ASIGNACIÓN DE TAREAS EN UN SISTEMA DE MULTICOMPUTADORAS

La planificación y la asignación de tareas en una malla de procesadores se constituye por una lista de tareas que permanecen en la cola, en espera de que le sean asignados un conjunto de procesadores, para iniciar su ejecución. El algoritmo de planificación de tareas es el responsable de especificar el orden en el cual la tarea, o las tareas, ingresarán a la malla de procesadores una vez que el algoritmo de asignación le informe de las submallas libres que existen en un tiempo *t* dentro de la malla de procesadores [6].

Una vez que una tarea o varias tareas se seleccionan por el algoritmo de planificación, estas se asignan dentro de la malla para iniciar su ejecución, sin interrupción hasta que finalizan, y se retiran de la malla de procesadores. A continuación, la cola de tareas se contrae y la subsecuente tarea permanece en la cabeza de la cola, para esperar su ingreso a la malla de procesadores.

Por ejemplo, suponga una cola de espera de 6 tareas, en la que cada tarea consiste de un conjunto de subtareas que van de 2 a 5 tareas, y una malla de 8 x 8 procesadores desde el procesador <1,1> hasta el procesador <8,8>. En un tiempo *t*, el proceso de planificación determinará cuántas y cuáles tareas ingresarán a la malla, dependiendo del número de procesadores libres; para este caso suponga que la tarea 1, en la cabeza de la cola, ha sido seleccionada para ser asignada a la malla, después de su ingreso la cola de espera se contraerá, la tarea 2 pasa a ser la cabeza de la cola, y todas las tareas que permanezcan en la malla esperarán una siguiente planificación y asignación para su ejecución. La tarea 6 pasará a la posición 5 de la cola y un espacio estará disponible para el ingreso de una tarea nueva, que podrá competir por su ingreso a la malla en la siguiente planificación.

Así como únicamente la tarea 1 fue planificada para su ejecución, se pueden planificar más tareas que ingresarán a la malla, dependiendo del número de procesadores desocupados que existan en la misma, y del tipo de planificación que los algoritmos utilicen para colocar las tareas.

Para llevar a cabo la asignación de las tareas a la malla de procesadores se han desarrollado dos estrategias de asignación de procesadores [6, 14]; su clasificación obedece al tipo de reconocimiento que se realiza en la malla de procesadores: las estrategias de asignación de procesadores contiguas y las estrategias de asignación de procesadores no contiguas.

Las estrategias de asignación de procesadores contiguas aparece cuando se tiene un reconocimiento parcial del sistema, y es posible asignar para la ejecución del trabajo únicamente submallas contiguas de procesadores a los trabajos que lo solicitan. La Figura 1 muestra una estrategia de asignación contigua cuando una tarea solicita una submalla libre de 4 procesadores, en una malla de tamaño 4 x 4. La identificación de los procesadores es desde el procesador <1,1> hasta el procesador <4,4>. Debido a la contigüidad de los procesadores libres en la submalla <1,1> hasta el procesador <2,2>, el algoritmo puede realizar la asignación de la tarea sin problema alguno.

Las estrategias de asignación de procesadores no contiguas surgen para eliminar las deficiencias presentadas por las asignaciones contiguas, e implican tener un algoritmo de reconocimiento total de la malla, de tal forma que si un trabajo de tamaño *n x n* no puede ser asignado a una submalla *m x m* libre, ocupe diferentes submallas libres aun cuando estas no se encuentren contiguas. La Figura 2 muestra una asignación de procesadores no contigua en la malla. En este ejemplo una tarea solicita una submalla de tamaño 2 x 2, de esta forma, ya que los procesadores <1,3> <2,3> <3,2> y <4,2> están disponibles en la

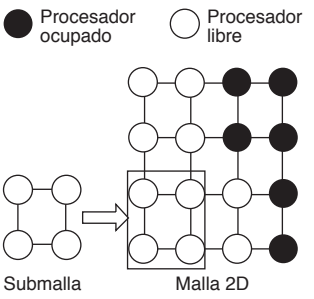

Figura 1. Estrategia de asignación de procesadores contigua, cuando una tarea solicita una submalla libre de 4 procesadores en una malla de tamaño 4x4.

malla, y la estrategia de asignación permite la no contigüidad de procesadores para que una tarea se ejecute, la asignación es posible.

Para mejorar el desempeño de un sistema de cómputo paralelo, el objetivo es buscar una estrategia que permita a los procesadores estar contiguos y, en caso de no encontrar submallas libres que satisfagan lo anterior, entonces se debe buscar que los procesadores estén lo más cercanos posibles, para evitar la sobrecarga de la comunicación en la malla; buscar submallas libres con pocos procesadores desagregados en la malla puede conducir a encontrar tareas con requerimientos de pocos procesadores en la cola de espera durante todas las planificaciones que se realizan, esto puede provocar que tareas que requieran gran cantidad de procesadores, sean relegadas hasta que existan submallas libres con un gran número de procesadores, debido a esto la estrategia debe evitar que tareas grandes caigan en inanición dentro del sistema.

Para ejemplificar lo anterior se proponen los siguientes casos descritos en [12], suponga una cola de espera con 6 tareas que esperan por ingresar, y las submallas libres desagregadas en la malla; aquí suponemos que las primeras 5 tareas T_0 a T_5 solicitan un reducido número de procesadores: 5, 4, 5, 6 y 3, respectivamente, y la tarea T_{14} solicita 29 procesadores; una vez que la planificación se realiza, las primeras 5 tareas podrán ser objeto de asignación en la malla, pero debido al número de recursos de la tarea T_{14} que solicita 29 deberá esperar si las condiciones del algoritmo de planificación no establece una estrategia para atender tareas con grandes solicitudes de recursos, y evitar que se produzca la inanición de tareas. De no ser asignada la tarea, deberá esperar a la siguiente planificación para competir por su asignación, y bajo la condición de que no vuelvan a existir tareas pequeñas que compitan de manera directa con ella, porque entonces dicha tarea deberá esperar indefinidamente.

