ASIGNACIÓN DE TAREAS A PROCESADORES EN UN SISTEMA DISTRIBUIDO DE TIEMPO REAL DURO UTILIZANDO ALGORITMOS GENÉTICOS Y LÓGICA DIFUSA

Edgardo Ferro, Ricardo Cayssials y Javier Orozco,
Dep. Ingeniería Eléctrica, Instituto de Ciencias e Ingeniería de Computación, CONICET.
Universidad Nacional del Sur, Av. Alem 1253, (8000) Bahía Blanca.
E-mail: ieferro@criba.edu.ar

Palabras Claves: Asignación, Diagramabilidad, Sistemas Distribuidos de Tiempo Real, Algoritmos Genéticos, Lógica Difusa.

Resumen

Se presenta un método basado en algoritmos genéticos para atacar el problema de asignación de un conjunto de tareas apropiativas, sobre un conjunto de procesadores distribuidos que deben trabajar en un entorno de tiempo real duro. Las tareas son cooperativas y utilizan como vía de comunicación una red local. Los coeficientes que ponderan la función de costo del algoritmo genético son calculados utilizando operadores difusos. Sobre el sistema existe un conjunto de restricciones que debe ser satisfecho para obtener una solución compatible con los requerimientos de tiempo real duro.

1. Introducción

El comportamiento de un Sistemas de Tiempo Real (STR) es correcto si los resultados que produce son siempre correctos desde los puntos de vista lógico y temporal. Complejos sistemas mecatrónicos de tiempo real contienen decenas de procesadores en su sistema de control. Para estos propósitos, el sistema operativo debe reducir lo más posible la sobrecarga impuesta por él mismo, balancear el factor de utilización, sincronizar el uso de recursos, cumplir con las restricciones temporales de las tareas y del medio de acoplamiento entre procesadores.

El proceso de asignación de tareas a procesadores en forma automática y estática teniendo en cuenta los objetivos antes descriptos resulta un problema NP-Completo (Garey 1979). Luego, el desarrollo de herramientas para tratar el problema resulta imprescindible a la hora de diseñar un sistema distribuido de tiempo real duro. Diferentes técnicas han sido utilizadas intentando la búsqueda de una solución, si es que existe alguna, (Ma 1990, Xu 1993) o encontrar la mejor

solución posible según una cierta función objetivo (Peng 1989, Blazewicz 1994). Investigaciones recientes cubren los siguientes tópicos al utilizar técnicas de búsqueda no guiada: Tabu Search, (Glover 1993); Recocido Simulado, (Tindell 1992, Nicholson 1993, Borriello 1994, Coli 1995); Redes Neuronales, (Cardeira 1996); Lógica Difusa (Orozco 1996); Algoritmos Genéticos, (Jones 1991, Mansour 1992, Sandnes 1996); métodos heurísticos, (Peng 1989, Ramamritham 1990, Santos 1997, Ferro 1996). En este trabajo se propone un método basado en algoritmos genéticos para particionar un conjunto de tareas donde los coeficientes que pesan la función de costo son obtenidos por un experto difuso antes de resolver el problema de asignación.

2. Restricciones

En ambientes de tiempo real son comunes los sistemas en los que un conjunto de tareas comparte un único recurso. Debido a restricciones temporales, el uso del mismo debe ser garantizado a cada tarea dentro de un cierto intervalo contado a partir del instante en que se genera el requerimiento.

Para el cálculo de la diagramabilidad se utiliza el método de las ranuras vacías (Santos 1993) y su extensión (Ferro 1996a) que incluye el bloqueo más largo K_i que puede sufrir una tarea cuando se comparten datos y/o recursos (ecuaciones 1 y 2). Para su ejecución, cada tarea requiere una determinada cantidad de memoria. Antes de asignar una tarea a un procesador, es necesario verificar que el mismo disponga de una cantidad suficiente para recibirla (ecuación 3).

Para un número importante de sistemas, es imprescindible que las tareas colaboren entre sí para alcanzar un objetivo predeterminado. Para ello deben transferir información entre sí y, si se encuentran en distintos procesadores, es necesaria una red de interconexión. La red por su parte debe implementar un mecanismo de acceso al medio que admita tráfico de tiempo real. En este trabajo consideramos que la misma utiliza una disciplina de rueda cíclica ya que resulta una disciplina natural en varios protocolos estandarizados. Luego, el peor retardo de comunicaciones que puede sufrir una tarea al enviar los datos a otra es calculado teniendo en cuenta que, una tarea para poder hacer uso del medio de comunicación mediante dicha disciplina, tendrá que esperar, en el peor caso, el tiempo de transmisión de todas las demás que comparten el mismo medio físico (ecuaciones 4 y 5).

Luego, cuando una tarea τ_p debe enviar datos a otra que reside en un procesador distinto, el

vencimiento de la tarea predecesora D_p debe ser corregido teniendo en cuenta el retardo de comunicaciones (Santos 1997, ecuación 6). Una tarea que requiere datos de otra para poder ejecutarse se denominará *sucesora*, siendo denominada la tarea origen de los datos *predecesora*. Una tarea sucesora estará lista para ser ejecutada cuando el último byte de información que transmitieron sus predecesoras es leído por el nodo que la aloja.

<table>
<tr><td colspan="3">Diagramación</td></tr>
<tr><td colspan="2">$$\sum_{h=1}^{m} \frac{C_h}{T_h} \le 1 \quad (1)$$
C_i = Tiempo de ejecución de la tarea τ_i.
T_i = Periodo de requerimiento de la tarea τ_i.</td><td>$$D_i \ge e_{(C_i+K_i)(i-1)} = \text{menor } t | t = K_i + C_i + \sum_{h=1}^{m} C_h \lceil t / T_h \rceil \quad (2)$$
C_i = Tiempo de ejecución de las tareas τ_i.
T_i = Periodo de requerimiento de la tarea τ_i.
D_i = Tiempo de vencimiento de las tarea τ_i</td></tr>
<tr><td>Memoria</td><td>Comunicaciónes</td><td>Precedencia</td></tr>
<tr><td>$$M_h \le M_p \ (3)$$
M_h: memoria requerida por τ_h.
M_p: memoria disponible en el procesador P</td><td>$$\sum_{i=1}^{m} B_i \le P * D_{min} \ (4) \quad \text{y} \quad \Delta = \left\lceil \frac{\sum_{i=1}^{m} B_i}{P} \right\rceil \ (5)$$
B_i: número total de bytes que τ_i envía a cada una de sus sucesoras. P: ancho de banda útil y D_{min}: mínimo plazo de vencimiento del conjunto de tareas</td><td>$$D_p^* = D_p - \Delta \quad (6)$$
D_p: vencimiento de la tarea predecesora.
Δ: retardo de comunicaciones y D^*_P el vencimiento corregido</td></tr>
</table>

Tabla 1: Conjunto de restricciones de un sistema distribuido de tiempo real duro

Las tareas que deben ser ejecutadas incondicionalmente en un procesador determinado reciben el nombre de *preasignadas*. Aquellas que no tengan predeterminado un procesador se las llama *libres* y pueden ser alojadas, en principio, en cualquiera de ellos.

Cuando es necesario crear *réplicas* de tareas para implementar sistemas con tolerancia a fallas, la tarea original y la réplica no podrán coexistir en un mismo procesador debiéndose tener en cuenta este hecho en el mecanismo de asignación.

3. Modelo

Un conjunto de *m* tareas periódicas de tiempo real apropiativas deben ser ejecutadas en *n* procesadores, cada uno de los cuales posee una cantidad determinada de recursos. Si no hubiese

condicionamientos podrían realizarse n^m asignaciones diferentes. El objetivo es obtener una asignación de las *m* tareas en los *n* procesadores que cumpla con las diferentes restricciones de un contexto de tiempo real duro. De las n^m asignaciones, es posible que sólo un subconjunto de ellas cumpla con las condiciones impuestas por el sistema de tiempo real. Es evidente que el problema puede no ser computable en tiempo razonable para valores de n^m considerables.

Las tareas son parte de un *trabajo* ϑ que es representado por un grafo dirigido y acíclico con la propiedad de clausura transitiva para evitar arcos redundantes. En el grafo, los *nodos* representan a las *tareas* y están identificadas por una dupla que indica su factor de utilización y las unidades de *recursos* que necesita. El peso de los arcos, indica la cantidad de información medida en bytes que intercambian las tareas.

Una tarea τ_i es denominada terminal, si y solo si no existen arcos salientes de τ_i. Así mismo una tarea τ_j es denominada fuente, si y solo si no existen arcos entrantes a τ_j. Por otro lado τ_i y τ_j son predecesor y sucesor respectivamente. ($\tau_i \prec \tau_j$) si y solo si existe un arco desde τ_i a τ_j y no existe $\tau_k \mid \tau_I \prec \tau_k \prec \tau_j$. A partir de la definición de las componentes del grafo podemos formalizar el modelo y definir la estrategia a adoptar.