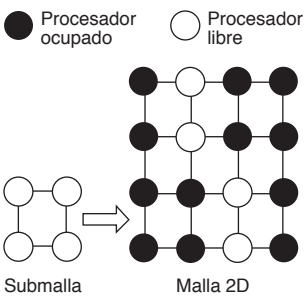

Figura 2. Estrategia de asignación de procesadores no contigua, cuando en una malla de procesadores de tamaño 4 x 4.

La política de asignación de procesadores FIFO (*First Input, First Output*, por sus siglas en inglés) es una política libre de inanición, y permite que todas las tareas sean asignadas de acuerdo con su tiempo de llegada a la cola de espera, pero esta política carece de planificación y su problema radica en que pueden existir procesadores libres dentro de la malla, pero no pueden ser asignados debido a que el número requerido de procesadores no se satisface con el número de procesadores libres dentro de la malla. Por ejemplo, suponga que ahora la tarea 14 permanece en la cabeza de la cola de espera y solicita 30 procesadores, las 5 tareas que han llegado no podrán ser asignadas debido a que el orden de asignación no lo permite y el número de procesadores libres que solicita la tarea, que permanece en la cabeza de la cola, no puede ser completado; de esta forma 29 procesadores estarán ociosos en la malla, y 29 procesadores serán requeridos por 5 tareas que esperan por entrar al mismo tiempo, y que serán atendidas hasta que la tarea 14 sea asignada a la malla.

Lo anterior establece varios objetivos que se contraponen, al diseñar una estrategia de planificación de tareas en una malla: por un lado, el sistema podrá tener recursos disponibles que no puede utilizar, y por el otro, tendrá tareas que corren el riesgo de esperar indefinidamente en la cola si no son planificadas adecuadamente.

En contraposición a las políticas de asignación, libres de inanición, los algoritmos que planifican tareas permiten que estas sean asignadas de una manera más rápida, debido a que en cada asignación podrán ser colocadas más de una tarea a la vez, lo que produce que no exista una relación del número de tareas con el número de asignaciones a la malla, por ejemplo, debido a la cola de espera de las tareas de la figura 3, y considerando que ha existido planificación de tareas y el método de asignación es no contiguo, entonces en una sola asignación podrán ingresar de manera directa las primeras 5 tareas, y la tarea T_5 podrá ir a la cabeza de la cola, liberando de esta

forma 5 espacios para que futuras tareas ingresen a la malla. Para elegir las tareas que ingresarán a la malla es necesario un proceso de elección basado en diferentes criterios: ubicación de la tarea en relación al número de procesadores que requiere (contigüidad), y asignar el mayor número de tareas en el mayor número de nodos libres. Un método de planificación y asignación de tareas para un sistema paralelo, que modele lo anterior, puede disminuir el número de asignaciones que se realicen en la malla de procesadores, disminuir el tiempo de permanencia de las tareas en la cola de espera, y maximizar el uso de los procesadores de la malla. De esta forma, minimizar y maximizar los objetivos en la asignación de tareas en una malla 2D se visualiza como un problema multiobjetivo.

Una forma de abordar un problema multiobjetivo es por medio de los algoritmos metaheuísticos que establecen estrategias para recorrer el espacio de soluciones de un problema, transformando de forma iterativa soluciones de partida [16]. En este trabajo se presenta una propuesta en la que, una vez que se visualizan un conjunto de mallas libres, y un conjunto de tareas posibles de asignar, se corre una metaheurística de búsqueda para elegir iterativamente la mejor solución que permita llegar a una optimización en la asignación, es decir, encontrar la asignación más barata en cuanto a la comunicación, la contigüidad y el número de procesadores que vayan a ser asignados. Las asignaciones necesarias que se deben realizar se consideran como asignaciones cuadráticas dinámicas en la malla, debido a que las combinaciones posibles se basan en el número de procesadores libres y el número de tareas que esperan por entrar.

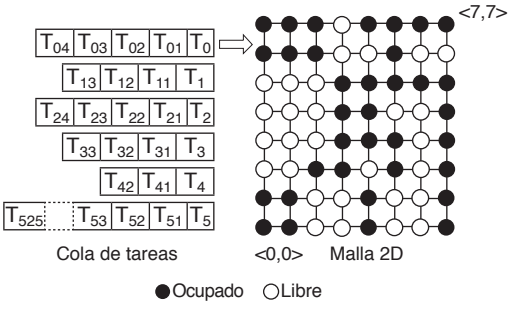

Figura 3. Asignación de tareas a una malla de procesadores, utilizando una estrategia de asignación no contigua.

Ahora, para contrarrestar los riesgos que se corren al utilizar metaheurísticas de búsqueda en el problema multiobjetivo se ha implementado una búsqueda parcial sistemática para evitar repeticiones, manteniendo un alto grado de aleatoriedad. Esta búsqueda se implementa mediante un recorrido exhaustivo, y haciendo uso del método de paro, el que establece el criterio de parada que determina un número máximo de generaciones o, en su defecto, un tiempo máximo de resolución [20-21] sin necesidad de llegar a completar todo el espacio de soluciones para evitar caer en óptimos locales o la pérdida de la diversidad de soluciones generadas.