Se define al multigrafo $\vartheta = \{\mathbf{T}, \mathbf{B}, \Phi_1, \Phi_2, \Phi_3\}$, donde $\mathbf{T} = \{\tau_1, \tau_2, ...\tau_m\}$ es el conjunto de nodos, $\mathbf{B} \subseteq \mathbf{T} \times \mathbf{T}$ es el conjunto de arcos, $\Phi_1 : \mathbf{B} \rightarrow \mathbf{Z}^+$ define los pesos de los arcos, $\Phi_2 : \mathbf{T} \rightarrow \mathbf{Z}^+$ define la primer componente asociada al nodo, $\Phi_3 : \mathbf{T} \rightarrow \mathbf{Z}^+$ define la segunda componente asociada al nodo y $\mathbf{Z}^+$ es el conjunto de los enteros positivos.

El problema de particionar en n-subconjuntos se reduce a encontrar una función de mapeo:

$$\Pi: \mathbf{T} \rightarrow \left\{\pi_1, \pi_2, ..., \pi_n,\right\} \text{ donde } \pi_i = \left\{\tau \in \mathbf{T} \mid \Pi(\tau) = \pi_i\right\} \quad \forall i \mid 1 \le i \le n \tag{7}$$

tal que

$$\pi_i \cap \pi_j = \varnothing \quad \text{para} \quad i \neq j \quad \text{y} \quad \bigcup_{i=1}^{n} \pi_i = \mathbf{T} \tag{8}$$

$U(\pi_i)$ indica el factor de utilización del subconjunto π_i y se define como $\Sigma\Phi_2(\tau)\ \forall\, \tau \in \pi_i$.

$M(\pi_i)$ indica el consumo de recursos del subconjunto π_i y se define como $\Sigma\Phi_3(\tau)$, $\forall\, \tau \in \pi_i$.

$X_{i,j}$ indica la cantidad de bytes que deberán ser transmitidos por el canal de comunicaciones al particionar entre dos subconjuntos π_i y π_j de Π y se define como $\Sigma\Phi_1(\tau_h, \tau_k)\ \forall (\tau_h, \tau_k) \in \mathbf{B} \mid\ \tau_h \in \pi_i$ y $\tau_k \in \pi_j$.

El objetivo es minimizar los pesos de Π al efectuar la partición:

$$Y(\Pi) = \sum_{\forall i,j | 1 \le i \le j \le n} X_{i,j}$$

al que el desbalance del factor de utilización W(Π) y de recursos R(Π) entre cada π_i sea mínimo.

$$W(\Pi) = \sum \left| U(\pi_i) - U(\pi_j) \right| \ \forall i, j \mid 1 \le i \le j \le n \tag{10}$$

$$R(\Pi) = \sum \left| M(\pi_i) - M(\pi_j) \right| \ \forall i, j \mid 1 \le i \le j \le n \tag{11}$$

4. Aproximación Difusa

El concepto de difuso se asocia al de la ambigüedad, incerteza, e intuitivamente aplicamos el adjetivo difuso a un conjunto que adolece de borde bien definidos. A partir de Bellman y Zadeh (1977) se conoce como lógica difusa, a la llamada lógica del razonamiento aproximado. Para un conjunto difuso, además de elementos que pertenecen y elementos que no pertenecen al mismo tendremos elementos con distintos grados de pertenencia al conjunto. En general es un inconveniente determinar este grado de pertenencia, el cual que *a priori* será impreciso. Luego, la lógica subyacente a la teoría de conjuntos difusos es multivalente, y los valores asignados corresponden a un conjunto (finito o no) de valores entre 0 y 1, donde no es aplicable el principio del tercero excluido.

Una relación binaria difusa **R** entre dos conjuntos difusos **S** y **S'**, es un subconjunto difuso de **S** x **S'**. Si **S** = $\{s_1,.....,s_m\}$ y **S'** = $\{s'_1,.....s'_n\}$ y **R** ⊆ **S** x **S'**, la relación difusa **R** puede expresarse por la siguiente matriz de *m* x *n*:

$$\begin{bmatrix} \mu_R(s_1,s'_1) & \mu_R(s_1,s'_2) & \mu_R(s_1,s'_3) & \mu_R(s_1,s'_n) \\ \mu_R(s_2,s'_1) & \mu_R(s_2,s'_2) & \mu_R(s_2,s'_3) & \mu_R(s_2,s'_n) \\ ..... & ..... & ..... & ..... \\ \mu_R(s_m,s'_1) & \mu_R(s_m,s'_2) & \mu_R(s_m,s'_3) & \mu_R(s_m,s'_n) \end{bmatrix}$$

La resolución de ecuaciones difusas tiene importantes aplicaciones y la existencia y determinación de sus soluciones es en general un problema ampliamente tratado (Terano 1991). Veamos el caso en el cual suponemos **A** ⊆ **S** y una relación binaria **R** ⊆ **S** x **S'**, donde obtenemos la solución **B** al aplicar la relación **R** al subconjunto **A** (notado **B** = **A** ○ **R**).