TRABAJOS PREVIOS

Varias han sido las investigaciones realizadas para desarrollar estrategias de asignación, tanto contiguas como no contiguas, en este trabajo revisamos las técnicas de asignación no contiguas en mallas 2D, más representativas.

La técnica Primer Ajuste (FF por sus siglas en inglés *First Fit*) [8], mediante la búsqueda de submallas 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 [8] se realiza un proceso iterativo de división de la submalla, en particiones de igual tamaño 2i, donde i es un entero positivo que representa el parámetro índice de la paginación; de esta forma se 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 [8], 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 [5], en el primer intento se busca asignar la tarea a una submalla contigua de procesadores, si esta falla, se realiza la partición de 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 [8] 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 hacer 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, por medio 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) [8], Asignación No Contigua Rápida (QNA por sus siglas en ingles *Quick Non-Contiguous Allocation*) [15] y la estrategia de asignación propuesta en [4]. 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. En la estrategia propuesta en [4], 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 2 x 2, y así sucesivamente hasta que el requerimiento es de la cantidad mínima de procesadores, para este caso 1 x 1. Cuando la primera técnica falla, se activa una segunda técnica, la BF, mediante esta 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. Con el hecho de aplicar dos 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.

El método de asignación lineal [4-5, 8] permite realizar la ubicación de las tareas de la parte inferior izquierda a la parte derecha de la malla; el reconocimiento se basa en una ruta lineal de la malla, lo que permite una fácil y rápida colocación de las tareas en las submallas libres, pero presentan dificultades cuando se inicia la liberación de tareas, debido al gran número de submalllas libres producidas por las tareas que son removidas de la malla y carece de un proceso de compactación. El método de las curvas de Hilbert [17-19] establece un llenado de espacios en forma de curvas, que visita cada punto en una malla cuadrada de tamaño 2 x 2, 4 x 4, 8 x 8, 16 x 16, o en cualquier orden de potencia 2; las aplicaciones de este método han sido en procesamiento digital de imágenes, especialmente en la compresión y dilatación de imágenes [19]. Tanto el método de las curvas de Hilbert como la asignación lineal no pueden asignar submallas libres en un orden diferente al método lineal establecido, lo que produce una alta segmentación de las tareas, e incrementa la transferencia de mensajes en la malla de procesadores, lo que nos permite tener un punto de comparación con el método descrito en este trabajo.

DEFINICIONES PREVIAS

El sistema propuesto es un sistema de multicomputadoras conectados 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 un sistema de este tipo.

Definición 1. Una malla *n-dimensional* tiene k0 x k1 x... x kn-2 x kn-1 nodos, donde ki es el número de nodos a lo largo de la i-ésima dimensión y ki ≥ 2. Cada nodo se identifica por n coordenadas, ρ0(a), ρ1(a),..., ρn-2(a), ρn-1(a), donde $0 \leq \rho i(a) < ki$ para $0 \leq i < n$. Dos nodos a y b son vecinos si y solo si ρi(a) = ρi (b) para todas las dimensiones excepto para una dimensión j, donde ρj(b) = ρj(a) ± 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 que 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:

$0 \leq x < W$ y $0 \leq y < L$.

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

Definición 3. En una malla 2D, M (W, L), una submalla 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 submalla y(x', y') es la esquina superior derecha. El nodo de la esquina inferior izquierda es llamado el nodo base de la submalla 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 x l procesadores.

Definición 4. En una malla 2D M (W, L), una submalla disponible S (w, l) es una submalla 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 $\vartheta = J_1,\ldots, J_n$ donde *n* es el número de tareas en el tiempo t y ϑk un conjunto de subtareas de la tarea k donde: $\vartheta_k = j_{k1}; j_{k2},\ldots, j_{kf(k)}$ y f(k) es el número de subtareas de la tarea k. Para cada subtarea $i \in s$ se tiene una tarea k tal que $Si \in J_k$ y un procesador $mi \in P$ en el que se debe ejecutar, consumiendo un tiempo $t \in \mathbb{N}$ ininterrumpido.

Definición 6. Debido a 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 (k, l) ésimos elementos representan la distancia entre los sitios k y l. Una asignación se representa por el vector p, el que 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:

$$\min_p \in \sum_{i=1}^{n}\sum_{i=1}^{n} f_{ij}dp(k)p(l) \qquad (1)$$

Donde:
i es la i-ésima tarea de la matriz de flujos
j es la j-ésima tarea de la matriz de flujos
k, l es la distancia entre el k-ésimo y l-ésimo sitio

Definición 7. Un problema de optimización es aquel cuya solución implica encontrar en un conjunto de soluciones candidatas, aquellas alternativas 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 $X = \{Xi: i=1,2,\ldots, n\}$. Si por Ui se denota al dominio o universo (conjunto de valores posibles) de cada una de estas n variables, el problema consiste en seleccionar el valor xi 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 $U = \{x = (x_i: i=1,2,\ldots,n): x_i \in Ui\}$. Las restricciones del problema reducen el universo de soluciones a un subconjunto de soluciones $S \subseteq U$, denominado espacio factible.

Definición 8. Minimizar los tiempos de espera de las tareas en la cola de espera está dado por:

$$t_{sj} = \sum_{i=1}^{s_j} t_{ij} \qquad (2)$$

Cuya función objetivo se establece como:

$$\min TQ = \sum_{i=1}^{s_j} t_{i1} + \sum_{i=1}^{s_j} t_{i2} + \ldots + \sum_{i=1}^{s_j} t_{ij} \qquad (3)$$

Sujeto a:
$Ti \in \mathbb{R}$ y $1 \leq j < |J|$

Donde:
i es la i-ésima tarea en la cola de espera
j es la j-ésima tarea en la cola de espera
t el tiempo total que la tarea tarda en ejecutarse dentro del sistema de multicomputadoras

UMDA para la Asignación Cuadrática Dinámica para modelar el Problema de Asignación de Tareas a Procesadores en la Malla 2D

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 [20-21]. 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 distribución marginal univariante (UMDA por sus sigla en inglés, *Univariate Marginal Distribution Algorithm*). Así, la distribución de probabilidad conjunta se factoriza como un producto de distribuciones univariantes independientes, es decir:

$$p_l(x) = p(x \mid D_{l-1}^{S_e} = \prod_{i=1}^{n} p_l(x_i) \quad (4)$$

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

$$p_l(x_i) = \frac{\sum_{j=1}^{N} \delta_j(X_i = x_i \mid D_{l-1}^{S_e})}{N} \quad (5)$$

Donde:

$$\delta_j(X_i = x_i \mid D_{l-1}^{S_e} = \begin{cases} 1 \, si \, en \, el \, j-ésimo \, caso \, de D_{l-1}^{S_e}, X_i = x_i \\ 0 \, en \, otro \, caso \end{cases}$$

El seudocódigo para el algoritmo UMDA es el siguiente:

UMDA

D0 ← Generar M individuos (la población inicial) al azar

Repetir para *l* = 1,2,… hasta el criterio de parada:

$D_{l=1}^{S_e} \leftarrow Seleccionar \, N \leq M \, individuos$

de

$D_{i=1}$ acorde con un método de selección

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

$$p_l(x_i) =$$

$$\frac{\sum_{j=1}^{N} \delta_j(X_i = x_i \mid D_{l-1}^{S_e})}{N}$$

Estimar la distribución de probabilidad conjunta.

Dl ← Muestrear M individuos (la nueva población a partir de $p_l(x)$).

DESCRIPCIÓN DEL MÉTODO PROPUESTO

El método propuesto está constituido por 6 subprocesos y expuesto en [11-13]. De forma resumida, en esta sección se expone cada uno de los subprocesos. Se parte del supuesto que se tiene un conocimiento *a priori*, de la medida del grado de comunicación entre la tarea principal y las subtareas que la constituyen, de todas las tareas que se encuentran en la cola de espera, así como de la relación entre las mismas subtareas. Para ejemplificar la relación entre la tarea principal y sus subtareas considere que se tiene una tarea T_1, con tres subtareas T_{11}, T_{12} y T_{13} la interacción que se puede dar entre ellas se ejemplifica en la Figura 4, por medio de líneas que muestran la transferencia de mensajes, de esta forma la tarea T_1 puede enviar y recibir mensajes de sus subtareas, y a su vez las subtareas podrán hacer lo mismo con la tarea principal y entre ellas mismas.

Se tiene también una matriz de distancias simétrica entre procesadores, que especifica los saltos que un mensaje debe realizar entre un procesador y otro. Debido a lo anterior, el proceso de asignar los procesadores se basa en llevar a cabo un cálculo de asignación de procesadores, a base de la disponibilidad de los mismos en la malla, y de las tareas que permanecen en la cola de espera. Esta asignación cuadrática permite determinar, mediante

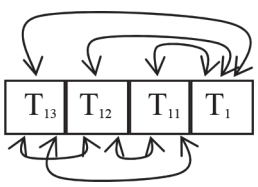

Figura 4. Paso de mensajes entre tareas, para una tarea con tres subtareas.

los cálculos y de la frecuencia de aparición de cada tarea en cada procesador cuál es la tarea o tareas que producen la mejor asignación que representa la solución más factible.

Mediante un ejemplo se describe el funcionamiento del método. En un tiempo *t* se tiene una malla de procesadores de tamaño 4x4, cuya matriz de estado se muestra en la Figura 5, donde 1 representa los procesadores ocupados que fueron asignados a una tarea en un tiempo *t-1,* y un 0 representa los procesadores libres que no han sido asignados a alguna tarea o subtarea, las distancias simétricas entre procesadores están dadas en la Tabla 1, por cuestiones de espacio se presentan únicamente la mitad de la tabla. La Tabla 2 representa la cola de espera conteniendo 4 tareas pendientes de ejecución, dichas tareas esperan por ejecutarse en la malla, y dos matrices de costos de comunicación representadas en la Tabla 3 y la Tabla 4 de las tareas T_1, T_2, T_3 y T_4 con sus subtareas y entre las subtareas mismas.

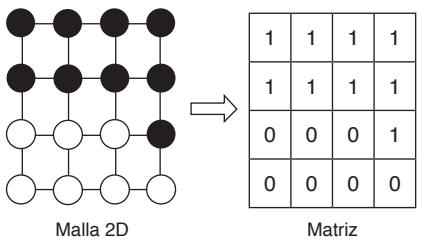

Figura 5. Malla de procesadores de tamaño 4x4, representada por una matriz.

Tabla 1. Distancia simétrica entre procesadores de la malla de tamaño 4x4 de la Figura 5.

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

Tabla 2. Cola de espera de las tareas en tiempo *t.*

T_1	T_{11}	T_{12}	T_{13}	T_{14}
T_2	T_{21}	T_{22}	T_{23}	T_{24}
T_3	T_{31}	T_{32}	T_{33}	T_{34}
T_4	T_{41}	T_{42}	T_{43}	T_{44}

Tabla 3. Matriz de costos de comunicación para las tareas T_1, T_2.