En un conjunto totalmente ordenado en el intervalo [0, 1] con último elemento, se define "pseudo complemento de *a* con respecto a b" al elemento:

$$a \to b = \begin{cases} 1 & \text{si} \quad a \le b \\ b & \text{si} \quad a > b \end{cases}$$

A partir de la definición de pseudo complemento entre elementos de un conjunto ordenado vamos a indicar la operación binaria $\to$ entre relaciones difusas. Si $\mathbf{A} \subseteq \mathbf{S}$, $\mathbf{B} \subseteq \mathbf{S}'$, la relación $\mathbf{A} \to \mathbf{B} \subseteq \mathbf{S}$ x $\mathbf{S}'$ se define por medio de:

$$\mu_{\mathbf{A}\to\mathbf{B}}(s,s') = \mu_{\mathbf{A}}(s) \to \mu_{\mathbf{B}}(s') \tag{12}$$

Dados $\mathbf{A} \subseteq \mathbf{S}$, $\mathbf{B} \subseteq \mathbf{S}'$, si existe al menos una solución de la ecuación $\mathbf{B} = \mathbf{A}$ o $\mathbf{R}$, entonces $\mathbf{R}' = \mathbf{A} \to \mathbf{B}$ es la mayor relación que la satisface (Dubois 1980). Si consideramos el problema inverso, es decir: dada una relación difusa $\mathbf{R} \subseteq \mathbf{S}$ x $\mathbf{S}'$ y un conjunto difuso $\mathbf{B} \subseteq \mathbf{S}'$, y si existe al menos una solución a la ecuación $\mathbf{B} = \mathbf{A}$ o $\mathbf{R}$, entonces ésta será el mayor conjunto difuso $\mathbf{A'} \subseteq \mathbf{S}$ que satisface $\mathbf{A}' = \mathbf{R} \to \mathbf{B}$ (Dubois 1980). Se pueden inferir diagnósticos y pronósticos a partir de síntomas observados, por medio de un conocimiento específico. La determinación de **R** en $\mathbf{B} = \mathbf{A}$ o $\mathbf{R}$, modela la adquisición de conocimiento a partir de experimentos, mientras que la determinación de **A** en $\mathbf{B} = \mathbf{A}$ o $\mathbf{R}$ modela la búsqueda de una causa difusa.

5. Calculo de los Coeficientes de Sintonización

Para el cálculo de los coeficientes que pesarán la ecuación de costo del algoritmo genético nos ocuparemos del caso donde la relación entre síntomas y causas se expresa por una ecuación difusa. Al iniciarse el proceso sólo obtenemos información ambigua, a menudo intuitiva: "factor de utilización relativamente alto", "bajo consumo de memoria", "velocidad del canal de comunicaciones relativamente alta", "aparente bajo número de procesadores disponibles para asignar", etc.

Sea $\mathbf{B} = \{b_1, b_2, ..., b_n\}$ una representación del conjunto de síntomas y sea $\mathbf{A} = \{a_1, a_2, ..., a_m\}$ una representación del conjunto de causas, y $r_{i,j}$ la intensidad de la relación entre la causa a_i y el síntoma b_j. Sea **R** la matriz elemento genérico para $r_{i,j}$, con $1 \le i \le m$, y $1 \le j \le n$.

De acuerdo a lo ya expresado, el máximo que satisface la relación es $\mathbf{A} = \mathbf{R} \to \mathbf{B}$, es decir:

$$a_i = \bigwedge_j \left(r_{i,j} \rightarrow b_j \right) \tag{13}$$

Si definimos a partir de la experiencia obtenida por el grupo (Orozco 1996) la matriz de relación y obtenemos los síntomas del problema a ensayar, podremos inferir la causa que los ocasiona. Las mismas están representadas por tres coeficientes que determina si son fuertes o no las restricciones antes enumeradas: α_1 (comunicaciones), α_2 (factor de utilización), α_3(recursos, eg: memoria). Supongamos la siguiente matriz de relaciones y síntomas para el siguiente ejemplo:

$$R = \begin{bmatrix} 0.5 & 0.8 & 0.7 & 0.5 & 0.1 & 0.7 & 0.5 \\ 0.1 & 0.3 & 0.4 & 1.0 & 0.1 & 0.3 & 0.3 \\ 0.7 & 0.8 & 0.6 & 0.3 & 0.8 & 0.4 & 0.1 \end{bmatrix} \quad B = \begin{bmatrix} 0.7 & 0.5 & 0.4 & 0.4 & 0.1 & 0.6 & 0.8 \end{bmatrix}$$

entonces la solución máxima aplicando la ecuación 13 será:

$$\alpha_1 = (0.5 \rightarrow 0.7 \ \wedge \ 0.8 \rightarrow 0.5 \wedge \ 0.7 \rightarrow 0.4 \wedge \ 0.5 \rightarrow 0.4 \wedge \ 0.1 \rightarrow 0.1 \wedge \ 0.7 \rightarrow 0.6 \wedge \ 0.5 \rightarrow 0.8$$
$$\alpha_1 = (1 \ \wedge \ 0.5 \ \wedge \ 0.4 \ \wedge \ 0.4 \ \wedge \ 1 \ \wedge \ 0.6 \ \wedge \ 1) = 0.4$$
$$\alpha_2 = (0.1 \rightarrow 0.7 \wedge \ 0.3 \rightarrow 0.5 \wedge \ 0.9 \rightarrow 0.4 \wedge \ 1.0 \rightarrow 0.4 \wedge \ 0.1 \rightarrow 0.1 \wedge \ 0.3 \rightarrow 0.6 \wedge \ 0.3 \rightarrow 0.8$$
$$\alpha_2 = (1 \ \wedge \ 1 \ \wedge \ 1 \ \wedge \ 1 \ \wedge \ 1 \ \wedge \ 1 \ \wedge \ 1) = 1.0$$
$$\alpha_3 = (0.7 \rightarrow 0.7 \wedge \ 0.8 \rightarrow 0.5 \wedge \ 0.6 \rightarrow 0.4 \wedge \ 0.3 \rightarrow 0.4 \wedge \ 0.8 \rightarrow 0.1 \wedge \ 0.4 \rightarrow 0.6 \wedge \ 0.1 \rightarrow 0.8$$
$$\alpha_3 = (1 \ \wedge \ 0.5 \ \wedge \ 0.4 \ \wedge \ 1 \ \wedge \ 0.1 \ \wedge \ 1 \ \wedge \ 1) = 0.1$$

de la solución obtenida se desprende que la causa mas importante es un alto factor de utilización global. Las precedencias deben ser tenidas levemente en cuenta, mientras que el consumo de memoria por parte de las tareas no es una constricción importante a tener en cuenta durante el proceso de asignación.

6. Algoritmos Genéticos

Los Algoritmos Genéticos AGs son métodos de búsqueda y optimización que imitan el proceso de selección natural, manipulando una población de individuos que contienen valores posibles de las variables de un problema especifico (Goldberg 1989). Estos algoritmos de búsqueda están basados en la selección y genética natural, combinando el concepto de supervivencia del mas apto, con un intercambio estructurado pero aleatorio de información. Al introducir sobre la población cambios aleatorios utilizando los operadores genéticos de cruce y mutación se evita la caída en un mínimo local impidiendo que se llegue al mínimo global. Al utilizar la formación histórica de la población los individuos pueden ser orientados hacia regiones del espacio de

búsqueda mas favorables. En un AGs. es necesario evaluar a todos los miembros de la población al principio de cada experimento. Durante la ejecución del algoritmo, sus operadores modifican las estructuras de la población creando nuevos miembros que deben ser evaluados antes de incorporarse a la población. Si utilizamos un enfoque basado en AGs que permite mapear del espacio del problema al espacio de soluciones (Ahmad 1995) estaremos sacando provecho del paralelismo inherente de los AGs. y del problema especifico a resolver.

7. Operadores Genéticos

Mediante el uso de los operadores de selección, cruce y mutación se obtiene la nueva generación formada por una selección de cromosomas de la generación previa. Los cromosomas de mayor aptitud tendrán una probabilidad mayor de ser seleccionados, existiendo la posibilidad de contribuir con mas de un descendiente en la nueva generación. Una vez creada la nueva generación, se elige un par de cromosomas en forma aleatoria. Estos cromosomas son partidos en una posición *p* elegida al azar y se intercambian los primeros *p* genes de un cromosoma con los primeros *p* genes del otro, con una probabilidad P_c. La mutación es un operador secundario que cumple la función de evitar la perdida de información que puede darse por los dos operadores anteriores. En este proceso se recorren todos los genes uno a uno y se los muta con una probabilidad P_m, que generalmente es mucho menor que la probabilidad de cruce.