	T_1	T_{11}	T_{12}	T_{13}	T_2	T_{21}	T_{22}
T_1	0	3	0	3	0	0	0
T_{11}	2	0	1	4	0	0	0
T_{12}	0	1	0	2	0	0	0
T_{13}	3	5	3	0	0	0	0
T_2	0	0	0	0	1	3	0
T_{21}	0	0	0	0	1	3	0
T_{22}	0	0	0	0	2	0	4

Tabla 4. Matriz de costos de comunicación para las tareas T_3, T_4.

	T_3	T_{31}	T_{32}	T_{33}	T_4	T_{41}
T_3	0	1	3	2	0	0
T_{31}	1	0	1	2	0	0
T_{32}	4	5	0	1	0	0
T_{33}	2	5	2	0	0	0
T_4	0	0	0	0	0	3
T_{41}	0	0	0	0	2	0

Generación de una Primera Asignación

Para generar la primera asignación se toma la matriz de estado de la malla, haciendo una elección aleatoria de las tareas tomadas de la cola de espera y cuyo número de procesadores que solicitan están disponibles en las submallas libres. Para realizar la elección aleatoria se genera un número aleatorio, tomando como base el número de tareas en la cola, si la tarea elegida cabe en la submalla o submallas libres, entonces se considera para calcular su asignación y costo, en caso contrario se elige otra tarea. Para nuestro caso de ejemplificación las primeras tres tareas elegidas son la tarea T_{11}, T_{12} y T_{13}; la forma en que se ubican en la malla se muestra en la Tabla 5.

Tabla 5. Ubicación de las tareas T_{11}, T_{12} y T_{13} que representan la primera asignación en la malla.

1	1	1	1
1	1	1	1
T_{11}	T_{12}	T_{21}	1
T_1	T_{13}	T_2	T_{22}

Para la obtención del primer valor de la función objetivo se realiza el cálculo del costo de la asignación para cada tarea, a base de los costos de comunicación entre tareas y las distancias entre procesadores, considerando el paso de mensajes de un procesador a otro y viceversa. Al considerar el paso de mensajes entre procesadores se debe calcular el costo de transferencia de los mismos, del origen al destino y viceversa, es decir, para el caso de ejemplificación la tasa de transferencia de la tarea T_1 a la subtarea T_{11} es diferente de la subtarea T_{11} a la tareaT_1, aunque puede ocurrir que ambos pesos sean iguales. Los valores de las distancias entre procesadores permanecen iguales. Así, los valores a calcular están dados en las operaciones mostradas en la Tabla 6 para la tarea T_1, y en Tabla 7 para la tarea T_2. Los totales de los respectivos individuos se suman para obtener el costo total de la solución, como se representa en la Tabla 6, con un total de 35 para la tarea T_1 y 17 para la tarea T_2 que aparece en la Tabla 7. La representación de los cálculos anteriores están dados por la ecuación (1).

Generación de la segunda asignación

La segunda asignación se genera al producir una nueva asignación en la submalla libre, y que corresponde al asignar las tareas T_3 y T_4 como lo muestra la Tabla 8. Se realiza el cálculo del segundo valor de la función objetivo de la misma forma que el anterior, los valores están dados en la Tabla 9 para la tarea T_3, y en la Tabla 10 para la tarea T_4.

Tabla 5. Matriz de asignación de tareas según la matriz de estado de la malla en el tiempo *t*, que representa una primera solución del problema de asignación cuadrática dinámica.

1	1	1	1q
1	1	1	1
T_{11}	T_{12}	T_{21}	1
T_1	T_{13}	T_2	T_{22}

Tabla 6. Cálculo del costo de transferencia de mensajes para la tarea T_1.

$T_1 \rightarrow T_{11}$	$T_{11} \rightarrow T_1$	(3+2)*1	5
$T_1 \rightarrow T_{12}$	$T_{12} \rightarrow T_1$	(0+0)*2	0
$T_1 \rightarrow T_{13}$	$T_{13} \rightarrow T_1$	(3+3)*1	6
$T_{11} \rightarrow T_{12}$	$T_{12} \rightarrow T_{11}$	(1+1)*1	2
$T_{11} \rightarrow T_{13}$	$T_{13} \rightarrow T_{11}$	(4+5)*2	18
$T_{12} \rightarrow T_{13}$	$T_{13} \rightarrow T_{12}$	(2+3)*1	5
		Total	35

Tabla 7. Cálculo del costo de transferencia de Mensajes para la tarea T_2.

$T_2 \rightarrow T_{21}$	$T_{21} \rightarrow T_2$	(1+2)*1	3
$T_2 \rightarrow T_{22}$	$T_{22} \rightarrow T_2$	(3+4)*1	7
$T_{21} \rightarrow T_{21}$	$T_{22} \rightarrow T_{21}$	(4+3)*1	7
		Total	17

Tabla 8. Matriz de asignación de tareas según la matriz de estado de la malla en un tiempo *t*, que representa una segunda solución del problema de asignación cuadrática dinámica.

1	1	1	1
1	1	1	1
T_{31}	T_{33}	0	1
T_3	T_{32}	T_4	T_{41}

Tabla 9. Cálculo del costo de transferencia de mensajes para la tarea T_3.

$T_3 \rightarrow T_{31}$	$T_{31} \rightarrow T_3$	(1+1)*1	1
$T_3 \rightarrow T_{32}$	$T_{32} \rightarrow T_3$	(3+4)*1	7
$T_3 \rightarrow T_{33}$	$T_{33} \rightarrow T_3$	(2+2)*2	8
$T_{31} \rightarrow T_{32}$	$T_{32} \rightarrow T_{31}$	(1+5)*2	12
$T_{31} \rightarrow T_{33}$	$T_{33} \rightarrow T_{31}$	(2+5)*1	7
$T_{32} \rightarrow T_{33}$	$T_{33} \rightarrow T_{32}$	(1+2)*1	3
		Total	39

Tabla 10. Cálculo del costo de transferencia de mensajes para la tarea T_4.

$T_4 \rightarrow T_{41}$	$T_{41} \rightarrow T_4$	(3+2)*1	5
		Total	5

Generación de la tercera y cuarta asignación

La generación de la tercera y cuarta asignación mostrada en la Tabla 11 se produce al asignar a la malla las tareas T_2 y T_4, cuyos valores obtenidos mediante los cálculos en la primera y segunda generación, son aplicados en esta para obtener el tercer valor de la función objetivo. En la cuarta generación se asignan las tareas T_2 y T_3 a la malla, para obtener un cuarto valor de la función objetivo. La asignación producida se muestra en la Tabla 12.

En nuestro ejemplo hemos mostrado un caso en el que todos los procesadores aparecen totalmente adyacentes, pero en caso de encontrarse disjuntos, el proceso para calcular las distancias es el mismo.

Evaluación de la población

Una vez que se han calculado los costos de las asignaciones en las Tablas: 6, 7, 9 y 10. Los totales obtenidos se suman para hacer la evaluación de un individuo mediante el valor obtenido en la función objetivo.

Tabla 11. Matriz de asignación de tareas según la matriz de estado de la malla en el tiempo t, que representa una tercera solución del problema de asignación cuadrática dinámica.

1	1	1	1
1	1	1	1
0	0	T_{21}	1
T_4	T_{41}	T_2	T_{22}

Tabla 12. Matriz de asignación de tareas según la matriz de estado de la malla en el tiempo *t*, que representa una cuarta solución del problema de asignación cuadrática dinámica.

1	1	1	1
1	1	1	1
T_{31}	T_{33}	T_{21}	1
T_3	T_{32}	T_2	T_{22}

Estimar el modelo probabilístico

En esta parte vamos a utilizar el modelo probabilístico más simple, en el que todas las variables que describen el problema son independientes, por tanto, calculamos las frecuencias de aparición de una tarea en cada celda vacía de la malla en el tiempo *t*, de una parte de la población que son los mejores individuos, mediante una selección por truncación y el porciento de la truncación. Para este caso las frecuencias de aparición pueden mostrarse en la Tabla 13, por cuestiones de espacio se muestran las frecuencias para el procesador 0 únicamente.

Guardar el mejor individuo

Este proceso es llevado a cabo al tomar de cada población generada, el mejor individuo en el momento en que se ordena la población actual. El proceso de ordenamiento de menor a mayor se genera para obtener la minimización de la asignación.

RESULTADOS OBTENIDOS

Esta sección presenta los resultados obtenidos en una segunda fase experimental del método propuesto. De igual forma, la comparación se realiza con los dos métodos de asignación de tareas: la asignación lineal y el método de las curvas de Hilberth.

Tabla 13. Frecuencias de aparición de cada tarea en cada celda.

	P(0,0)	P(0,1)	P(0,2)	P(0,3)	P(1,0)
T_1	1	0	0	0	0
T_{11}	0	0	0	0	1
T_{12}	0	0	0	0	0
T_{13}	0	1	0	0	0
T_2	0	0	3	0	0
T_{21}	0	0	0	0	0
T_{22}	0	0	0	3	0
T_3	2	0	0	0	0
T_{31}	0	0	0	0	2
T_{32}	0	2	0	0	0
T_{33}	0	0	0	0	0
T_4	1	0	1	0	0
T_{41}	0	1	0	1	0
å	4	4	4	4	3

Parámetros utilizados en los experimentos

Los parámetros utilizados en los experimentos son:

1. Dimensión de la malla. Atendiendo la definición 2 la que referencia la dimensión como M (W, L) que consiste de W X L procesadores, donde W es el ancho de la malla y L es la altura de la malla.
2. Número de tareas al inicio de la prueba. Se considera como un conjunto inicial de tareas para ejecutar la primera asignación directa en la malla de procesadores, una vez que las tareas se empiezan a retirar de la malla el método de asignación que utiliza la metaheurística es activado.
3. Número de subtareas por tarea. Es un parámetro variable definido por el usuario el que indica al sistema el número máximo de subtareas que una tarea puede tener y que en términos de un problema computable y paralelizable es el número de subproblemas en que podrá ser dividido.
4. Tiempo promedio de espera de las tareas. Es el cálculo de la sumatoria de los tiempos de espera de las tareas entre el número total de las tareas.
5. Número total de tareas. Es el número total de las tareas que el sistema atendió.
6. Tiempo de verificación de terminación de las tareas. Es un tiempo establecido por el sistema para la verificación de los tiempos de terminación de las tareas; cuando el sistema verifica la terminación de una tarea puede ocurrir:
   (a) Que la tarea haya finalizado su tiempo.
   (b) Que la tarea aun no finalice su tiempo y únicamente lo disminuya en relación al tiempo transcurrido en el sistema.
7. Tiempo total de ejecución. Es el tiempo total que el sistema tarda en ejecutar n tareas.
8. Número de asignaciones realizadas. Se establece como el número de asignaciones que el algoritmo realiza para procesar la totalidad de las tareas.
9. Promedio de subtareas que tienen las tareas. Se obtiene al sumar la totalidad de subtareas de cada tarea entre el número total de tareas.