8. Definición del Costo y la Aptitud

La función costo es la llave que refleja el éxito de la optimización. Seleccionaremos la suma ponderada de las ecuaciones (9), (10) y (11) para calcular el costo de cada cromosoma *i*:

$$Costo(i) = \alpha_1 * \mathrm{Y}(\Pi) + \alpha_2 * W(\Pi) + \alpha_3 * R(\Pi) \qquad (14)$$

donde α_1 , α_2 y α_3 son los pesos que controlan la función objetivo. Estos valores fueron obtenidos a partir de la ecuación difusa al analizar el tipo de problema que se pretende evaluar. Si el problema fuera duro en comunicaciones prevalecerá α_1 para reducir indirectamente el retardo de comunicaciones y poder cumplir con las restricciones de precedencia y diagramación. Si por el contrario el factor de utilización total de las tareas es alto o la memoria consumida es alta,

predominara el valor de α_2 o α_3 sobre los otros coeficientes. El valor de este costo fue utilizado para determinar la aptitud de cada cromosoma *i* en la siguiente forma:

$$A(i) = \frac{\left(CostoMaximo - Costo(i)\right)^{\theta}}{\sum_{j=1}^{N_p}\left(CostoMaximo - Costo(j)\right)^{\theta}} \quad (15)$$

donde *CostoMáximo* es el mayor costo de un cromosoma en la población y θ (cuyo rango va entre 1 y 5), es un parámetro usado para ajustar la escala de la aptitud y balancear la convergencia.

9. Algoritmo de Asignación

El propósito del algoritmo es particionar un conjunto dado de tareas en un grupos procesadores. El problema se reduce a proveer un algoritmo eficiente para buscar el espacio de todas las posibles soluciones, con la intención de poder encontrar una que minimice la función de optimización.

Los pasos que comprende el Algoritmo son:

1. **Calculo de Coeficientes:** Se analizan los síntomas y se resuelve la ecuación difusa para obtener α_1, α_2 y α_3.
2. **Inicialización:** Se leen los grafos que compone el problema y inicializa a: m^{ϑ}_{g} con el número de nodos del grafo ϑ, N_p con el tamaño inicial de la población, N_g con el número de generaciones a evolucionar, P_c con la probabilidad de cruce, P_m con la probabilidad de mutación, *n* con el número de subconjuntos disjuntos (que coincide con el número de procesadores del sistema).
3. **Generar la población inicial:** El primer cromosoma de la población inicial es construido a partir de los datos del problema, mientras que los restantes son obtenidos perturbando el cromosoma inicial. Cada cromosoma tiene asociado un arreglo de números reales que representan la *dominancia* de cada nodo del grafo. La dominancia *O* de cada nodo *i* para el primer cromosoma se calcula como:

$$O_i^1 = \Phi_2\left(\tau_i\right) * \text{INT}\left(random(0,1) * m_g^{\vartheta}\right) \quad (16)$$

Los (N_p-1)-esimos cromosomas restantes se obtienen perturbando aleatoriamente la ecuación anterior:

$$O_i^j = O_i^1 + f(-\eta, \eta) \quad \forall i \,|\, 1 \le i \le m_g^{\vartheta}, \quad \forall j \,|\, 2 \le j \le N_p \quad \eta = \text{int}\left(\max_i \left\{O_i^1\right\} / 10\right) \tag{17}$$

donde f(-η,η) es un número generado con una distribución de probabilidad uniforme entre -η y η.

4. **Particionar:** Una vez obtenidos los N_p cromosomas y ordenados sus genes por valores decrecientes del alelo, se ejecuta el algoritmo para particionar en *n* subconjuntos. El mismo selecciona el gen con el mayor valor de alelo y lo asigna al subconjunto que tenga el menor factor de utilización. El mismo es obtenido sumando los pesos de los nodos mapeados a ese subconjunto. La solución elaborada por la heurística para cada cromosoma calcula los valores resultantes de las ecuaciones (9), (10) y (11) para ser aplicados en el paso siguiente.
5. **Evaluación:** Se calcula el costo y la aptitud para cada uno de los *i* cromosomas dentro de la población.
6. **Selección:** Se obtienen los mejores exponentes de esta generación aplicando los operadores genéticos. Para representar la probabilidad de selección se utiliza la de una ruleta, en la cual cada cromosoma tiene asignada una cantidad de casilleros proporcional a su aptitud.