Descripción de experimentos

Los experimentos se han dividido en cuatro subsecciones, atendiendo a cada uno de los objetivos propuestos, pero a diferencia de la forma en que se propuso en trabajos anteriores, las cargas de trabajo en el sistema se han modificado con diferentes tamaños, para verificar el funcionamiento del algoritmo propuesto.

1. Número de asignaciones vs. número de tareas. Debido a que los métodos Lineal y Hilberth no utilizan una planificación de tareas en la cola de espera, sino que hacen uso de una planificación FIFO, los resultados en el número de asignaciones realizadas es el mismo con las cargas de 0 a 16538 tareas en procesamiento. El método propuesto tiende a tener resultados semejantes a los dos métodos con cargas menores a las 2048 tareas, como la muestra la Figura 6, pero cuando las cargas se incrementan, los resultados tienden a mejorar significativamente provocando que las tareas reciban atención más rápidamente para su ejecución. La planificación previa que se realiza en la cola de espera de las tareas, permite que no solamente una tarea sea asignada a la malla de procesadores, sino un conjunto de las mismas, provocando un decremento de los tiempos de espera y un incremento en el número de tareas que son aceptadas en la malla. Es importante señalar que debido a que el método de Hilberth y el método Lineal presentan los mismos resultados en cuanto al número de asignaciones, las

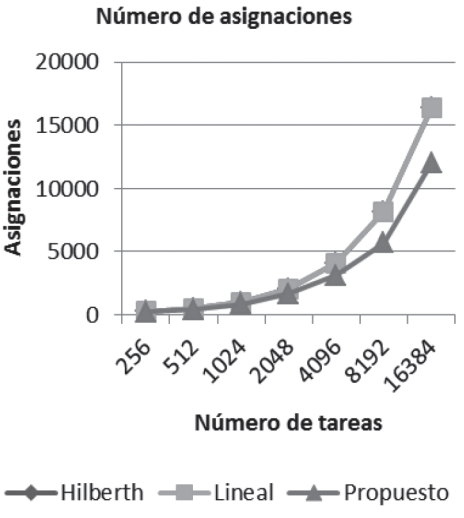

Figura 6. Número de asignaciones *vs.* número de tareas.

gráficas se sobreponen en la imagen, esto debido a que no utilizan una técnica de planificación previa a la asignación de las tareas en la malla.

2. Tiempo promedio de espera de las tareas (expresado en segundos). En la Figura 7 la comparativa de los tiempos de espera se observa que el método propuesto alcanza mejores tiempos con las cargas iniciales y finales en comparación con los otros métodos. En las cargas centrales el método propuesto iguala al método de las curvas de Hilbert y es superado por el método Lineal, debido a las planificaciones que el método debe realizar antes de la asignación, pero mejoran considerablemente los resultados en las subsiguientes cargas, es decir, planificar antes de asignar produce tiempos de espera largos cuando se tienen pocas tareas, pero al incrementarlas los tiempos de respuesta se mejoran.

3. Fragmentación externa. Para el porcentaje de ocupación de procesadores se grafica para cada carga de trabajo como se muestra en la Figura 8. En experimentos anteriores el método propuesto presentaba mejores resultados en cuanto a porcentajes ocupacionales; en este experimento, el número de subtareas por tarea se incrementó significativamente, lo que provoca que las cargas ocupacionales por cada vez que se realiza una asignación, un mayor número de procesadores quede libre, igualando los resultados al método lineal en diferentes cargas del sistema.

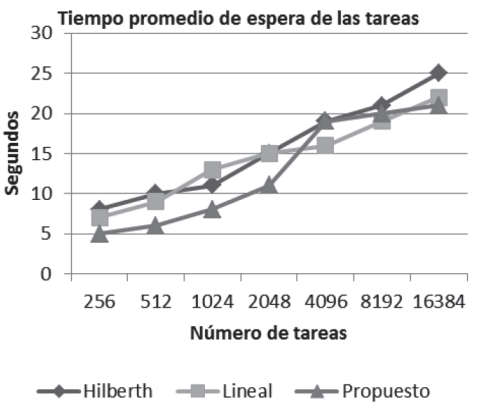

Figura 7. Tiempo promedio de espera de las tareas.

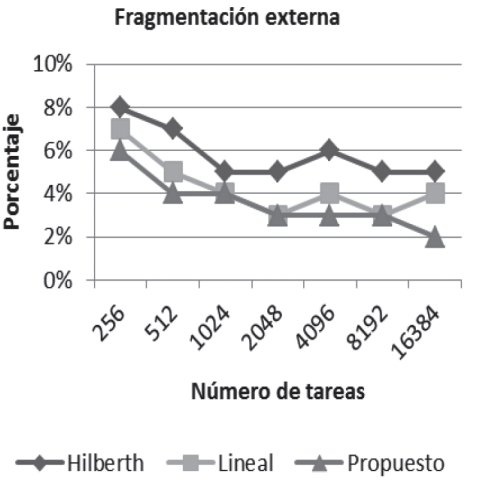

Figura 8. Fragmentación externa.