Los pasos 4 al 6 son repetidos para un número predefinido de iteraciones de acuerdo a N_g.

7. **Solución:** La mejor solución para el grafo ϑ_i es almacenada y se repiten los pasos 2 al 6 para cada uno de los grafos restantes.
8. **Verificación:** Una vez obtenida la solución para cada grafo, se realiza el análisis de todas las particiones encontradas. Sobre las *n* particiones resultantes se aplican los test de restricciones. Si estos son satisfechos en su totalidad para cada partición, se mapean las mismas a los procesadores correspondientes y una solución subóptima a sido alcanzada.

10. Ejemplo

Para evaluar la relación de éxito del algoritmo desarrollado, se construyo un generador aleatorio de grafos (Santos 1997) que representan a distintos trabajos con cuatro, ocho y doce tareas cada uno. En este caso, el número de tareas de cada grafo en el nivel inicial es elegido

aleatoriamente en el intervalo [1,3], [1,7] y [1,11] respectivamente. El número de tareas en los niveles sucesivos es elegido de igual manera que el primero, pero en el intervalo [1, número de tareas de no asignado en los niveles previos]. La conexión entre las tareas de un nivel y otro es establecida en cada caso con una probabilidad del 20%. La carga de comunicaciones medida en byte que es asociada a cada arco fue obtenida multiplicando por 300 el valor entero generado en el intervalo [1,10]. La cantidad de memoria de cada tarea es elegida aleatoriamente en el intervalo [500,5000]. La capacidad de memoria disponible es un determinado porcentaje mas que la memoria consumida por todas las tareas obtenida en el intervalo [15%, 50%],. Este valor es dividido por el número total de procesadores disponibles y asignada en forma homogénea a los mismos. Seis tareas son preasignadas aleatoriamente y dos pares de tareas del mismo nivel son elegidas en forma aleatoria para ser ejecutadas en distintos procesadores. Se ejecutaron cien problemas con un factor de utilización en el intervalo [0.3, 0.8]. Las tareas debían ser asignadas sobre seis procesadores acoplados por una red de comunicaciones operando a 2 Mbit/seg. El algoritmo fue programado en C y ejecutado en una Alpha AXP 3000. El promedio de tiempo para alcanzar la primer solución en los ejemplos donde la misma se pudo obtener fue de 1 segundo, con p_c = 0,1, p_m = 0,001, ϑ = 3, N_p = 100, N_g = 5000, n = 6 y m = 24.

11. Conclusiones

El algoritmo utilizado es una aproximación más al problema de asignación en un Sistema Multitarea-Multiprocesador de tiempo real. Es inmediato observar la dependencia entre el diseño de la matriz de relación y el éxito en la búsqueda de una solución. El problema más difícil es la elección adecuada de las variables difusas que intervienen en la detección de los síntomas. Los métodos basados en AGs son más rápidos que los que utilizan recocido simulado, lamentablemente en el trabajo presentado por Tindell no se especifica el tiempo insumido en alcanzar la solución presentada. Por otro lado el método presentado mejora las heurísticas presentadas en (Santos 1997, Ferro 1996) en cuanto a la calidad de la solución encontrada a costa de un mayor tiempo de cómputo. Comparado con la heurística utilizada en (Ramamrithan 1989) el algoritmo presentado obtiene soluciones para factores de utilización más altos.

El método presentado en Orozco 1996 fue el punto de partida para incorporar la lógica difusa al problema de asignación. El algoritmo genético híbrido presentado en (Mansour 1992) trata el

problema de asignación pero no en un contexto de tiempo real, usando solamente en la función costo el balance de carga. En un contexto puro de tiempo real no existen muchos aportes al problema de asignación usando AGs., en especial cuando se tratan todas las constricciones. Además la diferencia esencial con los AGs. tradicionales es que este algoritmo permite mapear del espacio del problema al espacio de soluciones, y brinda una solución casi óptima en un tiempo reducido, inclusive para problemas muy complejos donde los métodos tradicionales comienzan a decaer

12. Bibliografía

Ahmad, I. and Dhodhi, M. K. (1995). "On the m-Way Graph Partitioning Problem". Computer Journal, pp. 237-244.

Bellman, R. and Zadeh, L. (1977). "Local and Fuzzy Logic in Modern Uses of Multiple Valued Logic". Reidel Publ. Dordrecht, Netherlands.