CONCLUSIONES

Nuevas cargas de trabajo han sido propuestas en las experimentaciones explicadas en este trabajo de investigación; al evaluar cada uno de los objetivos se observa que las tendencias que muestra el algoritmo son de crecimiento en los tiempos de respuesta del sistema para cada parámetro evaluado; lo anterior nos demuestra que las cargas de trabajo tienen relación directa con el algoritmo que las procesa, es decir, a mayor carga los tiempos tienen un efecto devastador, si no existe el algoritmo adecuado tanto en la planificación como en la asignación de las tareas en la malla de procesadores. El método propuesto aun con cargas pesadas ofrece tiempos de respuesta satisfactorios, en comparación con los dos métodos de comparación, aunque con ciertas cantidades en el número de tareas, es similar a los otros dos métodos, lo que indica que son necesarios cambios en el algoritmo evolutivo, por lo que un mejoramiento en los ciclos de generación de poblaciones se hace necesario. Para finalizar el aspecto evaluación multiobjetivo, se debe concluir con el análisis del frente de Pareto, por lo que en un trabajo posterior se propondrá un análisis que tenga relación directa con las cargas de trabajo y con cada uno de los objetivos a ser evaluados. Lo cierto es que al evaluar el análisis de la planificación y asignación de las tareas, los objetivos que se persiguen no pueden ser vistos de manera aislada, porque al mejorar un objetivo puede ser sacrificado otro de manera directa.

REFERENCIAS

[1] A. Grama, A. Gupta, G. Karypis and G. Kumar. "Introduction to Parallel Computing". Second Edition. Addison Wesley. New York, USA. January 16, 2003. ISBN: 0-201-64865-2.

[2] S. Bani, I. Ababneh and M. Ould. "A 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] I. Ababneh and S. Bani-Mohammad. "A new window-based job scheduling scheme for 2D mesh multicomputers". Simulation Modelling Practice and Theory. Vol. 19, Issue 1, pp. 482-493. 2011.

[4] A. Bani. "Submesh Allocation in 2DMesh multicomputers: Partitioning at the Longest Dimension of Request". The International Arab Journal of Information Technology. Vol. 10, Issue 3, pp. 245-252. May, 2013.

[5] C. Chang and P. Mohapatra. "Performance improvement of allocation schemes for mesh-connected computers". Journal of Parallel and Distributed Computing. Vol. 52, Issue 1, pp. 40-68. 1998.

[6] D. Das and D. Pradhan. "Job Scheduling in Mesh Multicomputers". IEEE Transactions On Parallel And Distributed Systems. Vol. 9, Issue 1, pp. 57-70. January, 1998

[7] I. Foster. "Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering". Addison Wesley. New York. 1995.

[8] V. Lo, K. Windisch, W. Liu and B. Nitzberg. "Non-contiguous processor allocation algorithms for mesh-connected multicomputers". IEEE Transactions on Parallel and Distributed Systems. Vol. 8, Issue 7, pp. 712-726. 1997.

[9] H. Heiss. "Dynamic Partitioning of Large Multicomputer Systems Department of Informatics". Proc. Int. Conf. on Massively Parallel Computing Systems (IEEE MPCS94), Ischia. Vol. 1 pp. 1-6. University of Karlsruhe. Germany. 1994.

[10] C. Xavier and S. Iyengar. "Introduction to Parallel Algorithms". Editorial: Wiley-Interscience. New York. 1998. ISBN: 0-471-25182-8.

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

[12] A. Velarde, E. Ponce and E. Díaz. "Planning and Allocation Tasks in a Multi-computer System as a Multi-objective Problem". Advances in Intelligent Systems and Computing 227. EVOLVE 2013. International Conference held at Leiden University, Leiden, The Netherlands. Springer. July 10-13, 2013.

[13] A. Velarde. Tesis Doctoral, Planificación y asignación de tareas en un sistema de multicomputadoras usando algoritmos evolutivos multiobjetivo. Universidad Autónoma de Aguascalientes. México. Febrero 2014.

[14] S. Bani, M. Ould, I. Ababneh and L. Machenzie. "Non-contiguous Processor Allocation Strategy for 2D Mesh Connected Multicomputers Based on Sub-meshes Available for Allocation". Proceedings of the 12th International Conference on Parallel and Distributed Systems (ICPADS006). Minneapolis, Minnesota, USA. IEEE Computer Society Press. Vol. 2, pp. 41-48. 2006.

[15] R. Zolfaghari. "Efficient Algorithm for Processor Allocation in Mesh Multicomputers Network with Limitations and Assumptions". IJCEM International Journal Of Computational Engineering & Management. Vol. 16 Issue 4, pp. 5-13. July, 2013. ISSN: 2230-7893 (Online).

[16] K. Deb. "Multi-Objective Optimization using Evolutionary Algorithms". Editorial: John Wiley & Sons, LTD. New York, U. S. A. 2001. ISBN: 0-471-87339-X.

[17] J. Albert and R. Niedermeier. "On multi-dimensional Hilbert indexings". Theory of Computing Systems. Vol. 33. 2000.

[18] P. Walker P. Bunde and J. Leung. "Faster high-quality processor allocation". Sandia National Laboratories. URL: https://cfwebprod.sandia.gov/cfdocs/CompResearch/docs/lciconf.pdf

[19] L. Lo, A. Hua and C. Young. "GeMDA: A Multidimensional Data Partitioning Technique for Multiprocessor Database. Systems". Distributed and Parallel Databases, 9, pp. 211-236. Kluwer Academic Publishers. 2001.

[20] P. Larrañaga, J.A. Lozano and H. Mühlenbein. "Algoritmos de estimación de distribuciones en problemas de optimización combinatoria. Inteligencia Artificial". Revista Iberoamericana de Inteligencia Artificial. 2003.

[21] J.A. Lozano and P. Larrañaga. Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Kluwer Academic. 2005.