Blazewicz, J. and Ecker, K. (1994). ¨Multiprocessor task scheduling with Resource Requirements¨, Journal of Real Time Systems, 6(1).

Borriello, G. and D. Miles. (1994). "Task scheduling for real-time multiprocessor simulations". *11th Workshop on RTOSS*, pp. 70-73.

Caldeira, C. (1996). "Neural network versus Max-Flow Algorithms for Multiprocessor Real Time Scheduling". *8th IEEE Euromicro Workshop on Real Time Systems*, pp. 175-180.

Coli, M and Palazzari, P. (1995). "A New Method for Optimization of Allocation and Scheduling in Real-Time Applications". In *7th IEEE Euromicro Workshop on Real Time Systems*, pp. 262-269.

Dubois, D. and Prade, H. (1980). "Fuzzy Sets and Systems, Theory and Aplications". Academic press, new York.

Ferro, E., Orozco, J., Santos J. y Cayssials, R. (1996a). "Sincronización de tareas en Tiempo Real Duro utilizando el método de las Ranuras Vacias". *Información Tecnológica*. Vol 7, Nro 4, pp. 101-107.

Ferro, E., Santos J., Orozco, J. and Cayssials, R. (1996). "Tuning Parameters to improve a Heuristic Method: More and Better Solutions to an NP-Hard Real Time Problem". *8th IEEE Euromicro Workshop on Real Time Systems*, pp. 47-51.

Garey, M. and Johnson, D. (1979). "Computers and Intractability: A guide to the Theory of NP Completeness". W.H. Freeman, San Francisco, CA.

Glover, F., taillard E. and deWarra, D. (1993). ¨User's guide to Tabu Search¨, Annals of OR, pp. 3-28.

Goldberg, D. (1989). "Genetic Algorithms in Search, Optimization and Machine Learning". Addison-Wesley, Reading, MA.

Jones, D. and Beltramo, M. (1991). "Solving partitioning problem with genetic algorithm". *In Proc. Int. Conf. on Genetic Algorithms*, pp. 442-449.

Lawler, L. (1983). ¨Recent Results in the Theory of Machinge Scheduling¨, Springer Verlag, pp. 202-233.

Ma, P.-Yi R., Lee, E. and Tsuchiya, M. (1982). "A task allocation model for distributed computing systems". *IEEE TC*, 31(1), pp. 41-47.

Mansour, N. and Fox G. (1991) "A Hybrid Genetic Algorithm for Task Allocation in Multicomputers". Proceedings of the fourth International Conference on Genetic Algorithms, pp. 466-473.

Nicholson, M. (1993). ¨Allocating and Scheduling Hard Real Time Task on a point to point Distributed Systems¨. In Proceeding of Workshop on Parallel and Distributed Real Time Systems.

Orozco, J., Cayssials, R., Santos J. and Ferro, E. (1996). "Design of a Learning Fuzzy Production System to Solve an NP-Hard Real Time Assignment Problem". *8th IEEE Euromicro Workshop on Real Time Systems*, pp. 146-150.

Peng, D.and Shin, K.(1989). "Static Allocation of Periodic Task with Precedence Constraints in Distributed Real Time Systems". In Proceedings of the 9th International Conference on Distributed Computing Systems, pages 190-198.

Ramamritham, K. (1990). "Allocation and scheduling of complex periodic tasks". *Proc. 10th International Conference on Distributed Computing Systems*, pp. 108-115.

Sandnes, F. (1995). "A hybrid Genetic Algorithm Applied to Automatic Parallel Controller Code Generation". In *8th IEEE Euromicro Workshop on Real Time Systems*, pp. 70-75.

Santos, J. and Orozco, J. (1993). "Rate Monotonic Scheduling in Hard Real Time Systems". *Information Processing Letters*, No. 48, pp. 39-45.

Santos, J., Ferro, E., Orozco, J. and Cayssials, R (1997). ¨A Heuristic Approach to the Multitask-Multiprocessor Assignment Problem Using The Empty-Slots method and Rate Monotonic Scheduling". *Real-Time Systems Journal*, in press.

Terano, T., and Suseno, M. (1991). "Fuzzy System Theory and its Aplications". Academic press Inc. San Diego C. A.

Tindell, K., Burns, A. and Wellings, A. (1992). "Allocating Hard Real-Time tasks: An NP-Hard problem made easy". *Real-Time Systems*, 4, 2, pp. 145-165.

Xu, J. (1993). "Multiprocessor scheduling of processes with release times, deadlines, precedence and exclusion relations". *IEEE TSE*, 19,2, pp. 139-154.

Comentario [MC1]: